## ПРОГРЕСИВНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

## PROGRESSIVE INFORMATION TECHNOLOGIES

UDC 004.94 : 004.2

# ALGORITHMIC DIFFERENCES OF COMPLETE AND PARTIAL ALGEBRAIC SYNTHESIS OF A FINITE STATE MACHINE WITH DATAPATH OF TRANSITIONS

**Babakov R. M.** – Dr. Sc., Associate Professor, Professor of the Information Technologies Department, Vasyl' Stus Donetsk National University, Vinnytsia, Ukraine.

**Barkalov A. A.** – Dr. Sc., Professor, Professor of the Institute of Computer Science and Electronics, University of Zielona Gora, Zielona Gora, Poland.

**Titarenko L. A.** – Dr. Sc., Professor, Professor of the Institute of Computer Science and Electronics, University of Zielona Gora, Zielona Gora, Poland.

**Voitenko M. O.** – Student of the Faculty of Information and Applied Technologies, Vasyl' Stus Donetsk National University, Vinnytsia, Ukraine.

## ABSTRACT

**Context.** The problem of algorithmization the search for formal solutions of the problem of algebraic synthesis of a finite state ma-chine with datapath of transitions is considered. The concept of complete and partial solutions of this problem is proposed. The object of research is the automated synthesis of the finite state machine in the part of the function of transitions without taking into account the function of outputs. The basis of the algebraic implementation of the transition function is the author's approach to the transformation of state codes using a set of arithmetic and logical operations. The search for formal solutions to the problem of algebraic synthesis is a complex process that requires the use of appropriate methods and algorithms aimed at special coding of states and mapping of operations of transitions to individual state machine transitions. The use of partial solutions of the problem of algebraic synthesis can contribute to reducing the executing time of such algorithms and reducing the overall design time of digital control devices based on a finite state machine with an datapath of transitions.

**Objective.** Development and research of algorithms for finding complete and partial solutions to the problem of algebraic synthesis of a finite state machine with datapath of transitions.

**Method.** The research is based on the structure of a finite state machine with datapath of transitions. The synthesis of the circuit of the state machine involves the preliminary solution of the problem of algebraic synthesis. The result is the so-called formal solution of this problem, which contains two components. The first component is defined state codes, the second component is arithmetic and logic operations mapped to separate state machine transitions. Finite state machine can be synthesized in that case if transformation of given state codes in the process of making transitions is possible using a given set of operations. Verification of this possibility is carried out using the known matrix approach. It involves the formation and element-by-element mapping of two matrices – the matrix of transitions and the combined matrix of operations. As a result, a coverage matrix is formed, which reflects the possibility of implementing (covering) of state machine transitions by specified arithmetic and logic operations. Changing the way of encoding states or the set of operations can give different solutions to the problem of algebraic synthesis with a greater or lesser number of covered state machine transitions.

**Results.** Using the example of an abstract graph-scheme of the control algorithm, it is demonstrated that the solution of the problem of algebraic synthesis of a finite state machine with datapath of transitions can be considered a situation when one or more state machine transitions cannot be implemented using a given set of operations. It is proposed to call such situation as a partial solution of the problem of algebraic synthesis. The implementation of all transitions by specified operations gives a complete solution of this problem, but the number of complete solutions is always much smaller than the number of partial solutions. Therefore, in the general case, the search for complete solutions takes much more time and, moreover, is not always possible.

**Conclusions.** The design of a logical circuit of a finite state machine with datapath of transitions is possible in the case of a complete or partial solution of the problem of algebraic synthesis. In the case of a partial solution, those transitions that cannot be implemented by any of the available operations are implemented in a canonical way using a system of Boolean equations. The search for complete solutions generally takes more time than the search for partial solutions. This makes actual the development of algorithms and methods of synthesis of this class of finite state machine, based on the search for partial solutions of the problem of algebraic synthesis.

**KEYWORDS:** finite state machine, datapath of transitions, graph-scheme of algorithm, arithmetic and logical operations, algebraic synthesis, partial solution.

DOI 10.15588/1607-3274-2024-4-14

## ABBREVIATIONS

FSM is a finite state machine;
DT is a datapath of transitions;
GSA is a graph-scheme of algorithm;
TO is a transitions operation.

## NOMENCLATURE

$A$ is a sets of FSM states;
$X$ is a sets of logical conditions;
$Y$ is a sets of microoperations;
$M$ is a number of FSM states;
$L$ is a number of logical conditions analyzed by FSM;
$P$ is a number of microoperations formed by FSM;
$R$ is a bit capacity of state code;
$O$ is a set of transitions operations;
$O_i$ is an element of set $O$;
$I$ is a number of TO;
$B$ is a number of uncovered FSM transitions;
$B_{\min}$ is a minimal number of uncovered transitions;
$T$ is a code of FSM current state;
$D$ is a code of next state;
$G$ is a graph-scheme of algorithm.

## INTRODUCTION

Digital systems take a significant place in various spheres of human activity [1]. An integral component of a digital system is a control unit, the function of which is to coordinate the operation of all elements of the system [2, 3]. One of the ways of implementing a control unit is a finite state machine (FSM), in which a given control algorithm is implemented in the form of a logic circuit [4-6]. Compared to other types of control units, FSM is characterized by maximum speed due to the possibility of realizing multidirectional automatic transitions in one operation cycle. The disadvantage of FSM circuit is hardware expenses, which are maximum compared to other types of control units [2, 3, 7]. This worsens such characteristics of the FSM circuit as cost, energy consumption, dimensions, reliability, etc. [8, 9]. In this aspect, the scientific and practical problem of reducing hardware expenses in the FSM logic circuit is actual [3, 9, 10].

One of the well-known approaches to reducing hardware expenses in the FSM circuit is so-called operational transformation of state codes [11]. According to it, the transformation of state codes during state machine transitions is carried out not according to the system of canonical Boolean equations, but with using of a given set of arithmetic and logical operations (transitions operations, TO), which are performed on the code of the current state of the finite state machine. A set of such operations is implemented in the form of a set of corresponding combinational circuits, which form the so-called datapath of transitions (DT). The use of DT in the structure of FSM gives rise to the structure of a finite state machine with datapath of transitions (FSM with DT) [11]. The reduction of hardware expenses in the FSM with DT circuit compared to the canonical FSM is achieved due to the fact that as part of the DT, each combinational circuit is able to implement any number of FSM transitions with fixed hardware expenses for their implementation.

The synthesis of FSM with DT involves two main stages: algebraic synthesis and synthesis of the logic circuit of the FSM based on the results of algebraic synthesis [12]. Algebraic synthesis of the FSM is currently not sufficiently formalized and is not presented in the form of a method or an algorithmic and software implementation. This complicates the process of automated design and narrows the area of application of this FSM class.

