The 11th IEEE International Conference on Dependable Systems, Services and Technologies, DESSERT'2020
14-18 May, 2020, Kyiv, Ukraine

Verification of FPGA control systems by analyzing the correctness of state diagrams

Maryna Miroshnyk
professor, USURT
Kharkiv, Ukraine
marinagmiro@gmail.com

Alexander Shkil
associate professor, KhNURE
Kharkiv, Ukraine
oleksandr.shkil@nure.ua

Elvira Kulak
associate professor, KhNURE
Kharkiv, Ukraine
elvira.kulak@nure.ua

Dariia Rakhlis
associate professor, KhNURE
Kharkiv, Ukraine
dariia.rakhlis@nure.ua

Inna Filippenko
associate professor, KhNURE
Kharkiv, Ukraine
inna.filippenko@nure.ua

Anatolii Miroshnyk
graduate student, KhNURE
Kharkiv, Ukraine
anatolii.miroshnyk@nure.ua

***Abstract* – The work is dedicated to verification of automatic logic control systems by analyzing the correctness of state diagrams of control finite state machines which are represented in the form of the code in the hardware description language. As a method for state diagram analysis the, it is proposed to use the concept of orthogonality, as a system of incompatible events. Analysis of the correctness is carried out by analysis the results of behavioral modeling and logical synthesis using CAD tools.**

Keywords – finite state machine, state diagram, HDL-model, synthesis, orthogonal Boolean function.

I. Introduction

Main problems in the development of modern microelectronics market are the cost and design time reduction, which is achieved by improving the technical, informational and software provision for computer-aided design systems of the radioelectronic equipment (CAD REE). The most difficult and costly stage in the modern design cycle of digital devices (DD) is functional verification, i.e. the process of identification, localization and elimination errors in models of different levels of specification's abstraction, which takes up to 70% of the total design time.

The design process transforms set of specifications into implementation. At the description level, the specification defines the functionality that should be implemented in the system, but does not specify implementation which means. It can be argued that both the specification and the implementation are forms of functionality description. A specification is defined at higher level of abstraction than implementation. During the design process, the description is phased transformed from the highest level of abstraction to the lowest. Functional verification is a verification of the correspondence between refined description of functionality at the lowest level of abstraction and previously created description at the highest level of abstraction [1].

The whole set of permissible functional errors can be divided into two main types: errors that are specific for the implementation, but not specific for specification – are entered during implementation; and errors in the specification itself, that are manifested in the absence of behaviour description in some special situations, a conflict of requirements, undocumented behaviour. Errors of first type can be detected by different automated methods of the functional verification. Errors of the second type can be detected only manually during review of requirements and architecture of the designed system [2]. This work is devoted to identifying precisely these types of errors in the specifications.

Hardware Description Languages (HDL) are characterized by dualism. From one side, this is a code in a formal language with all its characteristics and properties, and from another side, this is a description for digital circuit with all restrictions imposed by the corresponding technological base. In addition, a computer-aided design tool (CAD) has a component called synthesizer which is located between the code in the hardware description language (HDL model) and digital circuit. A subset of HDL, which is correctly converted by a synthesizer into digital circuits, is called, synthesized HDL subset. In addition to the fact that HDL operators, which correctly describe the circuit, must be included in the synthesized subset, the structure of HDL model must fits to certain rules of optimal synthesis.

During description of the functioning algorithm of logic control DD in CAD tool, one of HDL-code's construction style is automata-based programming style, and for correct representation of HDL-models of control finite state machines (FSMs) from the point of circuit synthesis it's customary to use so-called automata-based pattern. Features of these patterns' construction affect synthesis results of automatic logic control devices in CAD using FPGAs. In the case of HDL-descriptions of control FSM's models, it is decided to use so-called two-processor FSM's pattern, in which the transition and output functions are calculated in one process, and the assignment of new state is performed in another process associated with synchronization [3]. Implementation methods of finite state machines models in the hardware description languages VHDL and Verilog, as well as the synthesis results of these models are described in detail in [4].

