Interactive Teaching of Elementary Digital Logic Design With WinLogiLab

Article *in* IEEE Transactions on Education · June 2004

DOI: 10.1109/TE.2004.824843 · Source: IEEE Xplore

2 authors, including:

Renate Sitte
Griffith University
71 PUBLICATIONS **843** CITATIONS

SEE PROFILE

Interactive Teaching of Elementary Digital Logic Design With WinLogiLab

Charles Hacker and Renate Sitte, *Member, IEEE*

***Abstract*—This paper presents an interactive computerized teaching suite developed for the design of combinatorial and sequential logic circuits. This suite fills a perceived gap in the currently available computer-based teaching software, with the purpose of providing alternative-mode subject delivery. The authors were, therefore, prompted to develop a Microsoft-Windows tutorial suite, WinLogiLab, comprising a set of interactive tutorials that show the link between Boolean algebra and digital combinatorial and sequential circuits. The combinatorial tutorials follow the initial design steps: from Boolean algebra, to truth tables, to minimization techniques, to production of the combinatorial circuit in a seamless way. Similarly, the sequential tutorials can design simple finite-state counters and can model more complex finite-state automata.**

***Index Terms*—Boolean algebra, computer-aided logic design, educational technology, Karnaugh map, logic gates, minimization software, online-interactive learning, Quine–McCluskey.**

I. Introduction

TEACHING elementary circuit design can be a challenge because electrical engineering students often do not see the immediate relationship between cause and effect, which can be seen, for example, in mechanical engineering experiments. Traditionally, circuit-design teaching follows a three-stage sequence from introductory logic to combinatorial circuits, and, several lectures later, to sequential circuits. While a number of packages dealing with circuit-related issues are available, they are not specifically designed for a seamless interactive-learning environment that demonstrates the progression from Boolean algebra through optimization, to the finally designed circuit in a modern Windows environment. To fill this gap, the authors have designed and implemented WinLogiLab. WinLogiLab is an interactive Microsoft (MS)-Windows-compatible computerized teaching suite to aid in the teaching of combinatorial and sequential logic design. This software is applicable to introductory digital design courses in Electrical Engineering, Computer Science, and Computer Engineering curricula. It serves both as student-centered self-paced learning and as a teaching demonstration tool. The main contribution of this work is to provide a set of interactive teaching aids to approach the basics of combinatorial and sequential digital circuit design. Its strength lies in its pedagogic value by *showing* to novices the link between Boolean algebra and the finally designed digital logic circuits, in a fully integrated environment, and the convenience of being MS-Windows compatible.

Manuscript received November 7, 2001; revised January 23, 2003.

The authors are with the Faculty of Engineering and Information Technology, Griffith University, Gold Coast Campus, Queensland 9726, Australia (e-mail: C.Hacker@mailbox.gu.edu.au).

Digital Object Identifier 10.1109/TE.2004.824843

In introductory undergraduate electrical engineering courses, students are required to understand digital logic design concepts. Students acquire the knowledge of how to design initially a digital logic circuit, allowing them to solve, for example, the following semi-real-life problem:

> "An alarm is required to activate if an intruder is detected from a window breaking, a pressure pad signaling, or a movement detector signal. However, the alarm must not activate if the person comes in the door, which is detected by the movement detector and pressure pad signaling together."

Students are requested to develop a combinatorial digital logic circuit that will perform this task. In this process, the truth tables must be derived and then solved into the most efficient Boolean algebra expression by applying the Karnaugh map, the Quine–McCluskey, or other algorithms. This expression results in the final digital circuit diagram, which contains the logic gates and connections obtained by the minimization process.

Many useful computer programs are available that achieve various aspects of digital logic design. They range from simple simulators, to specific teaching tools, to advanced and specialized software. The authors tested several of these packages for the purpose of introductory circuit design teaching. These packages often perform the functions traditionally required in undergraduate courses, for example, Karnaugh maps and Quine–McCluskey and Espresso minimizations. What makes WinLogiLab distinct is that it integrates all necessary functions into one tutorial suite.

Several programs can perform the simulation of a digital circuit, e.g., Logic Works [1], PSpice Mixed Mode Simulator [2], and Micro-Cap V [3]. Other programs can emulate the functions performed in a digital electronic laboratory, e.g., WinBreadboard [4] and Electronics Workbench [5]. The authors also looked at programs written to provide the teaching of digital logic circuits in the style of an electronic book, e.g., Digital Technology Learning Package [6] and Digital Logic Tutor I [7]. There are others that electronically design certain segments of digital logic, such as the Espresso Logic Minimizer [8]. A range of specialized designed software is available, which is developed for proprietary equipment, such as Programmable Logic Devices (PLD) and Field Programmable Gate Arrays (FPGA). They cover various aspects of digital logic design and simulation, e.g., Hardware Description Language (HDL) programs, such as Mentor Graphics [9] and Cadence [10], as well as software designed for ASIC/FPGA hardware, such as Xilinx [11] and Actel [12]. These programs are very efficient at designing very complex digital systems, but they are mainly for PLD and FPGA design.

