Integrating the Teaching of Computer Organization and Architecture with Hardware Design Early in Undergraduate Courses

Ney Laert Vilar Calazans, Fernando Gehm Moraes
Faculdade de Informática - Pontifícia Universidade Católica do Rio Grande do Sul (PUC
Porto Alegre - RS – BRAZIL - ZIP Code: 90619-900
E-mail: {calazans, moraes}@inf.pucrs.br

Abstract - This paper describes a new way to teach computer organization and architecture concepts wi hands-on hardware design experience very early in Computer Science curricula. While describing the ap addresses relevant questions about teaching computer organization, computer architecture and hardware design in computer science and related fields. The justification to concomitantly teach two often separately addressed twofold. First, to provide a better insight into the practical aspects of computer organization and architecture allow addressing only highly abstract design levels yet achieving reasonably performing implementations, t integrated teaching approach feasible. The approach exposes students to many of the essential issues incu analysis, simulation, design and effective implementation of processors. Although the former separation of suc disciplines has certainly brought academic benefits in the past, some modern technologies allow capitalizi integration. Indeed, the new approach is enabled by the availability of two new technologies, fast hardware platforms built with reconfigurable, hardware and powerful computer aided design tools for design entry, val implementation. The practical implementation of the teaching approach comprises lecture as well as laborato starting in the 3rd semester of an undergraduate Computer Science curriculum. In four editions of the first two c students have obtained successful processor implementations. In some cases, considerably complex applicati bubble sort and quick sort procedures were programmed in assembly and or machine code and run at the description language simulation level in the designed processors.

Index Terms – Computer Organization Teaching Methods, Undergraduate Curriculum, Hardware Des Hardware Description Languages, Digital System Prototyping.

1 INTRODUCTION

The advent of the electronic computer and its popularization in the last decades as a widespread brought about several changes in well-established academic sectors like Electrical Engineering, Physics and M departments. The creation of Computer Science courses emerging from single or joint efforts of such department change. Presently, besides the stable status acquired by independent Computer Science departments, there is a trend to offer degrees in subjects that are intermediate between Computer Science and many other disciplines. T from Computer Engineering, Computational Mathematics and Computational Physics to multidisciplinary efforts make sense just a few years ago, such as degrees mixing computers and Arts or computers and Law. Even technical degrees, it is necessary to provide the student with a basic set of concepts about the hardware, the softw interface and relationship between them. Thus, the contents of disciplines as computer organization and architecture are a mandatory part of any such degree. Of course, the amount and depth of material adequate to will vary widely, but the basic concepts are always present.

This work addresses the teaching of computer organization and architecture in more technical degree amount of material on hardware and on the hardware-software interaction is significantly large.

Computer architecture and organization are well-established disciplines in Computer Science and Engineering curricula. Quite often, these contents are covered early in undergraduate courses, and the approach i sound. This happens because of the crucial need to provide the student soon with a strong understanding of the between the physical reality of hardware and the software abstractions it implements.

On the other hand, complex digital systems design is a subject usually covered, if at all on elective co end of the undergraduate curriculum. Also, these digital systems design courses frequently restrict themselves aspects of the design process, such as the physical and logical abstraction levels.

However, computers *are* digital systems, formed either by a composition of several subsystems or ev on a single VLSI chip. Thus, an excellent way to teach and to understand their capabilities and limitations is by designing them. Two recent technological advances allow this to become a reality in the classroom. The availability of cheap, powerful VLSI hardware prototyping platforms based on reconfigurable hardware suc Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs) [1]. A good exa profusion of hardware aids is the list maintained by Guccione [2]. The second advance is the availability of powerful, free and/or commercial computer aided design (CAD) tools for high-level design entry, vali implementation. Examples of these tools are the current simulators and synthesizers based on Hardware Languages (HDLs).

This work describes a novel approach to teach computer organization and architecture through th simulation, design and effective construction of processors using the aforementioned advanced facilities.

The rest of this article is organized as follows. The next Section describes the state of the art in teac systems design in general and computer organization and architecture in particular. Follows Section 3, that approach proposed in view of the state of the art. Section 4 is about the context of the implemented course explores the courses structure and the employed teaching methods, considering the merging of digital syst techniques into computer organization and architecture. Section 6 assesses issues arising from the choices problem solving strategies. Section 7 introduces preliminary results on assessing the point of view of student courses. Finally, Sections 8 and 9 present the current state and the future of the reported work, as well as a few c

2 STATE OF THE ART

Computer organization and architecture involves a great deal of abstract concepts for the undergradua Today, most students start undergraduate courses having already had contact with computers as users, which was some two decades ago. In this way, they see the computer in a preconceived way, as a tool useful for tasks such browsing or text edition. This bares few obvious relationships to Boolean algebra, logic gates, registers or arith units. There is then an enormous gap to fulfill in order to teach computer organization and architecture adequate is caused by the increasing number of abstraction layers interposed between the real hardware and end-us applications. To close this gap, several teaching methods are currently being proposed. Most of them are based o use of practical work to support lectures.

Traditional approaches to introduce the study of computer organization and architecture, like the ex proposed by Patterson and Hennessy in [3], often limit practical work to using assembly language progra associated assembler and simulation tools. This Section reviews modern approaches that aggregate effective hard and sometimes hardware implementation to help understanding computer architecture abstractions and to bett students to further work on the field.

A frequently debated issue among those suggesting modern approaches is the use of commercial commercial (also called special-purpose or educational) tools. Several authors advocate the use of educational only way to improve teaching. The idea behind this attitude is that using commercial tools has some serious dis In [4], the authors state that commercial tools are not adequate in teaching, because they are oriented towards a market, because they provide many features not needed for educational purposes and because their leaning curve slowing down the learning of fundamental design concepts. The authors then present a specially produced environment called VISCP, based on schematic capture to help students in grasping digital systems concepts r Maurer [5, 6], on the other hand, proposes another environment, called winFHDL with similar characteristics bu a more abstract design level. He suggests that simulation is more important for the designer of hardware than hardware laboratories, and also proposes the extensive use of educational tools. In another work, Grünbacher four simulators, each one oriented to help in teaching specific concepts in modern computer architectures, like cache memory functionality and superscalar organizations.

