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

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

\textsuperscript{a} Department of Computer Engineering, Sakarya University of Applied Sciences, Sakarya, TURKEY.
e-mail: halitoztekin@subu.edu.tr, Tel: +90 264 616 0169
\textsuperscript{b} Department of Computer Engineering, Sakarya University, Sakarya, TURKEY.
e-mail: agulbag@sakarya.edu.tr.

Abstract

Logic circuits can generally be classified into two categories, 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 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’s a strong case to be made for use of skills acquired in programming courses to shorten the learning curve. Thus, this work has proposed a method for the implementation of the circuit 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 basic elements of the graphical user interface based programming language. The questionnaire was applied to assess the change in the student's learning activity and perceptions of the proposed method. The results provide that this method is an effective and engaging way of teaching the sequential circuit design.

Keywords: learning process, logic design, sequential circuit, programming language, computer architecture

1. INTRODUCTION

Logic design course is fundamental core requirements in sciences such as computer, 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 design in the industry. Logic design is the basis 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 circuit. While combinational circuits are not time dependent, sequential circuits are time dependent and are two broad categories of circuits defined in the digital electronics. It is sufficient to find logic equations for the outputs in the analysis of combinational circuits. However, the fact that sequential logic circuits require to have a memory where its outputs depend on the history of
its inputs. This will cause the current states of the inputs as well as the previous states of the output in the result function of the circuit. The design of sequential circuits consists of a five-step process that starts with a state transition diagram and ending with a circuit generation: (i) creating state transition diagram, (ii) converting the state transition diagram into a state transition table, (iii) choosing flip-flop types & including their excitation tables in the state transition table, (iv) minimizing the functions for the flip-flop inputs, 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 and more feedback loops are likely to occur. Also, the design process is long and arduous. Therefore, students get discouraged about in sequential circuit design than combinational circuit design. In consequence, students become bored and careless in class, 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 settings [3-6]. The use of computer programming as an educational tool to enhance learning in other disciplines is becoming increasingly common in 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 motivate our current research: shortening the process of sequential circuit design using a programming language, we propose a learning methodology aimed 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 conducted proposed 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 one of 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 them to verify sequential circuit designs.

Traditional teaching methods are inadequate to meet the needs of engineering students [18-20]. In recent years, efforts to increase students’ motivation and encouragement studies have been made. D. Zhang et al. [21] presented the experiment for students to master digital system design methods. Some studies mentioned on the contribution of cooperation with simulation programs 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 to build real designs by incorporating programmable logic devices used in contemporary industrial designs in the digital logic design course [25]. In a similar study, Wee et al. [26] by using physical logic blocks, aimed to enable students to physically communicate with circuits. So the connection between digital logic theory and digital circuit behavior have been visualized. In the other study, the three circuits designed in addition to the standard circuits to increase students’ interest in the material [27]. J. Dürre et al. [28] developed 3D-printable reconfigurable mechanic logic gates enable schools or universities to teach the foundations of logic circuits at low costs. For better stimulate the students’ learning interest, the teaching reform program is 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, it is included in another learning process in addition to the learning process of the logical design course. Our method proposes how to integrate the student’s programming ability into the logical design process. It is to encourage students of the department of computer engineering to use their programming skills to overcome difficulties such as learning difficulties 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 of a process that shows how to benefit from the cooperation of programming languages enhancing the learning process of engineering
students in the design of sequential circuits. The method allows the student to concentrate on
the first two steps in the design process. It disables memory excitation tables which most
often causes 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 to contribute to sequential circuit learning experiences for the computer
engineering and similar level of education.
While sequential circuits consist of combinational logic circuit as well as memory elements,
combinational circuits have the logic gates and inputs/outputs variable. A block diagram of a
sequential circuit is shown in Figure 1.

3.1. Component-based Modeling
The equivalents of these components which are used 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 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 on the form for \( x_{\text{input}} \) 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 check box for each external input port.

