REDUCING THE NUMBER OF LUTs FOR MEALY FSMs WITH STATE TRANSFORMATION

ALEXANDER BARKALOV a, LARYSA TITARENKO a, KAMIL MIELCAREK a,*

a Institute of Metrology, Electronics and Computer Science
University of Zielona Góra
ul. prof. Z. Szafrana 2, 65-516 Zielona Góra, Poland
e-mail: [A.Barkalov, L.Titarenko, K.Mielcarek]@imei.uz.zgora.pl

In many digital systems, various sequential blocks are used. This paper is devoted to the case where the model of a Mealy finite state machine (FSM) represents the behaviour of a sequential block. The chip area occupied by an FSM circuit is one of the most important characteristics used in logic synthesis. In this paper, a method is proposed which aims at reducing LUT counts for FPGA-based Mealy FSMs with transformation of state codes into FSM outputs. This is done using the combined state codes. Such an approach allows excluding a block of transformation of binary state codes into extended state codes. The proposed method leads to LUT-based Mealy FSM circuits having exactly three levels of logic blocks. Under certain conditions, each function for any logic level is represented by a circuit including a single LUT. The proposed approach is illustrated with an example of synthesis. The results of experiments conducted using standard benchmarks show that the proposed method produces LUT-based FSM circuits with significantly smaller LUT counts than is the case for circuits produced by other investigated methods (Auto and One-hot of Vivado, JEDI, and transformation of binary codes into extended state codes). The LUT count is decreased by an average of 17.96 to 91.8%. Moreover, if some conditions are met, the decrease in the LUT count is accompanied with a slight improvement in the operating frequency compared with circuits based on extended state codes. The advantages of the proposed method multiply with increasing the numbers of FSM inputs and states.

Keywords: Mealy FSM, FPGA, LUT, synthesis, extended state codes, combined state codes.

1. Introduction

Modern digital systems (Marwedel, 2018; Sklyarov et al., 2014) include a lot of sequential blocks, starting with rather simple binary counters and ending with very sophisticated digital controllers (Gajski et al., 2009; Borowczak and Vemuri, 2013). Very often, the behaviour of a sequential block is represented by the model of Mealy finite state machine (FSM) (Baranov, 1994; Micheli, 1994). In this paper, we discuss the case where Mealy FSMs circuits are implemented using internal resources of field-programmable gate arrays (FPGAs) (Intel, 2023; AMD, 2023a). This case is of great practical interest since a huge number of various projects are implemented on the basis of FPGAs (Ruiz-Rosero et al., 2019).

Several optimization problems arise when implementing FPGA-based FSM circuits (Kubica et al., 2021; Kubica and Kania, 2017; Barkalov et al., 2018). The main issues are a decrease in the chip area occupied by an FSM circuit, an increase in performance (the maximum operating frequency), and reducing the power consumption (Barkalov and Barkalov Jr., 2005; Sklyarov et al., 2014; Tiwari and Tomko, 2004). As a rule, reducing the area also reduces the power consumption (Barkalov et al., 2021). At the same time, it is very important that a reduction in the area does not lead to a significant decrease in the performance (Kubica et al., 2019; Maxfield, 2008). In this paper, we propose a method of area reduction which does not lead to a reduction in the maximum operating frequency. We discuss the case where an FPGA-based FSM circuit is implemented using look-up table (LUT) elements (Trimberger, 2015; Machado and Cortadella, 2020). We use the approach from the paper by Islam et al. (2020), where the chip area is estimated as an LUT count.

Modern LUTs have rather small amounts of inputs
A. Barkalov et al.

Fig. 1. State transition graph of Mealy FSM $S_1$.

(Intel, 2023; Microchip, 2023; AMD, 2023a). As a result, the methods of functional decomposition (Scholl, 2001; Soloviev and Czyzy, 1999) should be used to represent an FSM circuit by a network of interconnected LUTs. These circuits have many levels of logic and complex systems of spaghetti-type interconnections (Barkalov et al., 2020a). Therefore, it is very important to develop FSM design methods, where the negative impact of a limited number of LUT inputs is minimized. The methods of structural decomposition (Barkalov et al., 2021) are aimed at solving this problem.

The main contribution of this paper is a novel method of reducing LUT counts in logic circuits of Mealy FSMs. The method is based on a transformation of combined state codes, proposed in this paper, into FSM outputs. It is an improvement of the approach of Barkalov et al. (2022), where the transformation of maximum binary state codes into extended state codes is connected with the presence of a special block. This block generates extended state codes using codes of collections of outputs and identifiers. Thus, the main difference between our technique and the known FSM synthesis methods is the development of a new approach for representing state codes. As follows from our experiments, the proposed representation of state codes allows improving LUT counts in Mealy FSM circuits compared with the circuits produced by other investigated methods. In addition, due to the resulting decrease in the number of interconnections among the LUTs of an FSM circuit, our approach leads to a slight increase in the FSM performance.

The rest of the paper is organized as follows. Section 2 includes the basic information on LUT-based design of FSM circuits. The relative works are discussed in Section 3. The main idea of the proposed method is discussed in Section 4. Section 5 shows an example of synthesis. Section 6 includes experimental results. A concise conclusion ends the paper.

2. Synthesis of LUT-based FSMs

A Mealy FSM is represented using three sets and two functions (Baranov, 1994; Micheli, 1994). A set $A = \{a_1, \ldots, a_M\}$ includes internal states, a set $X = \{x_1, \ldots, x_L\}$ consists of FSM inputs, a set $Y = \{y_1, \ldots, y_N\}$ includes FSM outputs. A function of transition sets the dependence of the states of transitions on the current states and FSM inputs. A function of output shows the dependence of FSM outputs on the current states and FSM inputs. The state $a_1 \in A$ is an initial state.

An FSM can be represented by state transition tables (Micheli, 1994; Minns and Elliot, 2008), binary decision diagrams (Anceau, 1986; Mili, 2016; Kubica et al., 2019), and-inverter graphs (Brayton and Mishchenko, 2010) and graph-schemes of algorithms (Baranov, 2008).

