ПРОГРЕСИВНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

PROGRESSIVE INFORMATION TECHNOLOGIES

ПРОГРЕССИВНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

UDC 004.2 : 004.94

SYNTHESIS OF THE FINITE STATE MACHINE WITH DATAPATH OF TRANSITIONS ACCORDING TO THE OPERATIONAL TABLE OF TRANSITIONS

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

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

Babakov R. M. – Dr. Sc., Associate Professor, Associate Professor of department of information technologies, Vasyl Stus Donetsk National University, Vinnytsia, Ukraine.

ABSTRACT

Context. The problem of formalizing the description of a microprogram finite state machine based on the principle of operational transformation of state codes with the help of a modified transition table is considered. The object of research was a model of a microprogram finite state machine with datapath of transitions.

Objective. The goal of the work is development and research of a method for formally specifying a microprogram finite state machine with datapath of transitions in the form of a modified table of transitions containing sufficient information for synthesizing the logic circuit of the finite state machine in the basis of programmable logic devices.

Method. A new way of representing the formal solution of the problem of algebraic synthesis of a microprogram finite state machine with datapath of transitions in the form of an operational table of transitions is proposed. This table is a modification of the direct structural table traditionally used in the synthesis of microprogram finite state machines. The use of the previously known representation of the formal solution of the problem of algebraic synthesis in the form of a system of isomorphisms of automaton algebras is too formalized and makes it difficult to synthesize the logical circuit of the finite state machine due to the separate description of the transition and output functions. It is shown that the structure of a microprogram finite state machine with datapath of transitions requires information about the set of interpretations of state codes and the transition operations used to be entered into the traditional table of transitions. It is noted that the proposed operational table of transitions contains sufficient information for the synthesis of the logical circuit of the finite state machine. An example of constructing an operational table of transitions for a finite state machine given by a graph-scheme of the implemented control algorithm is shown. The example demonstrates various ways to interpret state codes. The procedure for synthesizing the circuit for generating codes of transition operations and the circuit for generating microoperations according to the operational table of transitions is proposed.

Results. An example of the implementation of the main stages of the synthesis of a finite state machine with datapath of transitions according to the operational table of transitions is considered. Examples of synthesized finite state machine models in the VHDL language are given, which take into account the peculiarities of the representation of finite state machine models in Xilinx Vivado CAD. The results of the synthesis of the finite state machine according to VHDL models in FPGA basis are shown.

Conclusions. The experiments carried out confirmed the sufficiency of the operational table of transitions for describing a microprogram finite state machine with operational transformation of state codes for the purpose of further synthesizing its logic circuit. Prospects for further research are the use of the proposed operational table of transitions in the development of various methods for the synthesis and optimization of microprogram finite state machine with operational transformation of state codes.

KEYWORDS: finite state machine, datapath of transitions, table of transitions, synthesis of logical circuit, graph-scheme of algorithm.

ABBREVIATIONS

FSM is a finite state machine;
DT is a datapath of transitions;
GSA is a graph-scheme of algorithm;
OTT is an operational table of transitions;
BMO is a block of microoperations;
PLD is a programmable logic device.

DOI 10.15588/1607-3274-2022-3-11

NOMENCLATURE

a_m is a current FSM state;
$K(a_m)$ is a current FSM state code;
a_s is a transition state;
$K(a_s)$ is a transition state code;
X is a set of logic conditions;
x_i is an element of set X;
L is a number of logic conditions (power of set X);
Y is a set of microoperations;
y_i is an element of set Y;
N is a number of microoperations (power of set Y);
Φ_h is a set of input memory functions for switching FSM memory from state a_m to state a_s;
h is a number of FSM transition;
A is a set of FSM states;
a_i is an element of set A;
M is a number of FSM states (power of set A);
B is a number of transitions of FSM;
O is a set of transition operations;
O_i is an element of se O;
P is a number of transition operations (power of set P);
W is e set of signals of code of transitions operation;
w_i is an element of set W;
R_W is a digit capacity of code of transitions operation (power of set W);
I is a number of interpretations of state codes of FSM;
T is a set of signals of curretnt state code of FSM;
T_i is an element of set T;
R is a digit capacity of state code (power of set T).

INTRODUCTION

Digital systems are widely used in various fields of activity [1]. Structurally, the digital system can be considered as a combination of operational unit and control unit [2–3]. The control unit is based on a formal description of behavior and can be implemented in the form of a finite state machine model [4–5]. There are two models of finite state machines – the Mealy machine and the Moore machine [4–5]. The logic circuit of any FSM model is characterized by such parameters as hardware expenses, clock frequency and power consumption. As follows from [6], there is a direct relationship between these characteristics. Optimization of the characteristics of FSM circuits is an important scientific and practical problem, the solution of which is devoted to many scientific papers around the world [1–7]. One of such characteristics, which focuses on the finite state machine structure considered in this paper, is the hardware expenses of implementing the logic circuit of the FSM in a given element basis.

The object of study is the process of synthesis of the logic circuit of a finite state machine with operational transformation of state codes.

This process in the case of a canonical finite state machine is performed according to a direct structural table, which is a formal description of the behavior of the FSM and contains sufficient information for the synthesis of its circuit. In the case of operational transformation of state codes, this table requires modifications taking into account the processes of information transformation that takes place in this class of FSM.

The subject of study is the finite state machine with operational transformation of state codes, in which the transformation of state codes is carried out using a finite set of arithmetical and logical operations.

The purpose of the work is formalization of the description of the finite state machine with operational transformation of state codes in the form of the modified direct structural table.

