p-ISSN 1607-3274 Радіоелектроніка, інформатика, управління. 2023. № 3
e-ISSN 2313-688X Radio Electronics, Computer Science, Control. 2023. № 3

ПРОГРЕСИВНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

PROGRESSIVE INFORMATION TECHNOLOGIES

UDC 004.94 : 004.2

TEST GRAPH-SCHEMES OF THE ALGORITHMS OF FINITE STATE MACHINES WORK FOR ASSESSING THE EFFICIENCY OF AUTOMATED SYNTHESIS IN XILINX VIVADO CAD

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 evaluating the effectiveness of the automated design of a microprogram finite state machine with the operational transformation of state codes using Xilinx Vivado CAD is considered. The object of the research was graph-schemes of control algorithms implemented by finite state machine and able to prove the effectiveness of the principle of operational transformation of state codes in comparison with standard synthesis methods built into the CAD, in the context of hardware expenses optimization.

Objective. Development and research of graph-schemes of control algorithms in order to substantiate the effectiveness of the application of structure of the finite state machine with datapath of transitions in comparison with the built-in methods of synthesizing finite state machines in Xilinx Vivado CAD in the basis of programmable logic devices.

Method. The research is based on the hypothetical assumption that the Xilinx Vivado CAD has built-in methods of automated design of the circuit of a finite state machine, the effectiveness of which, according to the criterion of hardware expenses, exceeds other known methods of optimizing hardware expenses in the finite state machine circuit. In order to refute this hypothesis, it is proposed to prove that in some cases known methods of hardware expenses optimization in the finite state machine circuit are more effective in comparison with the methods built into CAD. In this work, as a well-known optimization method, the method of operational transformation of state codes, which corresponds to the structure of a finite state machine with datapath of transitions, is chosen. The effectiveness of this method is demonstrated on the example of several test graph-schemes of algorithms, the structure of which is abstract and artificially adapted to the chosen optimization method. The adaptation of the selected graph-schemes of the algorithms consists in the fact that a relatively small number of transition operations is required for their implementation with the help of a finite state machine with datapath of transitions. This contributes to the simplification of the circuit of the finite state machine and the reduction of hardware costs for its implementation. At the same time, the test graph-schemes of the algorithms have the possibility of scaling, which allows to automate the construction of VHDL models of the corresponding finite state machines for graph-schemes of different sizes and to evaluate the optimization of hardware expenses for finite state machines of different complexity.

Results. Using the example of several graph-schemes of algorithms, it is demonstrated that in some cases none of the finite state machine synthesis methods built into the Xilinx Vivado CAD is able to surpass the method of operational transformation of state codes according to the criterion of hardware expenses for the implementation of a finite state machine circuit. At the same time, a several-fold gain in hardware expenses can be achieved, which indicates the expediency of using this method under certain conditions. The formal definition of such conditions for the considered and other known optimization methods is a separate unsolved scientific problem.

Conclusions. The conducted experiments confirmed that in some cases, the known methods of synthesis of finite state machines allow to obtain circuits with lower hardware expenses than when using the methods of synthesis of finite state machines contained in Xilinx Vivado CAD. This testifies to the general expediency of using existing and developing new methods of hardware expenses optimization in the circuit of the finite state machines and the current relevance of the theory of the synthesis of digital automata as a scientific direction.

KEYWORDS: graph-scheme of algorithm, finite state machine, datapath of transitions, hardware expenses, Xilinx Vivado CAD.

ABBREVIATIONS

FSM is a finite state machine;
DT is a datapath of transitions;
GSA is a graph-scheme of algorithm;
CPLD is a complex programmable logic device;
LUT is a look-up table;
XST is a Xilinx Synthesis Tool.

DOI 10.15588/1607-3274-2023-3-12

NOMENCLATURE

M is a number of FSM states;

A is a set of FSM states $\{a_1, ..., a_M\}$;

L is a number of logic conditions;

X is a set of logic conditions $\{x_1, ..., x_L\}$;

N is a number of microoperations;

Y is a set of microoperations $\{y_1, ..., y_N\}$;

R is a digit capacity of state code;

T is a transition function of the FSM;

H^{DT} is a number of hardware expenses in the circuit of FSM with DT;

H^{XST} is a number of hardware expenses in the circuit of FSM, synthesized by XST;

E – the efficiency of the structure of the finite state machine according to the criterion of hardware expenses used for the implementation of its logic circuit.

INTRODUCTION

Modern human activity is closely related to the use of digital systems [1]. One of the main components of the digital system is the control unit [2, 3]. There are various ways of implementing control units, among which the finite state machine (FSM) model stands out [4, 5]. This model implements a given control algorithm in the form of a hardware circuit and is characterized by the maximum hardware expenses among other models of control units. At the same time, the model ensures the maximum speed of execution of the control algorithm due to the possibility of performing multidirectional microprogram transitions in one cycle of the device. The structure of the FSM can correspond to the Mealy machine model or the Moore machine model [2–5].

