Available online at www.sciencedirect.com
SciVerse ScienceDirect
Procedia
Social and Behavioral Sciences
Procedia - Social and Behavioral Sciences 28 (2011) 305 – 310

WCETR 2011

Logic circuit design verification support tool - FitBoard

Katarína Jelemenská[a]*, Miroslav Siebert[a], Dominik Macko[a], Pavel Čičák[a]

[a]*Faculty of Informatics and Information Technologies, Slovak University of Technology, Ilkovičova 3, Bratislava 842 16, Slovakia*

Abstract

Logic circuits knowledge is the cornerstone of all the courses devoted to digital systems design. Therefore, it is important to ensure a high level of knowledge, understanding as well as the skills of all the students of informatics and computer science study programs. The present study introduces a method to improve the students' results by the means of a multi-purpose virtual verification panel called FitBoard. The above mentioned tool is focused specifically on understanding and practicing the complete set of logic gates. However, the tool also provides many additional features. FitBoard is platform independent, easy to use, configurable for various tasks and restrictions and useful in learning as well as in the assessment process. Compared to the other available logic gates simulators, the simplicity and intuitive user interface of FitBoard enables the students to concentrate on the assigned tasks without first studying the usage. The data collected during the course proved a positive response from both the teachers as well as the students.

Keywords: logic circuits; virtual verification panel; learning; assessment.

1. Introduction

Logic circuits design methods are considered as a fundamental knowledge in informatics and computer science study programs not only at the Faculty of Informatics and Information Technologies, Slovak University of Technology (FIIT STU) in Bratislava but also in universities worldwide (Baláž et al., 2005; Boluda et al., 2006; Burch, 2002, 2004; Donzellini, 2011; Giertl et al., 2009; James et al., 2005; Kohl et al., 2007; Pohronská et al., 2009). The reason lays in the fact that logic circuits are the basis of all the digital systems used these days. Therefore, proper understanding and acquisition of knowledge in the above mentioned area is essential. In addition, logic circuits represent the basic precondition for a successful study of the complex technologies and knowledge that is built on them. In courses devoted to the logic circuits design it is especially important to have a possibility to verify the designs and experiment with their versions. One of the most common restrictions faced when designing logic circuits represent the types of logic gates advised with a limited number of inputs. Therefore, it is necessary to impose such a constraint to the students' assignments. In practice, the students are required to design their logic functions with respect to the specified functionally complete set of logic gates. The above mentioned set can be used to construct any logic function in two-valued algebra. Previously, real logic gates panels were used for hands-on labs, whereby students could hard wire their Boolean functions to verify them. The above mentioned panels posed

* Katarína Jelemenská. Tel.: +421-910-942-659; fax: +421-2-654-20587.
E-mail address: jelemenska@fiit.stuba.sk.

real limitations in size, number and types of logic gates. Nowadays, these panels could be replaced by breadboards or some types of programmable logic devices. However, in addition to the higher price, other disadvantages, such as restricted availability, more complex usage requiring additional knowledge, restrictions setting limitations, etc also exist. From this point of view, a virtual version of the previously used logic gates panels, i.e. some type of a logic circuits simulator would be more advantageous. As a result, advantages, such as ease of use with no troubleshooting, faster, and more efficient could be accrued.

At FIIT STU, logic circuits represent an important part of the two introductory courses of the bachelor study programs. In the laboratory sections of the above mentioned courses, the students are required to complete several structured assignments in the computer laboratories. The available laboratory time is limited. Therefore, it is important to minimize the time that the students spend on learning how a simulator works. For this reason, it is important that the simulator should be extremely intuitive, easy to use and speed up the design as much as possible. On the other hand, the capacity required from the simulator is relatively simple. We do not require sophisticated editing abilities, but features that would provide additional verification possibilities, such as truth tables or Karnaugh maps. Other important features include the availability and portability, which enable the students to work at home, in case they are unable to complete the assignments in the lab.

