jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2018, VOL. 64, NO. 3, PP. 373–378
Manuscript received May 9, 2017; revised July, 2018. DOI: 10.24425/123535

On Transformation of a Logical Circuit to a Circuit with NAND and NOR Gates Only

Samary Baranov and Andrei Karatkevich

***Abstract*—In the paper we consider fast transformation of a multilevel and multioutput circuit with AND, OR and NOT gates into a functionally equivalent circuit with NAND and NOR gates. The task can be solved by replacing AND and OR gates by NAND or NOR gates, which requires in some cases introducing the additional inverters or splitting the gates. In the paper the quick approximation algorithms of the circuit transformation are proposed, minimizing number of the inverters. The presented algorithms allow transformation of any multilevel circuit into a circuit being a combination of NOR gates, NAND gates or both types of universal gates.**

***Keywords*—logic synthesis, logic devices, VLSI, minimization.**

I. Introduction

THE gates which are most popular in the logic synthesis are NOR and NAND gates. It follows from two facts. First one is the functional completeness - each Boolean function can be implemented by using a combination of NOR gates or NAND gates. The second fact is that those gates require few transistors (e.g., in NMOS logic a NAND gate is simpler than an AND or OR gate) [1]-[3].

However, people naturally use to think in the basis AND-OR-NOT, not in the basis NOR-NAND. Besides, almost all known methods for minimization of logic circuits, from Karnaugh maps to the algorithms used in the Espresso logic minimizer, produce results in the same AND-OR-NOT basis [4], [5]. Only after such minimization the special mapping algorithms are used to cover the circuit by the librarian elements, NOR and NAND gates as well. In case of two-level minimization in the form of sum of products or product of sums, such mapping is trivial. But in the FPGA circuits, consisting of the logic blocks, a multilevel implementation of the Boolean functions is typical [6]-[9]. Optimization of transformation of a multi-level circuit from AND-OR-NOT basis to NOR-NAND basis is not an easy task.

In this paper we discuss a rather simple method for transformation of any multilevel and multioutput circuit consisting of AND, OR and NOT gates into the circuit consisting of NOR and NAND gates without using of Boolean expressions. We show that the task of construction of an optimal circuit is reduced to the task of coloring of the circuit graph into two colors with minimization of violations in such a coloring.

S. Baranov is with Holon Institute of Technology, Israel (e-mail: baranov@hit.ac.il).

A. Karatkevich is with Institute of Electrical Engineering, University of Zielona Gora, Zielona Gora, Poland (e-mail: A.Karatkevich@iee.uz.zgora.pl).

TABLE I
Truth Table for Function NOR

$x1$	$x2$	$x1+x2$	$(x1+x2)'$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

II. Circuits with NOR gates

Table I presents the truth table for function NOR. This function produces a value 1 only when both arguments are equal to 0 (the first row), otherwise it is equal to 0.

Implementation of functions OR and AND with NOR gates is evident from the simple logical transformations. Thus, to realize OR-function $f = x1 + x2$ with NOR gates we must use the same inputs x1, x2 as the inputs for NOR gate and invert its output (which can be done by providing its output value to the inputs of another NOR gate). Implementation of AND-function $f = x1x2$ with NOR gates follows from the De Morgan's law: $x1x2 = (x1' + x2')'$, so the inputs for the NOR gate have to be inverted.

As the first example, we will discuss mapping of a logic circuit shown in Fig. 1 with NOR gates. Here a gate by gate transformation is used. Thus, every gate OR in this circuit is replaced by a gate NOR and an inverter at its output, and every AND gate is replaced by a NOR gate and the inverters at its inputs. Fig. 2 demonstrates the result of such transformation. In the circuit thus constructed, two sequential inverters may be found (such cases are dotted in Fig. 2). The final step consists of deleting of such pairs of inverters (Fig. 3).

