Received June 4, 2020, accepted June 29, 2020, date of publication July 2, 2020, date of current version September 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3006732

Improving Characteristics of LUT-Based Moore FSMs

ALEXANDER BARKALOV[1], **LARYSA TITARENKO**[1], **AND SŁAWOMIR CHMIELEWSKI**[2]
[1]Faculty of Computer, Electrical, and Control Engineering, Institute of Metrology, Electronics, and Computer Science, University of Zielona Góra, 65-516 Zielona Góra, Poland
[2]Institute of Science and Technology, State University of Applied Sciences in Głogów, 67-200 Głogów, Poland
Corresponding author: Alexander Barkalov (a.barkalov@imei.uz.zgora.pl)

ABSTRACT Almost any digital system includes sequential blocks which can be represented using a model of finite state machine (FSM). It is very important to improve such characteristics of FSM circuits as the number of logic elements, operating frequency and consumed energy. The paper proposes a novel design method targeting a decrease in the number of look-up table (LUT) elements in logic circuits of FPGA-based Moore FSMs. The method is based on using two sets of variables for encoding the collections of outputs. It results in a partition of the set of outputs by two blocks. The outputs from the first block depend on state variables, the outputs from the second block on additional variables. A method is proposed for splitting the set of outputs. The conditions for using the proposed method are given. An example of synthesis is shown. The results of experiments with standard benchmarks are discussed. The experiments outcomes show that the proposed approach allows diminishing the number of LUTs and consumed energy. Also, it leads to an increase in the operating frequency. The method targets rather complex FSMs when the number of state variables exceeds the number of LUT's inputs.

INDEX TERMS FPGA, Moore FSM, LUT, internal states, FSM outputs.

I. INTRODUCTION

The model of finite state machine (FSM) is used very often for specification and design of sequential blocks of digital systems [1], [2]. It is used, for example, for implementing: 1) the hardware-software interfaces of embedded systems [3]; 2) the complex functions such as hyper-tangent and exponential functions [4], [5]; 3) the activation functions in deep neural networks [6], [7]; 4) some blocks for integral stochastic computing [8]; 5) different stages of cascaded digital processing systems [9]–[11]; 6) the control units of computers and other digital systems [12]–[14]. As follows from [15]–[19], the model of Moore FSM is very often applied in logic design. Due to it, we chose this model in our current research.

Since the mid-twentieth century, many methods have been developed for FSM design [20], [21]. To compare outcomes of these methods, three basic metrics are used. They are: 1) the hardware amount, 2) the performance and 3) the consumed energy [12], [21]. Nowadays, the hardware amount is determined as a chip area occupied by an FSM circuit [22]. The performance is determined by either the propagation time or operating frequency. The operating frequency is inversely proportional to the number of logic levels in an FSM circuit. The consumed energy depends strongly on the hardware amount [15], [19]. It is known [19] that the reducing hardware amount leads to improved performance and energy consumption. Our article targets at reducing the number of look-up table (LUT) elements and their levels in Moore FSM circuits implemented with field programmable gate arrays (FPGA).

The FPGAs are widely used for implementing FSMs [23], [24]. The majority of FPGAs are based on LUTs [25], [26]. A LUT together with a flip-flop forms a logic element (LE). A slice includes up to four LEs. A configurable logic block includes up to four slices [25], [26]. The LE's output could be either combinational or registered (connected with the flip-flop). As a rule, the number of inputs, S_L, of a LUT does not exceed 6 [25], [26]. Very often, it leads to the necessity of functional decomposition for Boolean functions representing FSM circuits [27]–[30]. In turn, it results in an increase in the number of logic levels in a circuit. Also, it makes interconnections more complex. All this has a negative impact on both the operating frequency and power consumption [15], [31].

To improve the basic metrics of LUT-based FSMs, it is necessary to reduce the numbers of arguments in systems of Boolean functions (SBF) representing FSM circuits [32].

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

This can be achieved through the application of methods of structural decomposition [21], [33]. In this case, an FSM circuit is represented as a composition of several large logic blocks. Each block is represented by an SBF with unique systems of arguments and output functions [13], [14], [33], [34]. It leads to an increase in the number of different functions compared to FSM circuits based on the functional decomposition. But these functions are much simpler and their implementation requires less hardware than in the case when the functional decomposition is used [21].

The main contribution of this article is a novel approach for reducing the number of LUTs (and their levels) in the part of the Moore FSM circuit generating outputs. It allows improving all metrics of Moore FSMs.

II. BACKGROUND OF MOORE FSMs

A Moore FSM is defined as a 6-tuple [35], [36] including the following components: $X = \{x_1, \dots, x_L\}$ is a finite set of inputs, $Y = \{y_1, \dots y_N\}$ is a finite set of outputs, $A = \{a_1, \dots, a_M\}$ is a finite set of internal states, the functions of transitions and outputs, and the initial state $a_1 \in A$. There are many methods used for representing FSMs. In this article, we use a state transition table (STT) to represent a Moore FSM [36].

An STT has the following columns: a_m is an initial state; a_s is a state of transition; X_h is an input signal determining the transition $< a_m, a_s >$ and equal to a conjunction of some elements of the set X (or their compliments); h is a number of transition ($h \in \{1, \dots, H\}$). There is a collection of outputs $Y(a_m) \subseteq Y$ written in the column a_m of an STT. It includes outputs $y_n \in Y$ generated in the state $a_m \in A$.

There is an example of STT of Moore FSM S_1 represented by Table 1. The following sets could be derived from Table 1: $X = \{x_1, \dots, x_7\}$, $Y = \{y_1, \dots, y_9\}$, $A = \{a_1, \dots, a_{18}\}$. So, there are $L = 7$, $N = 9$ and $M = 18$. There are $H = 27$ rows in this STT.

When the set A is constructed, it is necessary to represent each state $a_m \in A$ by a binary code $K(a_m)$ having R bits. It is a step of state assignment. The state variables $T_r \in T$ are used for creating codes $K(a_m)$, where $|T| = R$.

If $R = M$, it is a one-hot state assignment. This method is very popular in FPGA-based design [37], [38]. But, for example, such systems as SIS [39] and ABC [22] by Berkeley use a binary state assignment. In this case, there is

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

We also use this approach in our article.

State codes are kept into a state register (RG). It includes R flip-flops with mutual pulses of synchronization (*Clock*) and clearing (*Start*). As a rule, D flip-flops create the RG for LUT-based FSMs [12], [35]. To change the content of RG, input memory functions $D_r \in \Phi$ are used, where $|\Phi| = R$.

A Moore FSM logic circuit is represented by the following SBFs:

$$\Phi = \Phi(T, X); \tag{2}$$

$$Y = Y(T). \tag{3}$$

TABLE 1. STT of Moore FSM S_1.

a_m	a_s	X_h	h
$a_1(-)$	a_2	1	1
$a_2(y_1y_2y_3)$	a_3	x_1	2
	a_4	$\bar{x}_1$	3
$a_3(y_3y_4y_6)$	a_5	1	4
$a_4(y_2y_7)$	a_5	1	5
$a_5(y_4y_8)$	a_6	x_2	6
	a_7	$\bar{x}_2x_3$	7
	a_4	$\bar{x}_2\bar{x}_3$	8
$a_6(y_3y_5y_9)$	a_8	1	9
$a_7(y_1y_4y_6)$	a_8	1	10
$a_8(y_3y_5y_6)$	a_9	x_3	11
	a_{10}	$\bar{x}_3x_4$	12
	a_{11}	$\bar{x}_3\bar{x}_4$	13
$a_9(y_2y_3y_6)$	a_{12}	1	14
$a_{10}(y_1y_4y_8)$	a_{12}	1	15
$a_{11}(y_1y_2y_7)$	a_{12}	1	16
$a_{12}(y_2y_6y_7)$	a_{13}	x_5x_6	17
	a_{14}	$x_5\bar{x}_6$	18
	a_4	$\bar{x}_5$	19
$a_{13}(y_2y_3)$	a_{15}	1	20
$a_{14}(y_1y_4)$	a_{15}	1	21
$a_{15}(y_1)$	a_{16}	x_1	22
	a_{17}	$\bar{x}_1x_7$	23
	a_{15}	$\bar{x}_1\bar{x}_7$	24
$a_{16}(y_3y_5y_6y_9)$	a_{18}	1	25
$a_{17}(y_1y_3y_4)$	a_{18}	1	26
$a_{18}(y_2y_7)$	a_1	1	27

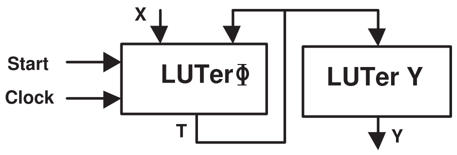


FIGURE 1. Structural diagram of LUT-based Moore FSM U_1.