1 PROBLEM STATEMENT

Suppose given finite state machine, characterized by the sets $A=\{a_1, ..., a_M\}$, $X=\{x_1, ..., x_L\}$, $Y=\{y_1, ..., y_N\}$ and implements a certain control algorithm. The synthesis of the logic circuit of the FSM provides for the implementation of the function of transitions $T=T(X, T)$ and the function of outputs $Y=Y(X, T)$ in the given elementary basis.

The paper solves the problem of synthesizing a finite state machine with datapath of transitions, in which the transition function $T=T(T, W)$ depends on the code of the current state and the code of transitions operation. To solve the problem, it is necessary to develop a formalized representation of the FSM with DT, which allows the following stages of the synthesis of the logic circuit of the FSM:

– synthesis of each structural blocks;
– construction of VHDL description of the synthesized FSM;
– FSM synthesis using Xilinx Vivado CAD using FPGA.

2 REVIEW OF THE LITERATURE

Various optimization methods for reducing of hardware expenses of FSM circuit are known today. Such methods include, for example, methods of structural decomposition [7]. Their use leads to FSM circuits with several levels of conversion of logic signals.

Another approach to reducing hardware amount, considered in this article, is to use the principle of operational transformation of state codes [8]. According to it, the transformation of state codes in an FSM is not carried out with the help of a canonical system of Boolean equations, but with using of a set of arithmetical and logicalal operations that form a special datapath of transitions. This structure of FSM with DT shows a fairly high efficiency in terms of hardware expenses [9].

In the work [10] considers an algebraic model of FSM with DT, according to which this FSM can be represented as a system of isomorphisms of partial algebras (transition algebras). Each transition algebra describes the rule of transformation of state codes for its subset of FSM transitions and assumes its own scalar or vector interpretation of state codes. The system of isomorphisms of algebras today is the only formal way to specify the FSM with DT. However, the synthesis of the FSM circuit directly by the system of isomorphisms of algebras is complicated due to different representations of FSM transitions.

DOI 10.15588/1607-3274-2022-3-11

A direct structural table is traditionally used to specify the FSM behavior [2, 7]. The method of synthesis of FSM according to the table of transitions is widely known and applied in practice [7]. This article proposes a new way to specify the FSM with DT, based on a modified table of transitions.

3 MATERIALS AND METHODS

The canonical FSM is usually set in the form of a direct structural table (table of transitions), the format of which is presented in Fig. 1 [2]. The purpose of columns of the table of transitions and its use for the synthesis of the FSM circuit are described in [7].

a_m	$K(a_m)$	a_s	$K(a_s)$	X_h	Y_h	Φ_h	h
...	...	...	...	...	...	...	...

Figure 1 – Structure of the table of transitions

Let the FSM be given in the form of a GSA G, [7], which is shown in Fig. 2, left. In the right part of Fig. 2 shows a description of GSA G in the *kiss* format, which is used to describe finite state machines in the test collection LGSynth93 [11]. GSA G is marked by the states of the Moore FSM, contains $M = 10$ states $a_0 – a_9$, $L = 3$ input signals $x_1 – x_3$, $N = 4$ output signals $y_1 – y_4$ and $B = 13$ FSM transitions. To encode 10 states, it is sufficient to use $R = 4$ binary digits.

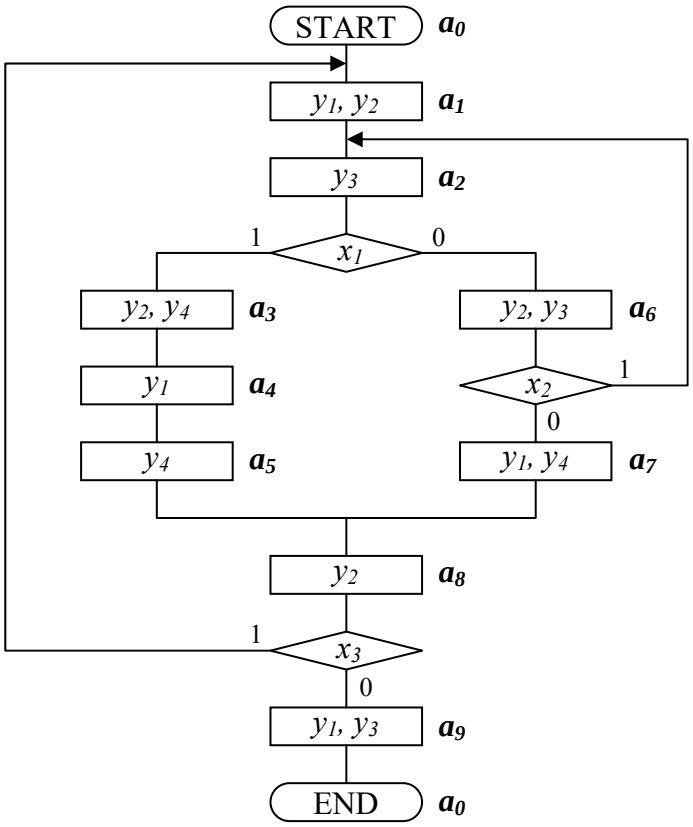
Let's synthesize for GSA G a finite state machine with datapath of transitions. Suppose the next transitions operations $O_1 – O_3$ are given:

$$O_1: \quad K(a_s) = K(a_m) + 9_{10}; \tag{1}$$

$$O_2: \quad K(a_s) = K(a_m) \,\&\, 1000_2; \tag{2}$$

$$O_3: \quad K(a_s) = K(a_m) \oplus 0100_2. \tag{2}$$

In operation O_1, a decimal constant 9 is added to the current state code $K(a_m)$. This means that a scalar decimal interpretation is used for the codes $K(a_m)$ and $K(a_s)$ when performing O_1. In operations O_2 and O_3 over the code $K(a_m)$ bitwise logical operations with binary constant are performed. Therefore, the codes $K(a_m)$ and $K(a_s)$ when performing operations O_2, O_3 are interpreted as binary vectors. Thus, when using operations (1)–(3) for state codes will be used two different interpretations: decimal number and binary vector. Note that the arguments and values of operations (1)–(3) are in the range [0; 15], i.e. in the range of numbers specified by four-digit binary codes. For example, in the circuit implementation of the operation "+9" the result is always modulo 16 (in fact, the lower four binary digits of the result are taken).




```
.i 3
.o 4
.p 12
.s 10
--- a0 a1 0000
--- a1 a2 1100
1-- a2 a3 0010
0-- a2 a6 0010
--- a3 a4 0101
--- a4 a5 1000
--- a5 a8 0001
-1- a6 a2 0110
-0- a6 a7 0110
--- a7 a8 1001
--1 a8 a1 0100
--0 a8 a9 0100
--- a9 a0 1010
```

Figure 2 – Graph-scheme of algorithm G

DOI 10.15588/1607-3274-2022-3-11

Let's perform an algebraic synthesis of FSM with DT, which is as follows:

– to each state of the FSM we will match the unique four-digit binary vector having the corresponding decimal interpretation;

– for each FSM transition we will match any operation from (1) – (3), which for the given state codes transforms the code $K(a_m)$ into the code $K(a_s)$.

The result of algebraic synthesis in graphical form is shown in Fig. 3. In each vertex, which is marked by the Moore FSM state, the state code is shown in scalar (decimal) and vector (binary) interpretations. Each FSM transition is marked by one of the operations (1) – (3): "+9", "& 1000" or "⊕ 0100". Since the operational transformation of state codes affects only the function of the FSM transitions and does not affect the function of the outputs, the microoperations in Fig. 3 are not shown, although they continue to correspond to Fig. 2.

As we can see, with the chosen values of state codes, all transitions within the GSA *G* are implemented using of operations (1)–(3). For example, the transition from state a_3 with code $K(a_3) = 9_{10} = 1001_2$ to state a_4 with code $K(a_4) = 2_{10} = 0010_2$ is performed using the operation "+9", and from the result $18_{10} = 10010_2$ the lower four digits 0010_2 were taken.

Let's modify the table of transitions of the canonical FSM as follows:

1. Instead of column $K(a_m)$ add columns $K_1(a_m)$, $K_2(a_m)$, ..., $K_I(a_m)$, which correspond to all used interpretations of state codes for all *I* interpretations.

2. Do the same with column $K(a_s)$, adding columns $K_1(a_s)$, $K_2(a_s)$, ..., $K_I(a_s)$ instead.

3. Add a column W_h containing information about the code of transitions operation that implements current FSM transition. The values of w_i specified in this column correspond to the values 1 in the binary code of the corresponding operation. Filling in this column is preceded by the step of encoding of transitions operations.

4. Remove the column Φ_h, because the conversion of state codes in the FSM with DT is carried out using a set of transitions operations, rather than a system of canonical equations of the transition function.

Let's call the received table as the operational table of transitions (OTT). In the general case, its structure corresponds to Fig. 4.

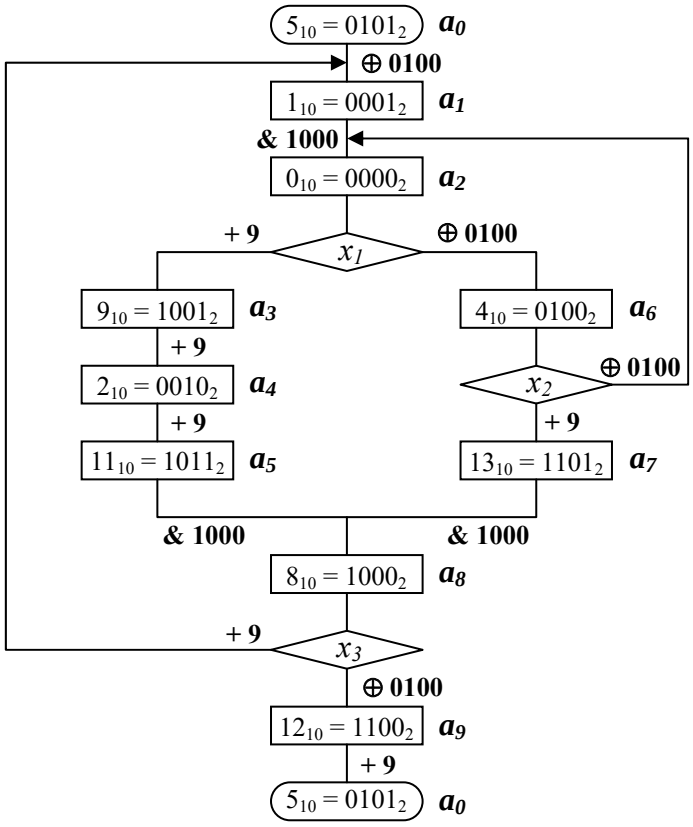


Figure 3 – Result of algebraic synthesis for GSA *G* and operations (1)–(3)

a_m	$K_1(a_m)$	...	$K_I(a_m)$	a_s	$K_1(a_s)$	...	$K_I(a_s)$	X_h	W_h	Y_h	h

Figure 4 – Structure of operational table of transitions

DOI 10.15588/1607-3274-2022-3-11

Let's present in the form of OTT the results of algebraic synthesis of the FSM, shown in Fig. 3. To do this, we encode the given transitions operations $O_1 - O_3$ with unique binary codes of bit size $R_W = \lceil \log_2 3 \rceil = 2$, which are formed by signals $W = \{w_1, w_2\}$. The result of coding is presented in Table 1. Taking into account the coding of transitions operations, the operational table of transitions, hich corresponds to GSA G and Fig. 3, is presented by Table 2.

