

IEEE Transactions on Education, 56 (2), 235-245, 2013

0018-9359 © 2013 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

The final publication is available at IEEE via <http://dx.doi.org/10.1109/TE.2012.2211598>

SDLDS - System for Digital Logic Design and Simulation

Zarko Stanisavljevic, Vladimir Pavlovic, Bosko Nikolic, Member, IEEE, and Jovan Djordjevic

***Abstract*—This paper presents the basic features of a software system developed to support the teaching of digital logic, as well as the experience of using it in the Digital Logic course taught at the School of Electrical Engineering, University of Belgrade, Serbia. The system has been used for several years, both by students for self-learning and laboratory work, and by teachers to automate the assessment and verification of students' work. The system allows users to design and simulate a switching circuit. It also collects data on all student activities and transfers these to school's information system. Finally, the paper gives figures demonstrating the overall benefits of the system.**

This work was partially funded by the Ministry of Education and Science of the Republic of Serbia (III44009).

The authors are with the Department of Computer Engineering, School of Electrical Engineering, University of Belgrade (corresponding author phone: +38111-3218-392; fax: +38111-324-8681; e-mail addresses: zarko.stanisavljevic@etf.bg.ac.rs, bosko.nikolic@etf.bg.ac.rs, jovan.djordjevic@etf.bg.ac.rs).

***Index Terms*—Digital circuits, digital systems, engineering education, simulation software, visual system**

I. INTRODUCTION

The teaching of digital logic has a very important place in many Electrical Engineering degree programs. At the School of Electrical Engineering, University of Belgrade (SEE-UB), Serbia, a basic course, Digital Logic, covers switching theory and combinational and sequential switching circuits. This course is designed to prepare students in programs such as Telecommunications, Power Systems, Electronics, Automatic Control, Microelectronics, or Software and Computer Engineering to follow digital technologies-related courses they will meet later in their studies.

Nowadays, the use of various software systems to support the teaching process is common. This is especially true in engineering education, where it is very important for students to be able to observe, explore and manipulate device characteristics and behaviors. In the area of digital logic, this includes not only designing, simulating and testing digital systems, but also acquiring, analyzing and interpreting data, and whenever possible, using that data to correct or improve the design. This calls for a software system that facilitates self-learning, supports supervised practical work in the laboratory (based on computer simulations) and automates the assessment and verification of students' work. To this end, the authors developed the System for Digital Logic Design and Simulation (SDLDS); this paper describes the basic features of SDLDS, and the experience of using it in teaching a digital logic course.

The paper is organized as follows: Section II explains the motivation for developing the SDLDS system, Section III briefly describes SDLDS, Section IV presents its use,

Section V gives assessment data and Section VI draws conclusions.

II. MOTIVATION

Digital computers are an indispensable part of modern life, and digitalization has dramatically changed many areas of electrical engineering. A course covering the basics of digital logic is thus necessary for students in this area of study. This is particularly true for computer sciences and engineering students, since such a course provides the background knowledge necessary for courses they take further on in their studies. The importance of digital logic is addressed in the document Computer Engineering 2004 - Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering, issued as recommendations by the IEEE and ACM societies [1], which identifies digital logic as a core area. In line with that, a course in the basics of digital logic [2]-[4] was added to the SEE-UB curriculum.

The first part of the course covers the basics of switching theory, Boolean and switching algebra and switching functions. Students are shown that a switching function, given by a truth table, can be represented by Boolean expressions in the form of canonical sum-of-products (CSPF) and canonical product-of-sums (CPSF). They are also shown that minimizing CSPF and CPSF using Karnaugh maps yields simpler expressions in the form of minimal sum-of-products (MSPF) and minimal product-of-sums (MPSF). The second part of the course introduces the notion of switching circuits, and defines two types of switching circuits: combinational and sequential. It then presents AND, OR, XOR, NOT, NAND and NOR logical elements, and explains how these should be connected to create a combinational switching circuit. It then introduces D, T, RS and JK flip-flop memory elements and shows how memory and logical

elements should be connected to obtain a sequential switching circuit. The final segment of this part of the course shows how the behavior of a combinational or sequential circuit can be synthesized, based on its formal specification, and its appropriate structural scheme obtained. The third part of the course deals with standard combinational and sequential switching modules, which realize functions of some frequently-used switching circuits in digital system designing. Standard combinational switching modules are multiplexers, demultiplexers, coders, decoders, left, right and rotate shifters, incrementers, decrementers, adders, subtractors, arithmetic units, logic units, ALU units and comparators. Standard sequential switching modules are parallel registers, shift registers and counters. First, the functionality of modules with fewer input and output signals is given, and their synthesis with logical and memory elements is carried out. Then various possible ways are shown of interconnecting these modules to obtain structural schemes with a larger number of input and output signals.

Digital Logic is an one-semester (14 weeks) elective course for all first-year engineering students. It consists of two lecture classes, one problems class and two laboratory classes per week (45 minute classes). The lectures cover theoretical aspects, the problems class helps students in solving practical problems, and the laboratory classes give practical experience of the operation of switching circuits. Three-hour-long exams are administered in June, July, September and October, referred to as exam terms. The authors decided to develop a software system that would support the teaching activities in this Digital Logic course by facilitating self-learning, supporting practical work and automating the assessment and verification of student work.

The system envisaged would allow students, at any time, in a user-friendly, visual and

interactive environment, to study the various facets of the topics and concepts covered. For learning combinational and sequential switching circuit synthesis, the system should follow the formal synthesis procedure. The laboratory exercises were intended to familiarize students with both the analysis of predefined modules and the synthesis of complex modules using these predefined ones. Students should be able to manually draw structural schemes in the simulator, and verify their correctness by simulating their behavior for various signal patterns. The simulation modes available should allow a user to carry out interactive simulations, display their results, obtain signal values and display the contents of registers. The possibility for self-learning, with the compulsory work in the laboratory, would motivate students to participate actively in all phases of the course. Furthermore, the system should automate the assessment and verification of students' work, gather data on all students' activities and be able to transfer these into the School's information system.

While there are significant differences among the many simulators used for teaching digital logic [5], making classification difficult [6], these were analyzed for their suitability for the Digital Logic course. Some simulators allow users to design a digital system and then simulate it; the most representative of these are HASE, JHDL, ISE WebPACK, PSIM, Logisim, Virtual Vulcan, and e-EDU. HASE (Hierarchical Computer Architecture design and Simulation Environment) is a digital circuit design and simulation environment, which allows digital design implementation to be represented graphically at several different levels of abstraction [7]-[9]. The primary use of JHDL (a structurally-based Hardware Description Language implemented with Java) is for the design of digital circuits for implementation in field-programmable gate arrays (FPGAs), with

particular attention being paid to supporting the Xilinx series of chips [10]-[12]. ISE WebPACK [13], [14] is a tool for FPGA and CPLD design offering HDL synthesis and simulation, implementation, device fitting, and JTAG programming. PSIM (Processor SIMulator) display is a graphical environment where the internal components and their interconnections are displayed so that the user can easily understand the signal paths taken by data and control during the processor operation [15]. Logisim is used to design and simulate digital logic circuits for educational purposes [16]-[18]. Virtual Vulcan Digital Logic Simulator is a software simulation of digital, microcomputer and robotic systems [19], [20]. The e-EDU [21], [22] system is a Web-based learning environment with a set of hands-on tools that introduce fundamental elements of digital logic. The use of programmable logic devices (PLD) implies that students make their designs using both schematic capture and VHDL, test the design using functional and timing simulations, and finally download it onto the PLD board [23].

