UDC 004.4–004.9

REVERSIBLE IF-DECISION DIAGRAMS

Prihozhy A. A.
Belarusian National Technical University, Minsk, Belarus,
prihozhy@yahoo.com

All logical quantum circuits are reversible [1–7]. This paper introduces reversible if-decision diagrams for modelling, synthesis, optimization and verification of the quantum circuits.

Reversible logical operations. Let $x = (x_1, \ldots, x_n)$ be a Boolean vector variable. A scalar Boolean function $f(x)$ is a mapping $B^n \to B$, B = {0, 1}. Let a vector Boolean function $F(x) = (f_1, \ldots, f_n)$: $B^n \to B^n$ is given by vector $(x_1, \ldots, x_{i-1}, f(x_1, \ldots, x_n), x_{i+1}, \ldots, x_n)$ of scalar functions $f_1 = x_1, \ldots, f_{i-1} = x_{i-1}, f_i = f, f_{i+1} = x_{i+1}, \ldots, f_n = x_n$. In $F(x)$, the number of components is equal to the number of variables.

Definition 1. Boolean function $f(x_1, \ldots, x_n)$ of n arguments is n-reversible if an index $i \in \{1, \ldots, n\}$ exists such that the vector function $F(x) = (x_1, \ldots, x_{i-1}, f(x_1, \ldots, x_n), x_{i+1}, \ldots, x_n)$ is bijective.

Let analyze Boolean binary operations for reversibility. Boolean binary exclusive or operation is given by $f = x_1 \oplus x_2$. The truth table in fig. 1 proves that $F = (x_1, x_1 \oplus x_2)$ is bijective and the $\oplus$ operation is 2-reversible.

Inputs		Outputs	
x_1	x_2	x_1	$x_1 \oplus x_2$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Figure 1 – Proof of 2-reversibility of Boolean binary exclusive or operation

Boolean binary conjunction is given by $f = x_1 \wedge x_2$, and Boolean binary disjunction is given by $f = x_1 \vee x_2$. The truth table in Figure 2a refutes the 2-reversibility of the conjunction as there are two input vectors which result in the same output vector. The truth table in fig. 2b refutes the 2-reversibility of the disjunction.

Inputs		Outputs	
x_1	x_2	x_1	$x_1 \wedge x_2$
0	0	0	0
0	1	0	0

Inputs		Outputs	
x_1	x_2	x_1	$x_1 \vee x_2$
0	0	0	0
0	1	0	1

1	0	1	0
1	1	1	1

a)

1	0	1	1
1	1	1	1

b)

Figure 2 – Refutation of 2-reversibility of: a – Boolean conjunction and b – Boolean disjunction

Let check the most important three ternary Boolean operations for 3-reversibility: *the if-then-else (ite)* operation; the majority (*maj*) operation, and the *xor-and-accumulation* (*xac*) operation. The ternary *xor* operation is a composition of two binary *xor* operations; therefore, it is 3-reversible.

The $ite(x_1, x_2, x_3)$ operation is given by $ite = x_1 \wedge x_2 \vee \neg x_1 \wedge x_3$. Its arguments are not symmetric; therefore, we consider two cases. Two truth tables in fig. 3 refute the operation to be 3-reversible. In case 1, when *ite* is substituted instead of the first variable x_1, input vectors (0, 0, 0) and (1, 0, 0) result in the same output vector (0, 0, 0), and input vectors (0, 1, 1) and (1, 1, 1) result in the same output vector (1, 1, 1), therefore, $F = (ite(x_1, x_2, x_3), x_2, x_3)$ is not bijective and *ite* is not 3-reversible. In case 2, when *ite* is substituted instead of the second argument x_2, input vectors (0, 0, 0) and (0, 1, 0) result in output vector (0, 0, 0), and input vectors (0, 1, 1) and (1, 1, 1) result in output vector (1, 1, 1). Therefore, function $F = (x_1, ite(x_1, x_2, x_3), x_3)$ is not bijective and function *ite* is not 3-reversible.