In this paper, FSMs are represented by state transition graphs (STGs) (Micheli, 1994; Senhadji-Navarro and Garcia-Vargas, 2018). An example of an STG for an FSM $S_1$ is shown in Fig. 1.

STG nodes correspond to FSM states. As follows from Fig. 1 there are $M = 8$ states in FSM $S_1$. STG arcs correspond to interstate transitions. There are $H = 15$ arcs in the STG (Fig. 1). An arc is directed from a current state $a_m \in A$ to a state of transition $a_s \in A$. The arc number $h \ (h \in \{1, \ldots, H\})$ is marked by a pair $\langle X_h, Y_h \rangle$. The symbol $X_h$ stands for a conjunction of FSM inputs (or their compliments) causing the $h$-th transition. The symbol $Y_h$ stands for a collection of outputs (CO) generated during the $h$-th transition.

The analysis of Fig. 1 allows obtaining the sets $X = \{x_1, \ldots, x_4\}$ and $Y = \{y_1, \ldots, y_7\}$. This gives the values $L = 4$ and $N = 7$. Obviously, the state $a_1 \in A$ is an initial state of $S_1$.

An FSM circuit is represented by some systems of Boolean functions (SBFs) (Baranov, 1994; Micheli, 1994). To get these SBFs, the step of state assignment should be executed. During this step, states $a_m \in A$ are represented by $R$-bit codes $K(a_m)$. The minimum value of $R$ is determined as

$$R = \lceil \log_2 M \rceil.$$  \hspace{1cm} (1)

The formula (1) determines maximal binary state codes (Sutter et al., 2002).

The bits of $K(a_m)$ correspond to state variables forming the set $T = \{T_1, \ldots, T_R\}$. State codes are kept in a state register (RG) consisting of $R$ flip-flops. To load the code $K(a_1)$ into the RG, a special pulse $Start$ is used. To change the RG contents, input memory functions (IMFs) are used. There are $R$ IMFs forming the set $\Phi$. As a rule, $D$ flip-flops are used in state registers (Sklarova et al., 2012). In our paper, we use this approach. Thus, there is a set $\Phi = \{D_1, \ldots, D_R\}$. The synchronization pulse $Clock$ allows changing the contents of the RG.

Using state codes allows transforming an STG into a
direct structure table (DST) (Baranov, 1994; Klimovich and Solov’ev, 2012). The following SBFs are derived from a DST:

\[ Y = Y(T, X), \quad \Phi = \Phi(T, X). \]

The system (2) corresponds to the function of output, the system (3) corresponds to the function of transitions.

In this paper, we discuss the case where the SBFs (2) and (3) are implemented using LUT-based configurable logic blocks (CLBs). A CLB includes LUTs, flip-flops, multiplexers and internal connections (Chapman, 2014). The flip-flops can be used as a hidden distributed state register (Skyyarov et al., 2014). In our previous work (Barkalov et al., 2018), we employed the symbol LUTer to define a logic block consisting of LUTs. Using this symbol, we can show a structural diagram of P Mealy FSM (Fig. 2).

In the P FSM, the LUTerΦ implements the SBF (3). The outputs of LUTs generating IMFs \( D_v \in \Phi \) are connected with the inputs of flip-flops. Therefore, there is a distributed RG inside LUTerΦ. LUTerY implements the SBF (2).

The largest manufacturer of FPGAs is the company AMD (Xilinx) (AMD, 2023a). Hence our research is focused on Xilinx FPGAs from the Virtex-7 family (AMD, 2019). A LUT of Virtex-7 has \( S_L = 6 \) inputs (AMD, 2023a). A slice of Virtex-7 consists of four 6-LUTs and eight flip-flops. Using internal multiplexers (MX) allows creating either two 7-LUTs or a single 8-LUT (Chapman, 2014).

As follows from the results of Baranov (1994; 2008), a sum-of-products (SOP) of functions \( f_i \in Y \cup \Phi \) can include up to 30–40 literals. Thus, there is an obvious contradiction between the extremely limited value of \( S_L \) and the very large number of arguments in functions \( f_i \in Y \cup \Phi \). This contradiction leads to multi-level circuits of LUTerΦ and LUTerY (Barkalov et al., 2020a). It is known (Wolf, 2004) that multi-level circuits have worse characteristics than their single-level counterparts. Therefore, it is necessary to decrease the number of logic levels in circuits of LUT-based Mealy FSMs.

3. Related work

The number of literals in each function representing an FSM circuit can be reduced by using various methods of structural decomposition (SD) (Barkalov et al., 2021; Senhadji-Navaro et al., 2015). The main idea of SD is the following. An FSM circuit is represented as a composition of logic blocks. Each block has its unique systems of inputs and outputs. SD assumes an increase in the number of logic levels of an FSM circuit. However, each of these blocks is represented by an SOP that has much fewer literals than the SOPs of the functions (2)–(3).

Numerous studies (Barkalov et al., 2021) have shown that this approach can significantly reduce the number of LUTs compared with that for P FSMs.

One of such methods is discussed by Barkalov et al. (2022). It is a method of transformation of states into COs \( Y_q \subseteq Y \). A particular CO is represented using states and identifiers \( I_q \subseteq Y \). Thus, each CO \( Y_q \subseteq Y \) is represented by a pair \( P_q = \langle a_m, I_q \rangle (g \in \{1, \ldots, G\}) \).

