FRI-ONLINE-1-CCT2-12

IMPLEMENTATION OF A MAGNITUDE COMPARATOR USING COMPUTER-BASED TOOLS[24]

Diego Fierro Álvarez
Erasmus Student at the Department of Telecommunications, University of Ruse
Universidad de Leon, Spain
E-mail: dfiere00@estudiantes.unileon.es

Assist. Prof. Adriana Borodzhieva, PhD
Department of Telecommunications,
University of Ruse "Angel Kanchev", Bulgaria
Tel.: +359 82 888 734
E-mail: aborodzhieva@uni-ruse.bg

Abstract: *The paper presents an approach for designing a magnitude comparator comparing two four-bit binary numbers. The magnitude comparator is implemented in Logisim using integrated circuits (IC) from the library 74xx and tested for different input combinations. Then the circuit is built in the environment of ISE Project Navigator necessary for programming the FPGA-based laboratory board, developed at the University of Ruse and used in the educational process. The work is related with the project in the course "Digital Devices" for students-bachelors of the specialty "Electronics".*

Keywords: *Magnitude Comparators, Digital Devices, Logisim, FPGA, Xilinx.*

INTRODUCTION

The biggest challenge for lecturers at the universities is to attract students' attention during the classes and to generate ideas effectively for creating lasting impressions. To effectively address this challenge, lecturers at the universities need to implement innovative ideas, making classroom experience much more attractive to students, for example: 1) creative teaching and learning; 2) audio and video tools; 3) real-world training; 4) brainstorming sessions; 5) stimulating the classroom environment; 6) teamwork, 7) project-based learning, etc. Some solutions using computer-based tools for increasing the attractiveness and effectiveness of the course "Digital Devices" at the University of Ruse are presented (Borodzhieva, A., Stoev, I., Mutkov, V., 2019).

The course "Digital Devices" introduces students-bachelor of the specialty "Electronics" at the University of Ruse with the main issues of digital electronics. The aim of this course is to connect the functionalities of the digital components with their microelectronic base and their applications in constructing pulse and digital devices. Various combinational logic circuits, such as arithmetic circuits, encoders and decoders, multiplexers and demultiplexers, code converters and comparators, are studied in the course. One of the topic covered in the course is "Synthesis and analysis of digital comparators".

The paper presents an approach for synthesis and analysis of a magnitude comparator comparing two four-bit binary numbers. The magnitude comparator is implemented in Logisim using integrated circuits (IC) from the library 74xx and tested for different input combinations. Then the circuit is built in ISE Project Navigator necessary for programming the FPGA-based laboratory board, developed at the University of Ruse and used in the educational process. The work is related with the project in the course "Digital Devices" for students-bachelors of the specialty "Electronics".

[24] The paper is presented on 13 November 2020 with original title: IMPLEMENTATION OF A MAGNITUDE COMPARATOR USING COMPUTER-BASED TOOLS

EXPOSITION

Digital Comparators

Digital comparators are used to compare two binary numbers with a certain number of bits and to give information: 1) whether the numbers are equal or not; 2) if the numbers are not equal, which of the two numbers is greater. The first comparators are called comparators for identity (equality) (Borodzhieva, A., Stoev, I., Mutkov, V., 2019).

A possible circuit of a digital comparator for identity of two four-bit numbers using XOR (exclusive OR, adding modulo 2) logic gates ($\oplus$) is shown below with the explanation of XOR logic gates (truth table), Fig. 1 (Borodzhieva, A., Stoev, I., Mutkov, V., 2019):

Truth table of the function XOR ("adding modulo 2", "exclusive OR", "logical inequality")

$Y = x_1 \oplus x_2 = \bar{x}_1.x_2 \vee x_1.\bar{x}_2$

№	x_1	x_2	Y
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

Structure circuit of the digital comparator, based on XOR logic gates:

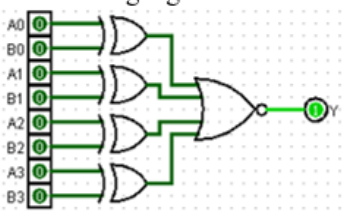


Fig. 1. A possible circuit of a digital comparator for identity with XOR logic gates

Its principle of operation is as follows: if two four-bit numbers $A = A_3A_2A_1A_0$ and $B = B_3B_2B_1B_0$ are equal, i.e. $A = B \Rightarrow A_i = B_i$ for $i = 0 \div 3$, then the following conclusion is drawn:

$$E_i = A_i \oplus B_i = 0 \Rightarrow Y = \overline{E_0 \vee E_1 \vee E_2 \vee E_3} = \overline{0 \vee 0 \vee 0 \vee 0} = \bar{0} = 1 \qquad (1)$$

If the two four-bit numbers are different (it is sufficient for only one pair of identical bits to be different, for example, $A_0 = B_0$, $A_1 \neq B_1$, $A_2 = B_2$ and $A_3 = B_3$, i.e. $E_0 = A_0 \oplus B_0 = 0$, $E_1 = A_1 \oplus B_1 = 1$, $E_2 = A_2 \oplus B_2 = 0$ and $E_3 = A_3 \oplus B_3 = 0$), then the following conclusion is drawn:

$$Y = \overline{E_0 \vee E_1 \vee E_2 \vee E_3} = \overline{0 \vee 1 \vee 0 \vee 0} = \bar{1} = 0. \qquad (2)$$

If all identical bits are equal, i.e. $A_i = B_i$ for $i = 0 \div 3$, then the outputs of XOR logic gates will be 0s and the output *Y* will be 1 (Borodzhieva, A., Stoev, I., Mutkov, V., 2019).

This scheme is easily extended with the aim of comparing numbers with more bits. For example, to compare two 8-bit binary numbers, it is necessary to add 4 additional XOR logic gates and to use eight-input NOR logic gate. But the main disadvantage of this circuit is that in case of inequality of numbers it cannot "say" which of the two numbers is greater.

More complicated comparators are comparators that compare two numbers. Example of a digital comparator comparing two 2-bit numbers is presented below with its truth table (Table 1). The synthesis of the digital comparator is performed by the classical methods for synthesis of combinational logic circuits. Logisim (http://www.cburch.com/logisim/) can be used for synthesis purposes. This is illustrated in Fig. 2 and described in detail in (Borodzhieva, A., Stoev, I., Mutkov, V., 2019).