An opposite view to enhance computer organization and architecture courses is to directly insert commercial hardware and software tools as basis for practical work. Nixon [8] proposes employing medium Programmable Logic Devices (PLDs) associated tools and proprietary languages for introducing digital syst concepts. The author reviews and compares the most prominent tools and languages products in the area of Pr Array Logic (PAL) design, including PALASM, PLPL, PLACE, ABEL and MAX+PLUS II. Nixon does not rec use of languages as VHDL for introductory levels, due to its complexity. Hamblen and others present two requ taken by computer engineering seniors at Georgia Institute of Technology in [9]. The prerequisite knowled courses includes digital design, FPGAs, VHDL modeling and synthesis, assembly language, C programming systems and computer architecture. The practical work involves the design, implementation and validation processor and associated basic software (assembly and C compiling tools). The design tools include commercial Xilinx FPGAs and retargetable compilers (Lcc) and assemblers (in-house). The implementation employs a rapid platform, ZyCAD Paradigm RP, comprising 16 10,000-gate FPGAs. Validation takes place through the use o and emulation based on commercial logic analyzers. Finally, the authors conclude that feedback from s increasingly encouraged the use of commercial tools, languages, devices and test and measurement equipment. T CAD tools, the VHDL language, FPGAs and logic analyzers. A third approach using mostly commercial tools by Kleinfelder and others in [10]. The authors implemented a set of five courses starting at the fourth quarter of Engineering curriculum at the West Point Military Academy. The courses employ increasingly more complex courses progress, starting with schematics using a student edition version of a commercial tool (Workview Offic by an introduction to PLDs and their hardware design using a simple HDL called CUPL. The introduction of V at the second course of the set, Computer Organization, where it is used for the specification and implementatio microcontroller. The last courses of the series expect the students to design a real system to solve a real proble both digital and analog components.

Other works have recently introduced the need to provide integrated laboratories for enabling hands-o for engineering and computer science students. The main goal of these efforts is providing capacity to solve re that most often require the students to deal with electrical, mechanical and other engineering problems. Exam laboratories are those at the Colorado University at Boulder [11] and at Auburn University [12].

Teaching hardware using extensive hands-on experience relies on the availability of hardware p support these design activities. Several authors have proposed the construction of such platforms, e. g. the platform described in [13] and the FPGA-based hardware tester depicted in [14].

Most of the results obtained in the present work points to the commercial tools use approach as the mo This is corroborated by the opinion of the majority of the students, as discussed in Section 7.

3 JUSTIFICATION OF THE APPROACH

There are two quite different ways of approaching the design of complex digital systems at the un level, the one employed in Electrical and/or Electronic Engineering, and that of Computer Science and/or Engineering.

Electrical/Electronic engineering students start with a strong emphasis in linear circuits, which are u the foundation to investigate electronic phenomena and devices. The study of digital circuits and systems pri techniques comes very often late in the curriculum as a special case behavior. In this way, electrical engineers strongly aware of physical consequences of designing digital circuits, like clock distribution problems constraints. However, they often fail to grasp architectural and software issues of a complex digital systems desig

On the other hand, Computer Science/Computer Engineering students get acquainted with digital sy earlier than Electrical Engineering students get. They employ abstract models from Boolean Algebra as basis circuit theory models. Thus, it is hard for them to deal with timing and power constraints affecting the performa the functionality of the whole system. However, they are usually well equipped to deal with higher levels of abst as architectural and software issues of digital systems.

As mentioned in the beginning of this paper, two technological advances, powerful CAD tools and fa prototyping facilities enable to teach computer organization through the effective implementation of processors. by allowing designers to overlook most physical synthesis issues, shifting the design effort to more abstract approach is particularly well suited to Computer Science students, since they may then use physical synthesis *button* activity, concentrating the design effort at higher level, organizational and architectural problems.

Teaching computer organization and architecture in this way helps Computer Science students to hardware design activities. Also, students are able to understand more thoroughly problems appearing in future c compiler performance optimization, memory management and input-output bottlenecks at the system and opera level.

However, the same set of tools can most often be used with lower level preoccupations in mind, since aids for performing tasks such as IC floorplanning, delay and area constraints, performance-driven placement, clock distribution. This would of course be a more appropriate approach to Electrical Engineering students.

From the above discussion, it is reasonable to conclude that the ideal team to design digital sy particularly computers, must be a composition of both engineers and computer scientists. One pro multidisciplinary teams is that they often fail to communicate properly, due to the different views each profess the same basic concepts. Another beneficial side effect of using the above teaching approaches is to communication gap, by providing common ground knowledge in these concepts to both classes of professionals is better understood by presenting a few examples of practical subjects where communication problems aris examples be external/internal memories and pipelining. The authors consider in the analysis only what comp students acquire to bridge the mentioned communication gap. Several other example subjects and the electrical

student side of bridging the gap could be devised as well.

A computer science student often learns to view memories simply, as tables coupled with means to data. The approach in this work proposes a simultaneous digital systems view of memory technology. Initially is study of memories in an abstract way, as tables. Follows an explanation of the internal structure of each memor decoding in hardware, how this decoding implements access to individual memory positions, and how the nee signals are generated. This view of memory technology is helpful in understanding the need to address t bandwidth problem not only in hardware design, but also in software and, above all, operating system design. computer scientists are able to better interact with hardware designers in issues like the SRAM/DRAM trade-off.

Pipelining is another point where communication can be a problem between hardware and softwar Software people usually deal with an abstract view of pipelines, as provided by the assembly language l resembles very little the multistage organization of modern hardware pipelines. The exercise of implementin hardware enables the computer science student to better understand the hardware issues as well as the implemen assembly language abstraction. This contributes to enhance his/her capacity to communicate with hardware speci

There is also the recent trend to provide curricula that are midway between Electrical Engineering an Science such as Computer Engineering. The students of these latter courses should benefit of the approach above, applied respectively in early and late moments of their undergraduate academic experience. The first bas as well as on Computer Science curricula, is to modify the traditional early courses on computer organ architecture to include training with the digital systems implementation of the structures needed to support the a of computers in these disciplines. The second idea is to employ elective courses at the end of the curricula to training of the interested student in advanced topics of digital systems and computer abstract design. This currently being applied in the Computer Science curriculum at the PUCRS by the authors.

