Adopting New Learning Strategies for Computer Architecture in Higher Education Case Study: Building the S3 Microprocessor in 24 Hours

Jean-Luc Dekeyser, Ahmad Shadi Aljendi

Adopting New Learning Strategies for Computer Architecture in Higher Education

Case Study: Building the S3 Microprocessor in 24 Hours

Jean-Luc DEKEYSER, A. Shadi ALJENDI
CRIStAL, UMR 9189
IRCICA, USR 3380
59650 Villeneuve d'Ascq, France
(jean-luc.dekeyser/ahmad.shadi.aljendi)@inria.fr

Abstract—**Teaching computer architecture is a difficult task, especially to inexperienced students who prefer developing their java programing skills rather than understanding how machine instructions are executed effectively on a silicon black box. Because manipulating electronic components in a classroom is highly complex and complicated, teaching activities used to be aligned to simulation and exercises on paper. During the last 2 years we have developed a set of practical experimentations on FPGA boards. From scratch the students, in twelve 2 hour lessons, build their own processors and execute small programs on them with their own assembler and assembly language. A step by step approach permits the students to understand by design the different concepts of processors that are introduced during the lectures. All the labs are based on intuitive realization of elementary bricks and then combining them like a game of Legos. The results after two years are more than positive and some students have chosen to pursue a Masters of Computer Science with a particular interest in architecture design. The S3 processor can be considered as PBL (Project Based Learning) that covers all revised Bloom's taxonomy levels of knowledge acquisition.**

Keywords—computer architecture teaching; higher education; FPGA; PBL; revised Bloom's taxonomy

I. INTRODUCTION

Since the 1990s, a lot of computer architecture curricula in higher education are dependent on simulations [1, 2]. Simulations are often used to tackle the complexity of hardware systems and the difficulty in covering the high variety of components [3]. Cifredo-Chacón et al. summarized many attempts of teaching computer architecture practically in [4]. Many of these attempts achieved limited success because of, among other things, the inability to fulfill requirements and prerequisites.

In [5], Bloom's taxonomy was applied on three different courses, one of which is computer architecture. The paper was limited to the original Bloom's taxonomy, not the revised one [6]. The conclusion of the paper stated that the use of Bloom's taxonomy in teaching these courses was at least as good as using traditional approaches.

A taxonomy derived from Bloom's original, revised Bloom and problem solving taxonomies was used in [7]. This is a 3-levels approach, covering exercises, problems and projects.

Our approach of teaching computer architecture using the S3 microprocessor allows a comprehensive coverage of learning outcomes required in highly standardized core computer science curricula such as those mentioned in the ACM/IEEE 2013 curricula recommendations [8]. This includes: Digital Logic and Digital Systems, Machine Level Representation of Data, Assembly Level Machine Organization, Memory System Organization and Architecture, Interfacing and Communication, Functional Organization and Performance Enhancements, the last two being electives in [8]. It also covers all the six classes of the learning thinking model of the revised bloom's taxonomy. Furthermore, it is a Project Based Learning (PBL) approach that permits the learners to develop a real life product over a long period of learning.

The remainder of this paper is as follows: in the next section we will cover all aspects of our computer architecture teaching using the S3 processor; this includes educational as well as technical aspects. In section III, the effectiveness of our approach is discussed before the conclusion and the future directions in section IV.

II. THE S3 PROCESSOR

A. Educational aspects

In the PBL paradigm, learning is organized around long term complex tasks, with an essential autonomous role of the learner within small groups, which result in a realistic product, without direction from the facilitators [9]. PBL is used in [10] to enhance computer architecture learning. However, in this attempt, no real processor was built by the learners.

The authors have high level expertise in teaching computer architecture in different academic environments; from very tightly resource constrained environments to very wealthy and developed ones. One of the main features of the S3 processor course is that it fits easily in all kinds of environments regardless of their level of resource availability. In tightly resource constrained environments, free online available simulations can be used, and in wealthier environments,

learners can build their processors on real world FPGA boards. FPGAs allow for interactive learning [4].

