One approach to study the topic "Logic and combinational circuits"

MICHAEL DOLINSKY
Faculty of Mathematics and Programming Technologies
Francisk Skorina Gomel State University
Sovetskaya Str.104, 246019, Gomel
BELARUS

Abstract. This article describes the technology of teaching the theme “Logic and combinational circuits” of basic digital electronics course to first/second-year students based on the DL.GSU.BY website. The main advantages of the technology include training adapted to the student, many years of experience in practical application and effectiveness. The following issues are consistently considered in the article: the theoretical foundations of the topic; library of standard components; tasks for simulation of standard components; assignments for the simulation of circuits made up of standard components; tasks for designing schemes; tasks for designing schemes according to programs; test questions and flash tasks; technology of using practical tasks.

Keywords: basic of digital electronics, website DL.GSU.BY, Gomel State University

Received: December 23, 2021. Revised: December 16, 2022. Accepted: January 18, 2023. Published: February 15, 2023.

1 Introduction

Blended learning has been actively developed before COVID, but with the advent of this disease it has become critical. Works in [1], [2] substantiate the need for a transition to blended learning, which combines the possibilities of traditional classroom learning and new information technologies. The works in [3-8] describe examples of such blended learning on various topics: mathematics [3], English language [4], earth sciences [5], basic medical knowledge [6], anatomy [7], and chemistry [8].

The author has been using blended learning for disciplines related to teaching programming and digital electronics for many years at the Faculty of Mathematics and Programming Technologies (specialties: "Information Technology Software", "Informatics and Programming Technologies", "Applied Informatics") of Gomel State University named by F.Skorina, [9], [10], [11].

To increase the effectiveness of training, the system for designing, simulation and debugging of digital electronics circuits HLCCAD [12] and the instrumental system for distance learning DL.GSU.BY [13] both developed under the guidance of the author are used. This work presents the author's approach to the introduction of blended learning in the basics of digital electronics.

2 Problem Formulation

In recent decades, the conditions for learning have been rapidly changing: computer tools and Internet technologies are being improved, the level of training and motivation of students is decreasing on average, and the requirements for knowledge and skills of university graduates are growing at the same time. This leads to the need to change the learning process, so that, on the one hand, theoretical knowledge is given in much more detail and in a much simpler language than was previously accepted, on the other hand, not only and not so much seminars with students were used to consolidate the knowledge gained, but rather solving practical problems. At the same time, a significant part of such tasks can be performed in systems that simulate projects developed by students. To involve poorly prepared students in the educational process, it is useful to use tasks of a control and training nature of various forms, when, in the process of performing a system of automatically checked tasks, the student implicitly receives training information. The technical basis of this approach to learning is the software packages developed at F. Skorina GSU under the guidance of the author.

This work is devoted to the description of such a modified methodology for teaching the topic "Logic and combinational circuits", taking into account the trends described above. Before this topic, students have already studied the topics "Introduction to the subject" and "Synthesis of combinational circuits from truth tables".

The topic "Introduction to the subject" is specifically intended for mastering the HLCCAD system, with which you can edit, simulate and debug the functional diagrams of digital devices, as well as to familiarize yourself with the basic logical operations NOT, AND, OR, XOR and the corresponding basic logic elements.

The topic "Synthesis of combinational circuits according to truth tables", on the one hand, consolidates knowledge and skills in applying the basic logical elements NOT, AND, OR, XOR to solve problems for the design of digital devices, on the other hand, develops the skills of analysis (and mental simulation) of circuits, composed of these basic logical elements, and on the third hand introduces the method of minimizing logical functions by means of Carnot maps.

3 Problem Solution

3.1 Theoretical Background

As part of the topic "Combination Circuits", according to the author's idea, it is necessary to study such basic combinational circuits as a decoder, encoder, multiplexer and adder and learn how to use them when designing digital combinational circuits of arbitrary complexity. It is not superfluous to recall that so far we are studying only such devices that, in a logical sense, on each input and output line can have one of two values - either 0 or 1.

Decoder is a device that generally has k inputs and 2^k (two to the power of k) outputs. Hand the output of the decoder will be 1 on that line, the number (code) of which matches the code of the binary number at the input. The rest of the lines will be zeros. For example, if the input is 011 (the binary code of the number 3), then the output will be 1 on line Y3, and all other lines will be zeros. Figure 1 shows a general view of the decoder, in which the inputs and outputs are represented as buses, respectively, from k and 2^k lines.

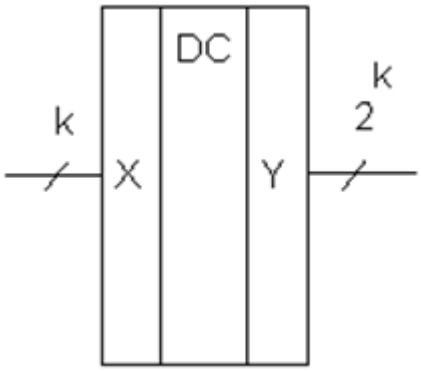

Figure 1. Decoder for k inputs and 2^k outputs

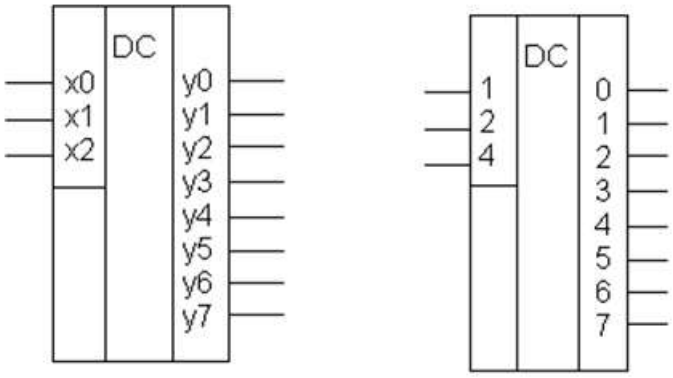

Figure 2. Two views of the decoder representation with 3 inputs and 8 outputs

Figure 2 shows examples of conventional graphic symbols of decoders that are found in the technical literature. The left figure shows that the inputs are usually denoted X0, X1, X2 (the lower digit is the highest), and the outputs are Y0,Y1,Y2,Y3,Y4,Y5,Y6,Y7. The right figure shows an alternative graphic symbol, which clearly shows that the bottom bit has a weight of 4, the middle one has a weight of 2, and the top one has a weight of 1. The outputs are denoted by numbers from 0 to 7, which are the only ones that can be obtained with all possible combinations of zeros and units at the inputs (in particular, 000 corresponds to the number 0, and 111 to the number 7).