Table 1 – Coding of operations $O_1 - O_3$

O_i	$w_1\ w_2$
O_1	0 0
O_2	0 1
O_3	1 0

In this example, the number of interpretations of the state codes $I = 2$ (scalar decimal value and binary vector). When using the transition operation O_1, the code $K_1(a_m)$ is converted into the code $K_1(a_s)$; when using operations O_2 and O_3, the code $K_2(a_m)$ is converted into the code $K_2(a_s)$. For example, the transition $h = 5$ is realized by operation O_1. Therefore, in this transition, the conversion of scalar interpretations of codes is performed, i.e. the code $K_1(a_m) = 9_{10}$ into the code $K_1(a_s) = 2_{10}$. This transformation is performed using the adder circuit with the preservation of four lower digits of the result.

Note that the dash in the column W_h means that for the implementation of the corresponding transition, values $w_i = 1$ are not formed, which corresponds to the code of operation O_1 ($w_1 = 0$, $w_2 = 0$).

4 EXPERIMENTS

Let's show an example of Table 1, that the information contained in the OTT is sufficient for the synthesis of the logic circuit of the FSM with DT. The structural model of the FSM with DT, corresponding to the Moore FSM, is shown in Fig. 5 and contains the next synthesized blocks:

– the block W generates a set of signals W with digit capacity $R_W = \lceil P \rceil$ according to formula (4), where P is the number of transition operations; X is the set of L input signals of the FSM corresponding to the logical conditions x_1, ..., x_l of the given GSA; T – state code of the FSM with digit capacity R

$$W = W(X, T); \tag{4}$$

– the block DT corresponds to the datapath of transitions and implements a set of transitions operations in the form of a set of separate combinational circuits, the outputs of which are multiplexed by the signal W and enter the FSM memory register that is part of the DT [8];

– the block BMO corresponds to the cicruit of formation of microoperations and implements the output function of the Moore FSM in the form of a set of microoperations $Y = \{y_1, ..., y_n\}$ according to formula (5) by analogy with [2, 7]

$$Y = Y(T). \tag{5}$$

Let's synthesize these blocks. Under the synthesis of the logic circuit of the machine we will understand the development of VHDL-model, which can be synthesized in the element basis of Xilinx FPGA [12].

Table 2 – Operational table of transitions (GSA G)

a_m	$K_1(a_m)$	$K_2(a_m)$	a_s	$K_1(a_s)$	$K_2(a_s)$	X_h	W_h	Y_h	h
a_0	5	0101	a_1	1	0001	1	w_1	–	1
a_1	1	0001	a_2	0	0000	1	w_2	y_1, y_2	2
a_2	0	0000	a_3	9	1001	x_1	–	y_3	3
			a_6	4	0100	$\bar{x}_1$	w_1		4
a_3	9	1001	a_4	2	0010	1	–	y_2, y_4	5
a_4	2	0010	a_5	11	1011	1	–	y_1	6
a_5	11	1011	a_8	8	1000	1	w_2	y_4	7
a_6	4	0100	a_2	0	0000	x_2	w_2	y_2, y_3	8
			a_7	13	1101	$\bar{x}_2$	–		9
a_7	13	1101	a_8	8	1000	1	w_2	y_1, y_4	10
a_8	8	1000	a_1	1	0001	x_3	–	y_2	11
			a_9	12	1100	$\bar{x}_3$	w_1		12
a_9	12	1100	a_0	5	0101	1	–	y_1, y_3	13

DOI 10.15588/1607-3274-2022-3-11

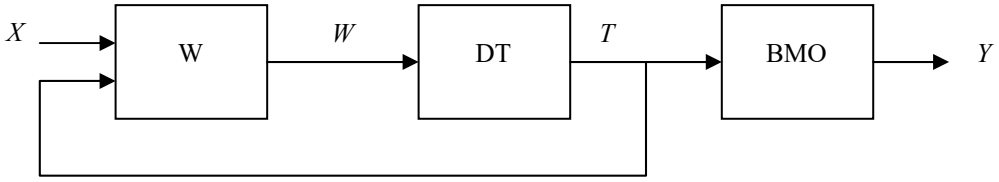


Figure 5 – Structural model of FSM with DT

Block W

This block implements a system of Boolean equations of function (4), in which each signal wi is formed in accordance with expression (6)

$$w_i = \bigvee_{h=1}^{H} C_{ih} A_m^h X_h \quad (i = \overline{1, R_W}) \tag{6}$$

In this formula, C_{ih} is a Boolean variable equal to 1, if and only if the function w_i is written in the OTT line number h; A_m^h is a conjunction of internal variables, corresponding to the binary representation of state code a_m from the line number h (for example, $A_5^7 = T_1\overline{T}_2T_3T_4$); X_h is a conjunction of signals of logical conditions, written in the column X_h of the line number h ($X_h = 1$ for unconditional transitions).

For this example, the system of equations of function (4) is the next:

$$\begin{cases} w_1 = \overline{T}_1T_2\overline{T}_3T_4 \vee \overline{T}_1\overline{T}_2\overline{T}_3\overline{T}_4\overline{x}_1 \vee T_1\overline{T}_2\overline{T}_3\overline{T}_4\overline{x}_3; \\ w_2 = \overline{T}_1\overline{T}_2\overline{T}_3T_4 \vee T_1\overline{T}_2T_3T_4 \vee \overline{T}_1T_2\overline{T}_3\overline{T}_4x_2 \vee T_1T_2\overline{T}_3T_4. \end{cases} \tag{7}$$

In the general case for the system (7) it is possible to carry out minimization in order to reduce the complexity of the circuit [4, p. 269].

