Incorporating programming languages in the enhancement of the learning process for sequential logic circuit design

H. Oztekin\textsuperscript{a,*} and A. Gülbag\textsuperscript{b}

\textsuperscript{a}. Department of Computer Engineering, Sakarya University of Applied Sciences, Sakarya, Turkey.  
\textsuperscript{b}. Department of Computer Engineering, Sakarya University, Sakarya, Turkey.

Received 15 October 2020; received in revised form 14 April 2022; accepted 27 June 2022

\begin{center}
\begin{tabular}{|c|c|}
\hline
\textbf{KEYWORDS} & \textbf{Abstract}. Logic circuits can generally be classified into two categories of combinational and sequential logic circuits. In the sequential circuit design, although it is easy to construct a state transition table from state transition diagram, creating an excitation table according to the memory type is a very laborious and time-consuming task. There are several software-based applications and hardware description languages to describe the structure and behavior of electronic circuits. This will take more time and persistence much like learning any other skill. Computer science degrees include courses focused mainly on programming languages. There is a strong case to be made for use of skills acquired in programming courses to shorten the learning curve. Thus, this work proposes a method for the implementation of the circuit incorporating only the state equations of the sequential circuit without dealing with excitation tables. This allows to model the behavior of circuit through buttonbox, checkbox, textbox, etc., which are the basic elements of the graphical user interface-based programming language. A questionnaire was utilized to assess the change in the learning activity of students and perceptions of the proposed method. The results provided that the method was an effective and engaging way of teaching the sequential circuit design. \hline
\end{tabular}
\end{center}

\copyright 2022 Sharif University of Technology. All rights reserved.

1. Introduction

Logic design course is a core requirement in sciences such as computer and electrical and electronic engineering in which students get their first exposure to hardware design. The content of such courses is important in solving the problems they will face in designing in the industry. Logic design is the basic organization of the circuitry of a digital system. The capability of the design will affect the operation of a hardware that can perform a number of operations.

There are two types of circuits in circuit design: combinational and sequential circuits. While combinational circuits are not time-dependent, sequential circuits are time-dependent. Furthermore, while it is sufficient to find the logic equations for the output in the analysis of combinational circuits, sequential logic circuits require a memory and their output depends on the history of the input. This determined the current state of the input as well as the previous state of the output in the result function of the circuit. The design of sequential circuits consists in a five-step process of (i) creating a state transition diagram, (ii) converting the state transition diagram into a state transition
table, (iii) choosing flip-flop types and including their excitation tables in the state transition table, (iv) minimizing the functions for the flip-flop input, and (v) using simplified functions for flip-flops and determining the combinational circuit to represent the output. Due to the presence of memory elements in the completed circuit diagram, having one or more feedback loops is likely to occur. Also, the design process is long and arduous. Therefore, students get discouraged about sequential circuit design in comparison with combinational circuit design. In consequence, students become bored and careless in class and do poorly on tests.

The use of traditional teaching-learning pedagogy in the engineering discipline, which requires analytical skill, technical expertise, and intuitive understanding, limits producing high-quality engineers. Therefore, innovative methods such as cooperative learning, peer learning, technology-supported learning, and participatory learning strategies have been developed [1,2]. Also, the computer technologies have opened new pathways of learning in the educational setting [3-6]. The use of computer programming as an educational tool to enhance learning in other disciplines is becoming increasingly common at all levels of education in many countries [7-13]. Using such facilities, courses should be updated from old paper-based methods to hands-on and computerized versions [14]. These issues motivated our current research: Shortening the process of sequential circuit design using a programming language, we propose a learning methodology aimed at encouraging students to place their skills in electronic circuit design. The basic premise of this methodology is that such design permits the realization of the design using the basic components of the GUI-based programming language without the need for in-depth learning.

The study is structured as follows. In Section 2, existing literature on teaching sequential circuit design is reviewed. Section 3 includes a comprehensive explanation of the conducted method using the visual programming language (C#) for teaching the sequential circuit design. Section 4 presents the application of a questionnaire to evaluate the proposed method and discusses the results obtained. Finally, Section 5 shares the results of the study and concludes with a view on future work.