As a conclusion of this Section, it is possible to state that the coupling of modern CAD tools, fast platforms and carefully designed teaching methods is the key to justifying the mix of computer organization/arch (VLSI) digital systems design very early in Computer Science/Computer Engineering curricula.

4 COURSES CONTEXT

The proposal for computer organization teaching is part of a revised Computer Science undergraduate curriculum, which went into effect in the 1st semester of 1997 (starting in March) at the authors' institu implemented as two required courses, a 4-hour a week course on computer organization and a 2-hour a week course, both taught in the 3rd semester. The sequence of courses more closely related to hardware in the cu depicted in Figure 1.

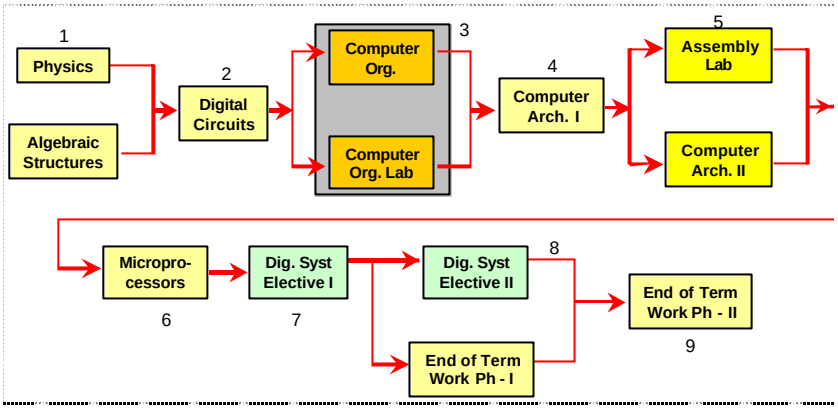


Figure 1 - Courses Context. The numbers indicate the semester within the curricula where the courses ta The arrows indicate prerequisites among courses.

The prerequisites for both courses on computer organization are an introduction to digital circuits, Algebraic Structures and another on Physics for Computer Science. These courses provide the student with combinational and sequential logic design techniques, lattice and Boolean algebra theory, and a brief account o electronics theory and instrumentation, respectively. In the third semester, relying upon this background, the st the principles of computer organization in theory and in practice. They start working with a simple von architecture at the logic and RTL (Register Transfer Level) levels of abstraction. In the second part of the course to the same subject employing higher abstraction levels of design, based on the use of HDLs (Hardware Languages). The detailed contents of these two courses are explored in Section 5.

Four other required courses, two on computer architecture, one in assembly language programming, microprocessors follows those on computer organization. Computer Architecture I explores computer a performance analysis, pipelining, basic software tools like assemblers, linkers and loaders, and the role of c translating high-level language programs to assembly language. Computer Architecture II includes input/output memory hierarchy and memory management, and principles of parallel computing. At the same time, in the fif the students are required to take a Lab course where they implement a practical application (like cryptography using assembly programming. Next, in the Microprocessors course they undertake comparative studies of microprocessors and microcontrollers.

This curriculum requires each student to take at least 16 credits on elective courses, corresponding t courses. These courses are taken on eight knowledge domains in Computer Science, namely digital systems software engineering, applied computing, artificial intelligence, computer graphics, distributed systems and comp The students can choose to take one or two of these elective courses on digital systems advanced topics. T emphasis of the elective courses are on complex digital systems design using HDLs, designing, implementing an some application on hardware prototyping platforms based on FPGAs.

The curriculum also requires students to produce a one-year End-of-Term Work. This work is cond individually or by groups of two students, under the advising of one academic. Students can of course choose s related to digital systems. In this way, every student is exposed at each period in the whole curriculum to hard regarding his future profession. This was one of the objectives of the 1997 revision.

5 COURSES STRUCTURE AND TEACHING METHOD

5.1 COURSES STRUCTURE

The Computer Organization (INF46183) course contents are distributed into four units. Unit 1 is *digital systems design process*. In this Unit, models for the design process like the Y-diagram [15] and oth introduced, together with a discussion of digital systems' classifications and the basic digital circuit design flow of CAD tools. Unit 2, *The CPU dassical model* is where the datapath and control unit partitioning is explored presentation of both von Neumann and Harvard organization models. During this part of the course, a step by ste design is developed at the RTL abstraction level. Unit 3, *Hardware description languages (HDLs)* presents an abstract way of designing digital systems and, in particular, processors. This includes the redesign of the CPU c Unit 2. Unit 4, *Advanced topics on computer organization*, introduces advanced hardware structures for datapath unit performance enhancement, such as pipelining and floating point arithmetic hardware. At the end of the le students are required to design a load-store 16-bit processor using the target HDL. The specification of this fi delivered to students during the first HDL introductory lectures. In this way, they have 45-50 days to complet and present a functional HDL simulation.

The theoretical course comprises approximately 30 2-hour lectures. One quarter of the lectures is c revising digital circuits' basics and in presenting design process models and taxonomy. One third of the lecture present the Cleopatra case study, developing and discussing its schematic-based design. Another quarter of th used to introduce the basics of a specific HDL and to redesign the case study with it. The rest of the time is take the advanced concepts in computer organization.

The Computer Organization Laboratory (INF46184) course is divided into three units. In Unit 1, entitl *digital circuit design*, students review basic concepts acquired in the previous Digital Circuits course, w schematic capture and simulation tools. They implement basic combinational and sequential blocks such as ad counters and finite state machines. Next, in Unit 2, *Classical CPU design*, they employ the same set of tools, t the case study of the lecture course step by step, using RTL schematic modules. Finally, Unit 3, *HDL hardwa* where an HDL language is introduced and used to describe hardware at the behavioral and structural levels, co complexity of the circuits they can handle with that of the previous approach. The target HDL chosen is VHDL [

The lab course comprises around 15 2-hour practical sessions. Table 1 shows a subset of current classes prepared for the students.

