Received February 24, 2022, accepted March 14, 2022, date of publication March 24, 2022, date of current version April 7, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3162070

Improving Characteristics of FSMs With Mixed Codes of Outputs

ALEXANDER BARKALOV[1], LARYSA TITARENKO[1], SŁAWOMIR CHMIELEWSKI[2], AND KAMIL MIELCAREK[1]

[1]Faculty of Computer, Electrical and Control Engineering, Institute of Metrology, Electronics and Computer Science, University of Zielona Góra, 65-516 Zielona Gora, Poland

[2]Institute of Science and Technology, State University of Applied Sciences in Głogów, 67-200 Glogow, Poland

Corresponding author: Kamil Mielcarek (k.mielcarek@imei.uz.zgora.pl)

ABSTRACT Practically, any digital system includes sequential blocks. This paper considers a case when LUT-based sequential blocks are represented by Mealy finite state machines (FSMs). The LUT count is one of the most important characteristics of an FSM circuit. In this paper, a method is proposed which aims at decreasing the LUT counts of FPGA-based Mealy FSMs with mixed encoding of the collections of outputs. To do it, a method of encoding of the fields of compatible states is proposed. The proposed approach leads to LUT-based Mealy FSM circuits having three levels of logic blocks. Each function for any logic level is represented by a circuit including a single LUT. There is given an example of FSM synthesis with the proposed method. The experiments are conducted using standard benchmark FSMs. The results of experiments show that the proposed approach produces LUT-based circuits with fewer LUTs than it is for circuits produced by other investigated methods (Auto and One-hot of Vivado, JEDI, the mixed encoding the collections of outputs). The LUT count is decreased by an average of 6.97 to 62.85 percent. These improvements are accompanied by a slight decrease in the maximum operating frequency. The frequency is decreased by up to 8.09%. The advantages of the proposed method increase as the number of FSM inputs and states increases.

INDEX TERMS Mealy FSM, FPGA, LUT, synthesis, mixed encoding of collections of outputs, fields of compatible states.

I. INTRODUCTION

Any digital system includes various combinational and sequential blocks [1]. As a rule, to implement circuits of combinational blocks (such as adders, shifters, decoders, and so on), standard library cells of computer-aided design (CAD) tools can be used [2]. Unfortunately, this approach cannot be used to implement circuits of sequential blocks (such as control units) [3]. A circuit of a sequential block is determined by block's behaviour. To specify the behaviour of a sequential block, it is necessary to use some formal model. In many cases, the models of Mealy finite state machines (FSMs) are used for this purpose [4], [5].

Circuit designers strive to minimize such characteristics of FSM circuits as: the hardware amount, the propagation time, and the power consumption. These characteristics are strongly interconnected [1], [6]. As a rule, the hardware amount (for example, the occupied chip area) has a strong influence on the rest of the FSM circuit characteristics [6], [7]. To optimize the hardware amount, various methods of decomposition can be used [8]. But the structural decomposition leads to an increase in the number of logic levels in the resulting FSM circuits. In turn, an increase in the number of logic levels leads to a decrease in the FSM performance

It is quite possible that a multi-level FSM circuit does not provide the required performance. In this case, it is necessary to reduce the number of logic levels. It is highly desirable that reducing the number of levels does not lead to a sharp increase in hardware amount of a resulting FSM circuit. One of these approaches is proposed in [9]. This is a method of mixed encoding of FSM outputs. This method is advisable to use if FSM circuits are implemented with field-programmable gate arrays (FPGAs) [10]–[12].

Nowadays, FPGAs are very popular platforms for implementing various digital systems [13]. This explains why we chose the FPGA-based Mealy FSMs as a research object in

The associate editor coordinating the review of this manuscript and approving it for publication was Engang Tian.

our current article. In this article, we discuss Mealy FSM circuits implemented using look-up table (LUT) elements, programmable flip-flops and interconnections of FPGAs. We use LUT count as a characteristic of hardware amount. Now, Xilinx is the largest manufacturer of FPGA chips [14]. Due to it, we are orienting this research on solutions of Xilinx. So, we consider ways of reducing the LUT count in FPGA-based Mealy FSMs.

A LUT is a logic block having NI_{LUT} inputs and a single output [10], [14], [15]. If an arbitrary Boolean function has up to NI_{LUT} arguments, then the corresponding circuit is implemented as a single LUT. Unfortunately, the value of NI_{LUT} does not exceed 6 [11], [16]. If a Boolean function depends on more than NI_{LUT} variables, then it should be decomposed using various methods of functional decomposition [1], [17], [18]. The functional decomposition (FD) results in multi-level FSM circuits with complex system of interconnections [19], [20].

As a rule, Mealy FSM circuits are represented by systems of Boolean functions (SBFs). The step of technology mapping [19], [21] is executed to transform these SBFs into a LUT-based FSM circuit. The outcome of this step tremendously affects the LUT count, maximum operating frequency and power consumption of a resulting FSM circuit [1], [22]. As shown in [23], time delays of the interconnection system are starting to play a major role in comparison with LUT delays. Also, more than 70% of the power dissipation is due to the interconnections [16]. So, the optimization of interconnections for LUT-based FSM circuits leads to increasing the operating frequency and reducing the consumption of power. The system of interconnections could be optimizing, for example, due to the twofold state assignment [24], [25].

The characteristics of LUT-based circuits can be improved by the increasing the number of LUT inputs. But, the results of research [13] predict that it is practically impossible to expect an increase in value of NI_{LUT}. As a rule, modern LUTs have no more than 6 inputs [14], [15]. An increase in the number of inputs leads to an imbalance of area-time-power characteristics of a LUT circuit. So, there is an imbalance between the numbers of arguments in SBFs representing FSM circuits and a rather small value of NI_{LUT}. To reduce the impact of this imbalance on quality of FSM circuits, it is necessary to improve synthesis methods of FPGA-based FSMs.

Our current paper considers the synthesis of LUT-based Mealy FSMs obtained using the method of mixed encoding of outputs [9], [26]. The mixed encoding of outputs allows, reducing the LUT counts of FPGA-based Mealy FSMs compared with equivalent FSMs' circuits based on using the methods of functional decomposition [8]. In the best case, this leads to two-level FSM circuits. But it is quite possible that there will be more than one level of LUTs generating input memory functions and additional variables encoding collections of FSM outputs. If performance is the dominant quality factor, then the number of levels in FSM circuit should be reduced. To reduce the number of levels, we propose a new synthesis method based on using codes of fields of compatible states.

The main contribution of this paper is a novel design method aimed at reducing the number of LUTs in circuits of FPGA-based Mealy FSMs with mixed encoding of outputs.

The main idea of the proposed approach is to use the codes of fields of compatible states for the state assignment. This approach is similar to the twofold state assignment [24], [25]. But our new method allows excluding the block of transformation of state codes which is necessary in the case of twofold state assignment. The experimental results show that this method allows decreasing the LUT counts of LUT-based FSMs compared with equivalent FSMs obtained using some known methods of FSM design.

The further text of the article includes five sections. The background of single-level LUT-based Mealy FSMs is shown in the second section. The state-of-the-art in synthesis of LUT-based FSMs is discussed in the third section. The main idea of the proposed approach is considered in the fourth section. In the fifth section, there is shown an example of synthesis of FSM with the encoding of the fields of compatible states. The sixth section includes the results of experiments. A short conclusion ends the paper.

II. BACKGROUND OF LUT-BASED MEALY FSMs

There is a lot of configurable logic blocks (CLB) [11], [14] in FPGAs manufactured by Xilinx. There are such CLBs as embedded memory blocks, digital signal processors, and microprocessors. To connect CLBs, a programmable routing matrix [10] is used. In this paper, we consider FSM design with CLBs consisted of LUTs, multiplexers and programmable flip-flops. A LUT has NI_{LUT} inputs and a single output. Networks of LUTs implement systems of Boolean functions representing an FSM circuit.

If a Boolean function depends on up to NI_{LUT} arguments, then it can be implemented by a single-LUT circuit. To implement sequential circuits, it is necessary to connect combinational outputs of some LUTs with inputs of flip-flops. As a rule, an FSM state register (SRG) is based on D flip-flops [2], [12]. To load state codes into SRG, the pulse of synchronization *Clk* is used. As a rule, the initial state has code with all zeros. To load zeros into all flip-flops of SRG, the pulse *Reset* is used. A programmable multiplexer selects the type of a CLB output (combinational or registered).

As mentioned in [3], [4], SBFs representing FSM circuits may depend on up to 50–70 Boolean variables. At the same time, modern LUTs have no more than 6 inputs. This imbalance leads to applying various methods of functional decomposition (FD) in LUT-based FSM design [1], [17]. FD-based circuits have one serious drawback: they have a lot of logic levels connected by "spaghetti-type" nections [11].

A Mealy FSM is represented by a vector $A = \langle I, O, S, \delta, \lambda, s_1 \rangle$ [3], [4]. The components of A have the following meaning: $I = \{i_1, \ldots, i_L\}$ is a set of inputs, $O = \{o_1, \ldots, o_N\}$ is a set of outputs, $S = \{s_1, \ldots, s_M\}$ is

TABLE 1. STT of Mealy FSM A_0.

s_c	s_T	I_h	o_n	h
s_1	s_2	i_1	$o_1o_2o_3$	1
	s_3	$\bar{i}_1 i_2$	$o_3o_4o_6$	2
	s_4	$\bar{i}_1 \bar{i}_2$	o_2o_7	3
s_2	s_3	i_3	o_4o_8	4
	s_5	$\bar{i}_3$	$o_3o_5o_9$	5
s_3	s_4	i_2	$o_1o_4o_6$	6
	s_3	$\bar{i}_2$	$o_3o_5o_6$	7
s_4	s_5	$i_1 i_2$	$o_2o_3o_6$	8
	s_6	$i_1 \bar{i}_2$	$o_1o_2o_7$	9
	s_7	$\bar{i}_1 i_2$	$o_2o_6o_7$	10
	s_9	$\bar{i}_1 \bar{i}_2$	o_2o_3	11
s_5	s_6	i_4	$o_1o_4o_8$	12
	s_8	$\bar{i}_4$	o_1o_4	13
s_6	s_7	i_5	o_1	14
	s_6	$\bar{i}_5 i_1 i_2$	o_6	15
	s_8	$\bar{i}_5 \bar{i}_1$	o_2o_7	16
s_7	s_8	i_5	$o_3o_5o_6$	17
	s_9	$\bar{i}_5$	o_1o_6	18
s_8	s_1	i_3	–	19
	s_5	$\bar{i}_3$	$o_3o_5o_9$	20
s_9	s_7	i_1	$o_3o_5o_6o_9$	21
	s_4	$\bar{i}_1 i_5$	o_1o_4	22
	s_1	$\bar{i}_1 \bar{i}_5$	–	23

a set of internal states, δ is a function of transitions, λ is a function of outputs, and s_1 is an initial state. Various tools can be used to represent the vector A, such as state transition graphs [4], binary decision diagrams [27], [28], and-inverter graphs [29], [30], graph-schemes of algorithms [3]. In this paper, we represent Mealy FSMs by state transition tables (STTs) [4].