Large hardware expenses for the implementation of the FSM logic circuit have an impact on such characteristics of the finite state machine as power consumption, dimensions, cost, reliability, etc. [6]. Optimizing the characteristics of FSM circuit, in particular hardware expenses, is an important scientific and practical problem, the solution of which is devoted to many scientific works all over the world [1–7]. The structure of the finite state machine with datapath of transitions (FSM with DT), which is considered in this paper, is specifically aimed at minimizing hardware expenses by means of operational transformation of state codes [8].

In practice, the synthesis of FSM circuits is carried out with the help of specialized CAD, oriented to the use of the elemental basis of certain FPGA manufacturers. One of the leading manufacturers of FPGA-type chips is Xilinx, which is also the developer of Xilinx Vivado CAD. A component of this CAD is the Xilinx Synthesis Tool (XST), which implements, in particular, a number of methods for the synthesis of finite state machines [9]. At the same time, the question of how much these methods contribute to the optimization of hardware expenses in comparison with other known methods remains unexplored. This work solves the scientific and practical problem of comparing the efficiency of FSM synthesis methods built into the Xilinx Vivado CAD with the method of operational transformation of state codes. The solution to this problem is carried out by using graph-schemes of algorithms (GSA), adapted specifically for the structure of FSM with DT and able to demonstrate the advantages of this structure in comparison with the methods of synthesis of FSM in Xilinx Vivado.

The object of study is the process of synthesizing the logic circuit of a finite state machine in Xilinx Vivado CAD according to the VHDL model that corresponds to the given GSA.

This process 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 separate VHDL model must be developed, in the synthesis of which the capabilities of the XST tool are not used.

The subject of study is graph-schemes of control algorithms, which allow to prove the principle possibility of building a circuit with lower hardware expenes in the case of FSM with DT in comparison with circuits synthesized by Xilinx Vivado CAD in automatic mode.

The purpose of the work is the development and research of graph-schemes of control algorithms in order to substantiate the effectiveness of the application of the structure of FSM with DT in comparison with the built-in methods of synthesizing FSMs in Xilinx Vivado CAD in the basis of programmable logic devices.

1 PROBLEM STATEMENT

Let us assume that the finite state machine is given by the graph-scheme of the algorithm G and is characterized by the sets of states $A=\{a_1, ..., a_M\}$, input signals $X=\{x_1, ..., x_L\}$ and microoperations $Y=\{y_1, ..., y_N\}$. The synthesis 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 base using Xilinx Vivado CAD. As a result of the synthesis of FSM according GSA G using the built-in XST tool, the circuit of the FSM is numerically characterized by hardware expenses H^{CAD}. As a result of the synthesis of FSM with DT in Xilinx Vivado without the use of XST, the circuit is characterized by hardware expenses H^{DT}.

The work solves the problem of finding several examples of graph-schemes of algorithms for which

$$H^{CAD} > H^{DT}, \quad (1)$$

which will prove the expediency of using (under certain conditions) the method of operational transformation of state codes instead of FSM synthesis methods built into Xilinx Vivado CAD.

2 REVIEW OF THE LITERATURE

Today, a wide range of methods for optimizing hardware expenses in the circuit of a finite state machine is known. These include, for example, the so-called methods of structural decomposition [7]. The essence of the meth-

DOI 10.15588/1607-3274-2023-3-12

ods lies in the multiple transformation of logic signals, which leads to corresponding changes in the structural scheme of the FSM.

In this article, the method of operational transformation of state codes is considered as a method of optimizing hardware expenses [8]. According to it, the transformation of state codes into FSM is not created using a system of canonical Boolean equations, but using a set of arithmetic and logical operations. Circuits that implement these operations are combined into 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 particular, in [10].

The paper [11] substantiates the effectiveness of FSM with DT in comparison with the canonical structure of FSM based on the criterion of hardware expenses. However, today the canonical structure of FSM plays a more theoretical role, while the practical implementation of FSM circuits is carried out with the help of appropriate CAD, for example, Xilinx Vivado CAD. This is primarily due to the use of an elemental basis supported by CAD (as a rule, an FPGA basis).

Since FSM is often used as part of designed digital systems, support for its synthesis is implemented at the Xilinx Vivado CAD 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 FSM circuit 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.

During research conducted by the authors, the following hypothetical assumption was put forward. The use of FSM synthesis methods built into the Xilinx Vivado CAD will always allow you to obtain a machine circuit with lower hardware expenses than using other methods of optimizing the device circuit, which are not part of this CAD. This assumption is based on the fact that the Xilinx Vivado product has long been known in the world and contains developed technologies for the synthesis of specialized digital devices. In addition, CAD is focused on the use of its own elemental basis, which allows it to use the technological features of microchips to optimize circuits. The question of comparing the effectiveness of the synthesis of finite state machine by different methods in CAD Xilinx Vivado is not sufficiently considered today and does not allow to confirm or refute the hypothesis. This article is devoted to the solution of this issue on the example of a finite state machine with datapath of transitions.