2. Background and motivation

In the computer science curricula of IEEE Computer Society [15,16], computer architecture and organization course is accepted as one of the key knowledge areas. In order to improve teaching and learning processes in engineering education, different educational approaches should be brought to the agenda in the field of engineering pedagogy [17]. One of the important components of this course is sequential circuit design. Teaching process in the sequential circuit design is not an easy task because of the length and complexity of the design process. For that reason, it includes laboratory activities that allow verifying sequential circuit designs.

Traditional teaching methods are inadequate to meet the needs of engineering students [18-20]. In recent years, efforts have been made to increase the motivation and encouragement of students. Zhang et al. [21] presented an experiment for students to master digital system design methods. Some studies mentioned the contribution of cooperation with simulation and web-based programs in the teaching of the logic design. While some of them [22,23] have given suggestions for the simulator preference to be used in the logic design course, some of them [24] developed web sites containing instructional materials. There are studies that help students build real designs by incorporating programmable logic devices used in contemporary industrial design in the digital logic design course [25]. In a similar study, Wee and Venema [26], by using physical logic blocks, aimed to enable students to physically communicate with circuits. Therefore, the connection between digital logic theory and digital circuit behavior were visualized. In another study, three circuits were designed in addition to the standard circuits to increase the students’ interest in the material [27]. Diirre et al. [28] developed 3D-printable reconfigurable mechanic logic gates that enabled schools or universities to teach the foundations of logic circuits at low costs. In order to better stimulate the students’ learning interest, a teaching reform program has been put forward by the project-driven methods [29-31]. In the studies in the literature, there is a process of either benefiting from a new simulation program or learning a new method. In other words, another learning process is included in addition to the learning process of the logical design course. Our method proposes how to integrate the students’ programming ability into the logical design process. It encourages students of the department of computer engineering to use their programming skills to overcome learning difficulties such as those experienced in the sequential circuit design process and the length of the design period. In this way, the proposed method saves the time that is needed to learn the use of logic simulator programs, which are helpful tools in the implementation of the sequential circuit. In addition, being able to implement the circuit without using the excitation table of the flip-flop reduces the overall design time.

3. Methodology

The methodology used in this work consists in a process that shows how to benefit from the cooperation of pro-
programming languages, enhancing the learning process of engineering students in the design of sequential circuits. The method allows the students to concentrate on the first two steps in the design process. It disables memory excitation tables, which most often cause a decrease in student motivation. This study, therefore, aims to model some of the components that positively affect student motivation in the design of sequential circuit learning experiences for the computer engineering and similar levels of education. While sequential circuits consist of combinational logic circuit and memory elements, combinational circuits have the logic gates and the input/output variable. A block diagram of a sequential circuit is shown in Figure 1.

3.1. Component-based modeling
Equivalents of the following components, which are used in sequential circuit design in GUI-based programming language, are given below.

3.1.1. Component 1: Input/output signal declarations
In GUI-based programming language, a bool type variable is defined for each binary signal in the boolean expressions for circuit. The declarations of the binary signals in the equation of state in the C# programming language are given in Table 1.

3.1.2. Component 2: Connection of input
(external) output ports
Checkboxes commonly used in GUI-based programming can be used to send binary signals to the input ports of the circuit. There are two input ports in the

Table 1. The declarations of input/output signals in C# programming.

<table>
<thead>
<tr>
<th>Boolean expression</th>
<th>Equivalence to C#</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Q_2 = q_1 + x \cdot q_2 + y )</td>
<td>Bool ( q_1 = \text{True/False} )</td>
</tr>
<tr>
<td>Bool ( q_2 = \text{True/False} )</td>
<td>( x = \text{True/False} )</td>
</tr>
<tr>
<td>( y = \text{True/False} )</td>
<td>( y = \text{True/False} )</td>
</tr>
</tbody>
</table>

Note: \( q_1, q_2 \): States of flip-flops;
\( Q_2 \): Next state of \( q_2 \).

Figure 1. Block diagram of the sequential circuit.

if (checkBox1.Checked==True)
  x=true;
else
  x=false;

Figure 2. The use of checkbox object as input port in C# programming.

equation of state given in the above table. Assuming that \( x \) is an external input port and \( y \) is an internal input port, it will be sufficient to drag and drop a checkbox from the toolbox to the form for \( x \) input port. The piece of code required to send a binary signal to the input port \( (x) \) is given in Figure 2. It should be noted that it is necessary to use a separate checkbox for each external input port.