An STT includes five columns [4]: a current state s_C; a state of transition (a next state) s_T; an input signal (a conjunction of FSM inputs) I_h causing a transition from s_C to s_T; a collection of FSM outputs $O_h \subseteq O$ generated during the transition $< s_C, s_T >$; the number of interstate transitions h ($h \in \{1, \ldots, H\}$). For example, Table 1 is the STT of a Mealy FSM A_0.

The following characteristics of FSM A_0 can be found from Table 1: $L = 5$ inputs, $N = 9$ outputs, $M = 9$ states, and $H = 23$ interstate transitions. Table 1 defines functions of transitions and outputs of FSM A_0. For example, the following functions are determined by the first row of Table 1: $\delta(s_1, i_1) = s_2$ and $\lambda(s_1, i_1) = \{o_1, o_2, o_3\}$.

To design an FSM circuit, it is necessary to encode states $s_m \in S$ by binary codes $K(s_m)$ having R_s bits. In this article, we use the maximum binary state assignment with

$$R_S = \lceil log_2 M \rceil. \tag{1}$$

The state variables from the set $T = \{T_1, \ldots, T_{RS}\}$ are used to create the state codes. The state variables are kept into the SRG. The input memory functions (IMFs) can change the contents of SRG. The IMFs create a set $D = \{D_1, \ldots, D_{RS}\}$. Using state codes and IMFs leads to transforming the initial STT into a direct structural table (DST). A DST has three additional columns: a code of a current state $K(s_C)$, a code of a state of transition $K(s_T)$, and a column D_h with a collection of IMFs equal to 1 to load a code $K(s_T)$ into SRG [2].

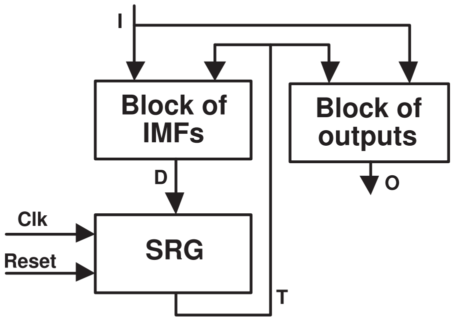


FIGURE 1. Structural diagram of P Mealy FSM.

A DST defines two SBFs representing an FSM circuit. These SBFs are the following:

$$D = D(T, I); \tag{2}$$

$$O = O(T, I). \tag{3}$$

The SBFs (2)–(3) represent a structural diagram of P Mealy FSM (Fig. 1).

In P Mealy FSMs, the block of IMFs is specified by SBF (2), the block of outputs is determined by SBF (3). The inputs of register SRG are connected with outputs of the block of IMFs. The outputs of SRG form a feedback necessary to create a sequential circuit [1]. In each cycle of FSM operation, the SRG contains a code $K(s_C)$. The pulse of synchronization *Clk* allows loading a code $K(s_T)$ into the SRG. If *Reset* $= 1$, then the code $K(s_1)$ is loading into SRG. In this paper, we consider a case when both blocks are implemented using LUTs, flip-flops and interconnections. In this case, the flip-flops of SRG are distributed among LUTs of the block of IMFs.

III. OPTIMIZING CIRCUITS OF FPGA-BASED MEALY FSMs

The first methods of synthesis of FPGA-based FSMs appeared in the mid-1980s. To date, a huge number of works on this topic have been published. The problems connected with synthesis and optimization of LUT-based FSMs are discussed, for example, in [2], [3], [6], [8], [12], [17], [23], [25], [27], [28], [31]. The analysis of numerous literature shows that characteristics of FSM circuits obtained by different synthesis methods may differ significantly. Three main characteristics are considered to estimate the quality of an FSM circuit: 1) the chip resources used for implementing the circuit; 2) the maximum operating frequency (or maximum time of cycle) and 3) the power consumption [2].

In the case of LUT-based FSMs, the following chip resources are used: 1) LUTs; 2) programmable flip-flops; 3) programmable interconnections; 4) the circuit of synchronization and 5) the programmable input-outputs. Obviously, the best circuit requires the minimum possible chip area; it has the highest possible operating frequency and the lowest value of power consumption. However, such a combination of characteristics is almost impossible. For example, a decrease in the LUT count is usually accompanied by a decrease in the performance [2].

In this article, we propose a method for improving the LUT count of FPGA-based Mealy FSMs. The method is based on encoding of fields of compatible states.

Each row of a DST determines the following conjunction F_h:

$$F_h = S_C \wedge I_h\big(h \in \{1, \ldots, H\}\big). \tag{4}$$

In (4), the symbol S_C stands for a conjunction of state variables corresponding to the code of a current state $s_C \in S$ from the h-th row of the DST. The functions (2)–(3) are represented as sum-of-products (SOPs) [4] depended on terms (4). There are $NL(f_j)$ literals in a SOP of a function $f_j \in D \cup O$.

If the condition

$$NL(f_j) \leq NI_{LUT} \tag{5}$$

takes place, then the corresponding circuit consists of a single LUT. This is the best possible solution. In this case, there are exactly $R_S + N$ LUTs in the FSM circuit. This is the best possible LUT count for an FSM circuit. If condition (5) is violated, then the corresponding LUT-based circuit is multi-level.

To implement multi-level LUT-based FSM circuits, various methods of functional decomposition (FD) are used [17]. The FD is a process during which some additional functions appear. It means that an initial SOP is broken down by partial SOPs corresponding to these additional functions. This process is terminated when each partial SOP includes not more than NI_{LUT} literals.

One serious drawback of FD follows from the analysis of work [17]. It is possible that different partial SOPs of the same function includes the same inputs $i_l \in I$ or/and the same state variables $T_r \in T$. This leads to a duplication of literals in different partial SOPs of the original SOP. In turn, this complicates the FSM circuit interconnection system. This phenomenon complicates the placement and routing process. This leads to rather slow FSM circuits with a high value of power consumption [24]–[26]. So, if condition (5) is violated, then the LUT count is equal to $N + R_S + N(F)$, where $N(F)$ is the number of additional functions different from $f_j \in D \cup O$.

To optimize LUT count of multi-level FSM circuits, it is necessary to diminish the value of $N(F)$. To solve this problem, a large number of FD-based methods have been developed [1], [17], [32], [33]. The analysis of some FD-based methods can be found in [1]. Various algorithms of FD are included into CAD tools aimed in implementation of FPGA-based digital systems.

The number of literals in SOPs representing an FSM circuit may be reduced due to the one-hot state assignment [4]. In this case, the following relation takes place: $R_S = M$ [4]. There are M flip-flops in the SRG, if the one-hot state assignment is used. In the case of FPGA-based design, this is not a problem due to a large summarized number of programmable flip-flops in CLBs. So, this approach is very often used in LUT-based FSM design. For example, the one-hot state assignment is used in the academic CAD system ABC by Berkeley [30]. Also, this approach is used in industrial CAD packages such as, for example, Vivado of Xilinx [34] and Quarts of Intel (Altera) [35].

The main drawback of the one-hot state assignment is an increase in the number of IMFs compared with their minimum possible number determined by (1). But these IMFs are much simpler than in the case of maximum binary state assignment [2]. There is a comparison of these state assignment methods in [36]. The research [36] shows that using one-hot codes improves FSM characteristics if there is $M > 16$. However, there is one more factor influencing the LUT-based FSM circuit characteristics. The rather small number of LUT inputs increases the influence of the value of NI_{LUT} on the characteristics of LUT-based FSM circuits [2]. As shown in [37], the maximum state assignment produces better LUT-based FSM circuits if there is $L \geq 10$.

So, sometimes better LUT-based circuits are based on the maximum binary state assignment. But sometimes, the using one-hot state codes gives better results. Thus, it is necessary to check which method will give the best results for a particular FSM. Due to it, we have compared the FSMs circuits produced by our proposed approach with LUT-based circuits of P Mealy FSMs produced by using: 1) the algorithm JEDI [38], 2) the method of binary state assignment Auto and 3) the one-hot state assignment of Vivado [34] by Xilinx [14]. We chose Vivado because we wanted to compare FSM circuits implemented with Xilinx FPGA chips of Virtex 7 family.

The reducing the power consumption is one of the very important problems connected with FPGA-based design [39]–[41]. This problem can be solved due to a special state assignment [42]. The goal of vast majority of these methods is a reducing the switching activity of an FSM circuit [43]. The following rule is used: the more often a pair $< s_i, s_j >$ occurs in an STT, the less is the Hamming distance for state codes $K(s_i)$ and $K(s_j)$ [40]. But this is not only the way for reducing the power consumption. As shown in [24]–[26], the power consumption can be reduced by minimizing the number of interconnections inside an FSM circuit. To reduce the number of interconnections, it is necessary to minimize the numbers of arguments in SBFs (2)–(3) [4]. This can be done using various methods of structural decomposition [8].

The structural decomposition (SD) is an efficient way of reducing LUT counts in Mealy FSMs logic circuits [8]. The main idea of these methods is the elimination of a direct connection between FSM inputs $i_l \in I$ and state variables $T_r \in T$, on the one hand, and outputs $o_n \in O$ and IMFs $D_r \in D$, on the other hand. To do it, some additional functions are introduced. These functions form a set F_{add} having $N(F_{add})$ elements. The functions $f_j \in F_{add}$ depend on the FSM inputs and state variables. In turn, the FSM outputs and IMFs use the functions $f_j \in F_{add}$ as arguments of corresponding SOPs. The structural decomposition leads to reducing the number of

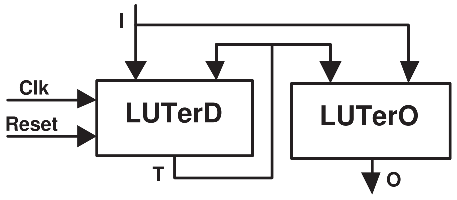


FIGURE 2. Structural diagram of LUT-based P FSM.