3 MATERIALS AND METHODS

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

– coding of states 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 FSM in its VHDL description, the following provisions should be observed:

1. The FSM states are specified in the form of a set of letters combined in an element of the Enumeration Type.

2. The memory register must be synchronous and have the possibility of being resetted to the initial state by a reset signal.

3. The implementation of the transition and output functions system is implemented using the case operator.

These requirements make it possible to specify an finite state machine in the VHDL language using one, two or three processes [9, 12–14]. Regardless of how many processes describe the FSM, the XST tool is capable of extracting the finite state machine from the VHDL code and coding the states according to the chosen coding method. For this purpose, the "fsm_extraction" parameter is provided in the synthesis process setting section, which can take on the following values [9]:

1. “One-hot”. A separate trigger is used to encode each state. The number of triggers is equal to the number of FSM states. At each point in time, only one trigger can have a ones value. To form the value of each trigger, a logical equation is used, in which the number of terms is equal to the number of transitions to the corresponding state.

2. “Sequential”. The XST tool finds in the FSM long sequences of states consisting of unconditional transitions, and encodes the states within them with consecutive binary codes of minimum sufficient digit capacity. As a result, the input signals of the FSM are not fed to the address inputs of the LUT elements, but only the code of the current state is fed, which usually has a small digit capacity compared to the number of input signals. Sequential encoding of states ensures more optimal filling of the static memory cells of LUT elements and reduces the number of unused cells.

3. “Johnson”. Coding of states is performed using the Johnson code. Each value of this code contains only one continuous sequence of ones, and any two adjacent values in the ordered sequence of values differ by only one digit. The Johnson code is a cyclic code with redundancy and allows you to reduce the number of electrical disturbances caused by the simultaneous switching of several bits of the register circuit.

4. “Gray”. Coding of states is performed using the Gray code, in which two adjacent values in an ordered sequence of values differ by the value of one binary bit, and the number of bits matches the number of bits in the case of sequential coding. It is advisable to use the Gray code for encoding chains of states, since each microprogram transition in such chain will be accompanied by a

DOI 10.15588/1607-3274-2023-3-12

change of only one digit in the memory register of the FSM.

5. “Auto”. The XST tool chooses one of the coding methods described above at its discretion based on the results of the analysis of the VHDL model of the FSM. The choice of the coding method also depends on other XST settings (for example, on the leading optimization strategy – hardware expenses or speed), but the generalized approach is as follows: if the FSM contains a small number of states, “One-hot” coding is used; with an average number of states, the Johnson code is used; with a large number of states, the Gray code is used.

6. “None”. The method of encoding states is not regulated. If in the five modes considered above, the coding of states is noted in the protocol of the synthesis process, then in this case, information about the coding of states is not provided. It is usually assumed that state codes correspond to the sequential number of state identifiers when they are listed in the VHDL description, starting from zero, but the XST tool does not officially define this and reserves the right to code states at its own discretion. This feature does not interfere with the synthesis of the correct FSM circuit, but it does not allow the application of additional optimization methods based on the known values of state codes.

The considered values of the “-fsm_extraction” parameter (except for the “Auto” and “None” parameters) correspond to different methods of FSM circuit synthesis built into Xilinx Vivado CAD. We will conduct a study of the effectiveness of the automatic FSM synthesis according to built methods in comparison with the synthesis of FSM with DT according to the criterion of hardware expenses. Efficiency will be determined by the following expression:

$$E^{DT} = \frac{H^{CAD}}{H^{DT}}, \qquad (2)$$

where H^{CAD} is the minimum possible hardware expenses for the implementation of the FSM circuit when using built-in CAD methods; H^{DT} – hardware expenes for the implementation of the FSM circuit with datapath of transitions. The unit of measurement of these parameters will be the number of used LUT elements of the selected FPGA chip. The value $E^{DT} > 1$ will mean that the circuit of the FSM with DT has lower hardware expenes compared to the FSM circuit synthesized by Xilinx Vivado built-in methods. Achieving the values $E^{DT} > 1$ will prove that, under certain conditions, the use of third-party methods for optimizing hardware expenses is more appropriate than the methods built into CAD.

In the context of the considered problem, the authors investigated five test GSAs G_1–G_5, which have the following common properties:

1. GSA reproduces only the transition function of the FSM and does not contain information about the output function (all operator nodes are empty). This makes it possible to determine the hardware expenses for the implementation of the transition function, although it cannot be considered an indicator of the efficiency of the FSM as a whole.

2. GSA has a regular structure, that is, it is a sequence of identical fragments. This approach, firstly, simplifies the scaling of the GSA by increasing the number of fragments, and secondly, allows you to automate the process of generating the VHDL code to describe the FSM.