According to the above definition, it is easy to compile a truth table (Figure 3), for example, for the decoder shown in Figure 2. To do this, on the left side (Inputs) we write out (in ascending order of the binary code) all possible combinations at the inputs, and on the right part, we write out the corresponding output combinations for each input combination.

x2	x1	x0	y0	y1	y2	y3	y4	y5	y6	y7
4	2	1	0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

$y0 = \overline{x2} \,\&\, \overline{x1} \,\&\, \overline{x0}$

$y1 = \overline{x2} \,\&\, \overline{x1} \,\&\, x0$

$y2 = \overline{x2} \,\&\, x1 \,\&\, \overline{x0}$

$y3 = \overline{x2} \,\&\, x1 \,\&\, x0$

$y4 = x2 \,\&\, \overline{x1} \,\&\, \overline{x0}$

$y5 = x2 \,\&\, \overline{x1} \,\&\, x0$

$y6 = x2 \,\&\, x1 \,\&\, \overline{x0}$

$y7 = x2 \,\&\, x1 \,\&\, x0$

Figure 3. Truth table and logic functions of a decoder with 3 inputs and 8 outputs

In order to obtain logical functions from truth tables (Figure 3), it is enough for each output (each column) to write out the disjunction of the terms corresponding to the input combinations, for which the output is 1. In this case, the input variable is

inverted if its value is 0. For example, consider the output y3. It is equal to 1 only with one input combination: 011 (that is, when x2=0, x1=1, x0=1). So the Boolean function for y3 is:

Y3=~x2 & x1 & x0

(the ~ sign denotes the negation of the variable it precedes).

Encoder. This is a device that performs the opposite function in relation to the decoder. In general, the encoder has 2^k input lines and k output lines (buses X and Y, respectively), there is also an additional single output G (Figure 4).

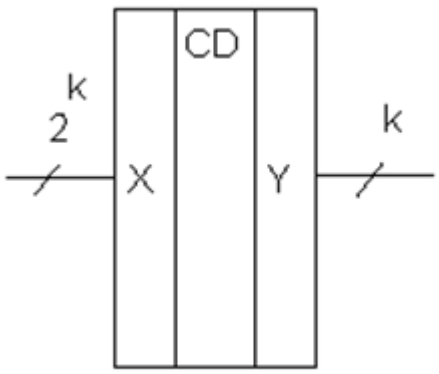

Figure 4. Encoder for 2^k inputs and k outputs

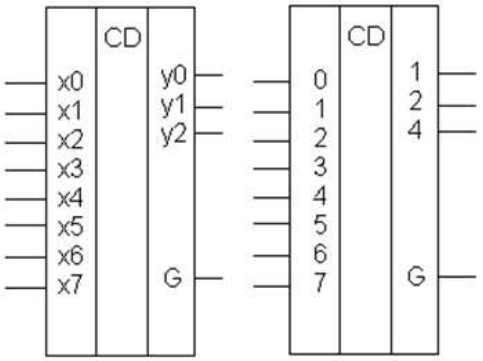

Figure 5. Two views of the encoder with 8 inputs and 3 data outputs

At the entrances X, 1 is on only one of the inputs and 0 on all other inputs. Then at the output Y there is a binary representation of the number of the line through which 1 came. On the G line there is a 1 if at least one of the inputs has a 1 and on the G line there is a 0 if all the X inputs are 0. For example, if the lines X3 is 1, and on all other lines X is 0, then the output should be the binary representation of the number 3 (that is, 1 on the Y0 and Y1 lines and 0 on all other Y lines). The G line should be 1 (there are 1 at the inputs).

Figure 5 shows examples of graphic encoder from 8 lines to 3, which are found in the technical liter-

In order to obtain logical functions from truth tables (Figure 7), it is enough for each output (each

ature. In the left figure, the inputs are displayed as x0 … x7, and outputs y0 … y2. In the right figure, the lines are displayed by numbers (0 ... 7), and the outputs are displayed by the weights of the corresponding lines (1,2,4), the lower digit is the most significant one.

According to the above definition, it is easy to compile a truth table (Figure 6), for example, for the encoder shown in Figure 5. To do this, on the left side (Inputs) we write out all possible combinations at the inputs, and on the right side we write out the corresponding output combinations for each input combinations (Figure 6)

x0 0	x1 1	x2 2	x3 3	x4 4	x5 5	x6 6	x7 7	y2 4	y1 2	y0 1	G
0	0	0	0	0	0	0	0	*	*	*	0
1	0	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	1	1
0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	1	1	1
0	0	0	0	1	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	1	0	1	1
0	0	0	0	0	0	1	0	1	1	0	1
0	0	0	0	0	0	0	1	1	1	1	1

.

Figure 6. Truth table encoder from 8 lines to 3

$$G = x0 \vee x1 \vee x2 \vee x3 \vee x4 \vee x5 \vee x6 \vee x7$$

$$y0 = \overline{x0} \,\&\, x1 \,\&\, \overline{x2} \,\&\, \overline{x3} \,\&\, \overline{x4} \,\&\, \overline{x5} \,\&\, \overline{x6} \,\&\, \overline{x7} \vee x0 \,\&\, \overline{x1} \,\&\, \overline{x2} \,\&\, x3 \,\&\, \overline{x4} \,\&\, \overline{x5} \,\&\, \overline{x6} \,\&\, \overline{x7} \vee \overline{x0} \,\&\, \overline{x1} \,\&\, \overline{x2} \,\&\, \overline{x3} \,\&\, \overline{x4} \,\&\, x5 \,\&\, \overline{x6} \,\&\, \overline{x7} \vee x0 \,\&\, \overline{x1} \,\&\, \overline{x2} \,\&\, \overline{x3} \,\&\, \overline{x4} \,\&\, \overline{x5} \,\&\, x6 \,\&\, \overline{x7} \vee \overline{x0} \,\&\, \overline{x1} \,\&\, \overline{x2} \,\&\, \overline{x3} \,\&\, \overline{x4} \,\&\, \overline{x5} \,\&\, \overline{x6} \,\&\, x7$$