2. Related work

Many applications devoted to logic gates simulation and verification are available. Some of these applications include freeware or shareware solutions, while others include commercial ones. The selected features of the evaluated simulators are compared and presented in Table 1. It is not required that all the applications should be installed on the local computer. Alternatively, the applications can be executable directly in a web browser (Java applets and Adobe Flash). However, this is not always considered as an advantage as such applications are usually functionally limited. For instance, in the applet version of Simcir and LogicSim users are unable to save the designs, while The Logic Lab stores a design to the web server and provides the URL address to retrieve it. In addition, the internet connection may not always be available. Thus, sometimes it may be preferable to work off-line.

Table 1. Comparative analysis of logic circuit simulators

	wires indicate values	**wire path calculation**	**custom wire paths**	**variable-input gates**	**save/open circuit**	**undo/redo functions**	**truth table display**	**Karnaugh map displ.**	**export to VHDL**	**restricted set of logic gates**	**platform**	**type**
FitBoard	yes	yes	no	yes	yes	yes	yes	yes	yes	yes	JVM	open source
LogicSim (Tetzl, 2009)	yes	no	yes	no	yes no	no	no	no	no	no	JVM / applet	open source
Simcir (Arase, 2000)	no	yes	no	no	yes no	no	no	no	no	no	JVM / applet	open source
The Logic Lab (Temmerman, 2009)	yes	yes	no	no	yes	no	no	no	no	no	Adobe Flash	freeware
Probe (Boothe, 1999)	yes	yes	no	no	yes	no	yes	no	no	no	applet	freeware
Logisim (Burch, 2011)	yes	yes	yes	yes	yes	yes	yes	yes	no	no	JVM	open source
LOG (Gillespie, 1985)	no	no	yes	yes	yes	no	no	no	no	no	Windows	freeware
Deeds (Donzellini, 2011)	no	no	yes	yes	yes	yes	no	no	no	no	Windows	freeware
CedarLogic (Kohl, 2007)	yes	yes	yes	yes	yes	yes	no	no	no	no	Windows	freeware
LOGiX (CommTec, 2008)	yes	no	yes	yes	yes	yes	yes	no	no	no	Windows	commercial
Logicly (Tynjala, 2010)	yes	yes	no	yes	yes	no	no	no	no	no	Adobe Flash	commercial
Digital-ProfiLab (Abacom, 2010)	no	no	yes	yes	yes	no	no	no	no	no	Windows	commercial

A majority of the compared simulators provide a graphical/textual toolbox interface for composing the logic circuit. The composition time is significantly affected by factors, such as the manner of adding as well as interconnecting the components to the canvas area. For instance, in LogicSim and Probe every component has to be selected and placed separately. The above mentioned approach is a far more cumbersome and time consuming approach when compared to the repeated placement of the selected component used in Logisim or the drag and drop technique implemented in Logicly, Simcir, and The Logic Lab. The automated wire path calculation available in some simulators can substantially speed-up the interconnection process of the components. However, if the algorithm allows overlapping wires, as in Probe or calculates just the straight line, as in Simcir the resulting circuit would hardly be readable. Therefore, for supporting larger circuits it is better to allow users to create custom wire paths (Burch, 2011). The undo and redo functions are often considered the basic editing functions. Thus, many students would refuse applications which omit these functions. However, in spite of this a number of simulators do not support these functions.

Most of the compared simulators support the variable-input gates. Thus, the logic gates, such as AND, OR etc. can have more then two inputs. Only a few applications are enabled to display the truth tables or Karnaugh maps for a composed circuit or export to VHDL, which are the advanced features useful for guiding beginners in circuit design. The exported VHDL model can be easily used to synthesize the circuit for the FPGA board. Thus, the beginners are enabled to verify the model on the real hardware without possessing knowledge about the VHDL design. To the best of knowledge none of the available simulators, except FitBoard facilitates imposing any restrictions concerning the usable set of logic gates. The above mentioned property can substantially ease the assessment of the students' assignments in case the functionally complete set of logic gates is specified. Thus, for the above mentioned reasons we decided to design and implement the new tool, which would be more focused on the specific needs of the basic courses on logic circuit design. The similar approach was used in (Pištek et al., in press).