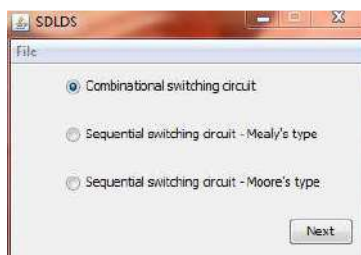
HASE and JHDL do not provide complete Web solutions for the simulation process. ISE Webpack, PSIM and Virtual Vulcan are available as desktop applications only. Logisim has no features with any Web-based modules or expansions, but its inclusion into a web-based system is possible. None of the systems analyzed, other than Logisim, support Karnaugh maps. The automation of the assessment and verification of student work is also not available, but might be achievable for open source simulators through additional modules. Course-specific services (e.g., personal assignments and progress, course materials, news) and general services (e.g., student data, courses and settings, e-Check-in, calendar) are available only with the e-EDU system.

While all these systems have been used successfully elsewhere, each was developed

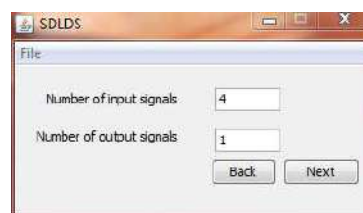
to meet very specific requirements, and this do not match SEE-UB requirements. The decision was therefore taken to develop the SDLDS system.

III. SDLDS SOFTWARE SYSTEM

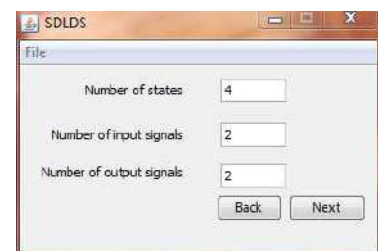
The SDLDS system allows a user to design and simulate a switching circuit. The design can be carried out in one of two ways. The first is that, based on the switching circuit formal specification, the system goes through the switching circuit synthesis, and gives its structural scheme. The second is that a user manually interconnects modules, available in the system, into a desired structural scheme. The behavior of either of these resulting structural schemes can be verified through simulation.



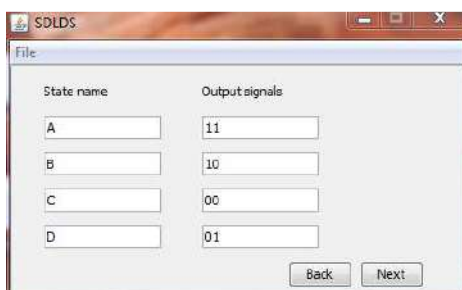
a)



b)



c)



d)



e)

Fig. 1. Screens for selecting the type of switching circuit and specifying its behavior

The design of a switching circuit, based on its formal specification, requires the user to first select the type of switching circuit, Fig. 1a, and then define its behavior.

The behavior of a combinational switching circuit is defined with a truth table, formed

in two steps. In the first step the SDLDS system, based on the number of input and output signals specified by the user, Fig. 1b, draws the truth table with as many rows and columns as there are combinations of input signals and output signals, respectively. In the second step the user fills the columns with the values of output signals for each combination of input signals. As an example, the truth table for the combinational switching circuit with four input signals, x_1 to x_4 , and one output signal, y , is shown in Fig. 2a. The output signal has value 1 for combinations of input signals 0001, 0101, 0110, 1001, 1010, 1111, value 0 for combinations of input signals 0000, 0010, 0111, 1000, 1011, 1100 and undefined value b for the remaining combinations of input signals. Based on the truth table, the system creates expressions in the CPSF and CSPF forms and draws their structural schemes, Fig. 2b and Fig. 2c, respectively. In addition, the system, based on Karnaugh maps, creates expressions in the MPSF and MSPF forms, Figs. 3a and c, respectively, and again draws their structural schemes, Fig. 3b and d, respectively. Its simulation is described below. In the case when the behavior of the combinational switching circuit can be presented with more than one expression, all of these are given.

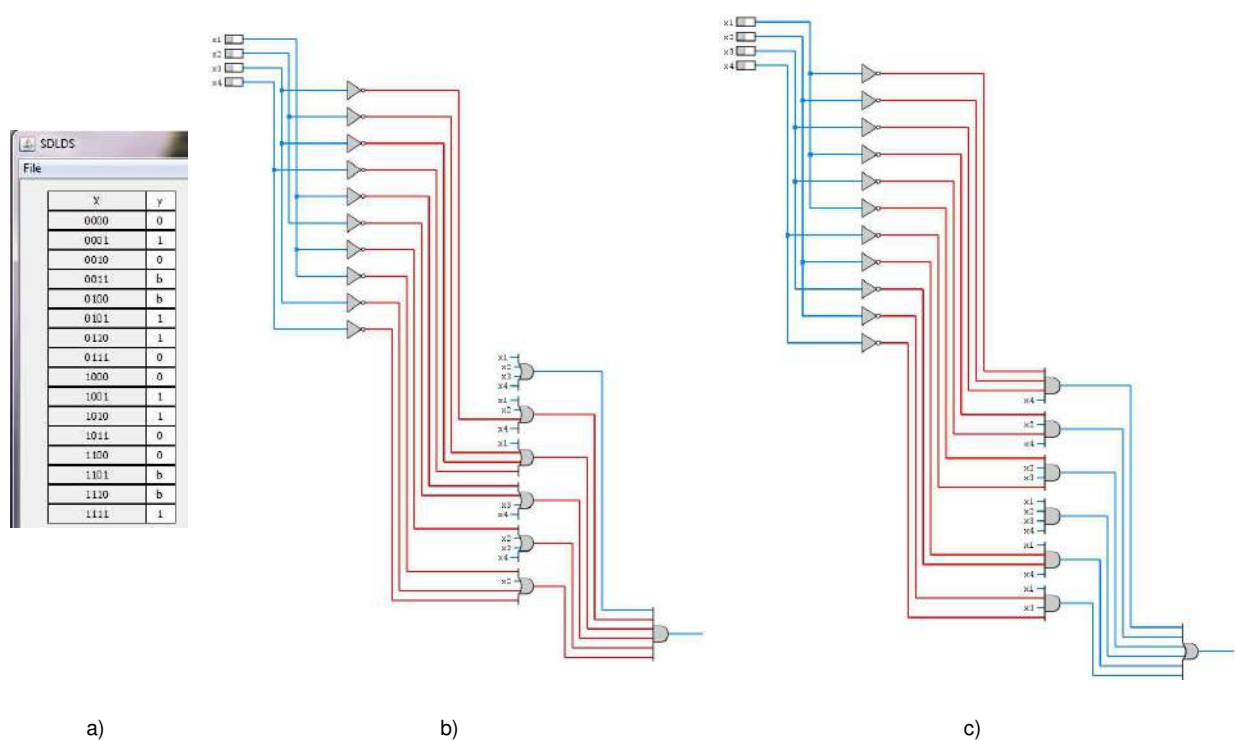
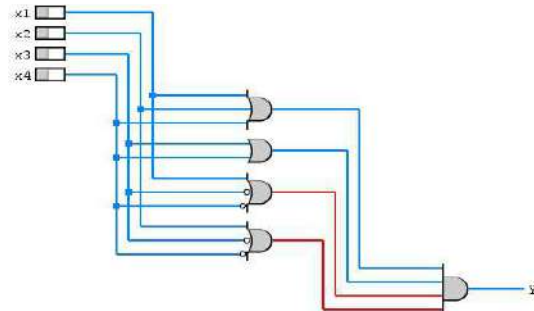
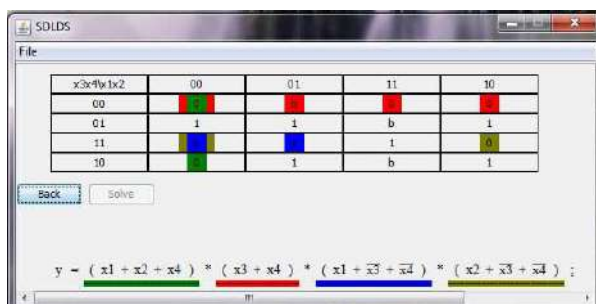
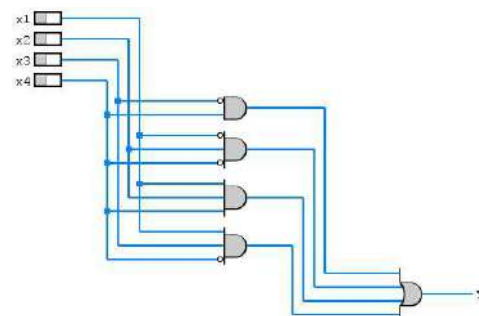
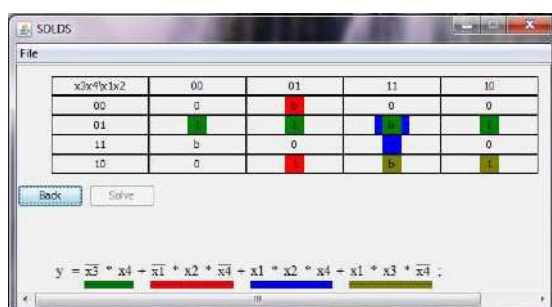