$$y1 = \overline{x0} \,\&\, \overline{x1} \,\&\, x2 \,\&\, \overline{x3} \,\&\, \overline{x4} \,\&\, \overline{x5} \,\&\, \overline{x6} \,\&\, \overline{x7} \vee x0 \,\&\, \overline{x1} \,\&\, \overline{x2} \,\&\, x3 \,\&\, \overline{x4} \,\&\, \overline{x5} \,\&\, \overline{x6} \,\&\, \overline{x7} \vee \overline{x0} \,\&\, \overline{x1} \,\&\, \overline{x2} \,\&\, \overline{x3} \,\&\, \overline{x4} \,\&\, x5 \,\&\, \overline{x6} \,\&\, \overline{x7} \vee x0 \,\&\, \overline{x1} \,\&\, \overline{x2} \,\&\, \overline{x3} \,\&\, \overline{x4} \,\&\, \overline{x5} \,\&\, x6 \,\&\, \overline{x7} \vee \overline{x0} \,\&\, \overline{x1} \,\&\, \overline{x2} \,\&\, \overline{x3} \,\&\, \overline{x4} \,\&\, \overline{x5} \,\&\, \overline{x6} \,\&\, x7$$

$$y2 = \overline{x0} \,\&\, \overline{x1} \,\&\, \overline{x2} \,\&\, \overline{x3} \,\&\, x4 \,\&\, \overline{x5} \,\&\, \overline{x6} \,\&\, \overline{x7} \vee x0 \,\&\, \overline{x1} \,\&\, \overline{x2} \,\&\, \overline{x3} \,\&\, \overline{x4} \,\&\, x5 \,\&\, \overline{x6} \,\&\, \overline{x7} \vee \overline{x0} \,\&\, \overline{x1} \,\&\, \overline{x2} \,\&\, \overline{x3} \,\&\, \overline{x4} \,\&\, \overline{x5} \,\&\, x6 \,\&\, \overline{x7} \vee x0 \,\&\, \overline{x1} \,\&\, \overline{x2} \,\&\, \overline{x3} \,\&\, \overline{x4} \,\&\, \overline{x5} \,\&\, \overline{x6} \,\&\, x7$$

Figure 7. Logic functions of the encoder from 8 lines to 3

column) to write out the disjunction of the terms corresponding to the input combinations, for which the

output is 1. In this case, the input variable is inverted if its value is 0. For example, y2 is equal to 1 in the last four rows of the truth table and therefore the logical function for y2 includes four terms.

The asterisks on the right side of the truth table reflect the fact that in the case when all X inputs are zeros, the Y outputs do not carry useful information (after all, there should be a line number along which 1 came, but there is no one).

Note that in the practical operation of circuits it is very difficult to ensure that 1 is obtained only on one of the input lines, therefore, in practice, they use **priority encoder** at the input of which you can apply as many 1s as you like, and at the output it has the number of the topmost line, along which 1 came.

Figure 8 shows the truth table of the priority encoder. '*" on the left side of the truth table means "whether this input is 0 or 1" .

x0 0	x1 1	x2 2	x3 3	x4 4	x5 5	x6 6	x7 7	y2 4	y1 2	y0 1	G
0	0	0	0	0	0	0	0	*	*	*	0
1	*	*	*	*	*	*	*	0	0	0	1
0	1	*	*	*	*	*	*	0	0	1	1
0	0	1	*	*	*	*	*	0	1	0	1
0	0	0	1	*	*	*	*	0	1	1	1
0	0	0	0	1	*	*	*	1	0	0	1
0	0	0	0	0	1	*	*	1	0	1	1
0	0	0	0	0	0	1	*	1	1	0	1
0	0	0	0	0	0	0	1	1	1	1	1

Figure 8. Truth table
Priority encoder from 8 lines to 3

$$G = x0 \vee x1 \vee x2 \vee x3 \vee x4 \vee x5 \vee x6 \vee x7$$

$$y0 = \overline{x0} \& x1 \vee \overline{x0} \& \overline{x1} \& \overline{x2} \& x3 \vee \overline{x0} \& \overline{x1} \& \overline{x2} \& \overline{x3} \& \overline{x4} \& x5 \vee \overline{x0} \& \overline{x1} \& \overline{x2} \& \overline{x3} \& \overline{x4} \& \overline{x5} \& \overline{x6} \& x7$$

$$y1 = \overline{x0} \& \overline{x1} \& x2 \vee \overline{x0} \& \overline{x1} \& \overline{x2} \& x3 \vee \overline{x0} \& \overline{x1} \& \overline{x2} \& \overline{x3} \& \overline{x4} \& \overline{x5} \& x6 \vee \overline{x0} \& \overline{x1} \& \overline{x2} \& \overline{x3} \& \overline{x4} \& \overline{x5} \& \overline{x6} \& x7$$

$$y2 = \overline{x0} \& \overline{x1} \& \overline{x2} \& \overline{x3} \& x4 \vee \overline{x0} \& \overline{x1} \& \overline{x2} \& \overline{x3} \& \overline{x4} \& x5 \vee \overline{x0} \& \overline{x1} \& \overline{x2} \& \overline{x3} \& \overline{x4} \& \overline{x5} \& x6 \vee \overline{x0} \& \overline{x1} \& \overline{x2} \& \overline{x3} \& \overline{x4} \& \overline{x5} \& \overline{x6} \& x7$$

Figure 9. Logic functions of the priority encoder

With such a truth table, the logical functions (Figure 9) at the outputs are also simplified Y - input variables, on which the result does not depend; do not fit into the corresponding term.

Note that initially, the encoder was designated in the technical literature as CD, and the priority encoder as PRCD. However, at present, often in the technical literature, the priority encoder is also denoted by the symbols CD, perhaps because the ordinary encoder is practically not used.

Multiplexer, is a device that generally has k address lines A and 2^k data lines X and only one output Y (Figure 10). The value is transferred to this output Y from that of the input lines X, the number of which is on the address lines A. For example, if the address line has the number 3, then the value from the input line X3 will be transferred to Y.

Figure 11 shows examples of UGO encoder from 8 lines to 3, which are found in the technical literature. In the left figure, the data inputs are displayed asx0 ... x7, and address inputs a0 ... a2. In the right figure, the data inputs are displayed by numbers (0 ... 7), and the address inputs are displayed by the weights of the corresponding lines (1,2,4), the lower digit is the most significant one.