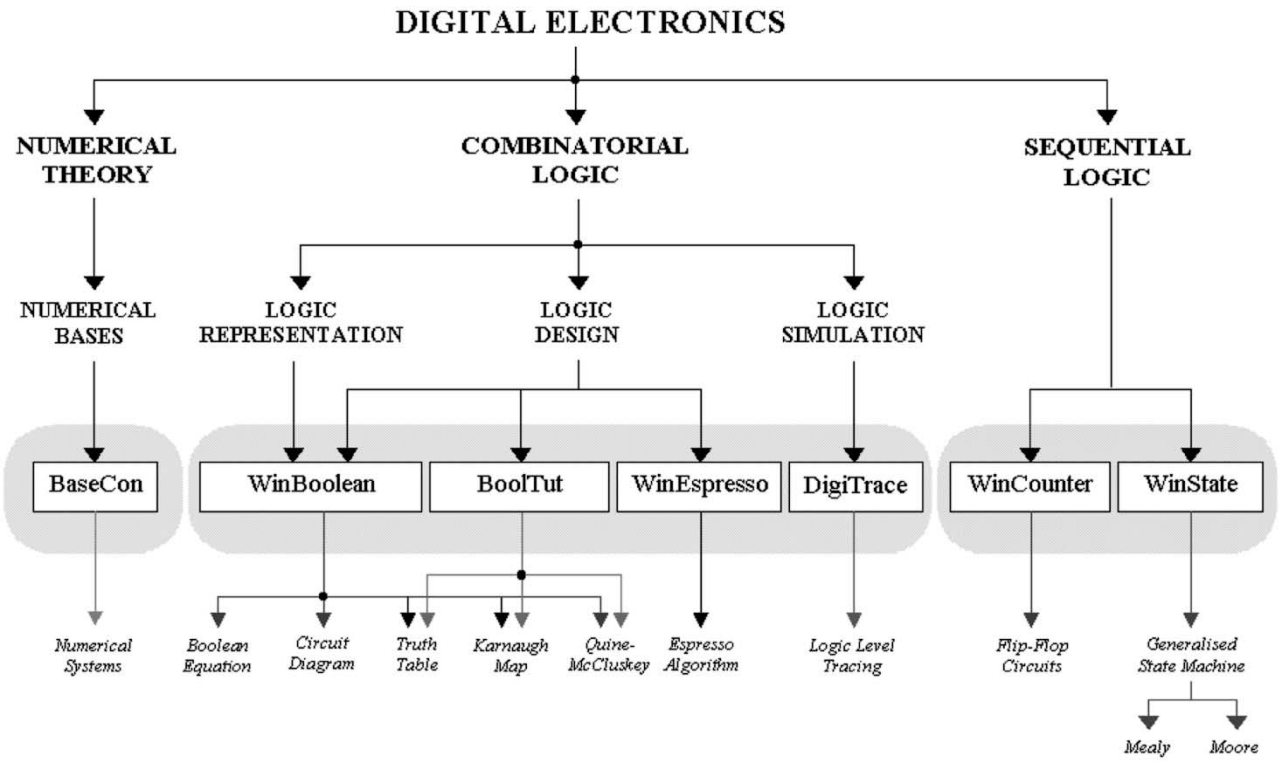

Fig. 1. Structure of WinLogiLab interactive computerized teaching suite.

Undoubtedly, these programs can be used in teaching at different levels, and some are perhaps better suited for specific purposes than others. For example, the HDL programs (such as Mentor Graphics, Cadence, Xilinx, and Actel) are very advanced design and simulation tools and are, indeed, successfully being used in teaching in more advanced courses; however, they are beyond the purpose of an introductory course because their complexity would detract the pupils' focus and attention. The size and complexity of these programs usually require specific training to learn a program's operation. Professional computer-aided design (CAD) tools can and have been successfully applied in undergraduate learning in specifically integrated curricula [13] and conventional courses. However, with student editions' availability either limited or unavailable altogether, the price and licensing requirements of such programs can be justified only for more advanced courses. For introductory courses that typically are attended by a high number of students, the high cost may not be cost-effective or within reach, in particular, when the purpose is to gain fundamental process understanding.

From search and experience, the authors found that MS-Windows-compatible teaching software was desirable. It would cover the complete introductory digital logic design process in an interactive way. The software was required to demonstrate the link between Boolean algebra, truth tables, logic circuits, and minimization techniques of combinatorial logic to junior students. In addition, the software was required to extend these fundamentals into sequential circuits to design flip-flop counters and finite-state machines (FSMs). The software also needed to be user friendly, and thus very intuitive to use, while producing formal presentations and realistic design outputs.

Most currently available digital logic software focuses on simulation rather than design because simulation is based on defined principles and offers greater commercial opportunity. The absence of software to perform digital logic design could, most likely, be attributed to the highly theoretical nature of the design, commonly resulting in very subjective designs. The programming of such subjective tasks is complex and must be elementary enough to enable automation.

II. The WinLogiLab Interactive Tutorial Suite

To address the distinct absence in software for the teaching of introductory digital logic design, the authors have developed WinLogiLab. WinLogiLab is an interactive MS-Windows-compatible computerized teaching suite to aid in the teaching of combinatorial and sequential logic design. Its emphasis lays in demonstrating to novice students the steps of elementary circuit design in a seamless transition sequence and in an interactive visual way.

The structure of WinLogiLab, including its major modules, is depicted in Fig. 1. Its major clusters are the combinatorial logic and the sequential logic modules. There is also a small cluster for introduction to Numerical Theory.

