Scientia Iranica D (2022) 29(6), 3219–3229

Sharif University of Technology
Scientia Iranica
Transactions D: Computer Science & Engineering and Electrical Engineering
http://scientiairanica.sharif.edu

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

H. Oztekin[a,*] and A. Gülbag[b]

a. *Department of Computer Engineering, Sakarya University of Applied Sciences, Sakarya, Turkey.*
b. *Department of Computer Engineering, Sakarya University, Sakarya, Turkey.*

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

KEYWORDS
Learning process;
Logic design;
Sequential circuit;
Programming language;
Computer architecture.

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.

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

*. *Corresponding author. Tel.: +90 264 616 0169*
E-mail addresses: halitoztekin@subu.edu.tr (H. Oztekin); agulbag@sakarya.edu.tr (A. Gülbag)

doi: 10.24200/sci.2022.56940.4990

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 other study, three circuits were designed in addition to the standard circuits to increase the students' interest in the material [27]. Dürre 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-

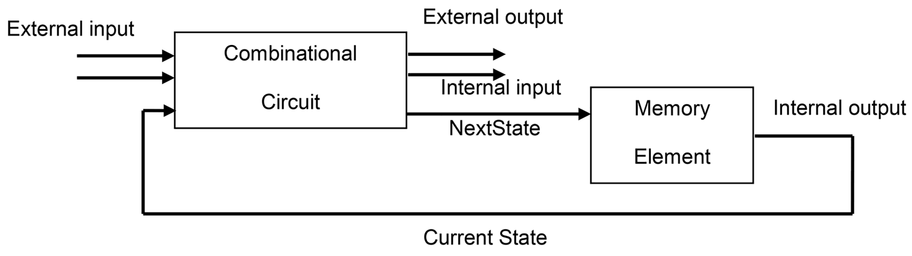


Figure 1. Block diagram of the sequential circuit.

gramming 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

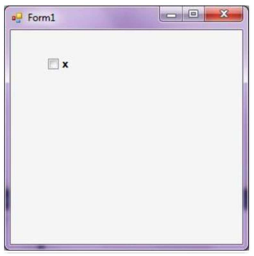
Checkboxes commonly used in GUI-based programming can be used to send binary signals to the input ports of the circuit. There are two input ports in the

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

Boolean expression	Equivalence to C#	
$Q2 = q1 + x \cdot q2 + y$	Bool	$Q2$ = True/False;
	Bool	$q1$ = True/False
	Bool	x = True/False
	Bool	$q2$ = True/False
	Bool	y = True/False

Note: $q1$, $q2$: States of flip-flops;

$Q2$: Next state of $q2$.



```
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 check box for each external input port.

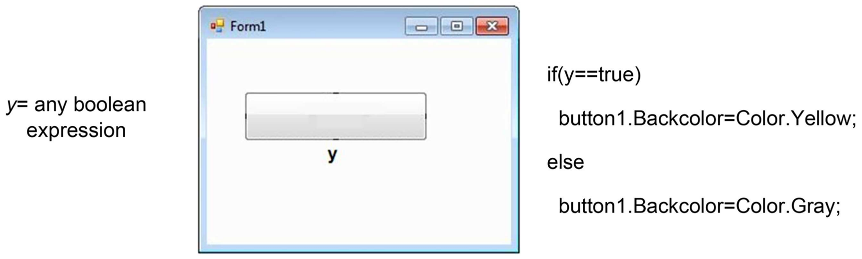
In displaying of the output ports, the button or label objects in visual programming language can be used by changing background color or content. In this study, binary information from the output port is represented by changing the background color of the button object. A sample application of how to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

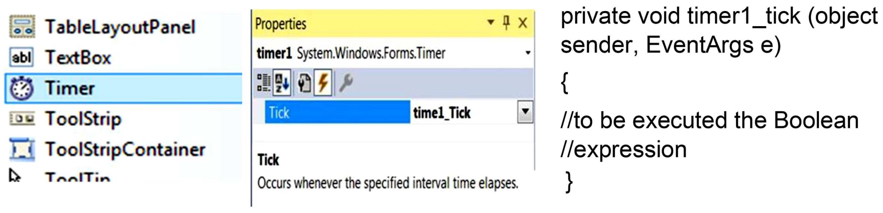
3.1.4. Component 4: Flip-flops (states)

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



```
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.