In [13], it is shown that the algebraic synthesis of FSM with DT is reduced to the solution of the problem of algebraic synthesis. The search for solutions of this problem is proposed to be carried out using the comparison of special matrices, the first of which reflects the system of transitions of the finite state machine with a given coding of states, and the second reflects the possibilities of converting state codes using a given set of arithmetic and logic operations. The rules for constructing and comparing such matrices are considered in [13].

**The object of the study** is the algebraic synthesis of a finite state machine with datapath of transitions.

Algebraic synthesis is considered completed under two conditions: if all states of the FSM are coded; if for each FSM transition, the transformation of the current state code into the transition state code can be performed using one of the specified transitions operations.

**The subject of the study** are algorithms for finding complete and partial solutions of the problem of algebraic synthesis of FSM with DT.

**The purpose of the work** is the development and comparative research of algorithms of finding complete and partial solutions of the problem of algebraic synthesis of finite state machine with datapath of transitions.

## 1 PROBLEM STATEMENT

Let the finite state machine with datapath of transitions be given in the form of a graph-scheme of the algorithm (GSA) [3, 10]. According to this GSA, a set of states $A=\{a_1, ..., a_M\}$, a set of input signals $X=\{x_1, ..., x_L\}$ and a set of microoperations $Y=\{y_1, ..., y_P\}$ are formed. The set of transitions operations $O = \{O_1, …, O_I\}$ is also given. Each element $O_i \in O$ represents a certain arithmetic or logical operation, which provides an appropriate interpretation of the binary codes of the FSM states. The GSA and the set $O$ are input data for the algebraic synthesis of FSM with DT.

This article solves the problem of developing algorithms for obtaining complete and partial solutions of problems of algebraic synthesis of FSM with DT for given GSA and a set of transition operations.

## 2 REVIEW OF THE LITERATURE

Known methods of optimizing the logic circuit of a finite state machine usually lead to changes in its structure [3, 6, 10]. This article considers the structure of finite state machine with datapath of transitions [11]. In it, the

DOI 10.15588/1607-3274-2024-4-14

transition function is implemented on the basis of the datapath of transitions, which provides a set of arithmetic or logical operations for the transformation of state codes. The synthesis of FSM with DT is considered in [12].

Let the FSM with DT be given by GSA $G$ (Fig. 1). The operation vertices of this GSA do not contain microoperations and are shown empty. This is done in order to focus precisely on the function of transitions of the FSM, and not on the function of outputs. The output function in FSM with DT is implemented in the same way as in FSM with a canonical structure [2–5, 10] and is not considered in this paper.

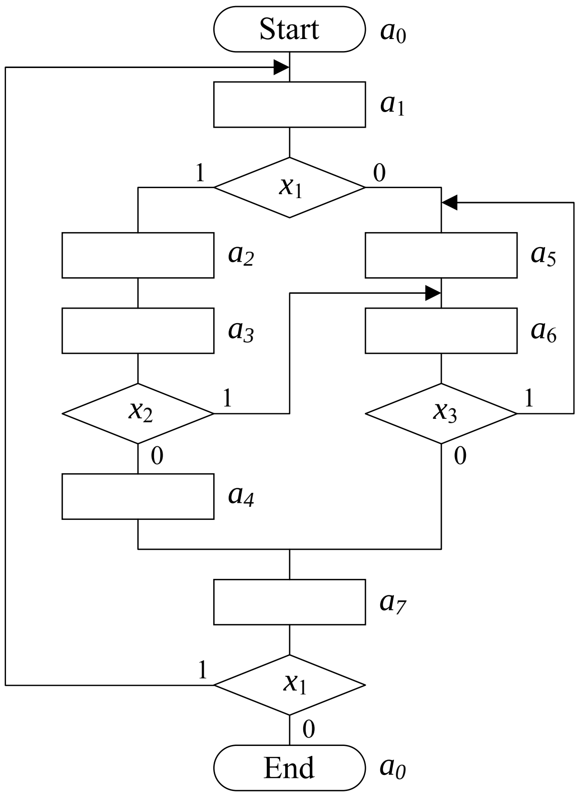

Figure 1 – Graph-scheme of algorithm $G$

GSA $G$ contains 8 states of the Moore state machine [10] $a_0 - a_7$, for coding of which $R = 3$ binary digits are sufficient. GSA contains both conditional and unconditional FSM transitions, the total number of which is 12.

Let the set of TO $O = \{O_1, O_2, O_3\}$ be also given, where

$$O_1: D = T + 1_{10}; \quad (1)$$

$$O_2: D = T \vee 010_2; \quad (2)$$

$$O_3: D = T \oplus 110_2. \quad (3)$$

In expressions (1)–(3), the symbol $T$ means code of current state, the symbol $D$ means code of transition state. TO $O_1$ represents the addition of one, TO $O_2$ is a disjunction with binary constant 010, TO $O_3$ is XOR operation with binary constant 110.

The result of any TO is code of transition state of the FSM. This means that result of TO must always have the same bit size as the argument of TO (current state code). This bit size in the article is denoted by the symbol $R$ and for GSA $G$ is equal to 3. If TO $O_2$ and $O_3$ are bitwise logical operations, TO $O_1$ is an arithmetic operation (increment). If result of the increment is one digit greater than the argument, $R$ lower digits should be taken as the result of the TO. Therefore, it is more correct to write TO $O_1$ as follows:

$$O_1: D = (T + 1_{10}) \bmod 8. \quad (4)$$

Here, *mod* means *modulo* – the operation of taking the remainder of a devision the value ($T$ + $1_{10}$) by 8, where the number 8 is calculated as $2^R$. This is equivalent to taking three lower digits from the increment result.

All arithmetic and logical operations should be arranged in a similar way. For example, if the logical operation of a cyclic bit shift is used, the shift must be carried out within $R$ bits, that is, within the bit size of the state code of the FSM. If the arithmetic operation of adding the decimal constant 5 is used, its result is also taken modulo $2^R = 8$. That is, $(6 + 5) \bmod 8 = 3$.

Now consider the algebraic synthesis of FSM with DT. It consists in the coordinated execution of the following actions [12]:

1) each state of the FSM is assigned an unique code from a set of admissible state codes (usually from the range of integers $[0; 2^R–1]$);

2) each FSM transition is mapped to a certain operation from the set $O$ (the same operation can be mapped to several transitions).

The result of these actions should be the operational implementation of all or the vast majority of state machine transitions. A transition has an operational implementation if the code of the initial state is transformed into the code of the state of the transition using the TO mapped to this transition. If none of the given TOs allows such a transformation, the transition has no operational implementation with the selected states coding and the given set of TOs.

In Fig. 2 an example of operational implementation of all transitions of GSA $G$ for the set of TOs formed by operations (1)–(3) is shown.

