p-ISSN 1607-3274 Радіоелектроніка, інформатика, управління. 2023. № 4
e-ISSN 2313-688X Radio Electronics, Computer Science, Control. 2023. № 4

ПРОГРЕСИВНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

PROGRESSIVE INFORMATION TECHNOLOGIES

UDC 004.94 : 004.2

SYNTHESIS OF VHDL-MODEL OF A FINITE STATE MACHINE WITH DATAPATH 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, Professor of Department of Information Technologies, Vasyl Stus Donetsk National University, Vinnytsia, Ukraine.

ABSTRACT

Context. The problem of building a program model of a finite state machine with datapath of transitions using VHDL language is considered. The model synthesis process is identified with the synthesis of this type of finite state machine, since the built model can be used both for the analysis of the device's behavior and for the synthesis of its logic circuit in the FPGA basis. The object of the research is the automated synthesis of the logic circuit of the finite state machine with the datapath of transitions, based on the results of which numerical characteristics of the hardware expenses for the implementation of the state machine circuit can be obtained. This makes it possible to evaluate the effectiveness of using this structure of the finite state machine when implementing a given control algorithm.

Objective. Development and research of a VHDL model of a finite state machine with datapath of transitions for the analysis of the behavior of the state machine and the quantitative assessment of hardware expenses in its logic circuit.

Method. The research is based on the structural diagram of a finite state machine with datapath of transitions. The synthesis of individual blocks of the structure of the state machine is carried out according to a certain procedure by the given graph-scheme of the con-trol algorithm. It is proposed to present the result of the synthesis in the form of a VHDL description based on the fixed values of the states codes of the state machine. The process of synthesizing the datapath of transitions, the block of formation of codes of transitions operations and the block of formation of microoperations is demonstrated. VHDL description of that blocks is carried out in a synthesizable style, which allows synthesis of the logic circuit of the finite state machine based on FPGA with the help of modern CAD and obtaining numerical characteristics of the circuit, in particular, the value of hardware expenses. To analyze the correctness of the synthesized circuit, the process of developing the behavioral component of the VHDL model, the function of which is the generation of input signals of the finite state machine, is considered. The classical combination of the synthesizable and behavioral parts of the model allows presenting the results of the synthesis of a finite state machine with datapath of transitions as a separate project that can be used as a structural component of the designed digital system.

Results. Using the example of an abstract graph-scheme of the control algorithm, a VHDL model of a finite state machine with datapath of transitions was developed. With the help of CAD AMD Vivado, a synthesis of the developed model was carried out and behavioral modeling of the operation of the finite state machine circuit was carried out. The results of the circuit synthesis made it possible to obtain the value of hardware expenses when implementing the circuit in the FPGA basis. According to the results of behavioral modeling, time diagrams were obtained, which testify to the correctness of the implementation of the functions of transitions and outputs of the synthesized state machine.

Conclusions. In traditional VHDL models of finite state machines, the states do not contain specific codes and are identified using literals. This allows CAD to encode states at its own discretion. However, this approach is not suitable for describing a finite state machine with datapath of transitions. The transformation of states codes using a set of arithmetic and logic operations requires the use of fixed values of states codes, which determines the specifics of the VHDL model proposed in this paper. This and similar models can be used, in particular, in the study of the effectiveness of a finite state machine according to the criterion of hardware expenses in the device circuit.

KEYWORDS: finite state machine, datapath of transitions, VHDL model, hardware expenses, AMD Vivado CAD.

ABBREVIATIONS

CPLD is a complex programmable logic device;
FSM is a finite state machine;
DT is a datapath of transitions;
GSA is a graph-scheme of algorithm;
LUT is a look-up table;
TO – transitions operation.

NOMENCLATURE

A, *X*, *Y* – sets of FSM states, logical conditions and microoperations accordingly;

M, *L*, *N* – number of FSM states, logical conditions and microoperations accordingly;

R – bit depth of state code;

B – number of FSM transitions;

© Barkalov A. A., Titarenko L. A., Babakov R. M., 2023
DOI 10.15588/1607-3274-2023-4-13

OPEN ACCESS

O – set of transitions operations;

I – number of transitions operations;

R_W – bit depth of code of transitions operation;

a_m, $K_1(a_m)$, $K_2(a_m)$ – current state and its scalar and vector codes;

a_s, $K_1(a_s)$, $K_2(a_s)$ – transition state and its scalar and vector codes;

X_h – logical conditions that ensure the transition h;

Y_h – microoperations formed during the transition h;

D_h – signals of code of transition state;

W_h – signals of code of transitions operations.

INTRODUCTION

Digital systems are widely used in human activity [1]. One of the central units of a digital systems is a control unit that coordinates the functioning of all system components [2, 3]. The control unit can be implemented in the form of a finite state machine (FSM), in which the control algorithm is implemented schematically [4, 5]. FSM can be implemented in the form of a Mealy FSM model or a Moore FSM model [2–5]. In comparison with other classes of control units, the FSM is characterized by maximum speed and maximum hardware expenses [2, 3]. Higher hardware expenses worsen such characteristics of the FSM circuit as cost, dimensions, energy consumption, reliability [6]. Therefore, the task of reducing hardware expenses in the finite state machine circuit is an important scientific and practical problem, forming a corresponding scientific direction [1–7].

One of the FSM types is a finite state machine with datapath of transitions (FSM with DT). Its structure includes a special datapath that converts states codes by a set of operations [8]. This approach allows, under certain conditions, to reduce hardware expenses in comparison with other FSM structures.

The design of the circuit of a digital device in the FPGA basis is carried out using specialized CAD based on the VHDL model of the device. At the moment, the problem of developing a VHDL model of the FSM with DT remains unresolved. This complicates the practical application of this class of finite state machines. This paper proposes a solution to the problem of building a VHDL model of an FSM with DT given by a graph-scheme (GSA) of control algorithm.

The object of the study is the automated synthesis of the logic circuit of a finite state machine with datapath of transitions in CAD AMD Vivado according to a VHDL model that corresponds to a given GSA.

The synthesis of a canonical finite state machine can be carried out in automatic mode using the XST tool built into CAD according to the VHDL model recommended by Xilinx [9]. In the case of FSM with DT, a VHDL model should be used, in the synthesis of which the capabilities of the XST tool are not used. One of the features of this model is the assignment of states codes of the FSM in the form of binary constants.

The subject of the study is a VHDL model of a finite state machine with datapath of transitions, which allows both the synthesis of the FSM circuit in the FPGA basis and the verification of the correctness of the functioning of the circuit by means of behavioral modeling in AMD Vivado CAD.

The purpose of the work is the development and research of the structure and methods of building a VHDL model of a finite state machine with datapath of transitions with the aim of systematizing approaches to the automated design of this class of finite state machines in the FPGA basis.

1 PROBLEM STATEMENT

Let us assume that a finite state machine with datapath of transitions is given by the graph-scheme of the algorithm G and is characterized by sets of states $A=\{a_1, ..., a_M\}$, input signals $X=\{x_1, ..., x_L\}$ and microoperations $Y=\{y_1, ..., y_N\}$. The design of the FSM logic circuit involves the implementation of the transition function $T=T(X, T)$ and the output function $Y=Y(X, T)$ in the FPGA element basis using AMD Vivado CAD (until 2023 – Xilinx Vivado CAD). The input data for design is the VHDL model of the designed device, which contains synthesized and behavioral parts and allows obtaining a quantitative value of hardware expenses for the implementation of the circuit of the state machine in a given element basis.

The work solves the problem of developing a VHDL model of an FSM with DT according to a given GSA and its investigation by means of AMD Vivado CAD.

2 REVIEW OF THE LITERATURE

In the modern theory of finite state machines, a wide range of methods for optimizing hardware expenses in the FSM circuit is known. For example, such methods are methods of structural decomposition [7], the essence of which consists in multiple transformation of logical signals, which leads to corresponding changes in the structural diagram of the FSM.