Not only is the learner able to follow the functionality of the processor on different levels of abstraction, but also they can develop and build the functionality that they desire in the processor, passing through all levels of the revised Bloom's taxonomy, and being able to make the "cognitive leap" from abstract to concrete [11]. PBL approach is used in building the S3 processor; Activities are organized so that learners start from the very beginning to build blocks that will be later used in the processor. Almost in every session, there is a new block or aspect that is used in the processor which is built or learned by the students. Learners are essentially autonomous during the sessions and also they have further suggested work to do as exercises if they want. They also work in small groups of 2 learners, and these groups also communicate and discuss difficult issues. At the end of the semester, a real processor with a real assembly language that can be used in order to program realistic algorithms is achieved.

B. Learning ecosystem

The course is a required module for second year computer science students. It does not have digital systems or computer architecture prerequisites. Even basic logical algebra knowledge is addressed in the theoretical part of the course and practiced in the first few practical lessons.

No previous VHDL knowledge is required. Also, building the processor itself does not need profound knowledge in VHDL. Only some basic aspects are needed during the course and it is explained during the sessions. Most of the work is done using schematics which are relatively easy to understand by learners.

Students have access to FPGA boards during practical sessions and access to computer simulation software throughout the day. The simulation tool is freeware, when high speed internet connection is available, learners can also easily download and install the simulation tool on their personal computers. Even when high speed internet is not available, when tested on an individual level in less developed regions and countries, learners could find solutions to install the simulation software and build their S3 processors.

Consequently, at the end of the semester, not only did learners build a complete high level modern processor but also learnt critical computer science and computer architecture aspects such as Registers, Buses, ALUs, Control Units, Memories, Instruction formats, microcommands, I/O interfaces, Von Neumann architecture, Pipeline notion, etc.

https://sites.google.com/site/l2s3ae/

In the classroom course, the learners worked in pairs, on desktop computers. Each computer have Xilinx ISE design suite. Each pair used a Digilent [12] nexys2 or nexys3 FPGA board. Upgrade to Nexys4 or Basys3 board is trivial and should be done soon on the website.

Learners had to understand the French language to follow the course. However, by February 2015 an English version was added to the website.

At the University of Science and Technology of Lille we have 5 groups of 30 students every year.

C. Course structure

The third semester offers 12 weeks of teaching (S3 processor name comes from 3rd Semester). Each week students follow:

- a lecture to introduce concepts, 1.5 hour
- an exercise session per group of 30 students to apply the concepts and to prepare their usage on the board, 1.5 hour per group
- the practical lab on PC Linux with their nexys boards, 2 hours per group (24 lab. hours)

This program is a step by step discovery of architecture design. The first web page of the course explains how to install the different tools (ISE webpack from Xilinx) to their own machines. A university pre install was done for all the PC classrooms.

The course is composed of the following units:

1. ISE discovery for Nexys:
    a. Learner builds first simple design
    b. I / O concepts of the Nexys2 or Nexys3
    c. Use of Xilinx and Digilent tools
    d. Learner passes through their first complete development chain
    e. Use a simple schematic diagram
    f. Create the bitstream file using ISE
    g. Test the electronic behavior on the board
    h. Link between schematic and pins of the FPGA
2. Logical gates:
    a. Build of first schematic using logic gates
    b. Use of the ISE libraries
    c. Simulate the behavior of the circuit using the Isim interactive interface (the simulation tool inside ISE)
3. 4-bits adder (This is the first part of the S3 ALU):
    a. Concept of reusability
    b. Build more complex circuits
    c. Advanced simulation methods
4. 16-bits display on Nexys:
    a. The concept of sequential circuits

b. Basic concept of an FSM (Finite State Machine) and time in circuit design

c. Seven segments display

5. FSM design and elementary concepts of VHDL (This lesson is the most VHDL oriented lesson):
   a. Introduction to FSM behavior with clocked design
   b. Real life application: Simple traffic light experimentation simulation and board validation using 6 LEDs and a few buttons to change the FSM behavior
   c. Basics of FSM VHDL coding
   d. Concept of a control Unit
6. Registers and buses:
   a. Introduction to the registers
   b. Concept of a bus
   c. Beginning of the S3 design with 15 registers connected to a bus (Fig. 1)
   d. Register transfer (achieved by a trivial FSM)
7. Instruction memory and load phase in Control Unit:
   a. Concept of memory (ROM)
   b. Concept of IP (Intellectual Property)
   c. Manipulating the program execution using a program counter register
   d. Understand and build the different instruction execution phases
8. Move instruction and first program:
   a. Format of instruction
   b. The first instruction MOV
   c. The concept of execution pipeline
   d. Instruction decoding using the FSM
9. ALU (supports a few thousands of different processing instructions if needed) :
   a. Concept of no-operand ALU
   b. Adding functionality to the ALU and control unit
10. Conditional instruction and immediate operand:
    a. Conditional MOV instruction (according to the value of the last ALU operation)
    b. Building branch instructions (conditional MOV to PC)
    c. Immediate addressing
11. Assembly language and S3 program development:
    a. Basic and simple assembly language (possible thanks to the simplicity of the instruction format)
    b. Learners create their own language
    c. Use of substitute string tool (such as SED [13]) to generate the configuration file of the ROM component
12. Data memory, stack and Call/Return implementation:
    a. Data memory with READ and WRITE instructions
    b. Autonomous stack