System (7) can be described in VHDL in different ways, for example, as a separate process (Fig. 6) [12].

In this model, the description of the buses *T*, *X* and *W* corresponds to the same FSM signals and is discussed below in the description of the architecture block.

Block DT

This block includes an operational part that implements operations (1) – (3) and their multiplexing, as well as a memory register designed to store the current FSM state. The functional diagram of these nodes is shown in Fig. 7. Since for the considered example the circuit of DT consists of standard functional blocks, special synthesis of this circuit is trivial and is not required.

In Fig. 8 VHDL-model of OAP, in which the operational part and the memory register are represented by separate processes, is showed.

The first process corresponds to the operational part of DT. The absence of the synchronization signal *C* in the list of sensitivity of the process indicates the asynchronous nature of the operation of this block. Within the process transformation of the state code *T* is performed using one of three transitions operations depending on the value of the operation code *W*. As will be shown below, for the signal *T* used data type "unsigned", which allows you to interpret this signal simultaneously as an unsigned integer and as a binary vector.

The second process corresponds to the memory register. Receiving data in the register, as well as the switching to the initial state 0101_2 in the presence of the *Reset* signal are carried out synchronously on the leading edge of the *C* signal.

Block BMO

The synthesis of this block is performed in accordance with the contents of column Y_h of the operational table of transitions (Table 2). In this case, to obtain a synthesized VHDL model of the block, it is possible to use the method considered for block *W* (build a system of Boolean equations for generated microoperations), or use the *case* operator belonging to the synthesized subset of the VHDL language. We use the second method, as a result of which we obtain the VHDL model of the BMO block, shown in Fig. 9.

In this model, after the start of the process, all digits of the output bus *Y* are given zero values. Then, depending on the values of the signals on the bus *T*, the required discharges of the bus *Y* are set to 1. It is expected that state code values not provided by the case operator should not appear on the *T* bus.

```
process (T, X)    -- Block W
begin
    W(1) <= (not(T(1)) and T(2) and not(T(3)) and T(4)) or
       (not(T(1)) and not(T(2)) and not(T(3)) and not(T(4)) and not(X(1))) or
            (T(1) and not(T(2)) and not(T(3)) and not(T(4)) and not(X(3)));

    W(2) <= (not(T(1)) and not(T(2)) and not(T(3)) and T(4)) or
            (T(1) and not(T(2)) and T(3) and T(4)) or
            (not(T(1)) and T(2) and not(T(3)) and not(T(4)) and X(2)) or
            (T(1) and T(2) and not(T(3)) and T(4));
end process;
```

Figure 6 – VHDL model of block W

DOI 10.15588/1607-3274-2022-3-11

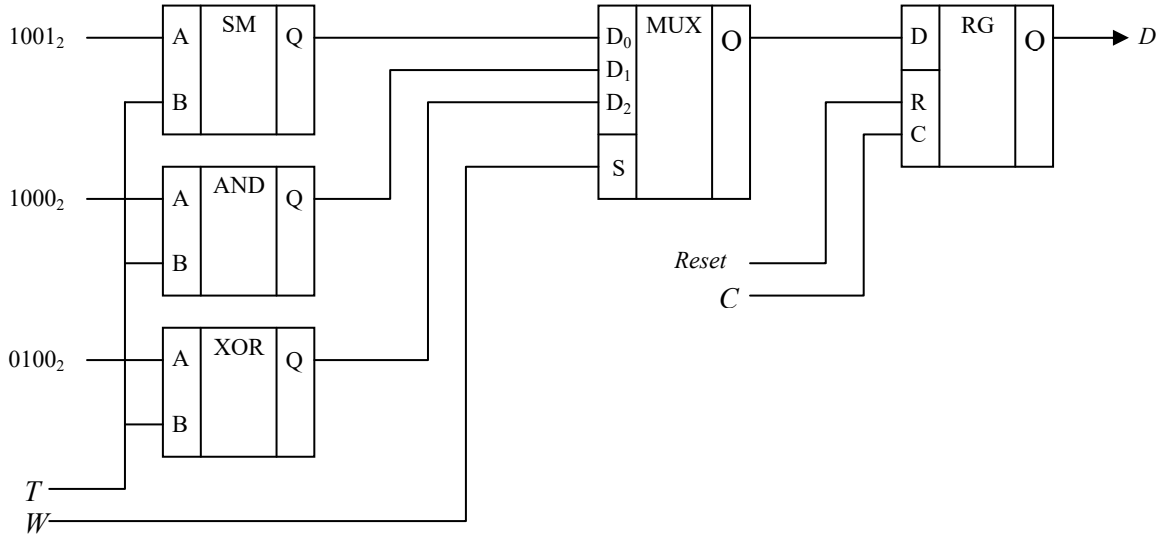


Figure 7 – Functional diagram of datapath of transitions

```
process (W, T)           -- Datapath of Transitions
begin
    if W = "00" then
        D <= T + 9;
    elsif W = "01" then
        D <= T and "1000";
    else
        D <= T xor "0100";
    end if;
end process;


process (C)              -- State Register
begin
    if rising_edge(C) then
       if Reset = '1' then
           T <= "0101";
       else
           T <= D;
       end if;
    end if;
end process;
```

Figure 8 – VHDL model of datapath of transitions

```
process (T)                    -- Microoperations
begin
    Y <= (others => '0');
    case T is
        when "0001" => Y(1) <= '1'; Y(2) <= '1';
        when "0000" => Y(3) <= '1';
        when "1001" => Y(2) <= '1'; Y(4) <= '1';
        when "0010" => Y(1) <= '1';
        when "1011" => Y(4) <= '1';
        when "0100" => Y(2) <= '1'; Y(3) <= '1';
        when "1101" => Y(1) <= '1'; Y(4) <= '1';
        when "1000" => Y(2) <= '1';
        when "1100" => Y(1) <= '1'; Y(3) <= '1';
        when others => null;
    end case;
end process;
```

Figure 9 – VHDL model of block BMO

DOI 10.15588/1607-3274-2022-3-11

Let's combine the considered VHDL descriptions into a single object entity, resulting in a synthesized model of FSM with DT (Fig. 10). A feature of this model is the presence of the output port *S*, which displays the code of the current state *T*. This is done in order to analyze functioning of the FSM in the process of behavioral modeling.