During verification of logical control system (LCS) it is advisable to apply the functional approach – check not the software HDL-code and the circuit, which was synthesized on its basis, but the functional model of the state machine – the state diagram. It should be noted that from the point of view of synthesis the FSM's pattern uniquely and correctly reflects the functional model of the finite state machine in the form of the state diagram.

During analysis of the control systems reliability, the concept of orthogonality is widely used as an incompatible events system. The concept of orthogonalization is also used for decomposition of logical functions during synthesis of

978-1-7281-9957-3/20/$31.00 ©2020 IEEE

digital systems and during verification of graph models of digital FSM for correctness. Rules for verification the state diagram for correctness are developed in sufficient detail and practically standardized. This is check for completeness, consistency, feasibility and the presence of generating circuits [5]. Proceeding from this, an actual task is to develop verification procedures for HDL models of finite state machines taking into account of the correctness under condition of state diagram from one side, and, rules of synthesizers' operation on technology platform of the CAD FPGA from another hand.

II. CORRECTNESS VERIFICATION OF HDL-MODEL, WHICH IS REPRESENTED AS STATE DIAGRAM

The syntactical correctness of the state diagram is determined by the fulfillment of the conditions for transition functions: consistency (orthogonality) and completeness. Consistency in the state diagram is provided in the case, if transitions are simultaneously forbidden to any of two or more arcs, which come out from one vertex. The completeness of the state diagram (marks' disjunction of all arcs, which is outgoing from a vertex, is equal to one) is checked after ensuring consistency.

A fragment of the state diagram for vertex a_i with K outgoing arcs is shown in fig. 1.

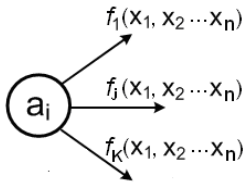

Fig. 1. Mapping of transition function conditions on the graph model

Each arc is associated with logical expression of the transition function conditions $f(x_1,x_2,\dots x_n)$ in the disjunctive normal form (DNF) also known as sum of products (SOP) form:

$$f(x_1,x_2,\dots x_n) = f_1(x_1,x_2,\dots x_n) \vee \dots \vee f_j(x_1,x_2,\dots x_n) \vee \dots \vee f_K(x_1,x_2,\dots x_n). \quad (1)$$

Completeness is checked for each vertex of state diagram by analyzing of transitions' conditions of all arcs, which is outgoing from this vertex, i.e. $\bigvee_{j=1}^{K} f_j(x_1,x_2,\dots x_n) = 1$. The completeness of conditions is defined as cover of all 2^n products (terms) of Boolean transitions functions set, where n – quantity of transitions' conditions (input variables which initiate transitions from this vertex), i.e. $f(x_1,x_2,\dots x_n) = 1$.

While ensuring the consistency, for each vertex of the state diagram, the orthogonalization of Boolean expressions of transitions' conditions is checked (the absence of common terms in Boolean expressions of conditions for different arcs) for arcs, which is outgoing from the considered vertex, i.e. $\forall(f_g \cdot f_h = 0)$, $g \neq h$ [6].

A normal disjunctive function of the algebra of logic is called orthogonal if all the conjunctive terms are mutually orthogonal. For such function, there is no set of variables' values, which belongs more than to one elementary conjunction, that is, on any set of variables' values, the value of one can be accepted only by one conjunction. If the logical function is presented in the form of a Karnaugh map, the images of conjunctions (products) will not intersect. An example of orthogonal DNF can be its canonical form (CDNF or CSOP), which consists of complete mutually orthogonal conjunctions [7].

A method for constructing an orthogonal complete system of transitions functions for the vertex of the state diagram is considering below.

