Teaching Hardware/Software Co-Design Using a Project-Based Learning Strategy

Pedro Miguens Matutino*†, Tiago Dias*†‡ and Pedro Sampaio*
*Instituto Superior de Engenharia de Lisboa – ISEL, Instituto Politécnico de Lisboa – IPL
Rua Conselheiro Emídio Navarro 1, 1959-007 Lisbon, Portugal
†INESC-ID, Rua Alves Redol 9, 1000-029 Lisbon, Portugal
‡POLITEC&ID, Estrada de Benfica 529, 1549-020 Lisbon, Portugal
Email: pedro.miguens@isel.pt, tiago.dias@isel.pt and pedro.sampaio@isel.pt

***Abstract*—This paper presents a project-based learning strategy for teaching hardware/software co-design in modern Computer Engineering undergraduate courses. This kind of approach is considered by several recent pedagogical studies as the ideal strategy to support great learning achievements and to increase the proficiency of the students both in the design of digital systems and programming. The proposed strategy, targeting the first-year students, focuses on deepening the knowledge of combinatorial logic, sequential circuits, and state machines, alongside the hierarchical development of software, including for peripheral management.**

***Index Terms*—Computer engineering education, project-based learning, digital systems and programming**

I. Introduction

Digital Systems and Programming are two introductory courses in the curricula of Computer Engineering, Computer Science, and Electrical Engineering undergraduate courses. Although these courses mostly address basic concepts of digital systems design and programming, students have difficulty understanding many of these fundamental ideas, which greatly difficult the teaching and learning process [1]. Extensive laboratory practice is considered nowadays as the best pedagogical strategy to enhance student learning [1], [2]. Furthermore, even in more recent approaches, the hardware and software topics usually are taught independently [3]–[6].

However, implementing such a learning-by-doing approach in undergraduate courses is extremely hard, also due to the continuous reduction in the time devoted to lecture and lab hours. To jointly tackle all these issues, we implemented a project-based learning strategy for the development of a hardware/software co-design project, in contrast to the traditional approach that is based on classroom lectures with small laboratory projects. The uniqueness of this strategy is that the students have to develop a semester-long hardware/software project, using a Hardware Description Language (HDL) and high level programming language for the hardware and software implementations, respectively.

This hardware/software co-design project is developed in Informatics and Computer Laboratory, which is a course in the second curricular semester of the BSc in Computer Science and Computer Engineering lectured at Instituto Superior de Engenharia de Lisboa (ISEL), the school of engineering of the IPL - Politécnico de Lisboa, in Portugal. In order to enroll in this course, the students have to get approval in two previous courses of the first curricular semester of the BSc: Programming and Logic and Digital Systems. Furthermore, the concepts addressed by the Electronics curriculum are also consolidated in this course.

This paper is organized as follows. Section II describes the course objectives and the intended learning outcomes, alongside with the continuous and final assessment. The devised project-based learning strategy and the case study example are presented in Section III and IV, respectively. Section V details the laboratory infrastructure used to support the development of the semester-long hardware/software projects. Section VI introduces and discusses the obtained results, including the drop-out, not pass and pass rates among nine semesters. This paper is concluded in Section VII with the final remarks.

II. Course Aims

The Informatics and Computer Laboratory course aims to consolidate the concepts taught in Programming and Logic and Digital Systems, through the realization of a project of medium complexity involving hardware structures that interact with software. Furthermore, students who successfully complete this course, as a consequence of learning outcomes, are expected to be able to:

1) Analyze, propose and implement the hardware and software components of medium complexity programmable digital systems to solve common day problems;
2) Write reports that describe the problems and choices made to achieve the proposed solutions;
3) Work in-group, manage time to perform multiple tasks simultaneously, meet the established deadlines;
4) Discuss and defend the proposed solution for each component implementation.

These learning outcomes are achieved with the following syllabus:

i) Design of digital circuits using programmable logic, using the basic concepts taught in the first semester course of Logic and Digital Systems;

978-1-7281-6732-9/20/$31.00 ©2020 IEEE

ii) Development of control programs for the implemented hardware structures, using the concepts acquired in the Programming course, also in the first semester;
iii) Serial and parallel transmission and reception;
iv) Synchronization and flow control in digital transmission;
v) Read and write cycles in memory components.