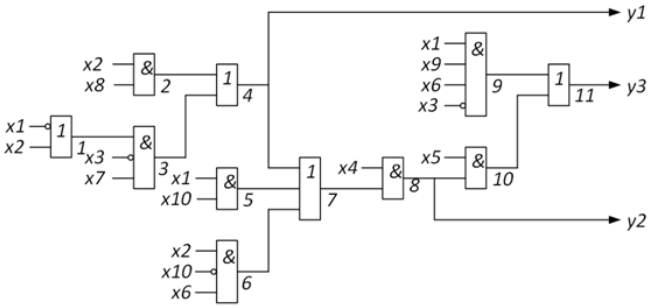

Fig. 1. Example 1 with AND and OR gates.

TABLE II
TRUTH TABLE FOR FUNCTION NAND

$x1$	$x2$	$x1+x2$	$(x1+x2)'$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

III. CIRCUITS WITH NAND GATES

Table II presents the truth table for function NAND. This function produces 0 only when both arguments are equal to 1 (the last row), otherwise its value is 1.

Implementation of the functions AND and OR with the NAND gates is analogous to the implementation with the NOR gates. Thus, to realize the function AND $f = x1x2$ with a NAND gate we must use the same inputs $x1$, $x2$ and invert the output. To realize the runction OR $f = x1 + x2$, we have to use the inverted inputs $(x1 + x2 = (x1'x2')'$, according to De Morgan law).

As an example, let us consider the mapping of the same logic circuit (Fig. 1) with NAND gates. Again, here we use a gate by gate transformation. Thus, the gates OR in this circuit are replaced by gates NAND with inverted inputs. Every gate AND in Fig. 1 is replaced by a gate NAND and an inverter at its output. The result of the transformation is shown in Fig.4 As above, in the circuit thus constructed, the couples of two sequential inverters may be found (such cases are dotted in Fig. 4). At the final step such pairs of inverters are deleted (Fig. 5).

IV. CIRCUITS WITH NOR AND NAND GATES

In both previous cases each AND gate and OR gate is replaced by a NOR gate or by a NAND gate with the inverters (which can be considered as the NAND or NOR gates with a single input) added when necessary. Such transformation

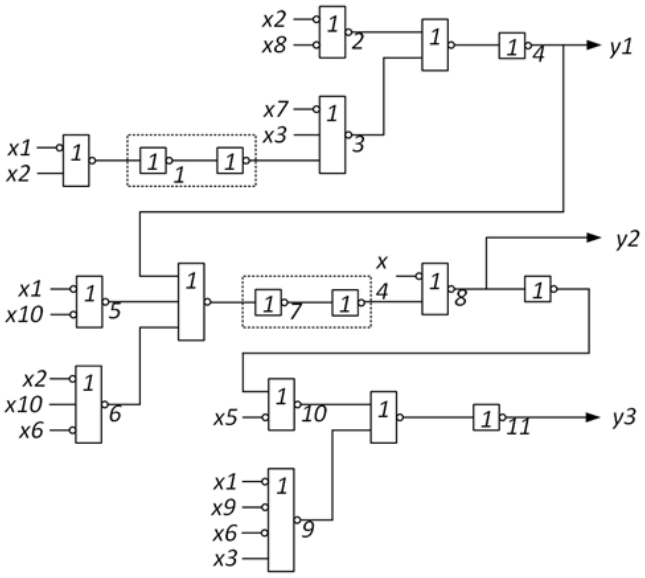

Fig. 2. Gate by gate mapping of the circuit in Fig. 1 with NOR gates.

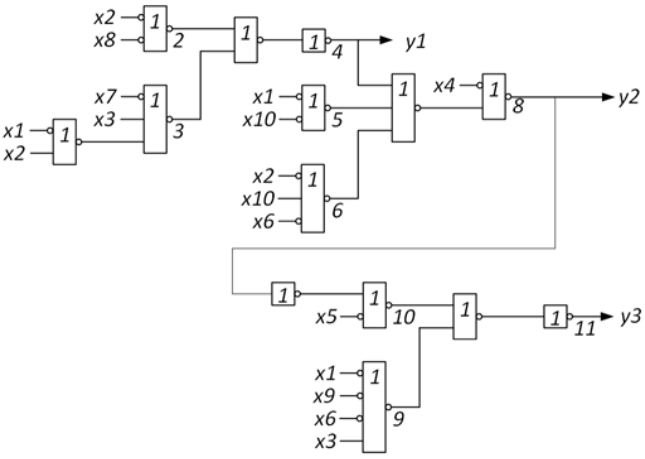

Fig. 3. Final step of the mapping with NOR gates.