Inputs			Outputs		
x_1	x_2	x_3	*ite*	x_2	x_3
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	0	1	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	1
1	1	0	1	1	0
1	1	1	1	1	1

a)

Inputs			Outputs		
x_1	x_2	x_3	x_1	*ite*	x_3
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	1	1	1

b)

Figure 3 – Refutation of 3-reversibility of Boolean ternary *ite* operation:
a – $F = (ite(x_1, x_2, x_3), x_2, x_3)$; b – $F = (x_1, ite(x_1, x_2, x_3), x_3)$

The *maj* operation is given by $maj = x_1 \wedge x_2 \vee x_1 \wedge x_3 \vee x_2 \wedge x_3$. Since all three arguments are symmetric, fig. 4 describes the truth table of function $F = (maj(x_1, x_2, x_3), x_2, x_3)$, which completely checks the 3-reversibility of *maj*. It can be noticed that each of two pairs of input vectors is mapped to the same marked output vector, therefore the F is not bijective and the *maj* is not 3-reversible.

Inputs			Outputs		
x_1	x_2	x_3	*maj*	x_2	x_3
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	1	1	1

Figure 4 – Refutation of 3-reversibility of Boolean ternary *maj* operation with $F = (maj(x_1, x_2, x_3), x_2, x_3)$

The *xac* operation is given by $xac = x_1 \oplus (x_2 \wedge x_3)$. The truth table of fig. 5a proves that function $F = (xac(x_1, x_2, x_3), x_2, x_3)$ is bijective and therefore operation *xac* is 3-reversible. It is interesting that *xac* is not reversible for $F = (x_1, xac(x_1, x_2, x_3), x_3)$. Among the considered three ternary Boolean functions, *xac* is the only reversible one.

Inputs			Outputs		
x_1	x_2	x_3	*xac*	x_2	x_3
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	1	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	0	1	1

a)

Inputs			Outputs		
x_1	x_2	x_3	x_1	*xac*	x_3
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	0
0	1	1	0	1	1
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	1	0	1

b)

Figure 5 – Check of 3-reversibility of Boolean ternary *xac* operation: a – it is 3-reversible with $F = (xac(x_1, x_2, x_3), x_2, x_3)$; b – it is not 3-reversible with $F = (x_1, xac(x_1, x_2, x_3), x_3)$

Definition 2. Function $f(x_1, \ldots, x_n)$ is n+1-reversible if for function $f'(x_1, \ldots, x_n, c) = f(x_1, \ldots, x_n)$ of n+1 arguments, where c = 0 or c = 1, the vector function $F = (x_1, \ldots, x_n, f'(x_1, \ldots, x_n))$ is bijective.

Above we have proved that the binary Boolean conjunction is not 2-reversible. Let check if it is 3-reversible. To do this, we write down $x_1 \wedge x_2 = xac(0, x_1, x_2)$ and $F = (xac(0, x_1, x_2), x_1, x_2)$. The truth table in Figure 6 that contains four value rows proves that the F function is bijective and the binary conjunction is 3-reversible.

Inputs			Outputs		
0	x_1	x_2	$xac(0, x_1, x_2)$	x_1	x_2
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	1	1	1

Figure 6 – Proof of 3-reversibility of binary Boolean conjunction using $F = (xac(0, x_1, x_2), x_1, x_2)$

Binary Boolean disjunction is not 2-reversible. Let check if it is 3-reversible. To do this, we write down $x_1 \vee x_2 = xac(0, x_1, x_2) \oplus x_1 \oplus x_2$ and $F = (xac(0, x_1, x_2) \oplus x_1 \oplus x_2, x_1, x_2)$. The truth table with four value rows in fig. 7 proves that the F function is bijective, and the binary disjunction is 3-reversible.

Inputs			Outputs		
0	x_1	x_2	$xac(0,x_1,x_2) \oplus x_1 \oplus x_2$	x_1	x_2
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	1	0
0	1	1	1	1	1

Figure 7 – Proof of 3-reversibility of Boolean binary disjunction using
$F = (xac(0, x_1, x_2) \oplus x_1 \oplus x_2, x_1, x_2)$

The 4-reversibility of the *ite* and *maj* ternary operations can be proved in a similar way. A superposition of the Boolean *exclusive or*, *xor-and-accumulation,* and constants 1 and 0 operations constitute a basis for describing any reversible Boolean function with the same or increased number of input and output variables. If the function of n arguments is n-reversible it can be directly described in the basis, otherwise a function extension can be constructed with additional arguments (ancillas). Searching for an extension with a minimal number of ancillas is a subject of optimization.