This syllabus is taught along the semester, in up to 30 lectures corresponding to 67.5 hours of contact. All these lectures take place in a laboratory. Usually, the lectures are divided into 15 lectures of 3 hours used to design and implement hardware modules, and 15 lectures of 1.5 hours for the design and implementation of the software modules. The total student job time is about 162 hours, apportioned between laboratory sessions and autonomous work. The lectures are intended for presentation, design, discussion, and implementation of the several modules that implement the system that solves the proposed problem. In order to design and implement each module, different hardware and software concepts are addressed. These concepts are presented and discussed in Section III.

At the beginning of the semester, the project, is presented to the students, which includes a brief description of the main features and the project specification. Jointly with the project specification, it is also provided the base structure that supports the solution, as well as, the block diagram of all the modules that compose the system. Students are required to prepare the laboratory session in advance, for which a milestone is previously set and published in the school educational electronic platform (Moodle) for each class. As soon as all the groups have outlined a solution, a period of debate is open to discuss the different solutions proposed by the groups, and whenever appropriate, alternative solutions are presented by the teaching staff. This methodology is also employed whenever a given group is confronted with a relevant implementation problem. At the end of the semester, each group prepares a complete report of the entire project, presenting in detail the choices taken and the obtained results.

This teaching methodology evidences the strong hands-on approach component of the course, since all the assessment is based on the developed project. At the end of the semester, the learning results are assessed by a group presentation of the project, and by an individual Viva Voce Examination (VVE). In addition, two Inter-Layer Assessments (ILA) are performed during the semester. The first one is in the fifth or sixth week of the semester, and the second one six weeks after that. These assessments are extremely important because they allow to establish deadlines and to give feedback to students about the project progress. Moreover, the teaching staff monitors the development and teamwork to prevented students from parasitizing their hard-working peers, sometimes as an extreme measure, new team arrangements are established.

The final grade (FG) is obtained by weighting the intermediate classifications (ILA), according to Eq. 1, where the minimum grade allowed for the ILA and VVE examinations is 10 and 8 values, respectively, in a grading scale of 1 to 20.

$$FG = 0.3 \times ILA + 0.7 \times VVE\ . \tag{1}$$

III. HW-SW CO-DESIGN IN A PROJECT-BASED LEARNING APPROACH

The hardware/software co-design project that is presented to students is of medium complexity and allows them to consolidate the concepts learned in the Digital Systems and Programming courses by implementing hardware structures using reconfigurable logic devices that interact with software running on a PC, which they are also asked to develop. The project is structured to address three main topics:

i) user I/O using a keypad and the corresponding decoding circuit, which allows deepening the knowledge of state machines and synchronization;
ii) synchronous serial communication, to review buffering problems and communication protocols;
iii) hierarchical software design, including the software for the management of peripherals.

The project is partitioned in several tasks, the first ones consisting of the development of a hardware module and the corresponding software device manager, while the last task is focused on the development of the application software. These tasks are carried out in groups of up to three students (size varies with enrollment), which are self-formed at the beginning of the course by the students without the meddling of teaching staff. Developing the project in a group enforces teamwork and communication capabilities, soft-skills that most students have not yet acquired due to being in their first year.

The project is well defined without too many open-ends, which allows the students to establish a development workflow for each component of the system, that can be considered as: *i*) definition of the requirements; *ii*) design; *iii*) implementation; *iv*) standalone component testing; *v*) integration with other components; and *vi*) integration tests. For each task, a milestone is previously set, as mentioned before, allowing the students to prepare each laboratory and to manage their work time. At the beginning of the course, these milestones are presented to students using a Gantt chart, to illustrate the project schedule and allow them to better manage their time.

IV. CASE STUDY

The project proposed for the 2019/20 fall semester, considered herein as a case study, proposes an implementation of an Access Control System to manage the opening of a door giving accesses to a specific area. This access is granted only when the user introduces a valid pair of *uin* and *pin*.

As shown in Fig. 1, the proposed system is based on: *i*) a *Keyboard Reader* (HW), to acquire the pressed keys; *ii*) a *Serial LCD Controller*, SLCDC (HW), for communication with a LCD; *iii*) a *Serial Door Controller*, SDC (HW), to interact with the door lock (*Door Mechanism*); and *iv*) the *Control* module (SW) implementing the application functionality.