In displaying of 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. An
example 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 executes a code block at a specified periodic of time repeatedly can be used to meet up with the function of the clock generator that are 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 inputs are included in the state equations (transition equation) of flip-flops. In this case, a flip-flop can be thought as a block which its input is next state and its output is present state. To model a flip-flop in a visual programming language, all needed is to assign the next state variable to the present state variable. It is customary to use the symbols with upper-case and lower-case letters, respectively, to implement the next state and the present state. Considering this situation, an example is given in Table 2.

3.1.5. Component 5 – Gates
There are basically 3 types of logic gates and these gates are the AND, OR, NOT. The other gate types are all fairly direct variations on three basic functions. Since logic gates can be implemented directly in the form of boolean operators in most 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.

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 their count sequence by using an additional control input. In this work, we used 2-bit bidirectional counter that can go in either direction, depending on the control input(m). When m=1 it should count up, if m=0 it should count down. The binary sequence for this 2-bit counter are 00, 01, 10 and 11. J-K type flip-flops with outputs qA, and qB are used in the design of this counter where qA is the LSB (Least significant bit) and qB is the MSB (Most significant bit).
**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
The next step is to obtain the state diagram, which is derived from the state table in Table 4 and is shown in Figure 6.

**Step-3:** Determining the flip-flop input equations
Since there are 4 states, the number of flip-flops required would be two. Now we want to implement the counter design using JK flip-flops. Next step is to determine the next-state values in the state table using the J-K flip-flop characteristic table. The binary values of the J and K inputs 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 to Karnaugh maps (Figure 7) to derive a simplified Boolean expression for each flip-flop input. The 1s in the Karnaugh maps are grouped with "don't cares" and the following expressions for the J and K inputs of each flip-flop are obtained:

\[ J_B = m \quad K_B = m \quad J_A = K_A = 1 \quad (1) \]

**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 you see the above case study-1, producing these 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 need 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 the traditional method.
**Step-2: Producing the next-state equation**

These equations are directly derived from the state table and are given below:

\[ QB = m \otimes (qB \oplus qA) \quad (2) \]

\[ QA = qA' \quad (3) \]

**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.

**Step-4 and 5: Connection to External Input(s)/Output(s)**

When the state table in Step-1 is examined, there is one external input port (m). To simulate this, it is sufficient to drag and drop a checkbox on the form. The next step is 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 are placed on the form (named Form1).

**Step-6: Declaration the variables**

The previous steps include adding control objects as necessary on the form to simulate the inputs and outputs 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 for this is to define all the variables in the state equations as boolean type. When the state equations obtained in Step-2 are examined, a total of 5 variables are seen: these are \( QB \), m, qB, qA and QA, respectively. The code snippet for this step is given in Figure 11.

**Step-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 as expressed 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 PROPOSED METHOD

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

Before the survey, the case studies presented in this study were implemented to the students and then the survey statements were given. The questionnaire study was applied to all students who took the course at the computer engineering department in Sakarya University of Applied Sciences. This questionnaire includes the statements about the topics: save time/effort, simplicity, ease of use. Table 7 shows the questions of student survey conducted after the case studies. Table 8 shows the results of student survey conducted after proposed method.

The survey conducted in this study shows that the students have a positive general opinion about the contributions of the proposed method. Almost all of the students declared that sequential circuit design is more difficult than combinatorial circuit design (\(M=4.67, MD=5, SD=0.53\)) and they may prefer visual programming languages in design (\(M=4.55, MD=5, SD=0.82\)). There is a significant large positive relationship between Q1 and Q2 (Spearman's \(r_s=0.9372, p\text{-value}=.001\)). This significant relationship was also reflected in the opinions of the students at the end of the survey. The students said that it is more fun to use the visual programming languages which they are familiar with in sequential circuit design. Students also thought that not using flip-flop excitation tables in the proposed method is a fun experience (\(M=4.64, MD=5, SD=0.84\)) and saves time (\(M=4.35, MD=5, SD=0.91\)). When compared to the traditional method applied in 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 is presented for simulation of sequential logic circuits, especially for computer engineering students. Students can test their designs with familiar visual programming languages without losing time on simulator and hardware details. Since combinational circuits are a part of sequential circuits, this topic is not included. The study is based on C# language, but it is 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 will be simplified by using the basic operations mentioned. Student feedbacks indicates that using the proposed method was helpful in gaining a better understanding of sequential circuit design. This is seen in the results as saving time and effort, simplicity and ease of use. The satisfactory outcomes obtained from student feedbacks provides that the
proposed method is a good way to foster motivation in computer organization and architecture course as well.