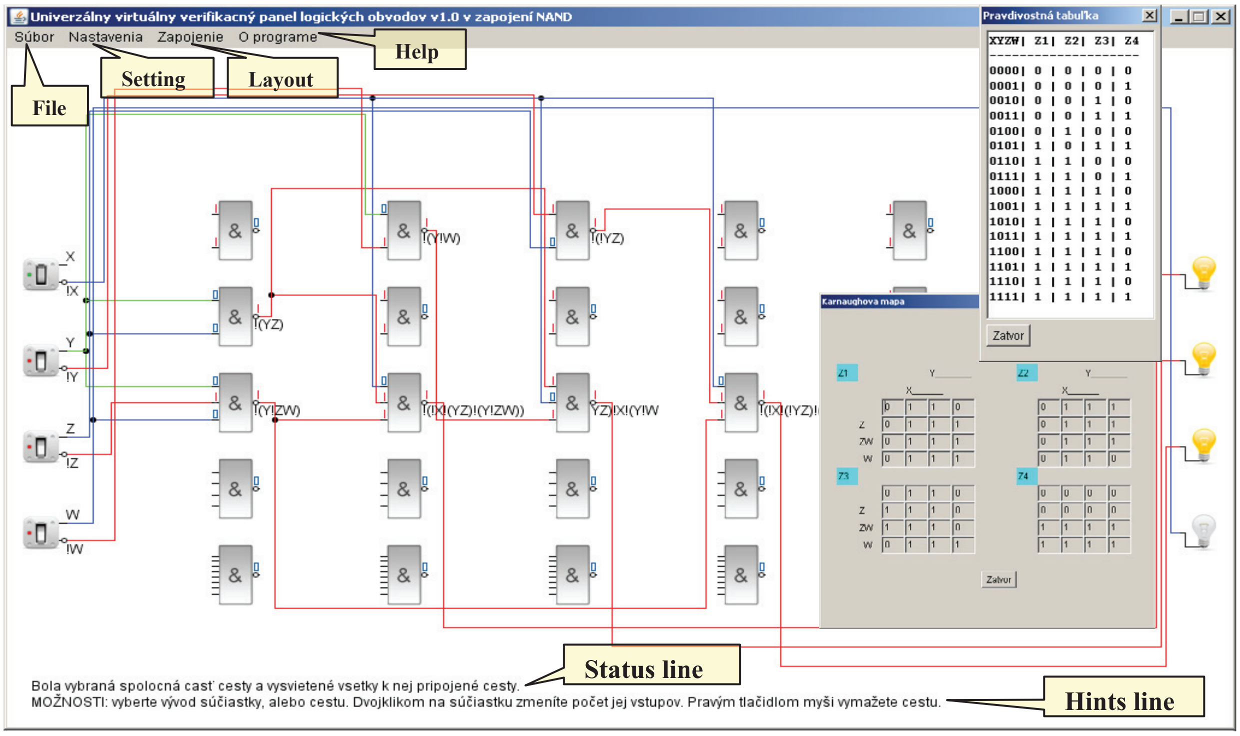
3. FitBoard functionality

The proposed program has been designed using the objective-oriented approach in the Java programming language, which ensures the portability and operating system independence. To keep the program usage as simple as possible the number of user settings and options is minimized. In addition, default settings have been used for most values. The program is active only in the case of user interaction. Thus, the program is activated by a mouse click and after the completion of all the necessary actions it goes into the 'sleep' state again. Consequently, this leads to minimal system requirements, whereby the program is enabled to run on older computers and operating systems.

3.1. Distribution of elements

At first, the user has to choose a specific complete set of logic gates, which would result in a particular type of logic gates panel. There are ten sets of logic gates available including the most common NAND, NOR, OR&NOT, AND&NOT, etc. The simulator also enables the use of multiplexers instead of logic gates and flip-flops (SR, JK, D or T) to build the sequential circuits.