Let’s consider GSAs investigated in this work.

GSA *G*$_1$

The GSA does not contain conditional nodes, has a completely linear structure and is marked by M states of the Moore machine (Fig. 1).

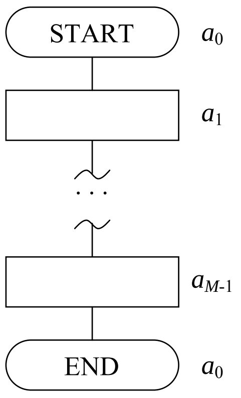

Figure 1 – GSA G_1

To implement sequential transitions in the corresponding FSM with DT, natural coding of states can be used, in which the binary code of the state coincides with its index in Fig. 1, and the state codes are transformed using a counter [3, 7].

GSA *G*$_2$

GSA corresponds to a Moore machine and has the following structure (Fig. 2):

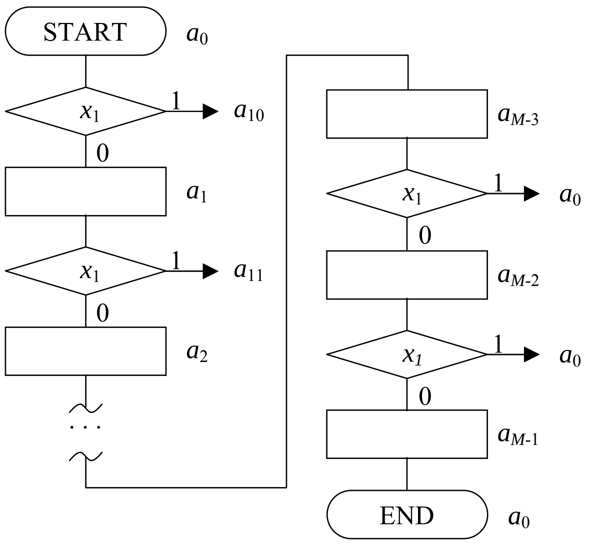

Figure 2 – GSA G_2

DOI 10.15588/1607-3274-2023-3-12

– each operator node is followed by a conditional node;

– the same logical condition x_1 is checked in all conditional nodes;

– when $x_1 = 0$ the transition leads to the next FSM state, when $x_1 = 1$ the transition leads ten states forward;

– for the last ten states, under the condition $x_1 = 1$ the transition leads to the state a_0.

In this case, the natural sequence of state coding is obvious, which makes it possible to implement the operational transformation of state codes based on the counter.

GSA G_3

The structure of the GSA is similar to the G_2 GSA, but the number of the state to which the transition is made under the condition $x_1 = 1$, is generated in a pseudo-random way within the limits of $[0; M-1]$. The general appearance of the GSA G_3, marked by the states of the Moore machine, is shown in Fig. 3.

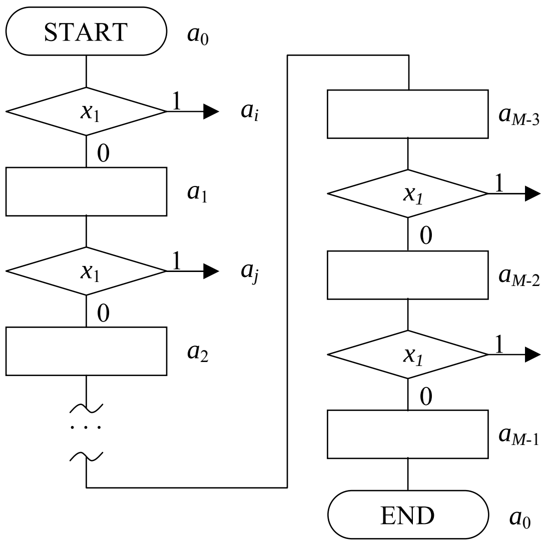

Figure 3 – ГСА G_3

In Fig. 3 states a_i, a_j, a_k, a_l mean some different states within the GSA. As in GSA G_2, states are coded in a natural order. Implementation of pseudo-random transitions is carried out with the help of a shift register with feedback based on the XOR operation. Such a register allows you to generate a sequence of $(2^R - 1)$ unique bit vectors that can be used to encode the states of the FSM. For this, the feedback must be organized according to a special primitive polynomial of length R, where R is the bit capacity of the state code of the FSM.

GSA G_4

GSA is similar in structure to G_2, but in each conditional node a different logical condition is checked, from x_1 to x_{M-1}. The general appearance of the GSA G_4, marked by the states of the Moore machine, is shown in Fig. 4.

GSA G_5

GSA is similar in structure to G_3, but in each conditional node a different logical condition is checked, from x_1 to x_{M-1}. The general appearance of the GSA G_5, marked by the states of the Moore machine, is shown in Fig. 5.