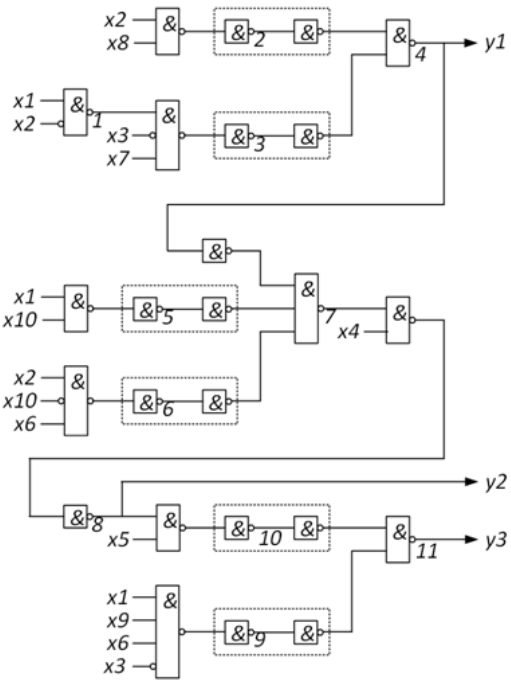

Fig. 4. Gate by gate mapping of the circuit in Fig. 1 with NAND gates.

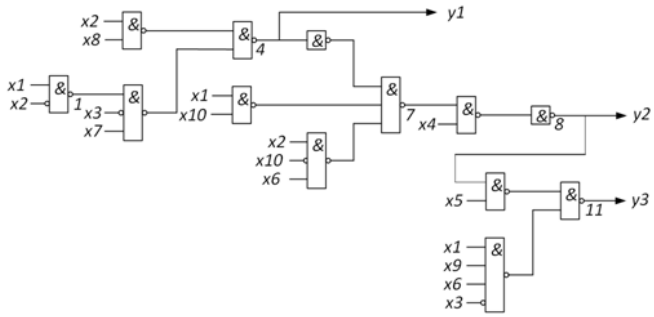

Fig. 5. Final step of the mapping with NAND gates.

is deterministic, and there is no room for optimization. The situation is different if we can use both NAND and NOR gates. Then, depending of which kind of gate replaces each AND and OR gate, number of the additional inverters can differ.

Suppose that a number 0 or 1 is used to mark every AND and OR gate, and the following simple rules are used to decide what gate (NOR or NAND) should cover the gate:

1) If an AND gate is marked by 1, then it is realized by a NAND gate;
2) If an OR gate is marked by 1, then it is realized by a NOR gate;
3) If an AND gate is marked by 0, then it is realized by a NOR gate;
4) If an OR gate is marked by 0, then it is realized by a NAND gate.

In Fig. 6a four copies of the same two-level circuit directly implementing function $f_1 = x1x2 + x3$ are shown. In these circuits, we numbered the gates by all possible combinations of zeroes and ones. Then, in Fig. 6b we have implemented these circuits with NOR and NAND gates according the rules presented above.

It can be seen in Fig. 6 that:

1) If two gates in the sequence are marked by different numbers (0-1 in the second circuit and 1-0 in the third circuit), then there is no inverter between the NOR and NAND gates;
2) If two gates in the sequence are marked by the same numbers (0-0 in the first circuit and 1-1 in the last one), then there is an inverter between the NOR and NAND gates.

