Combinational Circuits without False Paths*

A. Matrosova
Department of Applied Mathematics and Cybernetics
Tomsk State University
Tomsk, Russia
mau11@yandex.ru

D. Kudin
Gorno-Altaisk State University,
Gorno-Altaisk, Russia
dvkudin@gmail.com

E. Nikolaeva
Department of Applied Mathematics and Cybernetics
Tomsk State University
Tomsk, Russia
nikoleve-ea@yendex

Abstract

It is known that identifying false paths allows improving a circuit performance but finding false paths is associated with large calculations. In this paper we suggest methods of combinational circuit design that guarantee an absence of false paths in the resulting circuits. Some design methods keeping the specification formulae are considered. The sufficient condition of an absence of false paths in a combinational circuit is formulated. It is shown that the certain types of specification formulae together with the proper design methods keeping the formulae provide this condition for resulted circuits. Examples of the circuits without false paths are given.

Keywords: path delay fault (PDF), irredundant sum of products (irredundant SoP), disjoint sum of products (DSoP), binary decision diagram (BDD), Reed-Muller expression, false path.

1. Introduction

It is known that identifying false paths allows improving a circuit performance but finding false paths is associated with large calculations. Efforts have been made to minimize these calculations [1-3]. In this paper we suggest methods of combinational circuit design that guarantee an absence of false paths in the resulting circuits. Our approach is based on connection of circuit path with ENF literal properties. In particular if the certain path is false then there is no test pattern both for stuck-at one and stuck at zero fault of the corresponding literal of ENF. On the other hand, if there exists a test pattern either for stuck-at one fault or stuck-at zero fault for the literal of ENF, the corresponding path is not false. We analyze expressions generated by combinational circuit (system F_c). They differ from ENFs corresponding to the circuit outputs by an absence of sequences of numbers of gates that comprise the circuit paths. Stuck-at one and stuck-at zero faults of literals of such expressions are considered. We derive the sufficient condition of an absence of false paths in a combinational circuit. It is connected with the property of system F_c literals. Some types of expressions that fulfill the sufficient condition are considered. It means that if a combinational circuit originates such type of expressions then this circuit has no false paths. It is shown that some well-known synthesis methods may originate such expressions and consequently combinational circuits derived with using these methods have no false paths.

In the Section 2 the problem of keeping specification formulae (system F) that are used during design of a combinational circuit by the system F_c extracted from the resulting circuit is discussed. In the Section 3 some design methods that keep the specification formulae are considered. In the Section 4 the sufficient condition for an absence of false paths in a combinational circuit is formulated, also its implementation for the different synthesis methods and the corresponding specification formulae is illustrated.

2. Keeping specification formulae

A combinational circuit design as a rule consists of the several stages.

Firstly we obtain the system of incompletely specified Boolean functions.

Then this system is changed for the minimized system F of completely specified Boolean functions represented by the system of SoPs or the system of Reed-Muller expressions or other formulae. Here we will consider only systems of SoPs and systems of Reed-Muller expressions. Notice that we consider Reed-Muller expressions in which the products are pairwise orthogonal (products are connected each other by the operation + (XOR)). Call the system F as specification formulae for a combinational circuit

design or simply specification formulae. The number of formulae of the system F is equal to the number of the combinational circuit outputs.

Next the system F is implemented by a combinational circuit C consisting of gates. The circuit structure depends on applied synthesis method. In this paper we deal with the synthesis methods that keep the specification formulae.

Consider a combinational circuit C that is derived from the system F. The structure of the circuit C i.e. its gates and their connections are known. Here combinational circuit consists of either from gates OR, AND, NOR, NAND, NOT or from gates XOR, AND, NOT. In the last case, the gate NOT is used only for input variables. Having this information, we extract from the circuit C, the system F_c of SoPs or the system F_c of Reed-Muller expressions in the following way.

1. Move from the output of the circuit C to its inputs.

2. Substitute gate formulae instead of the proper internal variables of the circuit C. Each formula may depend on both internal and input variables of the circuit. Use De Morgan rules for circuits from the gates OR, AND, NOR, NAND, NOT so that inversions appear only on input variables

3. Eliminate brackets using the distributive law: $a(b \vee c) = ab \vee ac$ for a circuit from the gates OR, AND, NOR, NAND, NOT and distributive law: $a(b+c) = ab+bc$ for a circuit from XOR, OR, NOT gates. Here the operation "+" means XOR.