By now, the S3 processor is almost ready to develop interesting programs running on the board with their own development for each pair of students. The final processor architecture is depicted in figure 2.

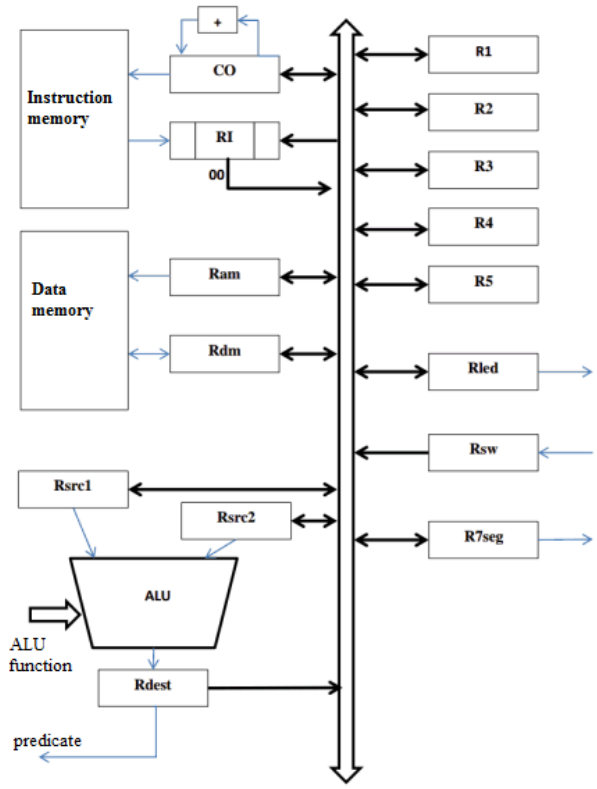

Fig. 1. The register structure of the S3 processor

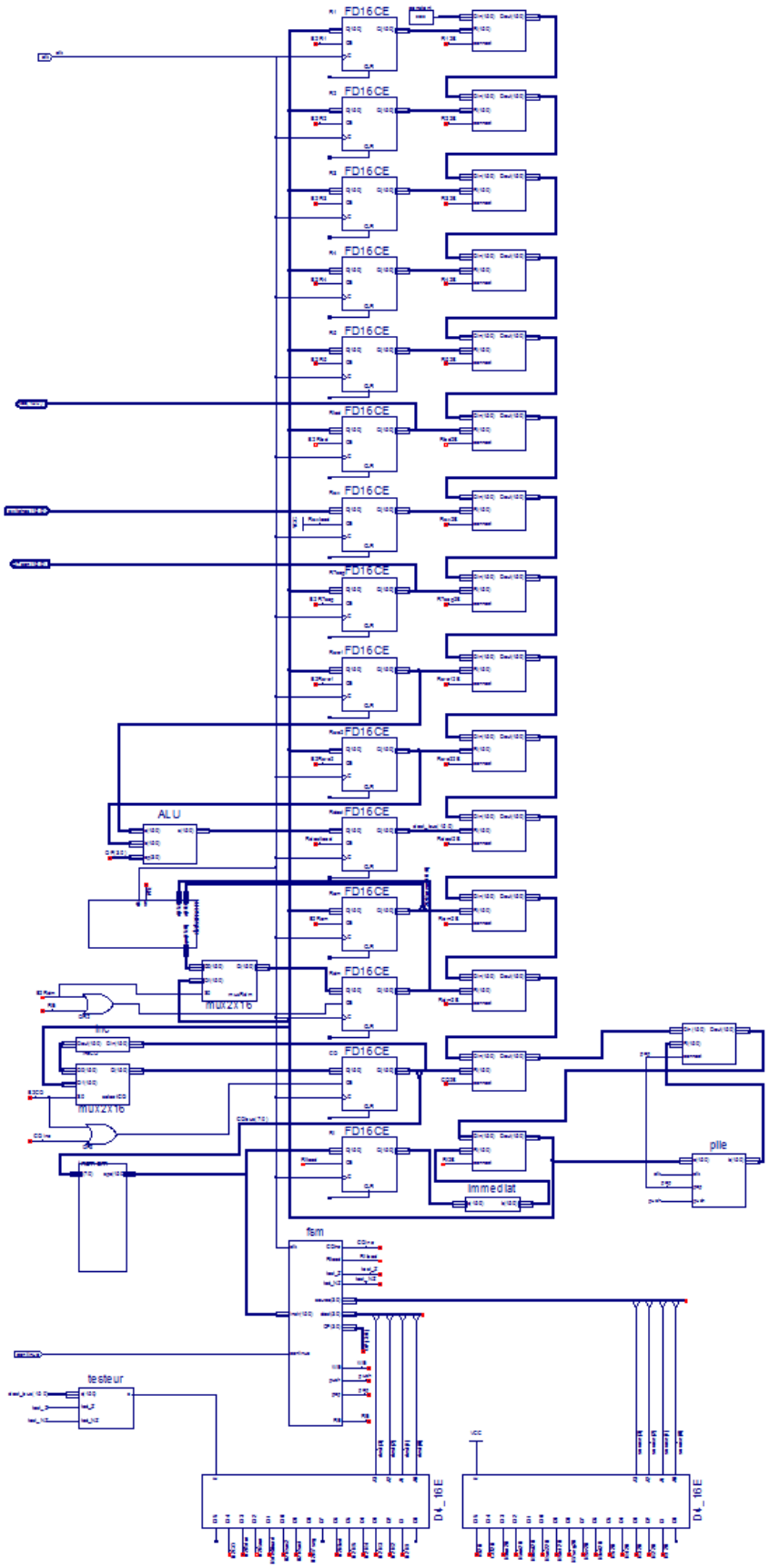

Fig. 2. Final result after 24 hours

D. Technical aspects

Many attempts were and are undertaken to create microprocessors in the higher education context. The DidaComp in [4], which manipulates 4 bits data with only 4 instructions and with a very limited amount of memory, is an example. In [14] a 32-bit Harvard RISC processor is proposed. The processor is synthesizable on FPGA using Verilog and VHDL. The processor has a constant restricted 32 instruction set. In [15], a basic model is provided to the learners. The model already contains the CPU, Memory, Address Decoder, Timer, and General Purpose Input/Output. Also, the system programming necessitates the use of the Eclipse IDE which is a relatively complicated environment.

The S3 process is a Harvard RISC processor with a very simple yet very easy to evolve architecture and microarchitecture. It provides a 16-bit processor, that executes one instruction per clock cycle with 2 stage pipeline (Load/decode, execute). The pipeline used delayed branch to avoid pipeline break, keeping a simple FSM design. Only a minimal instruction set is suggested for the S3 process, which allows the user/learner to add and implement the instructions needed for the running program.

The processor contains, as depicted in figure 1, 5 general purpose registers; R1 to R5. The user interface of the processor is realized using 3 registers; Rsw, which stores the 8-bit input data entered on the switches of the board, and Rled and R7seg, connected to the 8 LEDs and to the 7-segment display of the board which are used as the output means of the processor. The CO register and the RI register are used during the loading phase in order to point to the next instruction and store the executed instruction respectively. Ram and Rdm are registers used with the data memory. Finally, the Rsrc1, Rsrc2 and Rdest are the ALU registers.

E. Instruction set

All instructions are coded on 16 bits. We propose 4 fields of 4 bits

- Operation code: 4 bits to specify the type of operation: MOV, ALU Mov conditional Pause, READ, Write, etc.
- Source operand: 4 bits to select one register among 15. It is used for move operations
- Destination operand: 4 bits to specify 1 among 15 registers
- Immediate addressing: 8 bit can overlap operand source to specify immediate data coded on 8 bits as source of move operations.
- Instruction: on 12 bits overlapping source and destination we can specify the kind of operation realized by the ALU (up to 4096 different possible operations. Extension with particular IPs seems very easy. Input/output registers are implicit for ALU (Rsrc1 Rsrc2 and Rdest).