In Fig. 2, the decimal code of state and its binary equivalent are recorded in each vertex marked by the state of the Moore FSM. Decimal codes should be considered when the transition to this state or from this state is carried out using an arithmetic TO. Binary codes are required in the case of using logical TO. Also in Fig. 2, each transition branch is marked by an operation mapped at this transition. For ease of understanding, TO $O_1$ is marked with “+1”, TO $O_2$ – with “$\vee$ 010”, TO $O_3$ – with “$\oplus$ 110”.

It can be noted that in Fig. 2, each FSM transition is implemented (covered) by one of the given TOs. For example, a conditional transition from state $a_7$ with code $3_{10} = 011_2$ to state $a_1$ with code $5_{10} = 101_2$ is covered by the operation “$\oplus$ 110”, since $011_2 \oplus 110_2 = 101_2$.

© Babakov R. M., Barkalov A. A., Titarenko L. A., Voitenko M. O., 2024
DOI 10.15588/1607-3274-2024-4-14

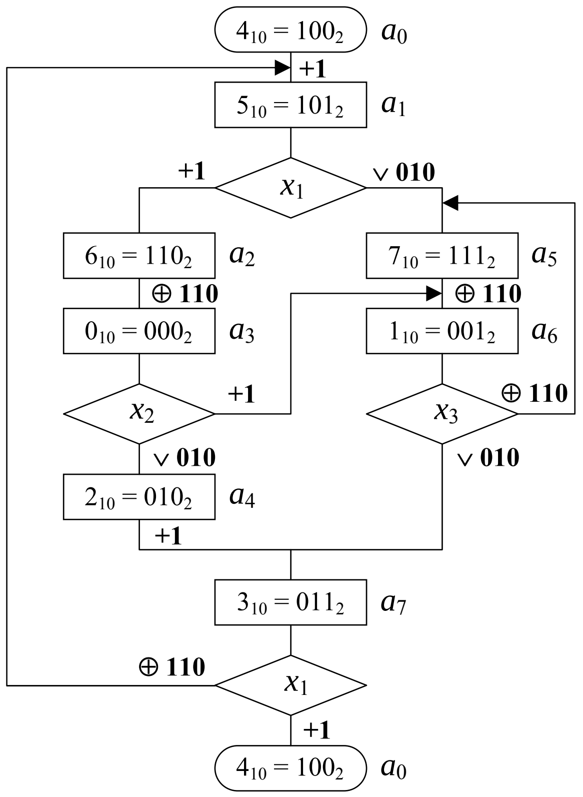

Figure 2 – Example of operational implementation of all transitions of GSA *G*

From Fig. 2, you can see that each TO is used to implement several FSM transitions. However, as part of the datapath of transitions, the circuit of each TO has fixed structure and fixed hardware expenses, which do not depend on the number of FSM transitions covered by this TO [11, 12]. For a large-sized GSA, the incrementer circuit can implement dozens and hundreds of transitions with fixed hardware expenses for their implementation. This is the basis of the well-known class of counter-based finite state machines [2]. FSM with DT is positioned as a generalization of an FSM with counter for the case of using a set of different arithmetic and logical operations. It is the possibility of implementing many FSM transitions with the help of a fixed set of TOs with fixed hardware expenses that ensures the saving of resources for the implementation the transition function of the FSM with DT compared to the canonical FSM [11].

If the states are coded in a different way or a different set of TOs is specified, part of the FSM transitions may remain uncovered. Consider a fragment of Fig. 2, in which codes of states of $a_2$ and $a_3$ are swapped (Fig. 3).

On this fragment, transitions $a_1 \to a_2$, $a_3 \to a_4$ and $a_3 \to a_6$ cannot be realized by any of the TOs $O_1 - O_3$. At the same time, the transition $a_2 \to a_3$ can be implemented using TO $O_3$. So, changing the codes of the two states added three FSM transitions, not covered by any of the given TOs.

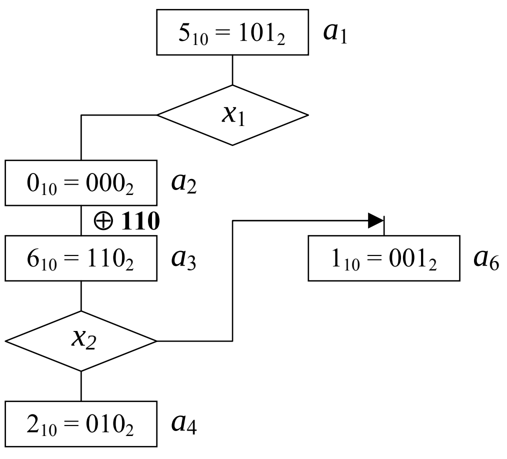

Figure 3 – Fragment of GSA *G* with uncovered state machine transitions

After carrying out the algebraic synthesis of FSM with DT, there is a stage of synthesis of the logical circuit of the FSM. This stage, in contrast to algebraic synthesis, is sufficiently formalized and is discussed in [12].

In [13], the situation when all FSM transitions are covered by given TOs is called a "formal solution to the problem of algebraic synthesis of FSM with DT". The term "formal" here means that in the presence of such a solution, the functioning of the FSM is possible according to the principle of operational transformation of state codes and the synthesis of the logical circuit of the FSM is possible. The formal solution does not guarantee a reduction in the hardware expenses in the FSM circuit, but only ensures the functioning of the FSM based on the datapath of transitions.

The examples given in [13] demonstrate the possibility of the existence of a set of formal solutions of the problem of algebraic synthesis for a given GSA. Also in [13] the concepts of effective and optimal solutions are introduced. These solutions form proper subsets on the set of formal solutions. They are interesting in that their use makes it possible to obtain a gain in hardware expenses in the resulting circuit of the FSM with DT compared to other FSM structures.

It should be noted that the synthesis of the logical circuit of FSM with DT is possible even if part of FSM transitions remains uncovered by existing transitions operations [14]. In this case, all transitions not covered by TOs are implemented separately from other transitions according to the same principle as in the transition circuit of canonical FSM [2, 3, 10]. For them, a system of Boolean equations of the form (5) is formed, for which a corresponding combinational circuit is synthesized.

$$D = D(T, X). \tag{5}$$

In expression (5), the transition function $D$ depends on the current state $T$ and input signals $X$. The number of system equations is equal to $R$ (bit size of the state code).

The combinational circuit synthesized in this way is considered as one of the operations of set $O$. It is added to the datapath of transitions and receives a separate code, according to which its output is multiplexed with the outputs of other OPs circuits. For the considered example, the structure of the datapath is shown in Fig. 4. Blocks $O_1 – O_3$ correspond to transitions operations from the set $O$. Block $O_4$ contains a combinational circuit constructed according to equation (5). It can be considered that the $O_4$ block implements a non-standard logical TO, which is performed on the code of the current state $T$ and the input signals $X$. The synthesis of such block is discussed in detail in [14]. In Fig. 4, the multiplexer MUX under the control of the signals of the TO code $W$ forms the code of the next state $D$, which enters the memory register of the FSM (not shown in the figure).