The following definitions are used: *f* – complete CDNF from *n* variables, that is $f(x_1,x_2,\dots x_n) = 1$ – Boolean function that takes the value 1 on all 2^n sets, f^* – complete CDNF from (*n-1*) variables, f^{**} – complete CDNF from (*n-2*) variables, f^{***} – complete CDNF from (*n-3*) variables and so on. Thus, $f = f^* = f^{**} = f^{***} \equiv 1$.

The orthogonality of Boolean function' terms is ensured by decomposing into the corresponding variables, taking into account the completeness of decomposition into all variables [8].

According to the first theorem, the decomposition will be a follows:

$$\begin{aligned} f &= \overline{x_1} \cdot f^* \vee x_1 \cdot f^* = \overline{x_1} \cdot 1 \vee x_1 \cdot f^* = \overline{x_1} \vee x_1 \cdot f^* = \overline{x_1} \vee \\ &\vee x_1 (\overline{x_2} \vee x_2 \cdot f^{**}) = \overline{x_1} \vee x_1 (\overline{x_2} \vee x_2 (\overline{x_3} \vee x \cdot f^{***})) = \\ &= \overline{x_1} \vee x_1 (\overline{x_2} \vee x_2 (\overline{x_3} \vee x_3 \cdot (\dots \cdot (\overline{x_n} \vee x_n)))). \end{aligned} \quad (2)$$

Thus, the complete Boolean function of *n* variables decomposes at least to (*n + 1*) conjunctions while preserving the essence of all *n* variables.

As an example, the complete CDNF $f(x_1,x_2,x_3) = 1$ is c considering. By definition, a CDNF is orthogonal. We write the complete CDNF and perform the decomposition by x_1 with the replacement of the left side of the decomposition inside brackets with 1.

$$\begin{aligned} f(x_1,x_2,x_3) &= \overline{x_1}\,\overline{x_2}\,\overline{x_3} \vee \overline{x_1}\,\overline{x_2}x_3 \vee \overline{x_1}x_2\overline{x_3} \vee \\ &\vee \overline{x_1}x_2x_3 \vee x_1\overline{x_2}\,\overline{x_3} \vee x_1\overline{x_2}x_3 \vee x_1x_2\overline{x_3} \vee x_1x_2x_3 = \\ &= \overline{x_1}(\overline{x_2}\,\overline{x_3} \vee \overline{x_2}x_3 \vee x_2\overline{x_3} \vee x_2x_3) \vee x_1(\overline{x_2}\,\overline{x_3} \vee \\ &\vee \overline{x_2}x_3 \vee x_2\overline{x_3} \vee x_2x_3) = \overline{x_1} \cdot 1 \vee x_1 \cdot (\overline{x_2}\,\overline{x_3} \vee \overline{x_2}x_3 \vee \\ &\vee x_2\overline{x_3} \vee x_2x_3) = \overline{x_1} \vee x_1(\overline{x_2}\,\overline{x_3} \vee \overline{x_2}x_3 \vee x_2\overline{x_3} \vee x_2x_3). \end{aligned} \quad (3)$$

Similar procedure of decomposition by x_2 for the expression in brackets is performed. Open brackets and get the complete orthogonal function of three variables (1).

Based on this, we can conclude that the complete orthogonal function of *n* variables has at least (*n + 1*) conjunctions. So, from each state of the finite state machine with the function of transitions' conditions from *n* variables $f(x_1,x_2,\dots x_n)$ there must be at least (*n + 1*) transitions.

$$f(x_1,x_2,x_3) = \overline{x_1} \vee x_1(\overline{x_2}(x_3 \vee \overline{x_3}) \vee x_2(\overline{x_3} \vee x_3)) = \\ = \overline{x_1} \vee x_1(\overline{x_2} \cdot 1 \vee x_2(\overline{x_3} \vee x_3)) = \\ = \overline{x_1} \vee x_1(\overline{x_2} \vee x_2\overline{x_3} \vee x_2x_3) = \\ = \overline{x_1} \vee x_1\overline{x_2} \vee x_1x_2\overline{x_3} \vee x_1x_2x_3 \,. \quad (4)$$

One of the ways of visual analysis of the transitions' functions orthogonality is representation of the orthogonal functions using Karnaugh maps. Karnaugh map for the orthogonal function (4) is shown in fig. 2 (a). From this map, it can be seen that for the orthogonal function, groups of ones for the complete CDNF don't intersect, i.e. conjunctions have no common parts.