The input variables A1, A0, B1, B0 (Fig. 2a), the output variables F0, F1 and F2, respectively representing the outputs $F_{A>B}$, $F_{A=B}$ and $F_{A<B}$ (Fig. 2b), and the truth table of the comparator (Fig. 2c) are defined in the *Combinational Analysis* module in Logisim. Then the minimization of the output functions with Karnaugh maps is performed (Fig. 2d). Logisim "suggests" constructing the circuit in: 1) Basis 1 (AND, OR, NOT logic gates) (Fig. 2e); 2) NAND logic gates without

limitations of the inputs number; 3) 2-input NAND logic gates. The circuit built by Logisim (Fig. 2e) is not optimized and it should be optimized (Fig. 2f) with regard to duplicated inverters before building it in ISE Project Navigator.

Table 1. Truth table of a digital comparator comparing two two-bit numbers

№	A_1	A_0	B_1	B_0	$A_{(10)}$	 = >	$B_{(10)}$	$F_{A>B}$	$F_{A=B}$	$F_{A<B}$
0	0	0	0	0	0	=	0	0	1	0
1	0	0	0	1	0	<	1	0	0	1
2	0	0	1	0	0	<	2	0	0	1
3	0	0	1	1	0	<	3	0	0	1
4	0	1	0	0	1	>	0	1	0	0
5	0	1	0	1	1	=	1	0	1	0
6	0	1	1	0	1	<	2	0	0	1
7	0	1	1	1	1	<	3	0	0	1
8	1	0	0	0	2	>	0	1	0	0
9	1	0	0	1	2	>	1	1	0	0
10	1	0	1	0	2	=	2	0	1	0
11	1	0	1	1	2	<	3	0	0	1
12	1	1	0	0	3	>	0	1	0	0
13	1	1	0	1	3	>	1	1	0	0
14	1	1	1	0	3	>	2	1	0	0
15	1	1	1	1	3	=	3	0	1	0

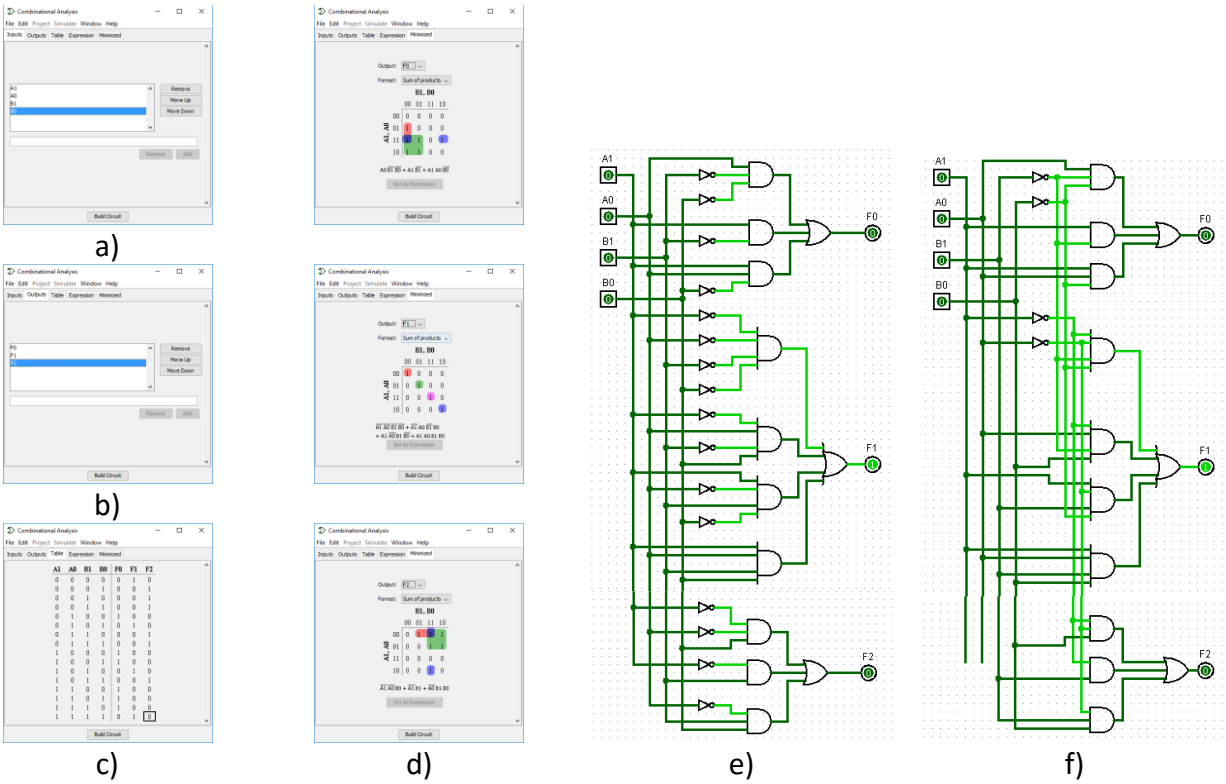


Fig. 2. Synthesis of the digital comparator in Logisim, in the Combinational Analysis module: a) input variables; b) output variables; c) truth table; d) minimization of the output functions; e) structural circuit in Basis 1, generated by Logisim (not optimized); f) structural circuit in Basis 1 optimized for the duplicated inverters (Borodzhieva, A., Stoev, I., Mutkov, V., 2019)

The output functions of the digital comparator after minimizing with Karnaugh maps are as follows:

$$F_0 = A_0\bar{B}_1\bar{B}_0 \vee A_1\bar{B}_1 \vee A_1A_0\bar{B}_0$$
$$F_1 = \bar{A}_1\bar{A}_0\bar{B}_1\bar{B}_0 \vee \bar{A}_1A_0\bar{B}_1B_0 \vee A_1\bar{A}_0B_1\bar{B}_0 \vee A_1A_0B_1B_0 \quad (3)$$
$$F_2 = \bar{A}_1\bar{A}_0B_0 \vee \bar{A}_1B_1 \vee \bar{A}_0B_1B_0$$

This approach is not applicable for synthesis of digital comparators comparing numbers with more bits because the truth table will consist of more rows. For example, for a digital comparator comparing numbers with 3 bits the truth table will consist of $2^6 = 64$ rows; for a digital comparator comparing numbers with 4 bits the truth table will consist of $2^8 = 256$ rows.

The synthesis of such comparators uses the approach described below in the paper, which avoids building the truth table of the device and minimizing the output functions with Karnaugh maps.