The modules in the combinatorial logic are WinBoolean, BoolTut, and WinEspresso. These modules cover logic gates, Boolean algebra, truth tables, and logic minimization techniques with Karnaugh maps and the Quine–McCluskey and Espresso algorithms. The sequential modules are WinCounter and WinState, which cover state counter design and the design of general-purpose FSMs.

Special attention was paid to implement WinLogiLab in a user-friendly environment. A tutorial with short pop-up instructions provides additional guidance to new students.

The components of the WinLogiLab tutorial suite can be used in two modes. One mode is a step-by-step, self-paced, set tutorial that guides the student through any of its tasks and serves as an introduction because it provides specific explanatory information. Alternatively, students can experiment by designing their own circuits as independent self-paced exercises in which students use their own input. Both modes offer ample help and feedback to the student in a self-paced learning mode, in particular, because the students can see an immediate relationship in cause and effect by what happens on the screen and, in addition, from pop-up feedback comments. This level of understanding would be difficult in conventional Boolean algebra teaching. Students do not need to go through all steps until the final circuit is designed. Input can occur at several points throughout the circuit design sequence, by either a manual input or a file saved from an earlier session.

In addition, a self-test is available to the student. In this test, a student is requested to answer a number of computer-generated multiple-choice or short-answer questions on digital logic. For example, the student can be requested to enter the Boolean equation for a given randomly generated circuit diagram. Similarly, other randomly generated questions on circuits, equations, truth tables, or minimization techniques are possible. A true/false feedback is given immediately after each answered test question, and a total score of the number of questions that were correctly answered is given at the end. WinLogiLab has also proven to be a useful lecturing aid when projected on a large screen.

The current state of development of the WinLogiLab tutorial suite can be obtained on CD from the authors or downloaded from the Griffith University School of Engineering web page at http://www.gu.edu.au/school/eng/mmt/MMTdownlds.html

In the following sections, the authors will briefly explain the features of each of the modules.

A. WinBoolean

WinBoolean is designed to show the link between the equivalent digital logic forms of logic gates, Boolean algebra, and truth tables. All three techniques can be used interchangeably; thus, a student in Digital Logic is required to be familiar with converting from one to any other of these three equivalent forms. Therefore, WinBoolean allows input in any of the three logic forms and automatically converts it to any of the other equivalent forms. Figs. 2 and 3 show sample screen images for input by circuit diagram and equation, respectively. The entered user data can then be simplified by a Karnaugh map or the Quine–McCluskey algorithm [14]. Fig. 4 shows an example of Quine–McCluskey output of WinBoolean. The entered user data can then be simplified by a Karnaugh map, a Quine–McCluskey algorithm, or Espresso algorithm to produce a minimized logic diagram.

B. BoolTut

BoolTut is designed to provide an interactive tutorial on the Karnaugh map or the Quine–McCluskey minimization process. The program operates with a user-supplied truth table and displays an animated step-by-step presentation on each individual

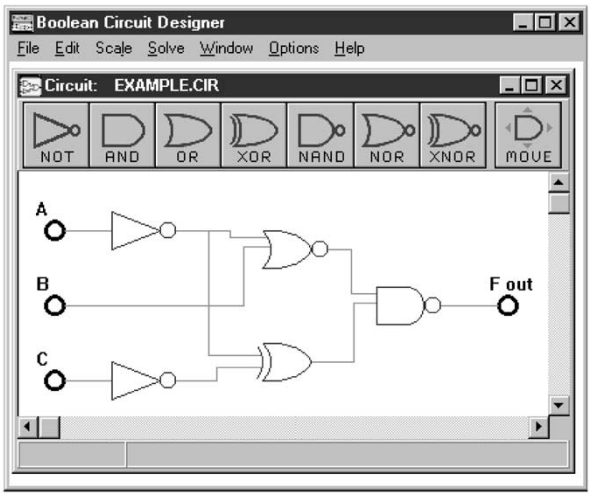

Fig. 2. Example of input as circuit diagram.

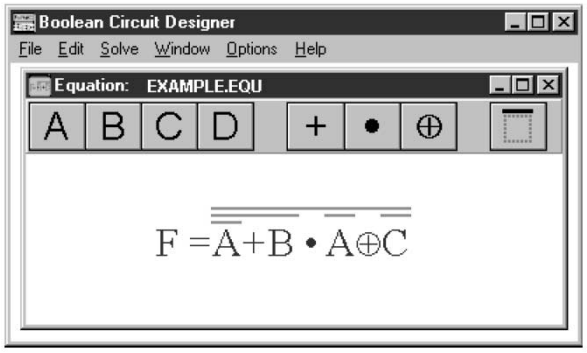

Fig. 3. Example input as Boolean equation.

