# State Assignment of Finite-State Machines by Using the Values of Input Variables

Valery Salauyou, Michal Ostapczuk

# State assignment of finite-state machines by using the values of input variables

Valery Salauyou, Michal Ostapczuk

Faculty of Computer Science, Bialystok University of Technology, Bialystok, Poland

valsol@mail.ru

**Abstract.** In this paper, we propose the method of FSM synthesis on field programmable gate arrays (FPGAs) when input variables are used for state assignment. For this purpose we offer a combined structural model of class A and class E FSMs. This paper also describes in detail the algorithms for synthesis a class AE FSM which consists of splitting of internal states for performance of necessary conditions for synthesis of the class E FSM and state assignment of the class AE FSM. It is shown that the proposed method reduces the area for all families of FPGAs by a factor of 1.19–1.39 on average and by a factor of three for certain families. Practical issues concerning the method and the specific features of its use are discussed, and possible directions of the elaboration of this approach are proposed.

**Keywords:** finite state machine (FSM), field programmable gate array (FPGA), state assignment, area minimization, state splitting, look up table

## 1 Introduction

In the general case, a digital system can be represented by a set of combinational circuits and finite state machines (FSMs). FSMs are also widely used as individual units as controllers and control devices. Usually, when working on a project, the designer has to develop new FSMs each time. It is clear that the parameters of FSMs used in a digital system to a large extent determine the success of the whole project. For this reason, the issue of minimization of FSMs is very important. As the FSM optimization criteria, one typically uses area, delay, and power consumption. Presently, field programmable gate arrays (FPGAs) are widely used in digital systems; for this reason, many FSM optimization methods are designed for the implementation of FSMs based in FPGAs.

The idea of using the values of the input and output variables of the FSM for encoding its internal states was first proposed in [1]. Later, this approach was elaborated in [2], where various combinations of the input and output variables that can be used for encoding the internal states are considered. The choice of the minimum number of input and output variables for encoding is an NP-hard problem. In [3], it was pro-

posed to use the values of the output variables of the Moor FSM as the codes of the internal states.

In [4], structural models of FSMs based on the architectural capabilities of FPGAs were proposed; these models make it possible to use the values of the FSM input and output variables as internal state codes. A new classification of structural models of FSMs is given. Here the class A and B FSMs are traditional of Mealy and Moore FSMs accordingly. In the class C (Mealy) and the class D (Moore) FSMs the value of an output vector completely coincides with the code of the present state of the FSM. In the class E (Mealy) and class F (Moore) FSMs the value of an input vector completely coincides with the code of the next state of the FSM.

In [5], the synthesis method of Mealy FSMs was proposed, where the values of output variables are used as the codes of FSM states. In this paper, we present the method of FSM synthesis which allows, unlike [5], to use the values of input variables as the codes of FSM states. For this purpose, the structural model of the class E FSM [4] is used.

## 2 Structural FSM models

The most general model of the Mealy FSM can be described by means of following equations:

$$a_{t+1} = \Phi(z_t, a_t);$$
$$w_t = \Psi(z_t, a_t),$$

where $\Phi$ is the transition function, $\Psi$ is the output function, $a_t$ is the present state of the FSM at time $t$ ($t$=1,2,3,...), $a_{t+1}$ is the next state of the FSM, $z_t$ is a collection of values of the input variables (the input vector) on the FSM input at time $t$, and $w_t$ is a collection of values of the output variables (the output vector) formed at time $t$.

The Mealy FSM in classification [4] received a title the *class A FSM*. The structural model of the Mealy FSM show on fig. 1,*a*, where $CL_\Phi$ is the combinational circuit forming the values of the transition functions, $CL_\Psi$ is the combinational circuit forming the values of the output functions, and RG is the FSM's memory.