LUTs, if the following conditions are true [8]:

$$N(F_{add}) \ll N + R_S; \tag{6}$$

$$N(F_{add}) \ll L + R_S. \tag{7}$$

It is possible that FD- and SD-based methods should be used together. It should be done if condition (5) is violated for some functions $f_j \in F_{add}$ [8].

The encoding of collections of outputs (COs) is an SD-based method having its roots in the microprogram control units [44]. In the 1950s, this method was used for reducing the number of bits in control memory words [45]. Next, it was used to optimize hardware of FSM circuits implemented with various programmable logic devices [2]. It is also used in FPGA-based FSMs [8].

For LUT-based FSMs, systems (2)–(3) determine a P FSM (Fig. 2). In his circuit, a LUTer is a circuit consisting of the LUTs, flip-flops, and programmable interconnections.

In P FSM, the LUTerD implements SBF (2), the LUTerO generates functions (3). The state register is distributed between LUTs of LUTerD. If the condition (5) if violated, then there is more than a single level of logic in this circuit.

If an STT includes Q different COs, then it is enough R_Q variables to encode them:

$$R_Q = \lceil log_2 Q \rceil. \tag{8}$$

The encoding of COs presumes a representing each CO $Y_q \subseteq O$ by a binary code $K(Y_q)$. The variables $z_r \in Z$ are used for the encoding, where $|Z| = R_Q$. Now, we can turn a P FSM into a PY Mealy FSM (Fig. 3).

In PY FSM, the LUTerD implements SBF (2). The LUTerZ implements the SBF

$$Z = Z(T, I). \tag{9}$$

$$O = O(Z). \tag{10}$$

To generate SBFs (2), (9), and (10), it s necessary [8]:

1) To encode states $s_m \in S$ by binary codes $K(s_m)$.
2) To create the DST of P FSM.
3) To encode COs $Y_q \subseteq O$ by binary codes $K(Y_q)$.
4) To transform the DST of P FSM into a DST of PY Mealy FSM.
5) To create a table of LUTerO.

Consider this process for Mealy FSM A_0 (Table 1). For FSM A_0, there is $M = 9$. Using (1) gives $R_S = 4$, $T = \{T_1, \ldots, T_4\}$, $D = \{D_1, \ldots, D_4\}$. Let us encode states in trivial way: $K(s_1) = 0000, K(s_2) = 0001, \ldots, K(s_9) = 1000$.

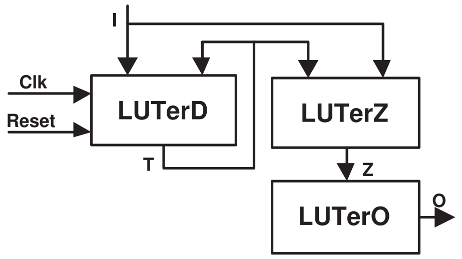


FIGURE 3. Structural diagram of PY Mealy FSM.

Using these codes allows creating the DST of FSM A_0 (Table 2).

Analysis of Table 2 gives the following COs: $Y_1 = \emptyset$, $Y_2 = \{o_1, o_2, o_3\}$, $Y_3 = \{o_3, o_4, o_6\}$, $Y_4 = \{o_2, o_7\}$, $Y_5 = \{o_4, o_8\}$, $Y_6 = \{o_3, o_9\}$, $Y_7 = \{o_1, o_4, o_6\}$, $Y_8 = \{o_3, o_5, o_6\}$, $Y_9 = \{o_2, o_3, o_6\}$, $Y_{10} = \{o_1, o_7\}$, $Y_{11} = \{o_2, o_4, o_6\}$, $Y_{12} = \{o_2, o_3\}$, $Y_{13} = \{o_1, o_4, o_8\}$, $Y_{14} = \{o_1, o_4\}$, $Y_{15} = \{o_1\}$, $Y_{16} = \{o_6\}$, $Y_{17} = \{o_1, o_6\}$, $Y_{18} = \{o_3, o_5, o_6, o_9\}$. There is $Q = 18$. Using (8) gives $R_Q = 5$ and $Z = \{z_1, \ldots, z_5\}$. Let us encode these COs in the trivial way: $K(Y_1) = 00000, K(Y_2) = 00001, \ldots, K(Y_{18}) = 10001$.

In a DST of PY FSM, the column O_h is replaced by the column Z_h. The column Z_h contains variables $z_r \in Z$ equal to 1 in a code $K(Y_q)$ of a CO from the h-th row of DST of P Mealy FSM. In the discussed case, the DST of PY FSM A_0 is represented by Table 3.

A table of LUTerO includes the columns Y_q, $K(Y_q)$, O_q, q. This table is constructed in the trivial way [3]. There are $Q = 18$ rows in the table of LUTerO in the discussed case. Five rows of this table are shown in Table 4.

If the condition

$$R_Q > NI_{LUT} \tag{11}$$

takes place, then the circuit of LUTerO is multi-level. To reduce the number of levels up to 1, the method of mixed encoding of COs (MECO) is proposed in [9].

The main idea of MECO is the following [9]. Consider the COs $Y_3 = \{o_1, o_2\}$ and $Y_4 = \{o_1, o_2, o_3\}$. If the output $o_3 \in O$ is eliminated from Y_4, then the following relation is true: $Y_3 = Y_4$. Now, there are $Q - 1$ COs instead of Q. The process of elimination is continued till the following condition is true:

$$R_Q = \lceil log_2 Q_0 \rceil. \tag{12}$$

In (12), the symbol Q_0 stands for the number of COs after eliminating some outputs.

The eliminated outputs form a set O_{oh}. The other outputs create a set $O_{mb} = O/O_{oh}$. The outputs $o_n \in O_{oh}$ are represented by one-hot codes. The outputs $o_n \in O_{mb}$ form COs which are encoded by binary maximum codes. This leads to PY_M Mealy FSM (Fig. 4).

In PY_M FSM, the LUTerD implements SBF (2). The LUTerZ implements SBFs (9) and

$$O_{oh} = O_{oh}(T, I). \tag{13}$$

TABLE 2. DST of P Mealy FSM A_0.

s_c	$K(s_c)$	s_T	$K(s_T)$	I_h	O_h	D_h	h
s_1	0000	s_2	0001	i_1	$o_1o_2o_3$	D_4	1
		s_3	0010	$\bar{i}_1 i_2$	$o_3o_4o_6$	D_3	2
		s_4	0010	$\bar{i}_1\bar{i}_2$	o_2o_7	D_3D_4	3
s_2	0001	s_3	0010	i_3	o_4o_8	D_3	4
		s_5	0100	$\bar{i}_3$	$o_3o_5o_9$	D_2	5
s_3	0010	s_4	0011	i_2	$o_1o_4o_6$	D_3D_4	6
		s_3	0010	$\bar{i}_2$	$o_3o_5o_6$	D_3	7
s_4	0011	s_5	0100	i_1i_2	$o_2o_3o_6$	D_2	8
		s_6	0110	$i_1\bar{i}_2$	$o_1o_2o_7$	D_2D_3	9
		s_7	0111	$\bar{i}_1 i_2$	$o_2o_6o_7$	$D_2D_3D_4$	10
		s_9	1000	$\bar{i}_1\bar{i}_2$	o_2o_3	D_1	11
s_5	0100	s_6	0101	i_4	$o_1o_4o_8$	D_2D_4	12
		s_8	0111	$\bar{i}_4$	o_1o_4	$D_2D_3D_4$	13
s_6	0101	s_7	0110	i_5	o_1	D_2D_3	14
		s_6	0101	$\bar{i}_5 i_1 i_2$	o_6	D_2D_4	15
		s_8	0111	$\bar{i}_5\bar{i}_1$	o_2o_7	$D_2D_3D_4$	16
s_7	0110	s_8	0111	i_5	$o_3o_5o_6$	$D_2D_3D_4$	17
		s_9	1000	$\bar{i}_5$	o_1o_6	D_1	18
s_8	0111	s_1	0000	i_3	–	–	19
		s_5	0100	$\bar{i}_3$	$o_3o_5o_9$	D_2	20
s_9	1000	s_7	0110	i_1	$o_3o_5o_6o_9$	D_2D_3	21
		s_4	0011	$\bar{i}_1 i_5$	o_1o_4	D_3D_4	22
		s_1	0000	$\bar{i}_1\bar{i}_5$	–	–	23

TABLE 3. DST of PY Mealy FSM A_0.

s_c	$K(s_c)$	s_T	$K(s_T)$	I_h	Z_h	D_h	h
s_1	0000	s_2	0001	i_1	z_5	D_4	1
		s_3	0010	$\bar{i}_1 i_2$	z_4	D_3	2
		s_4	0010	$\bar{i}_1\bar{i}_2$	z_4z_5	D_3D_4	3
s_2	0001	s_3	0010	i_3	z_3	D_3	4
		s_5	0100	$\bar{i}_3$	z_3z_5	D_2	5
s_3	0010	s_4	0011	i_2	z_3z_4	D_3D_4	6
		s_3	0010	$\bar{i}_2$	$z_3z_4z_5$	D_3	7
s_4	0011	s_5	0100	i_1i_2	z_2	D_2	8
		s_6	0110	$i_1\bar{i}_2$	z_5	D_2D_3	9
		s_7	0111	$\bar{i}_1 i_2$	z_2z_4	$D_2D_3D_4$	10
		s_9	1000	$\bar{i}_1\bar{i}_2$	$z_2z_4z_5$	D_1	11
s_5	0100	s_6	0101	i_4	z_2z_3	D_2D_4	12
		s_8	0111	$\bar{i}_4$	$z_2z_3z_5$	$D_2D_3D_4$	13
s_6	0101	s_7	0110	i_5	$z_2z_3z_4$	D_2D_3	14
		s_6	0101	$\bar{i}_5 i_1 i_2$	$z_2z_3z_4z_5$	D_2D_4	15
		s_8	0111	$\bar{i}_5\bar{i}_1$	z_4z_5	$D_2D_3D_4$	16
s_7	0110	s_8	0111	i_5	$z_3z_4z_5$	$D_2D_3D_4$	17
		s_9	1000	$\bar{i}_5$	z_1	D_1	18
s_8	0111	s_1	0000	i_3	–	–	19
		s_5	0100	$\bar{i}_3$	z_3z_5	D_2	20
s_9	1000	s_7	0110	i_1	z_1z_5	D_2D_3	21
		s_4	0011	$\bar{i}_1 i_5$	$z_2z_3z_5$	D_3D_4	22
		s_1	0000	$\bar{i}_1\bar{i}_5$	–	–	23

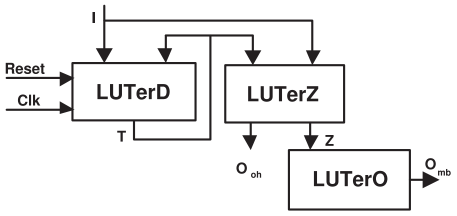


FIGURE 4. Structural diagram of PY_M Mealy FSM.