It is easy to see that the above is true not only for the combination AND-OR, but for three other possible combinations of AND and OR gates, too. Hence, minimization of the number of cases in which two connected gates are marked by the same number leads to minimization of number of the additional gates. Besides, if an output of the whole circuit is taken from a gate marked by 1, then an additional inverter at the output is needed. Analogously, an input of the whole circuit has to be inverted, if it is connected to an input of a gate marked by 0. However, we may suppose that for a circuit which is complicated enough the number of the internal connections is much greater than the number of the external ones, so minimization of the number of additional inverters inside the circuit is more important.

Consider a graph with vertices corresponding to the AND and OR gates. Let two vertices be connected by an edge if and only if output of one of the corresponging gates is connected to the input of another. Then, the task of minimization of the number of additional gates is reduced to the task of coloring of the graph by two colors (0 and 1) with minimization of the number of failures in such coloring - i.e. with minimization of the number of cases in which two connected vertices are colored by the same color.

Following from the above, the rules for transformation of any logic circuits with AND-OR gates into the circuit with NAND-NOR gates can be briefly formulated in the following way:

Step1. *Marking*. At this step, we mark each OR and AND gate of the OR-AND-NOT circuit with 1 or 0, minimizing the number of cases in which two connected gates are marked with the same number (such minimization will be discussed in more detail in the next section).

Step2. *Mapping*. At this step, each gate marked by 1, should be replaced with a "consonant" gate (OR by NOR, AND by NAND). Each gate marked by 0, should be replaced with a non-consonant gate (OR by NAND, AND by NOR). In such mapping, inverters appear only between gates marked by the same numbers (0-0 or 1-1).

One of the possible markings for the circuit shown in Fig. 1 is presented in Fig. 7. The circuit in Fig. 13 consisting of NOR and NAND gates is the result of mapping of the circuit consisting of AND and OR gates according to Fig. 7. Note that the number of gates of the circuit in Fig. 13 is less than the numbers of gates of the circuits shown in Figs 3 and 5 (and the same as of the circuit shown Fig. 1).

Since there are no violations in the marking of the circuit shown in Fig. 7, there are no inverters between gates in Fig. 13. Evidently, it is not always the case. In the circuit shown in Fig. 9 (Example 2), a violation in marking cannot be avoided (in this example two connected gates AND4 and AND7 are marked by 0), and an inverter between the corresponding gates has to be inserted in the transformed circuit shown in Fig. 10.

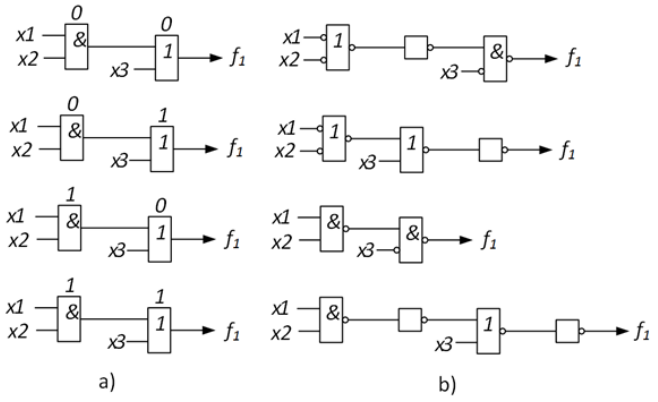

Fig. 6. Four implementations of the same circuit with NOR NAND gates.

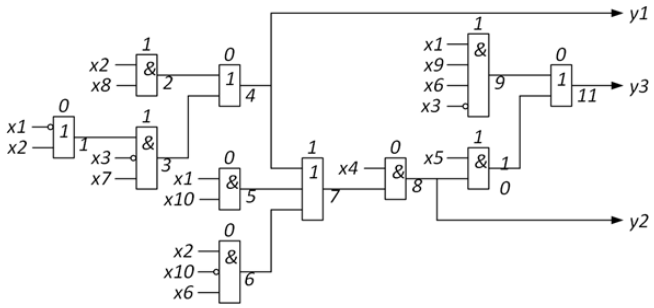

Fig. 7. Circuit from Fig. 1 with a marking.