5 RESULTS

Synthesis of the VHDL model shown in Fig. 10, in CAD Xilinx Vivado 2021.1 allowed to obtain hardware expenses for implementation of the FSM, equal to 7 LUT-elements (based on FPGA xc7a12ticsg325-1L FPGA of Artix-7 series).

To test the correctness of VHDL model of the FSM with DT, a behavioral part was developed that implements the next functionality:

– single generation of the *Reset* signal in the range of 10–90 *ns* from the beginning of the simulation;

– regular signal *C* generation with a duration of 20 *ns* with an interval of 100 *ns*;

– regular generation of input signals $x_1 - x_3$ with different values of lengths of 1 and 0 levels.

Fig. 10 shows a fragment of the time diagram of the FSM in the process of behavioral modeling. This fragment demonstrates the correctness of the transition function and the output function of the FSM. For example, at

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
-- ----------------------------------------------------------------------
entity OAP is
    generic (R: integer := 4;  L: integer := 3;
             RW: integer := 2; N: integer := 4);

    port (X: in std_logic_vector (1 to L);
          C: in std_logic;
          Reset: in std_logic;
          S: out unsigned (1 to R);
          Y: out std_logic_vector (1 to N));
end entity OAP;
-- ----------------------------------------------------------------------
architecture OAP_A of OAP is
signal T, D: unsigned (1 to R);
signal W: std_logic_vector (1 to RW);
begin
    process (C)           -- State register
    ...
    process (T, X)        -- Block W
    ...
    process (W, T)        -- Datapath of transitions
    ...
    process (T)           -- Block of microoperations
    ...
    S <= T;               -- For debugging
end architecture OAP_A;
```

Figure 10 – Synthesizable VHDL model of FSM with DT

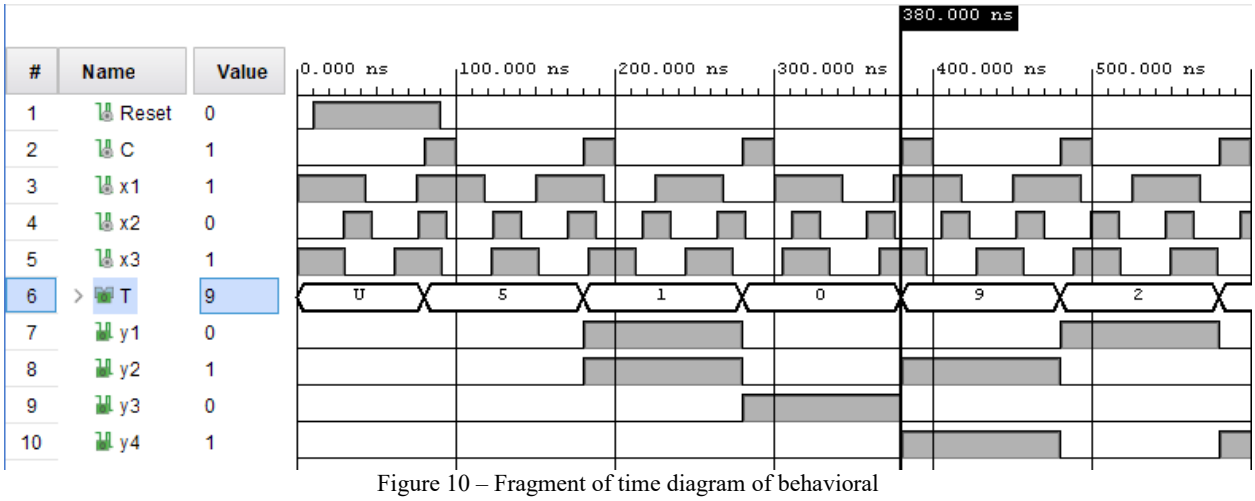


Figure 10 – Fragment of time diagram of behavioral modelling of work of FSM with DT (in interval from 0 to 600 ns)

DOI 10.15588/1607-3274-2022-3-11

time t = 380 *ns*, marked by a vertical marker, the FSM performs transition from the state with the code $T = 0$ (state a_2), analyzing value of signal x_1. Since at this point $x_1 = 1$, the transition is carried out to a state with code $T = 9$ (state a_3), which is consistent with Fig. 3. After the transition to state a_3, the formation of microoperations y_2 and y_4 is carried out, which is consistent with Fig. 2. Thus, the synthesis of FSM with DT for this example is performed correctly.

6 DISCUSSION

A finite state machine with datapath of transitions differs from a canonical finite state machine in that it uses a set of arithmetical and logical operations (transitions operations) to convert state codes, which form a datapath of transitions. Each transitions operation involves a certain interpretation of binary state codes and is formally specified on a set of interpreted values. One of the interpretations is a scalar representation in the form of an unsigned integer, which allows to define on the set of state codes operations of addition, subtraction, and so on. Interpretation of the state code in the form of a binary vector allows to specify over the state codes bitwise logical operations, shift operations and the others. Information about the used operations and methods of interpretation of binary state codes is necessary for the schematic implementation of the transition function of the FSM.

The input data for the synthesis of the logic circuit of the canonical FSM is a table of transitions (direct structural table), which contains information about the functions of transitions and outputs of the FSM. However, its use for FSM with DT is impossible due to the lack of information about the methods of interpretation of state codes and transitions operations. In this paper, it is proposed to use the so-called operational table of transitions for specification the FSM with DT, which provides extended information about the function of transitions of the FSM. The example considered in the paper showed that specification of the FSM with DT in the form of an operational table of transitions is sufficient for the synthesis of the logic circuit of the FSM in the form of VHDL model focused on the use element basis of FPGA-type.

The possibility of obtaining with the help of VHDL model numerical characteristics of hardware expenses for the implementation of the FSM circuit allows us to recommend the use of the operational table of transitions as a way to present the results of other known methods of hardware expenses in FSM circuit.

CONCLUSIONS

The article proposes a solution of scientific problem of formalizing the description of the processes of state codes transformation in a finite state machine with datapath of transitions, which allows to bring the description of this class of finite state machine in line with traditional description of other classes of finite state machines.

