International Journal of Engineering Education Vol. 35, No. 1(A), pp. 286–303, 2019
Printed in Great Britain

0949-149X/91 $3.00+0.00

Evaluation of a New Self-Study Platform for Introductory Digital Systems*

DAVID BANERES[1] and ROBERT CLARISÓ[2]
[1] Computer Science, Multimedia and Telecommunications Department and eLearn Center, Universitat Oberta de Catalunya (UOC), Barcelona 08018, Rambla del Poblenou 156, Spain. E-mail: dbaneres@uoc.edu
[2] Computer Science, Multimedia and Telecommunications Department, Universitat Oberta de Catalunya (UOC), Barcelona 08018, Rambla del Poblenou 156, Spain. E-mail: rclariso@uoc.edu

Most Bachelor programs in Computer and Electrical Engineering include an introductory course on the foundations of digital system design, e.g., combinational and sequential circuits. In this course, students have many difficulties understanding core concepts such as Boolean function minimization or circuit analysis. A potential remedy to these difficulties is self-assessment with automatic feedback, which can facilitate the acquisition of digital system design competences by allowing students to experiment with a large collection of exercises and quickly realize their mistakes. This paper aims to evaluate how an educational software tool with self-assessment features capable of providing feedback with different levels of granularity can help students acquire these core concepts. To this end, a tutoring tool to promote the acquisition of Boolean function minimization and circuit analysis skills has been implemented. Next, a quantitative and qualitative analysis on its use has been performed. Evidence shows that the self-assessment tutor has a positive impact on academic performance during the semesters under analysis. Additionally, survey responses show a high degree of acceptance and satisfaction with the tutoring tool.

Keywords: digital circuits; electrical engineering education; educational technology; automatic assessment

1. Introduction

Digital systems design is a key knowledge area in Computer and Electrical Engineering degrees. For instance, the ACM (Association for Computing Machinery) and the IEEE (Institute of Electrical and Electronics Engineers) joint task force guidelines for degrees in Computer Engineering [1] include "digital design" as a core knowledge area, with several core knowledge units such as basic logic circuits and modular design of combinational and sequential circuits. Furthermore, laboratories on circuits and electronics are considered a "must-have" experience, including tools for gate-level circuit design [2].

Introductory courses in this field typically cover the foundations of numeral systems, analysis of digital circuits and Boolean logic. In these courses, students should learn, among other skills, how to simplify a Boolean function and how to analyze a simple digital circuit.

Figure 1 illustrates two conventional activities that exercise these skills. On the one hand, *Boolean function minimization* consists in simplifying Boolean expressions (i.e., computing equivalent expressions which can be computed using fewer operations) in order to achieve more efficient circuit implementations. To this end, Karnaugh Maps are a popular minimization method which can also be applied to other disciplines [3–8]. On the other hand, *circuit analysis* refers to the modeling and simulation activities that provide insight into the operation of a circuit. Circuit behavior is commonly analyzed by inspecting the value of each output signal for a given set of input values. Two techniques for this purpose are truth tables and timing diagrams. Truth tables are recommended for circuits with less than five 1-bit input signals, while timing diagrams can be used for circuits with a larger number of inputs or bus signals.

Both Boolean function minimization and circuit analysis are considered "core learning outcomes" in the ACM/IEEE body of knowledge for computer

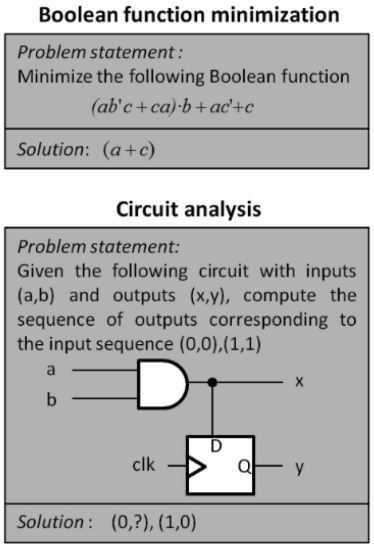

Fig. 1. Sample Boolean function minimization and circuit analysis activities.

* Accepted 14 October 2018.

engineering. Nevertheless, these competences are procedural skills that are best learned using a problem-solving learning methodology [9][10]. Experience shows that first-year students have difficulties in learning these concepts from non-interactive resources, e.g., text, video or fixed simulations. The ability to practice repeatedly and receive immediate feedback about their candidate solutions is the key to acquiring these skills. Then, more complex skills related to computer organization can be acquired more easily [11].

This paper aims to study how an interactive tool can help the students acquire the skills related to Boolean function minimization and analysis of digital circuits. The three main contributions proposed in this paper are:

- An interactive tool to minimize Boolean functions using Karnaugh Maps, capable of providing feedback about each step of the minimization process.
- An interactive tool to analyze digital circuits using timing diagrams.
- A quantitative and a qualitative analysis about the impact of the interactive tool on the student performance and engagement.

To analyze the impact of these new tools, three research questions have oriented this study:

RQ1. Has the utilization of the complete platform impacted on the student's performance?
RQ2. How has the utilization of the different tools impacted on the student's performance?
RQ3. Do the students have a positive perception of the tools?

Although there is a large number of tools to learn digital systems (See Section 2), these new tools offer relevant benefits in the learning-teaching process. For instance, instructors have full control of the collection of exercises (i.e., it is not a static repository, the list of exercises can be modified) and the level of assistance provided during self-assessment. Also, the tools have been added to the VerilUOC framework [12], which adds support for the analysis of individual and global student progression and plagiarism detection [13], student's effort estimation [14] and support to MOOC courses related to digital systems [15].

The paper is organized as follows: Section 2 introduces related work on support tools for teaching introductory concepts related to digital systems. Next, Section 3 describes the course where these tools will be deployed. Section 4 and Section 5 describe the designed support tools and Section 6 shows how they have been integrated into an existing intelligent tutoring system. Then, Section 7 summarizes the results of their application in the *Computer Fundamentals* course at an online university. Finally, Section 8 presents the conclusions and outlines future work.

2. Related work

This section describes the previous work on support tools for teaching digital systems in Electrical Engineering or Computer Science degrees. Due to the large number of works in the literature related to this topic, we focus specifically on tools for Boolean function minimization and circuit analysis.

2.1 Boolean function minimization

Several methods may be used to explain the minimization of Boolean functions: Karnaugh Maps [16], Quine-McCluskey [17–19] and ESPRESSO [20]. Within the academic community there are opposing views on which method is best suited to introduce the minimization process.

Boolean function minimization is a computationally complex problem. Karnaugh Maps and Quine-McCluskey are exact methods, i.e., they obtain the smallest function in terms of the least number of AND/OR Boolean operators. Thus, they are limited to small Boolean functions: up to 4–6 variables for Karnaugh Maps and up to 14–16 for Quine-McCluskey. Meanwhile, ESPRESSO is a heuristic method that may provide a non-exact solution. However, when the circuit is small the exact solution is commonly obtained.

Exact methods are commonly used to help students become familiar with the fundamentals of the minimization process since they can be used to perform the minimization manually. On the other hand, heuristic methods tend to be presented in more advanced courses, where students have a broad background and the goal is to design a complex circuit. Although Quine-McCluskey and ESPRESSO are valid alternative techniques, this paper focuses on Karnaugh Maps due to its simplicity compared to the other methods.

Many tools have been developed to teach Karnaugh Maps, each from a different perspective. Some methods focus on explaining the method step by step. For example, WinLogiLab [21] is a tool that helps to learn several fundamental concepts in digital systems, e.g., numerical representation and combinational and sequential circuits. In the combinational module, the concept of Karnaugh Maps is presented by letting students observe how the method is applied in a given exercise. A similar example is presented in KARMA [22], where a "teaching mode" is offered to illustrate the methodology.

WinLogiLab and KARMA also offer the possibility to obtain the correct solution to any Karnaugh Map exercise automatically. Other

tools also have this feature, such as WILEDS [23], Boole-Deusto [24], ITDiL [25], K-MAP [26], KarnaughMap445 [27] and LogicAid [28]. Currently, mobile applications such as KVD [29] or Mini-Karnaugh [30] are also available from mobile media stores. This functionality is useful for a student that needs to obtain the solution to an exercise, but it does not contribute to the learning process. A student needs to interact and to try to solve the exercise by himself in order to learn. Providing the correct solution automatically only serves as a didactical material.

Some of the previously mentioned tools (WILEDS, WinLogiLab, KARMA, Boole-Deusto and LogicAid) also support a "practice" mode. Although WinLogiLab and KARMA return the correct solution when a student fails, Boole-Deusto and LogicAid try to provide hints that guide her towards the correct solution.

After revising these tools, none of them supports the practice of the entire minimization process from the definition of the truth table to the minimization of the Boolean expression. The previously mentioned tools skip several steps in this process: defining the truth table and filling the Karnaugh map from the truth table. In these initial steps, many students tend to fail, since they do not understand the problem statement clearly or they make mistakes while building the Karnaugh Map. Moreover, students need hints or assistance in all the steps of the minimization process depending on their level of proficiency.

Newer tools have appeared in the past years. For example, SDLDS [31] is a digital systems learning tool to learn to minimize a digital circuit. This tool supports self-assessment in the circuit minimization process based on simulation and Karnaugh Maps minimization is subsumed within the minimization process. Moreover, the feedback for Karnaugh Maps is very limited and only valid within the circuit minimization. Similarly, DLD-VISU [32] was also proposed to learn and practice all type of activities related to digital systems minimization. Related to Karnaugh Maps, self-assessment is available for all the minimization steps. However, instructor setup is very limited to create exercises and only an access control list is available to control when students can access the tool.