Fig. 2. Truth table and structural schemes of CPSF and CSPF forms of the combinational switching circuit

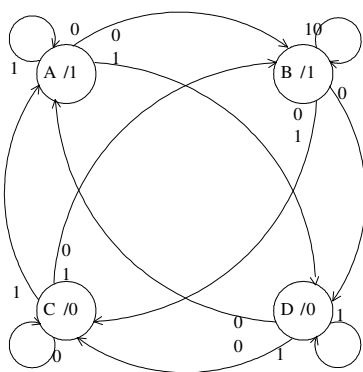


c)

d)

Fig. 3. Karnaugh maps and structural schemes of MSPF and MPSF forms of the combinational switching circuit

The behavior of a sequential switching circuit is defined with the state/output table, formed in two steps. In the first step the SDLDS system, based on the number of states and input and output signals, Fig. 1c, and state names and values of output signals, Fig. 1d, specified by the user, draws the state/output table. With the assumption that the Moore's type of sequential circuit is selected, Fig. 1a, the table contains as many rows and columns as there are states and combinations of input signals, respectively, plus a column with specified values of output signals, Fig. 1e. In the second step the user, using the state graph drawn on paper, Fig. 4a, fills the cells with the values of the next state for each combination of state and input signals. As an example, for the Moore's type of sequential switching circuit with two input signals x_1 and x_2 , four states A, B, C, D, and two output signals z_1 and z_2 , the state graph of which is given in Fig. 4a, the system draws the state/output table (Fig. 1e).



a)

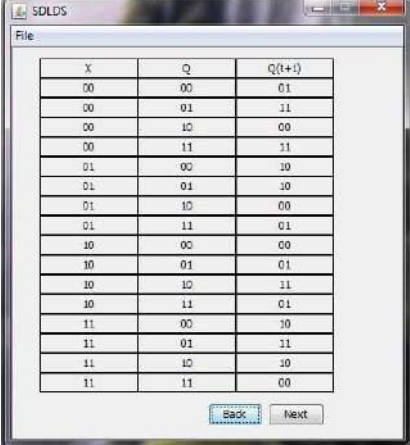
State name	Code value
A	00
B	01
C	11
D	10

b)

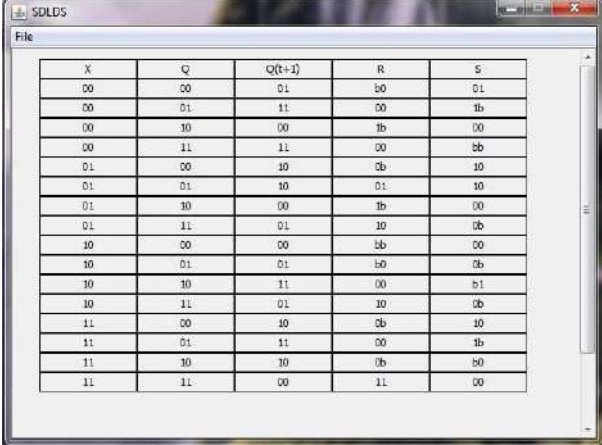
Q/x	00	01	10	11	z
00	01	10	00	10	11
01	11	10	01	11	10
10	00	00	11	10	01
11	11	01	01	00	00

c)

Fig. 4. The state graph, the state/output table, and the transition/output table for the sequential switching circuit



a)



b)

Fig. 5. The combinational state/output table and the excitation table for RS flip-flop for the sequential switching circuit

The rest of the synthesis is done in four steps. In the first step, the system, based on the state/output table, Fig. 1e, and user specified encoding of states, Fig. 4b, draws the transition/output table, Fig. 4c, and the combinational state/output table, Fig. 5a. In the second step, based on user-specified types of flip-flops, the system draws the excitation table, Fig. 5b. The third step represents the previously-described synthesis of the combinational switching circuit, which generates flip-flop excitation input signals, Fig. 6, and sequential switching circuit output signals. In the fourth step the system draws the structural scheme of the sequential switching circuit, Fig. 7.

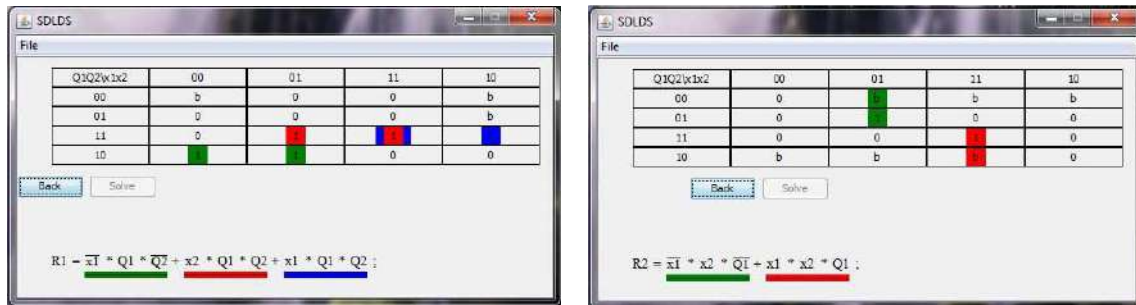


Fig. 6. Some of the screens for solving Karnaugh maps and obtaining final expressions in the form of MSPF for the sequential switching circuit