During eliminating brackets the operations of a formula simplification: $ab \vee b = b$, $ab \vee \bar{a}b = b$, $ab \vee \bar{a}c = ab \vee \bar{a}c \vee bc$, $a \vee \bar{a} = 1$, $a\bar{a} = 0$, $a \vee a = a$, $aa = a$ are forbidden.

4. Having executed steps 1-3 for each output of the circuit C we get system F_c of formulae.

Let f_c be the formulae of F_c corresponding to the one output sub-circuit of C.

Theorem 1. Each literal of f_c corresponds to the certain path correlating with one output sub-circuit of C and for each path of the sub-circuit there is at least one literal in the formula f_c.

Proof. Formula f_c differs from ENF of the same sub-circuit only by sequences of numbers of elements comprising the paths corresponding to ENF literals when applying the distributive law $a(b \vee c) = ab \vee ac$ and except using the operation EXOR instead of AND between products when applying the distributive law $a(b+c) = ab+bc$. It means that the proposal being right for ENF is right for f_c. The theorem is proved.

Consider the examples of the circuits of Fig. 1, Fig 2. Let their specification formulae be known and consists of formulae for one output circuits. For the circuit of Fig. 1 we have

$$f = \bar{a}be \vee b\bar{c}de \vee a\bar{b}c \vee ad\bar{} \quad (1)$$

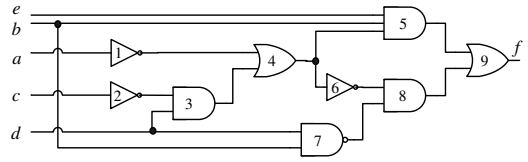

Figure 1. Combinational circuit from the gates OR, AND, NOR, NAND, NOT

For the circuit of Figure 2 we have

$$\begin{aligned} f = {} & x_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2x_3\bar{x}_4x_5 + x_1\bar{x}_2x_3x_4\bar{x}_5 + \\ & + x_1x_2\bar{x}_4x_5 + x_1x_2x_4\bar{x}_5 + \bar{x}_1\bar{x}_2\bar{x}_4x_5 + \\ & + \bar{x}_1\bar{x}_2x_4\bar{x}_5 + \bar{x}_1x_2\bar{x}_3\bar{x}_4x_5 + \bar{x}_1x_2\bar{x}_3x_4\bar{x}_5 + \\ & + \bar{x}_1x_2x_3\bar{x}_5 \end{aligned} \quad (2)$$

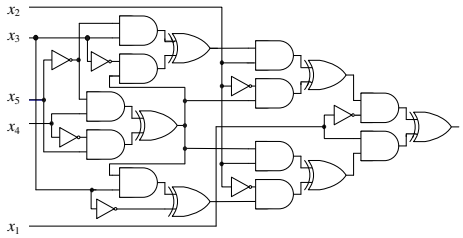

Figure 2. Combinational circuit from gates XOR, AND, NOT

For the circuit of Fig. 1 we extract the expression

$$f_c = \bar{a}be \vee b\bar{c}de \vee a\bar{b}c \vee acd\bar{} \vee a\bar{b}d\bar{} \vee ad\bar{}d\bar{} \quad . \quad (3)$$

For the circuit of Fig. 2 we extract the expression:

$$\begin{aligned} f_c = {} & x_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2x_3\bar{x}_4x_5 + x_1\bar{x}_2x_3x_4\bar{x}_5 + \\ & + x_1x_2\bar{x}_4x_5 + x_1x_2x_4\bar{x}_5 + \bar{x}_1\bar{x}_2\bar{x}_4x_5 + \\ & + \bar{x}_1\bar{x}_2x_4\bar{x}_5 + \bar{x}_1x_2\bar{x}_3\bar{x}_4x_5 + \bar{x}_1x_2\bar{x}_3x_4\bar{x}_5 + \\ & + \bar{x}_1x_2x_3\bar{x}_5 \end{aligned} \quad (4)$$

Thus the specification formula f (1) of the circuit of Fig. 1 and the formula f_c (3) extracted from the circuit structural description with using the rules 1-3 are different. As for the circuit of Fig. 2 its specification formula and the formula extracted from the structural description are the same: $f = f_c$. We will say that the circuit of Fig.2 keeps its specification formula but the circuit of Fig.1 does not keep its specification formula.

For the multi output circuit we will say that a circuit keeps its specification formulae if the formulae of the system F_c extracted from the circuit structural description with using rules 1-4 coincide with the specification formulae of the system F: $F = F_c$ .Otherwise the circuit does not keep the specification formulae.