To address these shortcomings, we propose a tool, *KeMAP*, where all the steps of the function minimization using Karnaugh Maps are covered and the level of guidance and personalized feedback provided to students can be preconfigured by instructors when the exercise is set up. Note that, students are also able to adjust the level of assistance established by the instructor if they intend to solve the exercise with a higher level of difficulty.

2.2 Circuit analysis

Circuit analysis is a harder skill to acquire since it has several prerequisites. First, students need to learn the basic digital components, such as logic gates and combinational and sequential simple components (multiplexer, decoders, registers, etc.). In the case of sequential circuits, it is also critical to understand how the clock signal affects the propagation of the input signals. Therefore, learning resources explaining all the concepts and previous work by students are required to avoid common mistakes [33, 34].

Different methodologies have been designed to learn circuit analysis mainly using laboratories. In [35], the instructors designed a course to learn FPGA design and synthesis using the proprietary software Xilinx. This software was used in the laboratory to design a circuit and to analyze its behavior.

However, some courses cannot rely on laboratories (i.e., in the context of distance learning) so simulation tools are used instead. Some tools provide a simulation feature to support the timing analysis of a digital circuit. For instance, tools like Logisim [36], LogicSim [37] or LogiFlash [38] provide an interactive mode to simulate the circuit behavior after changing the values of the input signals. Others like WinLogiLab, JLS [39], CEDAR [40] or LogicWorks [41] have an additional feature to visualize the circuit behavior with a timing diagram or oscilloscope. This type of feature is helpful to understand how the analysis of a circuit is performed. Also, some simulators include additional features related to computer architecture and organization [42]. However, simulation is not enough for practicing this skill: some degree of interaction is required.

As far as we know, no other tool lets students practice circuit analysis using timing diagrams. Existing tools return the simulation without giving students the opportunity to try on their own. To this end, we have developed a tool, *VerilChart*, to support the learning of circuit analysis skill. As in *KeMAP*, instructors can tune the amount of assistance available during practice and students can disable available hints.

3. Methodology

In this section, we describe the contents of the *Computer Fundamentals* course where the tools have been applied and the pedagogical aspects of the course design.

3.1 Course overview

Computer Fundamentals is an undergraduate course in the Computer Engineering and Telecommunica-

tions Technology Degrees. It is a first-semester course where no specific background is required. The contents of the course are divided into 5 units, organized as follows:

1. Introduction to digital systems.
2. Number representation: Positional systems, decimal and binary representations (signed magnitude, two's complement), conversion representations, and fixed and floating-point numbers.
3. Combinational circuits: Boolean algebra, function minimization, logic gates, combinational components and analysis and synthesis of simple combinational circuits.
4. Sequential circuits: sequential components, finite state machines, Moore's model, and analysis of sequential circuits.
5. Introduction to computer organization: description of different types of state machines, introduction to the computer organization for a generalized algorithmic machine, and synthesis of sequential circuits from state machines.

The practical content of this course is divided into 3 continuous assessment activities (CAA):

- First CAA: This activity is based on the second unit. Several problems related to number representation are proposed.
- Second CAA: This activity evaluates the third unit of the course. Exercises about function minimization using Karnaugh Maps, analysis and synthesis of combinational circuits are included.
- Third CAA: This activity evaluates the sequential part explained in the fourth and fifth units. Analysis using timing diagrams and design of a sequential circuit using Moore's model are assessed.

The final grade for the course is obtained by taking into account the scores of the CAAs and a final face-to-face exam. In the next section, we detail the learning method used to develop skills in the analysis and synthesis of digital circuits.

3.2 Learning method

The *Computer Fundamentals* course offers several types of teaching materials. There is a textbook material [43] where the theoretical aspects are introduced. All the concepts are presented together with several practical examples to illustrate their application. The traditional learning method based on textbooks is useful during the initial stage of the learning process: before practicing, students need to learn the theoretical background.

Related to the practice of these skills, each unit of the course has a large number of exercises with a proposed solution. Note that the continuous assessment activities and the final exam are also based on problem-solving. Thus, the student must practice the concepts of the course by solving problems.

In the number representation unit, students learn by repetition. There is a unique solution and several techniques that can be applied depending on the proposed exercise. A student acquires the techniques from the textbooks and she understands and comprehends them by applying them to a collection of exercises.

Meanwhile, in the next units where the analysis and synthesis of digital circuits are presented, the tool VerilUOC [12] is used. In VerilUOC, students design small combinational and sequential circuits using the GUI interface Logisim [36] and the tool helps them to automatically validate the design towards the solution provided by the instructors. However, this tool is not focused on learning function minimization or circuit analysis. VerilUOC only provides a circuit analysis feature based on simulation by switching the value of the input signals. Therefore, additional tools have to be provided.

Thus, we have developed the support tool *KeMAP* to practice function minimization using the Karnaugh Map methodology. An editor has been designed to apply all the steps in the function minimization process, that is, to fill up the truth table from the problem statement, to build the Karnaugh Map, to minimize it and to express the minimized Boolean expression.

Additionally, another tool called *VerilChart* has been designed to support circuit analysis. In this case, students learn to analyze the propagation of the input signals through the circuit (or even a state machine) by means of a timing diagram. Given that this is an introductory course, the analysis of the digital circuits and state machines assumes zero delay on components. This simplification allows students to focus on the operation of combinational

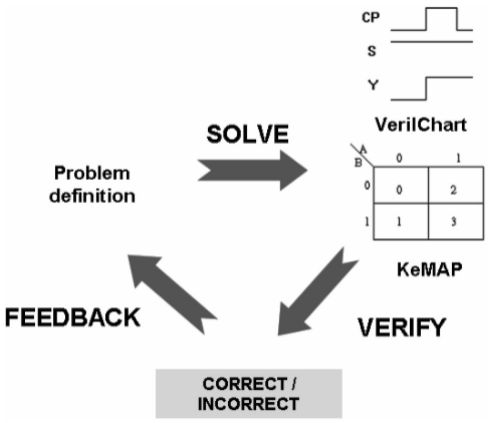

Fig. 2. Trial-and-error learning method.

and sequential components, leaving low-level electrical issues related to delays for more advanced courses within the Bachelor.

Similar to circuit design validation on VerilUOC, both tools have a verification module to check the correctness of the student submissions. The verification tool has been developed to help the students learn by a trial-and-error method (See Figure 2). Students receive an immediate feedback when solving an exercise and this feedback helps them to identify the errors and to fix them.

Instructors can manage the available exercises at any time and they can schedule the exercises depending on the current unit. Moreover, the level of detail of the feedback can be preconfigured by the teacher. In the initial exercises, the detailed feedback is enabled to help students to detect their errors. More advanced exercises or assessment exercises only return feedback related to the correctness of the proposed solution (e.g., correct/incorrect). The self-assessment results are stored and the teachers can review them to analyze the individual and global progression, and to give some personalized feedback to individual students in case of conceptual errors.

The next sections focus on the conceptual and technical description of the developed tools. We consider this information relevant to show that high-quality self-assessment tools can be designed with open source software. Additionally, this information provides detailed instructions to replicate the experiment. First, the *KeMAP* tool is introduced. Next, *VerilChart* is described.

4. KeMAP tool

4.1 Preliminaries

In this section, we define several theoretical concepts on Boolean function minimization which are used in the next sections.

A Boolean function describes the behavior of an output signal of a circuit in terms of the input signals. Each input is identified by one variable and its value can be either 0, 1 or ''don't care'' (i.e., not specified, where the input can take any value, either 0 or 1). A circuit with multiple outputs represents each output as one Boolean function. A *literal* is a variable or its complement. The conjunction (or product) of a set of literals is called a *cube*. A cube is called a *minterm* when the number of literals of the cube corresponds to the number of variables of the function. A *completely specified function* (CSF) has the outputs of all minterms specified by 0 or 1. On the contrary, an *incompletely specified function* (ISF) has some minterms unspecified, that is, they have a ''don't care'' value.

Although there are multiple ways and formats to represent a function, we focus on truth tables, sum-of-products, binary decision diagram (BDD) representations, and programmable logic array (PLA) format.

A truth table is an enumeration of all minterms with the respective value of the output. A sum-of-products (also called a *cover*) is defined as a disjunction (or sum) of cubes. A BDD is a data structure used to encode Boolean functions based on an acyclic graph representation. A variant of a BDD is an ROBDD (Reduced Ordered BDD), a canonical form with a fixed variable ordering where two covers of the same function have the same representation. Finally, PLA is a format used to describe Programmable Logic Arrays where there is control information about the inputs and outputs and a description of the AND and OR planes of the PLA with one line per product term. The PLA format can be used to define a Boolean function and it is supported by state-of-the-art CAD (computer-aided design) applications.

The size (or complexity) of a function can be computed by counting the number of literals of a sum-of-products. Then, a sum-of-products is *minimal* if there is no other sum-of-products function with fewer product terms and fewer literals.

4.2 Specification of the designed tool

The learning process has been developed in order to cover all the steps of the minimization process, from creating the truth table to the specification of the expression in sum-of-products for CSF and ISF Boolean functions. The candidate tool has to consider all the steps needed to solve a Karnaugh Map:

1. *Truth table definition:* The student fills the truth table from the exercise definition.
2. *Karnaugh Map edition:* The student fills the Karnaugh Map based on the values of the truth table.
3. *Karnaugh Map minimization:* The Karnaugh Map is minimized based on the filled values of the previous step.
4. *Sum-of-products expression:* The expression in sum-of-products form is specified based on the Karnaugh Map minimization.