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

Boolean expression	**Equivalence to C#**
$Q2 = q1 + x \cdot q2 + y$	$q1 = Q1$ //The first flip-flop $q2 = Q2$ //The second flip-flop

Note: $q1$, $q2$: States of flip-flops;
$Q1$, $Q2$: Next state of flip-flops

block the input to which is the next state and the output of which is the 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 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.

Description	**Equivalence to C# (reserved words)**
AND gate	&
OR gate	|
NOT gate	!
EXOR gate	^
EXNOR gate	!(^)

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 qA and qB were used in the design of this counter, where qA is the LSB (Least Significant Bit) and qB 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.

Present state		**Next state**			
		$m = 0$		$m = 1$	
qB	qA	QB	QA	QB	QA
0	**0**	1	1	0	1
0	**1**	0	0	1	0
1	**1**	1	0	0	0
1	**0**	0	1	1	1

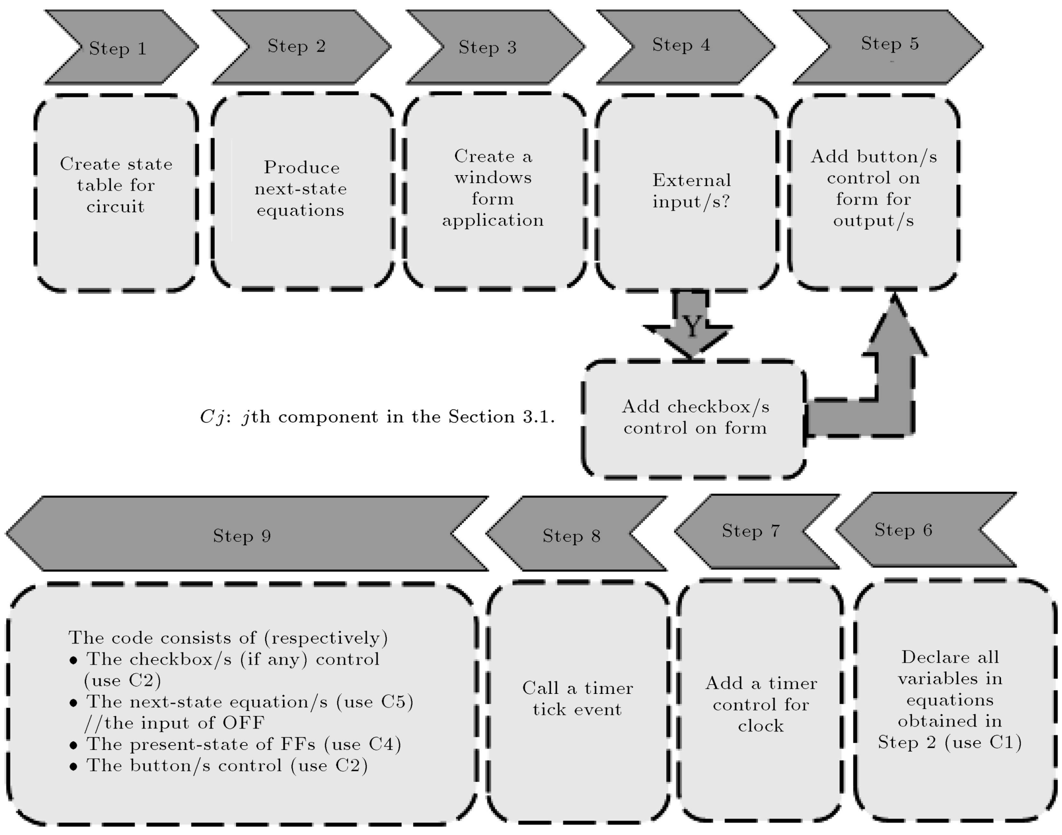

Step 1
Create state table for circuit
Step 2
Produce next-state equations
Step 3
Create a windows form application
Step 4
External input/s?
Y
Add checkbox/s control on form
Step 5
Add button/s control on form for output/s
Cj: jth component in the Section 3.1.
Step 6
Declare all variables in equations obtained in Step 2 (use C1)
Step 7
Add a timer control for clock
Step 8
Call a timer tick event
Step 9
The code consists of (respectively)
• The checkbox/s (if any) control (use C2)
• The next-state equation/s (use C5) //the input of OFF
• The present-state of FFs (use C4)
• The button/s control (use C2)