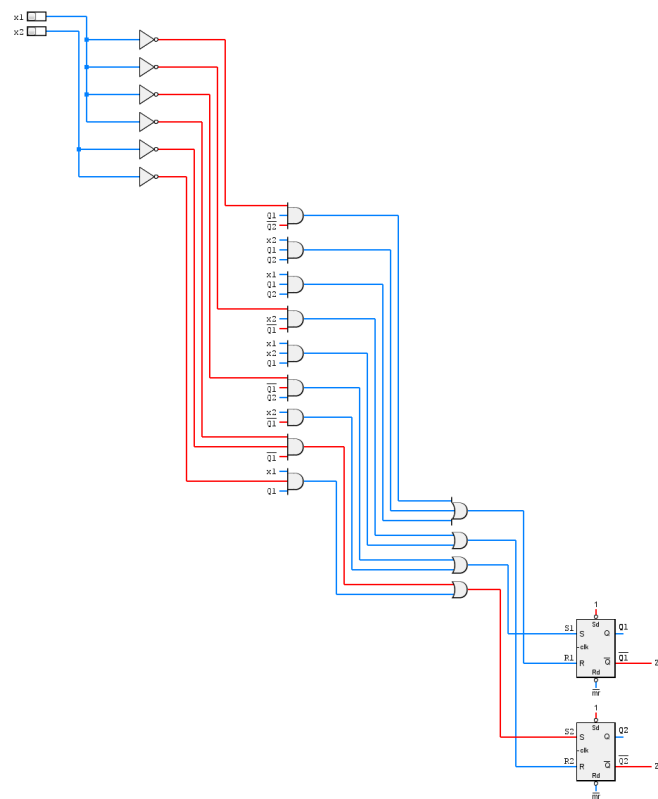


Fig. 7. Representation of the sequential switching circuit

The manual design of a switching circuit is carried out in the main window's Design Panel using available building modules, Fig. 8. Two types of modules in the *Component Box* section can be used to design a switching circuit: basic predefined modules and custom-made modules. The basic predefined modules are made available by the system and include logical and memory elements and standard combinational and

sequential modules as specified in Section II. The custom-made modules are created by users, by saving the structural schemes they designed. Both types of component can be used as parts of a more complex structural scheme. The structure of a switching circuit shown on the Design Panel is also given in the form of a hierarchical tree structure in the Tree View section.

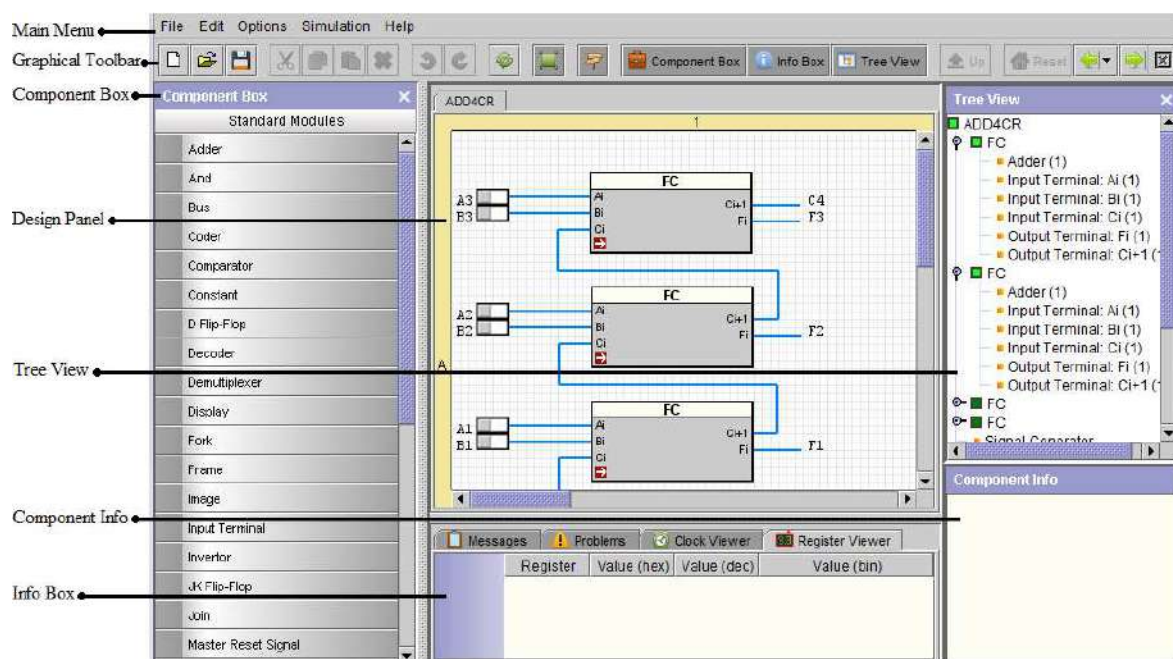


Fig. 8. Main window for the manual design of a switching circuit

Every component has input and/or output ports. When a component is placed on the Design Panel one connector is created for each port of the component. The user connects connectors of components placed on the Design Panel according to the structural scheme being drawn. The system refuses to perform any connection that is not allowed. For the proper functioning of sequential components, clock signals must be defined for each of them. This can be done by using the Set Clock option, available in the popup menu of the selected sequential component. A user can change the period of any clock signal at any time.

The simulation starts by choosing the desired structural scheme with option File → Open from the main menu. The simulation control is done either from the keyboard or by using the appropriate controls of the graphical user interface. Information about the current state of the simulation is given in the text message form that appears in the Messages subsection of the Info Box section. Only one component can be shown on the screen, so for complex structural schemes with several components, a user has to navigate between components by using tabs, Fig. 9a. The Up button in the Graphical Toolbar section, Fig. 9b, also allows a user to navigate one level up in a complex structural scheme. Option Step allows a user to choose by how many clock signals the simulation will move forward or back each time the CLK Forward or CLK Backward button is pressed, respectively. Button Reset is used for returning the simulation to the initial state. The current time is displayed by the label Time.

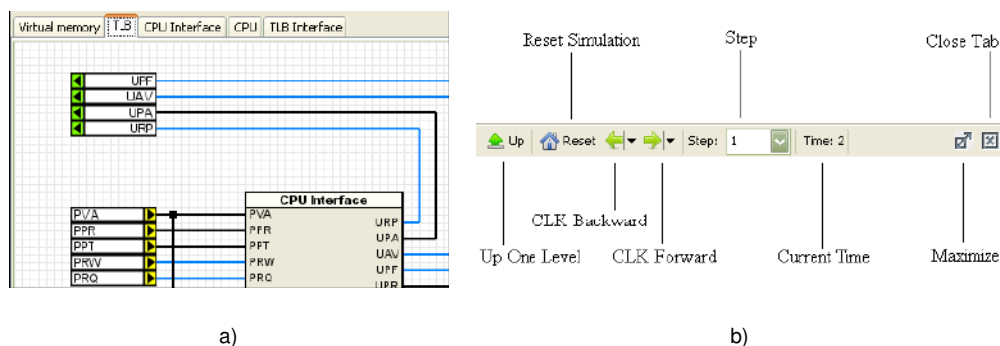


Fig. 9. Tabs and Graphical Toolbar

During the simulation, information on signal values and states of sequential modules can be obtained in various ways. The values of all the registers and RAM modules of a simulated structural scheme are given in the Info Box / Register Viewer section. The timing diagrams of signals are available in the Clock Viewer section. Register values are also shown through labels that are associated with each register. For connections containing one bit, the red, blue and green thin lines are used to indicate active, inactive

and high impedance values. For those containing more than one bit, black thick lines with labels denoting the value are used.

The SDLDS system was implemented as a desktop application in the Java programming language; it is platform independent. The system is available from the authors free of charge for use in other educational institutions.