Magnitude Comparators

A *magnitude comparator* is a combinational circuit that compares two given numbers and determines whether the first number *A* is equal to, less than or greater than the second number *B*. The output of the device is in the form of three binary variables representing the conditions $A = B$, $A > B$ and $A < B$, where *A* and *B* are the two numbers being compared. Depending on the relative magnitude of the two numbers, the relevant output changes its state. If the two numbers *A* and *B* are four-bit binary numbers and are designated as ($A_3A_2A_1A_0$) and ($B_3B_2B_1B_0$), the two numbers will be equal if all pairs of significant bits are equal, i.e. $A_3 = B_3$, $A_2 = B_2$, $A_1 = B_1$ and $A_0 = B_0$. In order to determine whether *A* is greater than or less than *B*, the relative magnitude of the pairs of significant bits, starting from the most significant bits (A_3, B_3) are inspected. If the bits of the pair under examination (A_3, B_3) are equal, then the comparison is done by successively comparing the next adjacent lower pair of bits (A_2, B_2). The comparison continues until a pair of unequal bits is reached. In the pair of unequal bits, if $A_i = 1$ and $B_i = 0$, then $A > B$, and if $A_i = 0$, $B_i = 1$ then $A < B$. If *X*, *Y* and *Z* are three variables respectively representing the $A = B$, $A > B$ and $A < B$ conditions, then the Boolean expression representing these conditions are given by the equations:

$$X = x_3.x_2.x_1.x_0 \text{, where } x_i = A_i.B_i \vee \bar{A}_i.\bar{B}_i \quad (4)$$

$$Y = A_3.\bar{B}_3 \vee x_3.A_2.\bar{B}_2 \vee x_3.x_2.A_1.\bar{B}_1 \vee x_3.x_2.x_1.A_0.\bar{B}_0 \quad (5)$$

$$Z = \bar{A}_3.B_3 \vee x_3.\bar{A}_2.B_2 \vee x_3.x_2.\bar{A}_1.B_1 \vee x_3.x_2.x_1.\bar{A}_0.B_0 \quad (6)$$

In equation (4) x_3 will be "1" only when both A_3 and B_3 are equal. Similarly, conditions for x_2, x_1 and x_0 to be "1" respectively are equal A_2 and B_2, equal A_1 and B_1, and equal A_0 and B_0. Using AND of x_3, x_2, x_1 and x_0 ensures that *X* will be "1" when x_3, x_2, x_1 and x_0 are equal to 1. Thus, $X = 1$ means that $A = B$. On similar lines, it can be visualized that equations (5) and (6) respectively represent $A > B$ and $A < B$ conditions (Maini, A. K., 2007), (Kumar, A., 2011), (Floyd, Th., 2006). Fig. 3 represents a description of the cases where the output functions are equal to 1 in regard to the terms in equations (4), (5) and (6), respectively in regard to the input variables. Fig. 4 shows the logic diagram of a four-bit magnitude comparator.

$$x_i = A_i.B_i \vee \bar{A}_i.\bar{B}_i$$
1 1 1 or 0 0

$$X = x_3.x_2.x_1.x_0$$
1 1 1 1 1 A = B

$$Y = A_3.\bar{B}_3 \vee x_3.A_2.\bar{B}_2 \vee x_3.x_2.A_1.\bar{B}_1 \vee x_3.x_2.x_1.A_0.\bar{B}_0$$
1 1 0 or 1 1 0 or 1 1 1 0 or 1 1 1 1 0 A > B

$$Z = \bar{A}_3.B_3 \vee x_3.\bar{A}_2.B_2 \vee x_3.x_2.\bar{A}_1.B_1 \vee x_3.x_2.x_1.\bar{A}_0.B_0$$
1 0 1 or 1 0 1 or 1 1 0 1 or 1 1 1 0 1 A < B

Fig. 3. Description of the cases where the output functions are equal to 1 in regard to the terms in the output equations of the comparator

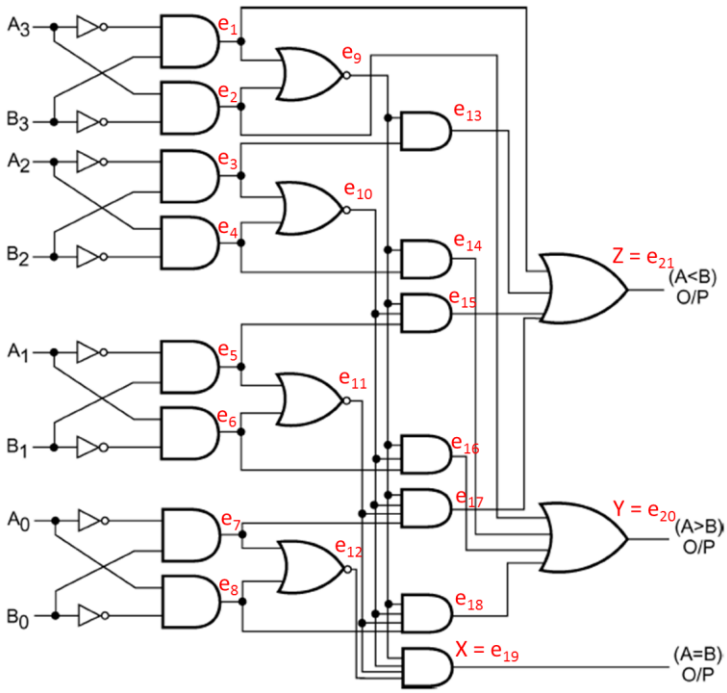


Fig. 4. Four-bit magnitude comparator

The intermediate variables at the outputs of each of the logic gates (excluding the inverters) are presented in the logic diagram in Fig. 4. These intermediate variables are used to write shorter expressions in subsequent implementations of the comparator.

$$e_1 = \overline{A}_3.B_3,\ e_2 = A_3.\overline{B}_3,\ e_3 = \overline{A}_2.B_2,\ e_4 = A_2.\overline{B}_2,$$
$$e_5 = \overline{A}_1.B_1,\ e_6 = A_1.\overline{B}_1,\ e_7 = \overline{A}_0.B_0,\ e_8 = A_0.\overline{B}_0 \quad (7)$$

$$e_9 = \overline{e_1 \vee e_2},\ e_{10} = \overline{e_3 \vee e_4},\ e_{11} = \overline{e_5 \vee e_6},\ e_{12} = \overline{e_7 \vee e_8} \quad (8)$$

$$e_{13} = e_9.e_3,\ e_{14} = e_9.e_4,\ e_{15} = e_9.e_{10}.e_5,\ e_{16} = e_9.e_{10}.e_6,$$
$$e_{17} = e_9.e_{10}.e_{11}.e_7,\ e_{18} = e_9.e_{10}.e_{11}.e_8 \quad (9)$$