The need for identifiers is explained as follows. For example, there are two transitions into state \( a_5 \in A \). A CO \( Y_3 \) is generated during the first one, a CO \( Y_1 \) is generated in the second case. To distinguish these COs, two identifiers \( I_1, I_2 \) are necessary. Now, these COs can be represented as pairs \( P_1 = \langle a_5, I_1 \rangle \) and \( P_2 = \langle a_5, I_2 \rangle \). For example, the pair \( P_1 \) determines \( Y_3 \) and \( P_2 \) determines \( Y_7 \). If \( N_m \) different COs are generated during transitions into a state \( a_m \in A \) then \( N_m \) identifiers are necessary to distinguish these COs. The number of identifiers for a particular FSM is determined as

\[ J = \max(N_1, \ldots, N_M). \]

To synthesize an FSM circuit, the identifiers should be encoded by binary codes \( K(I_j) \). These include

\[ R_V = \lceil \log_2 J \rceil \]

additional variables. We use elements of the set \( V = \{v_1, \ldots, v_{R_V}\} \) to encode the identifiers \( I_j \in SI \).

Now, codes \( K(Y_q) \) are represented as

\[ K(Y_q) = K(a_m) \ast K(I_j), \]

where \( \ast \) is a sign of concatenation. The codes (6) are transformed into one-hot codes of outputs. Such an approach leads to \( P_M \) Mealy FSM (Barkalov et al., 2020b). Its LUT-based structural diagram is shown in Fig. 4.
In $P_A Y$ FSM, LUTerTV implements the SBFs (3) and
\[ V = V(T, X), \]  
while LUTerY implements the SBF
\[ Y = Y(T, V). \]

Obviously, variables $v_r \in V$ replace inputs $x_e \in X$ in (5). As a rule, the following condition holds (Barkalov et al., 2020b):
\[ RV \ll L. \]

Thus the SOPs of the functions (3) are much simpler than their counterparts for the functions (2). As a result, there are fewer LUTs in the circuit of $P_A Y$ FSM compared with the equivalent P Mealy FSM (Barkalov et al., 2020b).

A SOP of function $f_i \in V \cup \Phi$ includes $NA(f_i)$ arguments. If the condition
\[ NA(f_i) > S_L \]
holds, then the circuit of LUTerTV is multi-level. To decrease the number of LUTs in LUTerTV, it is possible to use the twofold state assignment (Barkalov et al., 2018; 2020c). Such an approach is proposed by Barkalov et al. (2022).

The method of Barkalov et al. (2018) is based on finding a partition $\Pi_A = \{A^1, \ldots, A^K\}$ of the set $A$. Each class $A^k \in \Pi_A$ is characterized by sets $X^k$ and $T^k$. The set $X^k \subseteq X$ includes inputs $x_e \in X$ determining transitions from states $a_m \in A^k$. The set $T^k \subseteq T$ includes $R_k$ state variables encoding states $a_m \in A^k$ by codes $C(a_m)$. The value of $R_k$ is determined as
\[ R_k = \lceil \log_2(|A^k| + 1) \rceil. \]

The value of $M_k = |A^k|$ is increased by 1 to take into account the relation $a_m \notin A^k$.

Each state $a_m \in A$ has two codes: $K(a_m)$ and $C(a_m)$. There are $R_0$ state variables in the set $T = T^1 \cup T^2 \cup \ldots \cup T^K$.

Each class $A^k \in \Pi_A$ determines two SBFs:
\[ \Phi^k = \Phi^k(T^k, X^k), \]
\[ V^k = V^k(T^k, X^k). \]

We name the functions (12)–(13) partial functions. To obtain the functions (3) and (7), the disjunctions of partial functions $D_r^k \in \Phi^k$ and $V_r^k \in V^k$ should be found. This approach determines a $P_{ATY}$ Mealy FSM (Fig. 4).

In the $P_{AT}$ FSM, LUTerk implements the partial functions (12) and (13). LUTerOR generates state variables $T_r \in T$ and additional variables $v_r \in V$, LUTerY implements the SBF (3). To create codes $C(a_m)$, LUTerT implements the following SBF:
\[ T = T(T). \]

Some experiments (Barkalov et al., 2022) show that $P_{ATY}$ FSMs have better characteristics than the equivalent $P_A Y$ FSMs and FSMs based on various methods of state assignment. The following methods of state assignment were analyzed: Auto and One-hot of Vivado (Vivado, 2023) and JEDI (Sentowitch et al., 1992a; 1992b). The replacement of other models by $P_{ATY}$ FSMs allows reduction of the LUT counts (from 11.9% to 62.6%) and increasing the maximum operating frequency (from 11.7% to 18.4%).

Our analysis shows that $P_{ATY}$ FSMs have one serious drawback. Namely, the transformation $K(a_m) \rightarrow C(a_m)$ requires an additional block LUTerT, which consumes some internal resources of a chip (LUTs and interconnections). Consequently, it would be useful to eliminate LUTerT. Such an approach is proposed in the present paper.

4. Main idea of the proposed method

The proposed method can be applied if the condition (10) holds for a $P_A Y$ FSM. In this case, we suggest to find a partition $\Pi_B = \{B^1, \ldots, B^l\}$ of the set $A$. As for a
Reducing the number of LUTs for Mealy FSMs with state transformation

Fig. 5. Structural diagram of $P_{AC}Y$ Mealy FSM.

class $A^k \in \Pi_A$, each class $B^i \in \Pi_B$ defines a set $X^i \subseteq X$. The set $X^i$ includes $L_i$ inputs determining transitions from states $a_m \in B^i$.

If $|B^i| = M_i$, then it is enough to use

$$R_i = \lceil \log_2 M_i \rceil \quad (15)$$

state variables to encode states $a_m \in B^i$ by codes $C(a_m)$.

As in the case of $\Pi_A$, the following condition should hold for each class $B^i \in \Pi_B$:

$$R_i + L_i \leq S_L \quad (16)$$

In this case, each partial function is implemented by a single LUT. It is enough to use

$$R_S = \max(R_1, \ldots, R_I) \quad (17)$$

state variables to encode any state from any class $B^i \in \Pi_B$. These variables form a set $T_S = \{T_1, \ldots, T_{R_S}\}$.