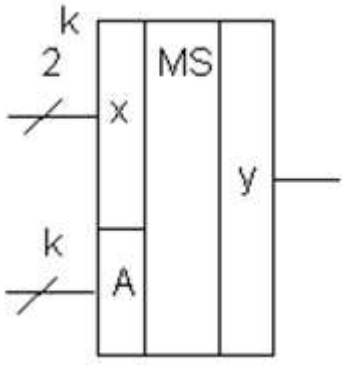

Figure 10. Multiplexer with k address lines

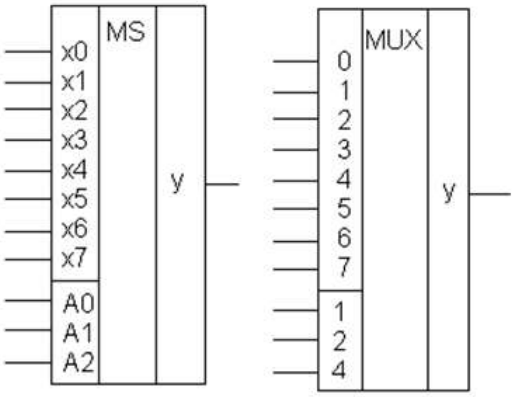

Figure 11. Two types of representation of the multiplexer with 3 address lines and 8 data lines

According to the above definition, it is easy to compile a truth table (Figure 12), for example, for the multiplexer shown in Figure 11. To do this, on the left side (Inputs) we write out all possible combinations at the inputs, and on the right side we write out the corresponding output combinations for each input combination (Figure 12).

A2 4	A1 2	A0 1	y
0	0	0	x0
0	0	1	x1
0	1	0	x2
0	1	1	x3
1	0	0	x4
1	0	1	x5
1	1	0	x6
1	1	1	x7

Figure 11. Truth table multiplexer with 3 addressable inputs

$$y = \overline{A2} \,\&\, \overline{A1} \,\&\, \overline{A0} \,\&\, x0 \vee \overline{A2} \,\&\, \overline{A1} \,\&\, A0 \,\&\, x1 \vee \overline{A2} \,\&\, A1 \,\&\, \overline{A0} \,\&\, x2 \vee \overline{A2} \,\&\, A1 \,\&\, A0 \,\&\, x3 \vee A2 \,\&\, \overline{A1} \,\&\, \overline{A0} \,\&\, x4 \vee A2 \,\&\, \overline{A1} \,\&\, A0 \,\&\, x5 \vee A2 \,\&\, A1 \,\&\, \overline{A0} \,\&\, x6 \vee A2 \,\&\, A1 \,\&\, A0 \,\&\, x7 \vee$$

Figure 12. Logic function multiplexer with 3 addressable inputs

Adder is a device that adds two k-bit numbers A and B, given an input carry C0. The result of the addition is a k-bit sum S and an output carry P (Figure 13).

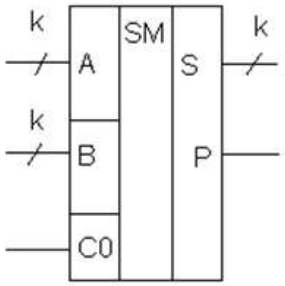

Figure 13. K-bit adder

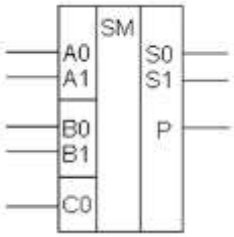

Figure 14. Two-digit adder

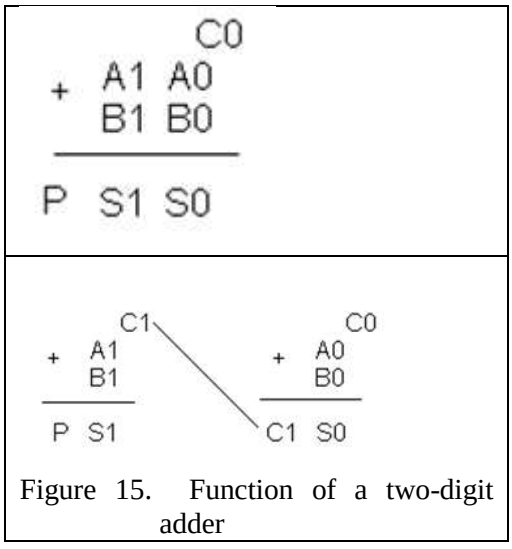

Figure 15. Function of a two-digit adder

Figure 14 shows an example of a UGO of a two-digit adder. Figure 15 shows the process of calculating the result with a two-digit adder: first, the least significant digits of the input numbers are addedA0 and B0 and input carry. It turns out the least significant digit of the sum S0 and an internal transfer from the 0th digit to the 1st, which is designated as C1. Then the most significant digits of the input numbers A1 and B1 and the internal carry C1 are added, the most significant digit of the sum S1 and the output carry P are obtained. Similarly, adders of a larger capacity are added bit by bit with the formation of a carry to the next digit.

The truth table of a single-bit adder is shown in Figure 16.

A1 B1 C1 / A0 B0 C0	P C1	S1 S0
0 0 0	0	0
0 0 1	0	1
0 1 0	0	1
0 1 1	1	0
1 0 0	0	1
1 0 1	1	0
1 1 0	1	0
1 1 1	1	1

Figure 16. Truth table of a single-digit adder

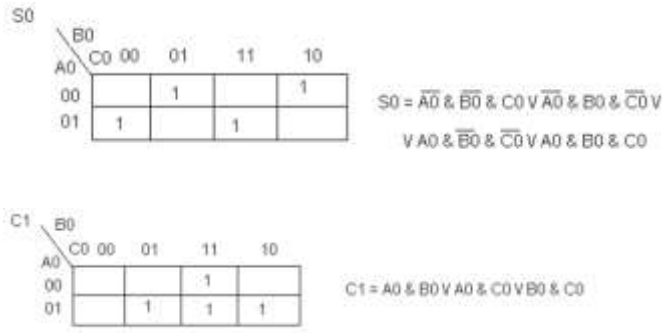

Figure 17. Logic functions of a single-bit adder

Figure 17 shows the process of obtaining the logic functions of a one-bit adder using Carnot maps for three variables. Obviously, S0 is not subject to minimization, and the logical function C1 is simplified to three terms.