6. REFERENCES

[1] Aqqal, A., Elhannani, A., Haidine, A., et al. “Improving the Teaching of ICT Engineering using Flipped Learning: a personalized model and a case study”. Production, 27(spe), pp. e20162274 (2017).
[2] ArunKumar, S., Sasikala, S., Kavitha, K., “Towards Enhancing Engineering Education through Innovative Practices in Teaching Learning”, Int. J. of Eng. and Adv. Tech., 8(2S), pp. 153 - 159 (2018).
[3] Keengwe, J., Onchwari, G., Wachira, P. “The use of computer tools to support meaningful learning”, AACE Review, 16(1), pp.77 - 92 (2008).
[4] Pérez-delHoyo, R., Mora, H., Martí-Ciriquián, P., et al. “Introducing innovative technologies in higher education: An experience in using geographic information systems for the teaching learning process”, Computer Appl. Eng. Educ., 28(5), pp.1110 - 1127 (2020).
[5] Mora, H., Pont, MTS., Guilló, AF, et al. “A collaborative working model for enhancing the learning process of science & engineering students”. Computers in Human Behavior, 103, pp.140 –150 (2020).
[6] Barak M., “Teaching Electronics: From Building Circuits to Systems Thinking and Programming”, In Handbook of Technology Education, M.J. De Vries, Ed., pp. 337 – 360, Springer, Cham (2018).
[7] Moreno-León J., Roble G. “Computer programming as an educational tool in the English classroom a preliminary study”, 2015 IEEE Global Engineering Education Conference (EDUCON), Tallinn, Estonia, pp. 961-966 (2015).
[8] Rahnavard, M., Alavi, S.M., Khorasani, S., et al. “Educational Robot for Principles of Electrical Engineering”, Scientia Iranica, 25(3), pp.1582-1592 (2017).
[9] Oren, M., Pedersen, S., Butler-Purry, K. L., “Teaching Digital Circuit Design With a 3-D Video Game: The Impact of Using In-Game Tools on Students' Performance”, IEEE Transactions on Education, 64(1), pp. 24-31 (2021).
[10] Li, S., Zheng, N., Gao, Y., et al. “Make experimental device to be a toy - the experimental teaching in Digital Logic Circuit course”, 2021 IEEE Frontiers in Education Conference (FIE), Lincoln, NE, USA, pp.1-5 (2021).
[11] Wilson, A., Hainey, T., Connolly, T. “Evaluation of Computer Games Developed by Primary School Children to Gauge Understanding of Programming Concepts”, Proceedings of the 6th European Conference on Games Based Learning, Cork, Ireland, pp. 549 - 558 (2012).
[12] Orni, MS., Michal, A., Mordechai, BA. “Learning computer science concepts with scratch”, Computer Sci. Education, 23(3), pp. 239-264 (2013).

[13] Malan, D.J., Leitner, H.H. “Scratch for budding computer scientists”, SIGCSE ’07: Proceedings of the 38th SIGCSE technical symposium on Computer science education, Covington, KY, USA, pp. 223-227 (2007).

[14] Lopez-Rosenfeld, M. “Tell Me and I Forget, Teach Me and I May Remember, Involve Me and I Learn: Changing the Approach of Teaching Computer Organization”, IEEE /ACM 1st Int. Workshop on Soft. Eng. Curricula for Millennials (SECM), Buenos Aires, Argentina, pp.68-71 (2017).