In this article, the method of operational transformation of states codes is considered as a method of hardware expenses optimization [8]. According to it, the conversion of states codes in the system of FSM transitions is carried out not by means of a system of canonical Boolean equations, but by means of a set of arithmetic and logical operations. Combinational circuits that implement these operations form the so-called datapath of transitions (DT). As a result, a structure of FSM with DT is formed, the synthesis of which is discussed in [10].

In paper [11], the justification of the effectiveness of FSM with DT in comparison with the canonical FSM structure according to the criterion of hardware expenses is presented. However, the canonical structure of FSM today has a rather theoretical value, while the practical implementation of FSM circuits is carried out with the help of appropriate CAD software, for example, AMD Vivado CAD. This is primarily due to the use of the FPGA element basis supported by CAD.

© Barkalov A. A., Titarenko L. A., Babakov R. M., 2023
DOI 10.15588/1607-3274-2023-4-13

OPEN ACCESS

Since FSM is often included in designed digital systems, support for its synthesis is implemented at the AMD Vivado CAD level as part of the XST tool [9]. This tool supports several FSM synthesis methods aimed at optimizing various characteristics of the device circuit when implemented in the FPGA basis. Modeling the process of synthesizing the circuit of the state machine allows you to obtain the numerical values of the hardware expenses in the circuit of the device, expressed in the number of used LUT-elements.

The XST synthesis tool, built into the AMD Vivado CAD, is able, under certain conditions, to find code fragments in the VHDL model of the device that correspond to the description of the finite state machine (by state machine we mean a machine with undefined states codes). This process is called finite state machine extraction (FSM extraction). For the found state machine, the XST tool performs the following actions:

– states coding according to the chosen method;

– synthesis of the register circuit in accordance with the chosen method of states encoding;

– synthesis and optimization of the circuit for transition and output functions.

To ensure the possibility of automatic extraction of the state machine, in its VHDL description the following provisions should be observed:

1. The FSM states are specified in the form of a set of literals combined in an element of the enumerated type.

2. The memory register must be synchronous and have the ability to be reset to the initial state by a Reset signal.

3. Implementation of the transition and output function systems is realized using the case operator.

These requirements make it possible to specify an FSM in the form of a VHDL model using one, two or three processes [9, 12–14]. Regardless of how many processes uses the state machine, the XST tool is capable of extracting the state machine from the VHDL code and coding the states according to the chosen coding method.

The disadvantage of using the XST tool is that it is not possible to set specific values of states codes during the FSM synthesis. This makes it impossible to use optimization methods that are based on special coding of states. These methods also include the method of operational transformation of states codes. Therefore, the XST tool cannot be used for the synthesis of an FSM with DT circuit. As a result, the requirements for the VHDL model of the FSM given in [9, 12–14] cannot be directly applied to the FSM with DT and need to be adjusted.

3 MATERIALS AND METHODS

The structural diagram of an FSM with DT is shown in Fig. 1 and contains the following blocks [10].

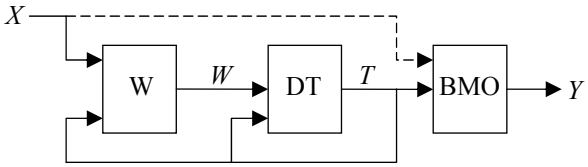

Figure 1 – Structural diagram of an FSM with DT

1. Block DT realizes the following function:

$$T = T(T, W), \quad (1)$$

that is, converts the current state code T to a transition state code using a transitions operation with the W code.

2. Block W realizes the following function:

$$W = W(T, X), \quad (2)$$

that is, it forms transitions operations codes that control the operations of the DT.

3. Block BMO realizes the function

$$Y = Y(X, T) \quad (3)$$

in the case of Mealy FSM or function

$$Y = Y(T) \quad (4)$$

in the case of a Moore FSM, that is, it provides the implementation of FSM output function. In fig. 1, the presence of a connection marked with a dashed line allows you to consider the structure as a Mealy FSM, the absence of a connection – as a Moore FSM.

The internal structure of the DT is shown in Fig. 2 [10].

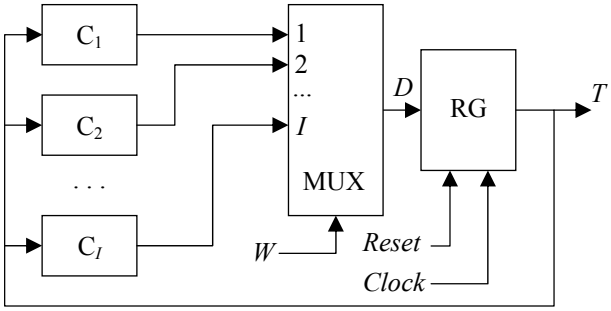

Figure 2 – Internal structure of DT

Blocks $C_1 – C_I$ correspond to combinational circuits implementing a set of transitions operations (TO) $O=\{O_1, ..., O_I\}$. In general, each TO can be arithmetic, logical or combined. When designing these blocks, if possible, each block should be optimized in order to increase performance and reduce hardware expenses.

The MUX block is an R-bit multiplexer with I-directions. Under the guidance of the W signals at the output of the multiplexer, the R-bit code D of the next FSM state is formed, which enters the input of the memory register RG.

The RG block is an R-bit synchronous register with the function of resetting to the initial state by the *Reset* signal. It should be noted that in the case of FSM with DT, the initial state does not necessarily have a zero code. This register performs the function of the memory of the datapath of transitions and the function of the memory register of the finite state machine.

Let FSM be given by GSA G (Fig. 3). This GSA is marked by states of Moore FSM and contains the set of states of the states $A=\{a_0, ..., a_{20}\}$ with cardinality $M=21$, the set of logical conditions $X=\{x_1, x_2, x_3\}$ with cardinality $L=3$, multiple set of microoperations (output signals)

© Barkalov A. A., Titarenko L. A., Babakov R. M., 2023
DOI 10.15588/1607-3274-2023-4-13

OPEN ACCESS

$Y=\{y_1, ..., y_7\}$ with cardinality $N=7$ and $B=29$ FSM transitions. GSA has an abstract structure and content of operator vertices and is intended to demonstrate the process of building a VHDL model of an FSM with DT.

The main and most difficult stage of the synthesis of an FSM with DT is the so-called algebraic synthesis of FSM. In the process of algebraic synthesis, the following occurs [10]:

1. The FSM states are matched with unique codes from a certain set of states codes. In the case of GSA G, $R=5$ binary digits are enough to encode $M=21$ states.

2. FSM transitions correspond to certain transitions operations from a given set of TOs. The use of one TO for the implementation of several state machine transitions is permissible and contributes to the reduction of the total number of used TOs and, accordingly, to the reduction of hardware expenses in the FSM circuit. Those transitions that cannot be implemented by any of the specified TOs should be implemented in a canonical way using a system of Boolean equations.