To find systems (2) – (3), an STT should be transformed into a structure table (ST) [14]. An ST is an expansion of the STT by the following columns: $K(a_m)$ is a code of the current state $a_m \in A$; $K(a_s)$ is a code of the next state (state of transition) $a_s \in A$; Φ_h is a collection of input memory functions equal to 1 to load $K(a_s)$ into RG.

The systems (2) - (3) determine a logic circuit of Moore FSM U_1 (Fig. 1). In Fig. 1, the symbol LUTer determines a block whose circuit is implemented with LUTs [21].

In FSM U_1, the LUTerΦ implements the system (2), the LUTer$Y$ the system (3). If a function D_r is generated as the output of some LUT, then this output is connected with a flip-flop. These flip-flops form a state register RG distributed among the logic elements. It explains the presence of pulses *Clock* and *Start* as inputs of LUTerΦ.

To improve the characteristics of Moore FSM's circuit, it is very important to reduce the chip area occupied by the circuit [2], [35]. The methods of solving this problem depend strongly on logic elements used for implementing FSM circuits [12], [32]. Let us analyze design methods targeting FPGA-based FSMs.

III. STATE-OF-THE-ART

The process of FSM design always has been associated with necessity of the solution of some optimization problems [2]. As a rule, when designing FPGA-based FSMs, four basic optimization problems arise [13], [32]. They are: 1) the decrease in the chip area occupied by an FSM circuit (the hardware reduction); 2) the reduction in the signal propagation time (the increase in the clock frequency); 3) the reduction in power consumption and 4) the improvement of testability. In this article, we consider first of these problems. The analysis of library [40] shows that for some benchmark FSMs there is $L+R \geq 20$. At the same time, for modern LUTs there is $S_L \leq 6$ [25], [26]. Thus, the following condition very often takes place:

$$L + R \gg S_L. \quad (4)$$

If condition (4) is satisfied for some FSM, then the problem of hardware reduction arises for the block LUTerΦ. If the following condition takes place

$$R > S_L, \quad (5)$$

then the LUTer$Y$ could be represented by a multi-level circuit. So, it is necessary to reduce the number of levels in the Moore FSM circuit if conditions (4) – (5) take places.

There are four main approaches for solving this problem, namely:

1) The optimal state assignment [1], [2], [28].
2) The functional decomposition of Boolean functions representing an FSM circuit [22], [27], [29], [30], [41].
3) The replacement of LUTs by embedded memory blocks (EMB) [1], [13], [15]–[17], [19], [32], [42].
4) The structural decomposition of an FSM circuit [13], [20], [21], [43].

We shall call the optimal state assignment a process of obtaining state codes allowing to reduce the number of arguments in functions (2) – (3). These functions are represented as sum-of-products (SOP). But there is a different nature of functions (2) and (3) in Moore FSMs.

The functions $D_r \in \Phi$ depend on terms $F_h(h \in \{1, \ldots, H\})$, where

$$F_h = A_m X_h (h \in \{1, \ldots, H\}). \quad (6)$$

In (6), the symbol A_m stands for a conjunction of state variables corresponding to the state code $K(a_m)$ from the h-th row of ST.

The functions $y_n \in Y$ depend on terms $A_m(m \in \{1, \ldots, M\})$ determined above.

The number of bits in $K(a_m)$ can be ranged from $\lceil log_2 M \rceil$ to M. If $R = M$, it is a one-hot state assignment [15]. When the one-hot is used, only a single state variable forms a conjunction $A_m(m \in \{1, \ldots, M\})$. It allows decreasing for the number of arguments in terms (6). It leads to circuits with less amount of LUTs and layers of logic than in the case of binary encoding. The results of investigations [15] show that one-hot is ´attractive for large FSMs, but a better implementation of small machines can be obtained using binary state assignment´. The results of investigations [17] show that binary encoding gives better results if $L > 10$.

One of the most popular state assignment algorithms is JEDI, which is distributed with the system SIS [39]. It targets a multi-level logic implementation. It maximizes either the size of common cubes in logic functions (the input dominant algorithm) or the number of common cubes in a logic function (the output dominant algorithm).

Modern industrial packages use different state assignment strategies. For example, there are the following methods used in the design tool XST of Xilinx [44]: the automatic state assignment; one-hot; compact; Gray codes; Johnson codes; speed encoding. The same methods are implemented in the design tool Vivado [45].

It is possible to encode the states $a_m \in A$ in such a manner that it minimizes the number of arguments in functions $y_n \in Y$ [32]. For example, the methods [46] could be used to solve this problem. It is important if the condition (5) takes place.

So, there is a lot of state assignment methods. It is really difficult to say which is the best for a particular FSM.

The functional decomposition is very popular in FSM design [27], [30], [41], [42]. If number of arguments for some function exceeds S_L, then the original function is broken down into smaller and smaller components. There are three approaches in this area: serial, parallel and balanced decomposition. These approaches are used, for example, in systems DEMAIN [47] or PKMIN [48]. Obviously, there are program tools for functional decomposition in any CAD system targeting FPGA-based design. One of the best CAD tools using this approach is the ABC package by Berkeley [22], [49].

Modern FPGA have a lot of embedded memory blocks [25], [26]. Using EMBs allows improvement for main characteristics of FSM circuits [17]. Because of it, there are many design methods targeting EMB-based FSMs [13], [14], [16]–[19], [41], [43]. The EMBs have a property of configurability. It means that such parameters as the number of cells and their outputs could be changed by a designer [24]. Typical configurations of EMBs are the following: $16K * 1$, $8K * 2$, $4K * 4$, $2K * 8$, $1K * 16$, $512 * 32$, $256 * 64$ (bits) [25], [26]. So, modern EMBs are very flexible and can be tuned to meet a particular FSM.

In the best case, an FSM circuit is implemented as a single EMB. It is possible if the following condition takes place [43]:

$$2^{L+R}(R+N) \leqslant V_0. \quad (7)$$

In (7), the symbol V_0 stands for the number of cells for EMB configuration with a single output. Our investigations [43] of library [37] shows that condition (7) is true for 68% of benchmarks.

If (7) is violated, then an FSM circuit could be implemented as: 1) a network of EMBs or 2) a network of LUTs and EMBs. The survey of different approaches for EMB-based design can be found in [30]. Let us point out that these methods could be used only if there are ´free´ EMBs, which are not used for design other parts of a digital system.

In the case of structural decomposition, an FSM circuit is represented by several blocks [21], [32]. Each block implements functions different from (2) – (3).

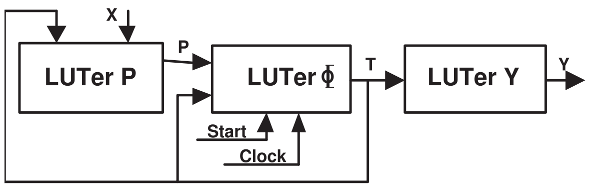


FIGURE 2. Structural diagram of Moore FSM U_2.

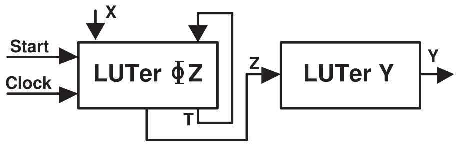


FIGURE 3. Structural diagram of PY Mealy FSM.

For example, in the case of replacement of inputs $x_e \in X$ [14], the set X is replaced by a set $P = \{p_1, \ldots, p_G\}$ such that $G \ll L$. Now functions (2) are represented as

$$\Phi = \Phi(T, P). \quad (8)$$

It leads to Moore FSM U_2 (Fig. 2), where the block LUTerP generates functions

$$P = P(T, X). \quad (9)$$

For Mealy FSMs, the encoding of collections of outputs (CO) could be used [20], [33]. In this case each collection $Y_q \subseteq Y$ is encoded by the binary code $K(Y_q)$. Let it be Q different collections in an FSM. In this case, R_Q bits are necessary for their encoding, where:

$$R_Q = \lceil log_2 Q \rceil. \quad (10)$$

Let us use variables $z_r \in Z$ for the encoding, where $|Z| = R_Q$. It leads to so called PY Mealy FSM (Fig. 3).