[15] ACM/IEEE-CS Joint Curriculum Task Force, “Computing Curricula 2001”, Report (OCoLC)654266454, Los Alamitos, Calif., submitted to Association for Computing Machinery(ACM) (2001)

[16] ACM/IEEE-CS Joint Curriculum Task Force, “Computer Engineering Curricula 2016”, submitted to Association for Computing Machinery(ACM) (2016)

[17] Kersten, S. “Approaches of Engineering Pedagogy to Improve the Quality of Teaching in Engineering Education”, In Vocational Teacher Education in Central Asia: Developing Skills and Facilitating Success, J. Drummer, Ed., pp. 129-139, Springer, Cham (2018).

[18] Uribe, R.B., Haken L., Loui M.C., “A design laboratory in electrical and computer engineering freshman”, IEEE Transactions on Education, 37(2), pp.194–202 (1994).

[19] Finelli, Cynthia J., Froyd, Jeffrey E., “Improving Student Learning in Undergraduate Engineering Education by Improving Teaching and Assessment”, Advances in eng. edu., Spring, pp. 1-30 (2019).

[20] Alcaraz, R., Martínez-Rodrigo, A., Zangróniz, R., et al. “Blending Inverted Lectures and Laboratory Experiments to Improve Learning in an Introductory Course in Digital Systems”, IEEE Transactions on Education, 63(3), pp. 144-154 (2020).

[21] Zhang, D., Xu, Z., Wang, L. et al. “Digital Logic Experiment Design Based on FPGA Development Board and OV2640 Camera”, 14th International Conference on Computer Science & Education (ICCSE), Toronto, ON, Canada, pp. 671-675 (2019).

[22] Prasad, P.W.C., Alsadoon, A., Beg, A., et al. “Incorporating simulation tools in the teaching of digital logic design”, IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014), Penang, Malaysia, pp.18-22 (2014).

[23] Kurniawan, W., Ichsan, M.H.H., “Teaching and learning support for computer architecture and organization courses design on computer engineering and computer science for undergraduate: A review”, 4th International Conference on Electrical Eng., Computer Sci. and Info. (EECSI), Yogyakarta, Indonesia, pp. 1-6 (2017).

[24] Noga, K., Radwanski, M., “Our experiences in teaching of digital logic”, In: Innovations in E-learning, Instruction Technology, Assessment, and Engineering Education, M. Iskander,
Ed., pp. 237-242, Springer, Dordrecht (2007).

[25] Manikas, T.W., Kane, G.R, Kohlbeck, J.G., “A Digital Logic Design Laboratory for Electrical Engineering and Computer Science Undergraduates”, Proceedings of the 36th ASEE Midwest Section Conference, pp. 1-4, (2002).

[26] Tan, W.L., Venema, S. “Using Physical Logic Gates to Teach Digital Logic to Novice Computing Students”, Proceedings of the IADIS International Conference Educational Technologies, Hong Kong, pp.11-18 (2019).

[27] Hoffbeck, Joseph P. “Using Practical Examples in Teaching Digital Logic Design”, ASEE Annual Conference & Exposition, Indianapolis, Indiana, pp. 24.1340.1 - 24.1340.14 (2014).

[28] Dürre, J., Payá-Vayá, G., Blume, H., “Teaching Digital Logic Circuit Design via Experiment-Based Learning - Print your own Logic Circuit ”, Proceedings of The 20th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI), USA, pp. 242-247 (2016).

[29] Wang, Z., Hu, S. “Teaching Reform on Digital Circuit and Logic Design Course”, American Journal of Agriculture and Crops", 2(2), pp.148-152 (2017).

[30] Yuan, H., Sun, W., Liu, L. “Research on Teaching Mode of Digital Logic Circuits Based on "MOOC + SPOCs + Flip Classroom", ICDEL ’18: Proceedings of the 2018 International Conference on Distance Education and Learning, Beijing, China, pp.117-121 (2018).

[31] Li, J., Li, C., Son, J.S., et al. “Effectiveness of Using My FPGA Platform for Teaching Digital Logic", ASEE Virtual Annual Conference Experience, Virtual On line, pp. 1-14 (2020).