When building the circuit in Fig. 4, the expression $x_i = A_i.B_i \vee \overline{A}_i.\overline{B}_i$ is not used for the implementation of x_i, but its equivalent, based on the following transformation:

$$x_i = \overline{\overline{A_i.B_i \vee \overline{A}_i.\overline{B}_i}} = \overline{\overline{A_i.B_i}.\overline{\overline{A}_i.\overline{B}_i}} = \overline{\left(\overline{A}_i \vee \overline{B}_i\right).\left(A_i \vee B_i\right)} = \overline{\overline{A}_i.B_i \vee A_i.\overline{B}_i} \quad (10)$$

$$x_3 = \overline{\overline{A}_3.B_3 \vee A_3.\overline{B}_3} = \overline{e_1 \vee e_2} = e_9, \quad (10a)$$
$$x_2 = \overline{\overline{A}_2.B_2 \vee A_2.\overline{B}_2} = \overline{e_3 \vee e_4} = e_{10}$$

$$x_1 = \overline{\overline{A}_1.B_1 \vee A_1.\overline{B}_1} = \overline{e_5 \vee e_6} = e_{11}, \quad (10b)$$
$$x_0 = \overline{\overline{A}_0.B_0 \vee A_0.\overline{B}_0} = \overline{e_7 \vee e_8} = e_{12}$$

The reason for this choice is that the products $\overline{A}_i.B_i$ and $A_i.\overline{B}_i$ are already necessary for the realization of the functions *Y* and *Z*, and the use of expression (10) would "save" 8 AND logic gates (2 integrated circuits), but requires the use of NOR logic gates instead of OR logic gates.

It follows that the output functions of the comparator can be written in the form:

$$X = x_3.x_2.x_1.x_0 = e_9.e_{10}.e_{11}.e_{12} \quad (11)$$

$$Y = A_3.\overline{B}_3 \vee x_3.A_2.\overline{B}_2 \vee x_3.x_2.A_1.\overline{B}_1 \vee x_3.x_2.x_1.A_0.\overline{B}_0 =$$
$$= e_2 \vee e_9.e_4 \vee e_9.e_{10}.e_6 \vee e_9.e_{10}.e_{11}.e_8 = e_2 \vee e_{14} \vee e_{16} \vee e_{18} \quad (12)$$

$$Z = \bar{A}_3.B_3 \vee x_3.\bar{A}_2.B_2 \vee x_3.x_2.\bar{A}_1.B_1 \vee x_3.x_2.x_1.\bar{A}_0.B_0 =$$
$$= e_1 \vee e_9.e_3 \vee e_9.e_{10}.e_5 \vee e_9.e_{10}.e_{11}.e_7 = e_1 \vee e_{13} \vee e_{15} \vee e_{17} \quad (13)$$

The circuit of the magnitude comparator (Fig. 4) is built in Logisim, an educational tool for designing and simulating digital logic circuits (http://www.cburch.com/logisim/), and presented in Fig. 5. Logisim allows to test the circuit for different input combinations and to see the values of all the intermediate variables and the output variables, equations (7)…(13).

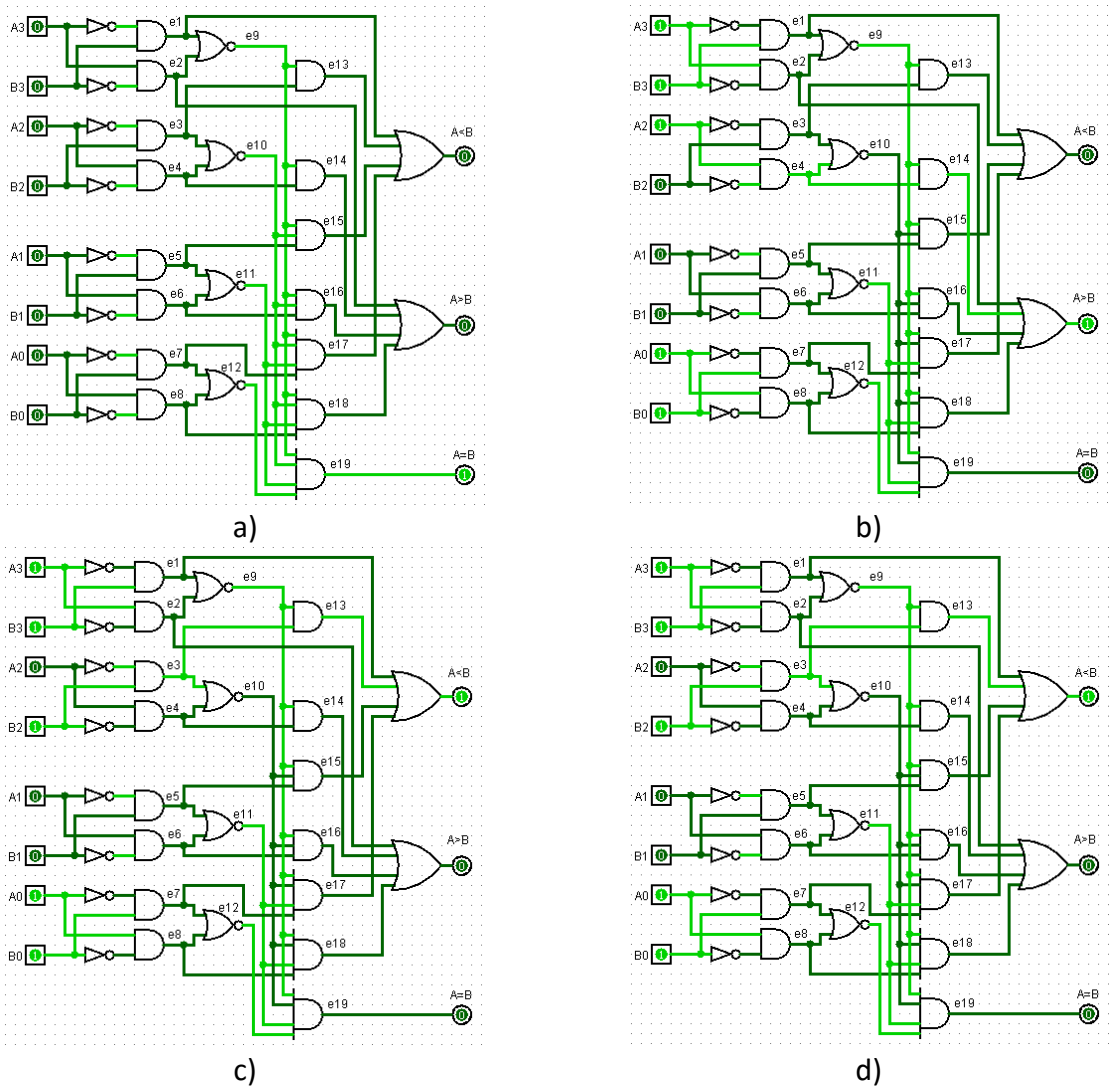


Fig. 5. Implementation of a four-bit magnitude comparator in Logisim:
a) initial state: **0000** = **0000** (0 = 0, equal); b) **1101** > **1001** (13 > 9, 1 different bit);
c) **1001** < **1101** (9 < 13, 1 different bit); d) **1001** < **1010** (9 < 10, 2 different bits)

The circuit of the magnitude comparator (Fig. 4) is built in Logisim using integrated circuits from the library 74xx (http://www.cburch.com/logisim/download/7400-lib.circ) of Texas Instruments (studied in the course) and presented in Fig. 6. The integrated circuits are numbered from 1 to 13 (IC1…IC13), necessary for the next considerations. The symbols of the integrated circuits are above the circuit of the magnitude comparator in order to avoid the congestion of the circuit and the impossibility to read the labels.