In PY Mealy FSM, the LUTerΦZ implements systems (2) and

$$Z = Z(T, X). \quad (11)$$

The LUTerY implements the functions

$$Y = Y(Z). \quad (12)$$

This method leads to a single-level circuit of LUTerY, if the following condition takes place:

$$R_Q \leq S_L. \quad (13)$$

If (13) is violated, then it is possible to use the mixed encoding of outputs [50]. In this case, some functions $y_n \in Y$ are represented as SOPs depending on the terms (6). Let us point out that this approach never has been used for Moore FSMs.

In this article, we discuss a case when the condition (5) takes place for some Moore FSM S. In this case, there is a multi-level circuit of the block LUTerY generating the system of outputs $Y = Y(T)$. To diminish the number of LUTs required for generating outputs $y_n \in Y$, we propose to divide the set Y by two disjoint sets ($Y = Y_L \cup Y_0$). We propose to use the method of encoding of collections of outputs including outputs $y_n \in Y_L$. These outputs depend now on some additional variables from the set Z. The outputs $y_n \in Y_0$ are still implemented as functions (3) depending on state variables $T_r \in T$. The proposed method is an evolution of ideas from our work [50]. In [50], we proposed a method of mixed encoding of outputs for Mealy FSMs. In this article, we have adapted the approach of mixed encoding for Moore FSMs where outputs depend only on states.

TABLE 2. Collections of outputs for FSM S_1.

q	Y_q	q	Y_q	q	Y_q
1	-	7	$y_1y_4y_6$	13	y_2y_3
2	$y_1y_2y_3$	8	$y_3y_5y_6$	14	y_1y_4
3	$y_3y_4y_6$	9	$y_2y_3y_6$	15	y_1
4	y_2y_7	10	$y_1y_4y_8$	16	$y_3y_5y_6y_9$
5	y_4y_8	11	$y_1y_2y_7$	17	$y_1y_3y_4$
6	$y_3y_5y_9$	12	$y_2y_6y_7$	-	-

IV. MAIN IDEA OF PROPOSED METHOD

Let us create a set V of CO_s $Y_q \subseteq Y$ for some Moore FSM. For example, there is $Q = 17$ in the case of Moore FSM S_1. These collections are listed in Table 2.

Let us use LUTs with $S_L = 3$ to implement an FSM circuit. Because $R = \lceil log_2 18 \rceil = 5$, the condition (5) takes place for FSM S_1. So, there is a multi-level circuit of LUTerY for U_1-based circuit of S_1.

Using (10) gives $R = R_Q > S_L = 3$. Let us try to diminish the number of COs to reach the equality

$$R_Q = S_L. \quad (14)$$

To do it, we should eliminate some functions $y_n \in Y$ from initial COs $Y_q \subseteq Y$.

Let the set V include COs Y_i, Y_j such that $Y_i = Y_j \bigcup \{y_n\}$. So, the elimination of y_n from Y_i leads to the equality $Y_i = Y_j$. It results in decrementing the number of COs $Y_q \in V$: $|V| = Q - 1$. Let $I(y_n)$ be a number of pairs $< Y_i, Y_j >$ such that the elimination of y_n from Y_i leads to $Y_i = Y_j$. Therefore, the elimination of y_n from Y_i leads to decrease in the number of COs by $I(y_n)$.

The elimination of y_n results in a transformation of the set V into a set V_1 having $Q_1 = Q - I(y_n)$ elements. Now, it is enough R_1 bits to encode the COs $Y_q \in V_1$:

$$R_1 = \lceil log_2 Q_1 \rceil. \quad (15)$$

Let the following condition take place:

$$R_1 \leq S_L. \quad (16)$$

In this case, the output y_n is implemented as (3), whereas the outputs $y_m \in Y \setminus \{y_n\}$ are represented as (12).

If the condition (16) is violated, then it is necessary to find an output y_m such that its elimination from COs $Y_q \in V_1$ leads to the set V_2 having Q_2 elements. Now, it is enough R_2 variables to encode the remaining COs:

$$R_2 = \lceil log_2 Q_2 \rceil. \quad (17)$$

Let the following condition take place:

$$R_2 < R_1. \quad (18)$$

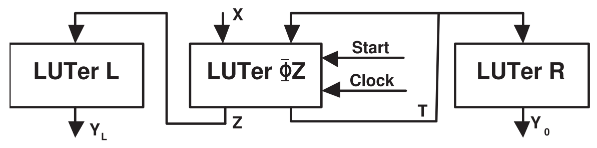


FIGURE 4. Structural diagram of Moore FSM U_3.

It means that the elimination of y_m results in further decreasing for the number of bits in codes of COs.

The process of elimination should be continued till the following condition will be true:

$$R_i = S_L. \tag{19}$$

The subscript $'i'$ shows that $R_i = \lceil log_2 |V_i| \rceil$. There are $N - i$ outputs $y_n \in Y_q$ where $Y_q \in V_i$.

Let Y_0 be a set of eliminated outputs. If $y_n \in Y_0$, then y_n is represented as (3). Let Y_L be a set of outputs creating the COs $Y_q \in V_i$. If $y_n \in Y_L$, then it is represented as (12). It leads to Moore FSM U_3 shown in Fig. 4.

In FSM U_3, the LUTerL implements outputs $y_n \in Y_L$, the LUTerR implements outputs $y_n \in Y_0$. The LUTerΦZ implements SBFs (2) and (11). So, the proposed approach results in the mixed encoding of outputs [50].

Let the outputs $y_n \in Y_L$ create Q_L collections. To encode them, it is necessary R_L variables:

$$R_L = \lceil log_2 Q_L \rceil. \tag{20}$$

Obviously, the following condition should take place:

$$R_L = S_L. \tag{21}$$

In this case, it is enough a single LUT to implement the circuit for each function $y_n \in Y_L$.

Because the condition (5) takes place, there is a multi-level circuit of LUTerR. To diminish the number of arguments in SOPs of functions $y_n \in Y_0$, it is possible to use the following approach for state assignment. Let us encode the states $a_m \in A$ in such a way that functions $y_n \in Y_0$ are represented by the minimum possible number of intervals of R-dimensional Boolean space. Let us name such approach the special state assignment (SSA). To execute the SSA, the methods from [46] could be used.

In this article, we propose the method of synthesis of Moore FSM U_3. The method includes the following steps:

1) Deriving the set V from the initial STT.
2) Dividing the set Y by sets Y_0 and Y_L.
3) Executing the special state assignment.
4) Encoding the COs $Y_q \subseteq Y_L$.
5) Creating the ST of FSM U_3.
6) Creating the systems $\Phi = \Phi(T, X)$ and $Z = Z(T, X)$.
7) Creating the systems $Y_L = Y_L(Z)$ and $Y_0 = Y_0(T)$.
8) Implementing the FSM logic circuit.

V. DIVIDING THE SET OF OUTPUTS

Table 3 depicts a pseudo-code of the proposed algorithm for dividing the set of collections of outputs. The algorithm is technology-dependent because it takes into account the number of LUT's inputs S_L. It uses as inputs the set V and the value of S_L. The algorithm requires not more than N cycles. The algorithm generates sets Y_0 and Y_L as its output. In the beginning, the set $Y_L = Y$ and the set V_L includes all COs $Y_q \subseteq Y$ (line 2).

TABLE 3. Pseudo-code of the algorithm of deviding the set of outputs.

(1)	function DividingSetOutputs (V, S_L); input: A set of COs (V), the number of inputs (S_L); output: Sets Y_0 and Y_L
(2)	/* Initialization */
(3)	$k := 1$ # cycle for outputs
(4)	/* Creating the queue γ having $I = \|Y_L\|$ elements /*
(5)	$i := 1$ # cycle for analysis of γ
(6)	/* To choose the i-th element of γ and finding ΔQ_i */
(7)	$R_k := \lceil log_2(Q_L - \Delta Q_i) \rceil$
(8)	if $R_k < R_{k-1}$ then
(9)	if $R_k = S_L$ then go to 17
(10)	else /* Modification of Y_L and Y_0 */
(11)	$k := k + 1$
(12)	if $k > N$ then go to 17
(13)	else /* Correction of V_L */; (14) go to 4
(15)	else $i := i + 1$
(16)	if $i \leqslant I$ go to 6; (17) end # end of algorithm

The main idea of the method is reduced to finding the outputs $y_n \in Y_L$ such that their excluding from Y_L leads to the maximum possible reduction in the number of COs $Y_q \subseteq Y_L \setminus \{y_n\}$. The search is organized as a cycle with a cycle variable k. It starts from the operator 3. Each cycle starts from the organizing the queue γ (line 4). The queue includes outputs $y_n \in Y_L$.