TABLE 4. Part of table of LUTerO of PY FSM A_0.

Y_q	$K(Y_q)$	O_q	q
Y_1	00000	–	1
Y_2	00001	$o_1o_2o_3$	2
Y_3	00010	$o_3o_4o_6$	3
Y_4	00011	o_2o_7	4
Y_5	00100	o_4o_8	5

The LUTerO implements SBF

$$O_{mb} = O_{mb}(Z). \tag{14}$$

As shown in [9], the replacement of PY FSM by equivalent PY_M FSM allows reducing the LUT count, if the condition (11) takes place. If condition (5) takes place for functions $f_i \in D \cup Z \cup O_{oh}$, then the maximum operating frequency of PY_M FSM is higher than this is for an equivalent PY FSM. This is connected with the fact that LUTerO of the PY FSM is represented by a multi-level circuit.

If the condition (5) is violated for functions $f_i \in D \cup Z \cup O_{oh}$, then both LUTerD and LUTerZ are represented by multi-level circuits. To implement these circuits, the methods of FD should be used. In this article, we propose an approach allowing to reduce the LUT counts in PY_M FSMs. The proposed method is based on the idea of encoding of fields of compatible states (FCSs).

IV. MAIN IDEA OF PROPOSED METHOD

The proposed method is based on the existence of a state compatibility. A state $s_m \in S$ determines three sets. A set $I(s_m)$ includes inputs $i_l \in I$ determining transitions from the state $s_m \in S$. A set $O(s_m)$ includes outputs generating during transitions from the state $s_m \in S$. A set $D(s_m)$ includes IMFs $D_r \in D$ equal to 1 to load into SRG codes of states of transitions from the state $s_m \in S$. For example, the following sets can be found from DST of P Mealy FSM A_0 (Table 2): $I(s_1) = \{i_1, i_2\}$, $O(s_1) = \{o_1, o_2, o_3, o_4, o_6, o_7\}$ and $D(s_1) = \{D_3, D_4\}$. If the encoding of COs $Y_q \subseteq O$ is used then the set $O(s_m)$ is replaced by a set $Z(s_m)$. The set $Z(s_m)$ includes variables $z_r \in Z$ generated during transitions from the state $s_m \in S$. The set $Z(s_1) = \{z_4, z_5\}$ can be found from Table 3.

In this paper, we consider the compatibility of states with respect to the value of NI_{LUT}. If a set $S^k \subseteq S$ includes M_k states, then it is enough

$$R_k = \lceil log_2(M_k + 1) \rceil \tag{15}$$

variables to encode states $s_m \in S^k$. In (15), the one is added to account for the relation $s_m \notin S^k$. The set $S^k \subseteq S$ determines a set $I(S^k)$ with inputs causing transitions from states $s_m \in S^k$. This set includes L_k elements.

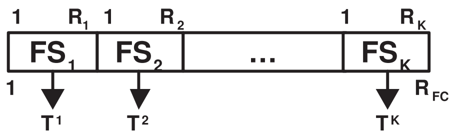


FIGURE 5. Structure of state codes $FC(s_m)$.

States $s_m \in S^k$ are compatible if the following condition takes place:

$$L_k + R_k \leq NI_{LUT}. \quad (16)$$

The following specific characterizes the set S^k: any function generated during transitions from states $s_m \in S^k$ is represented by a single LUT with NI_{LUT} inputs. Of course, this is true, if state codes have R_k bits determined by (15).

Let us find a partition $\Pi_S = \{S^1, \ldots, S^K\}$ of the set S by classes of compatible states. Each class $S^k \in \Pi_S$ determines a field FS_k in the state codes $FC(s_m)$. There is a structure of the code $FC(s_m)$ shown in Fig. 5.

There are K fields in the code $FC(s_m)$. There are R_k bits in the field FS_k. To encode states $s_m \in S^k$, the variables $T_r \in T^k$ are used, where $|T^k| = R_k$. There are

$$R_{FC} = R_1 + R_2 + \ldots + R_K \quad (17)$$

elements in the set $T = T^1 \cup T^2 \cup \ldots \cup T^K$. Obviously, there are R_{FC} bits in the code $FC(s_m)$.

To construct partition Π_S with a minimum number of classes, the methods [24], [25] can be used. These methods were used in the twofold state assignment. In the twofold state assignment, each state has two codes. A code $K(s_m)$ determines a state $s_m \in S$ as in element of the set S. A code $C(s_m)$ determines state $s_m \in S$ as an element of the set $S^k \in \Pi_S$. Having two types of state codes requires a special block of transformation $K(s_m)$ into $C(s_m)$. This block consumes some resources of FPGA chip.

In this paper, we propose to use only codes of FCS. This leads to $P_{FC}Y_M$ Mealy FSM shown in Fig. 6.

In $P_{FC}Y_M$ Mealy FSM, a LUTerk corresponds to the class $S^k \in \Pi_S$. The variables $T_r \in T^k$ encode states $s_m \in S^k$. The LUTerk implements the following systems of partial functions:

$$D^k = D^k(T^k, I^k); \quad (18)$$
$$Z^k = Z^k(T^k, I^k); \quad (19)$$
$$O^k_{oh} = O^k_{oh}(T^k, I^k). \quad (20)$$

The LUTerOR generates functions $o_n \in O_{oh}$, $z_r \in Z$, and $D_r \in D$ as disjunctions of partial functions:

$$o_n = \bigvee_{k=1}^{K} o^k_n (o_n \in O_{oh}); \quad (21)$$

$$z_r = \bigvee_{k=1}^{K} z^k_r (r = \overline{1, R_Q}); \quad (22)$$

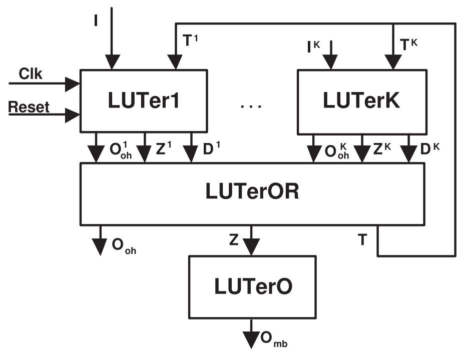


FIGURE 6. Structural diagram of $P_{FC}Y_M$ Mealy FSM.

$$D_r = \bigvee_{k=1}^{K} D^k_r (r = \overline{1, R_{FC}}). \quad (23)$$

The functions $D_r \in D$ enter flip-flops of SRG. The state variables $T_r \in T$ are outputs of SRG.

The LUTerO implements the SBF (14). If condition (12) takes place, then there are exactly $|O_{mb}|$ LUTs in the circuit of LUTerO.

To optimize hardware for the first circuit level, it is necessary to minimize the numbers of shared elements into eponymous sets I^k, O^k_{oh}, Z_k and D^k;

$$|I^q \cap I^g| \rightarrow min; \quad (24)$$
$$|O^q_{oh} \cap O^g_{oh}| \rightarrow min; \quad (25)$$
$$|Z^q \cap Z^g| \rightarrow min; \quad (26)$$
$$|D^q \cap D^g| \rightarrow min. \quad (27)$$

In (24)–(27), there is $q \neq g$ and $q, g \in \{1, \ldots, K\}$.

Meeting the condition (24) allows reducing the requirements for the electrical power of the input sources. In addition, the placement and routing process is simplified for the first level of $P_{FC}Y_M$ FSM. If conditions (25)–(27) take places, then the number of LUTs is reduced in LUTer1–LUTerK. In addition, the number of interconnections between LUTer1–LUTerK and LUTerOR is reduced, too.

If there is

$$K \leq NI_{LUT}, \quad (28)$$

then there are exactly $N_{oh} + R_Q + R_{FC}$ LUTs in the circuit of LUTerOR, where $N_{oh} = |O_{oh}|$. Also, there is only a single level of LUTs in LUTerOR. This is the best case.

Let $A(f_i)$ be the number of occurrences of the function $f_i \in O_{oh} \cup Z \cup D$ in the different sets O^k, Z^k, D^k. Obviously, if there is

$$A(f_i) \leq NI_{LUT}, \quad (29)$$

then a circuit implementing this function includes only a single LUT. This is why it is so important to find a partition Π_S that satisfies the conditions (25)–(27).

In this article, we propose a design method for $P_{FC}Y_M$ Mealy FSMs. The STT is used to represent an FSM. The method includes the following steps:

1) Constructing COs $Y_q \subseteq O$.
2) Representing set O as $O_{oh} \cup O_{mb}$.
3) Executing encoding of COs $Y_q \subseteq O_{mb}$ and creating SBF (14).
4) Creating the partition Π_S satisfying (24)–(27).
5) Encoding of states $s_m \in S$ by codes of FCSs.
6) Creating DST of $P_{FC}Y_M$ FSM.
7) Creating tables of LUTer1–LUTerK.
8) Deriving SBFs (18)–(20).
9) Creating table of LUTerOR.
10) Deriving SBFs (21)–(23).
11) Implementing FSM circuit with resources of a particular FPGA chip.

V. EXAMPLE OF SYNTHESIS

Consider an example of synthesis of $P_{FC}Y_M$ Mealy FSM A_0. The circuit is implemented using LUTs with $NI_{LUT} = 4$.

Step 1. As we found before, there are $Q = 18$ COs in the case of A_0. So, there is $R_Q = 5$. This means the condition (11) takes place. So, it is necessary to use the MECO approach.

Step 2. Using the method [9] gives the flowing sets of outputs: $O_{oh} = \{o_6\}$, $O_{mb} = \{o_1, \ldots, o_5, o_7, o_8, o_9\}$. Eliminating the output o_6 from COs leads to the new COs shown in Table 5.

Now, there is $Q_0 = 12$. Using (8) gives $R_Q = 4$. So the condition (12) is true. The division of set O is terminated. Also, there is the set $Z = \{z_1, \ldots, z_4\}$.

TABLE 5. Collections of outputs for FSM A_0.