Let a function of n arguments that is not n-reversible in general case be given by $f(x_1, \ldots, x_n)$. To find its representation or to transform it to a good quality reversible function, various expansions can be examined. In the paper we focus on decision diagrams. The most famous is the Binary Decision Diagram (BDD). Several BDD types are known, including complete, free, ordered, reduced diagrams [8–9]. A Reduced Ordered BDD (ROBDD) is a model for solving such problems as modelling, synthesis, test generation, and verification of digital systems, which are implemented as electronic, quantum or other circuits. Figure 8a depicts a BDD's nonterminal node. It is labeled with Boolean variable x_i and has two outgoing edges labeled *low* and *high* and leading to daughter sub-diagrams g and h. The Shannon expansion [10] defines the node semantics with the equation

$$f = x_i \wedge g \vee \neg x_i \wedge h \tag{1}$$

where $h = f_{xi=0}$ and $g = f_{xi=1}$ are residual functions called positive and negative cofactors respectively.

Works [8, 9] generalized the Shannon expansion to

$$f = c \wedge v \vee \neg c \wedge u \tag{2}$$

where c is an arbitrary Boolean function of n arguments, $v = min\ (f \mid c)$, $u = min\ (f \mid \neg c)$ and $min\ (f \mid c)$ is a minimization operation of function f over characteristic function c. Expansion (2) defines the semantics of a nonterminal node of the If-Decision Diagram (IFD) [11, 12] depicted in fig. 8b. It is easy to see that (1) and (2) use the *ite* ternary operation, which is not 3-reversible, therefore the BDD and IFD are not reversible-style representations.

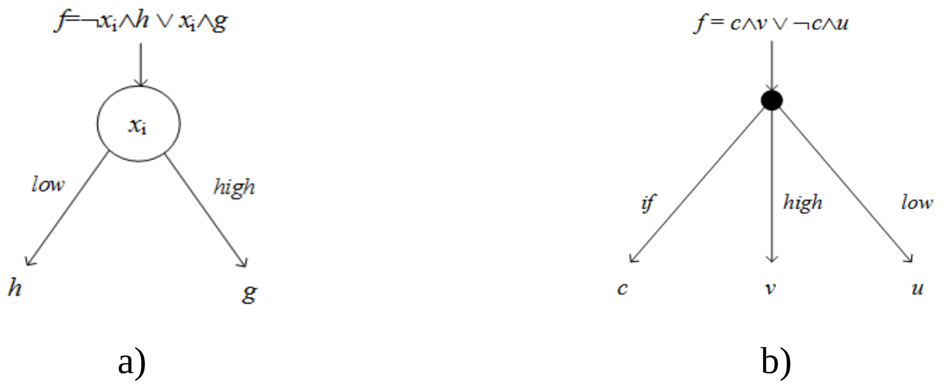

Figure 8 – Nonterminal node of: a – binary decision diagram; b – if-decision diagram

The positive and negative Davio [13] expansions (3) and (4) of Boolean function $f(x)$ are derived from the Shannon expansion (1):

$$f = h \oplus x_i \wedge e \tag{3}$$

$$f = g \oplus \neg x_i \wedge e \tag{4}$$

where $e = g \oplus h$. To prove (3), we equivalently transform it to (1) in the way as follows:

$\mathrm{f = h} \oplus x_i \wedge \mathrm{(g \oplus h)} =$

$= (h \wedge \neg(x_i \wedge (g \oplus h))) \vee (\neg h \wedge x_i \wedge (g \oplus h)) =$

$= (h \wedge (\neg x_i \vee \neg(g \oplus h))) \vee (\neg h \wedge x_i \wedge g \wedge \neg h) =$

$= (h \wedge \neg x_i \vee h \wedge \neg(g \oplus h)) \vee (x_i \wedge g \wedge \neg h) =$

$= (\neg x_i \wedge h) \vee (g \wedge h) \vee (x_i \wedge g \wedge \neg h) =$

$= (x_i \wedge g) \vee (\neg x_i \wedge h)$

Equation (4) can be proved in the similar way. Based on (2), the author of works [14, 15] developed the following xor-based expansions:

$$f = u \oplus c \wedge w \tag{5}$$
$$f = v \oplus \neg c \wedge w \tag{6}$$

where c is an arbitrary Boolean function of n arguments and $w = v \oplus u$. Expansions (5) and (6) generalize the positive and negative Davio expansions (3) and (4). Their proof is like the proof of (3).