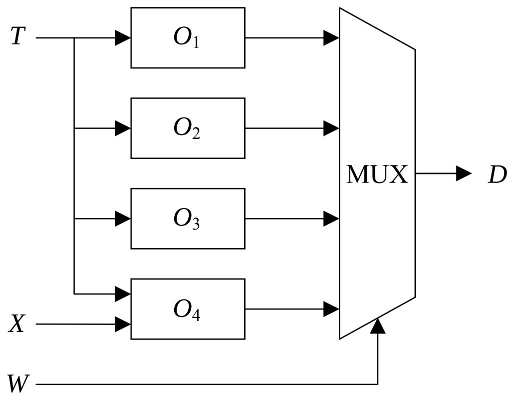

Figure 4 – Structure of datapath of transitions in case of canonical realization of several transitions

## 3 MATERIALS AND METHODS

According to the results of algebraic synthesis, two situations are possible:

1. All FSM transitions are covered by the specified TOs.

2. Part of FSM transitions remained not covered by the given TOs and should be implemented in a canonical way according to the system of Boolean equations.

Let us clarify the meaning of a formal solution of the problem of algebraic synthesis of FSM with DT, given in [13]. We will consider the result of algebraic synthesis to be a formal solution, regardless of the number of covered transitions. To distinguish between full and partial coverage of transitions, we introduce the following definitions.

A solution in which all FSM transitions are covered by a given set of transitions operations will be called a *complete solution of the problem of algebraic synthesis of FSM with DT.*

*A partial solution of the problem of algebraic synthesis of FSM with DT* will be called a solution in which part of the transitions remains uncovered by a given set of TOs.

In a partial solution, the number of uncovered transitions can vary from 1 to the total number of all transitions of a given FSM. In the extreme case, if all transitions remain uncovered, the FSM with DT degenerates into a canonical FSM with the according structure [3, 10].

The purpose of differentiating complete and partial solutions is as follows.

Currently, there are no effective algorithms for finding complete solutions to the problem of algebraic synthesis of FSM with DT. It is not obvious what codes should be assigned to the FSM states so that all transitions be covered by a given set of operations. With a fixed set of TOs, the search for formal solutions can be reduced to enumeration of variants for coding states with a set of admissible codes. For each encoding variant, a transition coverage check must be performed.

During the enumeration of variants of states encoding, the complete solution can be found at the beginning of the enumeration, at the very end of the enumeration, or not found at all. At the same time, going through all possible options for coding states requires considerable time. If the GSA contains $M$ states, for the coding of which $R$ binary bits are enough, then the number of state coding variants $N$ is determined by expression (6).

$$N = \frac{(2^R)!}{(2^R - M)!}. \qquad (6)$$

For our example, with $R = 3$, $M = 8$, we have $N = 40320$ (taking into account $0! = 1$). However, for FSM with 10 states, we will get $N = 29 \cdot 10^9$ coding variants. For GSA with $M > 16$, the number of state coding variants does not allow considering all variants within reasonable time limits.

We will proceed from the assumption that all complete solutions will lead to the same or similar hardware expenses in the FSM circuit. Therefore, it is sufficient to find only one complete solution, after which the enumeration of state coding variants can be stopped. This allows us to propose an algorithm for finding a complete solution to the problem of algebraic synthesis of FSM with DT, the flowchart of which is shown in Fig. 5.

The peculiarity of this algorithm is that the search for a solution takes place until the first solution is found. At the same time, the algorithm does not guarantee finding a complete solution, since a complete solution may not exist for a given GSA and a given set of TOs. The following statements should be considered valid:

1. The probability of finding a complete solution increases with an increase in the number of transitions operations used.

2. The probability of finding a complete solution decreases with an increase in the number of FSM states with a fixed set of TO.

It follows from the first statement that the smaller the number of TOs used, the lower the probability of finding a complete solution. However, the economy of hardware expenses in the circuit of FSM with DT is achieved precisely due to the reduction of the number of TOs and the simplification of the circuit of datapath of transitions. Therefore, the desire to reduce the number of used TOs contributes to the failure to find complete solutions.

It follows from the second statement that increasing the complexity of the control algorithm implemented by FSM also does not contribute to finding complete solutions. The complexity of the algorithm in this case is

DOI 10.15588/1607-3274-2024-4-14

measured by the number of its states, which affects the bit size $R$ of state code. Therefore, the more complex the FSM is, the more difficult it is to cover all its transitions with a given set of TOs, since the number of state coding variants increases, according to (5), in a non-linear dependence.

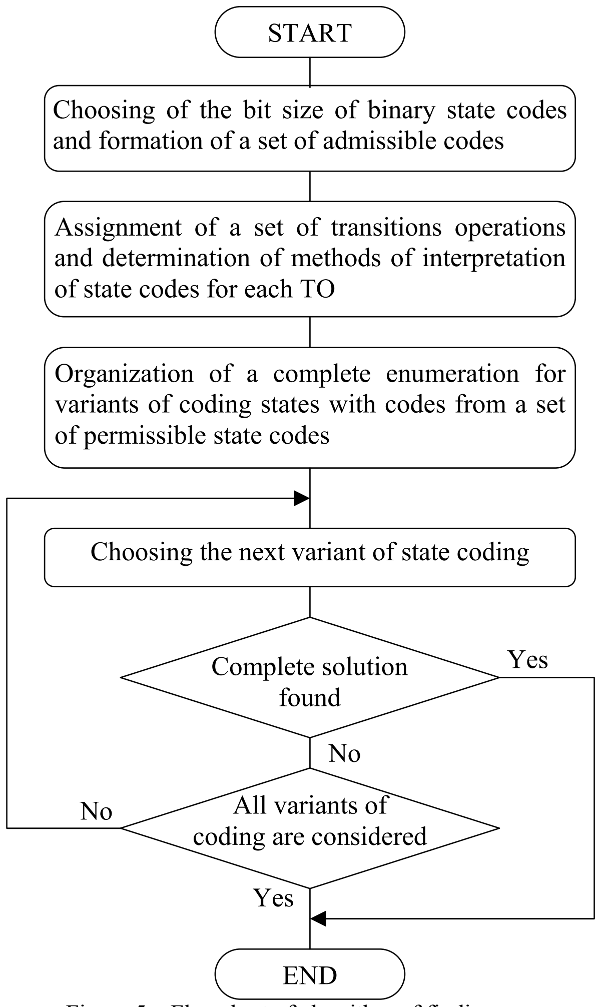
START
Choosing of the bit size of binary state codes and formation of a set of admissible codes
Assignment of a set of transitions operations and determination of methods of interpretation of state codes for each TO
Organization of a complete enumeration for variants of coding states with codes from a set of permissible state codes
Choosing the next variant of state coding
Complete solution found
Yes
No
All variants of coding are considered
No
Yes
END

Figure 5 – Flowchart of algorithm of finding a complete solution of problem of algebraic synthesis for given set of transitions operations

Thus, the desire to reduce the number of TOs and the increasing complexity of control algorithms decrease the probability and increase the time of finding complete solutions to the problem of algebraic synthesis. The algorithm shown in Fig. 5 should be used only if, for some reason, it is inadmissible to have uncovered transitions and to implement them in a canonical way using a system of Boolean equations. The following can be considered as such reasons:

– the task was set to reduce hardware expenses in the FSM with DT to the minimum possible level;

– a pre-synthesized device is used as an DT, which cannot be changed.