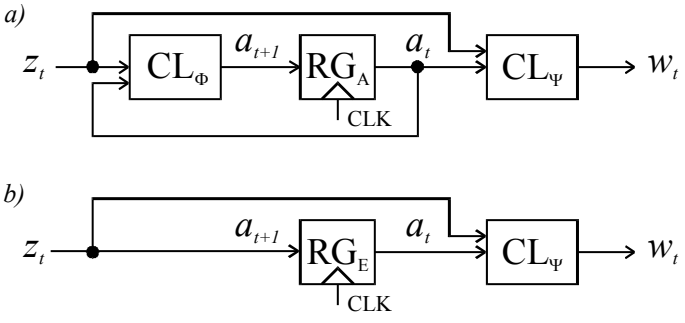


**Fig. 1.** The structural models of FSMs: *a* – the class A FSM; *b* – the class E FSM

In the *class E FSM* the value of the input vector $z_t$ determines the code of the next state $a_{t+1}$; therefore, the equations of functioning of the class E FSM have the following view:

$$a_{t+1} = z_t;$$
$$w_t = \Psi(z_t, a_t),$$

In contrast to the Mealy FSM, the structure of the class E FSM does not include the combinational circuit $CL_\Phi$ (fig. 1,*b*) that allows to build FSMs of a low cost (an area) and a high-speed performance.

However in practice, the "pure" type of the class E FSM meets very rarely. Therefore, in the present work we offer the combined model of the *class AE FSM*. In the class AE FSM the codes of internal states are divided on two parts: one part is defined by the value of input variables, and the second part is formed the same as in the class A FSM, by means of combinational circuit $CL_\Phi$ (fig. 2).

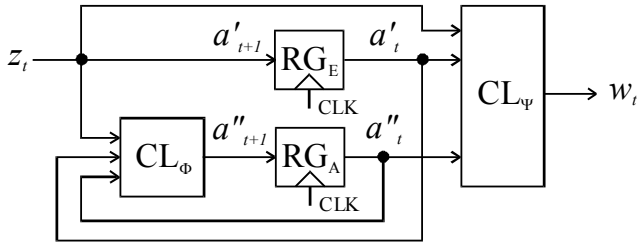


**Fig. 2.** The structural model of the class AE FSM

The memory of the class AE FSM is presented by two registers $RG_A$ and $RG_E$ which correspond to the class A and the class E FSMs. The internal state $a_t$ of the class AE FSM is defined by a concatenation of two states: $a'_t$ and $a''_t$, i.e. $a_t = \{a'_t, a''_t\}$, where $a'_t$ is the value on outputs of the register $RG_E$, and $a''_t$ is the value on outputs of the register $RG_A$. The code of the next state $a_{t+1}$ also is defined by the concatenation of two states: $a'_{t+1}$ and $a''_{t+1}$, i.e. $a_{t+1} = \{a'_{t+1}, a''_{t+1}\}$. Here the code of the state $a'_{t+1}$ coincides with the value of the input vector $z_t$, and the code of the state $a''_{t+1}$ is formed by the combinational circuit $CL_\Phi$ on the basis of the input vector $z_t$ and the code of the state $a_t$. Functioning of the class AE FSM can be described by means of following equations:

$$a'_{t+1} = z_t;$$
$$a''_{t+1} = \Phi(z_t, a_t);$$
$$w_t = \Psi(z_t, a_t).$$

## 3 The synthesis method of the class AE FSM

Let us present the class AE FSM by the structural diagram on fig. 3 that consists of two registers $RG_A$ and $RG_E$, and the combinational circuit CL. Here, in contrast to the structure on fig. 2, the combinational circuits $CL_\Phi$ and $CL_\Psi$ are united in one circuit C L. The combinational circuit CL receives the values of the input variables $X = \{x_1,\ldots,x_L\}$, the values of the variables $G = \{g_1,\ldots,g_L\}$, which define the state $a'_t$ code, and the values of the feedback variables $E = \{e_1,\ldots,e_R\}$, which define the state $a''_t$ code. Based on FSM algorithm functioning and arriving the values of the variables, the combinational circuit CL forms values of the output functions $Y = \{y_1,\ldots,y_N\}$ and the transition functions $D = \{d_1,\ldots,d_R\}$ which define the state $a''_{t+1}$ code.