The circuit is tested for different input combinations allowing to see the values of all the intermediate variables and the output variables, equations (7)…(13). Some of the results are presented in Fig. 7.

The same input combinations were used to be able to compare the results of Fig. 7 and Fig. 5. It can be seen that the intermediate and output variables coincide for each of the considered cases.

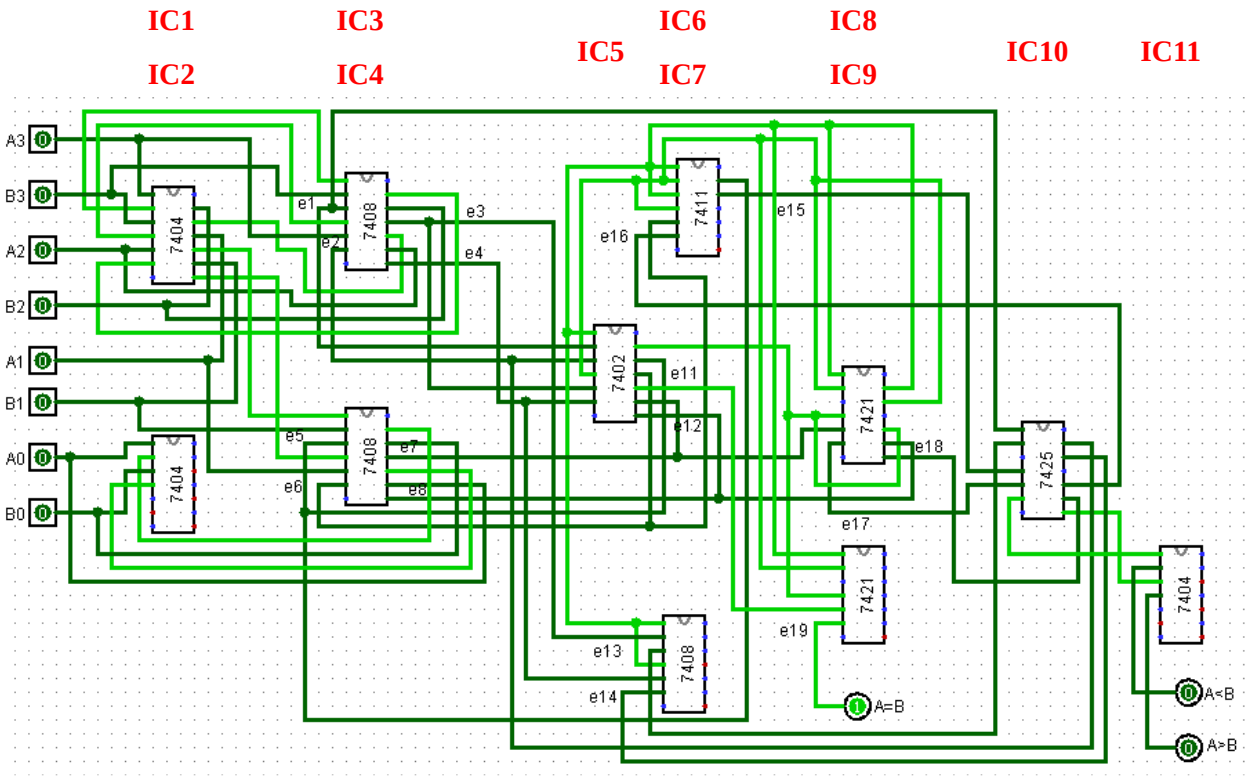


Fig. 6. Implementation of a four-bit magnitude comparator in Logisim with integrated circuits from the library 74xx (Texas Instruments)