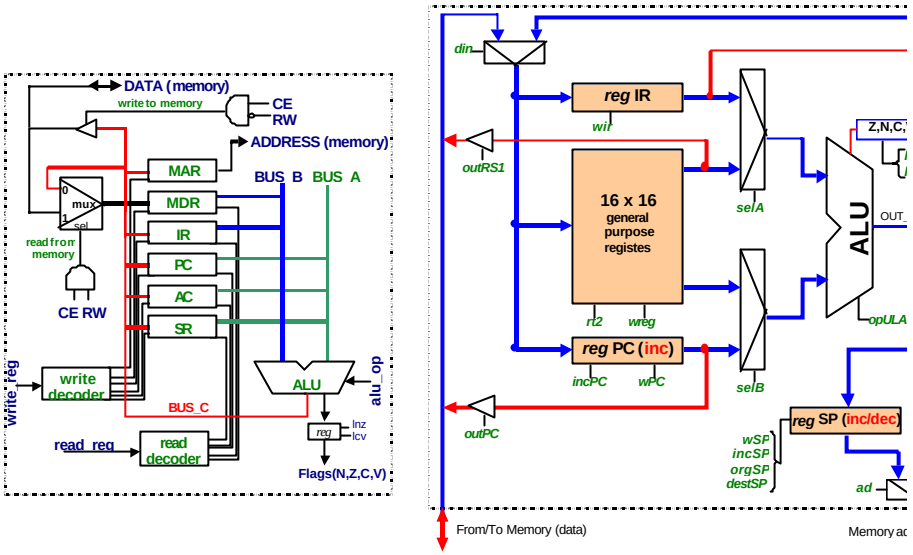
Class	Title	Goal
1	Design of an 8-bit adder (2 hours)	Schematic capture. Schematic capture editor training, use of desi hierarchy and principles of functional simulation.
2	Design of a 4-bit ALU (2 hours)	Schematic capture. Implementation and functional simulation of a combinational circuit which is widely used in microprocessors.
3	Implementation of an 8-bit accumulator in an FPGA prototyping platform (4 hours)	Schematic capture. Introduces the students to the use of physical tools and to the use of a prototyping platform (XS40/Xstend).
4	Design of an up/down counter (2 hours)	Schematic capture. Practice with state machines.
5	Design of a *timer* (4 hours)	Schematic capture. Students must implement a *timer*, with minut seconds. They must specify the counter structure, implement the test it in the prototype platform.
6	Simulation of a stack calculator (2 hours)	Schematic capture. Introduce the students to the concepts of data control circuits, and also the simulation of a programmable circui
7	Cleopatra assembly language (2 hours)	Use of a *simulator* for the example architecture (Cleopatra), aimi practice of assembly language programming and particularly the understanding of the architecture addressing modes.
8	Cleopatra data-path (2 hours)	Schematic capture. The students should finish the design of the g path block, using micro-commands to excite it, by means of func simulation.
9	Cleopatra control block (2 hours)	Schematic capture. The students implement the block responsible translating assembly instructions into micro-commands addresses micro-ROM.
10	Use of the VHDL simulator (2 hours)	VHDL. Design of a simple VHDL example, simulating it with a t This class aims introducing the use of the VHDL simulator.
11	Design and validation of an 8-bit multiplier (2 hours)	VHDL. Practice with several sub-circuits (VHDL entity-architec training) in the same design.
12	Implementation of the 8-bit multiplier (2 hours)	VHDL. The students must take into account the input/outputs cor imposed by the prototype board to define the multiplier external i The goal of this class is to implement and run the multiplier in th prototyping platform.
13	Behavioral bubble-sort in VHDL (2 hours)	VHDL. The goal of this class is to use VHDL as a programming l implementing a sorting algorithm.
14	Structural VHDL (2 hours)	VHDL. Implementation of a second version of the bubble sort alg comparing the physical results between the behavioral and struct versions.
15	Simulation of the VHDL Cleopatra version (4 hours)	VHDL. Introduce to the students to the design of complex test-be
16	Drink Machine (2 hours)	VHDL. Implements the classical example of the drink machine (exploring the facilities to implement state-machines in VHDL.
17	State machine to control RAM memory access (4 hours)	VHDL. Shows how an FPGA can control an external memory. U machines.

Table 1 – Example set of available classes for Computer Organization Laboratory. Among the 17 classes, on the available time, 9 to 10 are chosen each semester (30 hours - 15 classes), dividing contents evenly schematic capture and VHDL lab classes.

A subset of 9 to 10 of the practical sessions is picked at each semester for teaching as lab course c these, 60% are dedicated to get the students acquainted with CAD tools and to explore the schematic based processor implementation. Of the remaining sessions, 30% investigate the HDL-based approach and 10% introd synthesis concepts and train the use of the available prototyping platform [18]. This platform is constructed with FPGAs.

5.2 TEACHING METHOD

Both the lecture and lab courses are strongly based on the stepwise analysis of a single processor named Cleopatra [19]. This is in accordance with well-established practice of teaching computer organizati exploration of the MIPS [20] architecture in [3]. The datapath for the Cleopatra processor is depicted in Figure the emphasis is on computer organization models, the case study is neither complex nor modern. It consists in a s accumulator-based von Neumann core, with 13 operation codes and 4 addressing modes, amounting to instructions.



(a) Cleopatra datapath schematics. The registers are *Memory Address Register* (MAR), *Memory Data Register* (MDR), *Instruction Register* (IR), *Program Counter* (PC), *Accumulator* (AC), and *Subroutine Register* (SR). There are 7 control signals that compose the 13-bit microinstruction word generated by the Control Unit of the processor (not shown).

(b) Ramses-V datapath schematics. The regi *Instruction Register* (IR), *Program Counter* (I *Pointer* (SP), and a register file with 16 genera elements. There are 18 control signals that cor microinstruction word generated by the Contro the processor (not shown).

Figure 2 – Cleopatra and Ramses RT level datapathes explored in the INF46183 course.