The application has to implement two operating modes, commuted throw the switch M: the normal operation mode,

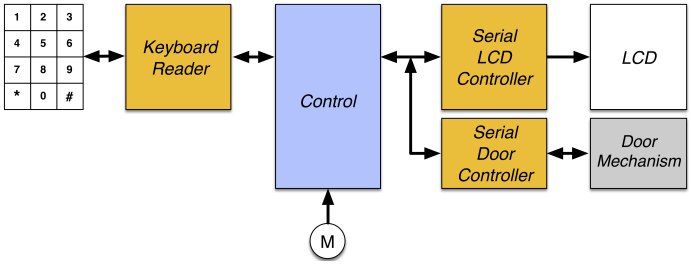


Fig. 1: Block diagram of the Access Control System.

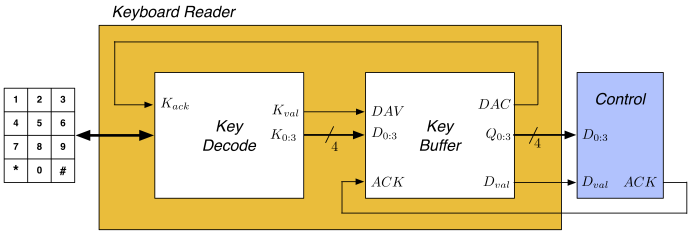


Fig. 2: Block diagram of the Keyboard Reader.

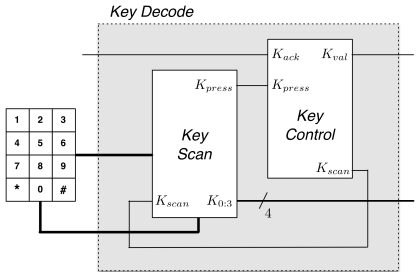
named as access mode and the other mode, named as maintenance mode. The maintenance mode allows to insert and remove users, and also to shutdown the system. The *Keyboard Reader* module is responsible for decoding the 12-key matrix keyboard, determining which key is pressed and making its code available in four bits to the *Control*, if it is available to receive it. If it is not available to receive it immediately, the key code is stored up to the limit of two codes. *Control* processes the data and sends the information containing the data to be displayed on the LCD through the SLCDC. The commands for the door mechanism are sent through the SDC. For physical reasons, and in order to minimize the number of interconnection wires, the communication between the *Control*, SLCDC and SDC modules is carried out using a serial protocol.

A. Keyboard Reader

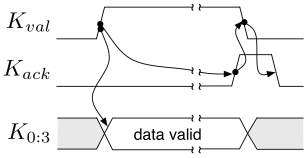
The *Keyboard Reader* is split into two blocks: the *Key Decode* and the *Key Buffer*, as depicted in Fig. 2. The first block decodes the pressed key, while the second one implements a data buffer, allowing that another key can be captured and decoded alongside with a delivery to the *Control* module.

Regarding the *Key Decode* module, whose architecture is shown in Fig. 3, students are provided with three different architectures for the *Key Scan* block, due to their restricted autonomy as first years students of an engineering course. These architectures are depicted in Fig. 4.

The students have to understand the operation of the three circuits, and also to justify the circuit they choose to implement. Beware, that all blocks used in the *Key Scan* have been taught in the Logical and Digital System course. Conversely, the *Key Control* is open-ended. It focuses in deepening the knowledge of state machines and synchronization, by implementing a full-interlocked protocol with the *Control* module, as shown in Fig. 3b.