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

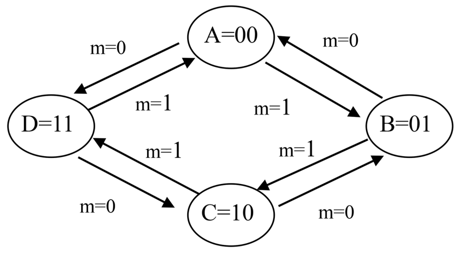


Figure 6. State diagram of 2-bit binary counter.

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:

$$JB = m', \qquad KB = m, \qquad JA = KA = 1. \qquad (1)$$

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

The second FF (MSB)		**The 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'

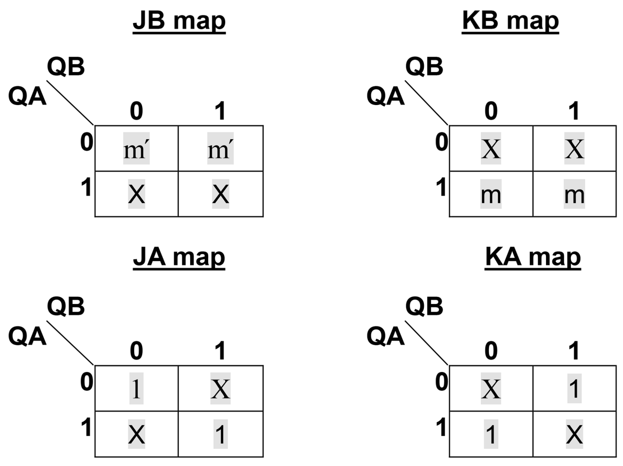


Figure 7. The Karnaugh map of the flip-flop input.

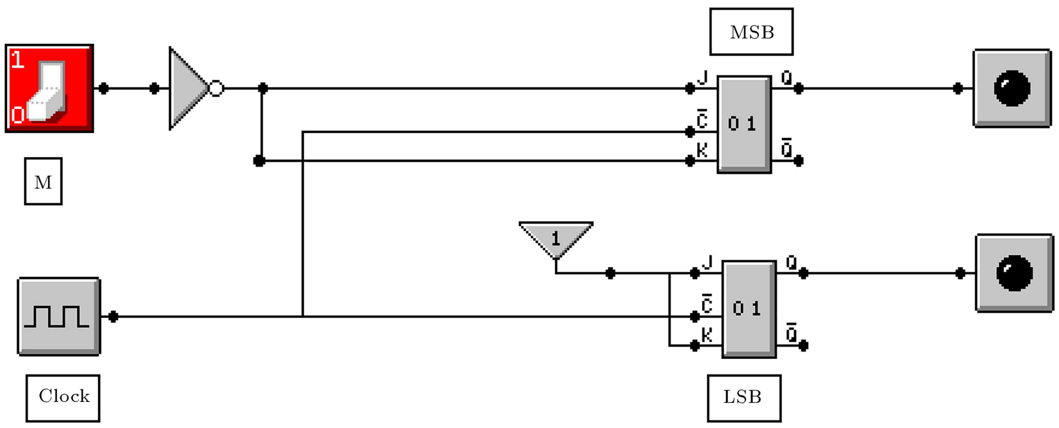


Figure 8. The circuit diagram of a 2-bit binary counter with up/down in multimedia logic simulator.