Karnaugh map for the orthogonal function $f(x_1,x_2,x_3) = \overline{x_1} \vee x_1\overline{x_2} \vee x_1x_2\overline{x_3}$ is shown in fig. 2 (b), but this function is not complete, since there is no group which is corresponds to conjunction $x_1x_2x_3$. This is due to the fact that the construction of this function violated the rule of completeness of the function, i.e. $f^* \neq 1$.

Karnaugh map for the orthogonal function $f(x_1,x_2,x_3) = \overline{x_1} \vee x_1\overline{x_2} \vee x_1x_2$ is shown in fig. 2 (c), but this function is also not complete, since there is no variable x_3. This is due to the fact that construction of this function violated the rule that in complete function supposed to be no less than ($n + 1$) conjunctions.

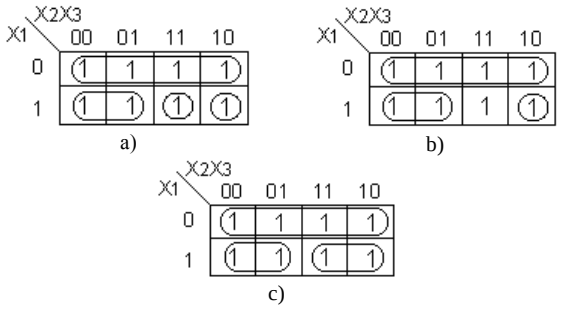

Fig. 2. Karnaugh maps for orthogonal functions

III. Comparison of Synthesis Results of HDL-codes of Correct and Incorrect State Diagramms

An example of the state diagram (fig. 3) with correct conditions for transitions from the a_1 state is considering.

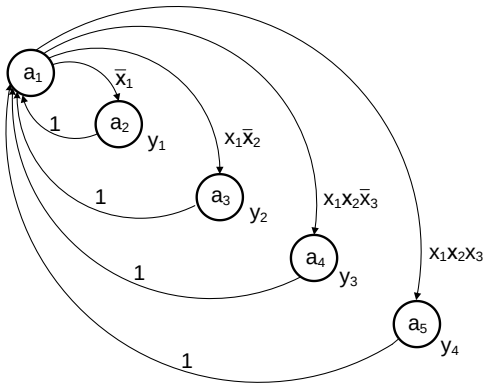

Fig. 3. The state diagram of Moore FSM with the correct conditions of transitions