The goal of the verification process in this tool is twofold. On the one hand, each step needs to check if the Boolean function is equivalent to the solution. In case of error, this verification strategy helps to point out the step where an error (or errors) has been detected. Furthermore, besides correctness, the minimized function has to be checked for minimality in steps 3 and 4.

This verification flow can be implemented from scratch or it can take advantage of any application that can manipulate Boolean functions. The verifi-

cation process designed in this paper reuses an academic tool called SIS that is outlined in the following section.

4.3 Design of the verification process

The academic tool SIS [44] has been used to handle the verification tasks related to Boolean function minimization. SIS is a modular interactive tool for the synthesis and optimization of digital circuits. Many different circuit manipulation algorithms have been integrated into SIS, allowing the user to choose among a variety of minimization and optimization techniques.

SIS supports several formats to load Boolean functions, such as equation, BLIF or PLA. In this case, the PLA format has been used as the format of the problem definition and the command 'read_pla' is used to load the function into the tool. The instructor defines the problem statement using the SIS PLA format. This format is suitable since truth tables and "don't cares" can be specified straightforwardly.

SIS has a built-in implementation of ESPRESSO, which is the one chosen to perform the minimization. Although ESPRESSO is a heuristic method and a non-exact solution could be obtained, there is a variant implemented in SIS called ESPRESSO-EXACT [45] that can be invoked with the command 'ESPRESSO -D exact'. This algorithm reduces the minimization problem to a binate covering problem where the exact solution is obtained.

SIS also offers all the necessary commands to check whether the submission generated by the student is equivalent to the instructors' solution. In this case, the command 'verify' is used. This tool compares two Boolean functions by computing the ROBDD representation and taking into account the "don't cares". Note that an ROBDD representation of a function is a canonical representation and, therefore, two functions with the same ROBDD are functionally equivalent.

Statistical information about the Boolean function such as the number of cubes or the total number of literals can also be retrieved. This information, collected with the command 'print_stats', is needed to detect whether the solution of the student is minimal or not. For this purpose, we cannot rely solely on functional equivalence checking (i.e., the command 'verify'). Moreover, the existence of "don't cares" means that there may be several potential minimal solutions, so minimality cannot be checked using a direct comparison. Therefore, the minimal function is verified by checking the functional equivalence of the function and whether the number of product terms and literals is minimal.

Finally, SIS also supports a scripting language. A script is a set of commands executed in sequential order. This feature helps to create the program to verify whether the solution obtained by the student is correct. The generated script is shown in Fig. 3. The figure illustrates the commands executed for each step of the Karnaugh map minimization. From the student input, four functions in PLA format are generated: *user_true_table, user_edit_KM, user_minimized_KM*, and *user_minimized_expr* that correspond to truth table definition, Karnaugh Map edition, Karnaugh Map minimization and sum-of-products expression respectively. The problem definition is stored in another function labeled as *problem_def*. The requirements to validate the solution of the student successfully are detailed next:

- In the first and second step, the student submission should be functionally equivalent to the problem definition.
- For the third and fourth step, the minimized function of the problem definition is obtained using ESPRESSO and it should be functionally equivalent to the student submission.
- In these last steps, in addition to functional equivalence, the number of product terms and literals of the student submission should be minimal.

After the execution of the script, the log is parsed. Incorrect steps of the minimization process are

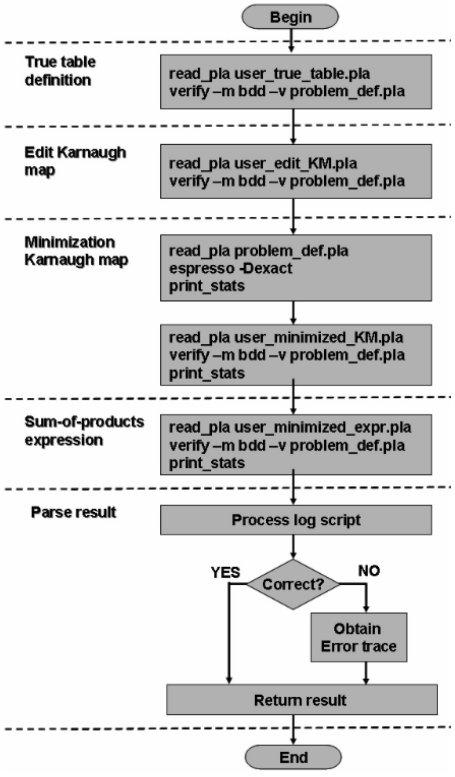

Fig. 3. Flowchart of the verification process in KeMAP.

detected to provide a personalized feedback of the verification process (described in Section 6.1) to the student.

Note that this verification process only considers sum-of-products minimization. In the specification of the tool, the product-of-sums was not taken into account. However, this process can be extended in a straightforward manner to support this type of minimization by following the next steps:

- Complementing the original function of the problem statement to transform the problem internally to a sum-of-products problem.
- Obtaining a complemented PLA from the Karnaugh Map edited by the student.
- Getting the Karnaugh Map minimization as it is since the problem definition has been already complemented and the selection of zeros internally refers to a selection of ones.
- Complementing the minimized product-of-sums expression of the student and applying De Morgan's law to the expression to transform the problem internally to a sum-of-products exercise.
- Applying the previous verification process.
- Finally, modifying the analysis of the error trace to return a personalized feedback based on a product-of-sums exercise.

5. VerilChart tool

5.1 Preliminaries

In this section, several concepts about circuit simulation and analysis used in the next sections are introduced.

A circuit can be simulated using different methods. The simplest method is to stimulate the circuit with different values for each input and inspect the value of the outputs. These stimulations are stored as a *trace* in order to record the behavior of the outputs for the set of input values.

In a real circuit, each block has a delay, i.e., the time needed to compute the output signal after a change in the inputs. In this case, we assume an ideal delay of zero since we are in an introductory course of digital systems and the students should focus on the analysis of digital circuits, not in the electrical issues that will be presented in higher-level courses.

5.2 Specification of the designed tool

The goal of *VerilChart* is to support the analysis of digital circuits. Note that a circuit can be represented in different ways: synthesized using digital components, as a state machine, using a truth table, a Boolean expression, etc. Thus, the tool should be able to check the solution of the student based on any valid representation of the circuit.

To this end, the tool needs to detect whether the timing diagram generated by the student is equivalent to the stimulation of the circuit with the same input values. Similar to the Karnaugh Map verification, the process can be implemented from scratch by generating a simulator system. However, we can use standard state-of-the-art simulation tools to perform the verification process.

5.3 Design of the verification process

In this case, the verification process has been developed using the hardware description language (HDL) Verilog [46] to describe the circuit and to simulate the inputs values provided by the timing diagram. An internal library with the specification of all digital components has been created. In this case, the switching delays of all components are set to zero to satisfy the requirements of the tool. The aim of this library was to be able to extend the verification process with delay information in the future. That is, specifying delay information on each problem statement could be easily added to this library during the verification process of the exercise. Then, the verification process is reduced to comparing the traces generated by the simulation system and the timing diagram of the student.

As we described in the previous section, the circuit may be specified in any representation in the problem statement. Internally, the circuit is always encoded using Verilog to unify the verification process. Note that using a standard HDL, such as Verilog, enables the possibility to handle synchronous and asynchronous signals in the timing diagram.

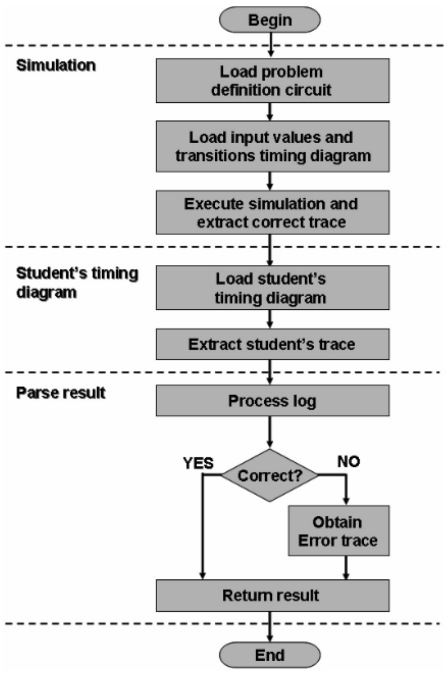

Fig. 4. Flowchart of the verification process in VerilChart.

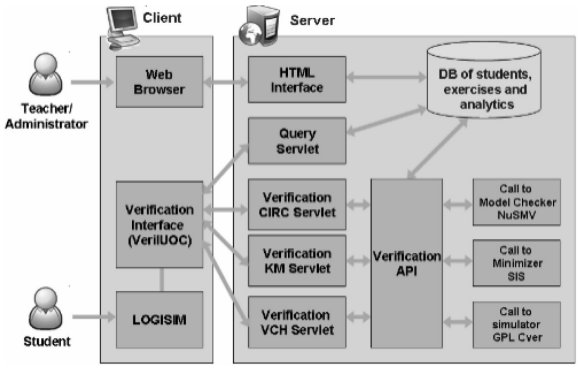

Fig. 5. Architecture of VerilUOC with the new verification services.

Figure 4 illustrates the verification flowchart. First, the values of input signals and transitions in the timing diagram are extracted. Next, a test is created from the problem definition circuit using the open source simulator *gplcver* [47]. This process produces the correct trace for each output for the timing diagram. Simultaneously, after a validation of the timing diagram of the student (i.e., the diagram should be completely filled) a second trace is obtained from the timing diagram filled by the student. Finally, both traces are compared to check equivalence. In case of error, the erroneous outputs and transitions are recorded to provide the personalized feedback (described in Section 6.2).