Summarizing, the process of transformation of an AND-OR-NOT circuit into a NOR-NAND circuit can be described in short as follows:

1) Mark the AND and OR gates with 1 and 0, minimizing the violations as described above;
2) If a gate is marked by 0, replace AND by NOR or OR by NAND;
3) If a gate is marked by 1, replace AND by NAND or OR by NOR;
4) If an output of the circuit is connected to an output of a gate marked with 1, invert the output;
5) If an input of the circuit is connected to an input of a gate marked with 0, invert the input;
6) If an output of a gate marked with 1 is connected to an input of an inverter, remove the inverter;
7) If an input of a gate marked with 0 is connected to an output of an inverter, remove the inverter;
8) Put an inverter between two connected gates if they have the same marks;
9) Replace every inverter by a NAND or NOR gate with connected inputs.

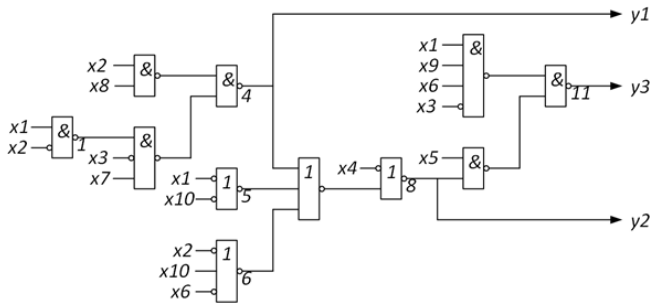

Fig. 8. Example 1 with NOR and NAND gates.

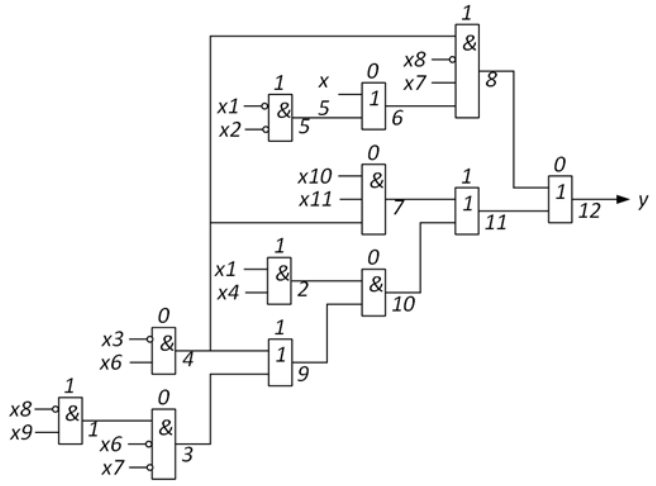

Fig. 9. Example 2 with AND and OR gates (marked).

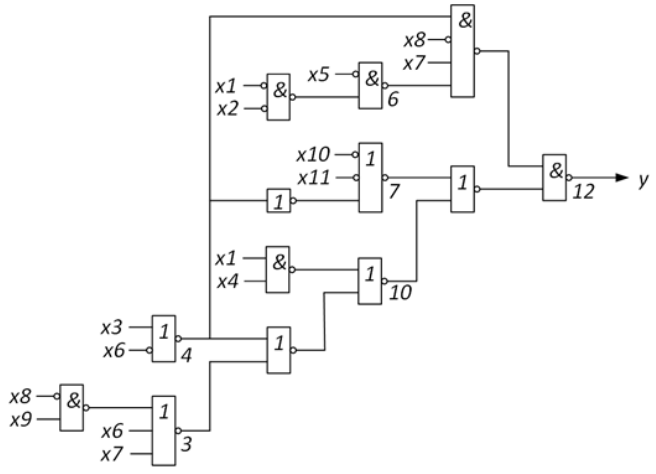

Fig. 10. Example 2 with NOR and NAND gates.

V. Optimized marking

As it was mentioned in the previous section, the task of optimal marking of a circuit has something in common with the task of graph coloring. Chromatic number of a graph is equal to 2 if and only if the graph is bipartite [10]. Hence, the considered task is reduced to the task of removing the minimal number of edges of a graph necessary to make it a bipartite graph, which is known as the *edge bipartization problem* [11]. The decision version of the problem is NP-hard; there are some exact (exponential time) and approximation (polynomial time) algorithms of edge bipartization [11]-[15]. The known approximation algorithms of edge bipartization use the linear programming methods [12].