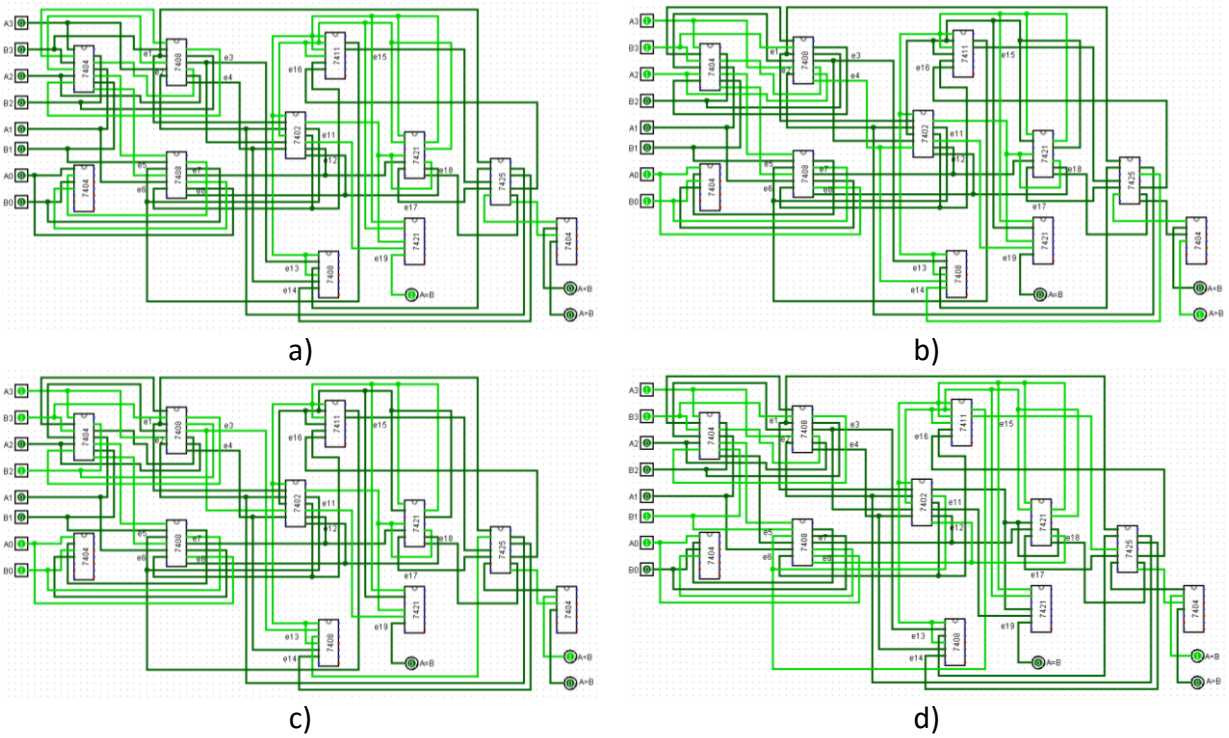


Fig. 7. Implementation of a four-bit magnitude comparator in Logisim with integrated circuits from the library 74xx (Texas Instruments):

a) initial state: **0000** = **0000** (0 = 0, equal); b) **1101** > **1001** (13 > 9, 1 different bit);

c) **1001** < **1101** (9 < 13, 1 different bit); d) **1001** < **1010** (9 < 10, 2 different bits)

For the implementation of the circuit of the magnitude comparator with integrated circuits from the library 74xx the following integrated circuits are used: 7404 (6 inverters) – 3 pieces (IC1, IC2, IC11); 7408 (4 two-input AND logic gates) – 3 pieces (IC3, IC4, IC7), 7402 (4 two-input NOR logic gates) – 1 piece (IC5), 7411 (4 three-input AND logic gates) – 1 piece (IC6), 7421 (2 four-input AND logic gates) – 2 pieces (IC8, IC9), 7425 (2 four-input NOR logic gates) – 1 piece (IC10).

The use of IC10 and IC11 was necessitated by the lack of implementation of an integrated circuit containing four-input OR logic gates, i.e. combining of NOR and NOT logic gates.

In the diagram of Fig. 6 the designations of the intermediate and output variables are written, which facilitates its tracking. The purpose in constructing the circuit in Fig. 6 is to observe the stepwise arrangement of the ICs, which follows the logic of the scheme of Fig. 4.

Of course, it can also be optimized by replacing the two inverters from IC11 with "free" inverters from IC2, thus eliminating IC11. In addition, the unused four-input AND logic gates from IC9 can be used as a two-input AND logic gate according to the scheme in Fig. 8a, and the unused three-input AND logic gate from IC6 can be used as two-input AND logic gate according to the scheme in Fig. 8b. This allows better use of IC2, IC6 and IC9 and the elimination of two integrated circuits – IC11 and IC7 (Fig. 8).

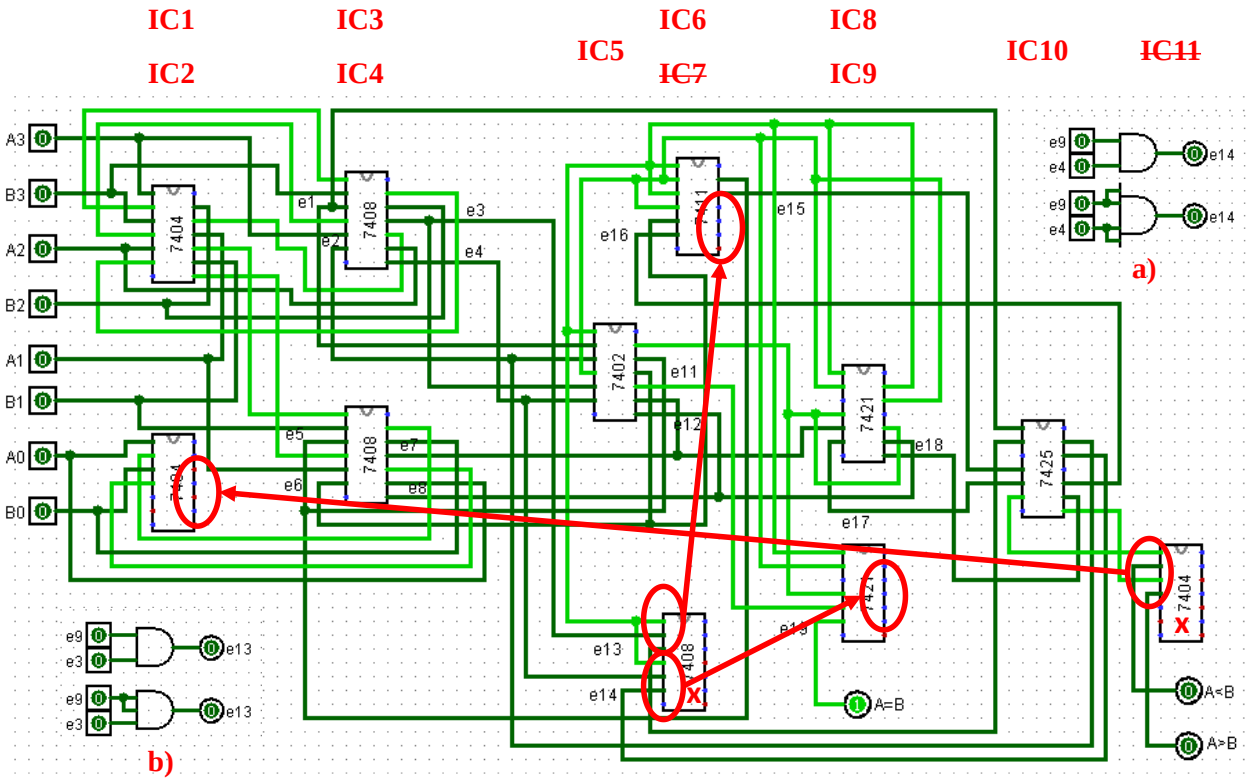


Fig. 8. Implementation of a four-bit magnitude comparator in Logisim with integrated circuits from the library 74xx (Texas Instruments), a way for optimizing the circuit

Next, the circuit of the magnitude comparator (Fig. 4) is built in the environment of ISE Project Navigator (https://www.xilinx.com/content/xilinx/en/downloadNav/design-tools/v2012_4---14_7.html) and presented in Fig. 9. The bit file, necessary for programming the laboratory board, is generated. After programming, the comparator is tested for different input combinations allowing to see the values of some intermediate variables and the output variables. Some of the results are presented in Fig. 10 and Table 2. The correspondence between the input, intermediate and output variables of the magnitude comparator and the input and output signals of the laboratory board are shown in the first two rows in Table 2.