Obviously, any state $a_m \in A$ should have its unique code. To reach this goal, we propose to encode a class $B^i \in \Pi_B$ with a binary code $C(B_i)$. There are

$$R_C = \lceil \log_2 I \rceil \quad (18)$$

bits in the codes $C(B^i)$. To encode classes, we use the elements of a set $T_C = \{\tau_1, \ldots, \tau_{R_C}\}$.

Using codes $C(B^i)$ and $C(a_m)$, where $a_m \in B^i$, gives a combined state code (CSC),

$$CC(a_m) = C(B^i) * C(a_m) \quad (19)$$

In (19), the symbol $*$ stands for the concatenation of codes. A $P_{AC}Y$ FSM based on (19) has the structural diagram shown in Fig. 5.

In $P_{AC}Y$ FSM, LUTeri implements the SBFs

$$\Phi^i = \Phi^i(T_S, X^i) \quad (20)$$

while LUTer implements the SBFs

$$T_C = T_C(T_C, \Phi^1, \ldots, \Phi^I) \quad (22)$$

$$V = V(T_C, V^1, \ldots, V^I) \quad (23)$$

The set $T_C$ includes class variables $\tau_r \in T_C$ and state variables $T_r \in T_S$: $T_C = T_C \cup T_S$. The block LUTerY implements FSM outputs represented as

$$Y = Y(T_C, V) \quad (24)$$

The set of IMFs includes $R_{CC}$ elements:

$$R_{CC} = R_C + R_S \quad (25)$$

Our analysis of standard benchmarks (LGSynth93, 1993) shows that the following relations hold:

$$2R < R_0 < 4R \quad (26)$$

$$R \leq R_{CC} \leq R + 1, \quad (27)$$

$$R_C \leq 4 \quad (28)$$

Accordingly, $P_{AC}Y$ FSMs have the following advantages compared with the equivalent $P_{AT}Y$ FSMs:

1. There is no LUTer in $P_{AC}Y$ FSMs. If there is the same LUT count for parts of circuit generating the SBFs (8), (12), (13) and (20)–(24), then $P_{AC}Y$ FSMs require fewer LUTs than the equivalent $P_{AT}Y$ FSMs.

2. $P_{AC}Y$ FSMs have fewer feedback signals between the RG and LUTs of the first level of logic. This allows simplification of the interconnects compared with $P_{AT}Y$ FSMs.

3. There is no special code to show that $a_m \notin B^i$. Hence the following condition can occur:

$$I < K \quad (29)$$

Reducing the number of blocks LUTer of the first logic level can lead to a decrease in the LUT counts for $P_{AC}Y$ FSMs.

In this paper, we propose the following method for synthesis of $P_{AC}Y$ FSMs:

1. Constructing the partition $\Pi_B$ with a minimum number of classes $I$.

2. Representing outputs $y_m \in Y$ by pairs $(a_m, I_j)$.

3. Encoding states, classes $B^i \in \Pi_B$ and identifiers in a way optimizing the SBF (24).

4. Constructing the SBF (24) representing LUTerY.
5. Constructing tables representing blocks LUTeri on the basis of the DST of $P_{AC}Y$ of FSM.

6. Constructing the SBFs (20) and (21).

7. Constructing a table of LUTeriF.

8. Constructing the SBFs (22) and (23).

9. Implementing the FSM circuit using internal resources of a particular FPGA chip.

We use the record Model($S_n$) to show that a particular model is employed for synthesis of the logic circuit for an FSM $S_n$. In the next section, we discuss an example of the synthesis of Mealy FSM $P_{AC}Y(S_1)$.

### 5. Example of synthesis

To design the logic circuit of $P_{AC}Y(S_1)$, we use LUTs having $S_1 = 4$. As follows from (16), each class $B^i \in \Pi_B$ should satisfy the condition (Barkalov et al., 2018)

$$L_i + R_i \leq 4.$$  
(30)

The value of $R_i$ is determined by (15).

This step is very important (Barkalov et al., 2018). Its outcome significantly affects the LUT count of the resulting circuit. We use the methods of (Barkalov et al., 2018; 2020c) to find a partition $\Pi_B$ with a minimum number of classes.

There is a set $A = \{a_1, \ldots, a_8\}$ in the discussed case. Using the method of Barkalov et al. (2018) gives a partition $\Pi_B = \{B^1, B^2\}$, where $B^1 = \{a_1, a_3, a_5, a_6\}, B^2 = \{a_2, a_4, a_7, a_8\}$. These classes determine sets $X^1 = \{x_1, x_2\}$ and $X^2 = \{x_3, x_4\}$ with $L_4 = L_2 = 2$. Using (15) and (17) gives $R_4 = R_2 = R_S = 2$. Obviously, the condition (20) holds for these classes. There are $I = 2$ classes in the partition $\Pi_B$.

The analysis of an STG (Fig. 1) allows representing FSM outputs by pairs $(a_m, I_j)$. There are $I = 2$, $G = 11$, $SI = \{I_1, I_2\}$. The representation is shown in Table 1. Let us discuss it below.

In the given case, there are $Q = 8$ COS $Y_q \subseteq Y$: $Y_1 = \emptyset, Y_2 = \{y_1, y_2\}, Y_3 = \{y_2, y_4\}, Y_4 = \{y_3\}, Y_5 = \{y_3\}, Y_6 = \{y_2, y_6\}, Y_7 = \{y_3, y_7\}$ and $Y_8 = \{y_2\}$. These COS are shown in the row $Y_q$ of Table 1. The last row of Table 1 includes pairs $P_1 - P_{11}$. Their contents follow from rows $a_m$ and $I_j$. For example, there are pairs $P_1 = (a_2, I_1)$, $P_2 = (a_5, I_1)$, and so on. To distribute the identifiers among the pairs $P_q$, we use the approach of Barkalov et al. (2022).

A code $CC(a_m)$ is represented by a conjunction $A_m$ of class variables $\tau \in \tau_c$ and state variables $T_{re} \in T_S$. An identifier $I_1$ is represented by a conjunction $CI_1$ of variables $v_i \in V$. A pair $P_{12} = (a_m, I_2)$ is represented by a conjunction $CP_1 = A_m \cdot CI_1$. Using these conjunctions, we can form the SOPs of the functions (24).