Y_q	Outputs	Y_q	Outputs	Y_q	Outputs
Y_1	–	Y_5	o_4o_8	Y_9	o_2o_3
Y_2	$o_1o_2o_3$	Y_6	$o_3o_5o_9$	Y_{10}	$o_1o_2o_7$
Y_3	o_3o_4	Y_7	o_1o_4	Y_{11}	$o_1o_4o_8$
Y_4	o_2o_7	Y_8	o_3o_5	Y_{12}	o_1

Step 3. Let us encode COs $Y_q \subseteq O_{mb}$ in a way minimizing the number of literals in SOPs (10). This minimizes the number of interconnections between blocks LUTerOR and LUTerO. To do it, the method [46] can be used. One of the possible solutions is shown in Fig. 7.

Using Table 5 and Fig. 7, the following SBFs can be obtained:

$$
\begin{aligned}
o_1 &= Y_2 \vee Y_7 \vee Y_{10} \vee Y_{11} \vee Y_{12};\\
o_2 &= Y_2 \vee Y_4 \vee Y_9 \vee Y_{10} = z_4;\\
o_3 &= Y_2 \vee Y_3 \vee Y_6 \vee Y_8 \vee Y_9 = z_1\bar{z}_3 \vee z_1\bar{z}_2;\\
o_4 &= Y_3 \vee Y_5 \vee Y_7 \vee Y_{11} = z_2;\\
o_7 &= Y_4 \vee Y_{10} = \bar{z}_1 z_4;\\
o_8 &= Y_5 \vee Y_{11} = \bar{z}_1 z_2;\\
o_9 &= Y_6 = z_1\bar{z}_2\bar{z}_3\bar{z}_4. \qquad (30)
\end{aligned}
$$

As follows from (30), there are 20 literals in SOPs of functions $o_n \in O_{mb}$. The maximum possible number of literals is determined as $N_{mb} * R_Q = 36$. So, using codes (Fig. 7) allows reducing the number of literals by 1.8 times.

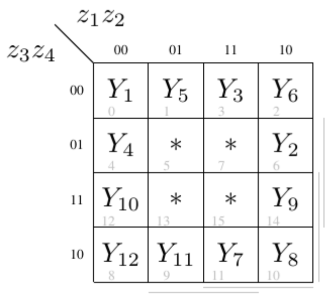


FIGURE 7. Outcome of encoding of COs.

Due to (12), the maximum number of LUTs in LUTerO is equal to N_{mb}. In the discussed case, functions o_2 and o_4 are generated by LUTerOR. So, in the discussed case, there are 7 LUTs in LUTerO.

Step 4. This step is very important [8]. Its outcome influences significantly the number of LUTs in FSM circuit [24]. Using approach [24], [25] gives the partition $\Pi_S = \{S^1, S^2, S^3\}$ where $S^1 = \{s_1, s_3, s_4\}$, $S^2 = \{s_2, s_5, s_8\}$, $S^3 = \{s_6, s_7, s_9\}$.

So, there are three FCSs in the discussed case. As follows from Table 1, this partition determines the sets $I^1 = \{i_1, i_2\}$, $I^2 = \{i_3, i_4\}$, and $I^3 = \{i_1, i_5\}$. There is $|I^1 \cap I^2| + |I^1 \cap I^3| \cap |I^2 \cap I^3| = 1$. This value is very close to minimum.

Step 5. In the discussed case, each class includes 3 elements. Using (15) gives $R_1 = R_2 = R_3 = 2$. Using (17) gives $R_{FC} = 6$. This determines the following sets: $T^1 = \{T_1, T_2\}$, $T^2 = \{T_3, T_4\}$, $T^3 = \{T_5, T_6\}$, and $T = \{T_1, \ldots, T_6\}$.

Due to (16), the outcome of encoding of states does not affect the number of LUTs in FSM circuit [24], [25]. We use codes 00 to show that states do not belong to a particular class. One of the possible outcomes of state assignment is shown in Table 6.

Step 6. There are the following columns in a DST of $P_{FC}Y_M$ FSM: s_C, $FC(s_C)$, s_T, $FC(s_T)$, I_h, O_{oh}, Z_h, D_h, h. The meaning of these columns is clear from Table 7.

In Table 7, the state codes are taken from Table 6, the COs are taken from Table 5. The codes $K(Y_q)$ are taken from Fig. 7.

Step 7. Tables of LUTer1–LUTer3 are created as parts of DST (Table 7). There are the same columns in the tables of LUTer1–LUTer3 and DST (Table 7). Because only R_k bits of $FC(s_C)$ include state codes for states $s_m \in S^k$, then only R_k bits are written on the column $FC(s_C)$ of the table of LUTerk. For example, the LUTer1 is represented by Table 8.

In column $FC(s_C)$, the variables $T_1, T_2 \in T$ are written (Table 8). We treat other state variables as "don't care" [4].

Tables of LUTer2 and LUTer3 are constructed in the same manner. We do not show them in this example.

Step 8. The SBFs (18)–(20) are derived from tables of LUTerk ($k \in \{1, \ldots, K\}$). The terms of SOPs (18)–(20) are determined by (4). For example the following product terms can be derived from Table 8: $F_1 = \bar{T}_1 T_2 i_1, \ldots, F_9 = T_1 T_2 \bar{i}_1 \bar{i}_2$.

TABLE 6. Codes of FCSs of Mealy FSM A_0.

State	FS_1		FS_2		FS_2	
	T_1	T_2	T_3	T_4	T_5	T_6
S_1	0	1	0	0	0	0
S_2	0	0	0	1	0	0
S_3	1	0	0	0	0	0
S_4	1	1	0	0	0	0
S_5	0	0	1	0	0	0
S_6	0	0	0	0	0	1
S_7	0	0	0	0	1	0
S_8	0	0	1	1	0	0
S_9	0	0	0	0	1	1

TABLE 7. DST of $P_{FC}Y_M$ Mealy FSM A_0.

s_c	$FC(s_c)$	s_T	$FC(s_T)$	I_h	O_{oh}	Z_h	D_h	h
s_1	010000	s_2	000100	i_1	–	z_1z_4	D_4	1
		s_3	100000	$\bar{i}_1i_2$	o_6	z_1z_2	D_3	2
		s_4	110000	$\bar{i}_1\bar{i}_2$	–	z_4	D_3D_4	3
s_2	000100	s_3	100000	i_3	–	z_2	D_3	4
		s_5	110000	$\bar{i}_3$	–	z_1	D_2	5
s_3	100000	s_4	110000	i_2	o_6	$z_1z_2z_3$	D_3D_4	6
		s_3	100000	$\bar{i}_2$	o_6	z_1z_3	D_3	7
s_4	110000	s_5	001000	i_1i_2	o_6	$z_1z_3z_4$	D_2	8
		s_6	000001	$i_1\bar{i}_2$	–	z_3z_4	D_2D_3	9
		s_7	000010	$\bar{i}_1i_2$	o_6	z_4	D_5	10
		s_9	000011	$\bar{i}_1\bar{i}_2$	–	$z_1z_3z_4z_5$	D_5D_6	11
s_5	001000	s_6	000001	i_4	–	$z_1z_2z_3$	D_6	12
		s_8	001100	$\bar{i}_4$	–	$z_1z_2z_3z_5$	D_3D_4	13
s_6	000001	s_7	000010	i_5	–	$z_2z_3z_4$	D_2D_3	14
		s_6	000001	$\bar{i}_5i_1i_2$	o_6	$z_2z_3z_4z_5$	D_2D_4	15
		s_8	001100	$\bar{i}_5\bar{i}_1$	o_6	z_4z_5	$D_2D_3D_4$	16
s_7	000010	s_8	001100	i_5	o_6	$z_3z_4z_5$	$D_2D_3D_4$	17
		s_9	000011	$\bar{i}_5$	o_6	z_1	D_1	18
s_8	001100	s_1	010000	i_3	–	–	–	19
		s_5	001000	$\bar{i}_3$	–	z_3z_5	D_2	20
s_9	000011	s_7	000010	i_1	o_6	z_1z_5	D_2D_3	21
		s_4	110000	$\bar{i}_1i_5$	–	$z_2z_3z_5$	D_3D_4	22
		s_1	010000	$\bar{i}_1\bar{i}_5$	–	–	–	23

The functions (18)–(20) are constructed in the trial way. For example, the following SOPs can be derived from Table 8:

$$
\begin{aligned}
o_6^1 &= F_2 \vee F_4 \vee F_6 \vee F_8 = \bar{T}_1T_2\bar{i}_1i_2 \vee \ldots;\\
z_1^1 &= F_1 \vee F_2 \vee F_4 \vee F_5 \vee F_6 \vee F_9 = \bar{T}_1T_2i_1 \vee \ldots;\\
D_1^1 &= [F_2 \vee F_3] \vee [F_4 \vee F_5] = \bar{T}_1T_2\bar{i}_1 \vee T_1\bar{T}_2. \qquad (31)
\end{aligned}
$$

The superscript "1" in (31) means that the corresponding functions are generated by LUTer1. There are 11 different elements in the columns O_{oh}, Z_h and D_h of Table 8. So, there are 11 LUTs in the circuit of LUTer1.

Analysis of Table 7 shows that the following variables are generated during the transitions from $s_m \in S^2$: $z_1, z_2, z_3, D_1, D_2, D_3, D_4, D_5, D_6$. So, there are 8 LUTs in the circuit of LUTer2. Next, we can find that variables $o_6, z_1, z_2, z_3, D_2, D_3, D_5, D_6$ are generated during transitions from states $s_m \in S^3$. So, there are 9 LUTs in the circuit

TABLE 8. DST of $P_{FC}Y_M$ Mealy FSM A_0.

s_c	$FC(s_c)$	s_T	$FC(s_T)$	I_h	o_{oh}	Z_h	D_h	h
s_1	01	s_2	000100	i_1	–	z_1z_4	D_4	1
		s_3	100000	$\bar{i}_1i_2$	o_6	z_1z_2	D_3	2
		s_4	110000	$\bar{i}_1\bar{i}_2$	–	z_4	D_3D_4	3
s_3	10	s_4	110000	i_2	o_6	$z_1z_2z_3$	D_3D_4	4
		s_3	100000	$\bar{i}_2$	o_6	z_1z_3	D_3	5
s_4	11	s_5	001000	i_1i_2	o_6	$z_1z_3z_4$	D_2	6
		s_6	000001	$i_1\bar{i}_2$	–	z_3z_4	D_2D_3	7
		s_7	000010	$\bar{i}_1i_2$	o_6	z_4	D_5	8
		s_9	000011	$\bar{i}_1\bar{i}_2$	–	$z_1z_3z_4z_5$	D_5D_6	9

TABLE 9. Table of LUTerOR of P_CY_M FSM A_0.