For each output y_n of the queue γ, the value of $Q(y_n)$ is calculated. It is equal to the number of COs $Y_q \in V_L$. Next, the elements of γ are ranked in the descending order of the value of $Q(y_n)$. Each cycle $k (k \in \{1, \ldots, N\})$ can have up to I steps where $I = |V_L|$.

During each step (starting from line 5), a single element of γ is analyzed (line 6). The value of ΔQ_i is calculated for a chosen output $y_n \in Y_L$. The ΔQ_i is equal to the number of COs excluded from V_L due to transferring y_n from Y_L into Y_0. Next, the value of R_k is calculated (line 7) as

$$R_k = \lceil log_2(Q_L - \Delta Q_i) \rceil. \tag{22}$$

If the excluding the i-th element of γ leads to reduced value of R_Q (line 8), then it is necessary to check the condition (14). It is executed as the operator 9. If condition (14) is true, then the solution is found. The algorithm is finished (go to 17). Otherwise, the modification is executed for sets Y_L and Y_0 (line 10). During this step, the output y_i is excluded from Y_L and included into Y_0 ($Y_L := Y_L \setminus \{y_i\}$ and $Y_0 := Y_0 \cup \{y_i\}$). The value of k is incremented (line 11). If $k > N$, then all outputs are analyzed and the algorithm is finished (line 12). Otherwise, the correction of V_L is executed (line 13). After obtaining new COs, the next queue γ is creating (line 14).

If condition (14) is violated (line 15), then the next element of γ should be analyzed ($i := i + 1$). If the queue is not exhausted, then its next element should be analyzed (go to 6 in line 16). Otherwise, the algorithm is terminated.

TABLE 4. The process of partition of the set Y.

y_n	$Q(y_n)$	ΔQ_i	R_1	$Q(y_n)$	ΔQ_i	R_2
y_1	7(i=2)	4	4 +	-	-	
y_2	6			4		
y_3	8 (i=1)	1	5 -	7 (i=1)	2	4 -
y_4	6			5		
y_5	3			3		
y_6	6			6 (i=2)	5	3 +
y_7	3			2		
y_8	2			1		
y_9	2			2		
Y_0	y_1			y_6		

TABLE 5. The COs after step 1.

q	Y_q	q	Y_q	q	Y_q
1	-	6	$y_3y_5y_9$	11	y_4
2	y_2y_3	7	$y_4y_6y_6$	12	$y_3y_5y_6y_9$
3	$y_3y_4y_6$	8	$y_3y_5y_6$	13	y_3y_4
4	y_2y_7	9	$y_2y_3y_6$	-	-
5	y_4y_8	10	$y_2y_6y_7$	-	-

Let us apply this algorithm to Moore FSM S_1. As follows from Table 2, there is $Q = 17$. Using (10) gives $R_Q = 5$. Let us use LUTs having $S_L = 3$ inputs. So, the condition (13) is violated and it is necessary to divide the set Y. The process of dividing is shown in Table 4.

There are outputs y_n, values of $Q(y_n)$ and ΔQ_i shown in the corresponding columns of Table 4. If some output $y_n \in Y$ is taken for analysis during the cycle i, then it is shown in the brackets in the column $Q(y_n)$. The sign '+' shows that the corresponding output is included into Y_0. The sign '-' means that the output y_n is excluded from the analysis. The row Y_0 shows the outcome of dividing.

Analysis of $Q(y_n)$ allows creating the queue $\gamma =< y_3, y_1, y_2, y_4, y_6, y_5, y_7, y_8, y_9 >$. Including y_3 into Y_0 (cycle 1) gives $\Delta Q_1 = 1$ and $R_1 = 5$. Because $R_1 = R_Q$, the analysis should be continued for the next element of the γ. The analysis of Y_q (cycle 2) shows that $R_1 = 4$. Because there is $R_1 < R_Q$, the output y_1 is included into the set Y_0. Because the condition (13) is violated, the next step should be executed. To do it, the new table of COs should be constructed (Table 5).

Using Table 5, the following queue is formed: $\gamma =< y_3, y_6, y_4, y_2, y_5, y_7, y_9, y_8 >$. Analysis of y_3 gives $R_2 = R_1 = 4$ (cycle 1). Analysis of y_6 gives $R_2 = 3 < R_1$. So, the output y_6 is included into Y_0. Because $R_2 = S_L$, the process of dividing is terminated.

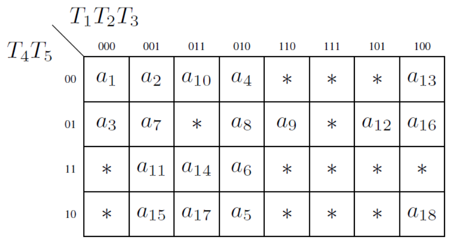
We have found the sets $Y_0 = \{y_1, y_6\}$, $Y_L = \{y_2, y_3, y_4, y_5, y_7, y_8, y_9\}$. Also, there is $R_L = 3$ and $Z = \{z_1, z_2, z_3\}$. Table 6 presents the new COs $Y_q \subseteq Y_L$.

VI. EXAMPLE OF SYNTHESIS

Let us discuss an example of synthesis for Moore FSM S_1. The steps 1 and 2 are already executed for this example. There is $M = 18$. Using (1) gives $R = 5$ and $T = \{T_1, \ldots, T_5\}$.

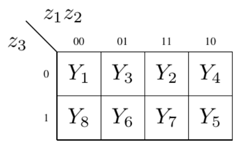
TABLE 6. The COs after dividing the set Y.

q	Y_q	q	Y_q	q	Y_q	q	Y_q
1	-	3	$y_3y_4y_6$	5	y_4y_8	7	y_3y_5
2	y_2y_3	4	y_2y_7	6	$y_3y_5y_9$	8	y_4



T_4T_5 \ $T_1T_2T_3$	000	001	011	010	110	111	101	100
00	a_1	a_2	a_{10}	a_4	$*$	$*$	$*$	a_{13}
01	a_3	a_7	$*$	a_8	a_9	$*$	a_{12}	a_{16}
11	$*$	a_{11}	a_{14}	a_6	$*$	$*$	$*$	$*$
10	$*$	a_{15}	a_{17}	a_5	$*$	$*$	$*$	a_{18}

FIGURE 5. The outcome of special state assignment.



z_3 \ z_1z_2	00	01	11	10
0	Y_1	Y_3	Y_2	Y_4
1	Y_8	Y_6	Y_7	Y_5

FIGURE 6. The outcome of encoding of COs.

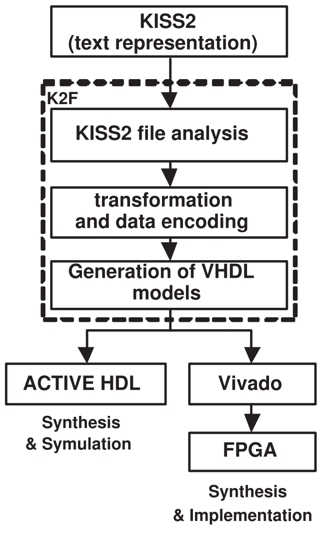


FIGURE 7. Design path based on K2F tool.

Let us execute the special state assignment for FSM S_1. Using the method from [46] gives the state codes shown in Fig. 5.

Let us encode the COs $Y_q \subseteq Y_L$ using the method [46]. It targets diminishing the number of arguments in the functions (12). The outcome is shown in Fig. 7.

The system (11) is generated by the LUTerΦZ. So, the ST of Moore FSM U_3 should include the column Z_h. This column contains variables $z_r \in Z$ equal to 1 in the code of CO Y_q for a state a_s from the row h ($h \in \{1, \ldots, H\}$). Table 7 is an ST for Moore FSM S_1. It includes state codes from Fig. 5, the function $D_r \in \Phi = \{D_1, \ldots, D_5\}$ and variables $z_r \in Z$. Now, the column a_m includes only outputs $y_n \in Y_0$. The column q includes the subscripts of COs Y_q from Table 6.