[32] “Multimedia Logic (Version 1.6) Emulator Program ", https://multimedia-logic.software.informer.com/1.6/.
List of Tables:

Table 1. The declarations of input/output signals in C# programming
Table 2. Connecting input and output of a flip-flop in C# programming
Table 3. The Equivalence to C# of logic gates
Table 4. State table of 2-bit binary counter
Table 5. The flip-flop inputs for next states according to J-K characteristic table
Table 6. The Curriculum for Digital Circuits Course
Table 7. The Student Satisfaction Survey
Table 8. Results of the student survey (N=34).

List of Figures:

Figure 1. Block diagram of sequential circuit
Figure 2. The use of checkbox object as input port in C# programming
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
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 flip-flop inputs
Figure 8. The circuit diagram of a 2-bit binary counter with up/down in Multimedia Logic Simulator
Figure 9. Creating the default form in C# programming language
Figure 10. Adding the checkbox and buttons on the form in C# programming language
Figure 11. The declaration of variables in the state equations
Figure 12. The order of the code to be insert to tick event
Figure 13. The screenshots of 2-bit binary down counter(m=0) after running the code
Figure 1. Block diagram of sequential circuit.

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

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

```csharp
if (y == true)
    button1.BackColor = Color.Yellow;
else
    button1.BackColor = Color.Gray;
```

Figure 3. The use of button object as output port in C# programming.
private void timer1_tick (object sender, EventArgs e)  
    {
    //to be executed the Boolean expression
    

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

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 flip-flop inputs.
Figure 8. The circuit diagram of a 2-bit binary counter with up/down in Multimedia Logic Simulator

Figure 9. Creating the default form in C# programming language.
public partial class Form1 : Form
{
    bool QB = false;
    bool m = false;
    bool qB = false;
    bool qA = false;
    bool QA = false;
}

Next-State Equations
(1): QB = m ⊕ (qB ⊕ qA)
(2): QA = qA

Figure 10. Adding the checkbox and buttons on the form in C# programming language.

Figure 11. The declaration of variables in the state equations.

Figure 12. The order of the code to be insert to tick event
Figure 13. The screenshots of 2-bit binary down counter ($m=0$) after running the code.
Table 1. The declarations of input/output signals in C# programming.

| Boolean Expression | Equivalence to C# |
|--------------------|-------------------|
| Q2=q1+x•q2+y       | bool Q2=True/False; |
|                    | bool q1= True/False; |
|                    | bool x= True/False; |
|                    | bool q2= True/False; |
|                    | bool y= True/False; |

q1, q2: present states of flip-flops --- Q2: next state of q2.

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

| Boolean Expression | Equivalence to C# |
|--------------------|-------------------|
| Q2=q1+x•q2+y       | q1=Q1 //The first flip-flop |
|                    | q2=Q2 //The second flip-flop |

q1, q2: present states of flip-flops --- Q1, Q2: next state of flip-flops

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

| Description       | Equivalence to C# (reserved words) |
|-------------------|------------------------------------|
| AND gate          | &                                  |
| OR gate           | |                                  |
| NOT gate          | !                                  |
| EXOR gate         | ^                                  |
| EXNOR gate        | !(^)                               |
Table 4. State table of 2-bit binary counter.

| Present State | Next State | m=0 | m=1 |
|---------------|------------|-----|-----|
| qb qa         | QB QA      | QB QA | QB QA |
| 0 0           | 1 1        | 0 1  | 0 1  |
| 0 1           | 0 0        | 1 0  | 0 0  |
| 1 1           | 1 0        | 0 0  | 0 0  |
| 1 0           | 0 1        | 1 1  | 1 1  |

Table 5. The flip-flop inputs for next states according to J-K characteristic table.

| Second FF(MSB) | First FF(LSB) | J-K characteristic table |
|----------------|---------------|--------------------------|
| JB KB JA KA    |               | J K Q                    |
| m X 1 X       |               | 0 0 Q                    |
| m' X X 1      |               | 0 1 0                    |
| X m 1 X       |               | 1 0 1                    |
| X m X 1       |               | 1 1 Q'                   |
The design of sequential logic circuit is comparatively much difficult than the combinational logic circuit.