The following is a SED file to transform a text assembly program into 16-bits instruction file:

```
s/\(.*\)/\U\1/
s/BEGIN/memory_initialization_radix=16;\n
memory_initialization_vector= \n 0000/
/--/d
s/NOP/0000/
s/PAUSE/F000/
s/MVI/2/
s/MZ/30/
s/MNZ/40/
s/MIZ/5/
s/MINZ/6/
s/MOV/00/
s/SUBC/1000/
s/SUB/1200/
s/ADDC/1300/
s/ADD/1100/
s/INV/1400/
s/AND/1500/
s/OR/1600/
s/INC/1700/
s/CPL2/1800/
s/CONCAT/1900/
s/ID/1A00/
s/DEC/1B00/
s/MUL8/1C00/
s/READ/7000/
s/WRITE/7100/
s/PUSHI/9/
s/PUSH/80/
s/POP/A00/
s/ZERO/0/
s/R1/1/
s/R2/2/
s/R3/3/
s/R4/4/
s/R5/5/
s/RLED/6/
s/RSW/7/
s/R7SEG/8/
s/RSRC1/9/
s/RSRC2/A/
s/RDEST/B/
s/RAM/C/
s/RDM/D/
s/CO/E/
s/RI/F/
s/ //g
s/END/;/
```

This file allows for 27 instructions. Learners, and users, can add other instructions and functionality depending on their applications and needs. Adding new instructions (up to few thousands) is a very simple task; once the functionality of the instruction is determined, it must be implemented in the ALU and added to the FSM and to the SED file.

Instruction	Code binaire	Code hexa
begin	0000 0000 0000 0000	0000
MOV Rsw Rsrc1	0000 0000 0111 1001	0079
MOV Rsw Rsrc2	0000 0000 0111 1010	007A
ADD	0001 0001 0000 0000	1100
MOV Rdest R7seg	0000 0000 1011 1000	00B8
end		

Fig. 3. Example of code transformation

Figure 3 shows an example of code transformation using the previous SED file. The user/learner writes the assembly language program (in the Instruction column) and passes it to the SED instruction along with the previous SED file in order to binary/hexa code that should be stored in the instruction memory of the S3 processor.

By now, it is obvious that the learner has acquired advanced processor architecture using modern technical and learning strategies. From the technical point of view, the learner has acquired sufficient knowledge about microprocessor functionality and architecture, as well as the use of FPGAs and IPs. This is a very important feature of the S3 processor; while still very simple, it covers without any added complication advanced topics such as hardware reusability and the use of IP cores. On the other hand, from the learning point of view, learners used all revised Bloom's taxonomy levels; remembering, understanding, applying, analyzing, evaluating and creating.

III. OUTCOMES

A. *Learning outcomes*

At the end of the semester, learners succeeded in building a very powerful microprocessor from scratch, including all needed hardware and software architecture aspects and principles. They could build the processor's register set, an ALU that is capable of including as much functionality as desired.

It is evident that learners have acquired solid knowledge in all computer architecture aspects, from basic and simple, as logic ports and input/output, up to very detailed ones such as microprocessor's microarchitecture.

They also learn how to implement these aspects on FPGAs using multi-level schematics and some VHDL programming. Advanced topics in digital design were covered and used by the learners, such as IP cores and hardware reusability.

Learners at the end of the semester are completely aware of all the details of the processor they have built. They can use it as a real processor and they can run and simulate the programs they write for it. Figure 4 is an S3 assembly program that allows the user to enter two 8-bit words on the switches, display them on the 7-segment display, calculate the sum and then display this sum on the 7-segment display. The figure also shows the hexadecimal machine language code for the program. Figure 5 shows the simulation of the execution of the program in Figure 4 using the Isim simulation tool.

Table I shows the outcomes of the course and how these outcomes were met. Learners were assessed using quizzes, exams, bonus points and particularly by following them during the whole semester. More than 75% of them completed at least ten lessons. After 10 weeks we could verify for these students functional processors that they could use to input/output and process data. 10-20% of the learners acquired very good knowledge in architecture and were motivated to work on the course at their homes. In each cell of this table, we list the outcome in bold. This table is constructed in the same manner as the "information literacy competency" taxonomy table in [16]. However, the table in [16] is built using the original Bloom's taxonomy. While we could easily construct the table for the original taxonomy, we would prefer here to present a table that we constructed and called **the revised Bloom's taxonomy in computer architecture education**.