Function	Class			Function	Class		
	1	2	3		1	2	3
D_1	1	1	1	o_6	1	0	1
D_2	1	1	1	z_1	1	1	1
D_3	1	1	1	z_2	1	1	1
D_4	1	1	0	z_3	1	1	1
D_5	1	0	1	z_4	1	0	0
D_6	1	1	0	–	–	–	–

of LUTer3. Totally, there are 28 LUTs in the circuits of LUTer1–LUTer3.

Step 9. The LUTs of LUTerOR execute disjunctions of eponymous functions with equal superscripts. In the discussed case, LUTerOR is represented by Table 9.

As follows from Table 9, there is no need in LUT for z_4. So, there are 10 LUTs in LUTerOR.

Step 10. The functions (21)–(23) are constructed using Table 9. The following SOPs, for example, can be derived:

$$
o_6 = o_6^1 \vee o_3^6;\ z_1 = z_1^2 \vee z_1^2 \vee z_1^3;\ D_4 = D_4^1 \vee D_4^2. \qquad (32)
$$

Step 11. This step is connected with the solution of different problems of technology mapping [1]. We do not discuss this step for our example.

So, there are 28 LUTs in LUTer1–LUTer3, 10 LUTs in LUTerOR, and 6 LUTs in LUTerO. This gives 44 LUTs in the circuit of $P_{FC}Y_M$ Mealy FSM A_0. There are exactly 3 levels of LUTs in this circuit.

VI. EXPERIMENTAL RESULTS

In this section, the results of experiments are shown. In these experiments, there are used benchmark FSMs from the library [47]. The library includes 48 benchmarks represented in the format KISS2. These benchmarks have a wide range of basic characteristics (the numbers of states, inputs, and outputs). The characteristics of these benchmarks can be found in many articles and books, for example, in [1], [7], [24], [28], [43]. Different researchers use these benchmarks to compare area and time characteristics of FSMs obtained using new and known design methods.

To conduct the experiments, we used a personal computer having the following characteristics: CPU:

Intel Core i7 6700K 4.24.4GHz, Memory: 16GB RAM 2400MHz CL15. As a platform for implementing FSM circuits we used the Virtex-7 VC709 Evaluation Platform (xc7vx690tffg1761-2) [48]. The FPGA chip of this platform includes LUTs with 6 inputs. To execute the technology mapping, CAD tool Vivado v2019.1 (64-bit) [34] was used. The results of experiments are taken from reports produced by Vivado. As the source information for the CAD tool, we used VHDL-based FSM models obtained by the transformation files in KISS2 format into VHDL codes. The transformation is executed by the CAD tool K2F [8].

As a rule, the LUT count (the number of LUTs) is used to estimate a chip area occupied by an FSM circuit [48]–[50]. Using results of experiments, we compared area (the LUT count) and time (the maximum operating frequency) characteristics of FSMs based on five different approaches. Three of them are P Mealy FSMs based on: 1) Auto of Vivado (it uses binary state codes); 2) One-hot of Vivado; 3) JEDI. The fourth objects for comparison are PY_M-based FSMs [8], [9] shown in Fig. 4. In the case of PY_M-based FSMs, we use JEDI to encode state codes. We compared the characteristics of these four FSMs with P_FCY_M-based FSM circuits.

It is known [8] that area and time characteristics of LUT-based FSM circuits depend strongly on the relation between numbers of inputs (L) and state variables (R_S), on the one hand, and the number of LUT inputs, on the other hand. As in our previous research [24], we have divided the benchmarks into five classes.

To divide the benchmarks into classes, we used numbers that are multiples of 6 (because in our experiments we used LUTs with $NI_{LUT} = 6$). The benchmarks belong to class of trivial FSMs (class 0), if $R_S + L \leq 6$. The benchmarks belong to class of simple FSMs (class 1), if $R_S + L \leq 12$. The benchmarks belong to class of average FSMs (class 2), if $R_S + L \leq 18$. The benchmarks belong to class of big FSMs (class 3), if $R_S + L \leq 24$. The benchmarks belong to class of very big FSMs (class 4), if $R_S + L > 24$. As research [24] shows, the greater the result of dividing $R_S + L$ by NI_{LUT}, the bigger the gain from using methods of structural decomposition compared to FD-based FSM circuits.

The benchmarks are divided by classes in the following way. The class 0 includes the benchmarks *bbtas*, *dk*17, *dk*27, *dk*512, *ex*3, *ex*5, *lion*, *lion*9, *mc*, *modulo*12, and *shiftreg*. The class1 contains the benchmarks *bbara*, *bbsse*, *beecount*, *cse*, *dk*14, *dk*15, *dk*16, *donfile*, *ex*2, *ex*4, *ex*6, *ex*7, *keyb*, *mark*1, *opus*, *s*27, *s*386, *s*840, and *sse*. The class 2 consists of the benchmarks *ex*1, *kirkman*, *planet*, *planet*1, *pma*, *s*1, *s*1488, *s*1494, *s*1*a*, *s*208, *styr*, and *tma*. The class 3 includes the benchmark *sand*. At last, the benchmarks *s*420, *s*510, *s*820, and *s*832 create the class 4.

The results of experiments are shown in Table 10–Table 13. These tables are organized in the same manner. The table columns are marked by the names of investigated methods. The table rows are marked by the names of benchmarks. In the row "Total", there are shown results of summation

TABLE 10. Experimental results (LUT count).

Benchmark	P-Auto	P-One-Hot	P-JEDI	PYM FSM	Our appr.	Class
bbara	17	17	10	12	12	1
bbsse	33	37	24	26	26	1
bbtas	5	5	5	7	7	0
beecount	19	19	14	16	16	1
cse	40	66	36	38	38	1
dk14	16	27	10	12	12	1
dk15	15	16	12	6	9	1
dk16	15	34	12	14	14	1
dk17	5	12	5	7	7	0
dk27	3	5	4	6	6	0
dk512	10	10	9	11	11	0
donfile	31	31	24	25	25	1
ex1	70	74	53	42	40	2
ex2	9	9	8	10	10	1
ex3	9	9	9	10	10	0
ex4	15	13	12	14	14	1
ex5	9	9	9	10	10	0
ex6	24	36	22	24	24	1
ex7	4	5	4	6	6	1
keyb	43	61	40	44	44	1
kirkman	42	58	39	35	30	2
lion	2	5	2	4	4	0
lion9	6	11	5	7	7	0
mark1	23	23	20	22	22	1
mc	4	7	4	6	6	0
modulo12	7	7	7	9	9	0
opus	28	28	22	24	24	1
planet	131	131	88	74	70	2
planet1	131	131	88	74	70	2
pma	94	94	86	72	68	2
s1	65	99	61	58	52	2
s1488	124	131	108	90	76	2
s1494	126	132	110	88	72	2
s1a	49	81	43	39	34	2
s208	12	31	10	10	10	2
s27	6	18	6	8	10	1
s386	26	39	22	24	24	1
s420	10	31	9	8	8	4
s510	48	48	32	31	26	4
s840	9	9	9	10	11	1
s820	88	82	68	58	52	4
s832	80	79	62	54	48	4
sand	132	132	114	101	92	3
shiftreg	2	6	2	4	4	0
sse	33	37	30	26	26	1
styr	93	120	81	73	68	2
tma	45	39	39	33	28	2
Total	1808	2104	1489	1382	1292	
%	139.94	162.85	115.25	106.97	100.00	

of values from columns. The row "Percentage" includes the percentage of summarized characteristics of FSM circuits produced by other methods respectively to $P_{FC}Y_M$-based FSMs. To point that the model of P FSM is used for methods Auto, One-hot, and JEDI, we name these methods as P-Auto, P-One-hot, and P-JEDI. Let us analyse the experimental results taken from reports produced by Vivado.

If we take the total number from the row "Total" of Table 10, then the following conclusion can be made: the P_FCY_M-based FSMs require the minimum amount of LUTs

TABLE 11. Experimental results for classes 2–4 (LUT count).

Benchmark	Auto	One-Hot	JEDI	PYM	Our approach	Class
ex1	70	74	53	42	40	2
kirkman	42	58	39	35	30	2
planet	131	131	88	74	70	2
planet1	131	131	88	74	70	2
pma	94	94	86	72	68	2
s1	65	99	61	58	52	2
s1488	124	131	108	90	76	2
s1494	126	132	110	88	72	2
s1a	49	81	43	39	34	2
s208	12	31	10	10	10	2
styr	93	120	81	73	68	2
tma	45	39	39	33	28	2
sand	132	132	114	101	92	3
s420	10	31	9	8	8	4
s510	48	48	32	31	26	4
s820	88	82	68	58	52	4
s832	80	79	62	54	48	4
Total	1340	1493	1091	940	844	
%	158.77	176.90	129.27	111.37	100.00	

compared with other investigated approaches. Our approach consumes fewer LUTs than it is for P-Auto (39.94% of gain), P-One-hot (62.85% of gain), P-JEDI-based FSMs (15.25% of gain), and PY_M-based FSMs (6.97% of gain). However, the gain (or loss) depends on which class the benchmark FSM belongs to.

For classes 0 and 1, the FSM circuits obtained using the JEDI-based state assignment are characterized by better LUT counts compared to their counterparts obtained using other methods studied. The proposed method loses out, since it is based on the encoding of collections of outputs. This means that even where it is not necessary, the block LUTerO should be implemented. But as follows from Table 10, for FSMs of classes 0 and 1, it is preferable to use unitary (one-hot) encoding of outputs. But already for class 2, our approach gives better results compared to the other investigated methods. At the same time, equivalent PY_M and $P_{FC}Y_M$ FSMs have almost the same LUT counts. The benefits of applying $P_{FC}Y_M$ FSMs instead of other models are evident for classes 2–4. To show it, we have created Table 11 with the experimental results for classes 2–4.

As follows from Table 11, the $P_{FC}Y_M$-based FSMs require significantly fewer LUTs than in the general case represented by Table 10. The P_FCY_M-based FSMs consume fewer LUTs than it is for P-Auto (58.77% of gain), P-One-hot (76.9% of gain), P-JEDI-based FSMs (29.27% of gain), and $P_{FC}Y_M$-based FSMs (11.37% of gain). So, to reduce the LUT count, it makes sense to use the encoding of the fields of compatible states starting from the average FSMs (class 2).

As follows from Table 12, the JEDI-based FSMs have the higher values of maximum operating frequency compared with other investigated FSMs. Our analysis shows that the difference in frequency depends largely on the FSM class.

TABLE 12. Experimental results (operating frequency, MHz).