The transitions' conditions function from the a_1 state is orthogonal, therefore, conditions of transitions are consistent. In addition, the function is complete. VHDL-model of this FSM is shown in fig. 4. Timing diagram of this FSM are shown in fig. 5.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity Fsm_right is
        port (x1, x2, x3, Clk, reset: in STD_LOGIC;
        y1, y2, y3, y4: out STD_LOGIC);
end;
architecture Fsm_right of Fsm_right is
        type State_type is (a1, a2, a3, a4, a5);
        signal State, NextState: State_type;
begin
        Sreg0_CurrentState: process (Clk, reset)
        begin
                if reset='1' then State <= a1;
                elsif Clk'event and Clk = '1'
                then State <= NextState;
                end if;
        end process;
        Sreg0_NextState: process (State, x1, x2, x3)
        begin
        case State is
        when  a1=> if x1='0' then NextState <= a2;
                elsif x2='0' then NextState <= a3;
                elsif x3='0' then NextState <= a4;
                else NextState <= a5;
                end if;
        when  a2=>  NextState <= a1;
        when  a3=>  NextState <= a1;
        when  a4=> NextState <= a1;
        when  a5=> NextState <= a1;
        when others => NextState <= a1;
        end case;
        end process;
        y1 <= '1' when State=a2 else '0';
        y2 <= '1' when State=a3 else '0';
        y3 <= '1' when State=a4 else '0';
        y4 <= '1' when State=a5 else '0';
        end;
```

Fig. 4. VHDL-model of Moore FSM with correct conditions of transitions

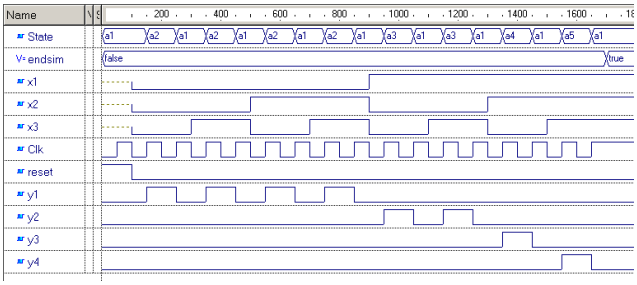

Fig. 5. Timing diagram of Moor FSM with correct conditions of transitions

It reflects the results of simulation in the system ALDEC Active-HDL on all combinations of conditions x_1, x_2, x_3.

The diagram shows that the transitions' conditions function is complete and orthogonal. During the period from 150 ns to 950 ns, the FSM changes to the state a_2 ($y_1 = 1$) as long as the condition $\overline{x_1}$ is true, i.e. ($x_1 = 0$) , and then returns back to a_1 ($y_1 = 0$). During the period from 950 ns to 1350 ns, the FSM changes to the state a_3 ($y_2 = 1$) as long as the condition $x_1\overline{x_2}$ is true, i.e. ($x_1 = 1, x_2 = 0$), and then returns back to a_1 ($y_2 = 0$). During the period from 1350 ns

to 1550 ns, the FSM changes to the state a_4 ($y_3 = 1$) as long as the condition $x_1 x_2 \bar{x}_3$ is true, i.e. ($x_1 = 1, x_2 = 1, x_3 = 0$), and then returns back to a_1 ($y_3 = 0$). During the period from 1550 ns to 1750 ns, the FSM changes to the state a_5 ($y_4 = 1$) as long as the condition $x_1 x_2 x_3$ is true, i.e. ($x_1 = 1, x_2 = 1, x_3 = 1$), and then returns back to a_1 ($y_4 = 0$).

The following example of the state diagram (fig. 6) is considering. Conditions of transitions from the state a_1 are incorrect from the point of view of transitions' conditions functions orthogonalization; during transition to a_4 and a_5 there is no variable x_1 in the term, but they don't contradict conditions of transitions $x_1 x_2 \bar{x}_3$ and $x_1 x_2 x_3$. Transitions' conditions functions are complete.

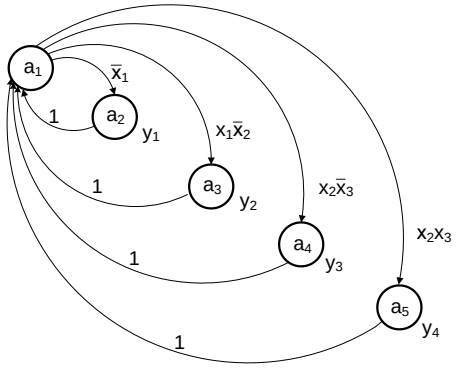

Fig. 6. State diagram of Moore FSM with consistent incomplete conditions of transitions

Transitions from state a_1 can be written in VHDL as follows (fig. 7 (a) or fig. 7 (b)). At the same time, such description is not stylistically correct, but it is not inconsistent and gives the same results during simulation as in fig. 5