We propose a quick approximation method of edge bipartization using a spanning tree construction. A *fundamental cycle basis* of an undirected graph is a minimal set of cycles that allows to obtain any cycle of the graph by applying the operation of symmetric difference. A fundamental cycle basis can be obtained from a spanning tree of the graph, where every edge belonging to the graph and not belonging to the spanning tree creates together with the edges of the tree a cycle belonging to the cycle basis [10]. A spanning tree can be found in linear time using depth-first search or breadth-first search [16], so the whole process of detecting the fundamental cycle basis is polynomial.

Now, it is easy to see that if all the cycles in the fundamental cycle basis have an even length, then the graph is bipartite (symmetric difference of two cycles of an even length results in a cycle of an even length). If there are the cycles with odd length, let us break them by removing some edges. To minimize the number of removed edges, it ts reasonable to detect a minimized hitting set of the collection of sets of edges of the odd cycles (and to remove the edges belonging to the hitting set). Such a set can be obtained by a polynomial time approximation algorithm of set covering [16].

Unfortunately, the graph obtained after this operation still may be not bipartite. Then, it has to be repeated until the graph is bipartite (at each iteration at least one edge is removed, hence the loop terminates). When a bipartite graph is obrained, it should be colored with 2 colors, which can be done by DFS or BFS [17].

The proposed marking method can be briefly described as follows.

1) Create for given circuit a graph G such that its vertices correspond to AND and OR gates; two vertices are connected by an edge if and only if output of one of the corresponging gates is connected to the input of the other one;
2) Construct a spanning tree of G by means of DFS or BFS;
3) Obtain a fundamental cycle basis from the spanning tree;
4) If every cycle in the obtained basis has even length, go to item 6;
5) For the sets of edges of all the odd length cycles in the basis, find a hitting set using a greedy algorithm;
6) Remove from G all the edges belonging to the obtained hitting set and go to item 2;
7) Color G with 2 colors by means of DFS or BFS;
8) Mark every AND and OR gate of the circuit with 0 or 1, according to the obtained coloring.

Consider the example shown in Fig. 11. The corresponding graph is shown in Fig. 12 (numbers of the nodes correspond to the numbers of the gates in Fig. 11). The solid lines depict the edges belonging to a spanning tree (Fig. 12a). The cycles created by other edges together with the spanning tree are as follws: 1-2-3, 2-3-4, 2-3-5-4 and 2-3-5-6-4. Three of them are of odd length (1-2-3, 2-3-4, 2-3-5-6-4). They have in common the edge 2-3, so removing this edge breaks all of them. However, the obtained graph is still not bipartite (Fig. 12b). A spanning tree for this graph is shown by the solid lines in Fig. 12b. The fundamental cycle basis obtained from the tree contains of the cycles 1-2-4-5-3, 3-4-5 and 4-5-6. All these cycles have odd length, and all of them have common edge 4-5. Removing of this edge leads to a bipartite graph shown in Fig. 12c - here the removed edges are marked by the dotted lines, and the white and black nodes show the results of the coloring.

The final result of the transformation, according to the proposed method, is shown in Fig. 13.

VI. One more possibility of optimization: Splitting of the gates

There is one more possibility of minimization of the number of gates when transforming logic circuit with AND-OR-NOT to a circuit in the NAND-NOR basis. Consider the circuit shown in Fig. 14a. Suppose that it is a part of a bigger circuit, and the gates have to be marked as it is shown, i.e. gates 1, 4, and 5 are marked with 0. The transformation as described in Section IV leads to the circuit shown in 14b, with two additional inverters introduced between the gates marked with the same marking - between gate 1 and gate 5 and between gate 4 and gate 5.