SDLDS consists of three independent modules integrated in a complete system: the Synthesis module, the Simulation module, and the Verification module. The Synthesis module allows users to follow the formal procedure for the synthesis of an arbitrary combinational or sequential switching circuit step by step. The Simulation Module makes it possible for users to design and simulate an arbitrary switching circuit. The Verification Module allows users to verify and assess the correctness of an arbitrary switching circuit realization. In order to make SDLDS work as described, the integration of these modules is done with four different types of file, as shown in Fig. 10.

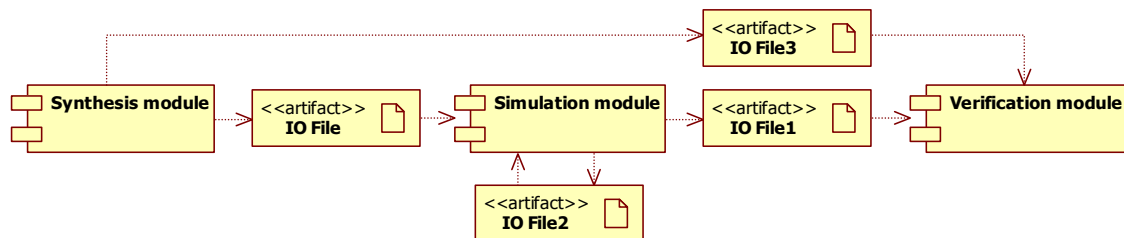


Fig. 10. Integration of the SDLDS system

IV. THE USE OF THE SDLDS SYSTEM

The SDLDS system has been used for the self-learning at home, practical work in the laboratory and knowledge assessment in exams.

A. Self-Learning

At the beginning of the course the SDLDS system and written materials covering the lectures and problems classes are made available to students. It should be noted that all structural schemes of switching circuits that appear in the materials are also given in the form that can be used by the SDLDS system, so that students can use SDLDS to verify for themselves various facets of the topics and concepts covered.

SDLDS covers combinational and sequential switching circuit synthesis by the formal procedure of the synthesis explained in Section III. For combinational circuit synthesis, the behavior of structural schemes obtained, Fig. 2 and Fig. 3, can be verified using the SDLDS's simulation features. Students are expected to verify whether the output signal values for various combinations of input signals agree with those in the truth table. Students should note that the same results are obtained for all four different structural schemes. In the case of a sequential circuit synthesis, the procedure is completed by deriving minimal expressions for the output signal and flip-flops input signals using Karnaugh maps, Fig. 6, and drawing its structural scheme, Fig. 7. Students are expected to verify whether the transitions of states and values of output signals for various combinations of input signals agree with those in the state graph. In both cases it is assumed that students use the structural schemes given in the written material to manually draw their functions and verify them by simulation, Fig. 12a.

B. Laboratory Work

The aim of the exercises is to familiarize students with both the analysis of predefined modules and the synthesis of complex modules using already-implemented ones. For their work in the laboratory students are given a manual providing the schematics of all

modules to be implemented. Students manually draw these in the simulator, and verify them by simulating their behavior for different patterns of signals.

The first group of exercises covers combinational modules, starting with multiplexers, demultiplexers, coders and decoders. An example is the implementation of a multiplexer with four inputs and one output (MP4/1) with AND, OR and NOT elements, Fig. 11a. The implemented multiplexer is saved as a complex module, and then used to implement a multiplexer with sixteen inputs and one output (MP16/1), Fig. 11b. In both cases they check whether the multiplexers function correctly by observing output signal values for all possible combinations of input signals.

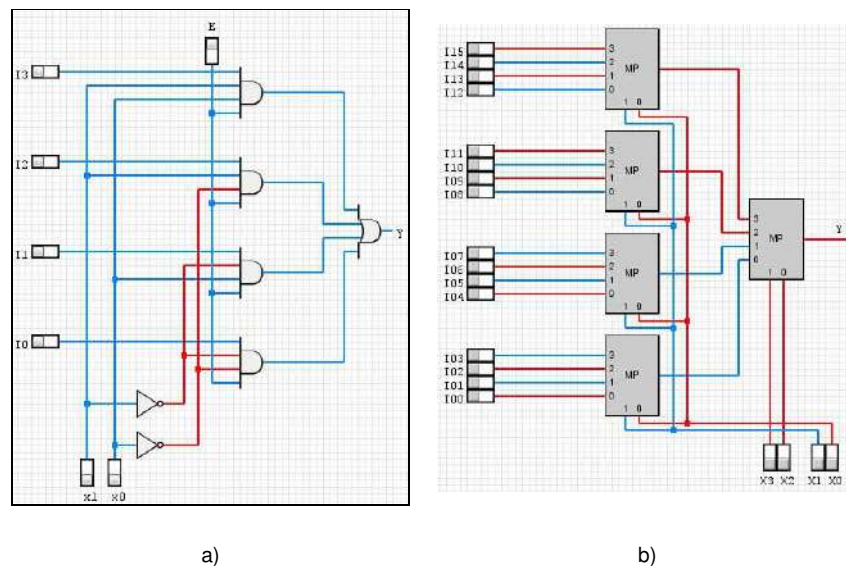


Fig. 11. Multiplexers MP4/1 and MP16/1 realized in the simulator

They continue with left, right and rotate shifters, incrementers, decrementers, adders, subtractors, arithmetic units, logic units, ALU units and comparators. An example of this group of exercises is the implementation of a one-bit FC adder by using AND, OR and NOT elements, Fig. 12a. The implemented adder is saved as a complex module, and then used to implement the four-bit ripple carry adder. Fig. 12b. To show that it is possible to realize of a faster adder, they are also required to implement a look-ahead

carry generator, Fig. 12c, and use it with one-bit FC adder to implement the four-bit carry-lookahead adder, Fig. 12d.