We will carry out an algebraic synthesis for GSA G under the condition that the set of transitions operations is formed by the following ones: $O = \{O_1, O_2, O_3\}$:

$$O_1: D = T + 7_{10}; \quad (5)$$

$$O_2: D = T \,\&\, 01001_2; \quad (6)$$

$$O_3: D = T \oplus 00011_2. \quad (7)$$

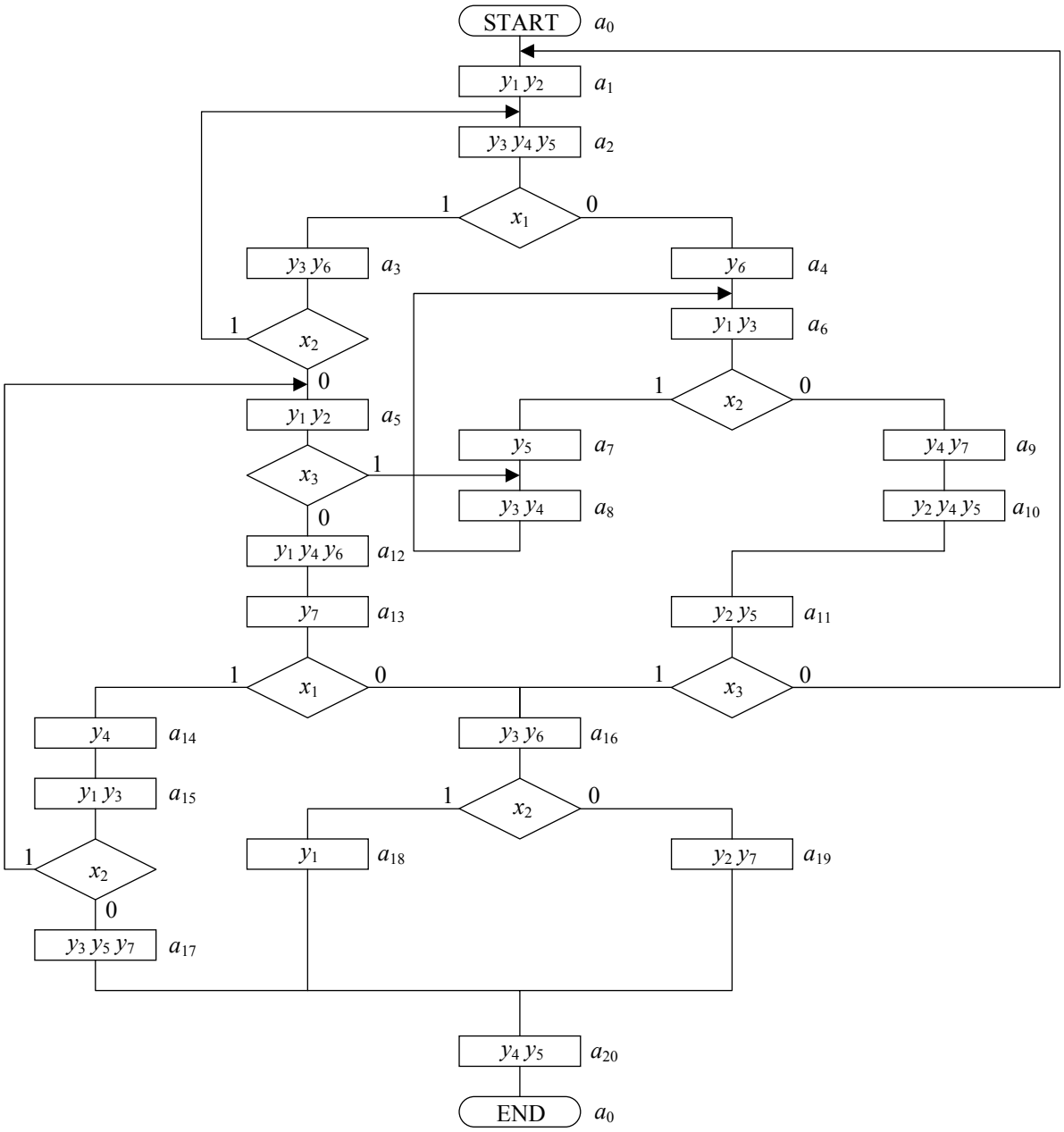

Figure 3 – Graph-scheme of algorithm G

© Barkalov A. A., Titarenko L. A., Babakov R. M., 2023
DOI 10.15588/1607-3274-2023-4-13

In these expressions, T is the code of the current FSM state, D is the code of the state of the transition, which is formed at the output of the multiplexer and enters the RG (Fig. 2).

TO O_1 corresponds to the operation of adding the decimal constant 7 to the code of the current state. The code of the current state T is interpreted as an unsigned decimal number of 5 binary digits size. The operation is implemented on the basis of a 5-digit binary adder, in which the carry from the higher bit is discarded. This is equivalent to the operation "$(T + 7)$ mod 32". For example, (25+10) mod 32 = 3.

TO O_2 is a bitwise logical conjunction operation on the binary value of the current state code T and the binary constant 01001.

TO O_3 is a bitwise logical operation XOR on the binary value of the current state code T and the binary constant 00011.

In general, some FSM transitions can be implemented in a canonical way without using the specified transitions operations. The circuit that implements all such transitions will act as a separate combinational circuit C_i as part of the DT (Fig. 2). In order for the multiplexer to be able to pass through the result of the operation of this circuit, we must consider it as a separate TO O_4, which has its own code. Formally, O_4 is some function Φ of the code of the current state of the automaton:

$$O_4: T = \Phi(T). \quad (8)$$

Successful execution of algebraic synthesis gives us a formal solution of the algebraic synthesis problem [10]. In general, there may be several formal solutions. As an example, consider the formal solution shown in Fig. 4 (the method of obtaining it is not considered in this paper).

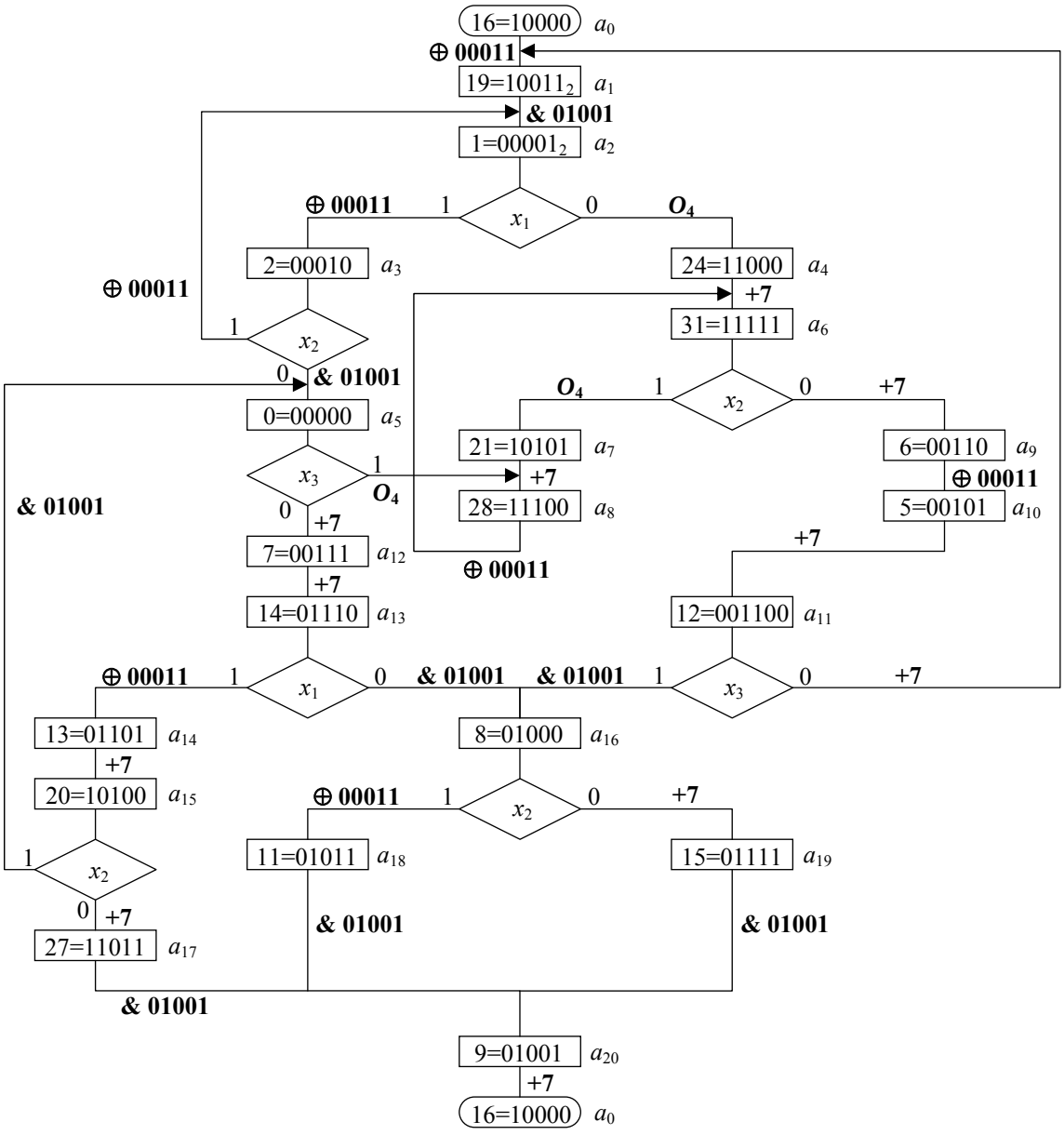

Figure 4 – Formal solution of problem of algebraic synthesis of an FSM with DT for GSA G (example)