In displaying the output ports, the button or label objects in visual programming language can be used by changing background color or content. In this study, binary information from the output port is represented by changing the background color of the button object. A sample application of how to do this is given in Figure 3.

3.1.3. Component 3: Generation of clock signal
Synchronization is important for a synchronous sequential circuit. This process is achieved by a timing device called a clock generator, which produces regular pulses. A timer object in the visual programming language that repeatedly executes a code block at a specified period of time can be used to provide the function of clock generator, which is needed by the storage elements in a synchronous sequential circuit. Firstly, the timer object is added to the form. Then, the boolean expression that needs to be employed is placed inside the event of the timer tick. These steps are shown in Figure 4.

3.1.4. Component 4: Flip-flops (states)
In our proposed method, the excitation tables are not used to obtain the state equations from the state tables. This means that only the present states and input are included in the state equations (transition equations) of flip-flops. In this case, a flip-flop can be thought of as a
Figure 3. The use of button object as output port in C# programming.

Figure 4. The use of timer object as clock circuit in C# programming.

Table 2. Connecting input and output of a flip-flop in C# programming.

<table>
<thead>
<tr>
<th>Boolean expression</th>
<th>Equivalence to C#</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_1 = Q_1 )</td>
<td>//The first flip-flop</td>
</tr>
<tr>
<td>( q_2 = q_1 + x \cdot q_2 + y )</td>
<td>//The second flip-flop</td>
</tr>
</tbody>
</table>

Note: \( q_1, q_2 \): States of flip-flops;
\( Q_1, Q_2 \): Next state of flip-flops

3.1.5. Component 5: Gates

There are basically three types of logic gates, namely the AND, OR, and NOT. The other gate types are all fairly direct variations of these three basic functions. Since logic gates can be implemented directly in the form of boolean operators in most of the modern programming languages, they are simpler to simulate than other components. Table 3 summarizes the equivalents of the keywords in the visual programming language (C#) needed for the logic gates.

Table 3. The equivalence to C# of logic gates.

<table>
<thead>
<tr>
<th>Description</th>
<th>Equivalent to C# (reserved words)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND gate</td>
<td>&amp;</td>
</tr>
<tr>
<td>OR gate</td>
<td></td>
</tr>
<tr>
<td>NOT gate</td>
<td>!</td>
</tr>
<tr>
<td>EXOR gate</td>
<td>^</td>
</tr>
<tr>
<td>EXNOR gate</td>
<td>!(^)</td>
</tr>
</tbody>
</table>

The flowchart of how to use the above-mentioned components in the sequential circuit modeling is given in Figure 5.

3.2. Case Study-1: 2-bit up/down counter (ordinary method)

Up/down counters are capable of counting in either direction through any given count sequence and they can be reversed at any point within the count sequence by using an additional control input. In this work, we used 2-bit bidirectional counter that could go in either direction, depending on the control input \( (m) \). When \( m = 1 \), it should count up and if \( m = 0 \), it should count down. The binary sequence for this 2-bit counter is 00, 01, 10, and 11. J-K type flip-flops with the output \( q_A \) and \( q_B \) were used in the design of this counter, where \( q_A \) is the LSB (Least Significant Bit) and \( q_B \) is the MSB (Most Significant Bit). This 2-bit counter circuit should be analyzed by the following steps:

Step 1. Creating state table: The state table for this 2-bit binary counter circuit is given in Table 4;

Step 2. Obtaining state diagram: This step is to obtain the state diagram, which is derived from the state table show in Table 4; the diagram is shown in Figure 6.

Table 4. State table of 2-bit binary counter.