There are two possible modes available. The mode selection depends on the presence of the file named conf.file, which describes the available configurations of the logic gates and flip-flops. The above mentioned mode is especially useful for exams when the students are obliged to use the specific configuration to solve the task. Thus, the students are compelled to use only the gates from the specified functionally complete set of logic gates with a specified number of inputs. In case the file is missing the user is prompted to specify a custom configuration. In both the modes the logic gates, flip-flops, and other elements are then automatically distributed on the canvas, thereby saving the user's time. An example of a resulting NAND gates panel is given in Figure 1. On the left-hand side there are four input switches providing input values in a direct and negated form and then thirty NAND gates. On the right-hand side there are four bulbs to signalize the output values. The bulb lights up when the input value is true. At the bottom of the window, the information concerning the occurred events as well as the available functions in the current state is displayed. The upper bar represents the Applications menu.



```
XYZW|  Z1|  Z2|  Z3|  Z4
------------------------
0000|  0  |  0  |  0  |  0
0001|  0  |  0  |  0  |  1
0010|  0  |  0  |  1  |  0
0011|  0  |  0  |  1  |  1
0100|  0  |  1  |  0  |  0
0101|  1  |  0  |  1  |  1
0110|  1  |  1  |  0  |  0
0111|  1  |  1  |  0  |  1
1000|  1  |  1  |  1  |  0
1001|  1  |  1  |  1  |  1
1010|  1  |  1  |  1  |  0
1011|  1  |  1  |  1  |  1
1100|  1  |  1  |  1  |  0
1101|  1  |  1  |  1  |  1
1110|  1  |  1  |  1  |  0
1111|  1  |  1  |  1  |  1
```

Z1

	0	1	1	0
Z	0	1	1	1
ZW	0	1	1	1
W	0	1	1	1

Z2

0	1	1	1
0	1	1	1
0	1	1	1
0	1	1	0

Z3

	0	1	1	0
Z	1	1	1	0
ZW	1	1	1	0
W	0	1	1	1

Z4

0	0	0	0
0	0	0	0
1	1	1	1
1	1	1	1

Bola vybraná spolocná casť cesty a vysvietené vsetky k nej pripojené cesty.
MOŽNOSTI: vyberte vývod súčiastky, alebo cestu. Dvojklikom na súčiastku zmeníte počet jej vstupov. Pravým tlačidlom myši vymažete cestu.

Figure 1. FitBoards user interface - the main window and truth table and Karnaugh maps pop-up windows.

3.2. Program control

When working in custom configuration, all the logic gates (except NOT) are initially two-input gates. This can be changed by double-clicking the respective gate and selecting the number of inputs in a dialog box. The available values are three, four, five, and eight.

The connection between the two elements can be created by simply clicking the left mouse button on the wire ends. The first end must be an output or input pin, while the second one can be either a pin or an existing wire. It is not necessary to hit the pin/wire very precisely and proximity of about 10 pixels is acceptable. The pin is highlighted to confirm the selection. After correctly selecting both the wire ends, the best route is automatically calculated. The wires are routed horizontally and/or vertically and snapped to a grid. The wires connected to the same output pin result in wire branching, which is denoted by junction nodes. Not surprisingly, the necessary check-ups are performed to prevent any incorrect wire connections. An incorrect attempt invokes an error message, which is displayed at the bottom of the window.

The simulation results are refreshed each time a new wire is added. The values of connected wires are displayed at the logic gate inputs. The resulting value and the logic function are displayed at the output. The more complex expressions are automatically simplified using the Boolean laws. These options can be individually disabled using the drop-down menu Settings (Nastavenia). The wire coloring can also be enabled in the same menu. The above mentioned option enables to see the values flowing through the wires, whereby red indicates a true value and blue indicates a false value. Any wire can be selected by clicking the left mouse button on the wire. The selected wire is colored bright green, which helps to easily track the path. The right mouse button click on the wire can be used to delete an existing connection. In the case of a branching wire only the final sections of the connection can be deleted. A left mouse button click is required to change the input switch status.