```
when  a1=> if x1='0' then NextState <= a2;
           elsif x1='1'and x2='0' then NextState <= a3;
           elsif x2='1'and x3='0' then NextState <= a4;
           else NextState <= a5;
           end if;
```

a)

```
when  a1=> if x1='0' then NextState <= a2;
           elsif x1='1'and x2='0' then NextState <= a3;
           elsif x2='1'and x3='0' then NextState <= a4;
           elsif x2='1'and x3='1' then NextState <= a5;
           end if;
```

b)

Fig. 7. Fragments of the VHDL-model of Moore FSM with consistent incomplete conditions of transitions

In addition, FSM models which are shown in fig. 3 and 6, give exactly the same correct results during synthesis. Synthesis was performed in the system XILINX ISE.

Next, an example of the state diagram (fig. 8) with a missing transition (by condition $x_1 x_2 x_3$) and incomplete condition for the transition from state a_1 to state a_4: $x_2 \bar{x}_3$ instead of $x_1 x_2 \bar{x}_3$ is considering. The transitions' conditions function in this case is non-orthogonal and incomplete.

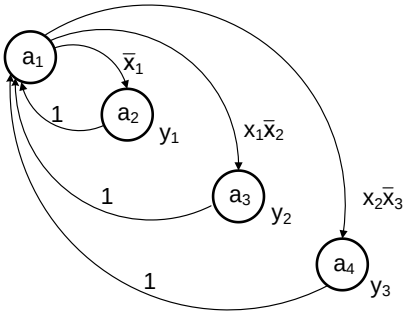

Fig. 8. Moore state diagram with missing transition

A fragment of the VHDL model of this FSM is shown in fig. 9.