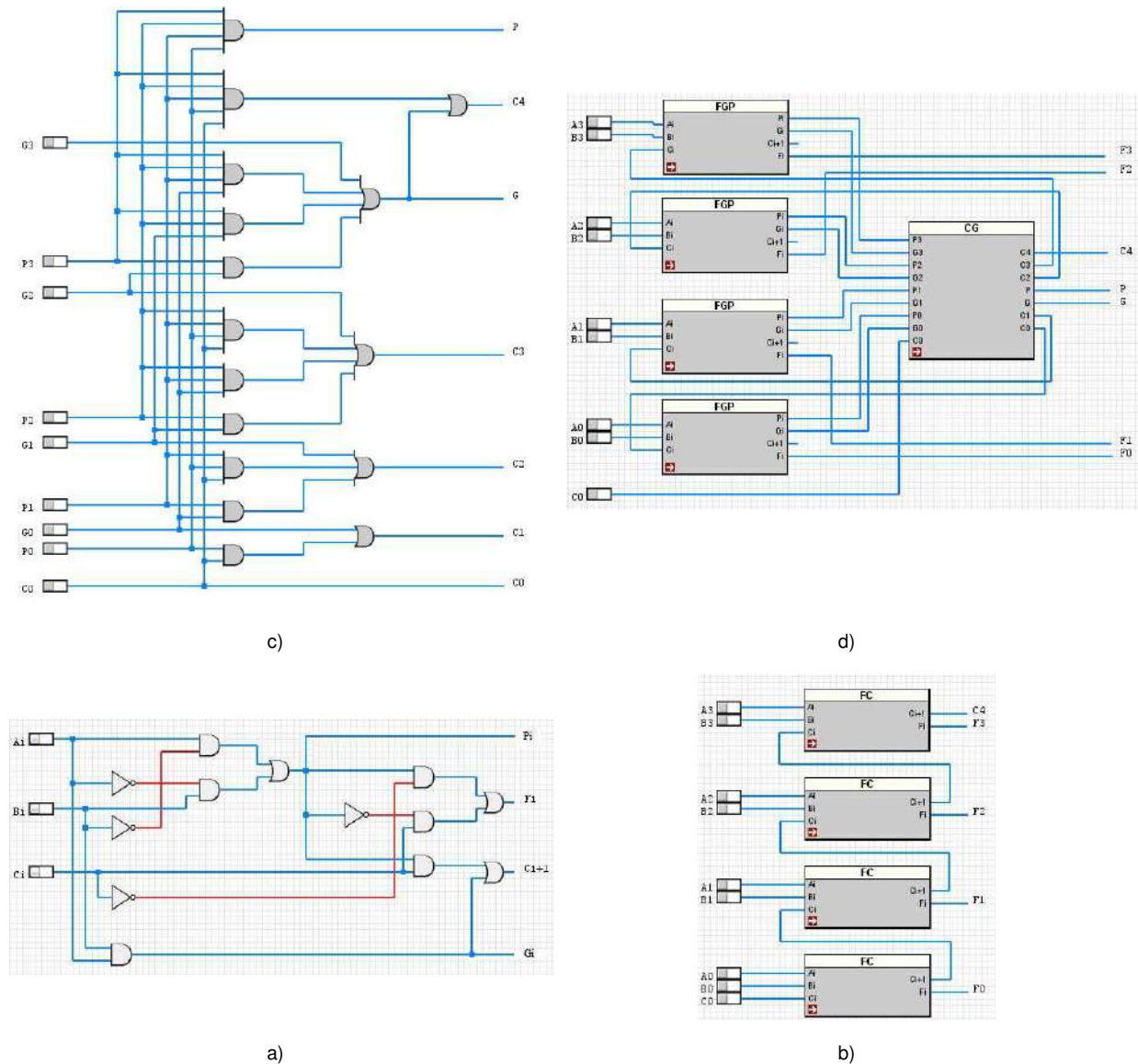


Fig. 12. One-bit FC adder, four-bit ripple carry adder, look-ahead carry generator, and four-bit carry-lookahead adder realized in the simulator

The second group of exercises covers sequential modules. They start with realizations of synchronous D, T, RS and JK flip-flops using NAND, Fig. 13a, or NOR elements. They continue with realizations of registers and counters using all four types

of flip-flops. Students implement a one-bit register with functions of load, shift left, shift right and clear, Fig. 13b. The implemented one-bit register is saved as a complex module, and then used to implement the four-bit register (Fig. 13c). Similarly they implement the one-bit counter with functions of load, increment, decrement and clear, and then use it to implement the four-bit counter.

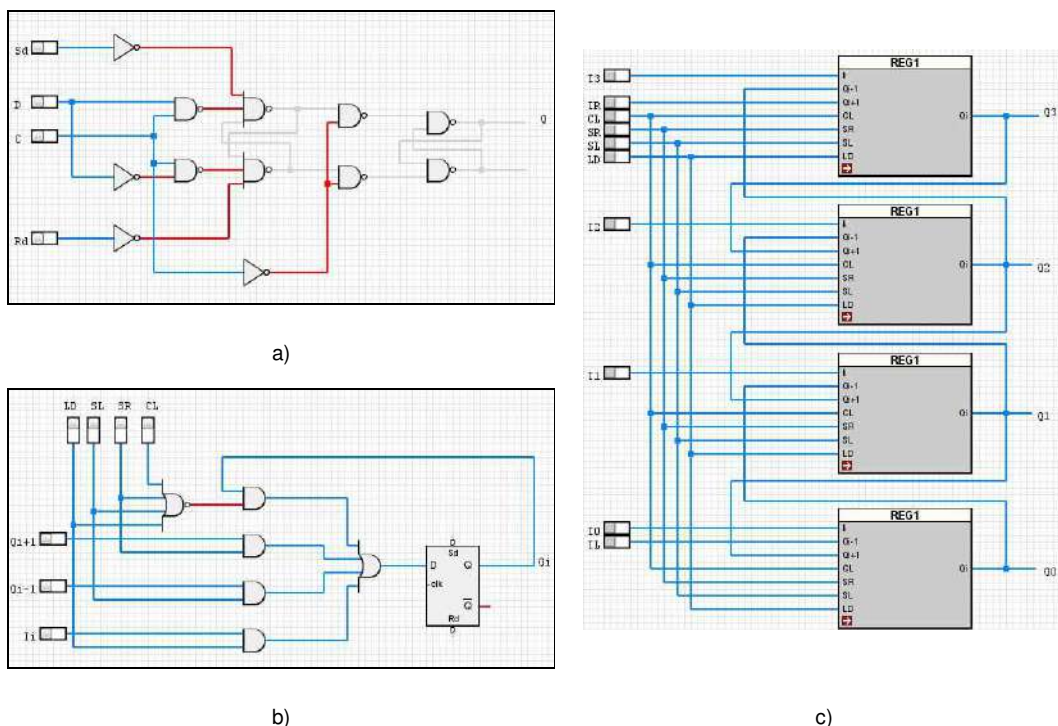


Fig. 13. D flip-flop realized by using NAND circuits, one-bit register realized by D flip-flop and four-bit register realized from one-bit registers

To motivate students to prepare themselves as well as possible for their lab work and to carry it out thoroughly, the final grade is based on accumulated points, some of which are awarded for their lab work. To this end, the SDLDS system allows teachers to create forms with questions covering typical situations for each exercise. When students conclude that a particular structural scheme functions correctly, they are required to simulate it following the prepared forms, and to answer questions. These answers are

saved in a database and, at the end of exercise, students are given their percentage of correct answers. At the end of the laboratory session, the teacher gets a detailed report on the number of points of every student.

C. The Exam

The two-part exam covers the design of switching circuits, and assesses students' knowledge.

In the first part of the exam students are given the specification of a combinational and a sequential switching circuit in the form explained in Section IV.A. Students are required to go through the complete procedure of the synthesis on the paper, manually draw the obtained structural schemes with the SDLDS system and test whether these function according to their specifications. When they conclude that their solutions are correct, they submit them through the SDLDS system. All the problems given to students are similarly solved by teachers with the SDLDS. As students submit their work, the SDLDS system compares its solutions with the students' ones and gives detailed reports. If appropriate, teachers can manually review students' work and assign points to parts of the solution.

In the second part of the exam, another set of tests assesses the knowledge acquired in the laboratory, where students have analyzed previously-implemented modules with the SDLDS system. Students are given structural schemes of several modules in the form explained in Section IV.B, along with multiple-choice questions on their behavior, Fig. 14. The system marks the responses and gives detailed reports.