The Layout (Zapojenie) menu offers options, such as Undo, Redo, Export to VHDL, Truth tables, Karnaugh maps, and Save as Picture. The keyboard short cuts are available for all the options. The truth tables and Karnaugh

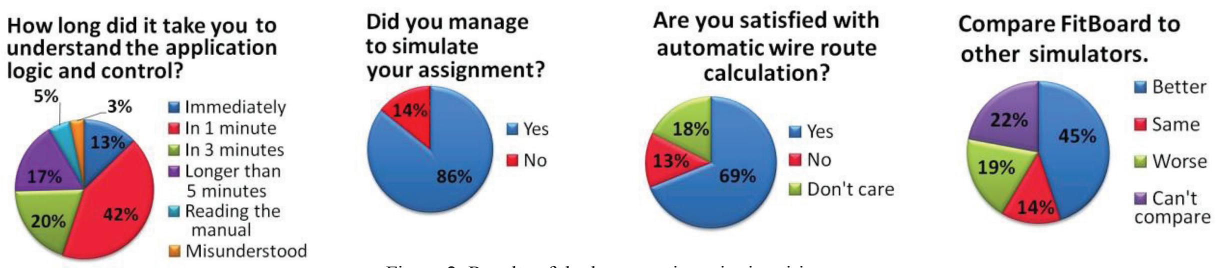


Figure 2. Results of the key questionnaire inquiries:

maps are displayed in a pop-up window (see Figure 1), whereby X through W denotes the inputs and Z1 through Z4 denotes outputs.

4. Results and experience

In academic year 2010/2011, the FitBoard was put to a pilot run in three courses: Computer engineering principles, Logic circuits, and Digital systems description. In the first two courses, the students were required to design simple combinational and synchronous sequential circuits using traditional design methods. In the first assignment, the students designed a converter between two numerical binary codes using the NAND gates. In the second assignment, the students designed a sequential circuit for the comparison of the two binary numbers. The third course is primarily devoted to methods based on hardware description languages (HDL), whereby FitBoard can be used to illustrate the relation between the circuit and its HDL description. The option Export to VHDL can especially be useful for the above mentioned purpose. However, in the present course it was not compulsory to use FitBoard. Thus, a majority of the students preferred simulators, which they were already familiar with. Altogether, more than 180 students were involved in the pilot run out of which 83 students filled out the questionnaire. The most important questionnaire results are given in Figure 2 and will be summarized here.

More than 85% of the students managed to simulate the assignment circuit in the FitBoard. The answers to the question “How long did it take you to understand the application logic and control?” confirm the satisfaction of the main aim of the present study- simplicity and intuitiveness. 74% of students replied it took them less then 3 minutes, 5% had to read the manual and 3% claim that they did not understand. The automatic wire route calculation approved 69% of the students and disapproved 13%. A majority of the students consider FitBoard as good as or even better compared to the previously used logic gates simulators (each of them used at least one).

5. Conclusions

Applying FitBoard into practice revealed some minor flaws, which have been corrected on the run. We are pleased with the students' poll results showing that the proposed logic gates simulator is suitable for the educational process, whereby its simplicity does not burden the students with complicated control. Simultaneously, the simulator also provides a quality tool to verify their practical knowledge. Compared to the real verification panels the proposed software solution brings some additional extra features, such as displaying logical values on wires and not only at the inputs and outputs of the circuit. Thus, it is possible to detect an error even at its inception and not after the completion of the circuit. Other interesting features include save/retrieve circuit, save as picture for documentation purposes, logic functions on gates’ outputs or truth tables and Karnaughs maps.

Acknowledgment

The authors would like to gratefully acknowledge the support provided by Slovak Science Grant Agency (VEGA 1/0649/09 “Security and reliability in distributed computer systems and mobile computer networks“) and HP Technology for Teaching grant 2007 “Mobile Education Center”.