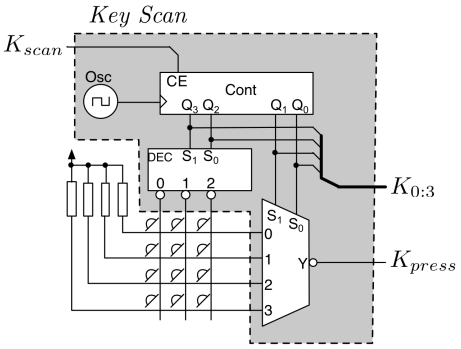


(a) Block diagram

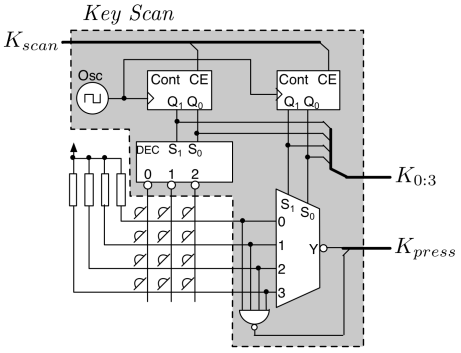


(b) Full-interlocked protocol

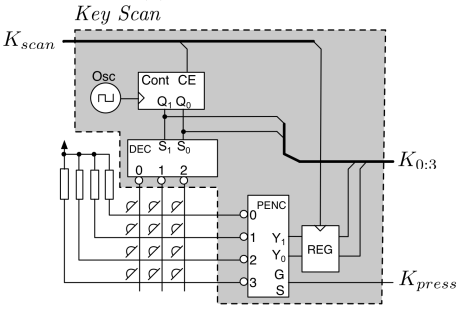
Fig. 3: *Key Decode* module



(a) Architecture I

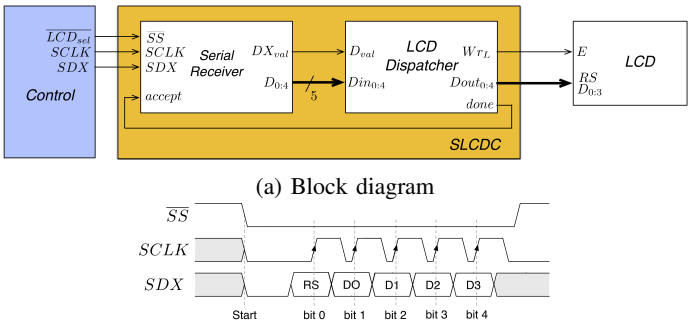


(b) Architecture II



(c) Architecture III

Fig. 4: *Key Scan* block



(a) Block diagram

(b) Serial protocol

Fig. 5: Serial LCD Controller.

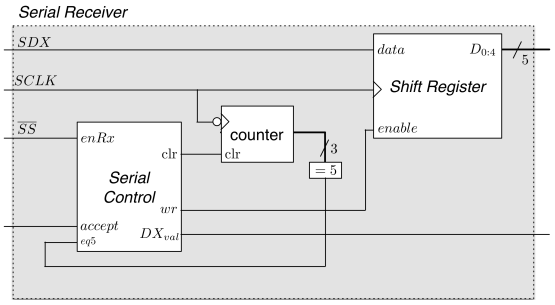
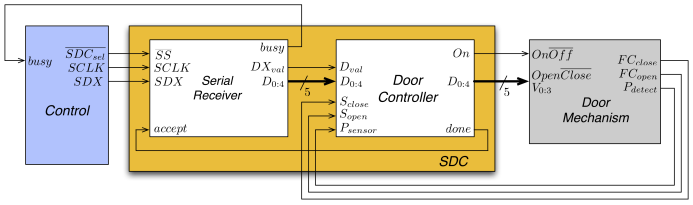
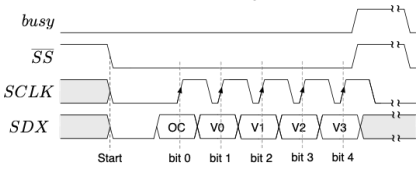


Fig. 6: Block diagram of Serial Receiver.



(a) Block diagram



(b) Serial protocol

Fig. 7: Serial Door Controller.

The *Key Buffer* module implements a data storage structure, capable of storing a four-bit word. It is also responsible for the interaction with the consumer system, i.e. the *Control*, for which it implements an identical protocol, as used in the interaction between the *Key Decode* and *Key Buffer*, depicted in Fig. 3b.

The *Key Buffer* stores the data into memory when the DAV (Data Available) signal becomes active. After storing the data, *Key Buffer* set the DAC (Data Accepted) signal, to notify the producer that the data was stored, implementing also a full-interlock protocol. Although the block diagram of this module is also open-ended, students are required to design it using a finite state machine (*Key Buffer Control*) and an external register.

When the *Control* wants to read data from the *Key Buffer*, it waits for the D_{val} signal to become active, collects the data and activates the ACK signal to acknowledge that it has been consumed. As soon as the ACK signal becomes active, the *Key Buffer Control* should invalidate the data by deactivating the D_{val} signal. It should only re-store a new word after *Control* has deactivated the ACK signal.