For each of the considered GSAs two VHDL models were built. The first model describes an FSM in the form of two processes and is intended for automated synthesis by Vivado XST. The second model represents the RTL description of the FSM with DT and is designed for circuit synthesis in Xilinx Vivado without using the “FSM Extraction” option.

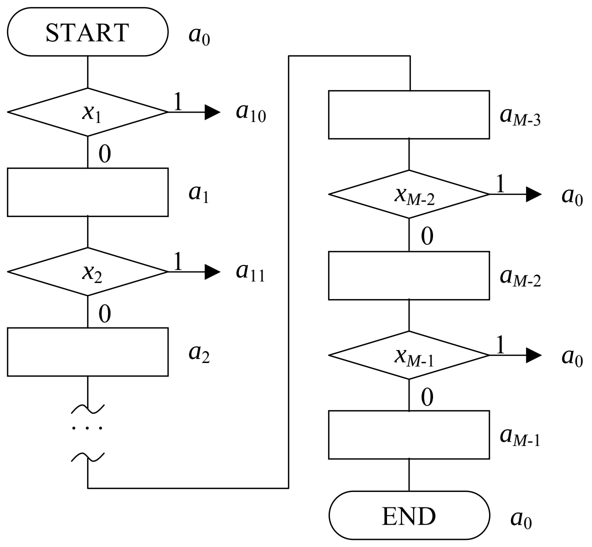

Figure 4 – GSA G_4

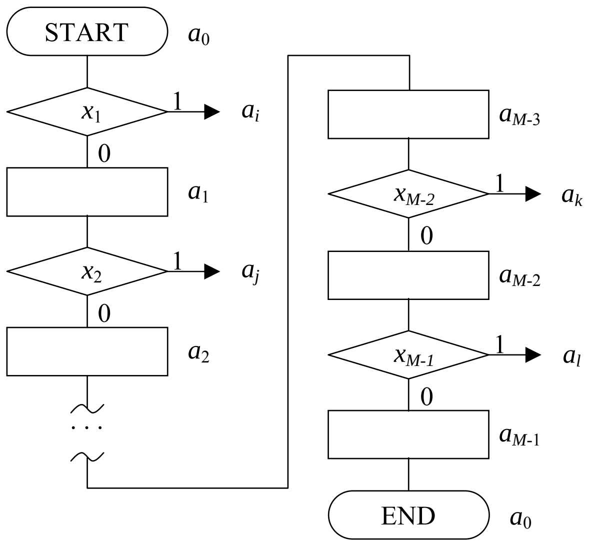

Figure 5 – GSA G_5

As an example, consider VHDL-models corresponding to GSA G_2. In Fig. 6 shows an example of the synthesized part of the FSM VHDL model intended for synthesis using XST. This model describes a machine with only

DOI 10.15588/1607-3274-2023-3-12

five states, but can easily be scaled to a machine with an arbitrary number of states. The model corresponds to the structure of the canonical FSM [2, 3, 9], but does not contain a part corresponding to the output function of the FSM. This is due to the fact that in the FSM with DT, the optimization of hardware expenses is carried out only in the transitions formation circuit, while the circuit of forming microoperations remains the same as in the canonical FSM. In order to emphasize the saving of hardware expenses precisely in the transition formation circuit, the used VHDL-models (as well as GSA) do not contain parts that correspond to the FSM output function).

```
package types is
    type state_type is (a0, a1, a2, a3, a4);
end package types;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
use work.types.all;

entity FSM is                      -- Canonical FSM
    port (x1: in std_logic;
          C: in std_logic;         -- Clock
          Reset: in std_logic;     -- Reset
          O: out state_type);      -- Current state
end FSM;

architecture FSM_A of FSM is
signal state, next_state: state_type;
begin

        process(C)           -- Memory Register
        begin
             if rising_edge(C) then
               if Reset = '1' then
                    state <= a0;
                else
                   state <= next_state;
               end if;
             end if;
        end process;

       process (state, x1) -- Trans. circuit
       begin
         case state is
           when a0 => if x1 = '0' then
                         next_state <= a1;
                      else
                         next_state <= a2;
                      end if;
           when a1 => if x1 = '0' then
                         next_state <= a2;
                      else
                         next_state <= a3;
                      end if;
           when a2 => if x1 = '0' then
                         next_state <= a3;
                      else
                         next_state <= a4;
                      end if;
           when a3 => if x1 = '0' then
                         next_state <= a4;
                      else
                         next_state <= a0;
                      end if;
           when a4 => next_state <= a0;
         end case;
       end process;

       O <= state;
end FSM_A;
```

Figure 6 – Synthesizable part of the VHDL model of canonical FSM for GSA G_2

We emphasize that in the model in Fig. 6 FSM states are declared by enumerating the literals “a0”, “a1”, etc. Specific state codes are not defined in this model. It is this approach that makes it possible to use the automatic extraction of the finite state machine by XST tool.

In Fig. 7 shows the VHDL model of the FSM with DT corresponding to GSA G_2 for the case of M=100 states.