TABLE 7. Structure table of U_3-based FSM S_1.

a_m	$K(a_m)$	a_s	$K(a_s)$	X_h	q	Z_h	Φ_h	h
$a_1(-)$	00000	a_2	00100	1	2	z_1z_2	D_3	1
$a_2(y_1)$	00100	a_3	00001	x_1	3	z_2	D_5	2
		a_4	01000	$\bar{x_1}x_4$	4	z_1	D_2	3
$a_3(y_6)$	00001	a_5	01010	1	5	z_1z_3	D_2D_4	4
$a_4(-)$	01000	a_5	01010	1	5	z_1z_3	D_2D_4	5
$a_5(-)$	01010	a_6	01011	x_2	6	z_2z_3	$D_2D_4D_5$	6
		a_7	00101	$\bar{x_2}x_3$	8	z_3	D_3D_5	7
		a_4	01000	$\bar{x_2}\bar{x_3}$	4	z_1	D_2	8
$a_6(-)$	01011	a_8	01001	1	7	$z_1z_2z_3$	D_2D_5	9
$a_7(y_1y_6)$	00101	a_8	01001	1	7	$z_1z_2z_3$	D_2D_5	10
$a_8(y_6)$	01001	a_9	11001	x_3	2	z_1z_2	$D_1D_2D_5$	11
		a_{10}	01100	$\bar{x_3}x_4$	5	z_1z_3	D_2D_3	12
		a_{11}	00111	$\bar{x_3}\bar{x_4}$	4	z_1	$D_3D_4D_5$	13
$a_9(y_6)$	11001	a_{12}	10101	1	4	z_1	$D_1D_3D_5$	14
$a_{10}(y_1)$	01100	a_{12}	10101	1	4	z_1	$D_1D_3D_5$	15
$a_{11}(y_1)$	00111	a_{12}	10101	1	4	z_1	$D_1D_3D_5$	16
$a_{12}(y_6)$	10101	a_{13}	10000	x_5x_6	2	z_1z_2	D_1	17
		a_{14}	01111	$x_5\bar{x_6}$	8	z_3	$D_2D_3D_4D_5$	18
		a_4	01000	$\bar{x_5}$	4	z_1	D_2	19
$a_{13}(-)$	10000	a_{15}	00110	1	1	$-$	D_3D_4	20
$a_{14}(y_1)$	01111	a_{15}	00110	1	1	$-$	D_3D_4	21
$a_{15}(y_1)$	00110	a_{16}	10001	x_1	6	z_2z_3	D_1D_5	22
		a_{17}	01110	$\bar{x_1}x_7$	3	z_2	$D_2D_3D_4$	23
		a_{15}	00110	$\bar{x_1}\bar{x_7}$	1	$-$	D_3D_4	24
$a_{16}(y_6)$	10001	a_{18}	10010	1	4	z_1	D_1D_4	25
$a_{17}(y_1)$	01110	a_{18}	10010	1	4	z_1	D_1D_4	26
$a_{18}(-)$	01110	a_1	00000	1	1	$-$	$-$	27

Let us explain, for example, the row 2 of Table 7. There is $a_m = a_2$. As follows from Table 1, $Y(a_2) = \{y_1, y_2, y_3\}$. So, there is y_1 in the column a_m. There is $a_s = a_3$ with $Y_3 = \{y_3, y_4, y_6\}$ (Table 1). After elimination $y_6 \in Y_0$, the CO $Y_3 = \{y_3, y_4\}$ (Table 6) should be generated in the state a_3. As follows from Fig. 7, there is $K(Y_3) = 010$. So, there is the symbol z_2 in the column Z_h. The state codes are taken from Fig. 5.

The functions (2) and (11) are derived from Table 7. They depend on the terms (6). It is done in the trivial way [14].

As follows from Table 7, the outputs $y_n \in Y_0$ are represented as

$$\begin{aligned} y_1 &= A_2 \vee A_7 \vee A_{10} \vee A_{11} \vee A_{14} \vee A_{15} \vee A_{17} \\ y_6 &= A_3 \vee A_7 \vee A_8 \vee A_9 \vee A_{12} \vee A_{16}. \end{aligned} \quad (23)$$

Using state codes (Fig. 5) transforms the system (23) into the following system

$$y_1 = \bar{T}_1T_3; \quad y_6 = T_5. \quad (24)$$

Using Table 6 and codes from Fig. 7, we can get the following system $y_n \in Y_L$:

$$\begin{aligned} &y_2 = Y_2 \vee Y_4 = z_1\bar{z}_3;\ y_3 = Y_2 \vee Y_3 \vee Y_6 \vee Y_7 = z_2; \\ &y_4 = Y_3 \vee Y_5 \vee Y_8 = \bar{z}_1z_2\bar{z}_3 \vee \bar{z}_2z_3; \\ &y_5 = Y_6 \vee Y_7 = z_2z_3; \quad y_7 = Y_4 = z_1\bar{z}_2z_3; \\ &y_8 = Y_5 = z_1\bar{z}_2z_3; \quad y_9 = Y_6 = \bar{z}_1z_2z_3. \end{aligned} \quad (25)$$

As follows from (25), there is only a single LUT in the circuit of LUTerR. As follows from (25) there are 6 LUTs in the circuit of LUTerL.

We do not discuss the last step of synthesis for this example. This step is connected with using some standard tools [25], [26] and a VHDL model of U_3.

VII. EXPERIMENTAL RESULTS

To investigate the efficiency of proposed method, we use standard benchmarks from the LGSynth93 library [40]. The library includes 48 Mealy FSMs taken from the practice of FSM design. They are presented in the KISS2 format. We transform these Mealy FSMs into equivalent Moore FSMs using three rules taken from [14]. Rule 1: if K different collections of outputs are generated during transitions in a state $a_m \in A$ of Mealy FSM, then this state is replaced by K states $(a_m^1, \ldots, a_m^K)$ of equivalent Moore FSM. Rule 2: each of states $a_m^k (k \in \{1, \ldots, K\})$ has a unique collection of outputs. Rule 3: all states $a_m^k \in A$ have the same function of transition. The characteristics of obtained Moore FSMs are shown in Table 8.

To use these benchmarks, we use the CAD tool named K2F [43]. It translates the KISS2 file into a VHDL model of an FSM. To synthesize and simulate the FSM, we use the Active-HDL environment. To get the FSM circuit, we use Xilinx Vivado 2019.1 tool [45]. The investigation path used

TABLE 8. Characteristics of benchmarks.

Benchmark	L	N	R	M	U_3
bbara	4	2	4	12	–
bbsse	7	7	5	26	–
bbtas	2	2	4	9	–
beecount	3	4	4	10	–
cse	7	7	5	32	–
dk14	3	5	5	26	–
dk15	3	5	5	17	–
dk16	2	3	7	75	+
dk17	2	3	4	16	–
dk27	1	2	4	10	–
dk512	1	3	5	24	–
donfile	2	1	5	24	–
ex1	9	19	7	80	+
ex2	2	2	5	25	–
ex3	2	2	4	14	–
ex4	6	9	5	18	–
ex5	2	2	4	16	–
ex6	5	8	4	14	–
ex7	2	2	5	17	–
keyb	7	2	5	22	–
kirkman	12	18	8	138	+
lion	2	1	3	5	–
lion9	2	1	4	11	–
mark1	5	16	5	22	–
mc	3	5	3	8	–
modulo12	1	1	4	12	–
opus	5	6	4	10	–
planet	7	39	8	48	–
planet1	7	39	8	48	–
pma	8	8	6	48	–
s1	8	16	7	80	+
s1488	8	36	8	168	+
s1494	8	36	8	168	+
s1a	8	6	7	86	+
s208	11	2	6	37	–
s27	4	1	3	6	–
s298	3	6	9	332	+
s386	7	7	5	23	–
s420	19	18	8	137	+
s510	19	17	8	173	+
s8	4	1	3	5	–
820	18	39	7	70	+
s832	18	39	7	70	+
sand	11	9	7	88	+
scf	27	56	8	141	+
shifreg	1	1	4	16	–
sse	7	7	5	26	–
styr	9	10	7	67	+
tav	4	4	5	27	–
tbk	6	3	7	90	+
tma	7	6	5	20	–
train11	2	1	4	14	–
train4	2	1	3	6	–

in our system is shown in Fig. 8. The Xilinx Vivado 2019.1 package was used for synthesis and implementation of FSM for a given benchmark.