© Barkalov A. A., Titarenko L. A., Babakov R. M., 2023
DOI 10.15588/1607-3274-2023-4-13
OPEN ACCESS

Since the operational transformation of states codes does not affect the FSM output function, in Fig. 4 inside operational vertices, microoperations are not shown. Instead of them, selected decimal values of states codes and their binary equivalents are shown. Each FSM transition is marked by a transitions operation that is mapped to it. Operation O_1 is marked with "+7", O_2 – "& 01001", O_3 – "⊕ 00011" for clarity. Transitions implemented in the canonical way ($a_2 \to a_4$, $a_5 \to a_5$, $a_6 \to a_7$) are marked with the symbol O_4.

Let's explain the operational implementation of transitions in Fig. 4. Transition from state a_0 coded with $K(a_0)=16_{10}=10000_2$ to state a_1 coded with $K(a_1)=19_{10}=10011_2$ is carried out using O_3. Transition from state a_{16} coded with $K(a_{16})=8_{10}=01000_2$ to state a_{19} coded with $K(a_{19})=15_{10}=01111_2$ is carried out using O_1. Transition from state a_3 coded with $K(a_3)=2_{10}=00010_2$ to state a_5 coded with $K(a_5)=0_{10}=00000_2$ is carried out using O_2.

Let's encode operations of transitions $O_1 - O_4$ with unique binary codes of bit depth $R_W = \lceil \log_2 4 \rceil = 2$, formed by variables $W = \{w_1, w_2\}$. The result of coding is presented in Table 1.

Table 1 – Coding of transitions operations

O_i	$w_1\ w_2$
O_1	0 0
O_2	0 1
O_3	1 0
O_4	1 1

Let us present the result of algebraic synthesis in the form of an operational table of transitions (OTT) [10], which for our example has the form of a Table 2.

In the Table 4, each row corresponds to a separate FSM transition, the number of which is indicated in column h. The W_h column contains codes of transitions operations according to the table 1. In each cell of the W_h column, only those variables w that are equal to 1 in the code of the corresponding TO, are shown. For example, the transition h=5 is implemented using operation O_3 with binary code 10 (w_1=1, w_2=0), so in the row h=5 in the column W_h, only the variable w_1 is indicated, which is equal to 1 in binary code 10.

Let's proceed to the construction of the VHDL model of the FSM with DT, the OTT of which corresponds to the table. 2. We will present the model in the form of synthesizable and behavioral parts [5, 12, 14]. Consider the description of the synthesized part.

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	16	10000	a_1	19	10011	1	w_1	–	1
a_1	19	10011	a_2	1	00001	1	w_2	$y_1\ y_2$	2
a_2	1	00001	a_3	2	00010	x_1	w_1	$y_3\ y_4\ y_5$	3
			a_4	24	11000	$\bar{x}_1$	$w_1\ w_2$		4
a_3	2	00010	a_1	1	00001	x_2	w_1	$y_3\ y_6$	5
			a_5	0	00000	$\bar{x}_2$	w_2		6
a_4	24	11000	a_6	31	11111	1	–	y_6	7
a_5	0	00000	a_8	28	11100	x_3	$w_1\ w_2$	$y_1\ y_2$	8
			a_{12}	7	00111	$\bar{x}_3$	–		9
a_6	31	11111	a_7	21	10101	x_2	$w_1\ w_2$	$y_1\ y_3$	10
			a_9	6	00110	$\bar{x}_2$	–		11
a_7	21	10101	a_8	28	11100	1	–	y_5	12
a_8	28	11100	a_6	31	11111	1	w_1	$y_3\ y_4$	13
a_9	6	00110	a_{10}	5	00101	1	w_1	$y_4\ y_7$	14
a_{10}	5	00101	a_{11}	12	01100	1	–	$y_2\ y_4\ y_5$	15
a_{11}	12	01100	a_{16}	8	01000	x_3	w_2	$y_2\ y_5$	16
			a_1	19	10011	$\bar{x}_3$	–		17
a_{12}	7	00111	a_{13}	14	01110	1	–	$y_1\ y_4\ y_6$	18
a_{13}	14	01110	a_{14}	13	01101	x_1	w_1	y_7	19
			a_{16}	8	01000	$\bar{x}_1$	w_2		20
a_{14}	13	01101	a_{15}	20	10100	1	–	y_4	21
a_{15}	20	10100	a_5	0	00000	x_2	w_2	$y_1\ y_3$	22
			a_{17}	27	11011	$\bar{x}_2$	–		23
a_{16}	8	01000	a_{18}	11	01011	x_2	w_1	$y_3\ y_6$	24
			a_{19}	15	01111	$\bar{x}_2$	–		25
a_{17}	27	11011	a_{20}	9	01001	1	w_2	$y_3\ y_5\ y_7$	26
a_{18}	11	01011	a_{20}	9	01001	1	w_2	y_1	27
a_{19}	15	01111	a_{20}	9	01001	1	w_2	$y_2\ y_7$	28
a_{20}	9	01001	a_0	16	10000	1	–	$y_4\ y_5$	29

© Barkalov A. A., Titarenko L. A., Babakov R. M., 2023
DOI 10.15588/1607-3274-2023-4-13
OPEN ACCESS
CC BY SA

```
entity FSM is
    generic(R: integer := 5;                -- State code capacity
            Rw: integer := 2;               -- W code capacity
            L: integer := 3;                -- Number of input signals
            N: integer := 5);               -- Number of microoperations
    port (X: in std_logic_vector(1 to L);   -- Input signals
          Y: out std_logic_vector(1 to N);  -- Microoperations
          C: in std_logic;                  -- Clock
          Reset: in std_logic);             -- Reset
end FSM;
```

In the "generic" section, the setting constants that determine the bit depth of the signal buses are defined. The "port" section contains the bus of input signals *X*, the bus of output microoperations *Y*, the synchronization signal *Clock* and the *Reset* signal, by which the code of the initial state of the FSM is written into the memory register.

The architecture section contains a description of the internal FSM signals, as well as a description of the structural blocks in the view of processes in accordance with Fig. 1 and 2. The beginning of the description of the architecture block looks like next:

```
architecture FSM_A of FSM is

signal T, D: unsigned(1 to R);         -- State code and Next state code
signal Canonic : unsigned(1 to R);     -- Result of 'canonical' transitions
signal nT: unsigned(1 to R);           -- Negative values of State code
signal nX: std_logic_vector(1 to L);   -- Negative values of input signals
signal W: std_logic_vector(1 to Rw);   -- Code of datapath operation

begin

nT <= not T;
nX <= not X;
```

Here, the "Canonic" signal is the code of the next state, formed in a canonical way. Its use will be discussed later.