6. The VerilUOC framework

This section describes the GUI interface of each tool. Both tools have been integrated into the VerilUOC framework [12], a multi-platform Java GUI application for acquiring synthesis skills for digital circuits. The platform provides a graphical user interface based on the Logisim tool, which aims to facilitate the design and simulation of logic circuits. The main goal of this visual editor is helping students practice the synthesis part of digital systems. An automatic verification service is also provided to check whether the solution designed by the student is equivalent to the solution supplied by the instructor. In case of different behavior, the verification service automatically produces a meaningful feedback that points out the error.

The platform has been designed to hide high-level aspects such as the utilization of HDL or the verification process. Also, the student does not need to install any auxiliary application (i.e., only the Java platform). Consequently, the efficiency of the students increases since they focus on the learning objectives instead of the technical aspects of the platform.

The platform has been designed using a client-server model (see Fig. 5) where the server has two roles: (1) acting as the information repository system and (2) providing the verification service. Technically, the server is a web application server (Apache Tomcat) connected to a database (MySQL) where all the information related to the collection of exercises, the user profile of students and instructors is stored. Moreover, detailed analytics information of submitted exercises is also gathered for each student. This information is very useful for instructors to analyze the learning progress of the students.

The system accepts two types of user profiles:

- *Student profile:* Students access the system using a GUI desktop application populated with all the possible exercises, progress indicators and access to the verification service.
- *Instructor profile*: Although instructors may also access the GUI application to test exercises and create exercises, they have access to protected services using a web-based application. This interface is used to manage the students and the repository of exercises, to revise the exercises performed by the students and to analyze performance analytics.

The verification service has been implemented as a modular application where new verification procedures can be easily added. There is a verification API from which the verification processes are invoked. This API is called from a servlet service depending on the type of exercise to be verified. To support KeMAP and VerilChart new services have been added to the verification API and the corresponding servlets have been implemented to support the verification of Karnaugh Maps (KM Servlet in Fig. 5) and timing diagrams (VCH Servlet in Fig. 5).

6.1 KeMAP interface

The front-end of the VerilUOC platform is a complementary module to the Logisim application. In this module, the new verification tools have been added, providing a GUI. Figs. 6 and 7 illustrate the interface of KeMAP where the different parts of the tools are detailed next with the corresponding number in the figure.

1. The first part shown in Fig. 6 summarizes all possible exercises and the progression of the student, i.e., the number of attempts, the incorrect and correct ones. For each exercise, the student can see in the bottom part of the window the problem definition in the PDF Description tab, the verification result in the Verification tab and other aggregated informa-

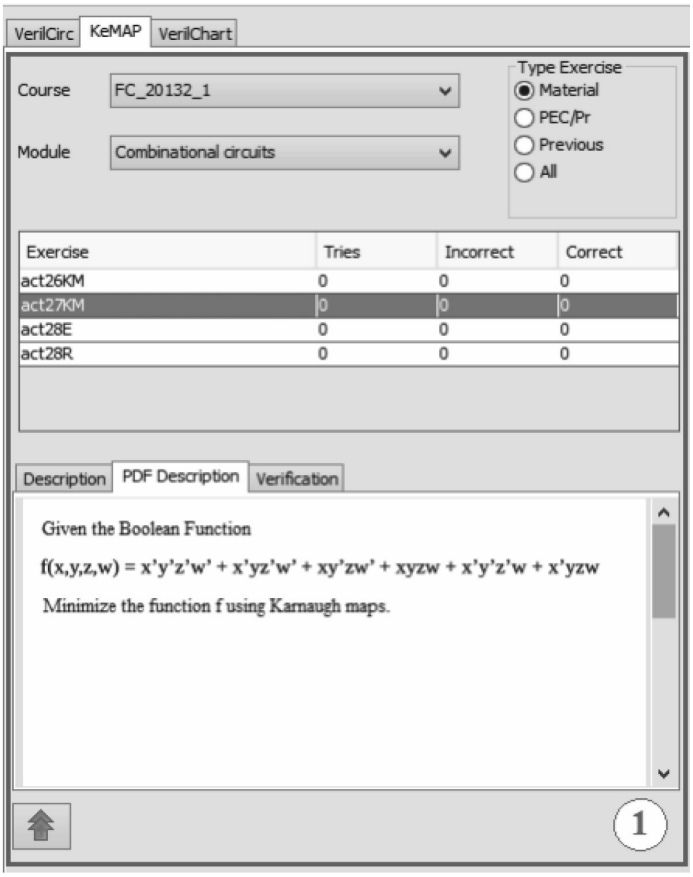

Fig. 6. Exercise selection in the KeMAP interface.

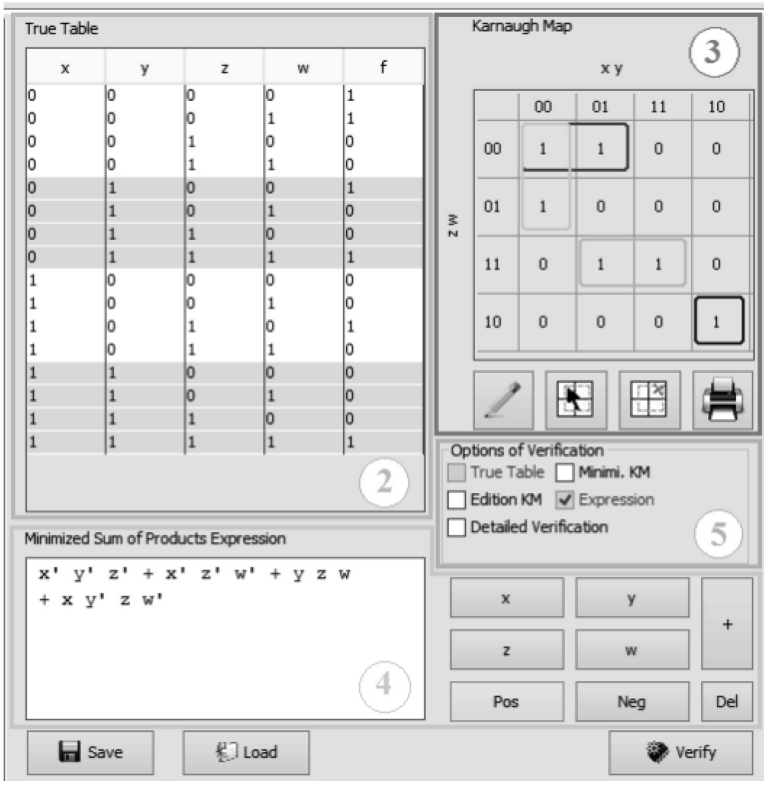

Fig. 7. KeMAP interface with (2) truth table, (3) Karnaugh map, (4) the minimized sum of products expression and (5) the level of difficulty setup.

tion such as the total number of students which have already solved the exercise.

2. The first step of the Karnaugh Map procedure is to fill up the truth table (see Fig. 7 from now on). The student fills the table from the problem definition by clicking on the output cell of the desired minterm. The clicking procedure switches on the ranging values 0, 1 and X (don't cares).
3. The Karnaugh Map is initially empty (all values to 0). The student should fill the values based on the truth table and, subsequently, perform the group selection. The value selection procedure uses the same interface as the definition of the truth table. First, the edition mode is enabled by selecting the pencil button. Then, the possible values 0, 1 and X can be chosen by clicking on each cell of the map. Next, the buttons "add group selection" or "remove group selection" are used to perform group selections. Here, the selection is made by clicking on a cell and dragging the mouse pointer to another cell. The produced rectangle from the origin to the ending cell is the group selection. The interface also has a "print" feature to allow students to export a snapshot of the Karnaugh Map as an image.
4. Finally, the minimized Boolean expression is written in this part of the interface. Although the Boolean expression can be written directly in the text box, the different buttons guide students while writing the expression, emulating a calculator interface.
5. As we described previously, each exercise has some verification hints available. There is one option for each step of the minimization process. When a hint for a step is selected, the verification process explicitly outputs the result of the verification for this step. In the case of error, this information helps identify the incorrect steps. There is also an extra hint called "Detailed verification". This advice is only recommended for novice students and it increases the level of detail of the verification by giving a minterm of the truth table where an error has been detected. All these hints can be configured by the instructor on the problem definition (the truth table option is disabled in Fig. 7). For example, all these options could be disabled for an assessment activity. Additionally, the enabled options can also be deselected by the student. For example, a student could disable the hint referred to the edition of Karnaugh Maps and to try to solve it by himself.