Benchmark	Auto	One-Hot	JEDI	PYM	Our appr.	Class
bbara	193.39	193.39	212.21	180.32	180.32	1
bbsse	157.06	169.12	182.34	148.65	148.65	1
bbtas	204.16	204.16	206.12	190.38	190.38	0
beecount	166.61	166.61	187.32	158.43	158.43	1
cse	146.43	163.64	178.12	138.55	138.55	1
dk14	191.64	172.65	193.85	166.53	166.53	1
dk15	192.53	185.36	194.87	169.14	169.14	1
dk16	169.72	174.79	197.13	161.52	161.52	1
dk17	199.28	167	199.39	159.12	159.12	0
dk27	206.02	201.9	204.18	196.65	196.65	0
dk512	196.27	196.27	199.75	188.17	188.17	0
donfile	184.03	184	203.65	179.63	179.63	1
ex1	150.94	139.76	176.87	162.94	154.53	2
ex2	198.57	198.57	200.14	191.34	191.34	1
ex3	194.86	194.86	195.76	189.12	189.12	0
ex4	180.96	177.71	192.83	177.76	177.76	1
ex5	180.25	180.25	181.16	162.01	162.01	0
ex6	169.57	163.8	176.59	158.65	158.65	1
ex7	200.04	200.84	200.6	192.69	192.69	1
keyb	156.45	143.47	168.43	137.48	137.48	1
kirkman	141.38	154	156.68	134.73	148.32	2
lion	202.43	204	202.35	190.18	190.18	0
lion9	205.3	185.22	206.38	177.13	177.13	0
mark1	162.39	162.39	176.18	159.58	159.58	1
mc	196.66	195.47	196.87	186.12	186.12	0
modulo12	207	207	207.13	198.32	198.32	0
opus	166.2	166.2	178.32	157.84	157.84	1
planet	132.71	132.71	187.14	173.49	181.54	2
planet1	132.71	132.71	187.14	173.49	181.54	2
pma	146.18	146.18	169.83	154.45	159.83	2
s1	146.41	135.85	157.16	152.19	155.47	2
s1488	138.5	131.94	157.18	152.95	154.31	2
s1494	149.39	145.75	164.34	156.22	158.05	2
s1a	153.37	176.4	169.17	162.84	164.53	2
s208	174.34	176.46	178.76	171.37	175.28	2
s27	198.73	191.5	199.13	178.76	178.76	1
s386	168.15	173.46	179.15	162.63	162.63	1
s420	173.88	176.46	177.25	184.72	193.32	4
s510	177.65	177.65	198.32	207.06	211.76	4
s8	180.02	178.95	181.23	168.32	168.32	1
s820	152	153.16	176.58	197.18	205.12	4
s832	145.71	153.23	173.78	190.64	194.32	4
sand	115.97	115.97	126.82	143.18	145.14	3
shiftreg	262.67	263.57	276.26	226.67	226.67	0
sse	157.06	169.12	174.63	151.67	151.67	1
styr	137.61	129.92	145.64	128.65	128.65	2
tma	163.88	147.8 1	64.14	151.22	151.22	2
Total	8127.08	8061.22	8718.87	8000.68	8066.29	
%	100.75	99.94	108.09	99.19	100.00	

For classes 0–1, $P_{FC}Y_M$-based FSMs have the same (rather bad) results as equivalent PY_M-based FSMs. These results can be explained by the presence of an unnecessary block LUTerO in both PY_M- and PY_M-based FSMs. That is, the LUTerO not only consumes the resources of the FPGA chip, but also creases the clock cycle time. So, for FSMs of classes 0 and 1, it is preferable to use JEDI for state assignment. However, for more complex benchmarks, joint using the encoding of fields of compatible states and mixed encoding of

TABLE 13. Experimental results for classes 2–4 (operating frequency, MHz).

Benchmark	Auto	One-Hot	JEDI	PYM	Our appr.	Class
ex1	150.94	139.76	176.87	162.94	154.53	2
kirkman	141.38	154	156.68	134.73	148.32	2
planet	132.71	132.71	187.14	173.49	181.54	2
planet1	132.71	132.71	187.14	173.49	181.54	2
pma	146.18	146.18	169.83	154.45	159.83	2
s1	146.41	135.85	157.16	152.19	155.47	2
s1488	138.5	131.94	157.18	152.95	154.31	2
s1494	149.39	145.75	164.34	156.22	158.05	2
s1a	153.37	176.4	169.17	162.84	164.53	2
s208	174.34	176.46	178.76	171.37	175.28	2
styr	137.61	129.92	145.64	128.65	128.65	2
tma	163.88	147.8	164.14	151.22	151.22	2
sand	115.97	115.97	126.82	143.18	145.14	3
s420	173.88	176.46	177.25	184.72	193.32	4
s510	177.65	177.65	198.32	207.06	211.76	4
s820	152	153.16	176.58	197.18	205.12	4
s832	145.71	153.23	173.78	190.64	194.32	4
Total	2532.63	2525.95	2866.80	2797.32	2862.93	
%	88.46	88.23	100.14	97.71	100.00	

collections of outputs reduces the number of levels in the circuits of $P_{FC}Y_M$-based FSMs compared to this characteristics of equivalent FSMs based on Auto, One-hot, and the mixed encoding of COs. To confirm this statement, we have formed Table 13.

As follows from Table 13, $P_{FC}Y_M$-based FSMs have practically the same operating frequency as equivalent JEDI-based FSMs. The loss relative to the JEDI-based FSMs is 0.14%. For classes 2–4, our approach gives the gain compared to P-Auto-, P-One-hot-, and PY_M-based FSMs. This gain is equal to 11.54%, 11.77%, and 2.29%, respectively.

We have proposed the method based on encoding of classes of compatible states to improve the LUT count compared to FSMs based on the mixed encoding of collections of outputs. Our research shows that starting from average FSMs (class 2) there is the gain in LUTs. The gain in LUT counts is accompanied by a loss in the maximum operating frequency. The JEDI-based FSMs have the best frequency characteristics. However, as the FSM class grows, this difference in frequency decreases regarding this characteristic of the equivalent JEDI-based FSMs.

VII. CONCLUSION

There are up to 7 billion transistors in modern FPGA chips [10]. Due to it, a very complex digital system may be implemented using a single FPGA chip. The complexity of the implemented systems is constantly increases, but the number of LUT inputs remains rather small. As research [11], [16] states, there is no sense in having LUTs with more than 6 inputs. If an FSM circuit is represented by SOPs for which the condition (5) is violated, various methods of functional decomposition should be applied during the LUT-based technology mapping. The functional decomposition leads to the multi-level FSM circuits with complex interconnection systems.

Both LUT counts and maximum operating frequency of FPGA-based FSM circuits may be improved using various methods of structural decomposition [8]. Very often, FSM circuits based on the structural decomposition have much better characteristics compared with their counterparts based on the functional decomposition [1], [17]. Our research [26], [27] shows that LUT-based FSM circuits with the mixed encoding of collections of outputs have better LUT counts than their counterparts based on the functional decomposition. But it is quite possible that there is more than a single level of LUTs in a circuit generating variables encoding of COs. In this case, it is necessary to use the methods of functional decomposition to implement this circuit (with all the negative consequences).

In our current paper, we propose to use the codes of fields of compatible states to avoid using the functional decomposition in FSM design. As a result, we propose the structural diagram and the design method of LUT-based $P_{FC}Y_M$ Mealy FSMs. This approach is similar to a twofold state assignment [24], [25]. But using the codes of fields of compatible states allows eliminating a special block of code transformer inherent in the case of twofold state assignment. As a result, we achieved a decrease in LUT counts (up to 6.97%) accompanied by a small decrease (up to 0.81%) in the maximum operating frequency compared to PY_M FSMs with JEDI state assignment.

The results of experiments show that the gain in LUT count increases as the complexity of an FSM (the total number of FSM inputs and state variables) increases. At the same time, the increase in the FSM complexity leads to a decrease in the loss in the maximum operating frequency.

Based on the research results, we think that the proposed method of encoding of the fields of compatible states has a good potential for use in the FSM design. In further research, we hope to use this method to improve the characteristics of LUT-based FSMs with twofold state assignment [24], [25].

REFERENCES

[1] M. Kubica, A. Opara, and D. Kania, *Technology Mapping for LUT-Based FPGA* (Studies in Systems, Decision and Control), vol. 713. Berlin, Germany: Springer, 2021.

[2] V. Sklyarov, I. Skliarova, A. Barkalov, and L. Titarenko, *Synthesis and Optimization of FPGA-Based Systems* (Lecture Notes in Electrical Engineering), vol. 294. Berlin, Germany: Springer-Verlag, 2014.

[3] S. Baranov, *Logic and System Design of Digital Systems*. Tallinn, Estonia: TUT Press, 2008.

[4] G. D. Micheli, *Synthesis and Optimization of Digital Circuits*. New York, NY, USA: McGraw-Hill, 1994.

[5] P. Minns and I. Elliot, *FSM-Based Digital Design Using Verilog HDL*. Hoboken, NJ, USA: Wiley, 2008.

[6] I. Grout, *Digital Systems Design With FPGAs and CPLDs*. Amsterdam, The Netherlands: Elsevier, 2011. [Online]. Available: https://books.google.pl/books?id=vggmNXdzayYC

[7] R. Czerwinski and D. Kania, *Finite State Machine Logic Synthesis for Complex Programmable Logic Devices* (Lecture notes in Electrical Engineering), vol. 231. Berlin, Germany: Springer-Verlag, 2013.

[8] A. Barkalov, L. Titarenko, K. Mielcarek, and S. Chmielewski, *Synthesis for FPGA-Based Control Units*, (Lecture Notes in Electrical Engineering), vol. 636. Berlin, Germany: Springer-Verlag, 2020.

[9] O. Barkalov, L. Titarenko, and S. Chmielewski, "Mixed encoding of collections of output variables for lut-based mealy fsms," *J. Circuits, Syst., Comput.*, vol, vol., 28, no. no. 8, pp. 1–21, 2018.

[10] S. M. Trimberger, *Field-Programmable Gate Array Technology*. Norwell, MA, USA: Kluwer, 2012.

[11] S. Kilts, *Advanced FPGA Design: Architecture, Implementation, and Optimization*. Hoboken, NJ, USA: Wiley, 2007.

[12] I. Skliarova, V. Sklyarov, and A. Sudnitson, *Design of FPGA-based circuits using Hierarchical Finite State Machines*. Tallinn, Estonia: TUT Press, 2012.