The scientific novelty of the work is to modify the direct structural table by adding to it information about algebraic interpretation and methods of transformation of state codes. The resulting operational table of transitions is proposed for the first time and contains sufficient information for the synthesis of the logic circuit of the FSM.

Practical use of the obtained results is possible in the development of formal methods of structural synthesis of finite state machines with operational transformation of state codes.

Prospects for further research are to develop methods of synthesis of FSM circuit, based on the formal representation of the FSM in the form of operational table of transitions.

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.
2. Sklyarov V., Sklyarova I., Barkalov A., Titarenko L. Synthesis and Optimization of FPGA-Based Systems; Volume 294 of Lecture Notes in Electrical Engineering. Springer, Berlin, Germany, 2014, 432 p.
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.
6. Grout I. Digital Systems Design with FPGAs and CPLDs. Elsevier Science, Amsterdam, The Netherlands, 2011, 784 p.
7. Baranov S. Logic Synthesis for Control Automata. Dordrecht, Kluwer Academic Publishers, 1994, 312 p.
8. 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.
9. Barkalov A. A., Babakov R. M. Determining the Area of Efficient Application of a Microprogrammed Finite-State Machine with Datapath of Transitions, *Cybernetics and Systems Analysis,* 2019,Volume 54 (3), pp. 366–375.
10. Barkalov A. A., Babakov R. M. Algebraic Interpretation of a Microprogram Finite-State Machine with Datapath of Transitions, *Cybernetics and Systems Analysis,* 2016, Volume 54 (3), pp. 366–375.
11. McElvain, K. LGSynth93 Benchmark; Mentor Graphics. Wilsonville, OR, USA, 1993.
12. Xilinx. XST UserGuide. V.11.3. Available online: https://www.xilinx.com/support/documentation/sw_manuals/xilinx11/xst.pdf (accessed on 07 June 2022).

Received 17.06.2022.
Accepted 12.08.2022.

DOI 10.15588/1607-3274-2022-3-11

УДК 004.2 : 004.94

СИНТЕЗ МІКРОПРОГРАМНОГО АВТОМАТА З ОПЕРАЦІЙНИМ АВТОМАТОМ ПЕРЕХОДІВ ЗА ОПЕРАЦІЙНОЮ ТАБЛИЦЕЮ ПЕРЕХОДІВ

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.

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

АНОТАЦІЯ

Актуальність. Розглянуто задачу формалізації опису мікропрограмного автомата, заснованого на принципі операційного перетворення кодів станів, за допомогою модифікованої таблиці переходів. Об'єктом дослідження була модель мікропрограмного автомата з операційним автоматом переходів. Мета роботи – розробка та дослідження способу формального завдання мікропрограмного автомата з операційним автоматом переходів у вигляді модифікованої таблиці переходів, що містить достатню інформацію для синтезу логічної схеми автомата в базисі програмувальних логічних пристроїв.

Метод. Запропоновано новий спосіб представлення формального рішення задачі алгебраїчного синтезу мікропрограмного автомата з операційним автоматом переходів у вигляді операційної таблиці переходів. Ця таблиця є модифікацією прямої структурної таблиці, зазвичай використовуваної при синтезі мікропрограмних автоматів. Використання раніше відомого представлення формального вирішення задачі алгебраїчного синтезу у вигляді системи ізоморфізмів автоматних алгебр є надто формалізованим і ускладнює синтез логічної схеми автомата внаслідок роздільного опису функцій переходів і виходів. Показано, що структура мікропрограмного автомата з операційним автоматом переходів вимагає внесення до традиційної таблиці переходів інформації про множину інтерпретацій кодів станів і використовувані операції переходів. Зазначено, що запропонована операційна таблиця переходів містить достатню інформацію для синтезу логічної схеми автомата. Наведено приклад побудови операційної таблиці переходів для автомата, заданого граф-схемою імплементованого алгоритму керування. У прикладі показані різні методи інтерпретації кодів станів. Запропоновано порядок синтезу схеми формування кодів операцій переходів та схеми формування мікрооперацій за операційною таблицею переходів.

Результати. Розглянуто приклад виконання основних етапів синтезу мікропрограмного автомата з операційним автоматом переходів за операційною таблицею переходів. Наведено приклади моделей синтезованого автомата мовою VHDL, які враховують особливості представлення моделей кінцевих автоматів у САПР Xilinx Vivado. Показано результати синтезу автомата за VHDL-моделями у базисі ПЛІС FPGA.

Висновки. Проведені експерименти підтвердили достатність операційної таблиці переходів для опису мікропрограмного автомата з операційним перетворенням станів кодів з метою подальшого синтезу його логічної схеми. Перспективи подальших досліджень полягають у використанні запропонованої операційної таблиці переходів при розробці різних методів синтезу та оптимізації мікропрограмних автоматів з операційним перетворенням кодів станів.

КЛЮЧОВІ СЛОВА: мікропрограмний автомат, операційний автомат переходів, таблиця переходів, синтез логічної схеми, граф-схема алгоритму.