Figure 8 and Figure 9 show sample output messages to illustrate how they adapt to the verification settings. Let us assume an exercise with the Boolean function f(ig, xa, xb) = ig' xa + ig' xb where an error has been introduced in the minterm ig'xa'xb. Fig. 8 depicts the possible feedback mes-

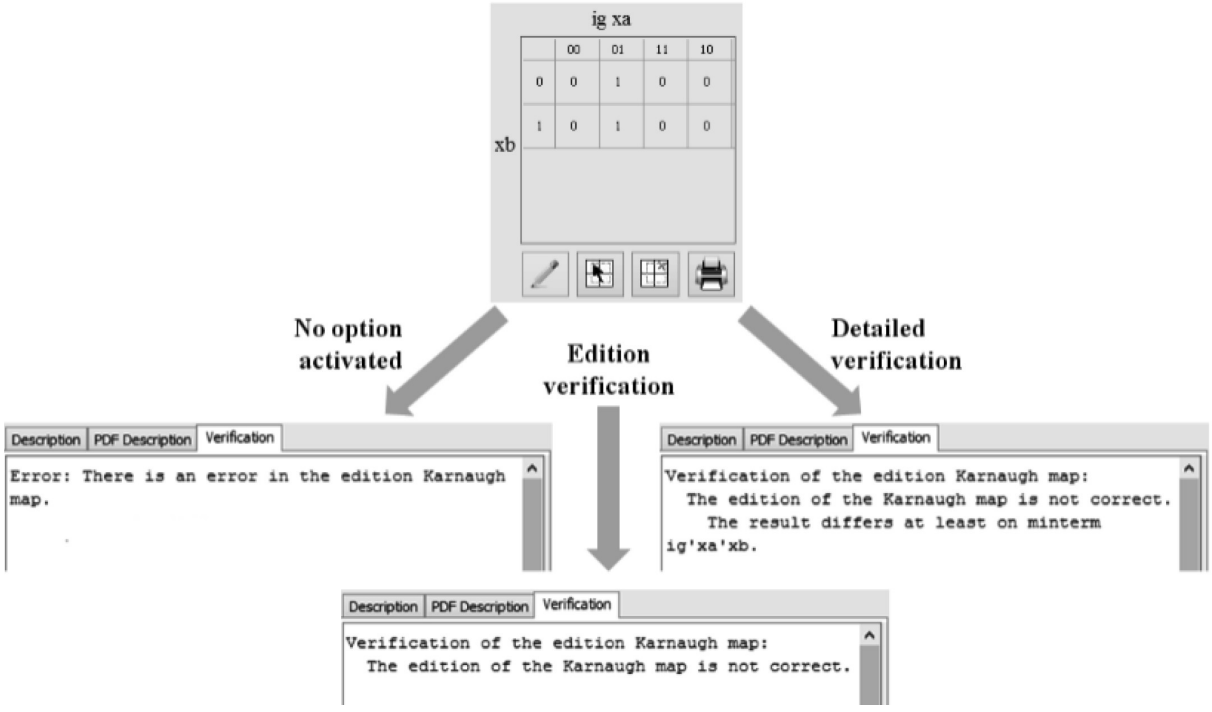

Fig. 8. Example of verification of a Karnaugh map edition using the KeMAP interface

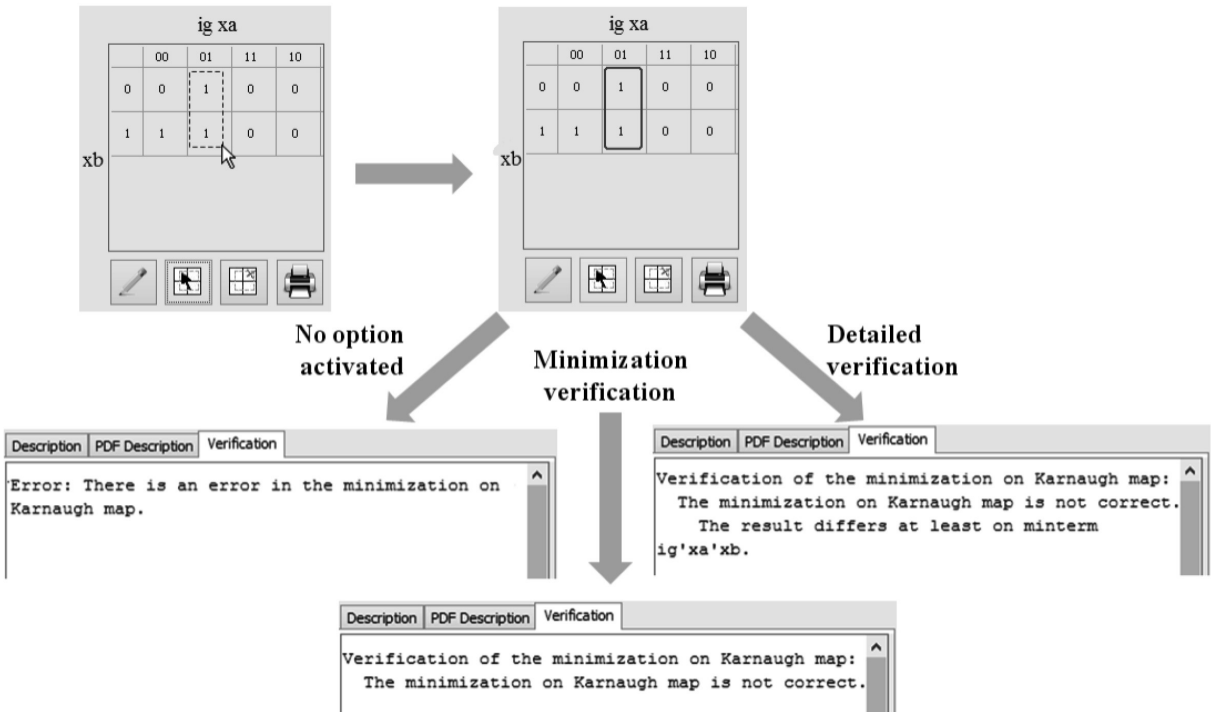

Fig. 9. Example of verification of a Karnaugh map minimization using the KeMAP interface.

sages based on the different settings. The first setting is the default one, without any enabled option. In this mode, KeMAP only prompts a message if there is an error, but no further information is provided.

When the edition mode verification is enabled, a message is always generated, even if the edition is correct. In this case, more specific messages are generated, i.e., to take into account "don't cares".

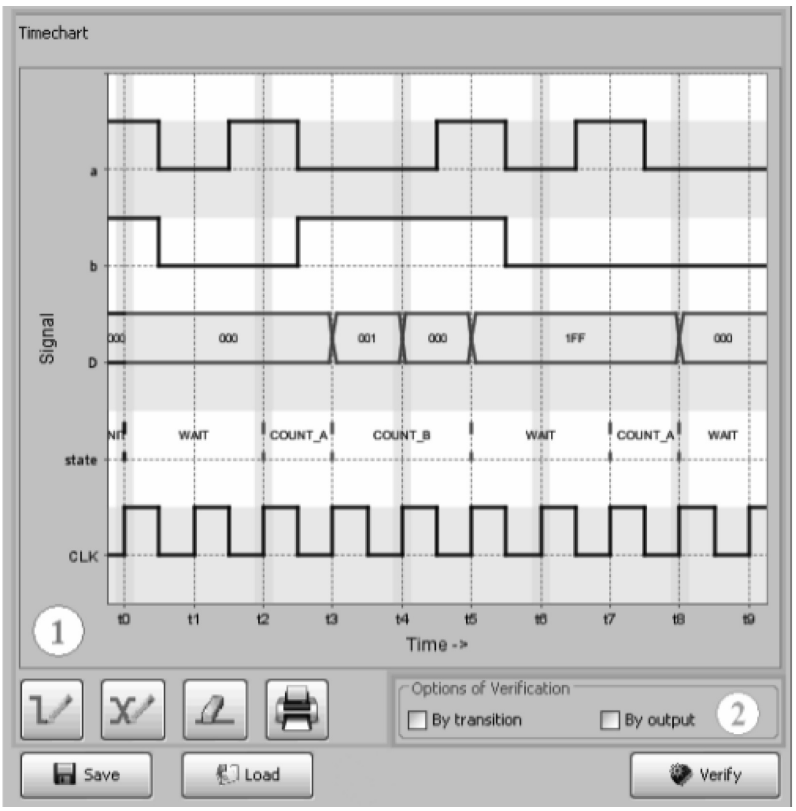

Fig. 10. VerilChart interface with (1) the timing diagram interface, and (2) the level of difficulty setup.

Finally, detailed verification enables the detection of a specific minterm where an error has been detected. On the other hand, Fig. 9 shows the verification process for the minimization of the Karnaugh Map. The mode of operation of the verification options works in the same way as in Fig. 8. Here, exceptions such as minimization errors, no minimal function or don't care inclusion are detected on the minimization verification without detailed information. Regarding Boolean expression minimization, it works like for the Karnaugh Map minimization, producing similar messages. For the sake of simplicity, this example is not illustrated.

6.2 *VerilChart interface*

The selection of an exercise in VerilChart uses an interface which is very similar to KeMAP's interface (shown in Fig. 6). For the sake of brevity, we omit this part. Fig. 10 shows VerilChart's interface for solving exercises and the type of feedback it can generate.

1. The main part of the tool is the creation of the timing diagram. The student should complete the timing diagram from the problem statement. This diagram supports different types of signals: 1-bit signals (represented by a single line in signals *a* and *b* in Fig. 10), bus signals (represented by slots with high and low lines in signal *D* in Fig. 10) and state values (represented by the name of the state in signal *state* in Fig. 10). Moreover, in the case of bus signals, different numerical representations can be used (binary, decimal, and hexadecimal) in order to further practice different number representations. These three types of signals allow the creation of timing diagrams for combinational and sequential circuits and state machines. Students edit the timing diagram by selecting the type of signal from the buttons below the timing diagram (1-bit signal and n-bit signal, which includes bus signals and state values). Then, a signal is created by clicking on the desired output. The selection could embrace a complete transition or a part of a transition since asynchronous input signals could generate reset values on sequential circuits. Similarly to KeMAP, a "delete signal" button and a "print" button are available.
2. The tool also has some hints available similarly to KeMAP. In this case, only two options can be selected: "by transition" showing in which transition a signal is incorrect; or "by output" showing an output where there is an error. Note that, when both options are selected, the outcome is similar to the "Detailed Verification" of KeMAP, since the output and transition where an error occurred is pointed out.