It is worth discussing the validity of choosing architectures like Cleopatra to teach the basics of mode organization. One first reason is that this is the first contact of the students with the abstract concept of Ins Architectures (ISAs), and it is necessary to build on the knowledge they acquired in the Digital Circuits c Furthermore, this approach gives the students a historical perspective of processor evolution. The first pro microprocessor generations were accumulator-based machines, due to the scarcity of integration availab consequent high cost of memory bits inside the CPU. Starting with this primitive organization, it is easy to expl students the bottlenecks that led to the CISC processor crisis. Among these, it is possible to mention the high

clock cycles per instruction (CPI) and the frequent access to a unified memory (the von Neumann or *stored progr* Understanding these problems paves the way to accepting innovations like pipelining (to reduce the CPI), the Ha of organization, and increasing the number of internal registers (reducing the so-called von Neumann bottleneck) of the course, students are faced with the need to implement a load-store architecture, and in the next course Architecture I) they are expected to implement pipelined processor organizations.

The Cleopatra core assists in introducing and discussing several new concepts in the lecture c specification allows exploring instruction sets, addressing modes and principles of assembly language programm architectural concepts are apprehended in practice using an in-house basic software development and validat described in Section 6. After familiarization with the architectural concepts and assembly language programm the proposition and the study of one datapath organization for implementing the Cleopatra processor. Among t explored, it is worth highlighting the most important ones. First, the role and use of control and data registers Next, the interaction between the memory and the processor is studied, differentiating von Neumann ar organizations. Follows the choices incurred in the ALU implementation and the busing strategies, finishing with unit structure, comparing hardwired and microprogrammed implementations.

Meanwhile, the lab course examines the digital systems design view of the computer organization in d CAD tools, among other tasks, to construct increasingly more complex modules of Cleopatra.

Both lecture and lab courses start with a traditional, schematic-based approach of processor core desig whole design work is redone with modern, HDL-based tools. There are three main reasons for doing this. First, s it is possible to design at very high abstraction levels and still obtain competitive implementations, due to the cu of current CAD tools. Second, providing students with a comprehensive insight into the panoply of hardware des and tools, by comparing these two significantly distinct approaches. Third, making it clear to students that HD programming languages, which is achieved by the mapping of schematic symbols and structures to the sy semantic structures of the chosen HDL. This last reason take instructors to insist with students that they n hardware" when using HDLs, even if the language they use allows them to view hardware as software.

Immediately after the study of the HDL starts in the lecture course, the end of course work spec handled to the students. This specification includes the instruction set architecture of a load-store processor, ar diagram of its datapath organization. One example datapath that is provided in the last edition of the course appea 2(b). The most important design constraint imposed by the specification is a fixed number of clock cycles per (CPI=2).

The last part of the lecture course (Unit 4) is dedicated to show how concepts presented in the first thr be extended to reach the real structure of modern processors. In this sense, this Unit is much more a preview of w to be explored in next courses than a deep or complete image of the subjects. The chosen topics are pipelining ar implementations of floating point arithmetic functional units. Pipelining is showed as a key concept for achie performance, improving the instruction throughput, without changing the instruction latency. The IEEE-754 st

serves as the basis to explore the implementation of floating-point hardware. After presenting the standard, an view of hardware multiplication and addition is explored. No real hardware implementations of these are invest due to the lack of time.

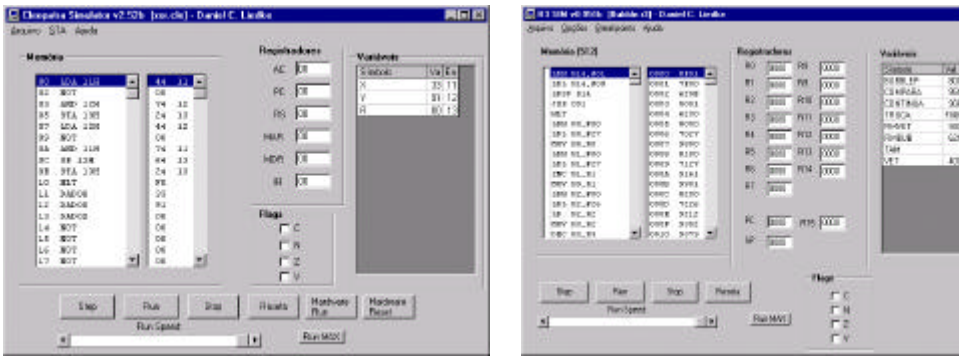
At the end of the semester, students have obtained an understanding of the basic processor design f levels of abstraction. They write assembly language programs that run on the machine they have impleme students are able to develop software and follow the data and control information flow internal to the hardware.

6 TOOLS

As mentioned in Section 2, the authors strongly advocate the use of commercial tools whenever p didactic reasons, the processors used to present the basic computer organization and architecture concep commercial ones. It would not be feasible for such an early course to do otherwise. Thus, it was necessary to deve in-house tools to support the introduction of the concepts. The next Section explores these tools, while Section 6 the employed commercial tools.

6.1 IN-HOUSE TOOLS

There are two sets of in-house built development tools, one for the Cleopatra case study and another fo final work. Both sets comprise an integrated environment with a text editor, an assembler and a simula corresponding processor. Figure 3 shows example windows for both environments.



(a) Cleopatra simulator (b) Ramses-III simulator

Figure 3 – Cleopatra and Ramses assembly language simulators developed for the INF46183/INF46184 co tools include a built-in text editor, an assembler, and a simulator whose graphical user interface (GUI) is s GUI allows controlling the execution speed, inspect and change values of registers, flags and memory. T window exhibits the symbol table generated by the assembler, and can also be edited during the simu

These environments are an invaluable support for the teaching of the ISA basic concepts, including th

control and data registers, addressing modes, assembly language and its relationship to object code, not to menti understanding of memory organization, variables, variable addresses (pointers) and variable contents.

Also employed is the SIS public domain academic synthesis system from Berkeley [22], for illustra unit logic synthesis strategies.

6.2 COMMERCIAL TOOLS

There are four sets of commercial tools employed in the Computer Organization lecture and lab course tools run over a Windows NT PC host computer. The most important of these is the CAD system Foundation, from Xilinx [23]. From this CAD system the most employed tools are schematic capture and module gener simulation, and logic and physical synthesis tools. It is also used the Active-HDL, version 3.5 from Aldec [2 simulation environment, which can be fully integrated with the Foundation software.