Now, let us split the gate OR into two OR gates, as shown in Fig. 14c. After that we do not have the same marking for two sequential gates, as far as the new gate between gates 1 and 5 can be marked with 1. As a result, the NAND-NOR circuit in Fig. 14d does not contain the additional inverters between

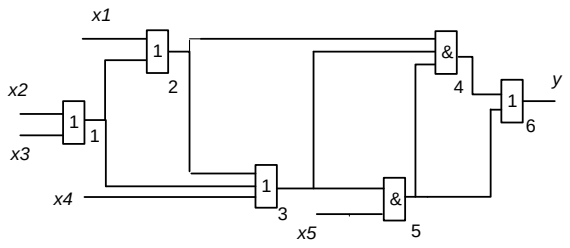

Fig. 11. Example 3 with OR and AND gates.

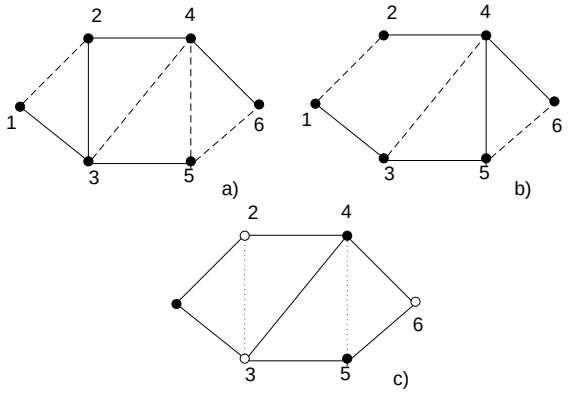

Fig. 12. Graph corresponding to Example 3.

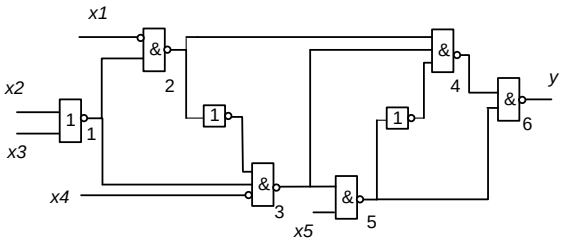

Fig. 13. Example 3 with NOR and NAND gates.

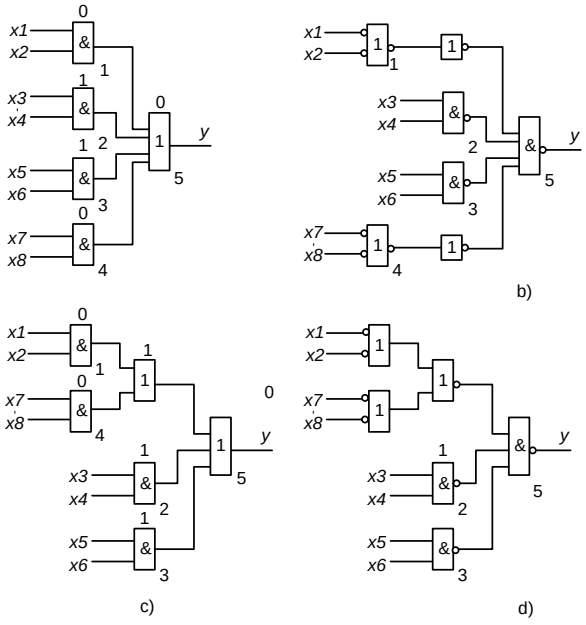

Fig. 14. Splitting of a gate in NOR-NAND circuit.

the gates. The circuit shown in Fig. 14d has less gates than the circuit shown in Fig. 14a.

The general rule allowing to use the presented possibility can be formulated as follows: If the outputs of more than one gate N marked with 1 (0) are connected to the inputs of a gate n also marked with 1 (0), then split the gate n into two gates n_1, n_2 (implementing the same logical function as n) in such a way that the outputs of all the gates belonging to N are connected to the inputs of gate n_1, the output of n_1 is connected to an input of n_2, and the outputs of all the gates marked with 0 (1), which were connected to the inputs of n, are connected to the inputs of n_2.