Figure 18 shows the logical functions of a two-bit adder

$$S0 = \bar{A0} \,\&\, \bar{B0} \,\&\, C0 \vee \bar{A0} \,\&\, B0 \,\&\, \bar{C0} \vee A0 \,\&\, \bar{B0} \,\&\, \bar{C0} \vee A0 \,\&\, B0 \,\&\, C0$$

$$C1 = A0 \,\&\, B0 \vee A0 \,\&\, C0 \vee B0 \,\&\, C0$$

$$S1 = \bar{A1} \,\&\, \bar{B1} \,\&\, C1 \vee \bar{A1} \,\&\, B1 \,\&\, \bar{C1} \vee A1 \,\&\, \bar{B1} \,\&\, \bar{C1} \vee A1 \,\&\, B1 \,\&\, C1$$

$$P = A1 \,\&\, B1 \vee A1 \,\&\, C1 \vee B1 \,\&\, C1$$

Figure 18. Logic functions of a two-digit adder

In fact, this is the whole theory that needs to be mastered when studying this topic. However, it is not enough to learn and be able to reproduce the corresponding figures and definitions. It is necessary to learn how to use these functional blocks when solving problems for the analysis and design of functionally more complex digital devices.

3.2 Library of standard components

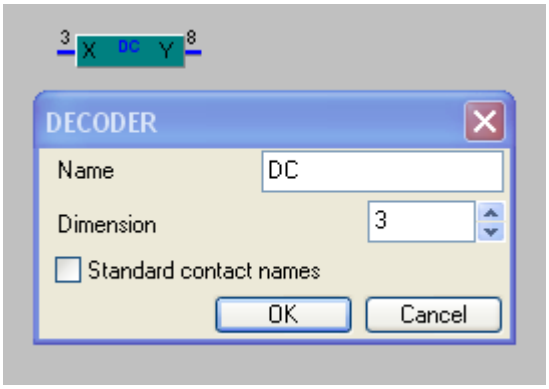

Figure 19. Decoder

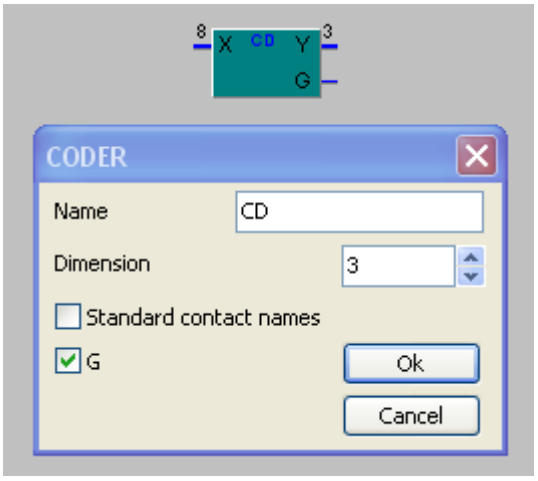

Figure 20. Priority encoder

The Standard component library of the HLCCAD system includes the elements shown in Figures 19-24. By default, the decoder (Figure 19) has a dimension (the number of digits at the input) of 3, which can be changed by the user. Similarly, the priority encoder (Figure 20) by default has a dimension (number of digits at the output) of 3, which can also be arbitrarily changed by the user.

There are 3 default multiplexers shown in Figures 21-23 respectively.

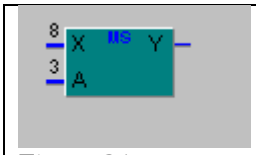

Figure 21. Multiplexer MS8

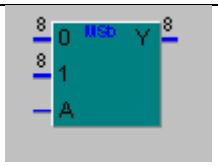

Figure 22. Multiplexer MSb8x2

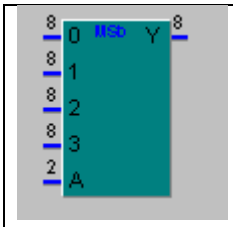

Figure 23.
Multiplexer MSb8x4

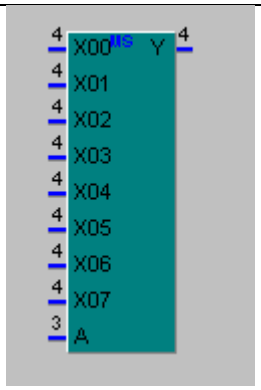

Figure 24.
Multiplexer MSb4x8

Multiplexer MS8– the standard multiplexer described in theory above has 8 data lines, 3 address lines and one output line.

Multiplexer MS8b8x2 This is the so-called bus multiplexer. It has an 8-bit bus at the output and two 8-bit data buses (0 and 1) and one address line A at the input. If A=0, then the values from the input bus 0 are bit-by-bit transferred to the output bus Y. If A=1 , then the values from input bus 1 are bit-by-bit transferred to the output bus Y.

Multiplexer MSb8x4 is also a bus multiplexer. It has an 8-bit Y bus at the output, four 8-bit data buses (0, 1, 2, 3) and a 2-bit address bus at the input.
If the address input is 00, then data from bus 0 is transferred to the output.
If the address input is 01 (leftmost bit), then data from bus 1 is transferred to the output.
If the address input is 10, then data from bus 2 is transferred to the output.
If the address input is 11, then data from bus 3 is transferred to the output.

In addition, any of these multiplexers can be converted to any arbitrary multiplexer by changing the number of bits in the address line and the number of bits in the data buses. For example, Figure 24 shows an MSb4x8 multiplexer with 3 address lines and 8 four-bit data buses. The output is also a four-bit bus. The first number in the notation (4) indicates the width of the input and output data buses, and the second number (8) indicates how many input data buses will be (simultaneously specifying the number of address lines required, as the base 2 logarithm of the number of input data buses: 3 is the logarithm of 8 based on 2) .

The adder (Figure 20) is 8-bit by default. However, this bit depth can be arbitrarily changed, in addition, it is possible not to display (in case of uselessness) the carry, overflow and sum contacts.

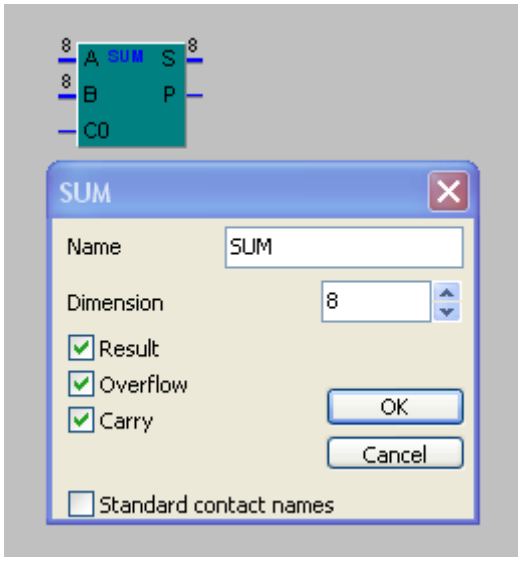

Figure 25. Adder

3.3. Tasks for simulation of standard components

Figures 26-30 show tasks in which the student is presented with a standard component and randomly generated input values. Using the information known from theoretical studies, it is required to calculate the values at the outputs of the component for given values at the input. The input data values are generated 10 times. The task is considered completed successfully only if the correct values were entered for all 10 times at each output (by clicking on the digit, the values are reversed).