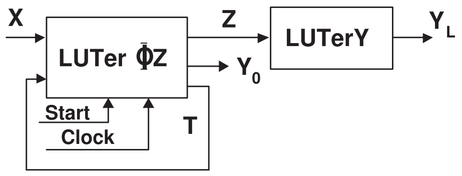


FIGURE 8. Structural diagram of PY_M Mealy FSMs.

The target platform was the FPGA device Xilinx Virtex-7 (XC7VX690tffg1761-2). It includes LUTs with $S_L = 6$ [44].

The column U_3 of Table 8 shows the feasibility of using the proposed method for a particular benchmarks. If condition (5) is true, then our method can be used. We marked these benchmarks with a plus in the corresponding row of Table 8.

There 16 signs ´+´ in Table 8. So, only corresponding 16 benchmarks are used in our research.

We compared our approach with four other methods. They are: 1) Auto of Vivado; 2) one-hot of Vivado; 3) JEDI-based U_1 and 4) DEMAIN-based U_1. The results of experiments are shown in Table 9 (for the number of LUTs), Table 10 (for the operating frequency, MHz), and Table 11 (for the consumed energy, Watts).

These tables are organized in the same manner. Their rows are marked with the names of benchmarks, the columns by design methods. The rows ´Total´ include results of summations for numbers from each column. We took as 100% results of addition for the method U_3. The rows ´Percentage´ show the percentage of summarized characteristics respectively to the results obtained for U_3.

As follows from Table 9, the proposed method allows diminishing the number of LUTs compared to other researched methods. There is the following gain: 1) 35% regarding to Auto; 2) 58% regarding to one-hot; 3) 13% regarding to JEDI-based FSMs and 4) 24% regarding to DEMAIN.

In all cases studied, our approach produces FSM circuits having exactly a single level of LUTs for blocks generating output functions $y_n \in Y$. Due to it, U_3-based FSMs have better results for operating frequency than it is for other methods used in the research. As follows from Table 10, our approach gives the following gain in the operating frequency: 1) 32,5% compared to Auto; 2) 32,5% compared to one-hot; 3) 10,2% compared to JEDI and 4) 28% compared to DEMAIN.

Reducing the numbers of LUTs and their levels allowed obtaining FSM circuits with lower energy consumption than for other methods. As follows from Table 11, U_3-based FSMs have the following gain in consumed energy: 1) 30,4% in comparison with Auto; 2) 40,8% in comparison with one-hot; 3) 12,1% in comparison with JEDI and 4) 22,9% in comparison with DEMAIN.

So, if $R > 6$, then our approach gives better results than they are for Auto, one-hot, JEDI and DEMAIN. Of course, it is true only for benchmarks [40] and the device XC7VX690tffg1761-2. Let us point out that we conducted similar research using device Xilinx Virtex-5 (XC5VLX30FF324) having LUTs with $S_L = 6$. To do it,

TABLE 9. Experimental results (the number of LUTs).

$Benchmark$	$Auto$	$One-Hot$	$JEDI$	$DEMAIN$	U_3
dk16	18	40	14	17	12
ex1	84	88	63	68	51
kirkman	50	69	46	49	42
s1	78	118	73	76	68
s1488	148	157	128	134	111
s1494	150	158	130	140	115
s1a	58	97	51	64	49
s298	27	49	22	24	20
s420	12	37	11	12	10
s510	57	57	38	46	34
s820	105	98	81	91	69
s832	96	94	74	84	65
sand	158	160	136	145	122
scf	74	91	63	69	54
styr	111	144	97	105	85
tbk	54	46	46	49	39
Total	1280	1503	1073	1173	946
Percentage	135%	158%	113%	124%	100%

TABLE 10. Experimental results (the operating frequency, MHz).

$Benchmark$	$Auto$	$One-Hot$	$JEDI$	$DEMAIN$	U_3
dk16	141,43	145,65	197,13	158,28	211,52
ex1	125,78	116,46	176,87	155,11	212,93
kirkman	117,81	128,33	156,68	119,82	174,73
s1	122,01	113,21	157,16	124,31	170,19
s1488	115,41	109,95	157,18	127,65	187,95
s1494	124,49	121,45	164,34	132,85	186,22
s1a	127,80	147,86	169,17	131,77	178,84
s298	122,79	121,21	168,76	135,73	186,37
s420	144,92	147,05	177,25	144,05	181,62
s510	148,04	148,04	198,32	152,65	209,36
s820	122,66	127,63	176,58	138,75	192,14
s832	121,42	127,69	173,78	133,36	192,87
sand	98,65	96,64	126,82	100,53	163,18
scf	135,57	136,31	177,26	146,78	184,69
styr	114,68	108,26	145,64	115,69	178,65
tbk	136,47	123,16	164,14	140,16	181,22
Total	2019,93	2019,2	2687,08	2157,49	2992,14
Percentage	67,5%	67,5%	89,8%	72%	100%

we used the Xilinx ISE 14.1 package [44] The results of these investigations confirmed our hypothesis about the feasibility of using the model U_3 when the condition (5) is met.

Each of quantities from Table 9 – Table 11 evaluates only one of the characteristics of FSM circuits. In this article, we propose a comprehensive assessment that takes into account all three characteristics (the number of LUTs, operating frequency and consumed energy). We propose to evaluate an FSM circuit by the following value:

$$\beta = \frac{f}{N_L * P} * 10^{10}. \tag{26}$$

The value of β is the inverse of the amount of energy (mJ) consumed per LUT of an FSM circuit.

We show these values for generalized characteristics of FSMs (Table 9 – Table 11) in Table 12. We use total characteristics to get the average value of β (row β of Table 12). We took as 100% the values of β for U_3. As follows from the row 'Percentage', our approach allows to improve this general characteristic in comparison with other researched methods.

VIII. SOME IMPORTANT ADDITIONAL ISSUES

A sequential block can be represented as either Mealy or Moore FSM. For both Mealy and Moore FSMs, the input

TABLE 11. Experimental results (the consumed energy, Watts).

$Benchmark$	$Auto$	$One-Hot$	$JEDI$	$DEMAIN$	U_3
dk16	3,560	3,290	3,014	3,258	2,918
ex1	4,922	3,562	2,811	2,899	2,476
kirkman	2,031	2,213	1,727	1,798	1,527
s1	3,222	3,756	3,021	3,134	2,848
s1488	4,778	4,915	4,257	4,355	3,683
s1494	3,694	3,813	3,578	3,613	3,058
s1a	1,586	2,412	1,449	1,922	1,785
s298	1,730	2,398	1,453	2,041	1,302
s420	3,005	3,544	2,574	3,152	2,248
s510	1,604	3,384	1,543	1,601	1,292
s820	1,883	1,996	1,878	1,881	1,682
s832	2,465	2,161	1,756	1,934	1,843
sand	2,515	2,504	2,193	1,814	1,732
scf	2,579	2,578	2,385	2,420	2,017
styr	1,467	1,556	1,307	2,391	1,112
tbk	2,852	3,325	2,424	3,134	2,132
Total	43,893	47,407	37,73	41,337	33,655
Percentage	130,4%	140,8%	112,1%	122,9%	100%

TABLE 12. General characteristics of LUT-based Moore FSMs.

$Total$	$Auto$	$One-Hot$	$JEDI$	$DEMAIN$	U_3
$NumberofLUTs$	1280	1503	1073	1173	946
$Consumedenergy, W$	43,893	47,407	37,73	41,337	33,655
$Frequency, MHz$	2019,93	2019,2	2687,08	2157,49	2992,14
$\beta, , mJ/LUT$	16,9	15,6	13,08	16,3	11,8
$Percentage$	143%	132%	111%	138%	100%

memory functions are represented as the system (2). As a rule, a Moore FSM has more states than an equivalent Mealy FSM [14]. It makes the system (2) of a Moore FSM more complex than its counterpart of an equivalent Mealy FSM. But the outputs of Mealy FSMs depend on inputs and state variables [14], [37]:

$$Y = Y(T, X). \tag{27}$$

Obviously, the functions (27) have more arguments than functions (3) of an equivalent Moore FSM. So, each FSM model has its own advantages and disadvantages. It is impossible to say unequivocally that one model is always better than another. Let us analyze the influence of specifics of Mealy and Moore FSMs on optimization methods for LUT-based design. We hope this will help to show more clearly the novelty of our approach to optimization of LUT-based Moore FSMs.

