A Visual Learning Tool for Effective Student Engagements in Computer Engineering Education: Case of Digital Logic Instruction

LUCIEN NGALAMU
Lewis University,
Romeoville, IL, USA

Abstract - This paper presents a work in progress that aims at designing a visual tool with multiple modules that can be used for effective tutoring of the core topics of computer engineering. The first aspect of the design that was implemented and tested in a classroom setting is related to a visual tutor that can be used to reinforce the learning abilities of students pursuing an introductory course of Digital Logic. The development is based upon Adobe Animate (formerly Macromedia Flash MX), which is used to create the necessary text, graphics, and interaction needed for the application. The results of three-year surveys indicated that the students found the tutor to be both useful and helpful and that it had encouraged them to learn more about Digital Logic. The use of animated and highly visual teaching aids was therefore proven to be effective in assisting students in their study. The visual tutor is called ”SmartStart”.

Keywords -Visual Tutor, Adobe Animate, Smart Start, Digital Logic

Received: September 29, 2019. Revised: April 3, 2020. Accepted: April 27, 2020. Published: May 5, 2020.

This paper has been proposed by the Associate Editor:
Prof. Klimis Ntalianis
kntal@image.ntua.gr
that coordinated the peer review process with the high
standards of the WSEAS

1. Introduction

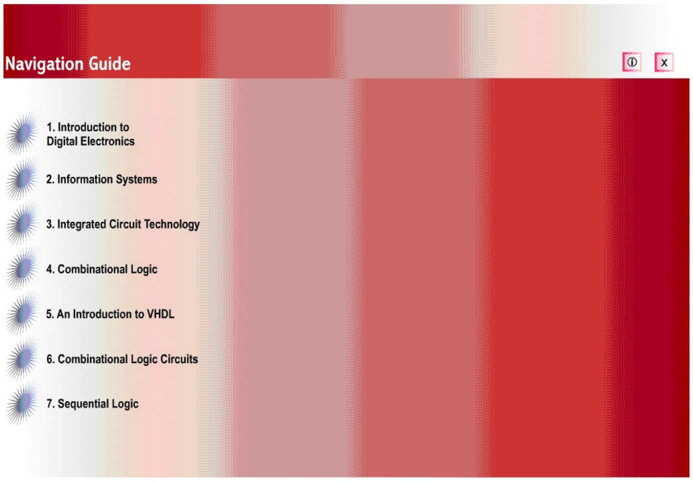
Tutoring is traditionally represented by a form of one-on-one interaction with students. In this paper, the one-on-one interaction concept was adopted and transformed into a Flash-based teaching aid. Multimedia is considered to be a powerful tool in education; Long [1] states that People retain 10% of the information they see; 20% of what they see and hear; and 80% of what they see, hear and do. These statistics present a strong argument for interactive learning via multimedia. Long’s postulate is of particular significance at the tertiary level as the tertiary experience is often criticized for limited student/teacher interaction, due to large student enrollment and heavy course loads that have to be taught in a limited time period. Less attention is paid to each individual student, and weaker students tend to fall behind, resulting in poor grades and high failure rates [2], [3]. In engineering disciplines, students’ learning abilities can be improved by adding to lectures additional tutorials with illustrative animations, simulations, or further explanations with visualizations. There has been an extensive use of visualization learning tools to teach engineering courses. Duran et al. [4] demonstrate that when teachers incorporate visual simulations of real-life into their classroom activities, students tend to have a much better learning experience. Specifically in the area of digital logic instruction, a relative number of work has been conducted, which consists on visualization and animation of digital logic design.

Iskander [5] has introduced the use of visual simulations to deepen the understanding of electromagnetism. Nguyen et al. [6] present a visual tutoring framework with interactive visualizations which help students improve their understanding of complex subjects in certain engineering courses.

Shoufan et al. [7] present a web-based education platform for the visualization and animation of the digital logic design process. This includes the design of combinatorial circuits using logic gates, multiplexers, decoders, and look-up-tables as well as the design of finite state machines. Hassan et al. [8] describe new software called Easy Electronic Software (EES), that was developed to allow simpler solutions for students taking the Digital Logic Design (DLD) subject especially on basic electronic components. The software provides a calculation method for estimating the value of DLD subject components, which are resistance and inductance. Stanisavljevic et al. [9] present a system for digital logic design and simulation (SDLDS). The system consists of three modules for design, simulation and evaluation. The design can be carried out either starting from a formal description or by instantiating and connecting library modules.