Expansions (3) and (4) lie in the basis of creating positive pFDD (fig. 9a) and negative nFDD (fig. 9b) functional decision diagrams respectively. Expansions (5) and (6) constitute a basis for creating positive pFIFD (fig. 9c) and negative nFIFD (fig. 9d) functional if-decision diagrams respectively [16–20]. Since pFIFD and nFIFD provide much larger possibilities for modelling logical functions due to replacing a variable x_i with an arbitrary logical function c, they are more suitable for modelling logical systems and for the design automation.

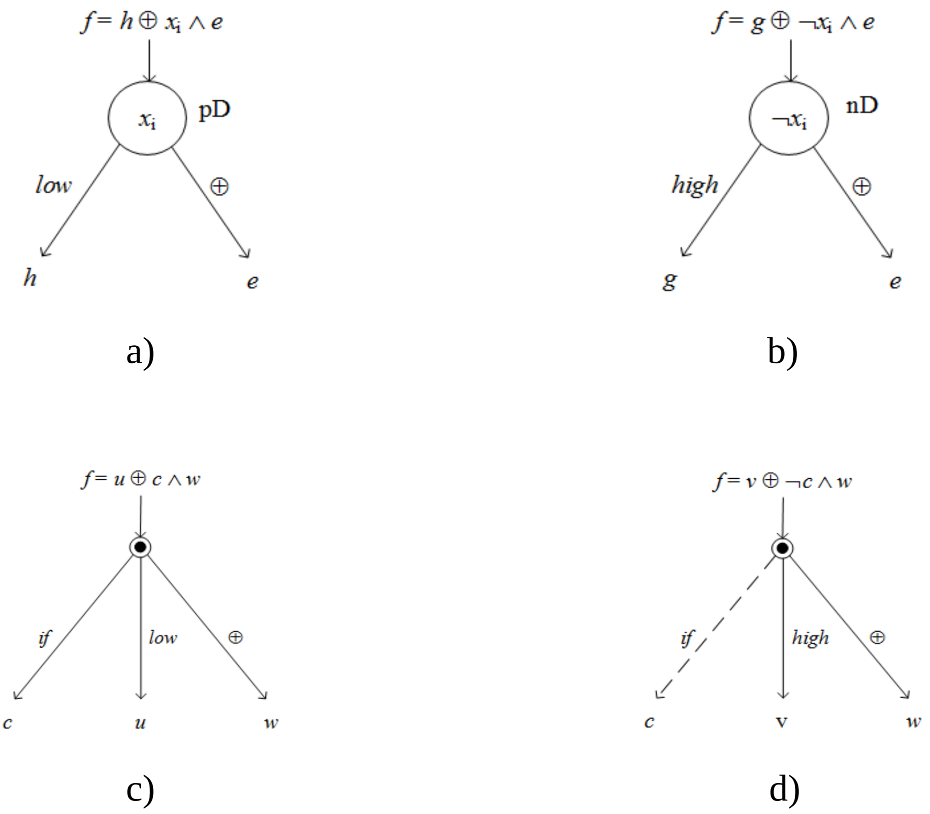

Figure 9 – Nonterminal node of: a – pFDD; b – nFDD; c – pFIFD; d – nFIFD

In case, when all three input variables are essential in any of the ternary Boolean operations defined by (3)–(6), and due to the operations' 3-reversibility (in fact these are the xor-and-accumulation Boolean ternary operation), we call the decision diagrams pFDD, nFDD, pFIFD and nFIFD reversible. In the case, no additional variables (ancillas) are needed. When one or two of three variables are unessential, the functions describe binary or unary operations. If the binary operation is 3-reversible (like Boolean conjunction or disjunction), an ancilla is needed. If the unary operation is 2-reversible (like Boolean negation) an ancilla is needed too. Fig. 10 depicts pFIFDs representing Boolean inversion, exclusive or, conjunction, and disjunction. Only one ancilla is needed for numerous inversions within a single pFIFD. Contrary, every conjunction or disjunction requires its own additional ancilla, therefore the operations are of high cost.