B. Serial LCD Controller

The LCD interface module (*Serial LCD Controller*, SLCDC) is responsible for receiving the bit wise data sent by the *Control* module, combine it to form data and control words, and later delivering it to the LCD controller. For this, two different modules are employed, as shown in Figure 5a.

The SLCDC receives, in series, a message consisting of five bits of information. Communication with the SLCDC takes place according to the protocol illustrated in Figure 5b, with the first bit of information, the RS bit, indicating whether the message is a control or data word, and the remaining bits containing the data to be transferred to the LCD.

The SLCDC *Serial Receiver* subblock consists of three main blocks, as shown in Fig. 6: *i*) a control block; *ii*) a serial to parallel conversion block; and *iii*) a counter of received bits, designated respectively by Serial Control, Shift Register, and Counter.

The *Dispatcher* subblock delivers the word received by the *Serial Receiver* to the LCD controller by activating the Wr_L and the DX_{val} signals. The *Dispatcher* then signals the *Serial Receiver* that the data has been delivered by activating the $done$ signal.

C. Serial Door Controller

The *Serial Door Controller* (SDC) is the interface module with the door mechanism. It implements the serial reception of the five bits word sent by the *Control* module, later delivering it to the door mechanism, as shown in Fig. 7.

Communication with the SDC takes place according to the protocol illustrated in Figure 7b, where the first bit is the $Open/Close$ (OC) bit that indicates whether the command is to open or close the door. The remaining bits contain the opening or closing speed information. The SDC indicates that it is available to receive new data after processing the previous command, by setting the $busy$ signal to logic level '0'. To enforce the concept of modularity and its benefits in the design of digital systems, students are challenged to implement the *Serial Receiver* block of the SDC using a structure similar to the *Serial Receiver* block used on the SLCDC.

If the command received is Open, the *Door Controller* must activate the signal $On\overline{Off}$ and set the signal $Open\overline{Close}$ to the high logic value, until the open door sensor (FC_{open}) is activated. However, if the command is Close, the *Door Controller* must activate the $On\overline{Off}$ signal and set the signal $Open\overline{Close}$ to the low logical value, until the closed door sensor (FC_{close}) is activated. When closing the door, if a

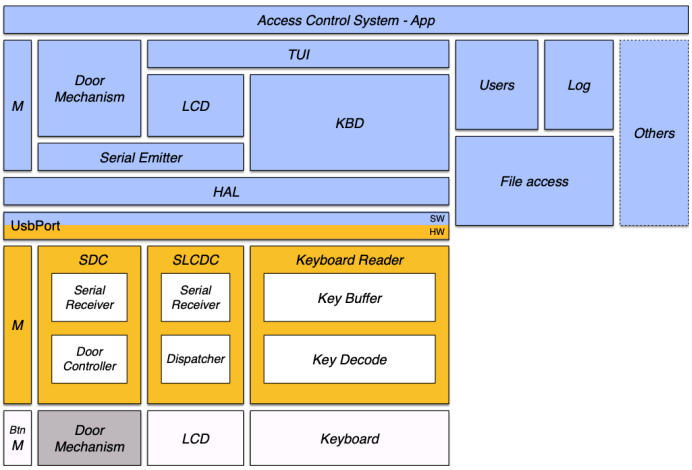


Fig. 8: Hierarchy software diagram of the Access Control System.

person is detected in the door area, by the sensor P_{detect}, the *Door Controller* must abort the closing operation and reopen the door. The *Door Controller* automatically, i.e., without the need for another Close command, closes the door as soon as there are no more persons in the door area. After completing any of the commands, the *Door Controller* signals the *Serial Receiver* that it is ready to process a new command by activating the *done* signal.

D. Control

The *Control* module is implemented in software, using the Java language and following the logical architecture presented in Fig. 8, using the blue color.

The signatures of the HAL, KBD, Serial Emitter, LCD and Door Mechanism classes are established and delivered to students. For the remaining classes, students are asked to propose both their signatures and implementations. This is the most open-ended phase in the proposed strategy, where can appeal to their creativity to come up with distinct project solutions.

V. Laboratory Infrastructure

The laboratory infrastructure is based on the μLIC Development Platform [8], but using the laboratory version of the μLIC_{χ} development board [7], depicted in Fig. 9, to implement the hardware modules, and a PC to support the development of the hardware and software modules. The hardware modules are described in VHDL, tested and syntetized using the Xilinx ISE software, and deployed to the educational hardware platform. The board has a Xilinx XC95144XL CPLD, with a built-in CPLD programmer, and two 8-bit parallel ports supporting data transfers between the hardware and the PC. Each group of students has one prototype set, that includes the development board and the peripherals required for the project development.