Below a process block describing the FSM memory register is shown. The register switches synchronously with the rising edge of the *Clock* signal.

```
process(C) -- Memory Register
begin
    if rising_edge(C) then
        if Reset = '1' then
            T <= "10000";
        else
            T <= D;
        end if;
    end if;
end process;
```

The peculiarity of this description is that by a *Reset* signal equal to one, the register is transferred to the initial state, the code of which, according to the results of algebraic synthesis (Table 2), is equal to 10000_2.

Let's synthesize the block W, which forms the signals w_1, w_2 of the transitions operation code (Fig. 1). We implement these signals using canonical Boolean equations according to the table 2 and expression (2).

$$w_1 = a_0 \vee a_2 x_1 \vee a_3 x_2 \vee a_5 x_3 \vee a_6 x_2 \vee a_8 \vee a_9 \vee \\ \vee a_{13} x_1 \vee a_{16} x_2;$$

$$w_2 = a_1 \vee a_2 \bar{x}_1 \vee a_3 \bar{x}_2 \vee a_6 x_2 \vee a_{11} x_3 \vee a_{13} \bar{x}_1 \vee \\ \vee a_{15} x_2 \vee a_{17} \vee a_{18} \vee a_{19}.$$

We will use binary vectors $<T_1, ..., T_5>$ to represent the FSM states codes. Then, according to the coding of the states given in the Table 2, the Boolean equations for signals w_1, w_2 take the following form:

$$w_1 = T_1\bar{T}_2\bar{T}_3\bar{T}_4\bar{T}_5 \vee \bar{T}_1\bar{T}_2\bar{T}_3\bar{T}_4T_5x_1 \vee \bar{T}_1\bar{T}_2\bar{T}_3T_4\bar{T}_5x_2 \vee \\ \vee \bar{T}_1\bar{T}_2\bar{T}_3\bar{T}_4\bar{T}_5x_3 \vee T_1T_2T_3T_4T_5x_2 \vee T_1T_2T_3\bar{T}_4\bar{T}_5 \vee \\ \vee \bar{T}_1\bar{T}_2T_3T_4\bar{T}_5 \vee \bar{T}_1T_2T_3T_4\bar{T}_5x_1 \vee \bar{T}_1T_2\bar{T}_3\bar{T}_4\bar{T}_5x_2;$$

$$w_2 = T_1\bar{T}_2\bar{T}_3T_4T_5 \vee \bar{T}_1\bar{T}_2\bar{T}_3\bar{T}_4T_5\bar{x}_1 \vee \bar{T}_1\bar{T}_2\bar{T}_3T_4\bar{T}_5\bar{x}_2 \vee \\ \vee T_1T_2T_3T_4T_5x_2 \vee \bar{T}_1T_2T_3\bar{T}_4\bar{T}_5x_3 \vee \bar{T}_1T_2T_3T_4\bar{T}_5\bar{x}_1 \vee \\ \vee T_1\bar{T}_2T_3\bar{T}_4\bar{T}_5x_2 \vee T_1T_2\bar{T}_3T_4T_5 \vee \bar{T}_1T_2\bar{T}_3T_4T_5 \vee \\ \vee \bar{T}_1T_2T_3T_4T_5.$$

In general, these equations can be minimized in any convenient way. In this paper, we will not perform minimization and will immediately present block W in the form of the following VHDL process:

© Barkalov A. A., Titarenko L. A., Babakov R. M., 2023
DOI 10.15588/1607-3274-2023-4-13
OPEN ACCESS

```
process (T, X, nT, nX) -- Block W
    begin
        W(1) <= (T(1) and nT(2) and nT(3) and nT(4) and nT(5)) or
                (nT(1) and nT(2) and nT(3) and nT(4) and T(5)) or
                (nT(1) and nT(2) and nT(3) and T(4) and nT(5) and X(2)) or
                (nT(1) and nT(2) and nT(3) and nT(4) and nT(5) and X(3)) or
                (T(1) and T(2) and T(3) and T(4) and T(5) and X(2)) or
                (T(1) and T(2) and T(3) and nT(4) and nT(5)) or
                (nT(1) and nT(2) and T(3) and T(4) and nT(5)) or
                (nT(1) and T(2) and T(3) and T(4) and nT(5) and X(1)) or
                (nT(1) and T(2) and nT(3) and nT(4) and nT(5) and X(2));

        W(2) <= (T(1) and nT(2) and nT(3) and T(4) and T(5)) or
                (nT(1) and nT(2) and nT(3) and nT(4) and T(5) and nX(1)) or
                (nT(1) and nT(2) and nT(3) and T(4) and nT(5) and nX(2)) or
                (nT(1) and nT(2) and nT(3) and nT(4) and nT(5) and X(3)) or
                (T(1) and T(2) and T(3) and T(4) and T(5) and X(2)) or
                (nT(1) and T(2) and T(3) and nT(4) and nT(5) and X(3)) or
                (nT(1) and T(2) and T(3) and T(4) and nT(5) and nX(1)) or
                (T(1) and nT(2) and T(3) and nT(4) and nT(5) and X(2)) or
                (T(1) and T(2) and nT(3) and T(4) and T(5)) or
                (nT(1) and T(2) and nT(3) and T(4) and T(5)) or
                (nT(1) and T(2) and T(3) and T(4) and T(5));
end process;
```

Please note that to ensure the correctness of the simulation, the process sensitivity list contains both direct and inverse values of the X and T signals.

Let's proceed to the synthesis of the datapath of transition. Let's clarify the structure of the DT shown in Fig. 2, according to the results of algebraic synthesis. The clarification of the structure consists in the fact that it contains four combinational circuits $C_1 - C_4$, which correspond to transitions operations $O_1 - O_4$, and the multiplexer is controlled by a two-bit binary code $W=<w_1, w_2>$.

As will be shown below, combinational circuits $C_1 - C_3$ have a trivial implementation using operators from the synthesizable subset of VHDL. However, block C_4 is non-standard, as it represents a canonical implementation of a certain part of an FSM transitions. In the general case, the code of the current state T and signals of logical conditions X are received at its inputs. So the refined structure of the OAP for GSA G is shown in Fig. 5.

Before developing the VHDL description of the DT, let's synthesize the C_4 block. For this purpose, we will use the technique discussed in [2, 3].

Transitions implemented in the canonical way, in Table 2, have numbers 4, 8 and 10. Let's do the following.

1. Let's agree to use the Boolean vector $<D_1, ..., D_5>$ to encode the transition state code.

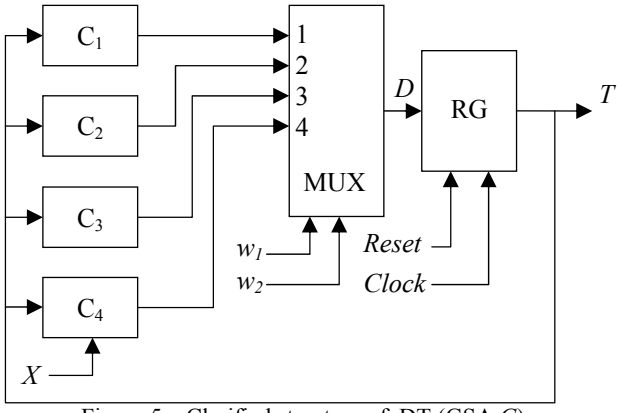

Figure 5 – Clarified structure of DT (GSA G)

2. Let's form a separate table of transitions from these lines, in which instead of the column W_h there is a column D_h. This column indicates those components of the vector $<D_1, ..., D_5>$ which are equal to 1 in the binary code of the transition state $K_2(a_s)$ of this row of the table.

As a result, we will get a table of transitions implemented in the canonical way (Table 3).