If the algorithm for finding a complete solution does not find a result, the algebraic synthesis of FSM with DT under the given conditions becomes impossible. To eliminate this shortcoming, an algorithm for finding a partial solution is proposed, the generalized flowchart of which is shown in Fig. 6.

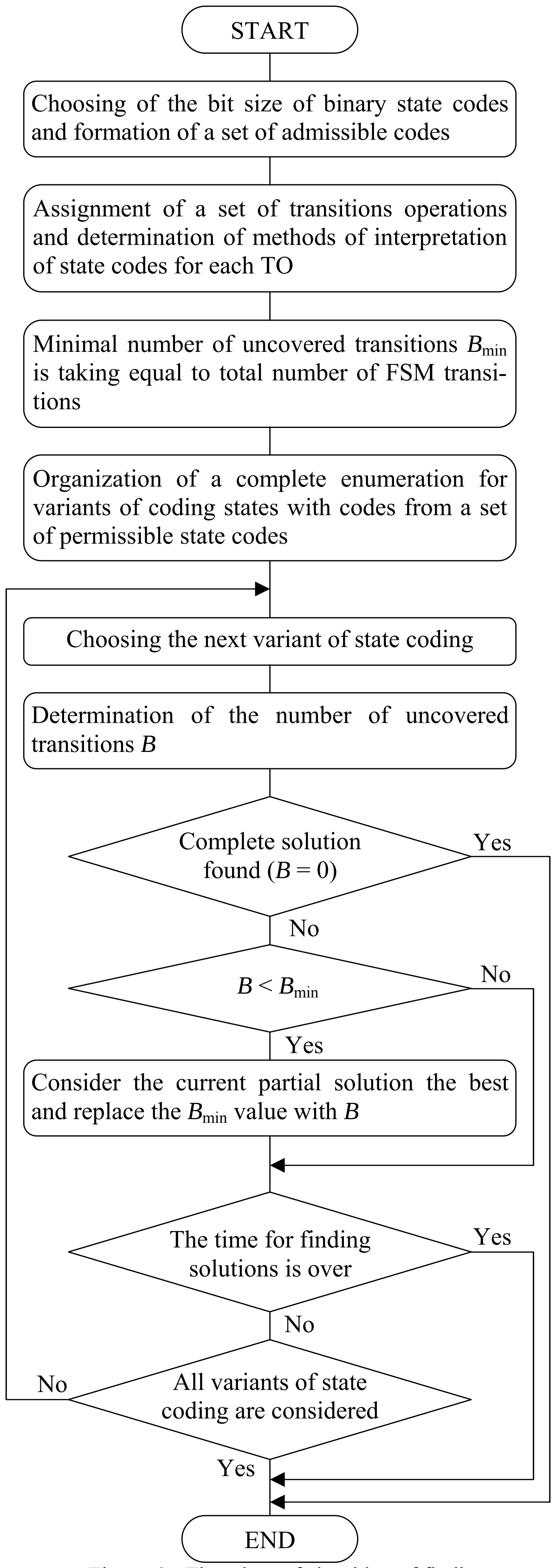
START
Choosing of the bit size of binary state codes and formation of a set of admissible codes
Assignment of a set of transitions operations and determination of methods of interpretation of state codes for each TO
Minimal number of uncovered transitions $B_{\min}$ is taking equal to total number of FSM transitions
Organization of a complete enumeration for variants of coding states with codes from a set of permissible state codes
Choosing the next variant of state coding
Determination of the number of uncovered transitions $B$
Complete solution found ($B = 0$)
Yes
No
$B < B_{\min}$
No
Yes
Consider the current partial solution the best and replace the $B_{\min}$ value with $B$
The time for finding solutions is over
Yes
No
All variants of state coding are considered
No
Yes
END

Figure 6 – Flowchart of algorithm of finding a partial solution of problem of algebraic synthesis

The differences between this algorithm and the algorithm for finding a complete solution are as follows.

1. This algorithm always gives a result sufficient for the synthesis of the logic circuit of the FSM. This result can be either a complete solution or a partial solution, but it will always be found. The algorithm for finding a complete solution (Fig. 5) may not give a result at all, which will make algebraic and further synthesis of the FSM impossible.

2. If the time constraints are set, the algorithm in the given time finds among the considered partial solutions one that has the minimum number of uncovered transitions. This makes it possible to maximally simplify the circuit that corresponds to the canonical implementation of uncovered transitions.

3. If time constraints are not specified, the algorithm after enumeration through all possible states coding variants gives the best possible result for a given GSA with a given set of transitions operations. It can be either a complete or a partial solution, which can be considered optimal under the given conditions of algebraic synthesis.

As in the previous algorithm, when developing the algorithm for finding partial solutions, the assumption was made that all complete solutions that exist for a given GSA and set of TOs lead to the same or close values of hardware expenses in the circuit of FSM. So, in the case of finding a complete solution, the algorithm finishes its work, and the found result is considered the best among all possible ones.

## 4 EXPERIMENTS

As part of experimental research, the proposed algorithm for finding partial solutions to the problem of algebraic synthesis of FSM with DT (Fig. 6) was implemented as a Python program. The implementation of the algorithm is universal and can be applied to any GSA specified by a file in the KISS format [15]. The purpose of the experiments was to reveal the ability of the algorithm to find partial solutions with the minimum possible number of uncovered transitions. The course of the experiments was as follows.

1. The FSM is given by GSA *G* (Fig. 1).

2. The program implements an enumeration of all possible variants of coding states with three-bit binary codes. In the process of enumeration, the set of TOs remained fixed. When finding a complete solution, the program continues the further enumeration of variants, and does not interrupt the search, as the algorithm predicts.

3. To determine the number of uncovered transitions, the matrix approach proposed in [10] is used. For fast processing of matrices, the standard Numpy library of the Python language is used.

4. Since GSA *G* contains only 12 transitions, the program counts not only the number of complete solutions, but also the number of partial solutions that contain one or two uncovered transitions. Such solutions for small-sized GSA can be considered as close as possible to a complete solution. The number of found complete solutions is also counted.

5. The experiment was repeated for different sets of transition operations.

Each experiment (each enumeration of all variants of states encoding for a certain set of TOs) lasted, in the case of GSA *G*, about one second. During this time, according to (5), 40320 state coding variants were enumerated.