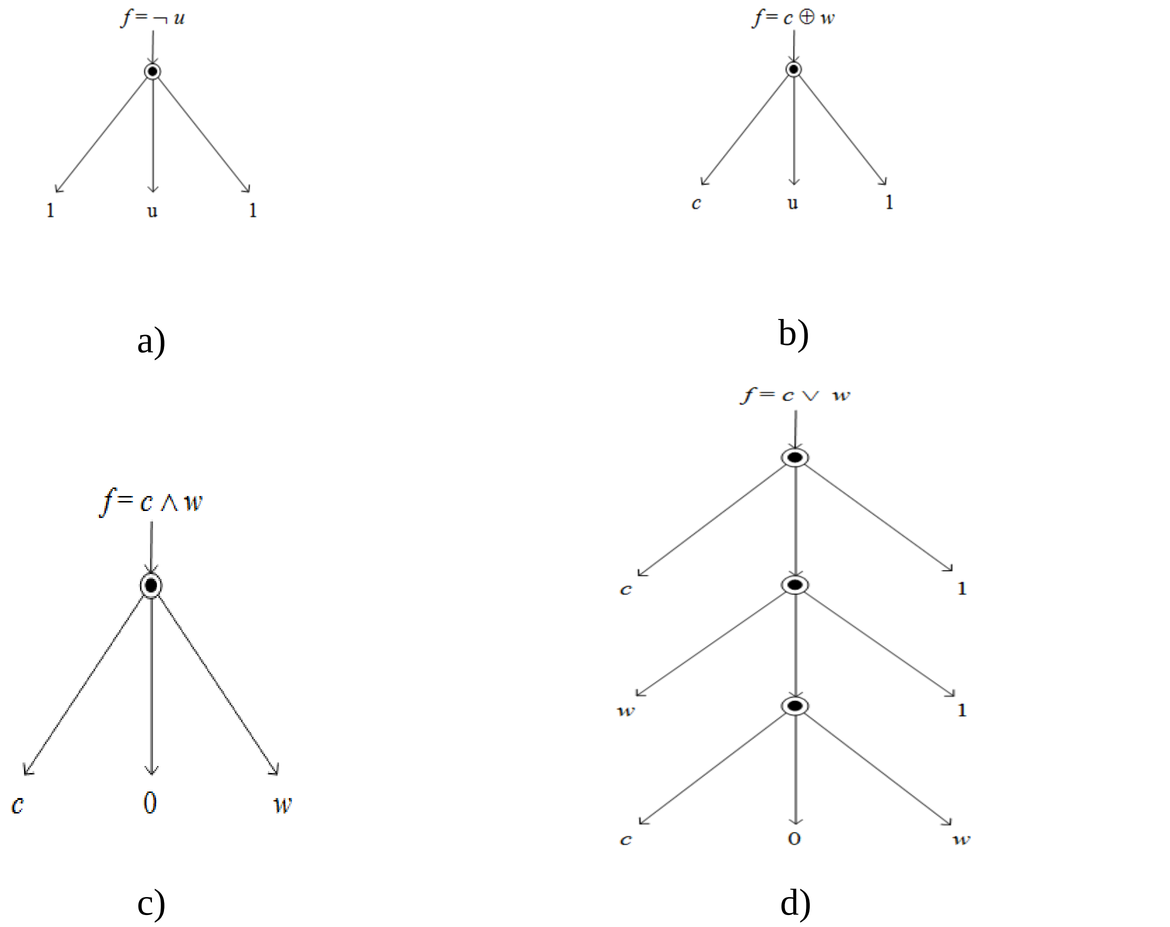

Figure 10 – Reversible pFIFDs of: a – inversion, b – exclusive or, c – conjunction; d – disjunction

Conclusion. The paper has introduced reversible if-decision diagrams as a model for synthesis, optimization and verification of logical quantum circuits. It has performed the analysis of reversibility of basic unary, binary, and ternary Boolean operations, and has shown that the binary exclusive-or and ternary xor-and-accumulation operations do not need ancillas. Any Boolean function can be represented by a superposition of the operations with or without ancillas. The operations allow the construction of reversible if-decision diagrams, which extend the functional decision diagrams.