```
when  a1=> if x1='0' then NextState <= a2;
           elsif x1='1'and x2='0' then NextState <= a3;
           elsif x2='1'and x3='0' then NextState <= a4;
           end if;
```

Fig. 9. Fragment of the VHDL-model of Moore FSM with missing transition

During simulation of the operation of this FSM (fig. 10), at first glance, everything is fine, but in fact, the variable x_3 is insignificant here, on the set $x_1, x_2, x_3 = 111$ the FSM should not go to any state, but the modeling system put him to the state a_4.

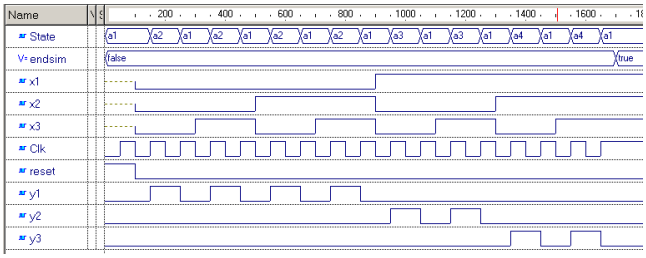

Fig. 10. Timing diagram of Moore FSM with missing transition

Likewise, when conditions $x_1, x_2, x_3 = 010$, the FSM can go into two states a_2 and a_4, which should not be in the case when the FSM works correctly, but the modeling system masks such situation, putting the FSM into the a_2 state. In this example, there is both a lack of completeness and the presence of a contradiction of the transitions' conditions, but this is clearly not manifested at the stages of syntax analysis and simulation.

During synthesis of this FSM can be problems especially in the case of older versions of CAD. For example, when using Xilinx ISE 10.1, a warning appears about four latches in addition to four flip-flops: *Found 4-bit latch for signal <NextState>. Latches may be generated from incomplete case or if statements. We do not recommend the use of latches in FPGA/CPLD designs, as they may lead to timing problems*. That is, instead of two triggers for the four states, 8 triggers of two types are synthesized. This should not be in a correctly synthesized FSM. At the same time, when using the latest version of Xilinx ISE 14.7, this warning will no longer exist.

Thus it is shown that problems associated with incorrect conditions are very difficult to identify during the design process. With equal probability, they can appear both on the timing diagram during behavioral simulation, and in the synthesis process (especially in cases with older versions of CAD). So, the verification of transitions' conditions for consistency and completeness must be carried out at the stage of forming the state diagram of the FSM.

IV. Conclusion

Construction a logical control system based on FPGA is a modern approach to computer-aided design. One of the most common ways to describe logical control systems is the finite state machine model, which description based on the state diagram. The correctness of the future HDL code depends on the correctness of the state diagram.

The concept of orthogonalization, used to decompose logical functions in the synthesis of digital systems [6], can be also used to check the state diagram for correctness [2]. As a result of the research, it was shown that the transitions' conditions function $f(x_1, x_2, \ldots x_n)$ is non contradictory if it is orthogonal. The orthogonal function of transitions' conditions $f(x_1, x_2, \ldots x_n)$, in its turn, is complete if its terms cover all sets $x_1, x_2, \ldots x_n$.

The verification of HDL model is carrying out at all stages of computer-aided design, namely, at the stage of behavioral simulation (by analyzing timing diagrams), at the stage of synthesizing of the RTL circuit (by analyzing the synthesis report) and at the stage of post-synthesis simulation (by analyzing the timing diagrams, taking into account the technological base). Due to the features of modeling system, missing transitions or contradictory conditions of transitions at the syntax checking stage are not fixed, moreover at simulation stage and automated synthesis they may go unnoticed (depending on the version of the synthesizer).

Therefore, verification of the state diagram for correctness is an important and integral step in the automated design of automatic logic control systems, the functioning algorithm of which is presented in the hardware description language.

References

[1] Rashinkar P. System-on-a-chip Verification: Methodology and Techniques Cadence Design Systems / P. Rashinkar, P. Paterson, L. Singh. – Kluwer Academic Publishers, 2002. – 372 p.

[2] Lam W. K. Hardware Design Verification: Simulation and Formal Method-Based Approaches. – Prentice Hall PTR, 2005. – 624 p.

[3] Shkil A. Design of Logical Control Units Based on Finite State Machines' Patterns/ M Miroshnyk; S. Poroshyn; A. Shkil; E. Kulak; I. Filippenko; D. Kucherenko; Y. Pakhomov; J. Salfetnikova; M. Goga//Proceedings of the 2018 IEEE East-West Design & Test Symposium.– 6 p. [Електронний ресурс] / IEEE Xplore Digital Library – Режим доступу: www / URL: https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?filter=issueId%20EQ%20%228524135%22& refinements= Author:Alexander%20Shkil&pageNumber=1&resultAction=REFINE.

[4] Pedroni, V. A. Finite state machines in hardware : theory and design (with VHDL and SystemVerilog) /Volnei A. Pedroni. – Cambridge, MA: MIT Press., 2013. – 338 p.

[5] Shalyto A. State Machine Design Pattern / A. Shalyto, N. Shamgunov, G. Korneev // .NET Technologies 2006 – Shot communication papers conference proceedings. 4-th International Conference in Central Europe on .Net Technologies. University of West Bohemia. May 29 – June 1, 2006 – P. 51-57.

[6] Baranov S. Logic and System Design of Digital Systems / S. Baranov. – Tallinn: TUT Press, 2008. – 267 p.

[7] Zakrevskij, A. Optimization in Boolean Space / A. Zakrevskij, Yu. Pottosin, L. Cheremisinova. - Tallinn: TUT Press, 2009. - 241 p.

[8] Shkil A. Design of real-time logic control system on FPGA / M. Miroschnyk, A. Shkil, E. Kulak, D. Rakhlis, I. Filippenko, M. Hoha, M. Malakhov, V. Serhiienko // Proceedings of 2019 IEEE East-West Design & Test Symposium (EWDTS'19), September 13-16, Batumi, Georgia, 2019. – P.488-491.