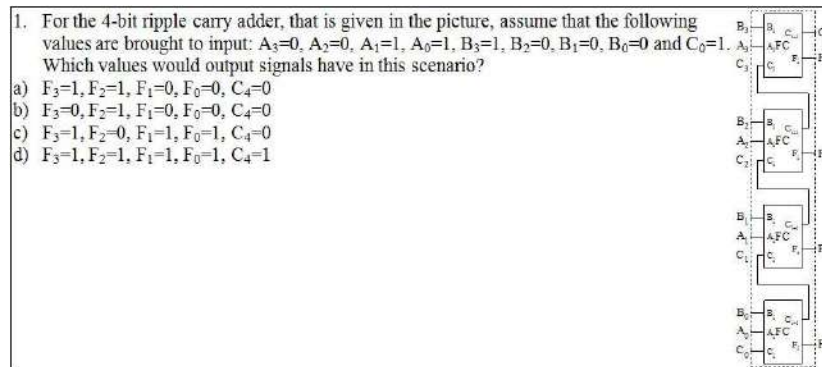


Fig. 14. Sample test question

V. BENEFITS AND ACADEMIC RESULTS

The use of SDLDS in teaching the Digital Logic course has brought benefits to both students and teachers. Since a portion of the final grade points are earned for their laboratory work, students are motivated to study throughout the semester. Also, because SDLDS generates all of them several structural schemes that can be used to achieve a particular behavior of a switching circuit, students develop an engineering approach to problem solving, where the solution is adopted based on predefined tradeoffs. Finally, students can carry out exercises for the synthesis of different combinational and sequential circuits on paper and check if their solutions are in agreement with those generated by SDLDS. In this way the SDLDS system can be used by students as an interactive tool for the exam preparation.

These benefits can be demonstrated by various metrics: the number of students who take the course, the percentage passing the exam, the average grade, and the average number of points for combinational and for sequential circuit synthesis assignments. The SDLDS system allows teachers to cover topics in digital logic with practical examples and interactive work, and automates the process of assessment and verification of students' work.

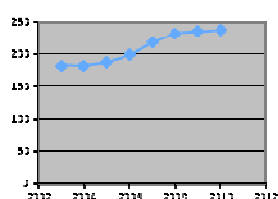
Before SDLDS was implemented in 2006, about 32% of first-year SEE-UB students chose this course, Fig. 15a. The others opted for one of four other courses covering other areas of electrical engineering and computing (2003: 183 out of 560; 2004: 182 out of 548; 2005: 188 out of 560). These numbers are almost identical with the number of students who enroll in the Department of Computer Engineering and the Department of Software Engineering in their later years of studies. However, from 2006, enrolment has grown steadily (2006: 199 out of 560; 2007: 218 out of 560; 2008: 232 out of 560; 2009: 235 out of 560; 2010: 238 out of 560), in the last year reaching over 50% of the total number of students. The increase from 32% to 51% is drawn for students from other departments, whose main areas of interest are not computers, indicating that abstract and challenging concepts in digital logic become interesting and easier to grasp when introduced with visual and interactive tools such as the SDLDS system.

The percentage of students passing the exam, Fig. 15b, has also shown constant growth (2003: 78%; 2004: 77%; 2005: 77%; 2006: 81%; 2007: 88%; 2008: 89%; 2009: 91%; 2010: 92%). Another interesting indication is that the number of students passing the exam in the first exam terms has also been constantly growing. The results in the first two exam terms (June and July) for the Digital Logic course and other courses taken by the same group of students show that more pass the Digital Logic course than the others. In the next two exam terms (September and October) this course is among those with the lowest percentage of students passing. This stems from the fact that students who actively participate in the lab work with SDLDS system usually find that it is possible, with little additional effort, to pass the exam in the first terms. Those remaining manage it with more difficulty later.

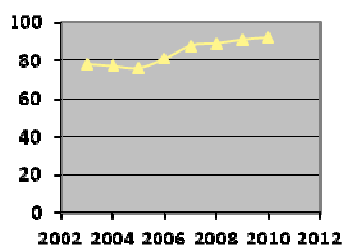
It is interesting to note the increase not only in the number of students passing the exam but also in their average grade (2003: 7.68 out of 10; 2004: 7.71; 2005: 7.72; 2006: 8.12; 2007: 8.34; 2008: 8.3; 2009: 8.4; 2010: 8.6). In the first year there was no improvement, probably indicating that some time was needed for students to realize that working with SDLDS at home was not an extra burden, but a way to prepare better for the exam, Fig. 15c.

The average number of points for the combinational circuit synthesis assignment at the exam was already high and only slight growth appears after the introduction of the SDLDS system (2003: 79%; 2004: 77%; 2005: 80%; 2006: 81%; 2007: 82%; 2008: 84%; 2009: 82%; 2010: 84%). This probably shows that students had no difficulties in understanding combinational circuit synthesis, Fig. 15d.

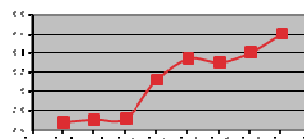
However, the average number of points for the sequential circuit synthesis assignment at the exam shows steady growth after the introduction of SDLDS (2003: 30%; 2004: 31%; 2005: 28%; 2006: 35%; 2007: 39%; 2008: 41%; 2009: 48%; 2010: 55%). The authors believe that is because sequential circuit synthesis is too abstract for students when done on paper, and SDLDS made it understandable, Fig. 15e.



a)



b)



c)

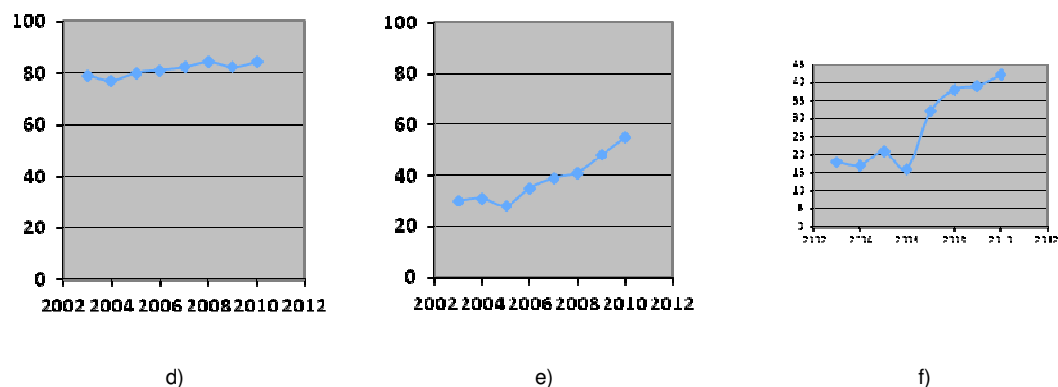


Fig. 15. Metrics

Two interesting effects of the introduction of SDLDS have been noticed with two different groups of student who take Computer Architecture courses in their second year of study. The first is for students from the Departments of Computer Engineering and of Software Engineering for whom the course is compulsory, and whose average grade has increased (2003: 7.34 out of 10; 2004: 7.25 out of 10; 2005: 7.4 out of 10; 2006: 7.22 out of 10; 2007: 7.63 out of 10; 2008: 7.81 out of 10; 2009: 8.02 out of 10; 2010: 8.09 out of 10). The second is for students from other departments for whom the course is an elective, Fig. 15f, whose numbers have shown a constant growth (2003: 18 out of 240; 2004: 17 out of 240; 2005: 21 out of 240; 2006: 16 out of 240; 2007: 32 out of 240; 2008: 38 out of 240; 2009: 39 out of 240; 2010: 42 out of 240).

Fig. 16 shows the SDLDS report of all students' activities in the form of points gained for their practical work in the laboratory and in the exam. Their final grades are transferred into the school's information system, which keeps information for each student about all their activities from the enrollment till the graduation.

COMPLETE REPORT
DATE: 05.06.2010.