As described previously in Section 5.1, this tool assumes no delays in any component, since the goal of the tool is to help students to analyze simple circuits.

Figure 11 illustrates an example of the possible output messages based on the verification options.

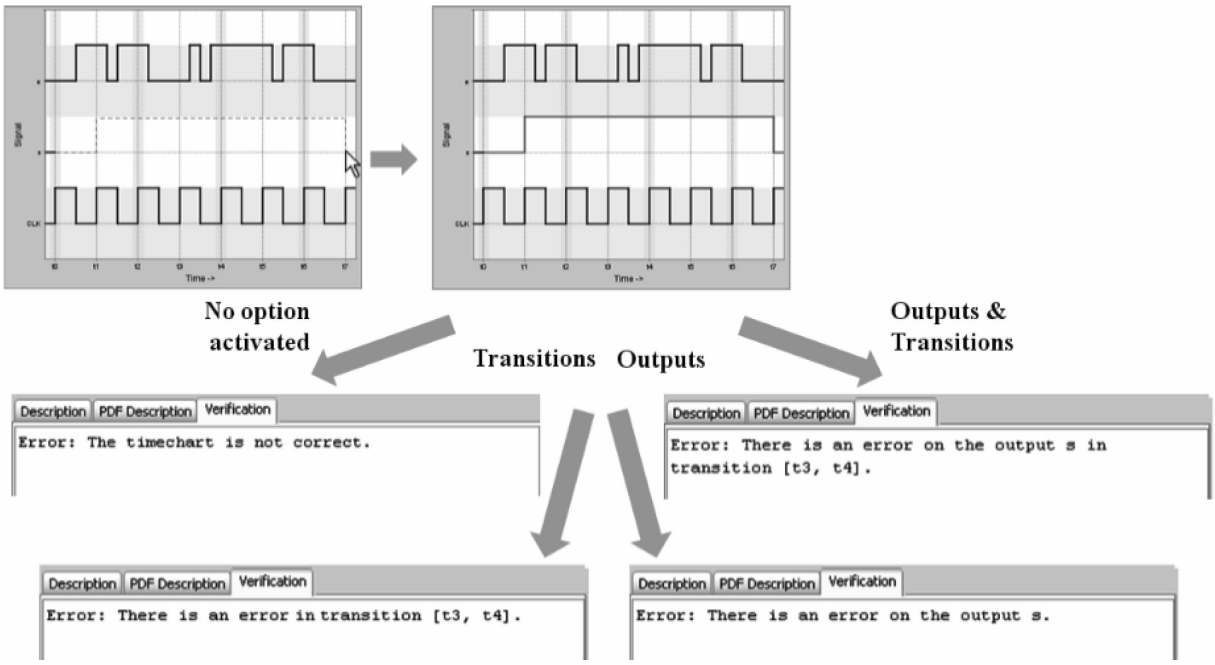

Fig. 11. Example of verification of a timing diagram using the VerilChart interface.

In this scenario, we consider an exercise asking students to complete a timing diagram of a simple d-latch without reset signal. An error has been introduced on transition t3-t4 to exemplify the possible error messages produced by the tool. In the default setting, VerilChart only produces correct/incorrect messages. If transition verification is enabled, a transition with an error is prompted. Note that this mode aims to show a transition where an error has been detected, but the output signal is not shown. Another option, output verification, analyses output signals individually. This flag helps to find an individual output where an error has been detected, but no further information is given. Finally, when both options are enabled, VerilChart runs as the detailed verification option on KeMAP, highlighting a transition of one output where an error has been detected. Recall that these options can be disabled by the instructor from the problem definition. Therefore, disabling some of these features produces exercises with different levels of difficulty.

6.3 Problem definition

For any tool, it is crucial to have an easy process to create new exercises. This section outlines how the exercises are set up for each tool.

As stated previously, KeMAP takes advantage of the PLA format to define the Boolean function. This format is similar to a truth table specification and the specification of a truth table of an exercise is straightforward. Then, the list of exercises is updated using the web-based interface of VerilUOC, where the corresponding hints of the exercise are defined.

On the other hand, VerilChart uses a special graphical interface in the desktop component only available for instructors to create the timing diagram. The instructor creates the diagram by defining the number of transitions, the inputs, the outputs, the width (number of bits) of the signals, the name of the states for state signals, and the value of the inputs through the transitions. This information is stored in a special file format (based on XML) and is uploaded through the same web-based interface of VerilUOC with the corresponding Verilog specification of the circuit and the available hints to create the exercise.

7. Results

This section summarizes the application of these new tools during the instruction of the Computer Fundamentals course and it gives insights to answer the research questions posed in the introduction. The students' performance is analyzed in the first subsection to give answers to the first and second research question. The opinion of the students has also been surveyed during two consecutive semesters to give answer to the third question.

7.1 Students' performance

The impact of the VerilUOC platform has been evaluated based on (a) the students' performance on the continuous assessment activities (CAA) where the tools have been used and (b) the final mark of the course. The objective of this analysis is to answer the first research question "RQ1: Has the utilization of the complete platform impacted on the student's performance?".

In order to perform this experiment, the continuous assessment activities and final exam had the same structure during the semesters under analysis:

- The CAA2 evaluates the combinational circuits. It includes one minimization exercise with Karnaugh maps, one exercise related to timing diagrams and three exercises related to logic circuit design.
- The CAA3 assesses the sequential content. It includes two exercises with timing diagrams, one exercise for designing a finite state machine and one exercise to materialize a finite state machine as a circuit.
- Finally, the exam has five exercises: one exercise about number representation, one minimization exercise with Karnaugh Maps, a combinational circuit design, a timing diagram of a sequential circuit and a design of a finite state machine.

Although the structure is equal among semesters, the content of the exercises differs. Thus, the difficulty may be affected. However, instructors assess the same contents through semesters. Related to tools, Semester 0 (used as the baseline) only used the verification of digital circuits while KeMAP and VerilChart were also available during Semester 1 and 2. Note that the utilization of the tools is not mandatory for students: they decide whether the utilization of the tools is useful for them.

Table 1 summarizes the students' performance results. We compared (a) two consecutive semesters (Semesters 1 and 2) where new tools were introduced with (b) the previous semester (Semester 0) where the validation of digital circuits was the only tool available. The table outlines the number of students who submitted each CAA and the students who passed each activity both in absolute (number of students) and relative (percentage) values. Note that students who did not submit the activities were also shown and not taken into account in calculating the total pass rate (i.e., total pass rate is the success rate from students who submitted the activity).

We can observe that the utilization of the Ver-

Table 1. Student performance analysis (using KeMAP/VerilChart vs not using them)

	CAA2			CAA3			Final Mark		
Submissions	**Sem. 0**	**Sem. 1**	**Sem. 2**	**Sem. 0**	**Sem. 1**	**Sem. 2**	**Sem. 0**	**Sem. 1**	**Sem. 2**
Sub. using tools	141	127	150	134	104	143	134	97	118
Sub. not using tools	156	34	41	115	18	21	70	23	13
Non-submitted	170	86	91	218	125	118	263	127	115
Total	467	247	282	467	247	282	467	247	282
Performance									
Absolute (# Students)									
Pass activity/using	134	108	128	120	90	128	108	84	105
Pass activity/not using	109	19	27	67	11	13	37	16	10
Total pass activity	243	127	155	187	101	141	145	100	115
Relative (Percent)									
Pass activity/using	95.0%	85.0%	85.3%	89.6%	86.5%	89.5%	80.6%	86.6%	89.0%
Pass activity/not using	69.9%	55.9%	65.9%	58.3%	61.1%	61.9%	52.9%	69.6%	76.9%
Total pass activity	81.8%	78.9%	81.2%	75.1%	82.8%	86.0%	71.1%	83.3%	87.8%

ilUOC platform improves the students' performance in the assessment activities in all semesters. Within the same semester, students using the tools have more than 20% higher pass rate than students who did not use them, and their global performance is also 4% higher. Moreover, compared to Semester 0, there is an increment in the number of students who passed the CAAs in Semester 1 and Semester 2. This increment is higher on CAA3.

When performance in the final mark is analyzed, similar results are obtained. Students who used the tool were most likely to pass the course compared to the rest of the students and there was an increment in the overall performance in the course.

However, an interesting result is observed here when Semester 0 is compared to the other ones: the pass rate is significantly lower on Semester 0 compared to the pass rate of CAAs. In other words, not all students that successfully pass the assessment activities passed the course. The problem here arose on exercises that students were not able to practice (Karnaugh Maps, timing diagrams and finite state machine). On the other semesters this difference decreased and most students that successfully passed the CAAs also passed the course. Thus, we can infer that the new tools supplied students with the required instruments to learn these concepts and pass the course.

After analyzing the observed data, we tested whether passing the activities CAA2, CAA3, and the course and using the tools for each semester were independent variables. We assumed as null hypothesis that using the tools has no impact in passing the activities or the course. It is interesting to note that after applying the Fisher's exact test (one-tailed) for statistical significance, we observe p-value < 0.02 in activities CAA2 and CAA3 for all semesters except in the final mark for Semester 1 and Semester 2 (0.054 in Semester 1 and 1.00 in Semester 2). Thus, we can reject the null hypothesis for the CAAs but not for the final mark. We further investigated by analyzing the impact of the VerilUOC platform on the students' grade.