* This work is partly supported by the TSU Competitiveness Improvement Program.

Consider some of design methods that provide keeping the specification formulae.

3. Design methods keeping the specification formulae

3.1. Multilevel synthesis method based on algebraic division

Specification formulae represent system F of m SoPs. Here m is the number of the circuit C outputs. Each SoP is an algebraic expression i.e. for any two products k_1, k_2 from a SoP neither $k_1 \le k_2$ nor $k_2 \le k_1$ take place. For example $ab \vee ad \vee cdef$ is algebraic SoP but $ab \vee ad \vee cdef$ is not algebraic SoP.

A multiplication of algebraic SoPs $D_1$and D_2 is an algebraic multiplication if sets of D_1, D_2 variables don't intersect. For example if $D_1 = ab \vee bc \vee ac$ and $D_2 = d \vee e$ then $D_1D_2 = (ab \vee bc \vee ac)(d \vee e)$ is algebraic multiplication.

Representation of a SoP D as $D = D_1D_2$ v D_3 is a division of a SoP D. If D_1D_2 is algebraic multiplication, D_3 is a reminder that contains the least number of products and after excluding brackets during multiplication we get the formula that coincides with the SoP D then this division is called a weak division.

Consider a weak division so that D_1 cannot be represented as $D_1 = k_1D_4$ where k_1 is a product and $D_2 = k$ where k is also a product. In that case $D = D_1k$ v D_3. Then D_1 is a kernel of the SoP D and k is a co-kernel of the SoP D. For the same D it is possible to execute several weak divisions and consequently find several kernels and co-kernels.

The idea of multilevel synthesis method is as follows. Kernels (sometimes co-kernels) are changed for the new variables. As a result, new SoPs are added to the previous System of SoPs but new system of SoPs is simpler than the previous one by the number of literals.

Theorem 2. Multilevel synthesis method keeps the specification formulae.

Proof. Consider current step of extracting kernel or/and co-kernel from some SoP. Notice that finding kernels and co-kernels is based on the weak division. It means that changing kernel or/and co-kernel for the proper SoP or the product we obtain the same previous SoP. Consequently multilevel synthesis method keeps the circuit specification formulae, that is $F = F_c$. The theorem is proved.

3.2 Two level synthesis methods

Specification formulae represent system F of SoPs. Two level synthesis methods are based on choosing a set of product factors and SoPs factors.

A product factor of the system F is derived from the certain product of the system F by elimination of some literals from the product. After getting a set of product factors each system F product is covered by the certain factors from this set.

They say that the certain factors cover the product k if a multiplication of these factors gives rise to all literals that are present in the product k. If at least one literal appears two or more times under multiplication then such covering is called redundant otherwise the covering is called irredundant.

A SoP factor of the system F is derived from the certain SoP of the system F by elimination of some products from the system SoP. After getting a set of SoPs factors each SoP of the system F is covered by the certain SoP factors from this set.

They say that the certain factors cover the SoP f of the system F if disjunction of these factors gives rise to all products that are present in the SoP f. If at least one product appears two or more times under disjunction then such covering is called redundant otherwise the covering is called irredundant.

As a result of covering the products and the SoPs of the system F by the chosen factors we get the combinational circuit C.

We restrict our consideration of two level synthesis methods in which only irredundant covering products and SoPs of the system F are acceptable. Call such methods as two level synthesis methods based on irredundant covering.

Theorem 3. Two level synthesis method based on irredundant covering keeps the specification formulae.

Proof. If we use two level synthesis method based on irredundant covering then during extraction SoPs f_c from the structural description of the circuit C we obtain only products of the system F and only its SoPs, that is $F = F_c$. The theorem is proved.

3.3. ROBDD based synthesis methods

3.3.1. Direct gate covering SBDD. Let a combinational circuit C behavior is described by the system of ROBDDs that are merged into Shared ROBDD (SBDD). SBDD is obtained from separate ROBDDs, corresponding to the different outputs of the circuit C as follows. All 1-terminal (0-terminal) nodes of ROBDDs are merged into one 1-teminal (0-terminal) node. If two or more internal nodes are roots of the isomorphic graphs then these graphs are merged into one internal node of SBDD. As a result we get graph (SBDD) with m roots and two terminal nodes. Here m is the number of the circuit C outputs. Notice that SBDD is compact representation of the system of the Disjoint Sum of Products (DSoPs). In Disjoint Sum of Products the cubes corresponding to any two

products don't intersect. Each DSoP may be extracted from SBDD. System F of DSoPs extracted from SBDD call specification formulae.