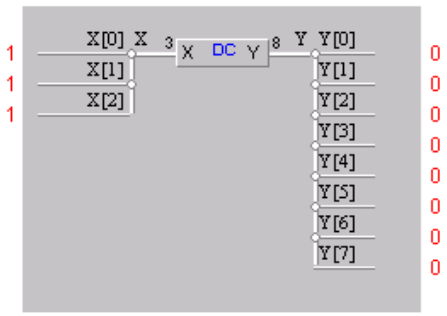

Figure 26. Decoder simulation

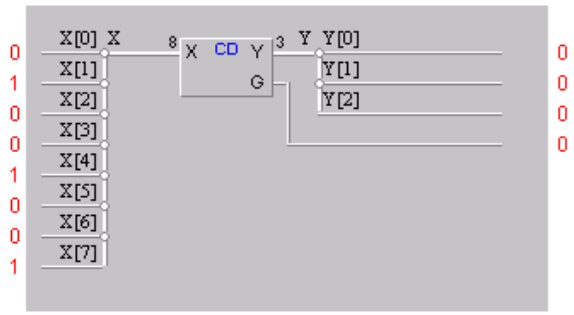

Figure 27. Simulation of the encoder

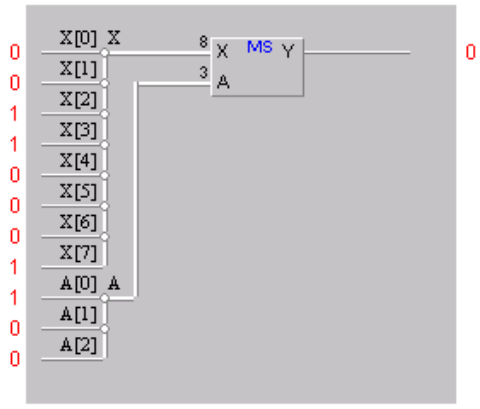

Figure 28. Simulation of the MS8 multiplexer

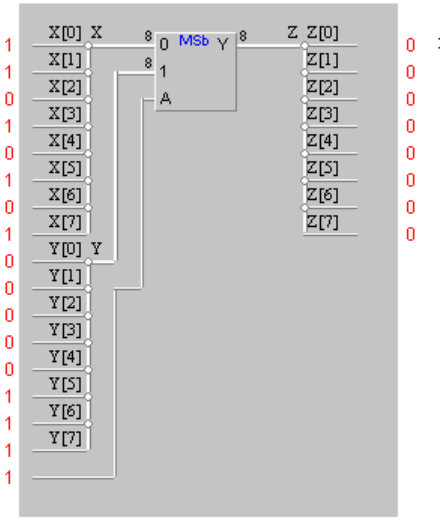

Figure 29. Simulation of the MSb8x2 multiplexer

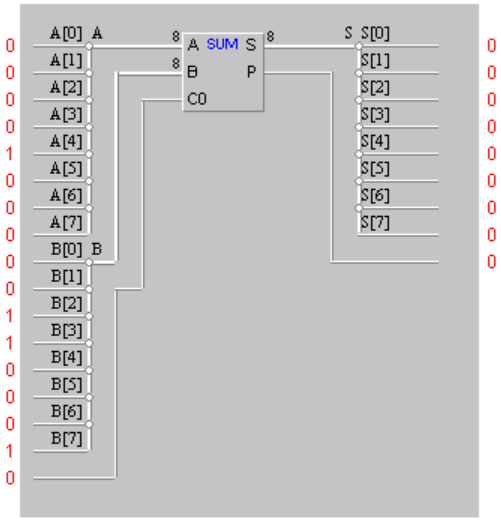

Figure 30. Simulation of the adder

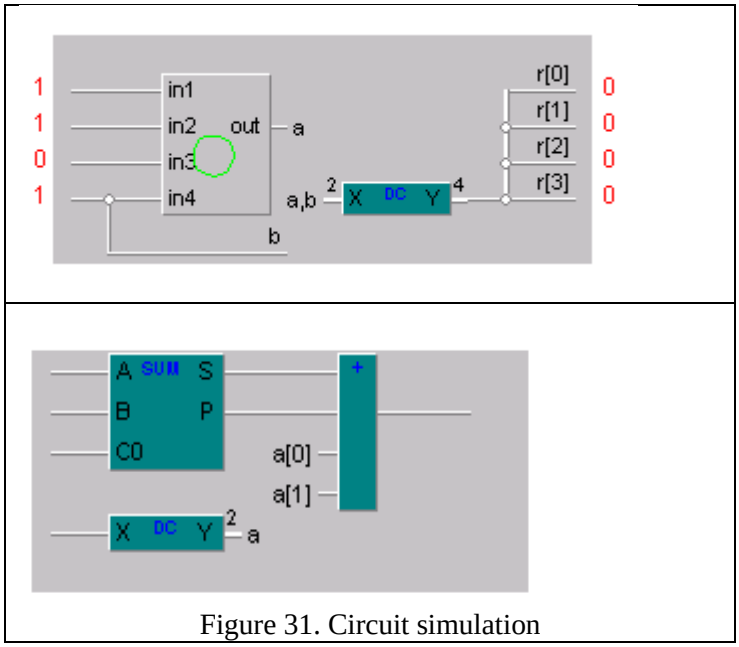

Figure 31. Circuit simulation

3.4 Tasks for the simulation of circuits made up of standard components

Figure 31 shows the task for simulating a hierarchical scheme. It is presented at the top of Figure 31. The content of the sub-circuit can be seen by clicking on the circle (the opening sub-circuit is shown at the bottom of Figure 31).

Remaining in form the same as the previously described previous buildings, in fact, this task forms the skills of analyzing complex digital circuits, when knowing how each of the functional elements works (in this case, the decoder, adder and element XOR) , you need to understand how the circuit made up of them works.

Such tasks, which are checked completely automatically, not only contribute to the formation of relevant skills, but also significantly improve the quality of training control. In addition, the system of tasks for circuit simulation can demonstrate to students examples of solving complex design problems, being the same way Propaedeutics tasks for subsequent design tasks.

3.5 Task for circuit design

Name	Bits	Type
X	3	In
Y	3	In
I	3	In
J	3	In
R	1	Out
S	1	Out

The first player (inputs X and Y) and the second player (in with the larger amount. If the first player wins, 1 is fed to t wins, the values at the outputs change. In the case when th outputs.