Index Number	Name and Surname	Laboratory Points (MAX 40)	Exam Points		Total Points (MAX 100)	Final Grade
			Practical Part Points (MAX 40)	Theoretical Part Points (MAX 20)		
2008/0465	Zivorad Markulić	17	0	5	22	5
2009/0182	Irfan Džanković	32.5	8	10	50.5	6
2009/0241	Danilo Petković	38	8	10	56	6
2009/0338	Nenad Ivanović	40	40	15	95	10
2009/0402	Jasmin Musić	34	25	20	79	8
2009/0407	Nemanja Stevanović	40	36	20	96	10
2009/0421	Miloš Tripović	38	20	5	63	7
2009/0425	Milić Džakula	35.5	33	15	83.5	9
2009/0503	Miloš Đurović	32.5	20	10	62.5	7
2009/0534	Marko Marić	31.5	12	0	43.5	5

Fig. 16. Accumulative report about students' work

VI. CONCLUSION

This paper has presented the basic features of the SDLDS system and the experience of using it in teaching the Digital Logic course at the SEE-UB. The overall benefits of using the SDLDS system are illustrated by, over the five years of its use, the number of students electing to take the course having increased from 32% to 51%, the percentage passing the exam having risen from 77% to 92%, the average grade improving from 7.68 to 8.6 out of 10, and the average number of points for sequential circuit synthesis assignment at the exam improving from 30 to 55 out of 100.

REFERENCES

- [1] IEEE Computer Society and ACM, "Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering," Available from: <http://www.eng.auburn.edu/ece/CCCE/CCCE-FinalReport-2004Dec12.pdf> (accessed December 2004).
- [2] Z. Kohavi, and N. Jha, "Switching and Finite Automata Theory," 3rd ed. Cambridge: Cambridge University Press, 2010.
- [3] B. Holdsworth, and R. C. Woods, "Digital Logic Design," 4th ed. London: Newnes, 2002.
- [4] M. Rafiquizzaman, "Fundamentals of Digital Logic and Microcomputer Design," 5th ed. Wiley, 2005.

- [5] A. Trost, and B. Zajc, "Logic emulators in digital systems education," *2011 IEEE EUROCON - International Conference on Computer as a Tool (EUROCON)*, Portugal, Lisbon, pp. 1-4, 2011.
- [6] B. Nikolic, Z. Radivojevic, J. Djordjevic, V. Milutinovic, "A Survey and Evaluation of Simulators Suitable for Teaching Courses in Computer Architecture and Organization," *IEEE Transactions on Education*, vol. 52, no. 4, pp. 449-459, 2009.
- [7] Institute for Computing Systems Architecture, School of Informatics, University of Edinburgh, "HASE - a computer architecture simulation environment," Available from: <http://www.icsa.inf.ed.ac.uk/research/groups/hase/> (accessed August 2005).
- [8] R. N. Ibbett, "HASE DLX Simulation Model," *IEEE Computer Society, IEEE Micro, Special Issue on Computer Architecture Education*, vol. 20, no. 3, pp. 38-47, May/June 2000.
- [9] Institute for Computing Systems Architecture, School of Informatics, University of Edinburgh, "COMPUTER ARCHITECTURE: HASE Dinero," Available from: <http://www.icsa.informatics.ed.ac.uk/research/groups/hase/projects/dinero/> (accessed August 2005).
- [10] Brigham Young University, "JHDL Overview," Available from: <http://www.jhdl.org/overview.html> (accessed August 2005).
- [11] B. Hutchings, P. Bellows, J. Hawkins, S. Hemmert, B. Nelson, and M. Rytting, "A CAD Suite for High-Performance FPGA Design," *Field-Programmable Custom Computing Machines, 1999. FCCM '99. Proceedings. Seventh Annual IEEE Symposium on*, Napa Valley, CA, USA, pp. 12-24, 1999.
- [12] P. Bellows, B. Hutchings, "JHDL - An HDL for Reconfigurable Systems," *FPGAs for Custom Computing Machines, 1998. Proceedings. IEEE Symposium on*, Napa Valley, CA, USA, pp. 175-184, 1998.
- [13] "ISE WebPACK Design Software," Available from: <http://www.xilinx.com/products/design-tools/ise-design-suite/ise-webpack.htm> (accessed December 2011).
- [14] D. Theys, A. Troy, "Lessons learned from teaching computer architecture to computer science students," *33rd Annual Frontiers in Education Conference*, Boulder, pp. F1C_7-F1C12, 2003.
- [15] M.A. Lusco, and C.E. Stroud, "PSIM: A processor SIMulator for basic computer architecture and operation education," *Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon)*, Concord, pp. 115-118, 2010.

- [16]C. Burch, "Logisim," Available from: www.cburch.com/logisim/ (accessed May 2007).
- [17]C. Burch, "Logisim: A graphical system for logic circuit design and simulation," *Journal of Educational Resources in Computing*, vol. 2, no. 1, pp. 5-16, 2002.
- [18]C. Burch, and L. Ziegler, "Science of Computing Suite (SOCS): Resources for a Breadth-First Introduction," *ACM SIGCSE Bulletin*, vol. 36, no. 1, pp. 437-441, 2004.
- [19]YOERIC Corporation, "YOERIC Corporation - WinBreadboard," Available from: <http://www.yoeric.com/virtualvulcan.htm> (accessed January 2005).
- [20]N. Glass, "Java Digital Breadboard Simulator: A simulator for an educational electronics environment," Available from: <http://www.cs.york.ac.uk/jbb> (accessed January 2005).
- [21]T. Robal, A. Kalja, "Creating interactive learning objects with web services," *EAAEIE Annual conference*, Spain, pp. 1-6, 2009.
- [22]T. Robal, A. Kalja, "Applying e-Environments in Teaching the Basics of Digital Logic," *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)*, San Diego, pp. 41-42, 2007.
- [23]T. Weng, Y. Zhu, and CK. Cheng, "Digital Design and Programmable Logic Boards: Do Students Actually Learn More?," *38th ASEE/IEEE Frontiers in Education Conference*, Saratoga Springs, pp. S1H-1 - S1H-1, 2008.

Zarko Stanisavljevic received the B.Sc. and M.S. degrees in electrical engineering from the University of Belgrade, Serbia. He is currently a Teaching Assistant in the Department of Computer Engineering at the School of Electrical Engineering, University of Belgrade, Serbia. His research interests include computer architecture and organization, digital systems simulation, network security, cryptography and internet programming.

Vladimir Pavlovic received the B.Sc. degree in electrical engineering from the University of Belgrade, Serbia. He is currently doing research related to his M.S. thesis at the Department of Computer Engineering at the School of Electrical Engineering, University of Belgrade, Serbia. His interests include embedded systems, Automotive Open System Architecture and digital systems simulation.

Bosko Nikolic (M'09) received the B.Sc., M.S., and Ph.D. degrees in electrical engineering from the University of Belgrade, Serbia. He is currently an Associate Professor with the Department of Computer Engineering, School of Electrical Engineering, University of Belgrade. His research interests include Internet programming, digital systems simulation, and the programming language Java.

Jovan Djordjevic received the B.Sc. degree in electrical engineering from the University of Belgrade, Serbia, and the M.S. and Ph.D. degrees in computer science from the University of Manchester, UK. He is currently a Professor of Computer Engineering at the Faculty of Electrical Engineering, University of Belgrade. His research interests include computer architecture and organization, fault tolerant computer systems, parallel computer systems, computer networks, digital systems simulation and distance learning.