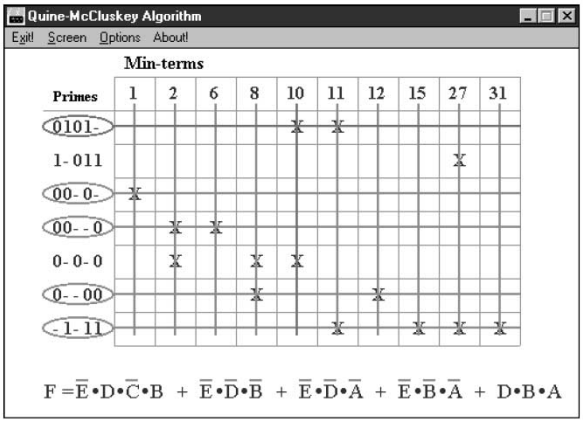

Fig. 4. Example of a final Quine–McCluskey output.

stage of the minimization process [15]. It allows the user to enter a truth table via the keyboard or mouse clicks, or alternatively, the program can randomly generate a truth table. The user then selects the minimization process to be demonstrated by either the Karnaugh map, the Quine–McCluskey algorithm, or Espresso. The Karnaugh map process is valid for up to four inputs, while the Quine–McCluskey process will allow up to 12 input variables. Although it is clear that a Karnaugh map can be used for more than four variables, a higher number of variables

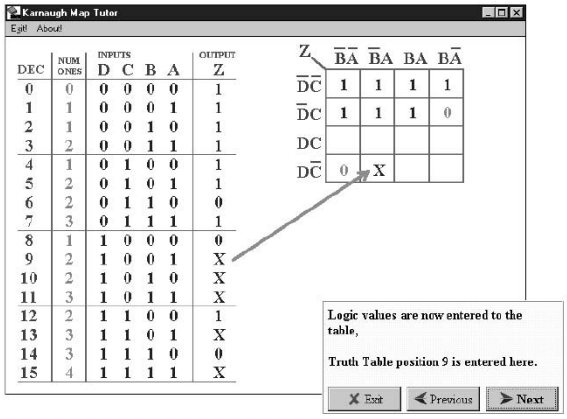

Fig. 5. Sample interactive Karnaugh map generation.

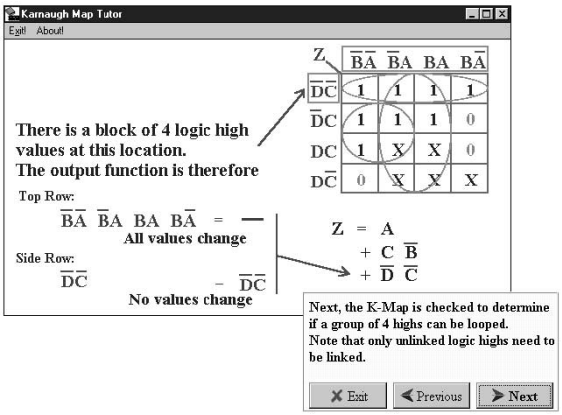

Fig. 6. Sample interactive Karnaugh map minimization.

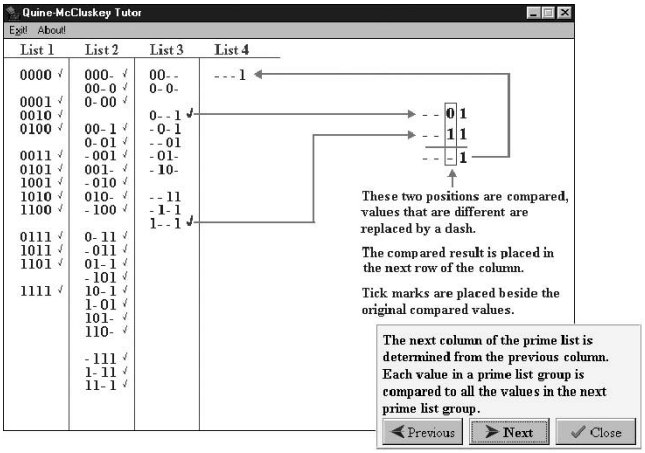

Fig. 7. Example of interactive minimization of the Quine–McCluskey Prime List.

noticeably slows the process down. There is a tradeoff with how much can be put on a computer screen without the need for constantly scrolling up and down to see one single diagram. For the purpose of understanding the process of deriving a Karnaugh map, the authors considered it more important to show the two modes, the truth table, and the Karnaugh map together on the screen to visualize interactions, rather than a high number of variables. For more complex minimizations with multiple outputs, the students would be using the Espresso algorithm. This technique is explained in the next section.

Figs. 5–7 show selected examples of different stages of the screen animations as they appear in the interactive tutorials.

C. WinEspresso

WinEspresso provides an alternative method for simplifying Boolean truth tables, using a more efficient approach. The rationale is that there are two main fields in Boolean truth table minimization: the *exact* and the *heuristic* techniques. The exact techniques use thorough Boolean algebra operations resulting in an optimal minimized solution. The heuristic techniques use rule-based approximations, which result in near-optimal minimized solutions. Brayton *et al.* discuss different logic-minimizing techniques and introduce or expand the heuristic techniques [16], [17].

The two exact techniques for truth table minimization are the Karnaugh map and the Quine–McCluskey algorithm. Both algorithms are implemented in the WinBoolean and BoolTut modules. These exact minimization techniques, in particular the Quine–McCluskey algorithm, require the generation of all possible solutions (known as primes) and then the finding of the best solution by a minimum cover procedure. In exact minimization techniques, when the number of inputs to the function becomes large, the computing process required increases exponentially. This process is very inefficient because of the extreme number of iterations that are required to generate all the possible solutions. For this reason, the process becomes impractical or impossible to implement [16].