## 5 RESULTS

Table 1 contains the results of research on the number of complete and partial solutions for GSA *G* and three TOs. Each row of the table 1 corresponds to the results of one conducted experiment. The table columns contain the following data:

1. The “Transitions operations” column shows three transitions operations used in each of the experiments. They are artificially selected so that there is at least one complete solution for them. Operations have designations similar to the designations in Fig. 2. For example, the mark “$\vee 110_2$” means the disjunction operation of the current state code with the binary constant 110.

2. In the “0” column the number of complete solutions found (i.e., solutions that have zero non-covered transitions) is indicated.

3. Columns “–1” and “–2” indicate the number of found partial solutions that have one or two non-covered transitions, respectively.

Table 1 – Number of complete and partial solutions for GSA *G* with using three TOs

| Transitions operations | | | 0 | –1 | –2 |
|---|---|---|---|---|---|
| + 1 | + 2 | $\times 2$ | 1 | 18 | 80 |
| + 1 | + 2 | $\times 4$ | 1 | 17 | 58 |
| + 1 | + 2 | + 4 | 8 | 16 | 184 |
| + 1 | + 2 | $\oplus 101_2$ | 8 | 56 | 200 |
| + 1 | + 2 | $\oplus 011_2$ | 4 | 20 | 122 |
| + 1 | + 2 | $\oplus 110_2$ | 4 | 4 | 58 |
| + 1 | + 2 | $\vee 111_2$ | 1 | 16 | 50 |
| + 1 | + 2 | $\vee 110_2$ | 2 | 10 | 41 |
| + 1 | + 2 | $\& 001_2$ | 2 | 10 | 41 |
| + 1 | + 2 | $\& 100_2$ | 2 | 12 | 70 |
| + 1 | + 2 | $\div 2$ | 2 | 19 | 109 |
| + 1 | + 2 | $\div 4$ | 1 | 19 | 78 |
| + 5 | $\vee 010_2$ | $\oplus 110_2$ | 4 | 4 | 58 |
| + 1 | $\vee 010_2$ | $\oplus 110_2$ | 4 | 4 | 58 |
| + 3 | + 6 | $\vee 010_2$ | 4 | 4 | 58 |
| + 5 | $\oplus 010_2$ | $\oplus 110_2$ | 8 | 8 | 160 |

Content of Table 1 demonstrates that in most cases the number of partial solutions with one uncovered transition exceeds the number of complete solutions. The number of partial solutions with two uncovered transitions in the conducted experiment always exceeds the number of solutions with one uncovered transition.

The fact that the values in the “–2” column exceed the values in the “–1” and “0” columns allows us to expect that during a full search, a partial solution with two uncovered transitions will be found earlier (and in less time) than solution with one uncovered transition or a complete solution. If, with two uncovered transitions, the resulting circuit of the FSM still has a satisfactory gain in hardware expenses, the search for such partial solutions can significantly speed up the process of algebraic synthesis.

Additional experiments showed that the number of solutions with three uncovered transitions is several times greater than the value in the “–2” column. Therefore, the search for less efficient solutions takes less time on average than the search for more efficient solutions. This is relevant in cases of GSA of large size, for which a complete enumeration of state coding variants is impossible within acceptable time limits.

Note that the third line from the bottom of the table shows 4 found complete solutions, one of which is shown as an example in Fig. 2.

The purpose of the following experiment is to determine the number of partial solutions with one and two uncovered transitions, provided that no complete solution exists for a given set of TOs (column "0" contains zero values). The results of the experiment are shown in the Table 2, the structure of which is similar to Table 1.

Table 2 – Number of partial solutions for GSA *G* in the absence of complete solutions

| Transitions operations | | | 0 | –1 | –2 |
|---|---|---|---|---|---|
| +1 | +2 | +3 | 0 | 40 | 312 |
| + 2 | + 3 | + 4 | 0 | 24 | 184 |
| + 2 | + 3 | × 2 | 0 | 0 | 25 |
| + 1 | × 4 | ÷ 2 | 0 | 0 | 26 |
| + 1 | $\oplus\ 011_2$ | ÷ 2 | 0 | 1 | 11 |
| + 2 | $\vee\ 011_2$ | ÷ 2 | 0 | 1 | 7 |
| + 1 | $\&\ 011_2$ | $\oplus\ 101_2$ | 0 | 3 | 28 |
| + 3 | × 2 | $\oplus\ 110_2$ | 0 | 1 | 35 |
| $\vee\ 001_2$ | $\&\ 010_2$ | $\oplus\ 011_2$ | 0 | 0 | 4 |
| $\oplus\ 101_2$ | $\oplus\ 110_2$ | $\oplus\ 111_2$ | 0 | 0 | 96 |
| + 1 | + 5 | $\oplus\ 100_2$ | 0 | 128 | 320 |
| × 4 | ÷ 2 | $\oplus\ 110_2$ | 0 | 2 | 12 |
| × 2 | ÷ 4 | $\oplus\ 011_2$ | 0 | 2 | 12 |
| + 2 | ÷ 2 | $\oplus\ 011_2$ | 0 | 2 | 54 |
| + 3 | $\&\ 101_2$ | $\vee\ 110_2$ | 0 | 2 | 18 |
| +2 | $\oplus\ 011_2$ | $\vee\ 100_2$ | 0 | 2 | 20 |

Content of Table 2 demonstrates for each of the considered sets of operations the existence of partial solutions in the absence of complete solutions. This allows for the synthesis of FSM with DT, provided that the partial solution leads to a decrease in hardware expenses in the FSM circuit.

## 6 DISCUSSION

The proposed algorithms for finding complete and partial solutions to the problem of algebraic synthesis of FSM with DT are based on the assumption that the set of transitions operations is given and fixed. Even under this condition, the number of variants of states encoding for GSA of medium size is too large for the practical application of algorithms. Reducing the time to perform enumerations of variants is possible thanks to the application of methods and algorithms of partial enumeration, as well as due to the parallelization of the search process with an orientation to multiprocessor systems.

Meanwhile, fixing the set of TOs in the general case is not mandatory. Nothing prevents you from making an "external" enumeration of various TOs, inside which there will be an "internal" enumeration of states coding variants. This, on the one hand, will increase the total number of iterations of the algorithm, and on the other hand, it can contribute to faster finding of effective solutions.

Algorithms for the rational formation of a set of TOs can be developed instead of enumeration of TOs. In this aspect, the algorithms proposed in this article will allow, on the example of small GSAs, to investigate the effectiveness of the application of certain TOs, spreading the obtained results to large GSAs.

## CONCLUSIONS

The article proposes a solution to the scientific problem of developing algorithms for finding complete and partial solutions to the problem of algebraic synthesis of a finite state machine with datapath of transitions. The conducted research is a component of a broader scientific problem devoted to the design automatization of this class of finite state machines.