The application software is written in Java code, using the integrated development environment IntelliJ IDEA. Students

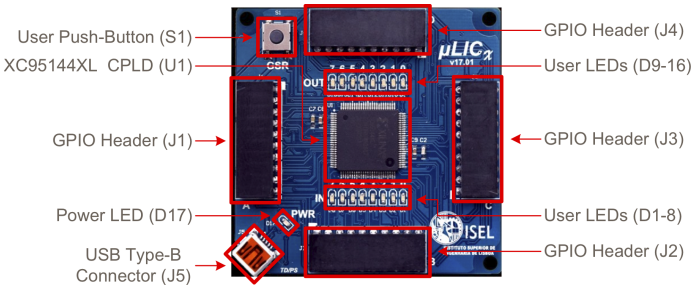


Fig. 9: μLIC_{χ} development board.

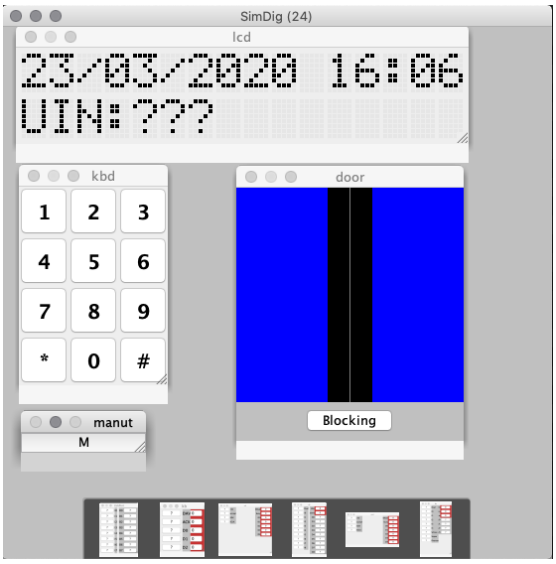


Fig. 10: Java simulation of Access Control System.

are provided with a custom Java library that implements the transactions with the μLIC_{χ} parallel ports.

Furthermore, in order to make it easier for all the students to develop their own version of the application, and test different functionalities, as well as discuss such implementation within their group, a Java simulator for the hardware is made available after the second Inter-layer assessment. In Fig. 10 is depicted the Java simulator of the Access Control System, available in [9].

VI. Results

The presented project-based learning strategy has been applied since the 2015/16 fall semester, with great success, with an average approval rate of 96% for the students that realize the final viva voce examination, as shown in Tab. I and Fig. 11. Even considering all the students that engaged the course at least by one class, the approval rate is higher than in the other courses in the same semester, achieving an average value of 82%.

A more care full analysis of the data presented in Tab. I shows that, on average, 7% of the students drop out before inter-layer assessment. This is also a lower value when compared with the other courses of the curricular plan. Fig. 11 depicts two moments of students drop-out, where *drop-out1*

TABLE I: Assessment and approval data since the 2015/16 fall semester.

Year	Semester	# enrolled students	ILA	VVE	FG			Approval rate
					#	total	$\geq 15/20$	vv exam
2015/2016	*fall*	63	0.79	0.79	46	0.73	0.15	0.92
	spring	91	0.90	0.90	80	0.88	0.30	0.98
2016/2017	*fall*	47	0.96	0.89	41	0.87	0.32	0.98
	spring	75	1.00	0.97	73	0.97	0.34	0.97
2017/2018	*fall*	50	0.96	0.90	44	0.88	0.23	0.98
	spring	81	0.90	0.78	58	0.72	0.47	0.92
2018/2019	*fall*	57	0.95	0.77	42	0.74	0.36	0.95
	spring	71	0.96	0.79	52	0.73	0.33	0.93
2019/2020	*fall*	45	0.98	0.89	39	0.87	0.18	0.98
average		**64**	**0.93**	**0.85**	**53**	**0.82**	**0.30**	**0.96**