In contrast, truth table minimization techniques have concentrated on heuristic solving methods, which avoid computing all possible results (the primes) [16], [18]. Instead they successively modify a given result until a suitable stopping criterion is met, as depicted in Fig. 8.

The minimization is thus accomplished more efficiently and rapidly. One should note that for inputs up to seven variables, Espresso always finds the absolute optimal solution, but for higher numbers of variables, it will find a near-optimal solution, not necessarily the absolute optimal solution.

As a component in WinLogiLab, this method provides practicing fast simplification of truth tables with up to 12 input and eight output variables simultaneously [19]. It allows for the visual circuit schematic output as well as the associated Boolean function.

D. DigiTrace

The DigiTrace module provides the final testing of a designed logic circuit by partially simulating the operation of the circuit. The simulation is performed by displaying the trace of the digital logic signals throughout the circuit, which provides sufficient visual indication of the circuit's function [20].

DigiTrace allows the user input of a digital circuit or the loading of a circuit developed by any of the other combinatorial WinLogiLab modules. The DigiTrace module then displays the visual trace of the logic levels through each component of the circuit. Simulated light-emitting diode (LED) lights indicate the logic levels. A red light represents a logic low (0), and a green light represents a logic high (1). The simulated circuit can have

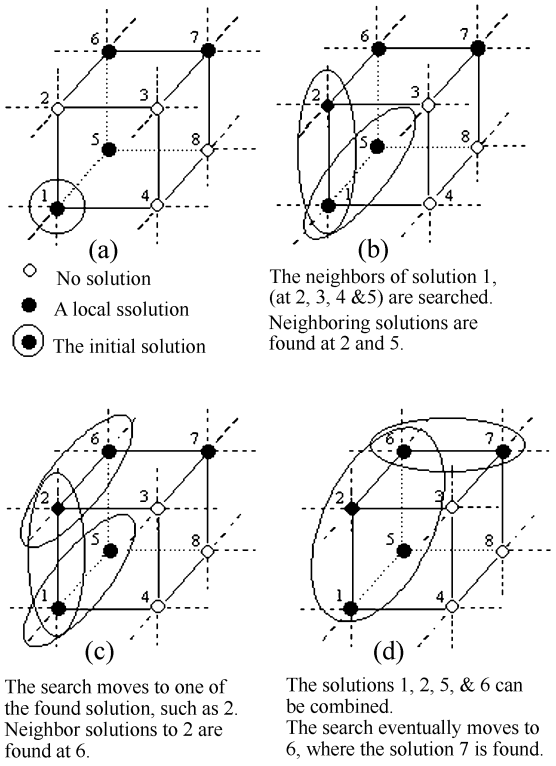

Fig. 8. Example of an Espresso heuristic procedure.

one to four inputs, and these inputs can all have varying logic levels. Thus, the logic levels throughout the circuit can be traced through for all possible combinations of the input logic levels.

The tracing can be done in one of the three following modes:

1) setting manually the values of the inputs in a circuit diagram (via mouse clicks) on a logic-level input control and observing the result of a simulated LED light at the output;
2) tracing all logic inputs where the result appears as an array of simulated LED lights;
3) tracing all waves displays a timing window showing multiple square-wave pulse output, similar to the outputs obtained from many commercial and shareware logic simulation packages.

E. Sequential Logic and WinCounter

Following the natural progress of lectures on introductory logic and combinatory circuit stages, sequential circuits are also included in the WinLogiLab.

Sequential circuits have a primitive internal memory, assembled with combinatorial logic to form a counter circuit. The counter circuits form a branch of FSM. In WinLogiLab, the sequential logic offers the state counter design with flip-flop components and the design of general-purpose FSM.

One might argue that flip-flops are not used much nowadays; however, they do have a pedagogical benefit. This benefit comes because flip-flop memory circuits can be wired in a number of modes. These modes are the data mode and toggle mode. A sequential logic circuit can utilize a flip-flop that is wired in any of these modes. The toggle mode is commonly used in counter

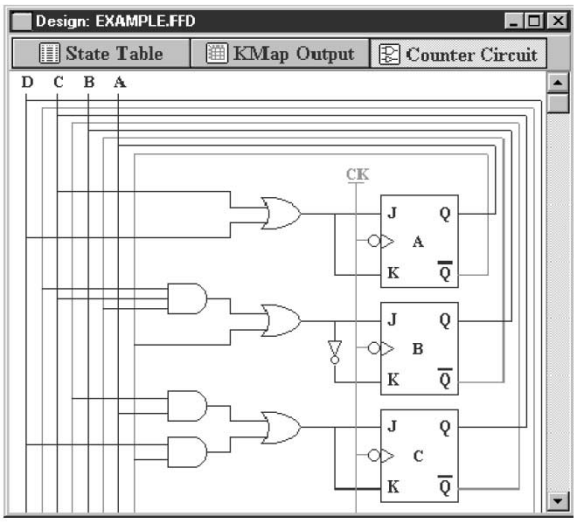

Fig. 9. Sample result of the designed circuit to implement the counter.

applications, where transitions are frequent, such as continuous changing counts. The data mode is more commonly used in data storage applications, such as random access memory (RAM) circuits.