The next set of tools is a hardware prototyping platform composed by two boards, the XS40 a produced by the enterprise Xess, Inc [18]. This is an educational platform useful for small hardware impl (supporting up to 5,000 equivalent logic gates). Attaching the boards to a PC parallel port allows the communicat host computer to take place. The last set of tools is the support software for design download and communicat the XS40/XStend platform and the host computer.

As stated before, the first Units of the Computer Organization courses rely upon the use of sche capture and validation tools, including the real implementation of hardware over the prototyping platform. For th the authors have chosen to use VHDL as the Hardware Description Language in both courses. Again, this is nc way to approach higher level abstract design entry, but they consider it worth, since VHDL is not only one complete, but also the most used of HDLs today. The same tools mentioned above also support other HDLs, suc and Abel.

The Active-HDL tool is used to guarantee functional validation of the VHDL code. One example o using this tool appears in Figure 4, which displays a partial simulation of a program executing in the Cleopatr Figure 4 stresses the functional nature of the simulation, where no relative delay among signals is considered.

As a conclusion of this Section, it is important to note that students are exposed in the courses to a c integrated framework for digital design. Students get used to several abstraction levels, starting with functional sp (either in schematic and/or VHDL) and going down until the physical synthesis, implementation and testing of re prototypes.

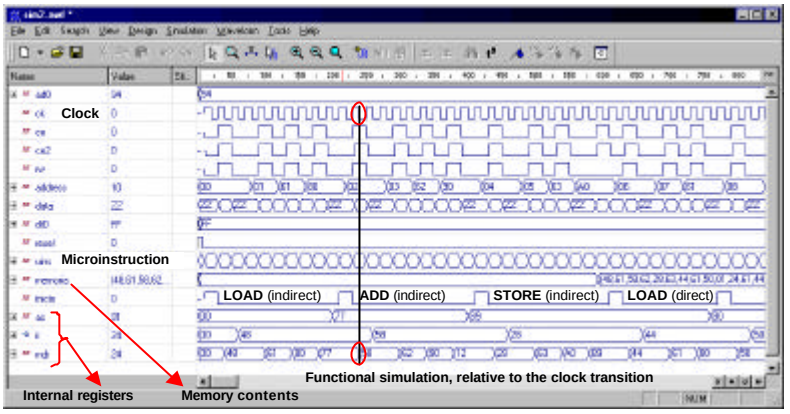


Figure 4 - Partial functional simulation of an object code program in the Cleopatra processor. The simula the clock by clock execution of a four-instruction sequence, LDA, ADD, STA, LDA. Signal inicio marks the cycle of each instruction. The relevant register contents are shown in the last three signals of the simulatio (AC, IR, and MDR).

7 RESULTS

The authors have elaborated a survey to measure the students' view of several points concerning th Organization courses INF46183 and INF46184. This Section presents the structure of the survey, the raw data ol applying the survey to more than a hundred students, and a critical analysis of some aspects of the data.

7.1 POPULATION AND SURVEY DESCRIPTION

The survey contains 16 questions, divided into 5 subjects. A first set of questions (Q1-Q3) concerns data of the population attending the courses. Information collected includes the semester when they took the cour first time (Q1), the number of times the students took the course until succeeding (Q2) and the fact of taking the lab courses together or not (Q3). The first question is the main key to the classification of the data, chosen t evolution of the approach along the time. Table 2 shows the number of students per semester that answered the su

Semester	98/I	98/II	99/I	99/II	Total
No. of students	26	29	37	20	112

Table 2 – Number of students that answered the INF46183/INF46184 survey.

A second set of questions is dedicated to evaluate the overall quality of the courses both in ther compared to other courses in the Computer Science curriculum. The specific questions concern:

- Q4. Adequacy of the courses to the serialization within the curriculum;

- Q5. Overall importance of the courses for the student formation;
- Q6. Overall quality of the contents presented in the courses;
- Q7. Classification among all courses in the curriculum;
- Q8. Degree of motivation to pursue hardware studies/work.

The third set of questions asks the students to evaluate specific contents of the lecture course and conc

- Q9. The use of the hypothetical architecture Cleopatra to approach the teaching of computer architecture c
- Q10. The use of VHDL to describe/validate/implement hardware;
- Q11. The validity of studying pipelining and floating-point arithmetic.

The fourth set of questions asks the students to evaluate the lab course. The specific questions concern

- Q12. The available resources for performing practical works;
- Q13. The amount of help the lab course represents in apprehending the lecture course concepts;
- Q14. The complexity of the employed CAD tools (most of them commercial ones).

The last two questions are related to the complexity (Q15) and validity (Q16) of the lecture courses fin

7.2 SURVEY DATA PRESENTATION AND DISCUSSION

The evaluation of the survey led the authors to choose a 0-5 range for the answers of the students. questions Q1 to Q3, a score of 5 represents the best evaluation for the course aspect and 0 represents the wo evaluation. The global score 2.5 for any question is the average value. Each particular aspect can be above or bel authors (and the students) expect from the courses based on this average value.

Table 3 presents the average scores for each question and for each semester, together with the avera questions Q4-Q16 for each semester. There is also a global overall score. Next, follows a set of conclusions re teaching approach proposed here, which can be extracted from the raw data. Some of the conclusions are rath Others corroborate the views of the authors while suggesting the approach. Still others indicate major difficult the students, pointing to directions to review and adapt the current method.

One of the most important conclusions comes from Q4, where the students agree, with a high degree (Global Average 3.77), that the courses are located where they should in the curriculum, occurring neither too e late. This is indeed one of the least obvious conclusions, and reinforces the position of the authors to expose stu earlier to hardware design than traditional methods. Also, most students consider the organization courses of h importance (Q5, Global Average 3.29).

Regarding the contents of the courses, the students evaluation was quite positive (Q9-Q11, all Average above 3.70), specially concerning the use of a hypothetical architecture to teach the basic computer

concepts. Also it is a consensus among students that using a standard language such as VHDL is a good cho hardware implementation at high levels of abstraction. This result indicates that there is no need to recur to languages such as CUPL, as suggested in [10].