На рис. 7 наведена VHDL-модель МПА з ОАП, що відповідає ГСА G_2 для випадку M=100 станів.

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

entity FSM is
    -- Bit capacity of state code:
    generic (R: integer := 7);

    port (C: in std_logic;
          Reset: in std_logic;
          D: out unsigned (R-1 downto 0));
end entity FSM;

architecture OAP_A of OAP is
signal state,
next_state: unsigned (R-1 downto 0);
begin

    process (C)   -- Memory Register
    begin
        if rising_edge(C) then
           if Reset = '1' then
               state <= "0000000";
           else
               state <= next_state;
           end if;
        end if;
    end process;

    process (state, x1)--Datapath of transitions
    begin
      -- If state is in range from a0 to A(M-10)
      if state < "1011010" then
           if x1 = '0' then -- If x1=0
              next_state <= state + 1;  --  +1
           else             -- If x1=1
              -- Transition to next state:
              next_state <= state + 10;
           end if;
       -- If it is state a(M-1)
       elsif state = "1100011" then
           next_state <= "0000000";-- To state a0

       -- If it is one of other ten states
       else
           if x1 = '0' then -- If x1=0
              -- Transition to next state:
              next_state <= state + 1;
           else  -- If x1=1
              -- Transition to initial state a0
              next_state <= "0000000";
           end if;
       end if;
    end process;