The WinCounter program aids circuit analysis and design of sequential state machine counters. State machine counters form the fundamental process of most logic control applications, including electronic timers, electronic clocks, remote controls, and industrial process controllers. WinCounter enables the circuit design, or circuit analysis, of sequential state machine counters. The counters are implemented with data- or toggle-mode flip-flop components and associated combinatorial logic gates [21].

The WinCounter software provides an interactive tutorial to exercise and understand flip-flop counter circuits. Its input can be either as an input counter sequence or as a circuit. Its output produces a state diagram. A counter circuit is designed from a desired count sequence that is entered by the user. This count sequence is input into a state transition table; the table is then minimized by the Karnaugh map process, resulting in the final logic circuit that will perform the required count sequence. An example of the screen images of this process is shown in Fig. 9.

An existing counter circuit can also be analyzed by the program. In this case, the user is required to input the known counter schematic. The program will then determine and display the Boolean functions to each of the flip-flop components and derive the state transition table of the circuit. Finally, a state transition table then models the final count sequence of the analyzed counter example.

F. WinState

The WinState tutorial software assists undergraduate students in understanding the function of general-purpose FSMs, in particular, the Mealy and Moore machines. The animated computer screen is ideal for conveying the required design and analysis procedures, while enabling students to better visualize the functioning of the FSM. In general, the purpose of FSMs as a modeling tool goes beyond the machines that are used in digital control logic systems. For example, FSMs can be used for lan-

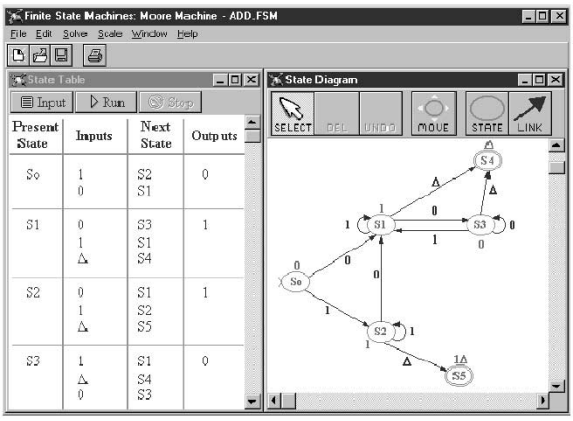

Fig. 10. Example of WinState input by constructing the FSM as a graph, and its automatically updated state transition table.

guage parsing, mathematical processing, communication network analysis, data encryption, and decryption [22].

FSMs operate on a recognizable (or legal) input string of symbols (not necessarily binary), which continuously modifies its internal state and produces a corresponding output symbol string. FSMs can be either Moore or Mealy types. In a Moore machine, the derived outputs depend solely on the present internal state of the FSM. In a Mealy machine, the derived outputs depend on the present state and the applied inputs [23]. WinState allows both the design and the analysis of Mealy or Moore general-purpose FSM, operating with user-provided input data [24].

FSMs can be represented in two modes: in a state transition table, which lists tables of present state, inputs, next state, and outputs, and in a directed graph, the state transition diagram. This diagram consists of nodes (circles), representing the states, and linking arcs, representing the transitions or next-state functions. Input symbols are placed above the arcs, while output states are placed either next to the node (for a Moore machine) or underneath the input symbol (for a Mealy machine). A linking arc is drawn for each possible input symbol, linking one node with another.

WinState allows users to enter their own FSM design. States and arcs can be picked from the menu and placed. By clicking on the arrow, one sees a pop-up box that prompts for input or output. Fig. 10 shows an example of a user input by constructing the FSM as a graph, and its equivalent state transition table is automatically updated while the graph is being drawn.

WinState allows a range of recognizable (legal) input symbols, not just 0/1. This range purposely extends its use beyond digital circuit design. A typical FSM learning example is the parsing of an input string (not necessarily binary), with a null character (Δ) to mark the end of a string. Once the FSM has been implemented, a user-defined input symbol string, in binary or in an other alphabet, can then be entered for execution. This procedure is equivalent to entering an instruction set or a rudimentary program. The string to be executed is input or edited by the user in the string dialogue window.

This input string is then "executed" on the FSM. To aid the user in visualizing the execution, the current-state and link conditions are highlighted on the FSM with thicker lines for the affected node and arc. This highlighting occurs while the program parses and executes the current input string and produces the output string.

III. Benefits of WinLogiLab and Discussion

There are a number of issues that the authors have specifically addressed. The most important aim was to demonstrate to the students the full transition from the numbering systems to the testing of the simulated circuit by tracing its functioning, where the student can learn by participating. The contents of topics offered in WinLogiLab map to what would be a typical undergraduate course.

A. Self-Study and Alternative-Mode Learning

Students are able to learn at their own pace, "discovering" principles while they experiment. Alternatively, they can step through a set tutorial until they feel confident to use their own input. Pop-up prompts provide guidance and feedback, and "undo" buttons allow for quick amendments.

BoolTut is different from the digital logic tutorial packages tested, as initially explained. The program has improved capabilities in letting the students enter their own initial data. This procedure is in contrast to other tested packages that are currently available that only present one "sample" input, going through the same steps and the same solving process each time the program is run. Other packages could not handle a large number of inputs by using an algorithm such as the Quine–McCluskey process. Again these issues were specifically addressed and resolved in BoolTut, in flexible data input, and demonstrated with step-by-step animations of both the Karnaugh map *and* the Quine–McCluskey minimization process.