References

Abacom (2010). *Digital-ProfiLab 4.0. Product specification.* Retrieved May 10th, 2011, from http://www.abacom-online.de/uk/html/digital-profilab.html

Arase, K. (2000). *Simcir the circuit simulator, Version 1.2.1.* Retrieved May 10th, 2011, from http://www.d-project.com/simcir/

Baláž, M., Gramatová, E.,Pikula, T., Fischerová, M. (2005). eTool for teaching and application of digital system testability techniques. *EUROCON 2005 Proceedings* (pp. 831-834). IEEE. doi: 10.1109/EURCON.2005.1630061

Boluda, J.C., Peiro, M.A., Torres, M.A., Girones, R., Palero, R.J. (2006). An active methodology for teaching electronic systems design. *IEEE Transactions on Education, 49 (3)*, 355 - 359.

Boothe, B. (1999). *Probe internet logic circuit simulator, Version 0.97.* Retrieved May 10th, 2011, from http://www.scit.wlv.ac.uk/~cm1970/probe/webpage/probeapp.html

Burch, C. (2002). Logisim: a graphical system for logic circuit design and simulation. *Journal on Educational Resources in Computing (JERIC), 2 (1),* 5-16.

Burch, C. (2011). *Logisim 2.7.1 webpage.* Retrieved May 10th, 2011, from http://ozark.hendrix.edu/~burch/logisim/

Burch, C. & Ziegler, L. (2004). Science of computing suite (SOCS): resources for a breadth-first introduction. *SIGCSE'04 Proceedings* (pp. 437-441). New York, NY: ACM. doi:10.1145/971300.971447

CommTec Software Engineering. (2008). *LOGiX - simulation of logic circuits.* Retrieved May 10th, 2011, from http://www.commtec.de/logix/index.php

Donzellini, G. (2011). *Deeds - digital electronics education and design suite.* Retrieved May 10th, 2011, from www.esng.dibe.unige.it/netpro/Deeds/Index.htm

Donzellini, G., & Ponta, D. (2009). From gates to embedded systems: a bottomup approach to digital design. *MSE09 Proceedings* (pp. 61 - 64). Printing House. doi:10.1109/MSE.2009.5270830

Giertl J., Révés M., Kolcun P., Jakab F., Fecilľak P., Kopka J. (2009). Towards the secure using of network-oriented educational systems. *ICETA 2009 Proceedings* (pp. 71-76). Košice, Slovakia: elfa.

Gillespie, D. (1985). *The Log system. UC Berkeley.* Retrieved May 10th, 2011, from http://www.cs.berkeley.edu/~lazzaro/chipmunk/describe/log.html

James, H. A., Hawick, K. A. & James, C. J. (2005). Teaching students how to be computer scientists through student projects. ACE '05 Proceedings of the 7th Australasian conference on Computing education (pp. 259-267). Darlinghurst, Australia: Australian Computer Society.

Kohl, C., & Shomper, K. (2007). CedarLogic - a new graphical digital logic CAD tool to aid in the teaching of Digital Logic Design. *Proceedings of 2007 ASEE International conference CD.* Honolulu, Hawaii: American Society for Engineering Education.

National Instruments. (2008) *Integrated design and test platform with NI Multisim, Ultiboard, and LabVIEW.* Data Sheet. Retrieved May 10th, 2011, from http://www.ni.com/multisim/

Pištek, P., Marcinčin, R., Palaj, T., Štrba, J. (in press). *Logical circuits design education based on virtual verification panel.* CISSE 2010 - Proceedings in press.

Pohronská, M., Krajčovič, T. (2009). ARM7 Based Embedded System for Education. *Proceedings of Applied Electronics 2009* (pp. 207-210). Pilsen, Czech Republic: Západočeská univerzita v Plzni.

Temmerman, K. (2009). *The Logic Lab.* Retrieved May 10th, 2011, from http://www.neuroproductions.be/experiments/the-logic-lab/

Tezer, M., & Bicen, H. (2009). The Preparations University Teachers Towards E-Education Systems. *Cypriot Journal Of Educational Sciences, 3*(1), 16-27.

Tetzl, A. (2009). *LogicSim logic simulator.* Retrieved May 10th, 2011, from http://www.tetzl.de/java_logic_simulator.html

Tynjala, J. (2010). *Logicly.* Retrieved May 10th, 2011, from http://logic.ly/