As follows from (27), the one-hot codes of outputs are generated by the block LUTerY. These codes have N bits. In Moore FSMs, outputs depend only on state variables. Here, state codes play role of maximum codes of collections of outputs. In Moore FSMs, the LUTerY transforms these maximum codes into one-hot codes of outputs. Due to this difference, different approaches are used to optimize characteristics of the part of FSM circuit generating outputs $y_n \in Y$.

As follows from (2), for Mealy FSMs, functions $f_i \in \Phi \cup Y$ depend on terms (6). If the condition

$$NA(f_i) > S_L \tag{28}$$

takes place, then the corresponding function $f_i (i \in 1, \ldots, N + R)$ is represented by a multi-level circuit. It is known, that multi-level circuits have less operating frequency and consume more power than their single-level counterparts [2], [33].

If condition (6) takes place, then a corresponding circuit can be optimized by elimination of the direct dependence of outputs $y_n \in Y$ on inputs $x_l \in X$. This could be done using the encoding of collections of outputs. This approach allows improving the characteristics of Mealy FSMs. It leads to PY Mealy FSMs shown in Fig. 3. But if condition (13) takes place, then the circuit of LUTerY is multi-level. To diminish the number of levels in circuits implementing FSM outputs, we propose the approach of mixed encoding of collections of outputs [50].

In the case of mixed encoding, some outputs $y_n \in Y_0$ depend on terms (6). They are represented by one-hot codes generated by LUTerΦZ. The outputs $y_n \in Y_R$ form new collections of outputs which are encoded by maximum binary codes. These codes are transformed into one-hot codes of $y_n \in Y_R$ by LUTerY. This approach leads to PY_M Mealy FSMs shown in Fig. 8

Our research [50] shows that this approach significantly reduces the number of LUTs in circuits of Mealy FSMs. The negative effect of this approach is a decrease in the operating frequency due to elimination of direct dependence between outputs and inputs.

This method cannot be used to optimize the circuit of Moore FSM because its outputs do not directly depend on inputs $x_l \in X$. Given this feature, we have adapted the approach of mixed encoding of collections of outputs proposed in [50]. If condition (5) takes place, then we construct two sets of collections of outputs. The outputs $y_n \in Y_L$ form collections encoded by maximum binary codes. To encode them, additional variables $z_r \in Z$ are used, where set Z includes $R_L = S_L$ variables. These codes are transformed by LUTerL into one-hot codes (Fig. 4). Because $R_L = S_L$, it is enough a single LUT to implement any function $y_n \in Y_L$. The collections of outputs $y_n \in Y_R$ are encoded using state variables. So, these outputs are still implemented as functions (3). They are generated by LUTerR. We execute the partition of the set Y by sets Y_L and Y_R in a way minimizing the number of outputs in the set Y_R. Such an approach allows encoding of states in a way minimizing the number of literals in functions $y_n \in Y_0$. The minimization can decrease the number of LUTs and their levels in the circuit of Moore FSM U_4 compared with equivalent FSM U_1.

So, the proposed approach is a new one. It is not a mixed encoding of collections of outputs proposed in [50]. We propose a quite different way for reducing the numbers of LUTs in circuits of Moore FSMs. In this new approach, only maximum codes of collections of outputs are used. This is a main scientific contribution of this article. Until now, we have considered FSMs as separate blocks. However, they are parts of digital systems. So, FSMs interact with other parts of digital systems. Theoretically, the Mealy FSMs have benefits such as lower resource usage and faster response to their inputs. But in the case of FPGA-based digital systems, these benefits can be significantly reduced.

For Mealy FSMs, outputs (27) are generated in parallel with input memory functions. But in practice, outputs (3) can depend on up to $L + R \approx 30$ arguments [14]. Due to the rather small amount of LUT's inputs ($S_L \leq 6$), it is necessary to use the methods of functional decomposition. After decomposition, the clock cycle time increases. For Moore FSMs, outputs (3) depend on R arguments. Moreover, there are methods of state encoding [36], [47] which allow minimizing the numbers of arguments in system (3). In the best case, there are exactly N LUTs in the circuit of LUTerY. This leads to a reduction of the difference in the operating frequency for equivalent Mealy and Moore FSMs.

Next, it is known [2], [24] that outputs of Mealy FSM are not stable. To make them stable, it is necessary to use an additional output register having N latches. To operate, the register consumes power. Also, it is necessary to use an additional circuit to deliver the master clock to the output register. This circuit consumes some resources of an FPGA chip (LUTs, interconnections, power). The output register adds an additional increase to the cycle time. For Moore FSMs, only the state register is required which includes only R flip-flops. If $R \ll N$, then the state register of a Moore FSM consumes significantly less power than a pair <state register, output register> of equivalent Mealy FSM. In the case of Moore FSM U_3, it is necessary to use flip-flops to keep additional variables $z_r \in Z$. But even in this case, it is necessary only $(R + R_L) \leq 2R$ flip-flops.

In the case of Moore FSMs, there are effective methods [24], [33] for optimizing the LUT-based circuits of blocks LUTerΦ generating input memory functions (1). These methods allow getting a circuit of LUTerΦ having practically the same amount of LUTs as for an equivalent Mealy FSM. The methods [24], [33] use classes of pseudoequivalent states of Moore FSMs. A single class of pseudoequivalent states corresponds to a state of an equivalent Mealy FSM. We do not discuss this approach in our article. But the corresponding methods can be found in [24], [33].

It follows from the above that it is difficult to say which FSM model (Mealy or Moore) will be better to implement the LUT-based circuit of a particular sequential block. It depends on characteristics of both an FSM (the numbers of inputs, outputs and states, relations of the number of functions for which the condition (28) is met to the total number of functions) and FPGA (number of LUT's inputs, architecture of a configurable logic block). The criteria of optimality of a digital system as a whole are also significant. For example, for embedded systems [3], it is very important to reduce the power consumption. It is quite possible that a Moore FSM-based sequential block will consume less power than a Mealy FSM-based block. Therefore, the final decision on which FSM model to use is made by a designer of a particular digital system.

IX. CONCLUSION

The paper presents an original approach targeting FPGA-based Moore FSMs. The proposed design method leads to FSM circuits having a single level of LUTs in the blocks generating outputs $y_n \in Y$. The method is based on dividing the set of outputs Y by the sets Y_0 and Y_L. The outputs $y_n \in Y_0$ depend on state variables, the outputs $y_n \in Y_L$ on some additional variables. The splitting outputs is performed so that the condition (21) takes place.

The experiments show that this approach leads to reducing such FSM characteristics as the number of LUTs, consumed energy and delay. We compared our approach with four other methods. The experiments were conducted with Xilinx Vivado 2019.1 package. Also, we propose a generalized characteristic for comparing different design methods. It is directly proportional to the FSM operating frequency and diversely proportional to the product of consumed power and the number of LUTs in an FSM circuit.

There is a single limitation to the application of the proposed method. Namely, it does not make sense to use our approach if the number of state variables does not exceed the number of LUT's inputs.

The proposed method belongs to the group of methods of structural decomposition [21]. Our future research is connected with attempts to apply this method for CPLD-and ASIC-based FSMs.

REFERENCES

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

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

[3] O. Barkalov, L. Titarenko, and M. Mazurkiewicz, *Foundations of Embedded Systems* (Studies in Systems, Decision and Control), vol. 195. Cham, Switzerland: Springer, 2019.

[4] B. D. Brown and H. C. Card, ''Stochastic neural computation. I. Computational elements,'' *IEEE Trans. Comput.*, vol. 50, no. 9, pp. 891–905, Sep. 2001.

[5] P. Li, D. J. Lilja, W. Qian, M. D. Riedel, and K. Bazargan, ''Logical computation on stochastic bit streams with linear finite-state machines,'' *IEEE Trans. Comput.*, vol. 63, no. 6, pp. 1474–1486, Jun. 2014.

[6] J. Li, A. Ren, Z. Li, C. Ding, B. Yuan, Q. Qiu, and Y. Wang, ''Towards acceleration of deep convolutional neural networks using stochastic computing,'' in *Proc. 22nd Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2017, pp. 115–120.