The Espresso algorithm is a heuristic technique for Boolean minimization, which has become a widely adopted and extensively discussed procedure that has been incorporated into the teaching curriculum of many university digital logic courses [18]. The Espresso algorithm was developed at the University of California-Berkeley, with its authorship being attributed to Rudell [8], [17]. The UNIX C compiler code for the Espresso algorithm is freely available for downloading from the Design Technology Warehouse, University of California-Berkeley [8].

The UNIX C Espresso compiler code [8] is readily adaptable for compiling to an MS-DOS (text base) executable program. A number of authors have generated MS-DOS executables from this Espresso code, including Espresso.exe by Changwook [25].

The original Espresso code takes as input a text-based Boolean truth table and generates a text-based minimized truth table output. The input truth table is provided from a text file, or as text data from the keyboard, while the minimized output is written to the computer's display.

A student's initial understanding in digital logic design is that a tabulated Boolean truth table is minimized to a logic circuit schematic or Boolean function. The "Berkeley standard PLA" text format of the original Espresso C code is adequate for representing a text-based input truth table and the corresponding text-based minimized output. However, its cryptic text-based format does not match the standard Boolean truth table input,

nor does it correspond to the logic circuit or Boolean function output.

These deficiencies were overcome by reimplementing the algorithm as Microsoft-Windows TM-compatible software, called WinEspresso. This new graphical user interface allows for student input of a standard visual tabulated Boolean truth table and, after minimization by the Espresso algorithm, allows for the visual circuit schematic output and the associated Boolean function.

Implementing the WinLogiLab in the MS-Windows environment gives further benefits. First, it modernizes the code for the now standard MS-Windows 95, 98, and NT computer teaching laboratories. The implemented WinEspresso code can currently operate on all MS-Windows platforms, from Windows 95 onwards. In addition, the windows graphical user interface enables the displaying and printing of all forms of truth table inputs and circuit schematic outputs. In addition, the program is fully mouse-click driven and includes helpful menu commands and toolbar buttons, with quick user-friendly operations, such as open, save, print, and copy. Furthermore, the software enables the user to use the menu items, or header toolbar buttons, to continuously switch between the input Boolean truth table and multiple output circuit displays. Finally, full copying and pasting capabilities, with the MS-Windows clipboard, enables the transfer of truth tables, circuit schematics, and Boolean functions to other applications, such as word processors. This latter capability was either not possible or very difficult to achieve with DOS text-based code.

To assess the usefulness of the tutorial suite, the author/lecturer administered a survey to the students of the introductory digital logic subject. The survey was repeated for two consecutive years. Overall results revealed that the majority of students found the tutorial suite useful and user friendly, with an overall rating of 4.35 out of 5.

The questionnaire also posed three open-ended questions: 1) what were the best features of the software; 2) what were the worst features; and 3) what improvements would the students suggest. The students' statements about the best features included that it was a good study aid and that the tutorials made it easy to design circuits, to solve Boolean equations, and to understand Karnaugh maps. Students also appreciated that results or outputs could be easily copied and pasted into other applications, such as reports for their assignment or tutorials.

Student statements on the worst features and suggested improvements were similar. Suggestions included adding an option for ANSI/IEEE standard logic circuit symbols, adding an option for the circuits to snap to a grid, allowing for higher numbers of input variables, and making improvements in the style and wording on various tutorials.

IV. Conclusion

This paper outlines the characteristics and implementation of WinLogiLab, a computer-based tutorial suite to aid in teaching the introductory design of digital logic circuits. WinLogiLab fills a perceived gap in computer-aided teaching by providing tutorials that link Boolean algebra and digital logic circuits. The tutorials help students in digital logic design to experiment with different minimization techniques. These minimizations can be performed by exact methods, using Karnaugh maps and Quine–McCluskey algorithm, or by a computation-efficient heuristic approximation, using the WinEspresso algorithm. The paper also explains the implementation of a computer-based tutorial for designing and simulating counters and general-purpose FSMs. WinLogiLab covers a range of topics typically taught in an introductory digital logic course, i.e., combinational and sequential circuits. It employs a graphical user interface in an MS-Windows environment, which provides students with interactive, visual, and user-friendly software. Both the user friendliness and ease of use helps students to understand better the subject material. A student evaluation survey produced favorable responses. The survey also provided valuable suggestions for further improvements in this digital logic design teaching and learning aid.

References

[1] Capilano Computing Systems, Logic Works Version 3.0, Benjamin Cummings, Redwood City, CA, 1995.

[2] Orcad/MicroSim Corporation. (2002, Oct) WinPSpice Mixed Mode Simulator—Version 9 (Student Version). [Online] Available: http://www.orcad.com

[3] A. Thompson, T. O'Brien, and B. Steele. (2002, Oct.). Micro-Cap V (Version 1.3), Spectrum Software. [Online] Available: http://www.spectrum-soft.com

[4] Yoeric Software. (2002, Oct) WinBreadboard Version 1.1 (Demo Version). [Online] Available: http://www.yoeric.com/

[5] Interactive Image Technologies. (2002, Oct) Electronics Workbench 3.0E. Distributed by Applied Electro Systems, Australia. [Online] Available: http://www.interactiv.com

[6] R. J. Delahoy, *Digital Technology Learning Package*. North Dandenong, Australia: R. J. Delahoy, 1995.