**The scientific novelty** of the article lies in the fact that for the first time algorithms for finding complete and partial solutions to the problem of algebraic synthesis have been developed, implemented and researched. Research of test GSA using the developed algorithms showed that for various sets of transitions operations, the number of partial solutions significantly exceeds the number of complete solutions. This emphasizes the relevance of the development of algorithms for the search of partial solutions using a shortened enumeration of state coding variants.

**The practical use** of obtained results is possible in the development of methods and algorithms of synthesis of FSM with DT within the framework of specialized CADs of digital control devices.

**Prospects for further research** consist in solving a range of scientific and practical problems related to the optimization of work time of proposed algorithms due to development of methods for shorted enumerations of variants of coding FSM states. This will make it possible to apply algorithms and carry out algebraic synthesis of FSM with DT for large-sized GSA, as well as automatically generate VHDL code for synthesizing FSM circuit in modern elementary bases [2, 5, 8, 9, 16].

## ACKNOWLEDGEMENTS

The work is supported by the state budget scientific research project of Vasyl' Stus Donetsk National University "Methods, algorithms and tools of computer-aided design of control units of computing systems" (state registration number 0122U200085).

## REFERENCES

1. Bailliul J., Samad T. Encyclopedia of Systems and Control. Springer, London, UK, 2015, 1554 p. DOI: https://doi.org/10.1007/978-1-4471-5058-9
2. Czerwinski R., Kania D. Finite state machines logic synthesis for complex programmable logic devices. Berlin, Springer, 2013, 172 p. DOI: https://doi.org/10.1007/978-3-642-36166-1
3. Baranov, S. Logic and System Design of Digital Systems. Tallin, TUTPress, 2008, 267 p.
4. Micheli G. D. Synthesis and Optimization of Digital Circuits. McGraw-Hill, Cambridge, MA, USA, 1994, 579 p.
5. Minns P., Elliot I. FSM-Based Digital Design Using Verilog HDL. JohnWiley and Sons, Hoboken, NJ, USA, 2008, 408 p. DOI: https://doi.org/ 10.1002/9780470987629
6. Skliarova I., Sklyarov V., Sudnitson A. Design of FPGA-based circuits using hierarchical finite state machines. Tallinn, TUT Press, 2012, 240 p.
7. Klimovich A. S., Solov'ev V. V. Minimization of mealy finite-state machines by internal states gluing, *Journal of Computer and Systems Science International*, 2012, Volume 51, pp. 244–255. DOI: https://doi.org/10.1134/S1064230712010091

8. Grout, I. Digital Systems Design with FPGAs and CPLDs / I. Grout. – Elsevier Science: Amsterdam, The Netherlands, 2008. – 784 p. DOI: https://doi.org/10.1016/B978-0-7506-8397-5.X0001-3
9. Kubica M., Opara A. and Kania D. Technology Mapping for LUT-Based FPGA, *Lecture Notes in Electrical Engineering,* Springer, Cham, 2021, Vol. 713, 207 p. DOI: https://doi.org/10.1007/978-3-030-60488-2
10. Baranov S. Logic Synthesis for Control Automata. Dordrecht, Kluwer Academic Publishers, 1994, 312 p.
11. Barkalov A. A., Babakov R. M. Operational formation of state codes in microprogram automata, *Cybernetics and Systems Analysis*, 2011, Volume 47 (2), pp. 193–197. DOI: https://doi.org/10.1007/s10559-011-9301-y
12. Barkalov A. A., Titarenko L. A., Babakov R. M. Synthesis of Finite State Machine with Datapath of Transitions According to the Operational Table of Transitions, *Radio Electronics, Computer Science, Control,* 2022, No. 3 (62), pp. 109–119. DOI: https://doi.org/10.15588/1607-3274-2022-3-11
13. Barkalov A. A., Babakov R. M. A Matrix Method for Detecting Formal Solutions to the Problem of Algebraic Synthesis of a Finite-State Machine with a Datapath of Transitions, *Cybernetics and Systems Analysis,* 2023, Volume 59 (2), pp. 190–198. DOI: https://doi.org/10.1007/s10559-023-00554-6
14. Barkalov A. A., Titarenko L. A., Babakov R. M. Synthesis of VHDL-Model of a Finite State Machine with Datapath of Transitions, *Radio Electronics, Computer Science, Control,* 2023, No. 4 (67), pp. 135–147. DOI: https://doi.org/10.15588/1607-3274-2023-4-13
15. Yang S. Logic synthesis and optimization benchmarks User Guide Version 3.0. Tech. rep. Microelectronics Center of North Carolina, P.O. Box 12889, Research Triangle Park, NC 27709, 1991.
16. Rawski M., Selvaraj H., Luba T. An application of functional decomposition in ROM-based FSM implementation in FPGA devices, *Journal of System Architecture*, 2005, Volume 51, pp. 424–434. DOI: https://doi.org/10.1016/j.sysarc.2004.07.004

Received 05.09.2024.
Accepted 07.11.2024.

УДК 004.94 : 004.2

# АЛГОРИТМІЧНІ ВІДМІННОСТІ ПОВНОГО І ЧАСТКОВОГО АЛГЕБРАЇЧНОГО СИНТЕЗУ МІКРОПРОГРАМНОГО АВТОМАТА З ОПЕРАЦІЙНИМ АВТОМАТОМ ПЕРЕХОДІВ

**Бабаков Р. М.** – д-р техн. наук, доцент, доцент кафедри інформаційних технологій Донецького національного університету імені Василя Стуса, м. Вінниця, Україна.

**Баркалов О. О.** – д-р техн. наук, професор, професор Інституту комп'ютерних наук та електроніки університету Зеленогурського, м. Зельона Гура, Польща.

**Тітаренко Л. О.** – д-р техн. наук, професор, професор Інституту комп'ютерних наук та електроніки університету Зеленогурського, м. Зельона Гура, Польща.

**Войтенко М.О.** – студент факультету інформаційних і прикладних технологій Донецького національного університету імені Василя Стуса, м. Вінниця, Україна.

## АНОТАЦІЯ

**Актуальність.** Розглянуто задачу алгоритмізації пошуку формальних розв'язків задачі алгебраїчного синтезу мікропрограмного автомата з операційним автоматом переходів. Запропоновано поняття повного та часткового розв'язків цієї задачі. Об'єктом дослідження є автоматизований синтез автомата в частині функції переходів без урахування функції виходів. В основі алгебраїчної реалізації функції переходів знаходиться авторський підхід до перетворення кодів станів за допомогою множини арифметико-логічних операцій. Пошук формальних розв'язків задачі алгебраїчного синтезу є складним процесом, що потребує використання відповідних методів і алгоритмів, спрямованих на спеціальне кодування станів та зіставлення операцій переходів окремим автоматним переходам. Використання часткових розв'язків задачі алгебраїчного синтезу може сприяти зменшенню часу роботи таких алгоритмів та зменшенню загального часу проектування цифрових пристроїв керування на базі мікропрограмного автомата з операційним автоматом переходів.