<table>
<thead>
<tr>
<th>Present state (QB QA)</th>
<th>Next state (QB QA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m = 0 )</td>
<td>( m = 1 )</td>
</tr>
<tr>
<td>0 0</td>
<td>1 1</td>
</tr>
<tr>
<td>0 1</td>
<td>0 0</td>
</tr>
<tr>
<td>1 1</td>
<td>1 0</td>
</tr>
<tr>
<td>1 0</td>
<td>0 1</td>
</tr>
</tbody>
</table>
**Step 3.** *Determining the flip-flop input equations:* Since there are four states, the number of flip-flops required would be two. Now, we want to implement the counter design using JK flip-flops. In this step the next-state values are determined in the state table using the J-K flip-flop characteristic table. The binary values of the J and K input in this 2-bit counter are shown in Table 5;

**Step 4.** *Transferring to Karnaugh maps and minimal expressions:* The next-state values obtained in Step 3 are placed into Karnaugh maps (Figure 7) to derive a simplified Boolean expression for each flip-flop input.

The first in the Karnaugh maps are grouped with “do not care” and the following expressions for the J and K input of each flip-flop are obtained:

\[ JB = m', \quad KB = m, \quad JA = KA = 1. \]  

---

**Figure 5.** The usage flow chart of the components in modeling the sequential circuit.

**Figure 6.** State diagram of 2-bit binary counter.

**Figure 7.** The Karnaugh map of the flip-flop input.
Step 5. Realization using a logic simulator program:
The final step is to form the sequential circuit by connecting the flip-flops of the combinational logic from the equations. The complete logic design of a 2-bit binary counter in multimedia logic simulator program [32] is shown in Figure 8.

3.3. Case Study-2: 2-bit up/down counter (proposed method)
As observed in Case Study-1, using conventional methods in the sequential circuits and the realization of circuit in the logic simulator program is a very laborious process. The method that we propose only needs the state table. It is sufficient to find equations related to the next states in the state table in order to simulate the same design through visual programming language. The steps required to simulate the circuit are given below, respectively:

Step 1. The first step in the proposed method is the same as that in the traditional method;

Step 2. Producing the next-state equation: These equations are directly derived from the state table and are given below:

\[ Q_B = m \oplus (q_B \oplus q_A), \]
\[ QA = q_A'. \]

Step 3. Building a windows form in C#: C# programmers make extensive use of forms to interact with the user. In this study, the first thing to do to simulate the sequential circuit is to run a windows form application in C# programming language using Visual Studio. The platform will present a blank form that allows us to use the components mentioned in the previous section to simulate the sequential circuit. Figure 9 shows how the default form (Form1) looks like;

Steps 4 and 5. Connection to external input(s)/output(s): When the state table in Step 1 of Case Study-1 is examined, there is one external input port \((m)\). To simulate this, it is sufficient to drag and drop a checkbox to the form. These steps are to
determine the number of outputs to be observed. Since it is desired to display the states of flip-flops in the circuit, it is necessary to use two button objects. As in the adding process of checkbox control, two button objects from the toolbox are added to the form. Figure 10 shows both the checkbox and button controls placed on the form (named Form1);

**Step 6. Declaration the variables:** The previous steps included adding control objects as necessary to the form to simulate the input and output of the circuit. From this step, it is explained how the state equations should be expressed in control objects in order to be simulated. The first thing to do is to define all the variables in the state equations in boolean type. When the state equations obtained in Step 2 are examined, a total of five variables are seen, namely QB, m, qB, qA, and QA. The code snippet for this step is given in Figure 11;

**Steps 7 and 8. Adding the clock generator and coding the equations:** The clock pulses determine when computational activities will occur in the sequential circuit. To simulate this in visual programming language, a timer object with the interval parameter at any frequency is added and the tick event for that object is called. The order of the code snippets to be inserted into the tick event should be expressed as in the flowchart (Figure 6). The entire code inserted into the event is presented in Figure 12. Figure 13 shows the results of compiling and running this program for 2-bit up counter circuit.

4. Evaluation of the proposed method

In this study, a questionnaire was used to provide some justification for the proposed method in the Digital Circuits course. Table 6 summarizes the curriculum required for this course.

Before the survey, the case studies presented in this study were assigned to the students and then, the survey statements were given. The questionnaire study was done with all students who took the course at the Computer Engineering Department in Sakarya University of Applied Sciences. This questionnaire included the statements about the topics: saving time/effort, simplicity, ease of use. Table 7 shows the questions in the student survey conducted after the case studies. Table 8 shows the results of the student survey conducted after the proposed method.