This information is summarized in Table 2. The mean and the standard deviation is shown for each activity / semester for students who used and did not use the tool. When scores from Semester 0 are compared with the rest, we observe how the mean increases significantly on CAA2 and Final mark up to 1 point. However, there is no such increment on CAA3 (0.2 on Semester 2). When students are compared within each activity, we can clearly see how the average of the mark is higher on student's who used the tools. However, let us analyze its statistical significance. We used the unpaired two-samples Wilcoxon test due to the non-normal distribution of the scores. Here, we assume as null hypothesis that the scores are worse or equal when

Table 2. Student grades comparison based on the mean and the standard deviation

	CAA2			CAA3			Final Mark		
Submissions	**Sem. 0**	**Sem. 1**	**Sem. 2**	**Sem. 0**	**Sem. 1**	**Sem. 2**	**Sem. 0**	**Sem. 1**	**Sem. 2**
Using tools	6.5 (2.3)	7.5 (1.8)	7.6 (1.7)	7.5 (1.6)	7.5 (1.7)	7.7 (1.6)	6.5 (2.3)	7.4 (2.2)	7.2 (2.2)
Not Using tools	4.8 (2.1)	5.2 (2.0)	5.6 (2.0)	5.6 (2.0)	5.2 (1.9)	6.0 (2.3)	4.8 (2.1)	5.8 (2.4)	5.3 (3.1)
p-value (unpaired two-samples Wilcoxon test)	$4.21e^{-10}$	$1.64e^{-5}$	$3.50e^{-6}$	$2.55e^{-10}$	$1.22e^{-4}$	0.00023	$1.20e^{-4}$	0.00032	0.02684

the tools are used. Here, we observe p-value < 0.001 except for the final mark on Semester 2 that is less than 0.05. Thus, we can reject the null hypothesis and conclude that the utilization of the tool has a statistical significance on the student's performance in terms of scores distribution.

The previous tables show the impact of the complete platform but there are no insights about the impact of individuals tools on passing the course. To answer to the second research question: "RQ2. How has the utilization of the different tools impacted on the student's performance?", Fig. 12 shows boxplots depicting the variation of the final mark based on which tools were used by the students. Boxplots also show the mean and the number of students that are represented in the x-axis in parenthesis. Few students used the tools individually so we cannot conclude based on the small samples on these cases that the tools individually helped to pass the course. Nevertheless, we can observe in both semesters (Semester 1 and Semester 2) that most of the student used the combination of the three tools and they mostly passed the course. Only 4 students on each semester failed on passing the course. Here, we can argue that there is a significant impact on the utilization of the complete platform on passing the course.

However, we are aware that this increment cannot be attributable solely to the utilization of the tools, since there are other variables that could affect this result, such as the complexity of the exercises, changes in the instructor team, better performing and/or motivated students, etc. Moreover, the comparison of the student performance should be performed within each semester and not among semesters (i.e., different groups with/without the tools). We could not make this distinction due to academic constraints and, therefore, the comparison is done before/after introducing the tools. Nevertheless, some of the aforementioned factors can be more critical to the student performance that any new academic resource.

In any case, we can observe that a higher percentage of students used the tools in the second semester. Our interpretation is that after the first semester of the introduction of the tools, students had a positive perception of the tools as new resources to learn the concepts of the course and to help to pass the assessment activities. Another significant finding is the effectiveness of combining different tools. In Semester 0, VerilUOC was used with only the circuit design tool. The integration of the three tools in a unique platform helped students to improve their performance and be able to pass the assessment activities. In other words, the complete platform improves performance and engagement on the course.

A threat to validity is the selection bias, since the utilization of the tools is optional and motivated students (who tend to have better academic performance) are likely to use the tools. However, we can observe that the number of students who did not submit the CAAs decreased in both semesters. This observation hints that the new tools increased the engagement of students and improved the drop-out rate since the number of motivated students grew.

7.2 *Opinion survey*

This evaluation is based on the opinion of students and it aims to answer the third research question: "RQ3. Do the students have a positive perception of the tools?" Table 3 summarizes the results of a student survey performed after each semester. Since it was optional, few students completed the questionnaire (23% on Semester 1 and 15% on Semester 2). In particular, students who dropped out during the course did not answer the questionnaire (50% on Semester 1 and 40% on Semester 2). Even though this is a small sample which is not representative (margin of error larger than 10%) so

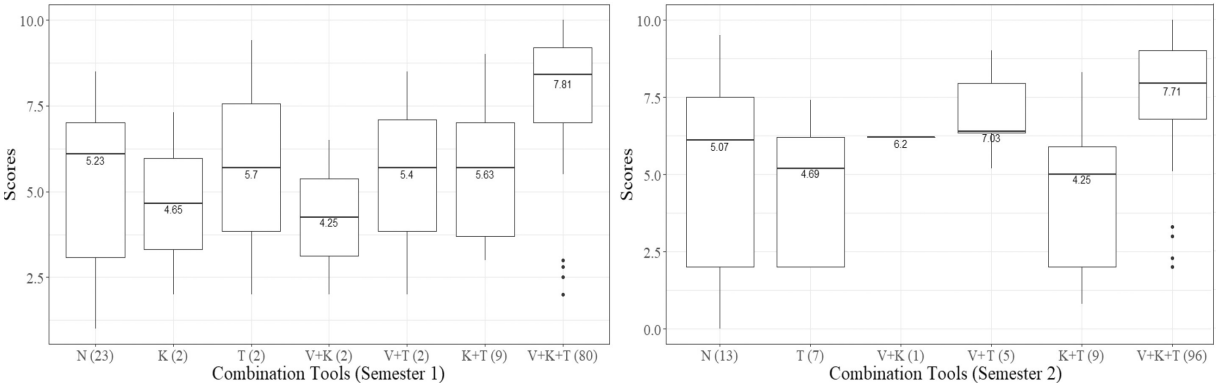

Fig. 12. Final score distribution based on the used tools during the course where the combinations are described as N = no tools, V = circuit verification, K = KeMAP and T = VerilChart.

Table 3. Results of the student survey

	KeMAP				VerilChart			
Statistics	**Semester 1**		**Semester 2**		**Semester 1**		**Semester 2**	
Number of students	247		282		247		282	
Number of respondents	58	23%	43	15%	58	23%	43	15%
Number of students used tool	54	93%	40	93%	53	91%	41	95%
Usage issues								
No issues	35	65%	32	80%	29	55%	36	88%
Few issues	17	31%	8	20%	21	40%	5	12%
Many issues	2	4%	0	0%	3	6%	0	0%
Student perception (1 to 5 scale, higher is better)								
The tool is easy to use	4.02		4.35		3.79		4.04	
The tool helped me to solve the CAA	4.38		4.80		4.39		4.63	
Overall score	4.44		4.75		4.24		4.46	

it is hard to draw conclusions, it provides insights related to the opinion of students who used the tools.

Most respondents used both tools (more than 90%). Regarding problems associated with the utilization of the tools, there were not many issues and many of them were related to students who were not able to understand the feedback of the tool. The reason was that detailed feedback was disabled by instructors on assessment exercises (while it was partially enabled on non-assessment activities), since the objective is that a student should try to find the correct answer without any help once he has gone through the initial training exercises. In this case, the output of the tools is binary (correct/incorrect). We assume that the problem arose for this reason since some students tried to solve assessment activities before trying the training exercises where detailed feedback could be used. We observed on the web-based analytics application that, after this first attempt, some students accessed previous exercises to better learn how to solve them. Moreover, we can observe that the number of issues decreased in the second semester. This could be due to an increased emphasis from the instructors about the need to practice similar exercises using the tools before solving the assessment exercises.

Finally, the tools were evaluated using a Likert scale (scale from 1 to 5, higher is better). Students agree that both tools were easy to use and user-friendly and they helped them solve exercises in the CAAs. Additionally, the overall score was very high, 4.44 and 4.24 on KeMAP and VerilChart respectively on Semester 1 and even higher on Semester 2 (4.75 and 4.46). Therefore, this shows a high degree of acceptance of the tools among students.

8. Conclusions and future work

In this paper, we have presented two tools to help students learn skills related to digital circuit design. Specifically, the tools consider Boolean function minimization based on Karnaugh Maps and analysis of digital circuits using timing diagrams. Both tools have been integrated into an intelligent tutoring system called VerilUOC. The integration improves the performance and engagement of students significantly.

The paper describes how each tool has been designed, from the GUI interface to the background process to verify the exercises automatically. We have also shown how the verification flow can be implemented using open source state-of-the-art tools.

The experimental results show a great impact on the assessment performance when the tools are used and a high level of acceptance from students who used the tools. Thus, we consider that a tool with a self-assessment features that provide feedback positively helps students to acquire the core concepts of function minimization and circuit analysis in digital systems. This type of tools combined with a problem-solving learning promotes the acquisition of these skills.

As future work, we are interested in extending the validation of the experiment: Can these tools be used in more advanced digital systems courses? In those courses, more advanced concepts will appear related to low-level electronic issues, such that circuit analysis with non-zero delays, or complex designs with the analysis of the quality with the VerilCIRC tool or the verification of HDL specifications.

Additionally, we are interested in the integration of learning analytics in the administration interface to improve the support to educators. These new features will assist in two areas: the early detection of students with learning problems (according to their usage pattern when solving exercises); and the identification of areas of improvement within the course (e.g., activities where many students tend to fail and may require additional educational

resources or changes in the instructional process). Additionally, exercise recommendation can be also explored by providing different adaptive learning paths depending on the progression of the student.

Acknowledgments—This work was supported by the eLearn Center from the Universitat Oberta de Catalunya through the project "LIS: Learning Intelligent System".