Let each node but the terminal one is covered with the gate sub-circuit of Figure 3. This sub-circuit implements the Shennon decomposition formula that is used under construction of ROBDD.

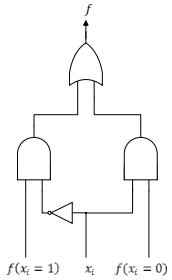

Figure 3. .An implementation of the Shennon decomposition formula

After covering we get the circuit *C*. Call this synthesis method as method of direct covering SBDD nodes by the corresponding sub-circuit from gates or simply the method of direct covering of SBDD by gates. Extract the system F_c of DSoPs from the circuit *C* of Fig. 2 in above mentioned way (Section 2).

Theorem 4. Synthesis method of direct covering SBDD by gates keeps the specification formulae.

Proof. Take into consideration that extraction of formula for the certain output of the circuit *C* using its structural description is similar to extraction of the DSoP from the corresponding ROBDD of SBDD. It means that extracted formula is the DSoP which belongs to the specification formulae, that is $f = f_c$ and consequently $F = F_c$. The theorem is proved.

Consider ROBDD of Fig. 4. For the simplicity we consider one output circuit whose SBDD coincides with ROBDD. Its specification formula is as follows:

$$f = x_1\overline{x}_2\overline{x}_3 \vee x_1\overline{x}_2x_3\overline{x}_4x_5 \vee x_1\overline{x}_2x_3x_4\overline{x}_5 \vee \\ \vee x_1x_2\overline{x}_4x_5 \vee x_1x_2x_4\overline{x}_5 \vee \overline{x}_1\overline{x}_2\overline{x}_4x_5 \vee \\ \vee \overline{x}_1\overline{x}_2x_4\overline{x}_5 \vee \overline{x}_1x_2\overline{x}_3\overline{x}_4x_5 \vee \overline{x}_1x_2\overline{x}_3x_4\overline{x}_5 \vee \\ \vee \overline{x}_1x_2\overline{x}_3\overline{x}_5 \qquad (5)$$

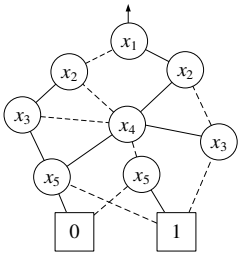

Figure 4. ROBDD

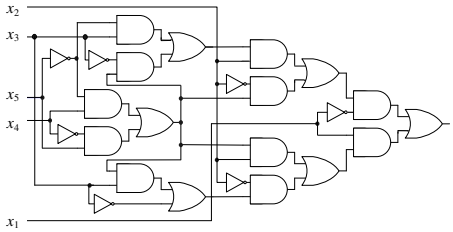

Figure 5. Combinational circuit obtained by direct covering ROBDD

Direct covering ROBDD by gate sub-circuit of Fig. 3 gives rise to the circuit of Fig.5 that keeps the specification formula (5).

3.3.2 Special gate covering SBDD. Change the system *F* of DSoPs corresponding to SBDD for the system *F**. Each formula *f**of *F** is derived from the formula of *F* by changing the operation $\vee$ (disjunction) for the operation + (XOR). Notice that formulae from *F** are Reed-Muller expressions. Take into consideration that the functions *f, f**of the systems *F, F** corresponding to the same circuit *C* output are equal. Consequently systems *F*, *F** are also equal. It is because of pairwise orthogonality of products of the formulae *f, f**.

Let *F** be specification formulae. We have SBDD that originates the system *F**. Use this SBDD and cover each its node but terminal one by the sub-circuit of Fig.6. As a result we derive the combinational circuit *C* of Fig.2.

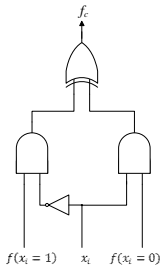

Figure 6. Special gate sub-circuit

For this circuit we have the formula f_c

$$f_c = x_1\overline{x}_2\overline{x}_3 + x_1\overline{x}_2x_3\overline{x}_4x_5 + x_1\overline{x}_2x_3x_4\overline{x}_5 + \\ + x_1x_2\overline{x}_4x_5 + x_1x_2x_4\overline{x}_5 + \overline{x}_1\overline{x}_2\overline{x}_4x_5 + \\ + \overline{x}_1\overline{x}_2x_4\overline{x}_5 + \overline{x}_1x_2\overline{x}_3\overline{x}_4x_5 + \overline{x}_1x_2\overline{x}_3x_4\overline{x}_5 + \\ + \overline{x}_1x_2x_3\overline{x}_5$$

Theorem 5. Special gate covering SBDD keeps the specification formulae.