[7] G. N. Foster, *Digital Logic Tutor I*. New York: MacMillan, 1994.

[8] UCB Design Technology Warehouse. (2002, Oct.) Espresso: Logic Minimization Algorithms. UNIX compiler code, EECS Dept., University of California-Berkeley. [Online] Available: http://www.cad.eecs.berkeley.edu/Software/software.html

[9] Mentor Graphics. (2002, Oct.) QuickSim & ModelSim HDL Simulator, Wilsonville, OR. [Online] Available: http://www.mentor.com

[10] Cadence Design Systems. (2002, Oct.) Ambit BuildGates, Irvine, CA. [Online] Available: http://www.cadence.com

[11] Xilinx, Inc.. (2002, Oct.), San Jose, CA. [Online] Available: http://www.xilinx.com

[12] Actel Corp.. (2002, Oct.), Sunnyvale, CA. [Online] Available: http://www.actel.com

[13] N. L. V. Calazans and F. G. Moraes, "Integrating the teaching of computer organization and architecture with digital hardware design early in undergraduate courses," *IEEE Trans. Educ.*, vol. 44, pp. 109–119, May 2001.

[14] C. Hacker and R. Sitte, "Development of a computer program, to electronically design digital logic circuits using Boolean algebra," in *Proc. 9th Annu. Australasian Association Engineering Education (AaeE97)*, Dec. 1997, pp. 353–357.

[15] ——, "A computer based tutorial, for demonstrating the solving of digital logics circuits," in *Proc. 10th Annu. Australasian Association Engineering Education (AaeE98)*, Sept. 1998, pp. 509–519.

[16] R. Brayton, G. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*. Boston, MA: Kluwer, 1984.

[17] R. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang, "A multi-level logic optimization system," *IEEE Trans. Computer-Aided Design*, vol. C-6, pp. 1062–1081, Nov. 1987.

[18] M. Theobald and S. M. Nowick, "Fast heuristic and exact algorithms for two-level hazard-free logic minimization," *IEEE Trans. Computer-Aided Design*, pp. 1130–1147, Nov. 1998.

[19] C. Hacker and R. Sitte, "Implementing the "Espresso—Two level logic minimizer" algorithm in the MS-Windows environment," in *Proc. 2nd Asia-Pacific Forum Engineering Technology Education, UNESCO Int. Conf. Engineering Education (UICEE99)*, July 1999, pp. 124–127.

[20] ——, "A computer based teaching program for the tracing of logic levels in a digital circuit," in *Proc. 4th Annu. UNESCO Int. Conf. Engineering Education (UICEE2001)*, Bangkok, Thailand, Feb. 2001, pp. 509–519.
[21] ——, "A computer based teaching program for the design and analysis of digital counter circuits," in *Proc. 3rd Annu. UNESCO Int. Conf. Engineering Education (UICEE2000)*, Feb. 2000, pp. 225–228.
[22] J. L. Gersting, *Mathematical Structures for Computer Science*. New York: Freeman, 1987.
[23] A. M. Tenenbaum and M. J. Augenstein, *Data Structures Using Pascal*. Englewood Cliffs, NJ: Prentice-Hall, 1981.
[24] C. Hacker and R. Sitte, "Implementing finite state machines in a computer based teaching system," in *Proc. SPIE Int. Symp. Microelectronics Micro Electro Mechanical Systems (MICRO/MEMS)*, Gold Coast, Australia, Oct. 28–29, 1999, pp. 100–117.
[25] P. Changwook. (1998) Espresso.exe. Shareware, Seodu Logic-Electronic Technology Institute. [Online] Available: http://eda.seodu.co.kr/~chang/download/espresso

Charles Hacker received the Dip.Eng. degree and the B.App.Sc. degree in physics from the University of Central Queensland (UCQ), Queensland, Australia, in 1987 and 1990, respectively, the Grad. Dip. Sc. degree in medical physics from the Queensland University of Technology (QUT), Queensland, Australia, in 1995, and the M.Phil. degree in electrical engineering from Griffith University, Queensland, Australia, in 2003.

He started working as a Demonstrator, Tutor, and Sessional Lecturer at the Physics Department, UCQ, in 1989. In 1991, he became a Lecturer in Electronic Engineering with the School of Engineering, Griffith University. His teaching areas include electronics, microprocessors, physics, and computer programming.

Renate Sitte (S'90–M'95) received the Systems Engineering degree (Ingeniero de Sistemas) from the Universidad de Los Andes, Los Andes, Venezuela, in 1985 and the M.Phil. and Ph.D. degrees from Griffith University, Queensland, Australia.

She is a Faculty Staff Member in engineering and information technology, Griffith University, where she has been teaching computing, software engineering, and discrete mathematics since 1986. Her research is in modeling and simulation, and she has one major ongoing project in virtual-reality prototyping microelectromechanical systems (MEMS).

Dr. Sitte is currently Vice-Chair of the IEEE Queensland Section and Chair of the IEEE Computer Chapter. She is also a Member of the Institute of Engineers Australia, contributing to the development of the ISO standards for software engineering.