Smartstart is a visual tool that is being developed to reinforce the understanding of digital logic. It provides an interactive tool which is oriented towards student self-learning. Smartstart is reflective of the constructivist-based style of teaching, where the student is the main agent responsible for his or her own learning. One of the important features of this tool resides in its use as a learning aid tool. Its impact on the learning abilities of first-year students is clearly demonstrated by the statistics mentioned in this paper. The screen-shot of Fig. 1 shows the topics cover by Smartstart. .

The topics’ material was sourced mainly from CPEN21000 notes [11] and Wakerly’s text [12]. The topic relating to VHDL was not covered in the project, due to the existence of EVITA [13]. EVITA, however, was made available via a



Navigation Guide

1. Introduction to Digital Electronics
2. Information Systems
3. Integrated Circuit Technology
4. Combinational Logic
5. An Introduction to VHDL
6. Combinational Logic Circuits
7. Sequential Logic

Fig. 1. Screen-shot of the Navigation Guide.

link provided by the tutor. The section that covers Information Representation, was developed as a QT-based [14] Software and integrated into SmartStart.

2 Design

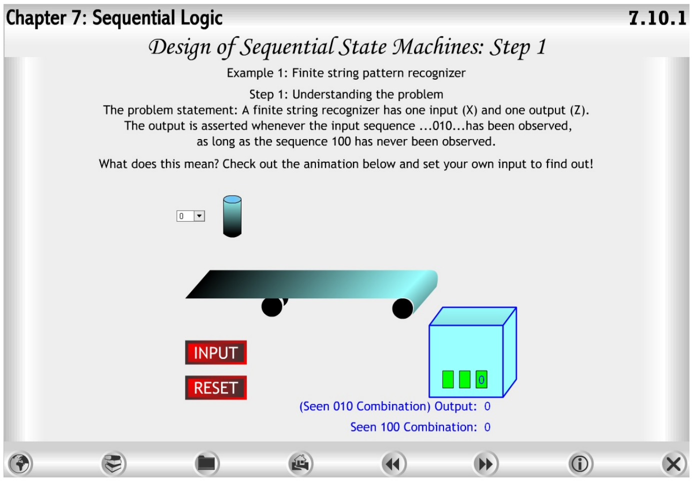
The two main aspects of the design are conceptual and technical. The design of individual sections is followed closely by implementation in Flash. The conceptual design of the tutor takes into consideration the needs of the student. The tutor seeks to address the different learning ability levels by catering to those who have the most need for one-on-one assistance. Students are allowed to learn at their own pace so that those who take slightly longer than others to grasp the material and to process the information, are given the opportunity. It is assumed that the student has some basic knowledge of electronics; the tutor was then built from this point upwards.

The majority of these examples are original and developed to enhance the theory from the notes and the tutor. In addition, practical examples are given where needed e.g. tips for setting up a basic circuit on a breadboard, BCD decoders and seven-segment displays, and rotary encoders. A variety of exercises is provided for the student to try on their own, with the option to view the answer with the click of a button. These exercises are usually given for topics where certain mechanical procedures are required later on. Motivation by means of hints (displayed by scrolling the mouse) is provided as the student is able to successfully complete each step and is encouraged to continue with other exercises. Some exercises are based on previously covered work thereby reinforcing earlier concepts. These exercises are automatically graded at the end, and students are provided with meaningful feedback. Detailed explanations of each topic, rather than general overviews were given, usually incorporating definitions and descriptions. This approach brought the knowledge down to a more basic understanding level and addressed any confusion the student may have, of the simpler concepts. The essential areas of each topic were also identified and the chapters sub-divided along these lines according to Bloom's Taxonomy. Explicit instructions are given, indicating to the user what action should be taken. A help file was also included, describing the navigational buttons and how the tutor is to be used. The diagrams, charts, tables, and graphics are all required to be as clear and easily visible, with short, easily-read labels indicating the main points.