Proof. Take into consideration that extracting the formula for a certain output of the circuit C using its structural description is similar to extracting Reed-Muller expression from the corresponding ROBDD of SBDD. It means that extracted formula is the Reed-Muller expression which coincides with the

specification formulae, that is $f_c = f^*$ and consequently $F_c = F^*$.The theorem is proved.

Thus we considered the certain synthesis methods that keep specification formulae. Such synthesis methods may be applied for providing useful circuit properties on the stage of forming specification formulae that is on the stage of a description of a circuit behavior. In this paper in particular we will provide an absence of false paths in combinational circuits for account of properties of specification formulae. It is possible to develop other synthesis methods that keep specification formulae. But this problem is out of our consideration.

4. Properties of some specification formulae that guarantee an absence of false paths

4.1. The sufficient conditions of a false path absence

It is known that that both robust or non robust detectable PDF demands test pair v_1, v_2 for its manifestation. In the paper [4] it is shown that a test pattern v_2 is a test pattern either for stuck-at one fault (for falling transition of the path) or stuck at zero fault (for rising transition of the path) of ENF literal corresponding to the considered fault path of the circuit that gives rise to the ENF. Notice that stuck-at one (stuck-at zero) fault means that each appearance of this literal in ENF products changes for the constant 1 (0). Finding a test pattern v_2 is reduced to looking through the products that contain the literal. If there is now test pattern both for stuck-at one and stuck at zero faults of the same path then this path is false one [4]. Alternately if there is a test pattern either for stuck-at one or for stuck-at zero fault of the ENF literal corresponding to the considered path then this path is not false.

Take into account that formula f_c corresponding to the one output sub-circuit of a circuit C differs from the ENF of the same sub-circuit only by the sequences of numbers of elements comprising the paths corresponding to ENF literals. For a while we consider only f_c representing either SoP or DSoP. Each literal of f_c corresponds to the certain path of the one output sub-circuit of the circuit C. Similar literals of the different products of f_c may be correlated either with the same path of the sub-circuit or the different paths. For example for the circuit of Fig. 1 we have ENF as follows:

$$E = \overline{a}_{1459} b_{59} e_{59} \vee b_{59} \overline{c}_{23459} d_{3459} e_{59} \vee$$
$$\vee a_{14689} \overline{b}_{789} c_{234689} \vee a_{14689} c_{234689} \overline{d}_{789} \vee$$
$$\vee a_{14689} \overline{b}_{789} \overline{d}_{34689} \vee a_{14689} \overline{d}_{34689} \overline{d}_{789}$$

This ENF originates the formula f_c after excluding the index sequences.

$$f_c = \overline{a}be \vee b\overline{c}de \vee a\overline{b}c \vee ac\overline{d} \vee a\overline{bd} \vee a\overline{dd}$$

In the formula f_c the literal $\overline{d}$ corresponds to the same path 34689 in the fifth and sixth products. This repeated literal in the sixth product corresponds to the path 789 and the same literal $\overline{d}$ in the fourth and fifth products corresponds to the different paths. The literal e in the different products of f_c corresponds to the only circuit path. Now we may formulate the sufficient condition of an absence of false paths in the following way.

If each literal of the system F_c has a test pattern either *for its stuck-at one fault or stuck-at zero fault then the corresponding circuit C has no false paths*

Consider examples of such systems.

4.2. System of Irredundant SoPs

Examine irredundant SoP consisting of prime implicants. Call *b*-fault of SoP [6] changing the literal from one product for the constant 1 that is disappearing literal in the product of the SoP. Call *a*-fault of SoP [6] changing the literal from one product for the constant 0 that is disappearing the certain product that contains this literal.

It is known that for each *a* (*b*)-fault of an irredundant SoP consisting from prime implicants there is a test pattern [6]. *It means that if f_c for one output sub-circuit of circuit C is* irredundant *SoP consisting of prime implicants then there is no false paths in this sub-circuit.*

If F_c consists of irredundant *SoPs then there are no false paths in the circuit C as a whole.*

Thus if we have the system F consisting of the irredundant SoPs *and we use either multilevel synthesis method or two level synthesis method based on irredundant factorization then we get the circuit C without false paths.*

Notice that if any circuit C gives rise to F_c (irrespective of the method used to design) that is a system of the irredundant SoPs consisting of prime implicants then this circuit has no false paths. For example the circuit in the Roth paper [7] has no false paths.