### Table 1. Representation of outputs for FSM $S_1$.

<table>
<thead>
<tr>
<th>$Y_q$</th>
<th>$y_1$</th>
<th>$y_2$</th>
<th>$y_3$</th>
<th>$y_4$</th>
<th>$y_5$</th>
<th>$y_6$</th>
<th>$y_7$</th>
<th>$y_8$</th>
<th>$y_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_m$</td>
<td>$a_2$</td>
<td>$a_5$</td>
<td>$a_4$</td>
<td>$a_6$</td>
<td>$a_4$</td>
<td>$a_3$</td>
<td>$a_2$</td>
<td>$a_4$</td>
<td>$a_7$</td>
</tr>
<tr>
<td>$I_1$</td>
<td>$I_1$</td>
<td>$I_1$</td>
<td>$I_1$</td>
<td>$I_1$</td>
<td>$I_2$</td>
<td>$I_2$</td>
<td>$I_2$</td>
<td>$I_2$</td>
<td>$I_1$</td>
</tr>
<tr>
<td>$P_{11}$</td>
<td>$P_1$</td>
<td>$P_2$</td>
<td>$P_3$</td>
<td>$P_4$</td>
<td>$P_5$</td>
<td>$P_6$</td>
<td>$P_7$</td>
<td>$P_8$</td>
<td>$P_9$</td>
</tr>
<tr>
<td>$P_{10}$</td>
<td>$P_{11}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Because of (16), each partial function is implemented as a single LUT circuit. In consequence, there is no influence of codes $C(B^1)$ and $C(a_m)$ on the LUT count. Therefore, classes and states should be encoded in a way optimizing the SBF (24).

We use the following rules for encoding of classes and states:

1. The classes with a maximum number of common outputs $y_n \in Y$ must have adjacent codes (with a Hamming distance equal to 1).
2. If states $a_m \in B^i$ and $a_m \in B^j$ are parts of pairs with the same outputs $y_n \in Y$, then these states should have adjacent codes.

The identifiers are encoded using the approach by Barkalov et al. (2022). It is the following: the more often an identifier appears in Table 1 the more zeros its code includes.

In the discussed case, there are sets $\tau_c = \{\tau_1\}$ and $T_s = \{T_1, T_2\}$. Using (5) gives $R_{TV} = 1$ and $V = \{v_1\}$. We can encode the classes in the following way: $C(B^1) = 1$ and $C(B^2) = 1$. The codes of states are shown in Fig. 6.

The identifier $I_1$ (resp. $I_2$) appears 7 (resp. 4) times in Table 1. Thus, there is $K(I_1) = 0$ and $K(I_2) = 1$.

The following condition holds for this example:

$$R_{ec} + R_{TV} \leq S_L.$$  
(31)

Because of it, each function (8) is represented by a single LUT. In this case, the proposed approach for encoding $B^i \in \Pi_B$ and $a_m \in B^i$ allows reducing the number of interconnections between the blocks LUTerF and LUTerY.

Using codes $C(B^i)$, $C(a_m)$, and $K(I_j)$ leads, for example, to the following SOPs:

$$y_1 = A_1 \cdot CI_1 \vee A_3 \cdot CI_1$$
$$= \tau_1 T_1 T_2 v_1 \vee \tau_1 T_1 T_2 v_1,$$
$$y_3 = A_3 \cdot CI_2 \vee A_1 \cdot CI_2 \vee A_2 \cdot CI_2 \vee A_7 \cdot CI_2$$
$$= \tau_1 T_1 T_2 v_1 \vee T_1 T_2 v_1.$$  
(32)
implementing the functions (22) and (23). Its table conveys the rows marked by symbols \( D_1, \ldots, D_{R_{\text{cr}}}, \)
\( v_1, \ldots, v_{R_V}. \) There are \( I + 1 \) columns in a table of LUTerF. The last \( I \) columns contain the numbers of blocks of LUTerI. If a partial function is generated by LUTerI, then there is 1 on the intersection of the corresponding function and the column \( i \).

Analysis of Table 2 shows that all \( R_{\text{cr}} + R_V \) partial functions are generated by each LUTerI \( (i = 1, 2) \). Hence, the SBFs (22) and (23) are constructed in a trivial way. They are the following:

\[
\begin{align*}
D_1 &= \tau_1 D_1^1 \lor \tau_1 D_1^2, \\
D_2 &= \tau_1 D_2^1 \lor \tau_1 D_2^2, \\
D_3 &= \tau_1 D_3^1 \lor \tau_1 D_3^2, \\
v_1 &= \tau_1 v_1^1 \lor \tau_1 v_1^2.
\end{align*}
\]

The variable \( \tau_1 \in T_i \) is connected with a control input of an MX. The data inputs of each MX are connected with the corresponding outputs of LUTer1–LUTer2.

Using the obtained SBFs, we can create the logic circuit of FSM \( P_{AC} Y(S_1) \). Let us estimate the LUT count of this circuit.

As follows from (34), in total, there are 12 LUTs in the circuit of LUTer1, LUTer2 and LUTerF. Because the condition (26) holds, there are \( N = 7 \) LUTs in the circuit of LUTerY. Consequently, there are 19 LUTs in the circuit of \( P_{AC} Y(S_1) \). Also, this circuit has three levels of LUTs. We do not show this circuit.

To compare the approach by Barkalov et al. (2022) with our new method, we synthesized the logic circuit of \( P_{AT} Y(S_1) \). If \( S_1 = 4 \), then \( K = 3 > I = 2 \). Consequently, there is a partition \( \Pi A = \{ A^1, A^2, A^3 \} \), where \( A^1 = \{ a_1, a_3, a_5 \} \), \( A^2 = \{ a_2, a_4, a_6 \} \), \( A^3 = \{ a_7, a_8 \} \). This means that the first level of FSM circuit includes three blocks. There are 14 LUTs in the circuits of LUTer1–LUTer3. There is \( R_1 = R_3 = R_4 = 2 \), but the LUTerTV consists of five LUTs. Thus, there are seven LUTs in LUTerY and six LUTs in LUTerT. In consequence, there are 32 LUTs in the circuit of \( P_{AT} Y(S_1) \).