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:

$$QB = m \otimes (qB \otimes qA), \tag{2}$$

$$QA = qA'. \tag{3}$$

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;

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

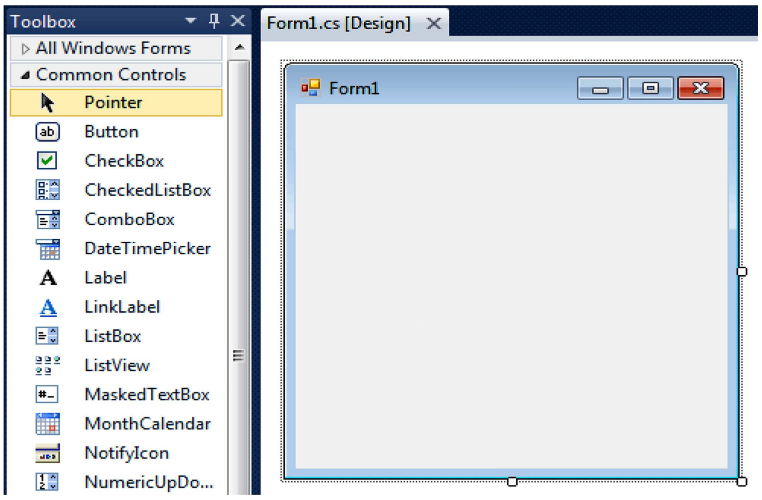


Figure 9. Creating the default form in C# programming language.

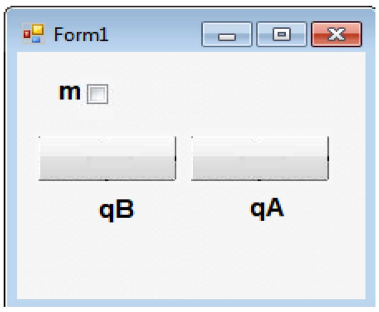
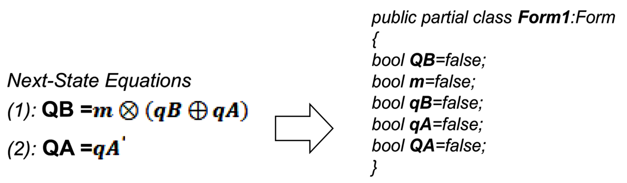


Figure 10. Adding the checkbox and buttons to the form in C# programming language.



Next-State Equations

(1): QB $= m \otimes (qB \oplus qA)$

(2): QA $= qA'$

```
public partial class Form1:Form
{
bool QB=false;
bool m=false;
bool qB=false;
bool qA=false;
bool QA=false;
}
```

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

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;

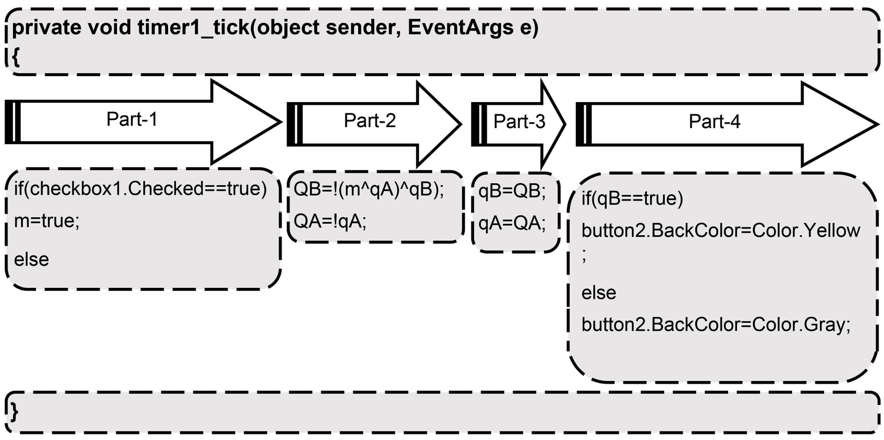
Steps 7 and 8. *Adding the clock generator and coding the equations:* The clock pulses determine when computational activities will occur in the sequential circuit. To simulate this in visual programming language, a timer object with the interval parameter at any frequency is added and the tick event 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



```
private void timer1_tick(object sender, EventArgs e)
{
```

```
if(checkbox1.Checked==true)
m=true;
else
```

```
QB=!(m^qA)^qB);
QA=!qA;
```

```
qB=QB;
qA=QA;
```

```
if(qB==true)
button2.BackColor=Color.Yellow
;
else
button2.BackColor=Color.Gray;
```

```
}
```

Figure 12. The order of the code to be insert to the tick event.