4.3. Irredundant system

Consider an irredundant system F of Boolean functions consisting of prime system implicants. Let this system be for example an implementation of a system of incompletely specified Boolean functions. Each system implicant consists of a product and its characteristic enumerating functions for which the product is valid. Notice that in irredundant system of Boolean functions any function from characteristic of a system product cannot be excluded. It means that it is impossible to exclude any product from a SoP derived from F (A SoP from F may contain not only prime implicants). Thus a-fault of any literal of the system F is detectable that is there is a test pattern for this fault.

Therefore if a system F is irredundant system of Boolean functions and either multilevel synthesis method or two level synthesis method based on irredundant factorization is used for deriving combinational circuit C, then there is a test pattern for a-fault of each literal of the SoP created by the sub-circuit of the circuit C. *It means that circuit C has no false paths.*

Notice that if any circuit C gives rise to F_c that is an irredundant system (irrespective of the method used to design circuit C) then this circuit has no false paths.

4.4 System of DSoPs

Consider a system F of DSoPs. Any a-fault of a DSoP is detectable because of pairwise orthogonality its products. If circuit C is derived from SBDD by direct covering of its nodes with sub-circuits implementing Shannon decomposition in above mentioned way then F_c is the system of DSoPs. *It means that circuit C has no false paths.*

Notice that if any circuit C gives rise to F_c that is a system of DSoPs (irrespective of the method used to design circuit C) then the circuit C has no false paths

4.5 System of Reed- Muller expressions

Consider a system F^* of Reed-Muller expressions. In the paper [5] is shown that each $a(b\ )$-fault of a formula f^*(f^* from F^*) is detectable. Be reminiscent that for each path of the circuit C there is at least one literal in the corresponding f_c if the circuit C is derived by using special gate covering SBDD nodes. In this case $f^* = f_c$). *It means that circuit C has no false paths*

One of the insights is that we cannot expand these results to a combinational part of a sequential circuit. The matter is that if we have a test pattern v_2 for stuck-at fault of the literal corresponding to the certain path there is no guarantee of delivering the proper v_1 even for non robust PDF because of a presence of state variables in a combinational equivalent. In this case the problem of false paths verification becomes essentially more complex in comparison with combinational circuits.

5. Conclusion

In this work it is shown that some well-known synthesis methods may be used to obtain combinational circuits without false paths. For that it is necessary to apply the proper specification formulae describing a circuit behavior. These specification formulae are also used in practice. One of the insights is that we cannot directly expand these results to a combinational part of a sequential circuit. The last problem demands additional efforts.

6. References

[1] H. Chang, J. Abraham "VIPER: An Efficient Vigorously Sensitizable Path Extractor", *Proceedings of 30th ACM/IEEE Design Automation Conference*, pp. 112-117, 1993.

[2] J. Bhadra, M. S. Abadir, J. A. Abraham. "A Quick and Inexpensive Method to Identify False Critical Paths Using ATPG Techniques: an Experiment with a PowerPC Microprocessor", *Proceedings of 36th ACM/IEEE Design Automation Conference*, pp. 737-741, 1999

[3] R. Raimi, J. A. Abraham. "Detecting False Timing Paths: Experiments on PowerPC(TM) Microprocessors", *Proceedings of IEEE 2000 Custom Integrated Circuits Conference*, pp. 71-74, May. 2000.

[4] A. Matrosova, V. Lipsky, A. Melnikov, and V. Singh, "Path delay faults and ENF", *Proceeding of EW&DT Symposium*, 2010, pp. 164-167.

[5] A. Yu. Matrosova, D. V. Kudin, E. A. Nikolaeva, E.V. Roumjantseva, "Providing full delay testability for circuits obtained by covering BDDs," *Vestnik of Tomsk State University. Control, Computers and Informatics*, 2013, no. 2, pp. 130-139 (in Russian).

[6] Kohavi I., Kohavi Z. "Detection of Multiple Faults in Combinational Logic Networks", *IEEE Trans. On Computers.* 1972. v.C-20, pp. 556-568.

[7] J.P. Roth, "Diagnosis of Automata Failures: A Calculus and a Method", *IBM Journal of Research and Development*, vol. 10, No.4, pp.278-291, July, 1966.