Using the proposed method helped saving of time in the design of sequential circuit.

It’s nice to be able to simulate a sequential circuit without using flip-flop excitation tables.

In the implementation phase of the sequential logic circuit, I prefer to use the accustomed graphical user interface-based programming languages instead of traditional logic simulators.

The proposed method helped me achieve the course learning outcomes.

Using the accustomed GUI based programming language helped to detect and correct errors in sequential logic circuit design.

### Table 6. The Curriculum for Digital Circuits Course.

| Subjects                  | Analog and Digital concepts | Digital Arithmetic | Number systems | Boolean Algebra |simp. of Boolean Functions | Logic Gates | Karnaugh Map | Midterm Week | Combinational Logic | Combinatorial Components | Sequential Logic | Sequential Components | Final Week |
|---------------------------|-----------------------------|--------------------|----------------|----------------|---------------------------|-------------|--------------|--------------|----------------------|------------------------|-------------------|--------------------|------------|
| Weeks                     | 1                           | 2                  | 3              | 4              | 5                        | 6           | 7            | 8            | 9                    | 10                     | 11               | 12                 | 13         | 14                  | 15                     |

### Table 7. The Student Satisfaction Survey.

| Statements                                                                 | Strongly Disagree | Disagree | Neutral | Agree | Strongly Agree |
|-----------------------------------------------------------------------------|-------------------|----------|---------|-------|----------------|
| The design of sequential logic circuit is comparatively much difficult than the combinational logic circuit. | 0                 | 0        | 0       | 0     | 0              |
| Using the proposed method helped saving of time in the design of sequential circuit.                              | 0                 | 0        | 0       | 0     | 0              |
| It’s nice to be able to simulate a sequential circuit without using flip-flop excitation tables.                    | 0                 | 0        | 0       | 0     | 0              |
| In the implementation phase of the sequential logic circuit, I prefer to use the accustomed graphical user interface-based programming languages instead of traditional logic simulators. | 0                 | 0        | 0       | 0     | 0              |
| The proposed method helped me achieve the course learning outcomes.                                                              | 0                 | 0        | 0       | 0     | 0              |
| Using the accustomed GUI based programming language helped to detect and correct errors in sequential logic circuit design. | 0                 | 0        | 0       | 0     | 0              |

Strongly Disagree: 1 point  Disagree: 2 point  Neutral: 3 point  Agree: 4 point  Strongly Agree: 5 point
Biographies

Dr. Halit Oztekin is an Associate Professor of Sakarya University of Applied Sciences in Serdivan, Sakarya, Turkey. He obtained his undergraduate degree in computer engineering (2002) from Sakarya University in Sakarya, Turkey. Oztekin received his Ph.D. 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 Associate Professor in 2019. His research interests lie 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 at Sakarya University. He has been working as an assistant professor in Department of Computer Engineering at Sakarya University. His research interests are in artificial intelligence and pattern recognition, computer architecture.

Table 8. Results of the student survey (N=34).

| Statements | Strongly Disagree | Disagree | Neutral | Agree | Strongly Agree | M      | MD | SD |
|-----------|------------------|----------|---------|-------|----------------|--------|----|----|
| Q1        | 0                | 0        | 1       | 9     | 24             | 4.67   | 5  | 0.53 |
| Q2        | 1                | 0        | 4       | 10    | 19             | 4.35   | 5  | 0.91 |
| Q3        | 0                | 2        | 2       | 2     | 28             | 4.64   | 5  | 0.84 |
| Q4        | 1                | 0        | 1       | 9     | 23             | 4.55   | 5  | 0.82 |
| Q5        | 0                | 2        | 5       | 11    | 16             | 4.20   | 4  | 0.91 |
| Q6        | 1                | 2        | 6       | 7     | 18             | 4.14   | 5  | 1.10 |

M: Mean   MD: Median   SD: Standard Deviation