The survey conducted in this study showed that the students had a positive general opinion about the contributions of the proposed method. Almost all of the students declared that sequential circuit design was more difficult than combinatorial circuit design ($M = 4.67$, $MD = 5$, $SD = 0.53$) and they might prefer visual programming languages in design ($M = 4.55$, $MD = 5$, $SD = 0.82$). There was a significantly large positive relationship between $Q1$ and $Q2$ (Spearman’s $r_s = 0.9372$, $p$-value $< .001$). This significant relationship
Table 6. The curriculum for the digital circuits course.

<table>
<thead>
<tr>
<th>Weeks</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subjects</td>
<td>Analog and digital concepts</td>
<td>Number systems</td>
<td>Binary arithmetic</td>
<td>Boolean algebra</td>
<td>Simp. of boolean functions</td>
</tr>
<tr>
<td>Weeks</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>Subjects</td>
<td>Logic gates</td>
<td>Karnaugh map</td>
<td>Midterm week</td>
<td>Combinational logic</td>
<td>Combinatorial components</td>
</tr>
<tr>
<td>Weeks</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>Subjects</td>
<td>Combinatorial components</td>
<td>Sequential logic</td>
<td>Sequential components</td>
<td>Final week</td>
<td></td>
</tr>
</tbody>
</table>

Table 7. The student satisfaction survey.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Strongly disagree</th>
<th>Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>The design of sequential logic circuit is comparatively much more difficult than the combinational logic circuit.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Using the proposed method helped saving time in the design of sequential circuit.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>It is nice to be able to simulate a sequential circuit without using flip-flop excitation tables.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>In the implementation phase of the sequential logic circuit, I prefer to use the accustomed graphical user interface-based programming languages instead of the traditional logic simulators.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The proposed method helped me achieve the course learning outcomes.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Using the accustomed GUI based programming language helped me detect and correct errors in the sequential logic circuit design.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

Note: Strongly disagree: 1 point; Disagree: 2 point; Neutral: 3 point; Agree: 4 point; Strongly agree: 5 point.
was also reflected in the opinions of the students at the end of the survey. The students said that it was more fun to use the visual programming languages which they were familiar with in the sequential circuit design.

Students also thought that not using flip-flop excitation tables in the proposed method was a fun experience ($M = 4.64$, $MD = 5$, $SD = 0.84$) and saved time ($M = 4.35$, $MD = 5$, $SD = 0.91$). When compared with the traditional method applied to the case studies, students specifically agreed that they learned more by using the proposed method ($M = 4.20$, $MD = 4$, $SD = 0.91$).

5. Conclusion

A method was presented for the simulation of sequential logic circuits, especially for computer engineering students. Students could test their designs with familiar visual programming languages without losing time on simulator and hardware details. Since combinational circuits were a part of sequential circuits, this topic was not included. The study was based on C$\#$ language, but it was very easy to adapt to any visual programming language. The sequential circuit design process using a programming language consists of stages such as assignment, logical operations, and display. For students with programming background, the sequential circuit design process would be simplified by using the basic operations mentioned. Feedback from students indicated that using the proposed method was helpful in gaining a better understanding of sequential circuit design. This was seen in the results as saving time and effort, simplicity, and ease of use. The satisfactory outcomes obtained from the student feedback provided that the proposed method was a good way to foster motivation in computer organization and architecture course as well.

References


Biographies

Halit Oztekin is an Associate Professor of Sakarya University of Applied Sciences in Sardovan, Sakarya, Turkey. He obtained his BSc degree in Computer Engineering in 2002 from Sakarya University in Sakarya, Turkey. Oztekin received his PhD in Computer Engineering from the University of Sakarya at Sakarya (UCLA) in 2012. He was appointed to the Department of Computer Engineering at Yozgat Bozok University in 2012 as an Assistant Professor. He was promoted
to the position of Associate Professor in 2019. His research interest lies in the area of computer architecture and organization with strong emphasis on educational tools using FPGAs.

Ali Gülbag received his BSc, MSc, and PhD degrees in Electrical and Electronic Engineering from Sakarya University. He has been working as an Assistant Professor in the Department of Computer Engineering at Sakarya University. His research interests are in artificial intelligence and pattern recognition as well as computer architecture.