Applying of such operation makes sense between the steps of *marking* and *mapping* of the presented transformation. Generally, it allows to use a single gate instead of several inverters.

VII. Conclusion

In the paper a method allowing to transform a combinational logical circuit from AND-OR-NOT basis to NAND-NOR basis is presented. The problem statement is caused by the fact that the NAND and NOR gates are universal, simple and because of that they are very popular in digital design. Each of the functions NAND and NOR is functionally complete, and in the paper the methods for quick transformation of a circuit into the circuius in NAND basis or NOR basis are described. However, using both of those types of gates allows obtaining more compact circuits than using only one, and using NAND-NOR basis is an original feature of the third of the proposed methods.

Using both types of the gates makes room for minimization of the number of the gates in the obtained circuits. We show that such minimization requires an edge bipartization of a graph describing the structure of the circuit. A new approximation method solving the edge bipartization problem is proposed.

Future research is going to concentrate on deeper investigation of minimization of the obtained circuits. The additional method of gate minimization described in Section VI requires studying of the cases in which output of a gate is connected to the inputs of several gates marked with the same number and some other possible circuit structures.

References

[1] D. A. Pinelli, *Fundamentals of Digital Logic Design With VLSI Circuit Applications*, Prentice Hall, 1990.
[2] M. M. Mano and Ch. R. Kime, *Logic and Computer Design Fundamentals*, 3rd ed. Prentice Hall, 2004.
[3] S. Baranov, *Logic and System Design of Digital Systems*, Tallinn: TUT Press, 2008.
[4] G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994.
[5] A. Zakrevskij, Yu. Pottosin and L. Cheremisinova, *Optimization in Boolean Space*, Tallinn: TUT Press, 2009.
[6] T. Luba, *Synteza układów logicznych*, Warszawa: WSISiZ, 2000.
[7] M. Rawski, T. Luba, Z. Jachna, and P. Tomaszewicz, "The influence of functional decomposition on modern digital design process." in *Design of Embedded Control Systems*. Springer-Verlag, 2005, pp. 193-204.
[8] P. N. Bibilo, *Decomposition of Boolean Functions by Means of Solving Logical Equations*, Minsk: Belaruskaya Navuka, 2009.
[9] G. Borowik, G. Labiak, and A. Bukowiec, "FSM-based logic controller synthesis in programmable devices with embedded memory blocks." in *Innovative technologies in management and science*. Cham Heidelberg : Springer International Publishing Switzerland, 2015 - (Topics in Intelligent Engineering and Informatics ; Vol. 10) - s. 123–151.
[10] R. Diestel, *Graph Theory*, Springer-Verlag, 2000.
[11] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke, "Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization", *Journal of Computer and System Sciences*, vol. 72, 2006.
[12] A. Avidor and M. Landberg, "The multi-multiway cut problem", *Proceedings of 9th SWAT*, LNCS, vol. 3111, Springer-Verlag, 2004, pp. 273284.
[13] F. T. Nobibon, C. A. J. Hurkens, R. Leus, and F. C. R. Spieksma, "Coloring Graphs Using Two Colors while Avoiding Monochromatic Cycles", *Informs Journal on Computing*, 6124(3), January 2010, pp. 229-242.
[14] F. T. Nobibon, C. A. J. Hurkens, R. Leus, and F. C. R. Spieksma, "Exact algorithms for coloring graphs while avoiding monochromatic cycles", *Proc. of 6th International Conference on Algorithmic Aspects in Information and Management*, Weihai, China, July 2010, pp. 229-242.
[15] P. Heggernes, P. Van 'T Hof, D. Lokshtanov, and Ch. Paul, "Obtaining a Bipartite Graph by Contracting Few Edges", *SIAM Journal on Discrete Mathematics*, vol. 27, issue 4, 2013, pp. 21432156.
[16] Th. H. Cormen, Ch. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. MIT Press and McGraw-Hill, 2009.
[17] M. Kubale (editor), *Graph Colorings*, *Contemporary Mathematics*, vol. 352, 2004.