B. *Website statistics*

From September 2014 more than 25000 pages were accessed, of which 85% from France because the English version was only released recently and 75% from Lille by our students during the semester. Since January, starting a new access count, we identified 31 different countries, with more than 300 different users (teaching in Lille finished in December).

```
begin                0000
PAUSE                F000
MOV Rsw Rsrc1        0079
MOV Rsw R7seg        0078
PAUSE                F000
MOV Rsw Rsrc2        007A
MOV Rsw R7seg        0078
ADD                  1100
PAUSE                F000
MOV Rdest R7seg      00B8
end
```

Fig. 4. S3 assembly language program (Entering 2 words of 8 bits from the switches to the ALU input register and transfer the result of ADDing them to the 7 segment output register) with the assembled machine language code

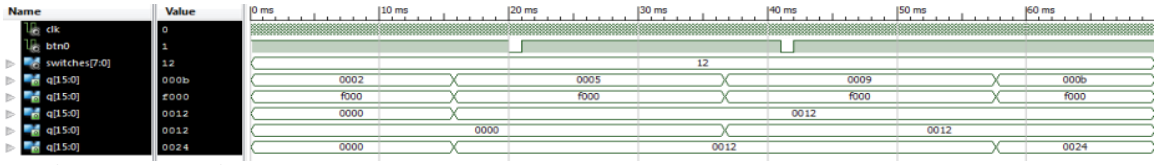

Fig. 5. Simulation of the the program of Fig. 4

TABLE I. COMPUTER ARCHITECTURE EDUCATION TAXONOMY

	Hardware	Software
Remember	**Learners will recognize and recall different architecture aspects and concepts** The course is conceived as a PBL course, which requires working on long term tasks. Learners need to recall many aspects during the semester	**Leaners will recall general algorithmic aspects and use them in programming the processor** Learners write assembly language programs for the processor, which needs basic algorithmic knowledge, that learners have acquired in a previous semester
Understand	**Learners will explain what the components of a hardware system do** Because learners build the system in a multilevel Lego game manner, they master the role of each component in the system	**Learners will describe the role of software in a system** Understanding the software architecture of the processor (machine language, micro-code, assembling programs) allows learners to assimilate the role of software in the processor
Apply	**Learners will identify When would the hardware suit their needs** The main objective of our course is to build a functional processor, so that learners understand well when and where the complete system, as well as each of its components, meet concrete needs	**Leaners will choose which software fits a certain situation** Learners acquire knowledge regarding the difference between hardware and software solutions, and when it is better to build functionality in hardware vs. when they should write a program to acquire it (hardware Inc is one cycle time less than software Add)
Analyze	**Learners will discover how a piece of hardware works** Learners build their processor using basic and simple digital circuits / components. Thus they fully understand the functionality of all pieces of hardware and of the overall system at the end	**Learners will explicate how a piece of software works** As in the case of hardware, the assembler is built using simple and basic tools. Accordingly, learners understand the functionality of the processor's software architecture
Evaluate	**Learners will select ways to improve hardware design** Learners discover different ways to solve one hardware problem and learn basics about slecting one of them	**Learners will explain how to produce quality software** Learners are briefly presented to software architecture optimization concepts such as pipelining, delayed branching as well as control flow instructions
Create	**Learners will be able to improve the design** Some learners continue their work in answering the challenging questions at the end of each lesson. These questions propose improvements to the component they have built in lab.	**Learners will build functionality and instructions to the processor and write programs** Being flexible, Learners can easily add instructions to the processor when they realize situations that require hardware functionality rather than software

With international access the English version's access should increase. In universities, Nexys3 is a well-known board throughout the world and a lot of students and teachers are looking for exercises and tutorials on google.

We are also aware of self-learning individuals, living in different countries that used the course to acquire knowledge about FPGAs, and computer architecture. They expressed their satisfaction with the course.

IV. CONCLUSION AND FUTURE DIRECTIONS

In this paper we presented a PBL approach to teach computer architecture in computer science higher education. The approach allows the learners to acquire knowledge in the considered field passing through all six levels of the revised Bloom's taxonomy.