УДК 004.2 : 004.94

СИНТЕЗ МИКРОПРОГРАММНОГО АВТОМАТА С ОПЕРАЦИОННЫМ АВТОМАТОМ ПЕРЕХОДОВ ПО ОПЕРАЦИОННОЙ ТАБЛИЦЕ ПЕРЕХОДОВ

Баркалов А. А. – д-р техн. наук, профессор, профессор Института компьютерных наук и электроники университета Зеленогурского, г. Зелена Гура, Польша.

Титаренко Л. А. – д-р техн. наук, профессор, профессор Института компьютерных наук и электроники университета Зеленогурского, г. Зелена Гура, Польша.

Бабаков Р. М. – д-р техн. наук, доцент, доцент кафедры информационных технологий Донецкого национального университета имени Васыля Стуса, г. Винница, Украина.

АННОТАЦИЯ

Актуальность. Рассмотрена задача формализации описания микропрограммного автомата, основанного на принципе операционного преобразования кодов состояний, с помощью модифицированной таблицы переходов. Объектом исследования являлась модель микропрограммного автомата с операционным автоматом переходов. Цель работы – разработка и исследование способа формального задания микропрограммного автомата с операционным автоматом переходов в виде модифицированной таблицы переходов, содержащей достаточную информацию для синтеза логической схемы автомата в базисе программируемых логических устройств.

Метод. Предложен новый способ представления формального решения задачи алгебраического синтеза микропрограммного автомата с операционным автоматом переходов в виде операционной таблицы переходов. Данная таблица является модификацией прямой структурной таблицы, традиционно используемой при синтезе микропрограммных автоматов. Использование ранее известного представления формального решения задачи алгебраического синтеза в виде системы изоморфизмов автоматных алгебр является слишком формализованным и затрудняет синтез логической схемы автомата по причине раздельного описания функций переходов и выходов. Показано, что структура микропрограммного автомата с

DOI 10.15588/1607-3274-2022-3-11

операционным автоматом переходов требует внесения в традиционную таблицу переходов информации о множестве интерпретаций кодов состояний и используемых операциях переходов. Отмечено, что предложенная операционная таблица переходов содержит достаточную информацию для синтеза логической схемы автомата. Приведен пример построения операционной таблицы переходов для автомата, заданного граф-схемой имплементируемого алгоритма управления. В примере продемонстрированы различные способы интерпретации кодов состояний. Предложен порядок синтеза схемы формирования кодов операций переходов и схемы формирования микроопераций по операционной таблице переходов.

Результаты. Рассмотрен пример выполнения основных этапов синтеза микропрограммного автомата с операционным автоматом переходов по операционной таблице переходов. Даны примеры моделей синтезированного автомата на языке VHDL, которые учитывают особенности представления моделей конечных автоматов в САПР Xilinx Vivado. Показаны результаты синтеза автомата по VHDL-моделям в базисе ПЛИС FPGA.

Выводы. Проведенные эксперименты подтвердили достаточность операционной таблицы переходов для описания микропрограммного автомата с операционным преобразованием кодов состояний с целью дальнейшего синтеза его логической схемы. Перспективы дальнейших исследований заключаются в использовании предложенной операционной таблицы переходов при разработке различных методов синтеза и оптимизации микропрограммных автоматов с операционным преобразованием кодов состояний.

КЛЮЧЕВЫЕ СЛОВА: микропрограммный автомат, операционный автомат переходов, таблица переходов, синтез логической схемы, граф-схема алгоритма.

ЛІТЕРАТУРА / ЛИТЕРАТУРА

1. Bailliul J. Encyclopedia of Systems and Control / J. Bailliul, T. Samad. – Springer : London, UK, 2015. – 1554 p.
2. Synthesis and Optimization of FPGA-Based Systems; Volume 294 of Lecture Notes in Electrical Engineering / [V. Sklyarov, I. Sklyarova, A. Barkalov, L. Titarenko]. – Springer : Berlin, Germany, 2014. – 432 p.
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.
6. Grout I. Digital Systems Design with FPGAs and CPLDs / I. Grout. – Elsevier Science : Amsterdam, The Netherlands, 2011. – 784 p.
7. Baranov S. Logic Synthesis for Control Automata / S. Baranov. – Dordrecht : Kluwer Academic Publishers, 1994. – 312 p.
8. 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.
9. Barkalov A. A. Determining the Area of Efficient Application of a Microprogrammed Finite-State Machine with Datapath of Transitions / A. A. Barkalov, R. M. Babakov // Cybernetics and Systems Analysis. – 2019. – Volume 54 (3). – P. 366–375.
10. Barkalov A. A. Algebraic Interpretation of a Microprogram Finite-State Machine with Datapath of Transitions / A. A. Barkalov, R. M. Babakov // Cybernetics and Systems Analysis. – 2016. – Volume 54 (3). – P. 366–375.
11. McElvain K. LGSynth93 Benchmark; Mentor Graphics / K. McElvain. – Wilsonville, OR, USA, 1993.
12. Xilinx. XST UserGuide. V.11.3. Available online: https://www.xilinx.com/support/documentation/sw_manuals/xilinx11/xst.pdf (accessed on 07 June 2022).

DOI 10.15588/1607-3274-2022-3-11