Table 3 – Table of transitions implemented in the canonical way (GSA G)

a_m	$K_1(a_m)$	$K_2(a_m)$	a_s	$K_1(a_s)$	$K_2(a_s)$	X_h	D_h	Y_h	h
a_2	1	00001	a_4	24	11000	$\bar{x}_1$	$D_1 D_2$	$y_3\ y_4\ y_5$	1
a_5	0	00000	a_8	28	11100	x_3	$D_1 D_2 D_3$	$y_1\ y_2$	2
a_6	31	11111	a_7	21	10101	x_2	$D_1 D_3 D_5$	$y_1\ y_3$	3

Each of the three transitions presented in the Table 3, corresponds to the own term formed by the conjunction of signals T_1, ..., T_5 of the code of the current state (column $K_2(a_m)$) and the corresponding signal of the logical condition (column X_h). Let's form these terms:

$$Q_1 = \overline{T}_1\overline{T}_2\overline{T}_3\overline{T}_4T_5\overline{x}_1;$$

$$Q_2 = \overline{T}_1\overline{T}_2\overline{T}_3\overline{T}_4\overline{T}_5x_3;$$

$$Q_3 = T_1T_2T_3T_4T_5x_2.$$

Now we can form the equation for signals $D_1 - D_5$ according to the contents of column D_h. Since the D_4 signal is not present in the D_h column, we will consider it always equal to 0.

$$D_1 = Q_1 \vee Q_2 \vee Q_3;$$

$$D_2 = Q_1 \vee Q_2;$$

$$D_3 = Q_2 \vee Q_3;$$

$$D_4 = 0;$$

© Barkalov A. A., Titarenko L. A., Babakov R. M., 2023
DOI 10.15588/1607-3274-2023-4-13

$$D_5 = Q_3 .$$

The VHDL code for the combinational circuit C_4 is given below. As you can see, the vector $<D_1, ..., D_5>$ in this model corresponds to the "Canonic" signal. Although this signal is declared as "unsigned (1 to R) ", each bit is generated separately. Also, pay attention to the presence in the process sensitivity list of both direct and inverse values of T and X signals.

```
        process (T, X, nT, nX) -- Canonical transitions
            variable Q1: std_logic;
            variable Q2: std_logic;
            variable Q3: std_logic;
        begin
            Q1 := nT(1) and nT(2) and nT(3) and nT(4) and T(5) and nX(1);
            Q2 := nT(1) and nT(2) and nT(3) and nT(4) and nT(5) and X(3);
            Q3 := T(1) and T(2) and T(3) and T(4) and T(5) and X(2);

            Canonic(1) <= Q1 or Q2 or Q3;
            Canonic(2) <= Q1 or Q2;
            Canonic(3) <= Q2 or Q3;
            Canonic(4) <= '0';
            Canonic(5) <= Q3;
        end process;
```

Although the combinational circuit C_4 is described as a separate process, structurally it is part of the DT block (Fig. 5). The description of the DT block in VHDL is as follows:

```
process (T, W, Canonic) -- Datapath
begin
    case W is
        when "00" =>                -- O1
            D <= T + 7;
        when "01" =>                -- O2
            D <= T and "01001";
        when "10" =>                -- O3
            D <= T xor "00011";
        when "11" =>                -- O4
            D <= Canonic;
        when others =>
            D <= "00000";
    end case;
end process;
```

The basis of this process is the "case" operator, which has four branches corresponding to operation codes of transitions $O_1 - O_4$. Operations $O_1 - O_4$ are implemented with the help of "+", "and" and "xor" operators, which are included in the synthesized subset of VHDL and can work directly with the "unsigned" data type. In the case of O_4, to the output bus D the result obtained from the output of the combinational circuit C_4 (input signal "Canonic") is passed through.

The following fragment of the VHDL code describes the BMO block that forms FSM output signals according to (4). The method of describing this block is not fundamental and can be implemented both with the help of the "case" operator and by setting a system of canonical Boolean equations by analogy with the considered blocks W and C_4. Also, this block can be synthesized using various methods of output function optimization [2–4, 7].

```
process (T, nT)
begin
    case T is
        when "10011" => Y <= "1100000";
        when "00001" => Y <= "0011100";
        when "00010" => Y <= "0010010";
        when "11000" => Y <= "0000010";
        when "00000" => Y <= "1100000";
        when "11111" => Y <= "1010000";
        when "10101" => Y <= "0000100";
        when "11100" => Y <= "0011000";
        when "00110" => Y <= "0001001";
        when "00101" => Y <= "0101100";
        when "01100" => Y <= "0100100";
        when "00111" => Y <= "1001010";
        when "01110" => Y <= "0000001";
        when "01101" => Y <= "0001000";
        when "10100" => Y <= "1010000";
        when "01000" => Y <= "0010010";
        when "11011" => Y <= "0010101";
        when "01011" => Y <= "1000000";
        when "01111" => Y <= "0100001";
        when "01001" => Y <= "0001100";
        when others =>  Y <= "0000000";
    end case;
end process;
```

The fragments of the VHDL code considered above form a synthesizable part of the VHDL model of the FSM with DT. For the correct functioning of the model, its first lines should be lines connecting the necessary libraries:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
```

As the last line, you should add the architecture block completion statement:

```
end FSM_A;
```

To check the correctness of the considered VHDL model, it is necessary to develop its behavioral part. The function of the behavioral part in our case is the generation of external signals and their supply to the FSM inputs. Time intervals of signals generation in this case are of no fundamental importance, since behavioral modeling will take place without reference to the physical characteristics of the device.

The behavioral part can be described by the following VHDL code fragment:

© Barkalov A. A., Titarenko L. A., Babakov R. M., 2023
DOI 10.15588/1607-3274-2023-4-13
OPEN ACCESS

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity Model is
    generic(R: integer := 5;                          -- State code capacity
            Rw: integer := 2;                         -- W code capacity
            N: integer := 7;                          -- Number of microoperations
            L: integer := 3);                         -- Number of logical conditions
    port (Y: out std_logic_vector(1 to N));           -- Microoperations
end Model;

architecture Model_A of Model is
signal C: std_logic;                                  -- Clock
signal Reset: std_logic;                              -- Reset
signal X: std_logic_vector(1 to L);                   -- Logical conditions

component FSM is
    generic(R: integer := 5;                          -- State code capacity
            Rw: integer := 2;                         -- W code capacity
            L: integer := 3;                          -- Number of logical conditions
            N: integer := 7);                         -- Number of microoperations
    port (X: in std_logic_vector(1 to L);             -- Input signals
          Y: out std_logic_vector(1 to N);            -- Microoperations
          C: in std_logic;                            -- Clock
          Reset: in std_logic);                       -- Reset
end component FSM;

begin
        process                                                         -- Clock
        begin
           C <= '0'; wait for 80 ns;
           C <= '1'; wait for 20 ns;
        end process;

        Reset <= '0' after 0 ns, '1' after 10 ns, '0' after 90 ns;     -- Reset

        process                                                         -- X1
        begin
           X(1) <= '1'; wait for 17 ns; X(1) <= '0'; wait for 37 ns;
       end process;

        process                                                         -- X2
        begin
           X(2) <= '1'; wait for 43 ns; X(2) <= '0'; wait for 36 ns;
       end process;

        process                                                         -- X3
        begin
           X(3) <= '1'; wait for 38 ns; X(3) <= '0'; wait for 17 ns;
       end process;

        L1: component FSM
            port map (X, Y, C, Reset);

end Model_A;
```

The behavioral part has the following features:

1. Microoperations formed by the FSM are displayed on the output port *Y*.

2. The *Clock* signal has an interval of 100 ns. The *Reset* signal is generated once at the start of the device's functioning.

3. Signals *X* are formed in separate processes, which makes them independent of each other. The intervals of the upper and lower levels are random and can have any values.