As can be seen, the transition from FSM \( P_{AT} Y(S_1) \) to FSM \( P_{AC} Y(S_1) \) allows us to reduce the LUT count by 1.68 times. Obviously, to implement the circuits of LUT-based FSMs, we should use various methods of technology mapping (Kubica et al., 2021; Krishnamoorthy and Tessier, 2003; Kubica and Kania, 2017; Ling et al., 2005; Zgheib and Ouais, 2015). This can be done using standard CAD packages such as, for example, Vivado by AMD (Xilinx) (Vivado, 2023). We do not show the results of implementation for our example.

In the next section, we present the results of experiments conducted with standard benchmarks (LGSynth93, 1993) and CAD tool Vivado.

| Table 2. Direct structure table of \( P_{AC} Y(S_1) \). |
| --- | --- | --- | --- | --- | --- | --- | --- |
| \( a_m \) | \( C(a_m) \) | \( a_s \) | \( CC(a_s) \) | \( X_h \) | \( I_h \) | \( \Phi_h \) | \( V_h \) |
| a1 | 00 | a2 | 101 | x1 | I1 | D1 D3 | – 1 |
| a3 | 010 | – | x1 x2 | I1 | D2 | – 2 |
| a4 | 100 | – | x1 x2 | I1 | D1 | – 3 |
| a2 | 01 | a3 | 010 | x3 x4 | I2 | D2 | v1 4 |
| a4 | 101 | – | x3 x4 | I1 | D1 | – 5 |
| a5 | 001 | – | x3 | I1 | D3 | – 6 |
| a3 | 10 | a4 | 100 | x1 | I2 | D1 | v1 7 |
| a5 | 001 | – | x1 | I2 | D3 | v1 8 |
| a4 | 00 | a5 | 001 | x3 x4 | I1 | D3 | v1 9 |
| a6 | 110 | a7 | 111 | – | D1 D2 | – 13 |
| a7 | 10 | a8 | 100 | 1 | I2 | D1 | v1 15 |

All other SOPs for LUTerY can be obtained in the same way.

Blocks LUTer1–LUTerI generate the functions (20) and (21). Thus, in a DST of \( P_{AC} Y \) FSM, each CO \( Y_q \subseteq Y \) should be replaced by a pair \( \langle a_m, I_j \rangle \). For example, the CO \( Y_2 = \{ y_1, y_2 \} \) is generated during the transition \( \langle a_1, a_2 \rangle \) caused by \( x_1 \). As follows from Table 1, this situation is represented by a pair \( \langle a_2, I_1 \rangle \) = P1. Thus, in the DST of \( P_{AC} Y(S_1) \), the identifier \( I_1 \) should replace CO \( Y_2 \). All rows of the DST are filled in the same way.

Thus, there are the following columns in the DST:

| \( a_m \) (a current state), \( C(a_m) \), \( a_s \) (a next state), \( CC(a_s) \), \( X_h \) (an input signal determining the transition \( \langle a_m, a_s \rangle \)), \( I_h \) (an identifier replacing the CO \( Y_q \subseteq Y \)), \( \Phi_h \) (a collection of IMFs equal to 1 to load the \( CC(a_s) \) into RG), \( V_h \) (a conjunction corresponding to \( K(I_h) \)), \( h \) (the number of transitions). There are \( H = 15 \) arcs in the STG (Fig. 1). Accordingly, DST of \( P_{AC} Y(S_1) \) has 15 rows (Table 2).

The tables of LUTer1–LUTer2 are constructed in a trivial way. For example, the table of LUTer1 includes the rows of Table 2 with the transitions from states \( a_1, a_3, a_5, a_6 \) \( \in B^1 \). We do not show these tables in our example. The SBFs (20) and (21) can be derived from Table 2.

For example, the following SOPs can be obtained from Table 2 (after minimizing):

\[
\begin{align*}
D_1^1 &= T_1 T_2 x_1 \lor T_1 T_2 x_1 \lor T_1 T_2, \\
D_2^3 &= T_1 T_2 x_3 \lor T_1 T_2 x_3 \lor T_1 T_2, \\
v_1^1 &= T_1 T_2, \\
v_1^2 &= T_1 T_2 x_3 x_4 \lor T_1 T_2 x_3 x_4.
\end{align*}
\]

The LUTerF includes up to \( R_{\text{cr}} + R_V \) multiplexers.
6. Experimental results

This section includes experimental results. We compare the proposed approach ($P_{AC-Y}$ Mealy FSMs) with some known methods including $P_{AT-Y}$ FSMs. In the experiments, we use the library of standard benchmarks LGSynth93 (LGSynth93, 1993). Here there are 48 benchmarks of different complexity. The benchmarks are represented in the format KISS2. These benchmarks have a wide range of basic characteristics (numbers of states, inputs, and outputs). The characteristics of these benchmarks can be found in many articles and books, (e.g., Barkalov et al., 2022; 2020b). Thus we do not show them in this paper. The benchmarks are used very often by different researchers to compare the basic characteristics of FSM circuits obtained with different design methods.

The experiments were conducted using a personal computer with the following characteristics: Intel Core i7 6700 K 4.2@4.4 GHz, 16 GB RAM 2400 MHz CL 15 Memory. To implement LUT-based FSM circuits, we used the Virtex-7 VC709 Evaluation Platform (xc7vx690tfg1761-2) (AMD, 2023b). The FPGA chip of this platform includes slices having four LUTs with six inputs. The internal multiplexers of the slices can be used for implementing fast multiplexers from 4:1 to 16:1. The step of technology mapping was executed by the CAD tool Vivado v2019.1 (64-bit) (Vivado, 2023). The reports of Vivado give us the results of the experiments (LUT counts and maximum operating frequencies). The process of circuit implementation starts from the transformation of KISS2-based files into VHDL-based FSM models. To execute the transformation, our CAD tool K2F (Barkalov et al., 2020b) was used.