Example:
X: 001
Y: 010
I: 100
J: 011
R: 0
S: 1

Figure 32. Task for circuit design

In tasks for designing a circuit (an example in Figure 32), it is required to design in HLCCAD required device, check that the project gives the correct answer using the example from the condition, send the project for review. Presentation of tasks and sending of solutions is done using the DL.GSU.BY system. The solution is transferred to the HLCCAD system on the server side, where the student's project is executed on a pre-prepared set of tests (input data and reference answers are given). The answers received by the student project are checked against the reference answers. If all the answers of the student project match all the reference answers, the task is considered accepted. In case of at least one discrepancy, the task is considered not accepted, but the test file is returned to the student and indicates at what point in the simulation time and on which contact his project gave an answer that differs from the reference one. The student can take this test file, connect to his project, run the simulation and analyze why the error occurs in his project. After correcting the project, you can run the simulation again and so on until you receive a message that there are no errors. After that, you can send the project back for testing. The main test will already be passed. However, on the server side, the student's project is checked not only on the main test, but also on the secret one, which is not given to the student, even in case of errors. This is done to protect against unscrupulous students who "adjust" projects to known reference answers. If student projects are found that pass the main test, but do not pass the secret one, the teacher analyzes why this happened. If there is a situation in the secret test that was not tested in the main test, then it is transferred to the main test, and the secret test is replenished. If an attempt to deceive on the part of a student is detected, disciplinary sanctions are applied to him.

3.6 Assignments for the design of circuits for programs

Figure 33 shows the task for designing schemes for programs. On the above side is an assembly language program Intel 80x86, on the below - program in the C-MPA microprogramming language. The student must first understand how the program works and what problem it solves. In case of difficulties, you can run the Winter program, execute the program step by step, observing the process of changing its variables. After it became clear what the program does, you need to develop a project in HLCCAD - that is, create a scheme that solves the same problem as the presented program.

```
        jmp     begin
a       db      121
res     db      0
begin:
        mov     cx,     7
        mov     bl,     128
l1:
        mov     al,     a
        mov     ah,     0
        and     al,     bl
        jne     r
        shr     bl
        loop    l1
r:
        mov     res,    cl
halt:
        jmp     halt            ;$E
```

```
int     __in  __bits(32) n;
int     __out  __bits(32) z;

void main(void) {
        int     __bits(32) r;
        int     __bits(32) i;
        int     __bits(32) mask;

        r=0;
        mask=0x01;

        for(i=0; i<32; i++){
                if((n&mask) != 0) {
                        r = (r << 1) |

                }
                mask = mask << 1;
        }
        z=r;
}
```

Figure 33. Tasks for designing circuits for programs

3.7 Test questions and flash tasks

Figure 34 shows examples of test questions offered to students to test knowledge on the above topic.

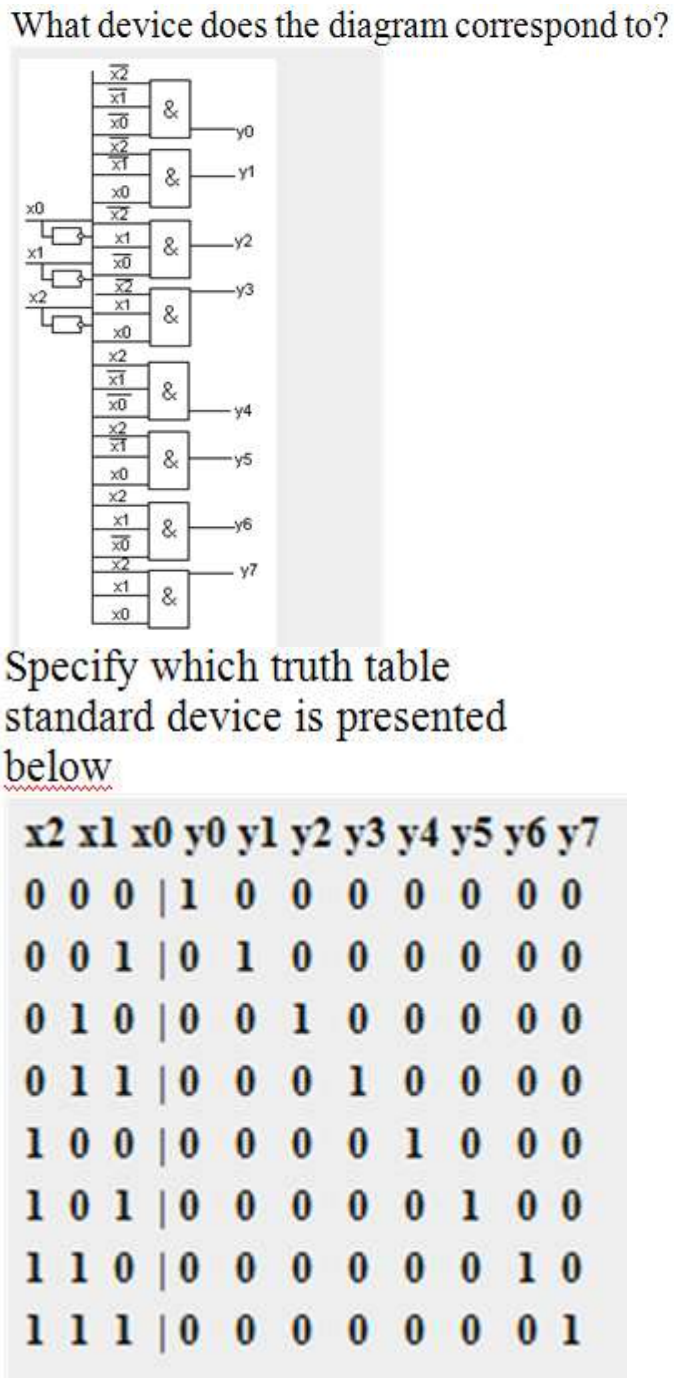

Figure 34. Test questions examples

Figure 35 shows examples of flash tasks performed by clicking in the positions of the picture corresponding to the question.