4 EXPERIMENTS

For the developed VHDL model of FSM with DT, the authors conducted experimental research with the help of CAD AMD Vivado version 2023.1 (Vivado ML Standard Edition, free version). The research sets two goals:

1. Checking the correctness of the work of the synthesized FSM circuit using behavioral modeling.

2. Checking the possibility of synthesis of MPA logic circuit in FPGA basis.

Achieving the first goal will allow us to consider the proposed approach to building a VHDL model of an FSM with DT correct. Achieving the second goal will confirm the possibility of using the developed model to evaluate

© Barkalov A. A., Titarenko L. A., Babakov R. M., 2023
DOI 10.15588/1607-3274-2023-4-13
OPEN ACCESS

the effectiveness of FSM with DT according to the criterion of hardware expenses [11].

5 RESULTS

Behavioral modeling of the developed model of FSM with DT for GSA *G* was performed in AMD Vivado CAD using standard modeling parameters. A fragment of the timing diagram of the state machine is shown in Fig. 6. Signals *T* and *D* are in unsigned decimal format, other signals are in binary format. Three markers are set on the diagram, which allow to analyze important moments of time in functioning of the FSM. Let's consider them.

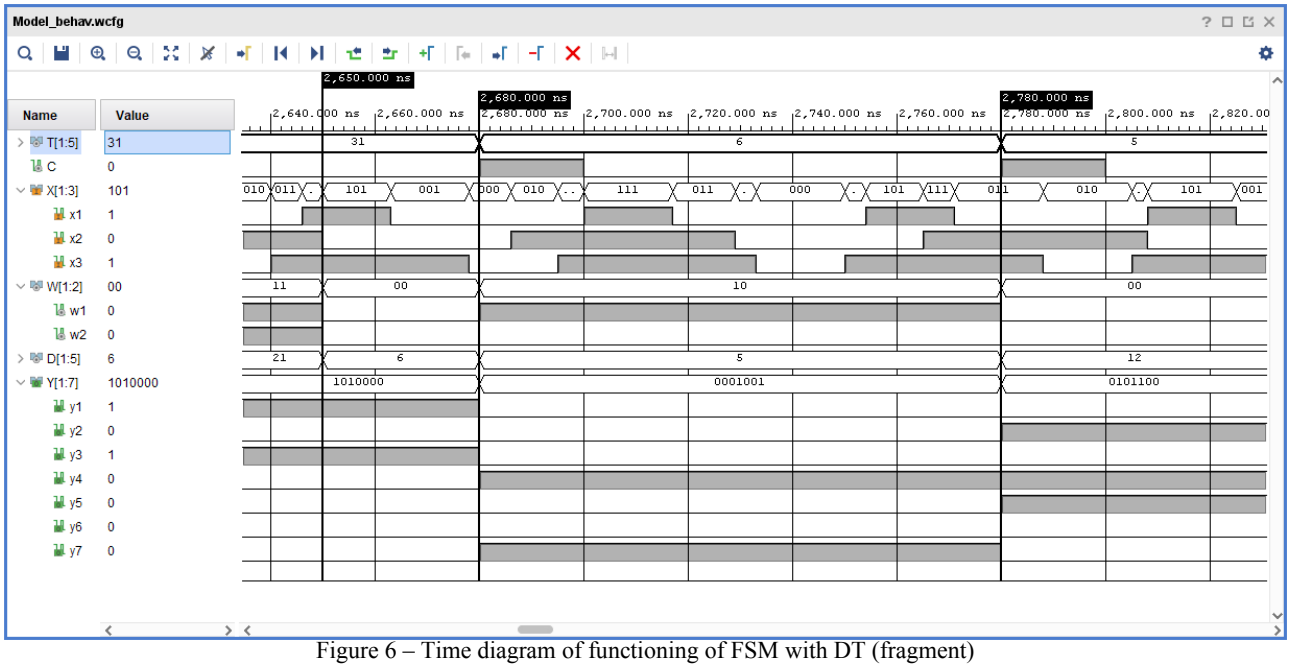

Figure 6 – Time diagram of functioning of FSM with DT (fragment)

Until the moment t_1=2650 *ns*, the FSM is in the state with code $T=31_{10}=11111_2$ (state a_6) and forms microoperations y_1, y_3. At time t_1, signal x_2 becomes equal to 0, which sets the values of signals w_1 and w_2 into zero values. This results to executing of operation O_1:

$$D = 31_{10} + 7_{10} = (38 \bmod 32)_{10} = 6_{10} = 00110_2.$$

This value is formed on bus *D* and is the state code of the transition of the FSM in the next cycle of functioning.

At the moment t_2=2680 *ns*, the rising edge of the synchronization signal *C* arrives. Following this signal, the value $D=6_{10}=00110_2$ is loaded into the memory register and appears on the bus *T*. Thus, the FSM correctly transitioned from state a_6 with code 31_{10} to state a_9 with code 6_{10}. Also, with a change in *T*, there is a change in the output signals: microoperations y_4, y_7 are formed on the *Y* bus. This coincides with Fig. 3 and confirms the correct functioning of the BMO circuit.

At the moment t_3=2780 *ns*, the FSM after the rising edge of signal *C* passes from the state with code $T=6_{10}=00110_2$ (state a_9) to the state with code $D=5_{10}=00101_2$ (state a_{10}). The transformation of the state code proceeds using the operation O_3: $00110_2 \oplus 00011_2 = 00101_2$. Since this transition is unconditional, the operation code $W(O_3)$=<10> is formed on the *W* bus immediately after the FSM transition to the state $T=00110_2$ (starting from the moment t_2) and does not change when the values of signals $x_1 - x_3$ change. Also, microoperations y_2, y_4, y_5 are formed at time t_4. This corresponds to state a_{10} in Fig. 3 and to transition h=15 in table 2.

Thus, it can be concluded that the developed VHDL model of FSM with DT is correct and corresponds to the given graph-scheme of algorithm *G*.

Let's check the possibility of synthesizing the developed VHDL model in the FPGA basis. Experiments have shown that stages of synthesis and implementation in the FPGA chip xc7a12tcpg238-1 occur without errors. As a result of the synthesis, the numerical values of the hardware expenses for the implementation of the synthesizable part of the VHDL model of the FSM with DT were obtained, consisting of 16 LUT elements and 5 triggers. Thus, the developed VHDL model can be used to evaluate the efficiency of the FSM circuit according to the criterion of hardware expenses.

6 DISCUSSION

The method of operational transformation of states codes, which is the base of the structure of FSM with DT, provides for special coding of states of the FSM. Values of states codes, selected transitions operations and their mapping to FSM transitions form a full picture, which is called a formal solution to the problem of algebraic synthesis of an FSM with DT. On the one hand, the special coding of states makes it impossible to use the finite state machine synthesis tools built into the XST AMD Vivado CAD [9]. On the other hand, knowing the specific values of states codes allows you to apply your own optimization methods aimed at optimizing hardware expenses in various structural blocks of an FSM with DT.

© Barkalov A. A., Titarenko L. A., Babakov R. M., 2023
DOI 10.15588/1607-3274-2023-4-13

Thus, the application of structure of FSM with DT requires the development of its own VHDL model, which differs from the models recommended by AMD Vivado CAD developers. The example considered in this paper demonstrates the following features of the approach to building such a model:

1. In the FSM with DT it is allow the use of any arithmetic and logical operations. They can provide a different interpretation to the states codes of the FSM – scalar (numeric) or vector (binary). Accordingly, the VHDL model must also support different ways of interpreting the states codes. In the considered example, the "unsigned" data type is used to represent states codes, which allows performing both arithmetic (scalar) and logical (vector) operations on states codes. If necessary, it will allow the use of other scalar types available in the VHDL language libraries. For example, to work with signed numbers, the "integer" data type can be used. In cases where the set of TOs contains only vector operations, it is sufficient to use only vector data type (such as "std_logic_vector") for states codes.