    D <= state;
end architecture OAP_A;
```

Figure 7 – Synthesizable part of the VHDL model of FSM with DT for GSA G_2

It should be noted that for the model in Fig. 6, the VHDL code length will increase along with the increase in the number of states of the machine. This is primarily due to an increase in the number of branches of the "case" operator. Instead, in the case of the equivalent FSM with DT (Fig. 7), a change in the number of states will only lead to a change in the bit rate and values of the constants in the model code. However, at the same time, this VHDL model maintains a "hard" binding to the GSA G_2 structure, which corresponds to the concept of FSM as a machine with "hardware" logic.

4 EXPERIMENTS

On the basis of the developed test GSAs, studies of the effectiveness of FSM with DT in comparison with the methods of synthesis of finite state machines built into the Xilinx Vivado CAD were carried out. The criterion of efficiency is the value of hardware expenses in the circuit of the FSM when it is implemented in the FPGA basis. The essence of the experiments was as follows:

1. VHDL models of canonical FSM and FSM with DT containing 100, 200, 500, 1000, 2000 states were built for each of GSA G_1–G_5.

2. For each model of the canonical FSM, the synthesis of the circuit was performed in the mode of automatic extraction of the finite state machine. The synthesis is performed for each of the following values of the "fsm-extraction" parameter: "one-hot", "sequential", "johnson"", "gray", "auto". According to the results of each synthesis, the numerical value of hardware expenses H^{CAD} was obtained, expressed in the number of used LUT-elements of the FPGA. The synthesis was carried out for the microchip xc7s6cpga196-2 of the Spartan-7 series. All Xilinx Vivado settings except the parameter "fsm_extraction" are selected by default.

3. For each GSA G_1–G_5, such a value of the "fsm-extraction" parameter is defined, at which the FSM circuit has the lowest hardware expenses in comparison with other values of this parameter. The values of hardware expenses obtained in this case are considered to be the minimum possible, which can be obtained when using FSM synthesis methods built into CAD.

4. For each model of FSM with DT, synthesis of the circuit was performed without using the mode of automatic extraction of the finite state machine. According to the results of each synthesis, the numerical value of the hardware expenses H^{DT} was obtained, expressed in the number of used LUT-elements of the FPGA. The synthesis conditions are the same as in the synthesis of models of canonical FSM.

5. For each model, the efficiency was calculated according to expression (2). As values H^{CAD}, the minimum values of hardware expenses are taken in accordance with clause 3.

It should be emphasized that for all models, the operability of the logic circuit of the FSM in Xilinx Vivado was verified using an additional VHDL model of the behavioral type. The function of this model was the generation of input signals, reset and synchronization signals.

5 RESULTS

The results of experimental studies are given in tables –5. The row of the table containing the minimum values according to clause 3 is marked with a gray background. The symbol "–" in the cells of the tables means that the numerical values of the hardware expenses for the corresponding parameters were not obtained due to the extremely high complexity of the resulting circuit.

Table 1 – Results of GSA G_1 studies

M		100	200	500	1000	2000
H^{CAD}, LUT	one-hot	60	116	292	570	1181
	sequential	9	15	18	23	27
	johnson	273	457	979	–	–
	gray	18	20	29	38	40
	auto	60	116	292	570	1181
H^{DT}, LUT		6	8	11	12	13
E		**1.5**	**1.88**	**1.63**	**1.92**	**2.08**

Table 2 – Results of GSA G_2 studies

M		100	200	500	1000	2000
H^{CAD}, LUT	one-hot	109	215	539	1489	3091
	sequential	18	18	23	27	22
	johnson	267	759	3047	–	–
	gray	26	38	56	68	66
	auto	109	215	539	1489	3091
H^{DT}, LUT		13	15	19	20	21
E		**1.38**	**1.20**	**1.21**	**1.35**	**1.05**

Table 3 – Results of GSA G_3 studies

M		100	200	500	1000	2000
H^{CAD}, LUT	one-hot	112	250	668	1454	3334
	sequential	44	94	239	511	1177
	johnson	419	1642	–	–	–
	gray	43	92	239	510	1166
	auto	112	250	668	1454	3334
H^{DT}, LUT		21	41	99	190	407
E		**2.05**	**2.24**	**2.41**	**2.68**	**2.86**

Table 4 – Results of GSA G_4 studies

M		100	200	500	1000	2000
H^{CAD}, LUT	one-hot	155	311	799	1958	4051
	sequential	130	269	656	1368	2767
	johnson	412	1219	3978	–	–
	gray	127	254	631	1240	2498
	auto	155	311	799	1958	4051
H^{DT}, LUT		51	72	155	318	586
E		**2.49**	**3.54**	**4.07**	**3.90**	**4.26**

Table 5 – Results of GSA G_5 studies

M		100	200	500	1000	2000
H^{CAD}, LUT	one-hot	145	298	831	1782	4050
	sequential	166	357	1048	2286	5123
	johnson	589	1967	–	–	–
	gray	155	348	995	2281	4922
	auto	145	298	831	1782	4050
H^{DT}, LUT		58	117	225	454	931
E		**2.50**	**2.54**	**3.69**	**3.92**	**4.35**

The content of the last row of each table is calculated as the result of dividing the values in the row of the table, marked with a gray background, by the value in the row H^{DT} of the table. For example, in the table 1, the value of 1.5 in the last row of the table is calculated as a fraction of the division of 9 by 6, where 9 is the minimum possible value of hardware expenses obtained using the XST tool, 6 is the value of hardware expenses for the implementation of circuit of the equivalent FSM with DT.

6 DISCUSSION

A finite state machine with datapath of transitions differs from a canonical finite state machine in that it uses a set of arithmetic and logic operations to transform state codes, which form a datapath of transitions. The complexity of the DT circuit depends on the number of operations used by datapath to transform state codes. If, for a given GSA, the complexity of the DT turns out to be less than the complexity of the transitions formation circuit of the canonical FSM, the use of the structure of FSM with DT is preferable compared to the canonical FSM.

The synthesis results shown in tables 1–5 demonstrate a clear gain (sometimes several times) in hardware expenses when implementing the FSM transition function using the method of operational transformation of state codes and the structure of FSM with DT. This allows us to draw the following conclusions.

1. Finite state machine synthesis methods built into Xilinx Vivado CAD do not always give the best result in terms of hardware expenses in comparison with third-party FSM optimization methods.

2. The use of finite state machine synthesis methods built into CAD does not allow choosing specific values of state codes. This makes it impossible to simultaneously use other known methods of optimization of the FSM circuit, in particular, optimization of the output function circuit. Instead, the use of FSM with DT is based on pre-selected values of state codes, which potentially allows combining other optimization methods with the method of operational transformation of state codes.