[7] Y. Xie, S. Liao, B. Yuan, Y. Wang, and Z. Wang, ''Fully-parallel area-efficient deep neural network design using stochastic computing,'' *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 12, pp. 1382–1386, Dec. 2017.

[8] A. Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu, and W. J. Gross, ''VLSI implementation of deep neural network using integral stochastic computing,'' *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 10, pp. 2688–2699, Oct. 2017.

[9] N. I. Rafla and I. Gauba, ''A reconfigurable pattern matching hardware implementation using on-chip RAM-based FSM,'' in *Proc. 53rd IEEE Int. Midwest Symp. Circuits Syst.*, Aug. 2010, pp. 49–52.

[10] J. Glaser, M. Damm, J. Haase, and C. Grimm, ''TR-FSM: Transition-based reconfigurable finite state machine,'' *ACM Trans. Reconfigurable Technol. Syst.*, vol. 4, no. 3, pp. 1–14, Aug. 2011, doi: 10.1145/2000832.2000835.

[11] N. Das and P. A. Priya, ''FPGA implementation of reconfigurable finite state machine with input multiplexing architecture using hungarian method,'' *Int. J. Reconfigurable Comput.*, vol. 2018, pp. 1–15, 2018, Art. no. 6831901, doi: 10.1155/2018/6831901.

[12] 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.

[13] I. Skliarova, V. Sklyarov, and A. Sudnitson, *Design FPGA-Based Circuits Using Hierarchical Finite State Machine*. Tallinn, Estonia: TUT Press, 2012.

[14] S. Baranov, *Logic Synthesis of Control Automata*. Norwell, MA, USA: Kluwer, 1994.

[15] 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*. Cham, Switzerland: Springer-Verlag, 2002, pp. 363–370.

[16] J. Cong and K. Yan, ''Synthesis for FPGAs with embedded memory blocks,'' in *Proc. ACM/SIGDA 8th Int. Symp. Field Program. Gate Arrays*, New York, NY, USA, 2000, pp. 75–82, doi: 10.1145/329166.329183.

[17] V. Sklyarov, ''Synthesis and implementation of RAM-based finite state machines in FPGAs,'' in *Proc. Field-Program. Log. Appl., Roadmap Reconfigurable Comput.* Villach, Austria: Springer-Verlag, 2000, pp. 718–728.

[18] R. Senhadji-Navarro and I. Garcia-Vargas, ''High-performance architecture for Binary-Tree-Based finite state machines,'' *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 4, pp. 796–805, Apr. 2018.

[19] A. Tiwari and K. A. Tomko, ''Saving power by mapping finite-state machines into embedded memory blocks in FPGAs,'' in *Proc. Design, Autom. Test Eur. Conf. Exhib.*, 2000, pp. 916–921.

[20] 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.

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

[22] R. Brayton and A. Mishchenko, ''ABC: An academic industrial-strength verification tool,'' in *Computer Aided Verification*. Berlin, Germany: Springer, 2010, pp. 24–40.

[23] 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.

[24] I. Grout, *Digital Systems Design With FPGAs and CPLDs*. Amsterdam, The Netherlands: Elsevier, 2011.

[25] Altera. (Jan. 2020). *Cyclone IV Device Handbook*. [Online]. Available: http://www.altera.com/literature/hb/cyclone-iv/cyclone4-handbook.pdf

[26] *Virtex-5 Family Overview*, Xilinx, San Jose, CA, USA, 2020.

[27] C. Scholl, *Functional Decomposition with Application to FPGA Synthesis*. Boston, MA, USA: Kluwer, 2001.

[28] T. Kam, T. Villa, R. Brayton, and A. Sangiovanni-Vincentelli, *A Synthesis of Finie State Machines: Functional Optimization*. Boston, MA, USA: Springer-Verlag, 2010.

[29] M. Nowicka, T. Łuba, and M. Rawski, ''FPGA-based decomposition of Boolean functions: Algorithms and implementation,'' *Adv. Comput. Syst.*, vol. 10, pp. 502–509, Oct. 1999.

[30] I. Garcia-Vargas and R. Senhadji-Navarro, ''Finite state machines with input multiplexing: A performance study,'' *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 5, pp. 867–871, May 2015.

[31] X. Wu, M. Pedram, and L. Wang, ''Multi-code state assignment for low power design,'' *IEE Proc. Circuits, Devices Syst.*, vol. 147, no. 5, pp. 271–275, Oct. 2000.

[32] A. Barkalov, L. Titarenko, M. Kołopieńczyk, K. Mielcarek, and G. Bazydło, *Logic Synthesis for FPGA-Based Finite State Machines* (Studies in Systems, Decision and Control). Cham, Switzerland: Springer, 2015, vol. 38. [Online]. Available: http://link.springer.com/book/10.1007/978-3-319-24202-6

[33] A. A. Barkalov, L. A. Titarenko, and A. A. Barkalov, ''Structural decomposition as a tool for the optimization of an FPGA-based implementation of a mealy FSM,'' *Cybern. Syst. Anal.*, vol. 48, no. 2, pp. 313–322, Mar. 2012.

[34] A. Barkalov and A. Barkalov, Jr., ''Design of mealy finite-state machines with the transformation of object codes,'' *Int. J. Appl. Math. Comput. Sci.*, vol. 15, no. 1, pp. 151–158, 2005.

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

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

[37] H. Kubatova and M. Becvar, ''FEL–code: FSM internal state encoding method,'' in *Proc. 5th Int. Workshop Boolean Problems*, Jan. 2002, pp. 109–114.

[38] H. Kubatova, *Design Embedded Control Systerm*. New York, NY, USA: Springer-Verlag, 2005, pp. 177–187.

[39] E. Sentowich, K. Singh, L. Lavango, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. S. P. , R. Bryton, and A. Sangiovanni-Vincentelli, ''SIS: A system for sequential circuit synthesis,'' Dept. EECS, Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/ERL M92/41, 1992.

[40] LGSynth93. (Feb. 2020). *International Workshop on Logic Synthesis Benchmark Suite (Lgsynth93)*. [Online]. Available: https://people.engr.ncsu.edu/brglez/CBL/benchmarks/LGSynth93/LGSynth93

[41] 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.

[42] M. Rawski, P. Tomaszewicz, G. Borowski, and T. Łuba, ''Logic Synthesis Method of Digital Circuits Designed for Implementation with Embedded Memory Blocks on FPGAs,'' in *Design of Digital Systems and Devices*, M. Adamski, A. Barkalov, and M. Węgrzyn, Eds. Berlin, Germany: Springer-Verlag, 2011, pp. 121–144.

[43] M. Kołopieńczyk, L. Titarenko, and A. Barkalov, ''Design of emb-based Moore fsms,'' *J. Circuits, Syst., Comput.*, vol. 26, no. 7, pp. 1–23, 2017, doi: 10.1142/S0218126617501250.

[44] (Jan. 2020). *Xilinx*. [Online]. Available: http://www.xilinx.com

[45] (Jan. 2020). *VIVADO*. [Online]. Available: http://xilinx.com/products/design-tools/vivado.html

[46] S. Achasova, *Synthesis Algorithms for Automata With PLAs*. Voice, Russia: Soviet Radio, 1987.

[47] (Jan. 2020). *DEMAIN*. [Online]. Available: http://zpt2.tele.pw.edu.pl/Files/demain/demain.htm

[48] T. Michalski and Z. Kokosiński, "Functional decomposition of combinational logic circuits with pkmin," *Tech. Trans. Electr. Eng.*, vol. 2, pp. 191–202, Oct. 2016.
[49] (Jan. 2020). *ABC System*. [Online]. Available: https://people.eecs.berkeley.edu/~alanmi/abc/
[50] 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.

ALEXANDER BARKALOV received the M.Sc. degree in computer engineering from the Donetsk Politechnical 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 with 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 with the Institute of Informatics and Electronics, University of Zielona Góra, Poland. Her current research interests include the theory of telecommunication systems, theory of antennas, and theory of digital automata and its applications. She has taken part in a number of research projects sponsored by the Ministry of Science and Higher Education of Ukraine, from 1993 to 2005.

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. Since 2017, he has been a Lecturer with the State University of Applied Sciences, Głogów. His current research interests include methods of synthesis and optimization of control units in field-programmable logic devices, hardware description languages, perfect graphs and Petrie nets, algorithmic theory and safety of UNIX, and network systems.