Students should be able to use the tutor without any training sessions. The design of the graphical interface must, therefore, be simple and user-friendly. Each section of the tutor is given color-coded backgrounds and buttons for easy identification. Different colors are also associated with certain elements e.g. combinational gates were given a standard fill color. This allowed for easy identification and served to reinforce the visual impact. Self-explanatory tags are given to the navigational buttons, and clear symbols are used to communicate their meaning. Buttons were given an easily identifiable fill to distinguish them from other elements, along with a different rollover state. By using a common set of navigational buttons, the buttons only had to be understood once. For each function identified below, a button was assigned. The system is considered using event-oriented decomposition [7]. In the event-oriented approach, there were certain common events that were identified. Once applicable, the user should be able to perform these specific functions while perusing the material in the tutor. In order to evaluate the response of the students to the tutor, a feedback form was developed. Several aspects of this questionnaire was adopted from Questionnaire Design [8]. No personal information was requested from the students, and an assurance of confidentiality was ensured. It is well established that this technique encourages honest responses. A combination of open-format and closed-format questions was utilized. Generally, open-format questions do not elicit specific information, and respondents are less inclined to give an answer when they have to record sentences or statements. However, this was suitable for determining opinions and views. Closed format questions provide a fixed number of options, from which a single one is chosen, similar to multiple-choice. This was favored more than open-format questions, in that users are more inclined to select one of a possible set of answers. The choices provided were required to be as objective as possible, to avoid leading the respondent to select a specific one. A neutral option was provided, catering for those who were unsure, or unwilling to provide an answer. The options were modeled after lecturer-assessment forms. A quantitative value was used to obtain an overall rating of the tutor.

3 Finite States Machine Design

This section is based on an example of the finite string pattern recognizer, recognizing an output of 1 for the 010 combinations, given that the combination 100 is not seen. This is encompassed in six steps on separate pages. The first two of these steps are the least mechanical and are explained using detailed animation, while the latter four utilize simple diagrams. In order to understand the problem given and to simplify the given requirements, a model was presented, abstracting the information into a visual form. It is based on an example of the finite string pattern recognizer, recognizing an output of 1 for the 010 combinations, given that the combination 100 is not seen. This is encompassed in six steps on separate pages. The first two of these steps are the least mechanical and are explained using detailed animation, while the latter four utilize simple diagrams. In order to understand the problem given and to simplify the given requirements, a model was presented, abstracting the information into a visual form. The operation of the model is shown in Figure 3. Initially, the movie clip itself is created, with the input motion tweeted downwards (falling onto the conveyor), horizontal (sliding across) and downwards again (sliding down to the box representing the memory). This box also contains previous values, represented within three dynamic textboxes. The first frame of this clip accepted inputs from the Combo Box, while in the final frame; the values corresponding to the dynamic textboxes are shifted left to accommodate the input. Thus the variable associated with each textbox was equated to the one to its right, with the rightmost equated to the input. Six variables were needed, three for the memory, one to recall whether the 100 combinations are seen, one for the input, and one for the output. An algorithm is developed to determine the combinations, which would then be checked with respect to these three variables. The second step considers the state diagram, explaining the need for each state. The state diagram is broken up into its constituent parts of states, arrows, and labels. Groups of these are placed on separate layers, in four folders. Each folder matched a particular section of the state diagram. Each 10-frame sequence contains a new set of states, along with the associated arrows and labels, "Play" buttons were also included at these intervals, along with their handlers. The clip was stopped at each point, and only when the button was clicked, would it continue to play for a further ten frames until stopped again. Figures 4-5 gives an overview of the design steps.


Chapter 7: Sequential Logic 7.10.1

Design of Sequential State Machines: Step 1

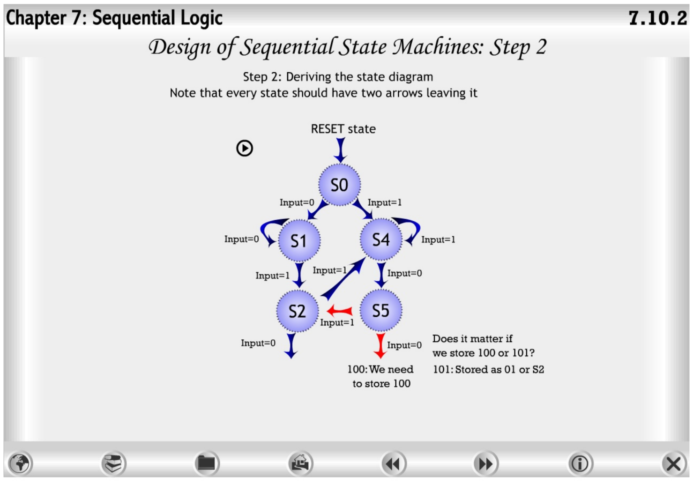
Example 1: Finite string pattern recognizer

Step 1: Understanding the problem
The problem statement: A finite string recognizer has one input (X) and one output (Z).
The output is asserted whenever the input sequence ...010...has been observed,
as long as the sequence 100 has never been observed.