References

1. Nielsen M., Chuang I. L. Quantum computation and quantum information. Cambridge: Cambridge University Press; 2000. – 700 p.

2. Sasanian Z., Wille R., Miller D. M. Realizing reversible circuits using a new class of quantum gates. In: Groeneveld P, editor. The 49th Annual design automation conference 2012; 2012 June 3–7; San Francisco, USA. New York: Association for Computing Machinery; 2012. – 36–41 p.

3. Smith, K. Technology-dependent quantum logic synthesis and compilation [dissertation] [Internet]. Dallas: Southern Methodist University.Available from: https://scholar.smu.edu/engineering _electrical _etds/30.

4. Prihozhy, A. A. Synthesis of quantum circuits based on incompletely specified functions and if-decision diagrams, Journal of the Belarusian State University. Mathematics and Informatics, 2021, Volume 3. – 84–97 p.

5. Prihozhy, A. A. Modelling reversible circuits by if-decision diagrams. VIII Международная научно-техническая интернет-конференция «Информационные технологии в образовании, науке и производстве», 21–22 ноября 2020 года [Электронный ресурс] / Белорусский национальный технический университет. – С. 163–168.

6. Zulehner A., Niemann P., Drechsler R., Wille R. Accuracy and Compactness in Decision Diagrams for Quantum Computation. Design, Automation and Test in Europe Conference – DATE, 2019. – 280–283 p.

7. Lukac M., Kameyama M., Perkowski M., Kerntopf P. Minimization of quantum circuits using quantum operator forms. arX-ib: 1701.01999.

8. Lee, C. Y. Representation of Switching Circuits by Binary-Decision Programs /C.Y. Lee //Bell Systems Technical Journal, 1959, Vol. 38, No 4. – 985–999 p.

9. Bryant, R. Graph-based algorithms for Boolean function manipulation / R. Bryant, // IEEE Trans. on Comp. 35. – 677–691 p.

10. Shannon, C. E. The Synthesis of Two Terminal Switching Circuits. Bell System Technical Journal. – Vol. 28. –1949. – № 1.– 59–98 p.

11. Прихожий, А. А. Частично определенные логические системы и алгоритмы / А.А. Прихожий // Минск, БНТУ. – 2013. – 343 с.

12. Прихожий, А. А. Обобщение разложения Шеннона для частично определенных функций: теория и применение. Системный анализ и прикладная информатика. – 2013, № 1–2. – С. 6–11.

13. Davio M., Deschamps J. P., Thayse A. Discrete and Switching Functions. New York: McGraw-Hill, 1978. – 729 p.

14. Прихожий, А. А. Новые разложения булевых функций по операции исключающее или в системах логического проектирования. Системный анализ и прикладная информатика. – 2014, № 1–3. – 9–16 с.

15. Prihozhy, A. A. If-Diagrams: Theory and Application. Proc. 7th Int. Workshop PATMOS'97. – UCL, Belgium, 1997. – 369–378 p.

16. Prihozhy A. A., Brancevich P. U. Parallel Computing with If-Decision-Diagrams Proc. Int. Conference PARELEC'98. – Poland, Technical University of Bialystok. – 1998. – 179–184 p.

17. Prihozhy, A. A. If-Decision Diagram Based Synthesis of Digital Circuits. Information Technologies for Education, Science and Business, Minsk, Belarus. –1999. – 65–69 p.

18. Prihozhy, A. A., Becker, B. If-Decision Diagram Based Modeling and Synthesis of Incompletely Specified Digital Systems. Electronics and communications, Special Issue on Electronics Design. – 2005. – 103–108 p.

19. Prihozhy, A. A. High-Level Synthesis through Transforming VHDL Models / A. A. Prihozhy // Chapter in Book "System-on-Chip Methodologies and Design Languages". – Kluwer Academic Publishers. – 2001. – 135–146 p.

20. Prihozhy, A. A. Synthesis of parallel adders from if-decision diagrams. System Analysis and Applied Information Science. – 2020. – No 2. – 61–70 p.