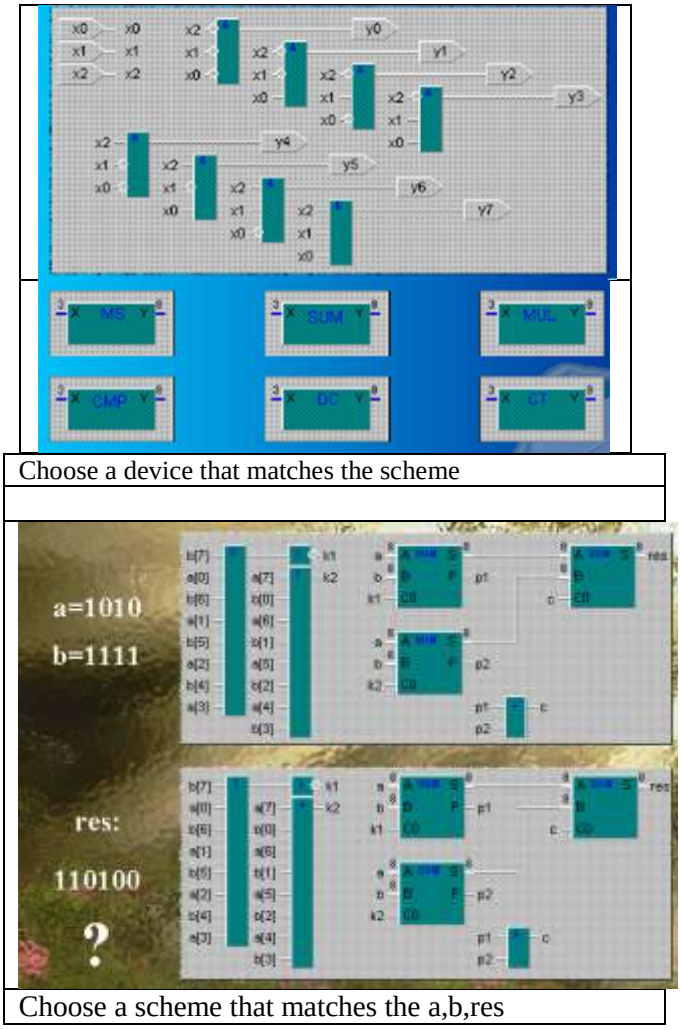

Figure 36. Examples of flash tasks

3.8 Technology of using practical tasks

Lecture: After presenting the above theoretical material using a computer and projection equipment, students who have a laptop at least on each table (and often almost every student personally) are offered in teams (no more than two people) or personally solve problems of their choice from all the types described above. At the same time, more complex tasks are evaluated by a large number of bonus points, which stimulates the solution of the most difficult tasks that the student is able to solve. In addition, earned bonuses are divided by the number of people in the team, which encourages one to decide if the student is able to do it. A variety and a large number of tasks of varying complexity leads to the fact that each student finds tasks that correspond to their level of preparation and motivation.

Practice: At a practical lesson, a student, depending on his level of training, can choose a task / group of tasks of any of the types described above in any of the sections (in order of increasing complexity and, accordingly, increasing marks): training, practice control, individual tasks. In individual tasks, the solution of the problem is credited only to the student who first solved this problem. In addition, in individual

tasks in each topic, for example "Combination schemes", only one solved problem increases the mark for each student.

Independent work: Students can complete tasks at home from study, and individual tasks during absenteeism, or simply to improve grades or self-study.

4 Conclusion

This article describes the methodology developed by the author and repeatedly tested for studying the topic "Logical and combinational circuits" focused on working in groups of students with fundamentally different levels of motivation and preliminary training. A serious technical basis of the methodology is the developed instrumental system of distance learning (Distance Learning Belarus - http://dl.gsu.by). The introduction of this teaching methodology provided significant changes in the quality of education, especially for the least prepared and motivated category of students. At the same time, the most prepared students are also satisfied with this approach to learning.

References

[1] Aznam N., Perdana R., Jumadi J, Nurcahyo H., Wiyatmo Y. The Implementation of Blended Learning and Peer Tutor Strategies in Pandemic Era: A Systematic Review // Advances in Social Science, Education and Humanities Research, Vol. . 541, 2021, pp.906-914

[2] Kakeshita T., Ohtsuki M. Survey and Analysis of Computing Education at Japanese Universities: Non-IT Departments and Courses // Olympiads in Informatics, Vol. 13, 2019, pp.57-80

[3] Jones K., Ravishankar S. Higher Education 4.0. The Digital Transformation of Classroom Lectures to Blended Learning // Springer Singapore, 2021

[4] Zaugg H., Graham C., Lim C., Wang T. Current and Future Directions of Blended Learning and Teaching in Asia // chapter 16 in the book "Blended Learning for Inclusive and Quality Higher Education in Asia", 2021 , pp.301-327

[5] Stahl G. Redesigning Mathematical Curriculum for Blended Learning // Education. Sciences, Vol. 11, 2021, pp.165-177

[6] Astarilla L., Warman D. The Effect of Google Classroom in Blended Learning on University Students' English Ability // Journal of English for Academic, Vol 8, No 1, 2021, pp.12-23

[7] Antwi-Boampong A. Blended Learning Adoption in Higher Education: Presenting the Lived Experiences of Students in a Public University from a Developing Country // The Turkish Online Journal of Educational Technology, Vol. 20, Issue 2, 2021, pp.14-22

[8] Oktaria S., Sasongko R., Kristiawan M. Development of Blended Learning Designs using Moodle to Support Academics of The Curriculum in University of Bengkulu // Jurnal Studi Guru dan Pembelajaran (Indonesia), Vol. 4, no. 1, 2021, pp.118-126

[9] Dolinsky M. Teaching Algorithms and Programming First Year University Students on Base of Distance Learning System DL.GSU.BY // WSEAS Transactions on Advances in Engineering Education, vol. 19, pp. 52-57, 2022

[10] Dolinsky M. Experience of Blended Learning the Fundamentals of Digital Electronics for First/Second Year University students On Base of Distance Learning System DL.GSU.BY // International Journal of Education and Learning Systems, vol. 7, pp. 59-64, 2022

[11] Dolinsky M Experience of Blended Learning in the Subject - Architecture of Computers // Journal of Information Technology and Digital World, vol. 4, issue 3, pp. 167-182, 2022

[12] Dolinsky M. Tool HLCCAD for Blended Learning the Fundamentals of Digital Electronics" // International Journal of Circuits and Electronics, vol. 7, pp. 47-55, 2022

[13] Dolinsky M. Instrumental System of Distance Learning DL.GSU.BY and Examples of its Application // Global Journal of Computer Science and Technology Interdisciplinary, vol. 22, issue 1, pp. 45-53, 2022

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The author contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The author has no conflict of interest to declare that is relevant to the content of this article.