What does this mean? Check out the animation below and set your own input to find out!

(Seen 010 Combination) Output: 0
Seen 100 Combination: 0

Fig. 2. Design of the sequential state machines: understanding the problem.


Chapter 7: Sequential Logic 7.10.2

Design of Sequential State Machines: Step 2

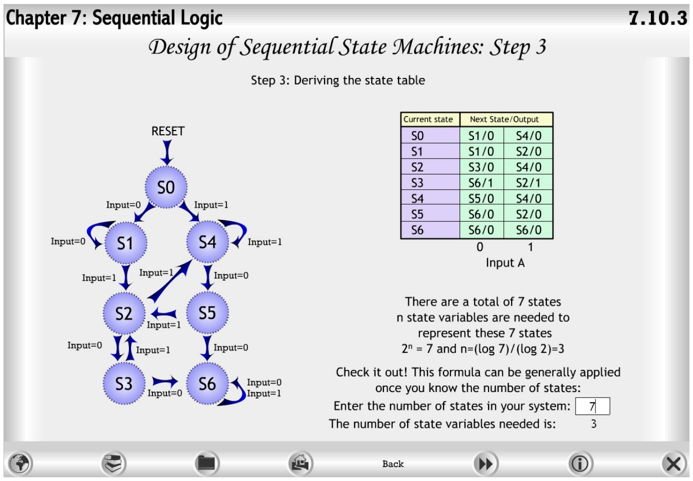
Step 2: Deriving the state diagram
Note that every state should have two arrows leaving it

Does it matter if
we store 100 or 101?

100: We need
to store 100

101: Stored as 01 or S2

Fig. 3. Design of the sequential state machines: Deriving the State Diagram.


Chapter 7: Sequential Logic 7.10.3

Design of Sequential State Machines: Step 3

Step 3: Deriving the state table

Current state	Next State/Output	
S0	S1/0	S4/0
S1	S1/0	S2/0
S2	S3/0	S4/0
S3	S6/1	S2/1
S4	S5/0	S4/0
S5	S6/0	S2/0
S6	S6/0	S6/0
	0	1

Input A

There are a total of 7 states
n state variables are needed to
represent these 7 states
$2^n = 7$ and n=(log 7)/(log 2)=3

Check it out! This formula can be generally applied
once you know the number of states:
Enter the number of states in your system: 7
The number of state variables needed is: 3

Fig. 4. Design of the sequential state machines: deriving the state table.

4 Results

The tutor was published as a Windows Projector file (.exe), ensuring that it was a standalone Flash application. It was then posted on the CPEN21000 course on Blackboard at intervals during the semester for the students to access, with the relevant chapters included. In order to evaluate the response of the students to the tutor, a feedback form was distributed. This form contained questions addressing several aspects of the tutor, including if the tutor was used, if the response was positive, and if the visual aspect played a role. The results also provided a general gauge of how helpful the tutor was, and if it contributed to a better understanding of Digital Logic. 100% of those who filled out the feedback form either agreed or strongly agreed that the instructions were clear, and over 88% also found the tutor simple to use and easy to navigate. This indicated that the written directives corresponding to the symbols were successful and that the majority found the interface and buttons easy to use and understand. The responses indicated that the tutor was found to be sufficiently visual in nature, with 94% of the students in agreement that it was visually appealing, and that there was sufficient use of animation. In particular, relating to the level of interaction, the response here was the most favorable, with 65% of answers indicating strong agreement that the application was sufficiently interactive. 88% either agreed or strongly agreed that there as sufficient color and graphics. Additionally, over 82% of the respondents indicated that the animations and dynamic illustrations were helpful, with 15 out of the 17 analyzed, concurring that the animations properly supported the information. Material and Content Generally, 76% agreed or strongly agreed that the content was easy to understand, with 82% of responses indicating that the tutor had, in fact, helped them to understand Digital Electronics. 48% however, were either neutral or disagreed that the tutor had prepared them for the topics to be covered. This was possibly due to the short delay between their starting the topic, and receiving that section of the tutor. It also suggested that rather than the tutor being used as a preparatory tool, it should be used to reinforce the lectures. When asked whether the lecturer could focus on more critical topics, of all the questions this faced the highest level of disapproval, with eight students disagreeing, and five others neutral. This indicated that though the tutor was helpful, the students maintained that the lectures were necessary for their understanding. Several students also indicated that there were insufficient exercises and examples, with a minority of 29% disagreeing with this statement. Unlike tutorials, a full list of such exercises was not given, rather they were incorporated into the tutor. Further work is therefore proposed for the addition of more exercises and examples at the end of each chapter. An overall rating was given between zero (0) and nine (9) for a class of 17 students, the standard deviation and mean were found to be 1.40999 and 7.285 respectively. The overall results suggested that the students generally liked the tutor, and found it useful. This was also supported by the fact that approximately 88% of the students found that it had encouraged them to learn more about Digital Logic. Moreover, 82% of responses indicated that the tutor had, in fact, helped them to understand the subject. The response also indicated that though they found the tutor to have aided in learning, it was not a replacement for the lectures themselves.