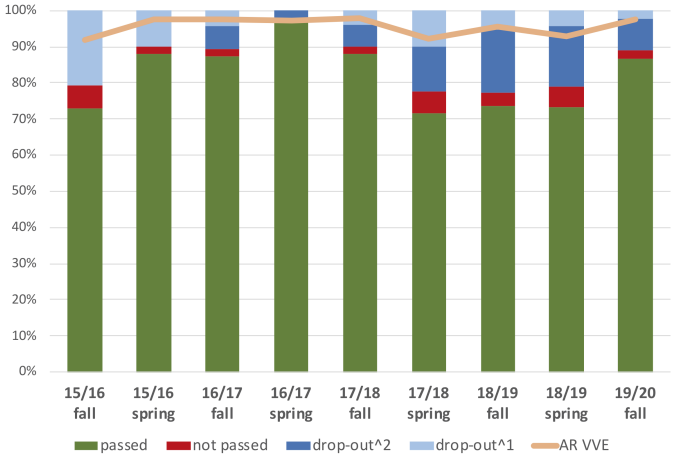


Fig. 11: Drop-out and approval rates by semester.

means that they give up before inter-layer assessment, and *drop-out2* when they drop before submitting the final report and miss the viva voce examination. Note that almost all students that take the final examination pass, as the line AR VVE in Fig. 11 shows. This is achieved because the students have been engaged in the project during an entire semester, got proficient in the developed system and underlying topics.

From the obtained results it can be concluded that this methodology motivates the students to be engaged with the project until the end of the semester, which results in drop-out values lower than 10%. Furthermore, the high pass rate, shows that the majority of the students met the learning outcomes. In addition, we have come to the conclusion that quite often one or two students of each group achieve higher grades due to providing more assertive answers and demonstrating better knowledge in the examination.

VII. Conclusions

This paper presents a project-based learning strategy for teaching hardware/software co-design in modern Computer Engineering undergraduate courses. This strategy focuses on deepening the knowledge of combinatorial logic, sequential circuits, and state machines, alongside the hierarchical development of software, including for peripheral management. This kind of approach is considered by several recent pedagogical studies as the ideal strategy to support great learning achievements. In our case, we have managed to improve the approval rates up to 96% by using the proposed strategy. Furthermore, it has been observed that the students not only have managed to consolidate their hardware and software competences but also aggregate both perspectives. In addition, students have improved their documentation skills.

Acknowledgments

This work was supported by national funds through FCT, Fundação para a Ciência e a Tecnologia, under project UIDB/50021/2020.

References

[1] H. Ochoa and M. Shirvaikar, "A Survey of Digital Systems Curriculum and Pedagogy in Electrical and Computer Engineering Programs", Proc. 2018 ASEE Gulf-Southwest Section Annual Meeting, Austin, TX, Apr. 2018.

[2] C. M. Kellett, A Project-Based Learning Approach to Programmable Logic Design and Computer Architecture, in IEEE Transactions on Education, vol. 55, no. 3, pp. 378-383, Aug. 2012. DOI: 10.1109/TE.2011.2179301.

[3] O. B. Adamo, P. Guturu and M. R. Varanasi, An innovative method of teaching digital system design in an undergraduate electrical and computer engineering curriculum, 2009 IEEE International Conference on Microelectronic Systems Education, San Francisco, CA, 2009, pp. 25-28. doi: 10.1109/MSE.2009.5270837.

[4] A. J. Araujo and J. C. Alves, A project based methodology to teach a course on advanced digital systems design, WSEAS Transactions on Advances in Engineering Education, vol. 5, no. 6, pp. 437–446, 2008.

[5] S. Amamou, L. Cheniti-Belcadhi, Tutoring In Project-Based Learning, Procedia Computer Science, Volume 126, 2018, Pages 176-185,ISSN 1877-0509, https://doi.org/10.1016/j.procs.2018.07.221.

[6] C. M. Kellett, A Project-Based Learning Approach to Programmable Logic Design and Computer Architecture, in IEEE Transactions on Education, vol. 55, no. 3, pp. 378-383, Aug. 2012.

[7] T. Dias, μLIC_{χ} LIC development board, Reference manual, 2017, https://www.researchgate.net/publication/338749880_uLIC-X_Deveopment_Board_Reference_Manual.

[8] T. Dias, P. Sampaio and P. M. Matutino, A Portable Lab for the Practical Study of Modern Computer Engineering, in Technology, Teaching and Learning of Electronics (TAEE), Jun. 2020.

[9] https://github.com/pmmiguens/AccessControlSystem.