**Мета.** Розробка і дослідження алгоритмів пошуку повного і часткового розв'язків задачі алгебраїчного синтезу мікропрограмного автомата з операційним автоматом переходів.

**Метод.** В основу дослідження покладено структуру мікропрограмного автомата з операційним автоматом переходів. Синтез схеми автомата передбачає попереднє розв'язання задачі алгебраїчного синтезу. Результатом є так званий формальний розв'язок цієї задачі, який містить в собі дві складових. Першою складовою є визначені коди станів, другою складовою – арифметико-логічні операції, зіставлені окремим автоматним переходам. Автомат може бути синтезований в тому випадку, якщо при заданих кодах станів їх перетворення в процесі виконання переходів можливе за допомогою заданої множини операцій. Перевірка цієї можливості здійснюється за допомогою відомого матричного підходу. Він передбачає формування і поелементне зіставлення двох матриць – матриці переходів і об'єднаної матриці операцій. В результаті формується матриця покриття, яка відображає можливість реалізації (покриття) автоматних переходів за допомогою заданих арифметико-логічних операцій. Зміна способу кодування станів або набір операцій може давати інші розв'язки задачі алгебраїчного синтезу з більшою чи меншою кількістю покритих автоматних переходів.

**Результати.** На прикладі абстрактної граф-схеми алгоритму керування продемонстровано, що розв'язком задачі алгебраїчного синтезу мікропрограмного автомата з операційним автоматом переходів може вважатись ситуація, коли один або більше автоматних переходів не можуть бути реалізовані за допомогою заданого набору операцій. Таку ситуацію запропоновано називати частковим розв'язком задачі алгебраїчного синтезу. Реалізація усіх без винятку переходів за допомогою заданих операцій дає повний розв'зок цієї задачі, однак кількість повних розв'язків завжди є значно меншою за кількість часткових розв'язків. Отже, в загальному випадку пошук повних розв'язків займає набагато більше часу і до того ж є не завжди можливим.

**Висновки.** Проєктування логічної схеми мікропрограмного автомата з операційним автоматом переходів можливе у випадку наявності повного або часткового розв'язку задачі алгебраїчного синтезу. У випадку часткового розв'язку ті переходи, які не можуть бути реалізовані жодною з наявних операцій, реалізуються в канонічний спосіб за допомогою системи булевих рівнянь. Пошук повних розв'язків в загальному випадку потребує більше часу, ніж пошук часткових розв'язків. Це робить актуальним розробку алгоритмів і методів синтезу даного класу автоматів, основаних на пошуку часткових розв'язків задачі алгебраїчного синтезу.

**КЛЮЧОВІ СЛОВА:** мікропрограмний автомат, операційний автомат переходів, граф-схема алгоритму, арифметико-логічні операції, алгебраїчний синтез, частковий розв'язок.

## ЛІТЕРАТУРА

1. Bailliul J. Encyclopedia of Systems and Control / J. Bailliul, T. Samad. – Springer: London, UK, 2015. – 1554 p. DOI: https://doi.org/10.1007/978-1-4471-5058-9
2. Czerwinski R. Finite state machines logic synthesis for complex programmable logic devices / R. Czerwinski, D. Kania. – Berlin: Springer, 2013. – 172 p. DOI: https://doi.org/10.1007/978-3-642-36166-1
3. Baranov S. Logic and System Design of Digital Systems / S. Baranov. – Tallin : TUTPress, 2008. – 267 p.
4. Micheli G. D. Synthesis and Optimization of Digital Circuits / G. D. Micheli. – McGraw-Hill : Cambridge, MA, USA, 1994. – 579 p.
5. Minns P. FSM-Based Digital Design Using Verilog HDL / P. Minns, I. Elliot. – JohnWiley and Sons : Hoboken, NJ, USA, 2008. – 408 p. DOI: https://doi.org/ 10.1002/9780470987629
6. Skliarova I. Design of FPGA-based circuits using hierarchical finite state machines / I. Skliarova, V. Sklyarov, A. Sudnitson. – Tallinn : TUT Press, 2012. 240 p.
7. Klimovich A. S. Minimization of mealy finite-state machines by internal states gluing / A. S. Klimovich, V. V. Solov'ev // Journal of Computer and Systems Science International. – 2012. – Volume 51. – P. 244–255. DOI: https://doi.org/10.1134/S1064230712010091
8. Grout I. Digital Systems Design with FPGAs and CPLDs / I. Grout. – Elsevier Science: Amsterdam, The Netherlands, 2008. – 784 p. DOI: https://doi.org/10.1016/B978-0-7506-8397-5.X0001-3
9. Kubica M. Technology Mapping for LUT-Based FPGA / M. Kubica, A. Opara and D. Kania // Lecture Notes in Electrical Engineering. – Springer, Cham. – 2021. – Vol. 713. – 207 p. DOI: https://doi.org/10.1007/978-3-030-60488-2
10. Baranov S. Logic Synthesis for Control Automata / S. Baranov. – Dordrecht : Kluwer Academic Publishers, 1994. – 312 p.
11. Barkalov A. A. Operational formation of state codes in microprogram automata / A. A. Barkalov, R. M. Babakov // Cybernetics and Systems Analysis. – 2011. – Volume 47 (2). – P. 193–197. DOI: https://doi.org/10.1007/s10559-011-9301-y
12. Barkalov A. A. Synthesis of Finite State Machine with Datapath of Transitions According to the Operational Table of Transitions / A. A. Barkalov, L. A. Titarenko, R. M. Babakov // Radio Electronics, Computer Science, Control. – 2022. – No. 3 (62). – P. 109–119. DOI: https://doi.org/10.15588/1607-3274-2022-3-11
13. Barkalov A. A. A Matrix Method for Detecting Formal Solutions to the Problem of Algebraic Synthesis of a Finite-State Machine with a Datapath of Transitions / A. A. Barkalov, R. M. Babakov // Cybernetics and Systems Analysis. – 2023. – Volume 59 (2). – P. 190–198. DOI: https://doi.org/10.1007/s10559-023-00554-6
14. Barkalov A. A. Synthesis of VHDL-Model of a Finite State Machine with Datapath of Transitions / A. A. Barkalov, L. A. Titarenko, R. M. Babakov // Radio Electronics, Computer Science, Control. – 2023. – No. 4 (67). – P. 135–147. DOI: https://doi.org/10.15588/1607-3274-2023-4-13
15. Yang S. Logic synthesis and optimization benchmarks / Yang S. // User Guide Version 3.0. – Tech. rep. Microelectronics Center of North Carolina, P.O. Box 12889, Research Triangle Park, NC 27709, 1991.
16. Rawski M. An application of functional decomposition in ROM-based FSM implementation in FPGA devices / M. Rawski, H. Selvaraj, T. Luba // Journal of System Architecture. – 2005. – Volume 51. – P. 424– 434. DOI: https://doi.org/10.1016/j.sysarc.2004.07.004