5 Conclusion

The SmartStart Tutor for Digital Logic was developed using Adobe Animate, with seven chapters. Feedback forms were then distributed. Of the 17 forms analyzed, 80-90% found the instructions clear, and the interaction, animations, and graphics to be satisfactory. The tutor was also rated to be at an average of 7.285. Generally, the main features of the tutor were favored and found useful, but between one and two students had slight difficulties with the interface and the speed of the animation. The development of Flash-based educational tools can be applied to many other areas. The blend of graphical design and coded interaction is most applicable in explaining objects that are themselves dynamic, such as motors or pulleys, or where certain properties vary with respect to time. It can also be used to explain lab exercises and lab tutorials when the user has to manually perform a sequence of actions. A Flash model can also be provided for real objects, in complex explanations of a device, especially in cases where dynamic movement and 3-D are combined, and it is difficult to visualize. Other topics including Logic Memory Systems, Modern Programmable Logic Devices, Introduction to Computer Arithmetic and Synchronous Design Practices can be further implemented to build on the previous work completed. Additional examples and exercises can also be added, preferably at the end of each chapter, summarizing the information in a precise format for the student. The use of voice and sound can also be added, therefore adding an audio-visual context.

References

[1] M. Iskander, ”technology-based Electromagnetic Education, IEEE Trans. On Microwave Theory and Techniques,” Vol. 50, Issue 3, pp. 1015-1020, 2002.

[2] Mantz Yorke, ”Formative assessment in higher education: Moves towards theory and the enhancement of pedagogic practice”, Higher Education 45: 477501, Springer Verlag 2003.

[3] S. Paul Wright, Sandra P. Horn and William L. Sanders, “Teacher and Classroom Context Effects on Student Achievement: Implications for Teacher Evaluation,” Journal of Personnel Evaluation in Education 11: 57-67, Springer Verlag 1997

[4] M. Duran, S. Gallardo, S. Martinez-Torres, and F. Barrero, A Learning Methodology using Matlab/Simulink for Undergraduate Electrical Engineering Courses Attending to Learner Satisfaction Outcomes, Int. Journal of Technology and Design Education, 17 (1), pp. 55 73, 2007.

[5] M. Iskander, technology-based Electromagnetic Education, IEEE Trans. On Microwave Theory and Techniques, Vol. 50, Issue 3, pp. 1015-1020, 2002.

[6] Lott, D. Schall and K. Peters, Action Script 3.0 Cookbook: Solutions for Flash Platform and Flex Application Developers, OReilly Media; 1 edition (October 1, 2006)

[7] Abdulhadi Shoufan, Zheng Lu, and Sorin A. Huss, A Web-Based Visualization and Animation Platform for Digital Logic Design, IEEE Trans. on Leaning Technologies, Vol. 8, No. 2, April - June 2015, pp. 225 - 239.

[8] Rosilah Hassan, Norul Huda Yusof, and Syahanim Mohd Salleh, Easy electronic software for digital logic design, Procedia - Social and Behavioral Sciences 59 (2012) 498 507, Elseiver.

[9] Z. Stanisavljevic, V. Pavlovic, B. Nikolic, and J. Djordjevic, Sdldssystem for digital logic design and simulation, IEEE Trans. Educ., vol. 56, no. 2, pp. 235245, May 2013.

[10] Lott, D. Schall and K. Peters, Action Script 3.0 Cookbook: Solutions for Flash Platform and Flex Application Developers, OReilly Media; 1 edition (October 1, 2006)

[11] CPEN21000 Logic Design, Blackboard, Lewis University, Romeoville, IL

[12] J. Wakerly, Digital Design: Principles and Practices, 4th ed. New Jersey: Prentice Hall 2005.

[13] EVITA VHDL Tutorial, http://www.aldec.com/products/tutorials/.

[14] Qt Tool, https://www.qt.io/download