2. In the considered example, the combinational circuit C_4 implements FSM transitions, which are implemented in a canonical way according to the system of Boolean equations. The necessity of using such a circuit is due only to the results of the algebraic synthesis carried out by the authors for a given GSA *G*. In general, situations are possible when all FSM transitions are implemented using a given set of TOs. In this case, there will be no need for a circuit like C_4.

3. The number of combinational circuit reflects the number of different transitions operations. In our case, it is a number of lines of the "case" operator, which corresponds to the multiplexer in Fig. 2. In general, the smaller number of combinational circuit in the datapath (that is, the smaller number of different TOs) corresponds to decrease in hardware expenses in the FSM with DT circuit.

4. Algebraic synthesis of FSM with DT demands the formation of set of transitions operations. Such a formation can occur both at the beginning of algebraic synthesis and during of its execution. When forming the set of OT, it is necessary to ensure that these operations can be implemented with the help of operators from the synthesizable subset of the VHDL language. If there is a need to implement a non-standard operation (scalar or vector), you can use an approach similar to the development of the C_4 circuit in the above example.

5. FSM with DT belongs to the class of FSM with "hardware" logic. It should be understood that, in general, any change in the input data (the GSA, the set of TOs, the bit depth of states codes, etc.) requires a complete re-synthesis of the FSM and its VHDL model. This applies to the circuits of any custom digital devices.

CONCLUSIONS

The paper proposes a solution to the scientific problem of developing a VHDL model of a finite state machine with datapath of transitions. The correctness of the model was checked in AMD Vivado CAD.

The scientific novelty of the work lies in the fact that all stages of development of a VHDL model are demonstrated on a specific example, which allows you to understand its peculiarities and differences from typical models of finite state machines. The main feature of the proposed model is that it is not focused on the use of finite state machine synthesis tools built into CAD and can be used in CADs of different FPGA manufacturers.

The practical use of the obtained results is possible in the development of methods of synthesis, optimization and evaluation of the efficiency of a finite state machines with datapath of transitions, as well as other structures and methods aimed at optimizing the characteristics of an FSM circuit.

Prospects for further research consist in solving a range of scientific and practical problems related to the development, implementation and evaluation of the effectiveness of the structures and methods of synthesis of finite state machines with optimized hardware expenses.

ACKNOWLEDGEMENTS

The paper 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. Xilinx. XST UserGuide. V.11.3. Available online: https://www.xilinx.com/support/documentation/sw_manuals/xilinx11/xst.pdf (accessed on 12 April 2023).
10. 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, Volume 3 (62), pp. 109–119.
11. 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.
12. Czerwinski R., Kania D. Finite State Machine Logic Synthesis for Complex Programmable Logic Devices. Berlin, Springer, 2013, 172 p.
13. Mano M. Digital design (4th Edition). New Jersey, Prentice Hall, 2006, 624 p.
14. Zwolinski M. Digital System Design with VHDL. Boston, Addison-Wesley Longman Publishing Co., Inc. 2000, 416 p.

Received 04.10.2023.
Accepted 30.11.2023.

© Barkalov A. A., Titarenko L. A., Babakov R. M., 2023
DOI 10.15588/1607-3274-2023-4-13

OPEN ACCESS

УДК 004.94 : 004.2

СИНТЕЗ VHDL-МОДЕЛІ МІКРОПРОГРАМНОГО АВТОМАТА З ОПЕРАЦІЙНИМ АВТОМАТОМ 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, професор кафедри інформаційних технологій Донецького національного університету імені Василя Стуса, м. Вінниця, Україна.

АНОТАЦІЯ

Актуальність. Розглянуто задачу побудови програмної моделі мікропрограмного автомата з операційним автоматом переходів мовою VHDL. Процес синтезу моделі ототожнюється із синтезом даного типу автомата, оскільки побудована модель може бути використана як для аналізу поведінки пристрою, так і для синтезу логічної схеми в базисі FPGA. Об'єктом дослідження є автоматизований синтез логічної схеми мікропрограмного автомата з операційним автоматом переходів, за результатами якого можуть бути отримані чисельні характеристики апаратурних витрат на реалізацію схеми автомата. Це дозволяє оцінити ефективність використання даної структури мікропрограмного автомата при реалізації заданого алгоритму керування.

Мета. Розробка і дослідження VHDL-моделі мікропрограмного автомата з операційним автоматом переходів для аналізу поведінки автомата та кількісної оцінки апаратурних витрат в його логічній схемі.

Метод. В основу дослідження покладено структурну схему мікропрограмного автомата з операційним автоматом переходів. Синтез окремих блоків структури автомата здійснюється за певною процедурою відповідно до заданої граф-схеми алгоритму керування. Результат синтезу запропоновано представляти у вигляді VHDL-опису, що оснований на фіксованих значеннях кодів станів автомата. Продемонстрований процес синтезу операційного автомата переходів, блоку формування кодів операцій переходів та блоку формування мікрооперацій. VHDL-опис даних блоків здійснюється у синтезованому стилі, що дозволяє провести синтез логічної схеми автомата в базисі FPGA за допомогою сучасних САПР та отримати числові характеристики схеми, зокрема значення апаратурних витрат. Для аналізу коректності роботи синтезованої схеми розглянуто процес розробки поведінкової складової VHDL-моделі, функцією якої є генерація вхідних сигналів автомата. Класичне поєднання синтезованої та поведінкової частин моделі дозволяє представити результати синтезу мікропрограмного автомата з операційним автоматом переходів як окремий проєкт, що може бути використаний в якості структурної складової проєктованої цифрової системи.

Результати. На прикладі абстрактної граф-схеми алгоритму керування розроблено VHDL-модель мікропрограмного автомата з операційним автоматом переходів. За допомогою САПР AMD Vivado проведено синтез розробленої моделі та проведене поведінкове моделювання роботи схеми автомата. Результати синтезу схеми дозволили отримати значення апаратурних витрат при реалізації схеми в базисі FPGA. За результатами поведінкового моделювання отримані діаграми часу, які свідчать про коректність реалізації функцій переходів та виходів синтезованого автомата.

Висновки. У традиційних VHDL-моделях кінцевих автоматів стани не містять конкретних кодів і ідентифікуються за допомогою літералів. Це дозволяє САПР проводити кодування станів на власний розсуд. Однак такий підхід не підходить для опису мікропрограмного автомата з операційним автоматом переходів. Перетворення кодів станів за допомогою множини арифметико-логічних операцій вимагає використання фіксованих значень кодів станів, що визначає специфіку VHDL-моделі, запропонованої в даній роботі. Дана і подібні моделі можуть бути використані, зокрема, при дослідженні ефективності мікропрграмного автомата за критерієм апаратурних витрат в схемі пристрою.

КЛЮЧОВІ СЛОВА: мікропрограмний автомат, операційний автомат переходів, VHDL-модель, апаратурні витрати, САПР AMD Vivado.

ЛІТЕРАТУРА

1. Bailliul J. Encyclopedia of Systems and Control / J. Bailliul, T. Samad. – Springer : London, UK, 2015. – 1554 p.
2. Sklyarov V. 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. Xilinx. XST UserGuide. V.11.3. Available online: https://www.xilinx.com/support/documentation/sw_manuals/xilinx11/xst.pdf (accessed on 12 April 2023).
10. 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. – Volume 3 (62). – P. 109–119.
11. 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.
12. Czerwinski R. Finite State Machine Logic Synthesis for Complex Programmable Logic Devices / R. Czerwinski, D. Kania. – Berlin : Springer, 2013. – 172 p.
13. Mano M. Digital design (4th Edition) / M. Mano. – New Jersey : Prentice Hall, 2006. – 624 p.
14. Zwolinski M. Digital System Design with VHDL / M. Zwolinski. – Boston : Addison-Wesley Longman Publishing Co., Inc, 2000. – 416 p.

© Barkalov A. A., Titarenko L. A., Babakov R. M., 2023
DOI 10.15588/1607-3274-2023-4-13