The same input combinations were used with the aim of comparing the results of Fig. 10 (Table 2), Fig. 7 and Fig. 5. It can be seen that the intermediate and output variables coincide for each of the considered cases.

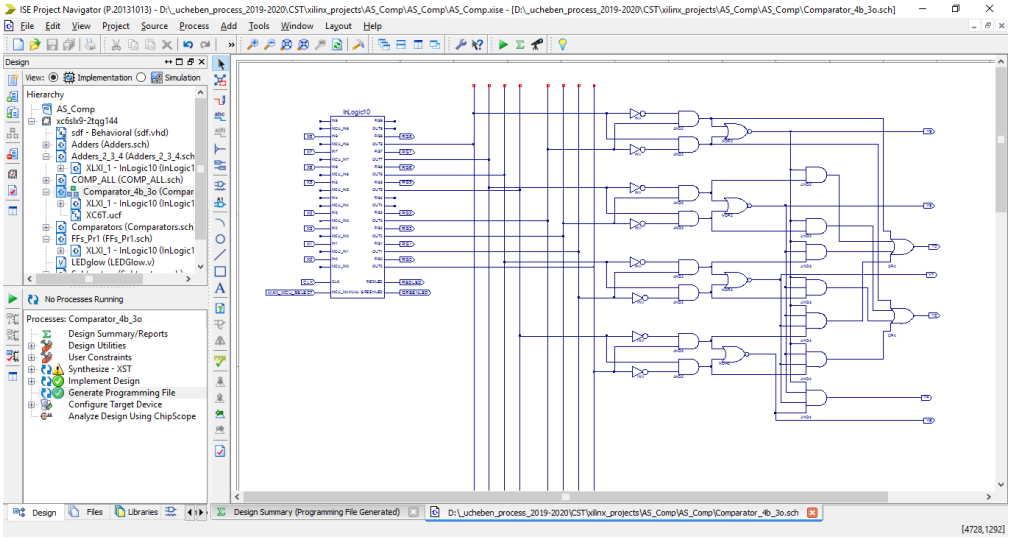
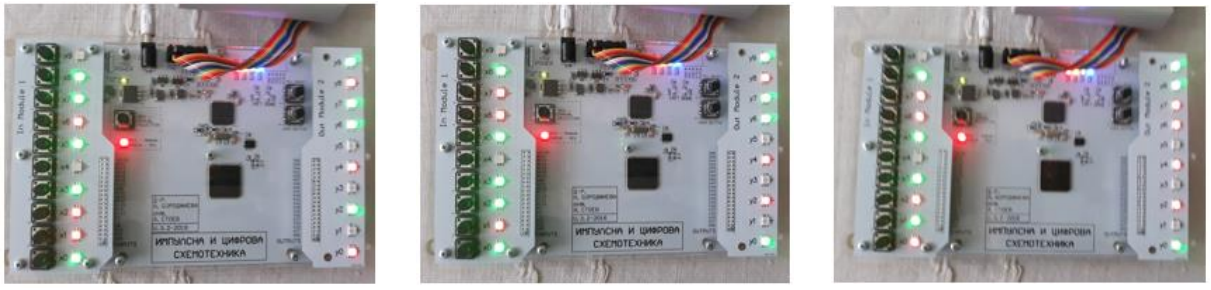


Fig. 9. FPGA implementation of the magnitude comparator, comparing two 4-bit binary numbers, based on their schematic presentation in ISE Project Navigator

Table 2. Some examples when testing the magnitude comparator on the board

A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	C_3	C_2	C_1	C_0	X	Y	Z
x8	x7	x6	x5	x3	x2	x1	x0	y9	y8	y7	y6	y4	y2	y0
1	1	0	1	1	1	0	1	1	1	1	1	1	0	0
1	1	0	1	1	0	0	1	1	0	1	1	0	1	0
1	0	0	1	1	1	0	1	1	0	1	1	0	0	1
1	0	0	1	1	0	1	0	1	1	0	0	0	0	1
1	0	1	0	1	0	0	1	1	1	0	0	0	1	0

Note: C_i - the output of the *i*-th NOR logic gate comparing the bits A_i and B_i



a) b) c)

Fig. 10. Testing the magnitude comparator on the laboratory board

a) **1101** > **1001** (13 > 9, 1 different bit); b) **1001** < **1101** (9 < 13, 1 different bit);
c) **1001** < **1010** (9 < 10, 2 different bits)

At the end, an MS Excel-based application (https://www.microsoft.com/bg-bg/microsoft-365/excel) is developed illustrating the principle of operation of the magnitude comparator comparing two 4-bit binary numbers. The application is based on the equations (7)…(13) using the tools for: 1) validating the data (Fig. 11, block 1); 2) inserting formulas (Fig. 11, block 2); 3) conditional formatting (Fig. 11, block 3). The MS Excel-based application allows the user to select from the bits of the two operands *A* and *B*, using drop-down menus to select one of the two

possible values 0 or 1 (Fig. 11, block 1). The output functions of each stage (intermediate and output variables) are implemented by using built-in functions in MS Excel that allow the use of Boolean functions, such as OR, AND, and XOR, and IF function in order to correctly visualize the result 0s or 1s (instead of False and True) (Fig. 11, block 2). Finally, conditional formatting is used in order to display 0s and 1s with different font and background (Fig. 11, block 3) for better information presentation. The formulas defined in the cells of MS Excel are based on the formulas (7)…(13) (Fig. 11, block 4).