3. The theory of synthesis and optimization of circuits of finite state machine remains relevant today, provided that modern CAD digital systems and elemental basis are used. A promising scientific and practical direction is the formalization and algorithmization of well-known methods of FSM circuit optimization.

Certain limitations of the conducted research should also be taken into account when analyzing the obtained results.

1. The comparative analysis of hardware expenses was carried out only for the circuit of transition function without taking into account expenses in the circuit that implements the output function. If, for a given FSM, the hardware expenses for the implementation of the output function significantly exceed expenses for the implementation of the transition function, the effect of using an FSM with DT instead of a canonical FSM will be much smaller. However, from the point of view of Xilinx Vivado CAD, the use of FSM with DT is not the only third-party approach to reducing hardware expenses. Therefore, the potential possibility of surpassing the methods built into CAD remains for other well-known synthesis and optimization methods.

2. The structure of studied GSAs is artificially adapted in such a way that the implementation of microprogram transitions in the corresponding FSM with DT takes place with the help of a smaller number of operations. This made it possible to obtain efficiency values greater than 1. This fact indicates the need to optimize the FSM with DT circuit, which is possible under the condition of development and application of special methods of synthesis of this FSM class. At that time, the use of methods built into CAD allows for synthesis in automatic mode, following only the rules of building VHDL models to ensure automatic extraction of the finite state machine by means of XST.

However, these limitations do not negate (and in some ways emphasize) the general possibility and expediency of using known optimization methods, as opposed to the methods built into CAD.

CONCLUSIONS

The article proposes a solution to the scientific problem of researching the effectiveness of the method of operational transformation of state codes in comparison with the methods of synthesis of finite state machines built into Xilinx Vivado CAD, according to the criterion of hardware expenses.

The scientific novelty of the work consists in the experimental confirmation of the advantage (under certain conditions) of the use of well-known methods of synthesis of finite state machines (in particular, the method of operational transformation of state codes) compared to the methods of synthesis of finite state machines built into Xilinx Vivado CAD. This confirmation is provided by the efficiency values obtained during research (Tables 1–5). Thus, the hypothesis that finite state machine synthesis methods built into Xilinx Vivado are always able to generate FSM circuits with lower hardware expenses compared to other hardware expenses optimization methods is disproved.

The practical use of the obtained results is possible in the development of methods for evaluating the effectiveness of the structure of a finite state machine with datapath of transitions, as well as other structures and methods aimed at optimizing the characteristics of the circuit of a finite state machine.

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 structures and methods of synthesis of finite state machines with optimized hardware expenses.

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. 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 19.06.2023.
Accepted 11.08.2023.

УДК 004.94 : 004.2

ТЕСТОВІ ГРАФ-СХЕМИ АЛГОРИТМІВ РОБОТИ МІКРОПРОГРАМНИХ АВТОМАТІВ ДЛЯ ОЦІНКИ ЕФЕКТИВНОСТІ АВТОМАТИЗОВАНОГО СИНТЕЗУ В САПР XILINX VIVADO

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 оцінки ефективності автоматизованого проєктування мікропрограмного автомата з операційним перетворенням кодів станів із використанням САПР Xilinx Vivado. Об'єктом дослідження були граф-схеми алгоритмів керування, що імплементуються мікропрограмним автоматом та здатні довести ефективність принципу операційного перетворення кодів станів у порівнянні зі стандартними методами синтезу, вбудованими в САПР, в контексті оптимізації апаратурних витрат.

Мета. Розробка і дослідження граф-схем алгоритмів керування з метою обгрунтування ефективності застосування структури мікропрограмного автомата з операційним автоматом переходів у порівнянні із вбудованими методами синтезу автоматів в САПР Xilinx Vivado в базисі програмувальних логічних пристроїв.

Метод. В основу дослідження покладено гіпотетичне припущення про те, що САПР Xilinx Vivado має вбудовані методи автоматизованого проектування схеми мікропрограмного автомата, ефективність яких за критерієм апаратурних витрат перевершує інші відомих методи оптимізації апаратурних витрат в схемі автомата. З метою спростування даної гіпотези запропоновано довести, що в окремих випадках відомі методи оптимізації апаратурних витрат в схемі автомата є більш ефективними у порівняні з методами, вбудованими в САПР. В даній роботі в якості відомого методу оптимізації обраний метод операційного перетворення кодів станів, що породжує структуру мікропрограмного автомата з операційним автоматом переходів. Ефективність цього методу доводиться на прикладі кількох тестових граф-схем алгоритмів, структура яких є абстрактною і штучно адаптована до обраного методу оптимізації. Адаптація обраних граф-схем алгоритмів полягає в тому, що для їх реалізації за допомогою мікропрограмного автомата з операційним автоматом переходів потрібна відносно мала кількість операцій переходів. Це сприяє спрощенню схеми автомата і зменшенню апаратурних витрат на її реалізацію. Разом з тим тестові граф-схеми алгоритмів мають можливість масштабування, що дозволяє автоматизувати побудову VHDL-моделей відповідного автомата для граф-схем різного розміру і оцінити оптимізацію апаратурних витрат для автоматів різної складності.

DOI 10.15588/1607-3274-2023-3-12

Результати. На прикладі декількох граф-схем алгоритмів продемонстровано, що в окремих випадках жоден із методів синтезу кінцевих автоматів, вбудованих в САПР Xilinx Vivado, не здатен перевершити метод операційного перетворення кодів станів за критерієм апаратурних витрат на реалізацію схеми мікропрограмного автомата. При цьому може досягатись кількаразовий виграш у витратах апаратури, що свідчить про доцільність використання даного методу за певних умов. Формальне визначення таких умов для розглянутого та інших відомих методів оптимізації є окремою невирішеною науковою проблемою.

Висновки. Проведені експерименти підтвердили, що в окремих випадках відомі методи синтезу мікропрограмних автоматів дозволяють отримати схеми автоматів із меншими витратами апаратури, ніж при використанні методів синтезу автоматів, вбудованих в САПР Xilinx Vivado. Це свідчить про загальну доцільність використання існуючих і розробки нових методів оптимізації апаратурних витрат в схемі автомата та про сьогоденну актуальність теорії синтезу цифрових автоматів як наукового напрямку.

КЛЮЧОВІ СЛОВА: граф-схема алгоритму, мікропрограмний автомат, операційний автомат переходів, апаратурні витрати, САПР Xilinx 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.

DOI 10.15588/1607-3274-2023-3-12