References

1. ACM/IEEE-CS Joint Task Force on Computing Curricula. 2016, *Computer Science Curricula 2016*, ACM Press and IEEE Computer Society Press.
2. M. Rashid, System Level Approach for Computer Engineering Education, *International Journal of Engineering Education*, **31**(1–A), 2015, pp. 141–153.
3. D. J. Russomanno and R. D. Bonnell, A pedagogical approach to database design via Karnaugh maps, *IEEE Transactions on Education*, **42**(4), 1999, pp. 261–270.
4. Y. S. Zhang, Determining All Candidate Keys Based on Karnaugh Map, 2009 International Conference on Information Management, Innovation Management and Industrial Engineering, **4**, 2009, pp. 226–229.
5. K. N. Plataniotis and S. Stergiopoulos, A Karnaugh map {2, 2} secret sharing scheme for color images. In *Proceedings of the 16th international conference on Digital Signal Processing (DSP'09)*, 2009, pp. 1141–1146.
6. N. Ben Neji, and A. Bouhoula and M. Kimura, Enabling flexible packet filtering through the K-map priority elimination technique, *2011 IEEE 36th Conference on Local Computer Networks (LCN)*, Oct. 2011, pp. 1–8.
7. M. Makys and S. Kozak, Effective Method for Design of Traffic Lights Control, In *Proceedings of the 18th IFAC World Congress*, 2011, pp. 14934–14939.
8. M. Yang, Y. Li, L. Zeng, D. Jin and L. Su, Karnaugh-map like online embedding algorithm of wireless virtualization, *2012 15th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, 2012, pp. 594–598.
9. C. Hundhausen, P. Agarwal, R. Zollars and A. Carter, The Design and Experimental Evaluation of a Scaffolded Software Environment to Improve Engineering Students' Disciplinary Problem-Solving Skills, *Journal of Engineering Education*, **100**(3), 2011, pp. 574–603.
10. A. Yadav, D. Subedi, M. A. Lundeberg and C. F. Bunting, Problem-based Learning: Influence on Students' Learning in an Electrical Engineering Course, *Journal of Engineering Education*, **100**(2), 2011, pp. 253–280.
11. S. Sánchez, M. Megías and J. Prieto Blázquez, SiMR: A simulator for learning computer architecture, *International Journal of Engineering Education*, **27**(2), 2011, pp. 238–247.
12. D. Baneres, R. Clarisó, J. Jorba, M. Serra, Experiences in Digital Circuit Design Courses: A Self-Study Platform for Learning Support, *IEEE Transactions on Learning Technologies*, **7**(3), 2014, pp. 1–15.
13. D. Baneres, Towards an Analytical Framework to Enhance Teaching Support in Digital Systems Design Course. *The 9th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2015)*, 2015 pp. 148–155.
14. D. Baneres, Principles for an Effort-Aware System, *Advances on P2P, Parallel, Grid, Cloud and Internet Computing: Proceedings of the 11th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC–2017)*, 2017, pp. 576–585.
15. D. Baneres and J. Saíz, Intelligent Tutoring System for Learning Digital Systems on MOOC Environments. *The 10th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2016)*, 2016, pp. 95–102.
16. M. Karnaugh, The Map Method for Synthesis of Combinational Logic Circuits. *Transactions of the American Institute of Electrical Engineers part I*, **72**(9), 1953, pp. 93–599.
17. W. V. Quine, The Problem of Simplifying Truth Functions, *The American Mathematical Monthly*, **59**(8), 1952.
18. W. V. Quine, A Way to Simplify Truth Functions, *The American Mathematical Monthly*, **62**(9), 1955.
19. E. J. McCluskey, Minimization of Boolean Functions, *The Bell System Techical Journal*, Nov. 1956.
20. R. K. Brayton, G. D. Hachtel, C. T. McMullen and A. L. Sangiovanni-Vincentelli. Logic Minimization Algorithms for VLSI Synthesis, The Kluwer International Series, *Engineering and Computer Science*, **2**, Springer US, 1984.
21. C. Hacker and R. Sitte, Interactive teaching of elementary digital logic design with WinLogiLab, *IEEE Transactions on Education*, **47**(2), 2004, pp. 196–203, May.
22. C. E. Klock, F. R. Schneider, M. V. N. Gomes, D. S. Moura, R. P. Ribas, A. I. Reis, KARMA: A Didactic Tool for Two-Level Logic Synthesis, *IEEE International Conference on Microelectronic Systems Education*, 2007, pp. 59–60.
23. A. A. Kassim, S. A. Kazi and S. Ranganath, A web-based intelligent learning environment for digital systems, *International Journal of Engineering Education*, **20**(1), 2004, pp. 13–23.
24. J. Garcıa, J. Sanz and B. Sotomayor, A new approach to educational software for logic analysis and design, *International Conference on Education (IADAT e2004)*, 2004.
25. M. Ayob, K. Chellappan and N. M. Ali, Intelligent tutoring tool for digital logic design course (ITDiL), TENCON 2001. *Proceedings of IEEE Region 10 International Conference on Electrical and Electronic Technology*, **2**, 2001, pp. 899–902.
26. Karnaugh Map Minimizer. http://k-map.sourceforge.net/, Accessed 15 May 2018.
27. KarnaughMap445. http://www.puz.com/sw/karnaugh/index.htm, Accessed 15 May 2018.
28. LogicAid. http://logicaid.software.informer.com/, Accessed 15 May 2018.
29. KVD: Karnaugh-Veitch-Diagram mobile app. http://play.google.com/store/apps/details?id=com.mhsoft.kvd, Accessed 15 May 2018.
30. Mini Karnaugh: https://play.google.com/store/apps/details?id=serban.stoenescu2, Accessed 15 May 2018.
31. Z. Stanisavljevic, V. Pavlovic, B. Nikolic and J. Djordjevic, SDLDS—System for Digital Logic Design and Simulation, *IEEE Transactions on Education*, **56**(2), 2013, pp. 235–245.
32. A. Shoufan, Z. Lu and S. A. Huss, A Web-Based Visualization and Animation Platform for Digital Logic Design, *IEEE Transactions on Learning Technologies*, **8**(2), 2015, pp. 225–239.
33. G. L. Herman, M. C. Loui, C. Zilles, Students' Misconceptions About Medium-Scale Integrated Circuits, *IEEE Transactions on Education*, **54**(4), 2011, pp. 637–645.
34. G. L. Herman, C. Zilles and M. C Loui, Flip-Flops in Students' Conceptions of State, *IEEE Transactions on Education*, **55**(1), 2012, pp. 88–98.
35. S. A. Edwards, Experiences teaching an FPGA-based embedded systems class, *SIGBED Rev.*, **2**(4), 2005, pp. 56–62.
36. C. Burch, Logisim, a graphical tool for designing and simulating logic circuits, http://www.cburch.com/logisim/, Accessed 15 May 2018.
37. A. Tetzl, Logicsim, logic simulator, http://www.tetzl.de/java_logic_simulator.html, Accessed 15 May 2018.
38. M. Damm, F. Bauer, G. Zucker and K. Waldschmidt, LogiFlash—Understanding Digital Technology An interactive flash-based simulation tool for digital circuits, ICL 2009, *Proceeding in the Interactive Conference on Computer Aided Learning*, 2009, pp. 167–171.
39. D. A. Poplawski, A pedagogically targeted logic design and simulation tool, *Proc. Workshop on Computer Architecture education*, 2007, pp. 1–7.
40. CEDAR, CEDAR logic simulator, http://sourceforge.net/projects/cedarlogic/, Accessed 15 May 2018.
41. C. Computing, LogicWorks 5 Interactive Software. Prentice Hall, 2004.
42. B. Nikolic, Z. Radivojevic, J. Djordjevic and V. Milutinovic, A survey and evaluation of simulators suitable for teaching courses in computer architecture and organization, *IEEE Transactions on Education*, **52**(4), 2009, pp. 449–458.
43. R. Costa, M. Peiron, Ll. Ribas, F. Sánchez and A. J. Velasco, Fundamentos de Computadores, Fundació per a la Uni-

versitat Oberta de Catalunya (FUOC). 2011. [http://ocw.uoc.edu/informatica-tecnologia-y-multimedia/fundamentos-de-computadores/materiales-1/] (in Spanish), Accessed 15 May 2018.
44. E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton and A. Sangiovanni-Vincentelli, *SIS: A system for sequential circuit synthesis*, U.C. Berkeley, Tech. Rep., 1992.
45. R. L. Rudell, A. Sangiovanni-Vincentelli, Multiple-Valued Minimization for PLA Optimization, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **6**(5), 1987, pp.727–750.
46. F. Vahid, Verilog for Digital Design. Wiley, 2007.
47. Verilog simulator gplcver, http://gplcver.sourceforge.net/, Accessed 15 May 2018.

David Baneres received a BSc in Computer Science (2001) and a PhD in Computer Science (2008) from the Universitat Politècnica de Catalunya, Spain. He is currently a full-time lecturer at Universitat Oberta de Catalunya. He has also been a part-time associate professor at Universitat Autònoma de Barcelona (2010–2012). His research interests include optimization techniques on logic synthesis domain, formal methods and innovative e-learning systems.

Robert Clarisó received a BSc in Computer Science (2000) and a PhD in Computer Science (2005) from the Universitat Politècnica de Catalunya, Spain. He is currently a full-time lecturer at Universitat Oberta de Catalunya. He has also been a part-time associate professor at Universitat Politècnica de Catalunya (2006) and Universitat Autònoma de Barcelona (2006–2011). His research interests include formal verification, software engineering and tools for e-learning.