	Question	98/1	98/2	99/1	99/2	Average
Student profile	Q2	1.35	1.45	1.36	1.50	1.41
	Q3	4.42	4.83	4.32	4.75	4.58
Overall quality	Q4	3.77	3.82	3.72	3.75	3.77
	Q5	2.85	3.28	3.27	3.75	3.29
	Q6	3.46	3.21	3.38	4.21	3.56
	Q7	2.54	2.32	2.50	3.65	2.75
	Q8	1.88	1.59	2.08	3.20	2.19
Contents	Q9	3.73	3.62	3.70	4.75	3.95
	Q10	3.85	2.86	3.68	4.53	3.73
	Q11	3.62	3.52	3.92	4.00	3.76
Lab course	Q12	2.35	2.97	3.58	3.30	3.05
	Q13	3.24	2.93	3.92	4.00	3.52
	Q14	3.15	3.00	2.97	3.70	3.21
Final work	Q15	2.27	2.38	2.03	1.80	2.12
	Q16	3.08	2.93	2.92	4.05	3.25
Average from questions Q4 to Q16		3.06	2.96	3.21	3.75	3.24

Table 3 – Average scores for the INF46183/INF46184 survey questions, classified by the first question, semester where the students took the courses for the first time, and excluding consideration of Q1 to Q3

The lab course was also seen as being of fundamental importance to complement the lecture cou apprehension (Q13, Global Average 3.52). Students of the first two editions of the lab course did not evalu resources very well. This is easily explained by the lack of available prototyping boards in these editions. T increased significantly as the prototyping platforms started to be used (Q12). Another point of the approach pr which is certainly sensitive to criticism is the use of commercial CAD tools. Question Q14 reassures that the cho seen as a good one by the students (Global Average 3.21). There is no need to underestimate the student's capa with such complex tools.

The final work is seen as extremely demanding (Q15, Global Average 2.12), and this situation points revise its basic requirements. More significant though, is the students' evaluation concerning the importance work, which is very good (Q16, Global Average 3,25). It is important to notice that the three first averages were and the last one was significantly higher (4.05). The explanation is in the use of the final work as the starting p practical work in the following course, Computer Architecture I, practice started only in the last period. This c motivated students to evaluate very well what they had learned in the previous courses.

8 CURRENT STATE AND EVOLUTION

In the first two editions of the Computer Organization courses (98/1 and 98/2) only design entry an simulation tools have been employed, due to the lack of prototyping platforms, obtained only in 1999/1. The n adapting the course contents to the new reality, inserting lab experiments to address physical synthesis, a performance results of this step, and downloading and testing of designs on the prototyping platform. This reformulation of lecture and lab contents. As a result, a better balance between abstract design issues and effe systems implementation was achieved in the available time. Table 1 reflects part of the structure of the lab course

Another consequence of the availability of effective hardware design resources is the extension of th proposed here to the Computer Architecture (I and II) and Microprocessor courses in the same curriculum objective is to provide students with a continued exposition to hardware design issues during their academic life topics that are to be covered in such extensions are, for example, pipelining and cache hardware implementa Currently, the Computer Architecture I course has already been adapted. There, the students design a simple enhancement for the version of the Ramses architecture they worked in Computer Organization.

Connecting the subjects covered here to the elective courses on digital systems at the end of the cu another ongoing work. In the future, the authors intend to explore other advanced aspects of digital systems. One the development of CAD algorithms. With the hardware and software support tools available, there is one subj recent and important that can be examined in elective courses, the hardware software partitioning and codesign [2 not least, the authors are considering the application of digital systems design technology to other, apparent disciplines in Computer Science, such as compilers, computer networks and computer graphics.

9 CONCLUSIONS

The main goals set while developing the work proposed in this paper are considered attained. It is the authors that abstract digital systems design and computer organization subjects have been successfully integrated teaching approach. The first results of this work were reported in [26].

After teaching the courses four times, the authors have achieved a considerable level of satisfaction on the students as discussed in detail in Section 7. Several successful functional processor implementations have be and some of these present outstanding quality. In some cases, considerably complex applications such as bubb quick sort procedures were programmed and run in the designed processors by the students alone.

Some questions can be posed and answered in the context of the approach proposed here. First, it is co ask *"How early can Computer Science students start designing digital systems design?"* The authors' demonstrates that students may start at the 3rd semester of an undergraduate curriculum, and yet achieve a hig success. Second, *"How far can Computer Science students go in doing digital systems hardware design?"* So fa been possible to answer this question fully, since the first students are right now taking the first of the elective digital systems. The authors are very optimistic towards this question, based on the experiments cond

undergraduate and graduate students at the research level.

A problem that only now starts to be treated by the authors is the quantitative assessment of student p and effectiveness of the approach, and this should receive increasing attention in the next editions of the courses.

The positive aspects of the approach proposed here (as obtained from the survey data in Section teaching of computer organization together with digital systems design at early stages of undergraduate courses standard HDLs like VHDL; (*iii*) use of hardware prototyping platforms in lab courses; and (*iv*) final work in design and validation of a computer organization/architecture. On the other hand, the negative aspects are seen failure rate in the lecture course; (*i*) overall complexity of the final work; and (*ii*) difficulty in motivating proceed studying hardware subjects.

Presently, several measures address the solution to the negative aspects identified above, although i early to evaluate the efficacy of these measures. First, there was a migration of subjects from the Unit on advanc the Computer Organization course to the Computer Architecture I course, reducing the overall contents of the fi the final work implementation of the Computer Organization course is presently used in the Computer Architectu enhancing the integration and continuity of contents in the two courses. The amount of practical work in the cl been increased, to enhance students' motivation. Finally, the final work complexity has been reduced by the e instructions and structures that are seldom used in the Ramses ISA. The 2000/I and 2000/II results have alrea small decrease in the failure rate.

In a near future other measures are taking place. The most important of these is in the Digital Circuits Figure 1). This course is presently under the responsibility of the Electrical Engineering Department, which course in a traditional way, just like it does for engineer students. This course will pass to the responsibility of th Science department, where the ideas presented in Section 3 of this work can be applied fully. This is perceived main source of students' lack of motivation for the subsequent hardware courses.