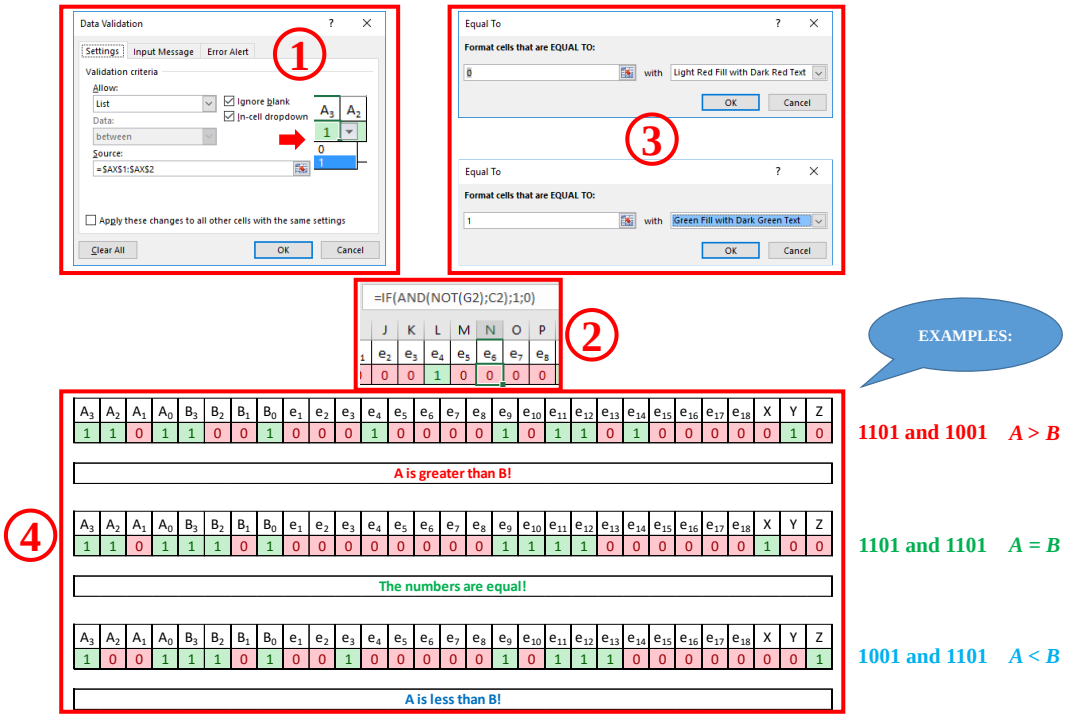


Fig. 11. MS Excel-based application illustrating the principle of operation of the magnitude comparator comparing two 4-bit binary numbers

Some results using the application are presented in Fig. 11, block 4. The results of Fig. 11, Fig. 10 (Table 2), Fig. 7 and Fig. 5 might be compared. It can be seen that the intermediate and output variables coincide for each of the considered cases.

Magnitude comparators in IC form

Magnitude comparators are available in IC form. For example, 7485 is a four-bit magnitude comparator of the TTL logic family. IC 4585 is a similar device in the CMOS family. 7485 and 4585 have the same pin connection diagram and functional table. The logic circuit inside these devices determines whether one four-bit number, binary or BCD, is *less than*, *equal to* or *greater than* the second four-bit number. It can perform comparison of straight binary and straight BCD (8-4-2-1) codes. These devices can be cascaded together to perform operations on larger bit numbers without the help of any external gates. This is facilitated by three additional inputs called cascading or expansion inputs available on the IC. These cascading inputs are also designated as $A = B$, $A > B$ and $A < B$ inputs. Cascading of individual magnitude comparators of the type 7485 or 4585 is discussed in the following paragraphs. IC 74AS885 is another common magnitude comparator. The device is an eight-bit magnitude comparator belonging to the advanced Schottky TTL family. It can perform high-speed arithmetic or logic comparisons on two eight-bit binary or 2's complement numbers and produces two fully decoded decisions at the output about one number being either greater than or less than the other. More than one of these devices can also be connected in a cascade arrangement to perform comparison of numbers of longer lengths (Maini, A. K., 2007).

CONCLUSION

The paper presents an approach for designing a magnitude comparator comparing two four-bit binary numbers using computer-based tools. The magnitude comparator is implemented in Logisim using integrated circuits (IC) from the library 74xx and in ISE Project Navigator for programming the FPGA-based laboratory board, developed at the University of Ruse. MS Excel-based application for illustrating the principle of operation of the magnitude comparator is also developed. The applications will be used in the educational process in the courses "Pulse and Digital Devices", "Pulse and Digital Circuits", and "Digital Circuits" in the University of Ruse.

Using exercises with the laboratory board excites students' minds and captures their interest. Incorporating audio-visual materials to addition to the textbooks during the practical classes, for example laboratory boards, movies, pictures, helps the students understand the concepts better. Demonstrating through real-life situations and brainstorming sessions during the classes, where students must determine what devices are implemented on the laboratory board, looking at their truth tables, numerous ideas are obtained because multiple brains are focused on one single idea, and also everyone is involved into the discussion. A classroom environment, decorated with these laboratory boards, helps stimulate students' minds and helps think and learn better. Working together as a team during the classes also encourages the students to learn and think better, and of course, the end result of the collaborative effort is always immense. Therefore, active learning engages students in learning, using activities such as reading, writing, discussion, or problem solving, and promotes analysis, synthesis, and evaluation of the class content. Active learning during the classes also provides students with informal opportunities for feedback on how well they understood the material.

ACKNOWLEDGMENTS

This paper is supported by the National Scientific Program "Information and Communication Technologies for a Single Digital Market in Science, Education and Security (ICTinSES)", financed by the Ministry of Education and Science of Bulgaria. The work presented in this paper is completed as partial fulfilment of Project 2020 – FEEA – 03 "Design and development of a multifunctional robot for implementation and evaluation of autonomous navigation algorithms" financed under the Scientific and Research Fund of the University of Ruse "Angel Kanchev".

REFERENCES

Kumar, A. (2011). Fundamentals of Digital Circuits. https://easyengineering.net/fundamentals-of-digital-circuits-by-anand-kumar/

Maini, A. K. (2007). Digital Electronics: Principles, Devices and Applications, DOI 10.1002/9780470510520, researchgate.net/publication/290729475_Digital_Electronics_Principles_Devices_and_Applications/citations

Floyd, Th. (2006). Digital Fundamentals. Ninth Edition, Pearson Prentice Hall, Pearson Education, Inc., Upper Saddle River, New Jersey.

Floyd, Th. (2006). Fundamentos de sistemas digitales. Novena Edición, Pearson Prentice Hall, Pearson Education, Inc., Upper Saddle River, New Jersey (printed in Madrid, Spain, in Spanish), https://www.academia.edu/22270123/Fundamentos-de-sistemas-digitales-floyd-9ed.

Borodzhieva, A., Stoev, I., Mutkov, V. (2019). Application of Active Learning Methods in the Course "Digital Electronics" in the Topic Digital Comparators Using FPGA Design. 2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging, SIITME 2019, 23 – 26 October 2019, Cluj-Napoca, Romania, IEEE Xplore Compliant Proceedings, pp. 160-163, ISBN: 978-1-5386-5577-1, doi: 10.1109/SIITME47687.2019.8990786.

Logisim documentation, http://www.cburch.com/logisim/

7400 series Logisim library, Ben Oztalay, http://www.cburch.com/logisim/download/7400-lib.circ

Xilinx documentation, https://www.xilinx.com/content/xilinx/en/downloadNav/design-tools/v2012_4---14_7.html

MS Excel, https://www.microsoft.com/bg-bg/microsoft-365/excel (in Bulgarian)