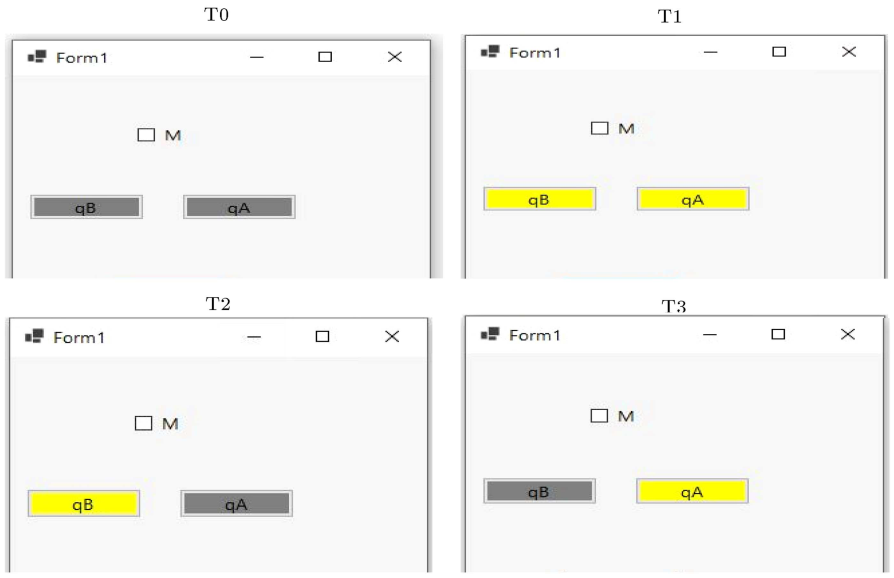


Figure 13. The screenshots of 2-bit binary down counter ($m = 0$) after running the code.

Table 6. The curriculum for the digital circuits course.

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

Table 7. The student satisfaction survey.

Statements	**Strongly disagree**	**Disagree**	**Neutral**	**Agree**	**Strongly agree**
The design of sequential logic circuit is comparatively much more difficult than the combinational logic circuit.	○	○	○	○	○
Using the proposed method helped saving time in the design of sequential circuit.	○	○	○	○	○
It is 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 the traditional logic simulators.	○	○	○	○	○
The proposed method helped me achieve the course learning outcomes.	○	○	○	○	○
Using the accustomed GUI based programming language helped me detect and correct errors in the sequential logic circuit design.	○	○	○	○	○

Note: Strongly disagree: 1 point; Disagree: 2 point; Neutral: 3 point; Agree: 4 point; Strongly agree: 5 point.

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

Note: M: Mean; MD: Median; SD: Standard Deviation.

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

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), p. e20162274 (2017).
2. ArunKumar, S., Sasikala, S., and 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., and 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, M.T.S., Guilló, A.F., 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. and 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., and 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., and 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, M.S., Michal, A., and Mordechai, B.A. "Learning computer science concepts with scratch", *Computer Sci. Education*, **23**(3), pp. 239–264 (2013).

13. Malan, D.J. and 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, C.J. and Froyd, J.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. and 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. and 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., and 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. and 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, J.P. "Using practical examples in teaching digital logic design", *ASEE Annual Conference & Expositio*n, Indianapolis, Indiana, pp. 24.1340.1–24.1340.14 (2014).

28. Dürre, J., Payá-Vayá, G., and Blume, H. "Teaching 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. and 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., and 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 Online, pp. 1–14 (2020).

32. Multimedia Logic (Version 1.6) Emulator Program, https://multimedia-logic.software.informer.com/1.6/.

Biographies

Halit Oztekin is an Associate Professor of Sakarya University of Applied Sciences in Serdivan, 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.