Most of the material employed in the implemented courses, like case study specifications and des presentations, course notes and free software are available at the following URLs (for the moment, most of this m Portuguese): http://www.inf.pucrs.br/~moraes/Dorglab.html, http://www.inf.pucrs.br/~calazans/org_comp.htm that is not freely available to students can be obtained by contacting the authors directly by e-mail.

10 ACKNOWLEDGEMENTS

The authors would like to acknowledge the continued support of Xilinx, Inc., through its Universit Aldec, Inc. has provided the authors with two years of free licenses of its VHDL simulator. The authors are also CNPq (Research grants 522939/96-1 and 520091/96-5) and FAPERGS (Research grants 94/01340-3 and 96/5036 have been funding their research work. They are in great debt with Daniel Carvalho Liedke, a former s implemented the software tools. The authors are also grateful to all students that answered the survey.

11 REFERENCES

[1] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic. “Field-Programmable Gate Arrays”. Kluwe Publishers, 206p. 1992.

[2] S. Guccione. “List of FPGA-based Computing Machines”. Available at: http://www.io.com/~guccione/HV

[3] D. A. Patterson and J. L. Hennessy. “Computer organization and design: the hardware/software interfac Kaufmann Publishers, Inc. San Mateo, CA, 964p. 2nd Edition, 1998.

[4] L. Rodríguez Pardo, M. Moure, M. Valdés, and E. Mandado. “VISCP: a Virtual Instrumentation and C Electronic Engineering Learning,” in: *1998 Frontiers in Education Conference*, Tempe, AR, pp. November 1998. Available at : http://fairway.ecn.purdue.edu/~fie/.

[5] P. M. Maurer. “Electrical Design Automation: an essential part of computer engineer’s education,” in: *19 in Education Conference*, Tempe, AR, November 1998. Available at: http://fairway.ecn.purdue.edu/~fie/.

[6] P. M. Maurer. “Enhancing the Hardware Design Experience for Computer Engineers,” in: *1998 F Education Conference*, Tempe, AR, , pp. 60-63, November 1998. Available at: http://fairway.ecn.purdue.e

[7] H. Grünbacher. “Teaching Computer Architecture/Organization using Simulators,” in: *1998 Frontiers i Conference*, Tempe, AR, pp. 1107-1112, November 1998. Available at: http://fairway.ecn.purdue.edu/~fie

[8] M. S. Nixon. “On a Programmable Approach to Introducing Digital Design,” IEEE Transactions on Edu 40, no. 03, pp.195-206, August 1997.

[9] J. O. Hamblen, H. L. Owen, S. Yalamanchili, and B. Dao. “An Undergraduate Computer Engineering Ra Prototyping Design Laboratory,” IEEE Transactions on Education, vol. 42, no. 01, pp.08-14, February 19

[10] W. Kleinfelder, D. Gray, and G. Dudevoir. “A Hierarchical Approach to Digital Design Using Com Design and Hardware Description Languages,” in: *1999 Frontiers in Education Conference*, San Jose, P 18-13c6-22, November 1999. Available at: http://fairway.ecn.purdue.edu/~fie/.

[11] J. P. Avery, J. L. Chang, M. J. Piket-May, J. F. Sullivan, L. E. Carlson, and S. C. Davis. “The Integrat and Learning Lab,” in: *1998 Frontiers in Education Conference*, Tempe, AR, November 1998. A http://fairway.ecn.purdue.edu/~fie/.

[12] J. Y. Hung, and S. M. Wentworth. “An Integrated Approach for Electrical Engineering Laboratories *Frontiers in Education Conference*, Tempe, AR, November 1998. Available at: http://fairway.ecn.purdue.

[13] A. Zemva, A. Trost, and B. Zajc. “A Rapid Prototyping Environment for Teaching Digital Logic De Transactions on Education, vol. 41, no. 4, p.342 (CD ROM supplement, folder 2), November 1998.

[14] N. R. McKenzie, C. Ebeling, L. McMurchie, and G. Borriello. “Experiences with the MacTester in Comp and Engineering Education,” IEEE Transactions on Education, vol. 40, no. 01, pp.12-21, February 1997.

[15] D. Gajski and R. Kuhn. “New VLSI tools,” *IEEE Computer*, vol. 16, no. 12, pp.11-14, December 1983.

[16] N. L. V. Calazans. “Automated Logic Design of Sequential Digital Systems, Chapter 1: Introduction,” p de Janeiro: Imprinta Gráfica e Editora Ltda, 1998. (in Portuguese). Available at ftp://ftp.inf.p calazans/pubs/prjlog/v1.0/Cap1.ps

[17] S. Mazor and P. Langstraat. “A guide to VHDL”. Kluwer Academic Publishers. Norwell, MA, 1992.

[18] Xess Corporation. FPGA products: http://www.xess.com/FPGA/.

[19] F. G. Moraes, N. L. V. Calazans, F. Silva, and M. Barrios. “Cleo-LIRMM: an experiment of dedicate implementation on embedded systems prototyping platforms”. In: *V IBERCHIP Workshop*, (Lima-Peru 81-90. (in Portuguese).

[20] G. Kane, and J. Heinrich. “MIPS Risc Architecture”. Prentice Hall, Englewood Cliffs, NJ, 1992.

[21] IEEE Computer Society. “IEEE Standard for Binary Floating-Point Arithmetic”. IEEE Std 754-1985, August 1985.

[22] Sentovich,E. et al. “SIS: a system for sequential circuit synthesis,” Technical report UCB/ERL M92/41. U California, Berkeley, CA, May 1992. Available at http://www.cad.eecs.berkeley.edu/Respep/R usrDoc.html.

[23] Xilinx Inc. Homepage: http://www.xilinx.com/.

[24] Aldec Inc. Homepage: http://www.aldec.com/.

[25] G. De Micheli and M. Sami. “Hardware Software Co-Design”. Kluwer Academic Publishers, 467p. Janua

[26] N. L. V. Calazans, F. G. Moraes. “VLSI Hardware Design by Computer Science Students: How early ca How far can they go?” in: *1999 Frontiers in Education Conference*, San Jose, PR, pp. 13c6-12 to 13c6-17 1999. Available at: http://fairway.ecn.purdue.edu/~fie/.