We compared area and time characteristics of FSM circuits obtained using the proposed method with four other approaches. Three of them are Mealy FSMs based on (i) Auto of Vivado (the value of R is selected from the interval $([\log_2 M], M)$), (ii) One-hot of Vivado, (iii) JEDI. The model of P Mealy FSM was used in these three cases. Also, we compared our approach with $P_{AT-Y}$ FSM, proposed by Barkalov et al. (2022).

Our previous research (Barkalov et al., 2021; 2022) shows that both the LUT count and maximum operating frequency depend strongly on the relation between the sum $L + R$, where the value of $R$ is determined by (1), on the one hand, and the number of inputs $S_L$, on the other. Thus, we divided the benchmarks into five classes. The benchmarks create the class of trivial FSMs (Class 0) if $R + L \leq 6$, the class of simple FSMs (Class 1) if $6 < R + L \leq 12$, the class of average FSMs (Class 2) if $12 < R + L \leq 18$, the class of big FSMs (Class 3) if $18 < R + L \leq 24$, the class of very big FSMs (Class 4) if $R + L > 24$. As previous research (Barkalov et al., 2021) shows, the larger the class number, the bigger the gain from using methods of structural decomposition.

For the library the LGSynth, the benchmarks are divided as follows. Class 0 consists of the benchmarks bbtas, dk17, dk27, dk512, ex3, ex5, bion, bion9, mc, modulo12, and shiftryg. Class 1 includes the benchmarks bbara, bsse, beecount, cse, dk14, dk15, dk16, donfile, ex2, ex4, ex6, ex7, keyb, mark1, opus, s27, s386, s840, and sse. Class 2 contains the benchmarks ex1, kirkman, planet, planet1, pma, s1, s1488, s1494, s1a, s208, stygr, and tma. Class 3 is created by a single benchmark sand. At last, the benchmarks s420, s510, s820, and s832 form Class 4.

The results of experiments are shown in Table 3 (the LUT counts) and Table 4 (the maximum operating frequency). These tables are organized in the same manner. The columns are marked by the names of the investigated methods. The names of the benchmarks are shown in the rows. There are results of summation of values from the columns in the row “Total”. The row “%” includes the percentage of summarized characteristics of FSM circuits produced by other methods with present to $P_{AC-Y}$-based FSMs. The following conclusions can be made on the basis of the experimental results.

As from Table 3, the circuits of $P_{AC-Y}$-based FSMs use a minimum number of LUTs compared with the other investigated methods. There is the following gain in the LUT counts: (i) 64.81% compared with Auto-based FSMs; (ii) 91.8% compared with FSMs based on one-hot state assignment; (iii) 35.73% compared with JEDI-based FSMs; (iv) 17.96% compared with $P_{AT-Y}$-based FSMs.

As from Table 4, our approach allows obtaining FSM circuits with the highest operating frequency. There is the following gain in the operating frequency: (i) 10.49% compared with Auto-based FSMs; (ii) 11.21% compared with FSMs based on one-hot state assignment, (iii) 3.07% compared with JEDI-based FSMs; (iv) 2.17% compared with $P_{AT-Y}$-based FSMs.

We think that the improvement in performance related to $P_{AT-Y}$-based FSMs is associated with a decrease in the number of interconnects. This number is decreased due to the following factors. Firstly, as follows from (26)–(28), there is a significant reduction in the number of interconnects. This number is decreased due to the following factors. Firstly, as follows from (26)–(28), there is a significant reduction in the number of interconnects.

As from the following experiments, there is a significant reduction in the number of interconnects. The following conclusions can be made on the basis of the experimental results.

The results of experiments are shown in Table 3 (the LUT counts) and Table 4 (the maximum operating frequency). These tables are organized in the same manner. The columns are marked by the names of the investigated methods. The names of the benchmarks are shown in the rows. There are results of summation of values from the columns in the row “Total”. The row “%” includes the percentage of summarized characteristics of FSM circuits produced by other methods with present to $P_{AC-Y}$-based FSMs. The following conclusions can be made on the basis of the experimental results.

As from Table 3, the circuits of $P_{AC-Y}$-based FSMs use a minimum number of LUTs compared with the other investigated methods. There is the following gain in the LUT counts: (i) 64.81% compared with Auto-based FSMs; (ii) 91.8% compared with FSMs based on one-hot state assignment; (iii) 35.73% compared with JEDI-based FSMs; (iv) 17.96% compared with $P_{AT-Y}$-based FSMs.

As follows from Table 4, our approach allows obtaining FSM circuits with the highest operating frequency. There is the following gain in the operating frequency: (i) 10.49% compared with Auto-based FSMs; (ii) 11.21% compared with FSMs based on one-hot state assignment, (iii) 3.07% compared with JEDI-based FSMs; (iv) 2.17% compared with $P_{AT-Y}$-based FSMs.

We think that the improvement in performance related to $P_{AT-Y}$-based FSMs is associated with a decrease in the number of interconnects. This number is decreased due to the following factors. Firstly, as follows from (26)–(28), there is a significant reduction in the number of interconnects. The following conclusions can be made on the basis of the experimental results.
dominate logic delay. Our approach allows reducing the number of interconnections. Therefore, this reduction of interconnections is responsible for the increase in the value of the maximum operating frequency.

As follows from Tables 3 and 4, using the combined state codes allows obtaining FSM circuits with better area (the LUT count) and time (the maximum operating frequency) characteristics compared with the equivalent \( P_{AT} \times Y \) FSMs. Also, our approach produces better results compared with those obtained using the known methods.