[13] J. Ruiz-Rosero, G. Ramirez-Gonzalez, and R. Khanna, "Field programmable gate array applications—A scientometric review," *Computation*, vol. 7, no. 4, p. 63, Nov. 2019.

[14] *X. FPGAs*. Accessed: Feb. 15, 2021. [Online]. Available: http://www.xilinx.com/prodcts/silicon-devices/fpga.html

[15] Altera. *Cyclone IV Device Handbook*. Accessed: Feb. 15, 2021. [Online]. Available: http://www.altera.com/literature/hb/cyclone-iv/cyclone4-handbook.pdf

[16] I. Kuon, R. Tessier, and J. Rose, "FPGA architecture: Survey and shallenges—Found trends," *Electr. Design Autom.*, vol. 2, no. 2, pp. 135–253, 2008.

[17] C. Scholl, *Functional Decomposition With Application to FPGA Synthesis*. Boston, MA, USA: Academic, 2001.

[18] L. Machado and J. Cortadella, "Support-reducing decomposition for FPGA mapping," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 1, pp. 213–224, Jan. 2020. [Online]. Available: https://www.cs.upc.edu/~jordicf/gavina/BIB/files/iwls2018_FPGA.pdf

[19] A. Mishchenko, S. Chatterjee, and R. K. Brayton, "Improvements to technology mapping for LUT-based FPGAs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 2, pp. 240–253, Feb. 2007. [Online]. Available: https://https://people.eecs.berkeley.edu/~brayton/publications/2006/tcad06_map.pdf

[20] M. Kubica, D. Kania, and J. Kulisz, "A technology mapping of FSMs based on a graph of excitations and outputs," *IEEE Access*, vol. 7, pp. 16123–16131, 2019.

[21] M. Kubica and D. Kania, "Area–oriented technology mapping for LUT–based logic blocks," *Int. J. Appl. Math. Comput. Sci.*, vol. 27, no. 1, pp. 207–222, Mar. 2017.

[22] M. Rawski, T. Łuba, Z. Jachna, and P. Tomaszewicz, *Design of Embedded Control Systems* (The influence of functional decomposition on modern digital design process). Boston, MA, USA: Springer-Verlag, 2005, pp. 193–203.

[23] W. Feng, J. Greene, and A. Mishchenko, "Improving FPGA performance with a S44 LUT structure," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, New York, NY, USA, 2018, pp. 61–66, doi: 10.1145/3174243.3174272.

[24] A. Barkalov, L. Titarenko, and K. Mielcarek, "Hardware reduction for LUT–based mealy FSMs," *Int. J. Appl. Math. Comput. Sci.*, vol. 28, no. 3, pp. 595–607, Sep. 2018.

[25] A. Barkalov, L. Titarenko, and K. Mielcarek, "Improving characteristics of LUT-based mealy FSMs," *Int. J. Appl. Math. Comput. Sci.*, vol. 30, no. 4, pp. 745–759, 2020.

[26] A. Barkalov, L. Titarenko, and S. Chmielewski, "Improving characteristics of LUT-based Moore FSMs," *IEEE Access*, vol. 8, pp. 155306–155318, 2020.

[27] M. Kubica and D. Kania, "Decomposition of multi-output functions oriented to configurability of logic blocks," *Bull. Polish Acad. Sci. Tech. Sci.*, vol. 65, no. 3, pp. 317–331, Jun. 2017.

[28] A. Opara, M. Kubica, and D. Kania, "Strategy of logic synthesis using MTBDD dedicated to FPGA," *Integration*, vol. 62, pp. 142–158, Jun. 2018.

[29] A. Mishchenko, R. Brayton, J.-H. R. Jiang, and S. Jang, "Scalable don't-care-based logic optimization and resynthesis," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 4, no. 4, pp. 1–23, Dec. 2011. [Online]. Available: https://people.eecs.berkeley.edu/~alanmi/publications/2009/fpga09_mfs.pdf

[30] (2021). *ABC System*. Accessed: Jan. 2021. [Online]. Available: https://people.eecs.berkeley.edu/~alanmi/abc/

[31] A. Opara, M. Kubica, and D. Kania, "Methods of improving time efficiency of decomposition dedicated at FPGA structures and using BDD in the process of cyber-physical synthesis," *IEEE Access*, vol. 7, pp. 20619–20631, 2019.

[32] M. Rawski, H. Selvaraj, and T. Łuba, "An application of functional decomposition in ROM-based FSM implementation in FPGA devices," *J. Syst. Archit.*, vol. 51, nos. 6–7, pp. 423–434, 2005.

[33] S. Khatri and K. Gulati, *Advanced Techniques in Logic Synthesis, Optimizations and Applications*. New York, NY, USA: Springer, 2011.

[34] (2021). *Vivado*. Accessed: Feb. 15, 2021. [Online]. Available: https://www.xilinx.com/products/design_tools/vivado.html

[35] (2021). *Quartus Prime*. Accessed: Feb. 15, 2021. [Online]. Available: https://www.intel.com/content/www/us/en/programmable/downloads/software/quartus-ii-we/121.html

[36] A. S. Klimovich and V. V. Solov'ev, "Minimization of mealy finite-state machines by internal states gluing," *J. Comput. Syst. Sci. Int.*, vol. 51, no. 2, pp. 244–255, Apr. 2012, doi: 10.1134/S1064230712010091.

[37] V. Sklyarov, "Synthesis and implementation of RAM-based finite state machines in FPGAs," in *Proceedings of Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing*. Villach, Austria: Springer-Verlag, 2000, pp. 718–728.

[38] E. Sentowich, K. Singh, L. Lavango, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit synthesis," Univ. California, Berkely, CA, USA, Tech. Rep. UCB/ERL M92/41, 1992.

[39] L. Benini and G. De Micheli, "State assignment for low power dissipation," *IEEE J. Solid-State Circuits*, vol. 30, no. 3, pp. 258–268, Mar. 1995.

[40] G. Sutter, E. Todorovich, S. López-Buedo, and E. Boemo, "Low-power FSMs in FPGA: Encoding alternatives," in *Integrated Circuit Design. Power and Timing Modeling, Optimization and Simulation*. Berlin, Germany: Springer-Verlag, 2002, pp. 363–370.

[41] G. De Micheli, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Optimal state assignment for finite state machines," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 4, no. 3, pp. 269–285, Jul. 2006, doi: 10.1109/TCAD.1985.1270123.

[42] V. V. Solov'ev, "Changes in the length of internal state codes with the aim at minimizing the power consumption of finite-state machines," *J. Commun. Technol. Electron.*, vol. 57, no. 6, pp. 642–648, Jun. 2012, doi: 10.1134/S1064226912060113.

[43] A. H. El-Maleh, "A probabilistic pairwise swap search state assignment algorithm for sequential circuit optimization," *Integration*, vol. 56, pp. 32–43, Jan. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167926016300384

[44] M. Wilkes and J. Stringer, "Microprogramming and the design of the control circuits in an electronic digital computer," Ph.D. dissertation, Univ. Math. Lab., Cambridge, U.K., 1953.

[45] Baer and Koyama, "On the minimization of the width of the control memory of microprogrammed processors," *IEEE Trans. Comput.*, vol. C-28, no. 4, pp. 310–316, Apr. 1979.

[46] S. Achasova, *Synthesis Algorithms for Automata With PLAs*. Moscow, Russia: M: Soviet radio, 1987.

[47] K. L. B. McElvain, "International workshop on logic synthesis benchmark suite (LGSynth93)," Mentor Graphics, Wilonville, OR, USA (LGSynth93), Tech. Rep., 1993.

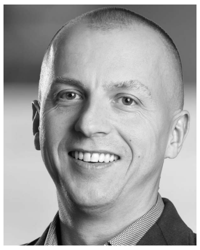
[48] A. Tiwari and K. Tomko, "Saving power by mapping finite-state machines into embedded memory blocks in FPGAs," Xilinx, San Jose, CA, USA, Tech. Rep., 2004, pp. 916–921.

[49] *VC709 Evaluation Board for the Virtex-7 FPGA User Guide*, UG887 (v1.6): Xilinx, San Jose, CA, USA, 2019.

[50] M. M. Islam, M. S. Hossain, M. Shahjalal, M. K. Hasan, and Y. M. Jang, "Area-time efficient hardware implementation of modular multiplication for elliptic curve cryptography," *IEEE Access*, vol. 8, pp. 73898–73906, 2020.

ALEXANDER BARKALOV received the M.Sc. degree in computer engineering from the Donetsk Polytechnical Institute (currently Donetsk National Technical University), Ukraine, in 1976, the Ph.D. degree in computer science from the Leningrad Institute of Fine Mechanics and Optics, Russia, in 1983, and the Doctor of Technical Sciences degree in computer science from the Institute of Cybernetics, Kiev, in 1995. Since 2003, he has been a Professor of computer engineering at the Institute of Informatics and Electronics, University of Zielona Góra, Poland. His current research interests include the theory of digital automata, especially the methods of synthesis and optimization of control units implemented with field-programmable logic devices.

LARYSA TITARENKO received the M.Sc., Ph.D., and Doctor of Technical Sciences degrees in telecommunications from the Kharkov National University of Radioelectronics, Ukraine, in 1993, 1996, and 2005, respectively. Since 2007, she has been a Professor of telecommunications at the Institute of Informatics and Electronics, University of Zielona Góra, Poland. She has taken part in of a number of research projects sponsored by the Ministry of Science and Higher Education of Ukraine (1993–2005). Her current research interests include the theory of telecommunication systems, theory of antennas, and theory of digital automata and its applications.



SŁAWOMIR CHMIELEWSKI received the M.Sc. degree in computer engineering from the Technical University of Zielona Góra, Poland, in 2001, and the Ph.D. degree in computer science from the University of Zielona Góra, Poland, in 2016. His Ph.D. thesis was devoted to synthesis methods targeting CPLD-based FSMs. Since 2017, he has been a Lecturer at the State University of Applied Sciences in Głogów. He is currently a specialist in the field of logistics of Lumel company. His research interests include logic synthesis, design of VLSI-based FSMs, and embedded systems.

KAMIL MIELCAREK received the M.Sc. degree in computer engineering from the Technical University of Zielona Góra, Poland, in 1995, and the Ph.D. degree in computer science from the University of Zielona Góra, Poland, in 2010. Since 2001, he has been a Lecturer at the University of Zielona Góra. His current interests include methods of logic synthesis and optimization of control units in FPGA logic devices, VLSI-based FSMs, hardware description languages, perfect graphs and Petri nets, algorithmic theory and safety of UNIX, and network systems.