The approach is based on building a processor from scratch. Practically, no previous digital design or computer architecture knowledge is needed. While based on simple aspects, building the S3 processor allows the learners to acquire basic knowledge in processor architecture and modern digital design using FPGAs. This knowledge is acquired thanks to the PBL approach. Learners build simple logic blocks and combine them on a semester long period in order to get more complex blocks.

Outcomes attainment was assessed using continuous follow of learners as well as quizzes, exams and bonus points which were given to motivated learners that could go far in the construction of the processor. Learners considered this approach more as a game, which improved considerably their motivation.

The Lego project approach of the course was very much appreciated by our new generation learners and we could evaluate this by their high motivation compared to approaches adapted in previous years. The simplicity of the course; no need to learn advanced VHDL or enter in the complicated details of the components, allowed the learners to acquire lasting knowledge about architecture, that we can easily identify two years later when they start master studies.

The authors are planning to transform the course into a mooc (Massively Open Online Course). While previous attempts to allow web-based teaching of computer architecture exist, up to our knowledge, a comprehensive computer architecture course such as the one considered in this paper has never been proposed.

A subsequent course for postgraduate students is available. It is based on the HoMade processor developed for research activity. This processor is developed by using different IPs in VHDL to extend the instruction set for mono and multiprocessor usages on Nexys board. Each student used his own board during the semester, thanks to the French ministry of education financial support.

ACKNOWLEDGMENTS

The authors would like to acknowledge the efforts of Ms. Diana Samson for reviewing the educational aspects of the paper.

REFERENCES

[1] Grunbacher, Herbert, and Maziar Khosravipour, "Teaching Computer Architecture."

[2] Shine, V. J., and P. K. Sathish, "Teaching Computer Architecture Using Simulation Tools." International Journal of Computer Science & Information Technologies 5.2 (2014).

[3] Wolffe, Gregory S., et al. "Teaching computer organization/architecture with limited resources using simulators." ACM SIGCSE Bulletin. Vol. 34. No. 1. ACM, 2002.

[4] Cifredo-Chacón, M., Ángel Quirós-Olozábal, and José María Guerrero-Rodríguez, "Computer architecture and FPGAs: A learning-by-doing methodology for digital-native students." Computer Applications in Engineering Education (2015).

[5] Machanick, Philip, "Experience of applying Bloom's Taxonomy in three courses." Proc. Southern African Computer Lecturers' Association Conference. 2000.

[6] Forehand, Mary, "Bloom's taxonomy." Emerging perspectives on learning, teaching, and technology (2010): 41-47.

[7] Kastelan, Ivan, M. Barak, V. Sruk, M. Anastassova, M. Temerinac, "An approach to the evaluation of embedded engineering study programs." Information & Communication Technology Electronics & Microelectronics (MIPRO), 2013 36th International Convention on. IEEE, 2013.

[8] "Computer Science Curricula 2013 Curriculum Guidelines for Undergraduate Degree Programs in Computer Science", http://www.acm.org/education/CS2013-final-report.pdf

[9] Solomon, Gwen, "Project-based learning: A primer." TECHNOLOGY AND LEARNING-DAYTON- 23.6 (2003): 20-20.

[10] Martínez-Monés, Alejandra, E. Gómez-Sánchez, Y. A. Dimitriadis, "Multiple case studies to enhance project-based learning in a computer architecture course." Education, IEEE Transactions on 48.3 (2005): 482-489.

[11] Djordjevic, Jovan, Bosko Nikolic, and Aleksandar Milenkovic, "Flexible web-based educational system for teaching computer architecture and organization." Education, IEEE Transactions on 48.2 (2005): 264-273.

[12] www.digilentinc.com

[13] http://sourceforge.net/projects/gnuwin32/files/sed/4.2.1/sed-4.2.1-setup.exe/download

[14] Poduel, Bikash, P. Kanasakar, S. R. Chhetri, S. R. Joshi, "Design and Implementation of Synthesizable 32-bit Four Stage Pipelined RISC Processor in FPGA Using Verilog/VHDL." Nepal Journal of Science and Technology 15.1 (2015): 81-88.

[15] Lee, Jong Hyuk, S. E. Lee, H. C. Yu, T. Suh, "Pipelined cpu design with fpga in teaching computer architecture." Education, IEEE Transactions on 55.3 (2012): 341-348.

[16] Vitolo, Theresa, and Chris Coulston. "Taxonomy of information literacy competencies." Journal of Information Technology Education: Research 1.1 (2002): 43-52.