### Table 3. Results of the experiments (the LUT count).

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Auto</th>
<th>One-Hot</th>
<th>JEDI</th>
<th>TCAD21</th>
<th>Our appr. Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>bbtas</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>dk17</td>
<td>12</td>
<td>5</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>dk27</td>
<td>3</td>
<td>4</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>dk512</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>ex3</td>
<td>9</td>
<td>9</td>
<td>11</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>ex5</td>
<td>9</td>
<td>9</td>
<td>11</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>lion</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>lion9</td>
<td>6</td>
<td>11</td>
<td>5</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>mc</td>
<td>4</td>
<td>7</td>
<td>4</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>modulo12</td>
<td>7</td>
<td>7</td>
<td>11</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>shifreg</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table 4. Results of the experiments (the maximum operating frequency, MHz).

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Auto</th>
<th>One-Hot</th>
<th>JEDI</th>
<th>TCAD21</th>
<th>Our appr. Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>bbtas</td>
<td>204</td>
<td>16</td>
<td>204</td>
<td>16</td>
<td>204,12</td>
</tr>
<tr>
<td>dk17</td>
<td>199</td>
<td>28</td>
<td>167</td>
<td>0</td>
<td>199,38</td>
</tr>
<tr>
<td>dk27</td>
<td>206</td>
<td>02</td>
<td>201</td>
<td>9</td>
<td>204,18</td>
</tr>
<tr>
<td>dk512</td>
<td>196</td>
<td>27</td>
<td>196</td>
<td>27</td>
<td>197,75</td>
</tr>
<tr>
<td>ex3</td>
<td>194</td>
<td>86</td>
<td>194</td>
<td>86</td>
<td>195,76</td>
</tr>
<tr>
<td>ex5</td>
<td>185</td>
<td>20</td>
<td>185</td>
<td>21</td>
<td>182,16</td>
</tr>
<tr>
<td>lion</td>
<td>202</td>
<td>43</td>
<td>200</td>
<td>6</td>
<td>201,96</td>
</tr>
<tr>
<td>lion9</td>
<td>205</td>
<td>3</td>
<td>185</td>
<td>22</td>
<td>203,68</td>
</tr>
<tr>
<td>mc</td>
<td>196</td>
<td>66</td>
<td>195</td>
<td>47</td>
<td>181,03</td>
</tr>
<tr>
<td>modulo12</td>
<td>207</td>
<td>00</td>
<td>207</td>
<td>01</td>
<td>13,799</td>
</tr>
<tr>
<td>shifreg</td>
<td>262</td>
<td>67</td>
<td>263</td>
<td>57</td>
<td>276,26</td>
</tr>
</tbody>
</table>

### Reducing the number of LUTs for Mealy FSMs with state transformation

\[ \text{LUT count} = \frac{\text{Total LUTs}}{n} \]

where \( n \) is the number of states in the FSM.
of state assignment (Auto, One-hot, and JEDI) and functional decomposition. In consequence, the proposed approach has a rather good potential and can be used in CAD tools targeting LUT-based FSM synthesis.

7. Conclusion

A characteristic trend of our time is the widespread application of various VLSI chips for implementing digital systems. An important problem associated with VLSI-based design is that of reducing the area occupied by the digital system circuit. This fully applies to the VLSI-based synthesis of FSM circuits. In this case, it is necessary to reduce the numbers of arguments in the sum-of-products of Boolean functions representing an FSM circuit. As a rule, the required chip area is estimated as an LUT count of a particular circuit (Islam et al., 2020). Thus, to reduce the required chip area, it is necessary to diminish the LUT count in an FSM logic circuit.

Structural decomposition (Barkalov et al., 2021) is one of the efficient ways to solve this problem. Various methods of structural decomposition are connected with a change in the FSM structural diagram compared with single-level P Mealy FSMs. Of particular interest are methods that can simultaneously reduce the LUT count and increase the maximum operating frequency. Such a method is proposed in this paper.

The developed approach is aimed at improving the characteristics of LUT-based $P_{AT}Y$ Mealy FSMs with the transformation of collections of outputs into extended state codes. Our method is based on using combined state codes having two parts. One is a code of some class including a particular state. The other represents this state as a class element. Due to this approach, we eliminated the code transformer used in $P_{AT}Y$ Mealy FSMs. Also, the number of state variables is significantly reduced compared with equivalent $P_{AT}Y$ Mealy FSMs. As a result, we have the improvement in important characteristics such as the LUT count (on the average, by 17.96%) and maximum operating frequency (on the average, by 2.17%).

The results of our experiments show that the proposed approach allows a reduction in LUT counts in FSM circuits. Moreover, this improvement does not lead to a decrease in the FSM performance. Consequently, this method improves two main characteristics of FSM circuits. We think that it can be used in CAD tools aimed at LUT-based implementations of Mealy FSMs.

References


Reducing the number of LUTs for Mealy FSMs with state transformation


Alexander A. Barkalov worked at Donetsk National Technical University (DNTU) from 1976 till 1996 as a tutor. He then cooperated actively with the Kiev Institute of Cybernetics (IC) named after Victor Glushkov. He obtained his degree of a doctor of technical sciences (informatics) in 1995 from the IC. From 1996 till 2003 he worked as a professor of DNTU. Since 2003 he has been working as a professor at the Faculty of Computer, Electrical and Control Engineering of the University of Zielona Góra.

Larysa Titarenko obtained her degree of a doctor of technical sciences (telecommunications) in 2005 from the Kharkov National University of Radioelectronics (KNURE). Till 2003 she worked as a professor of the KNURE. Since 2005 she has been working as a professor at the Faculty of Computer, Electrical and Control Engineering of the University of Zielona Góra.

Kamil Mielcarek received his MSc degree in computer engineering from the Technical University of Zielona Góra, Poland, in 1995 and his PhD degree in computer science from 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 synthesis and optimization of control units in field-programmable logic devices, hardware description languages, perfect graphs and Petri nets, as well as algorithmic theory and safety of UNIX and network systems.

Received: 28 April 2023
Revised: 16 October 2023
Accepted: 14 November 2023