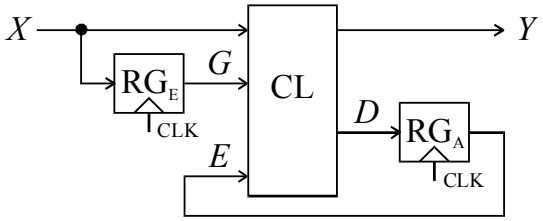


**Fig. 3.** The structural diagram of the class AE FSM

Let $A$ is a set of internal states of the FSM. Let us denote by $U(a_i)$ the set of all the transition conditions in the state $a_i$, $a_i \in A$:

$$U(a_i) = \{X(a_m,a_i) \mid a_m \in B(a_i)\}, \tag{1}$$

where $X(a_m,a_i)$ is a collection of values of the input variables (a transition condition or an input vector) that initiates the transition from the state $a_m$ to the state $a_i$, $B(a_i)$ is the set of states, transitions from which terminate in the state $a_i$.

The state $a_i$, $a_i \in A$, is a state of the class E FSM, i.e. $a_i \in A_E$, if next conditions are satisfied:

$$| U(a_i) | = 1 \tag{2}$$

$$U(a_i) \cap U(a_j) = \varnothing \text{ at } i \neq j \text{ for all } a_j \in A, \tag{3}$$

where $|M|$ is the cardinality (the number of elements) of the set $M$, $\varnothing$ is an empty set.

A performance of the condition (2) guarantees that all transitions to the state $a_i$ are carried out by the same transition condition (i.e. the state $a_i$ has only one code), and a performance of the condition (3) provides a determinacy of a FSM behavior (i.e. the transition condition in some state $a_i$ does not initiate passages to other FSM states).

The FSM is the class E FSM if all its states are states of the class E FSM, i.e. $A_E = A$. In other words, the finite state machine is the class E FSM if for all its states conditions (2) and (3) are satisfied.

The satisfaction of the conditions (2) is carried out by splitting of FSM states. Let, for example, for some state $a_i$, $a_i \in A$, takes place $|U(a_i)| = Q > 1$. It is possible to split

a state $a_i$ on states $a_{i\_1},...,a_{i\_Q}$ so that the transitions in each state $a_{i\_q}$ were defined only by one transition condition, i.e. was fulfilled $|U(a_{i\_q})| = 1$, $q = \overline{1,Q}$. Now instead of one state $a_i$, we have $Q$ states for which conditions (2) are satisfied.

In case of violations of the conditions (3), the synthesis of the class E FSM is impossible, since the determinacy of the FSM behavior is broken. In this case, it is offered to use the combined model of the class AE FSM (fig. 2). For this purpose, the second part of the code of the class A FSM is added to the code of each internal state of the class E FSM, which corresponds to the states $a''_t$ and $a''_{t+1}$. The last is carried out by special state assignment of the FSM.

## 4 Splitting of internal states for performance of necessary conditions for synthesis of the class E FSM

Note that splitting the internal states is an equivalent transformation of the FSM and it does not change the operation algorithm of the FSM. Let $M$ be the number of internal states of the FSM, $P(a_i)$ be the set of transitions of the FSM from the state $a_i$, $i = \overline{1,M}$, $C(a_i)$ be the set of transitions of the FSM that terminate in the state $a_i$, $a_i \in A$, and $X(a_i)$ be some a transition condition to the state $a_i$, $X(a_i) \in U(a_i)$. Then the algorithm splitting of internal states for performance of necessary conditions for synthesis of the class E FSM has the following view.

**Algorithm 1**.

1. For each internal state $a_i$, $a_i \in A$, according to (1) the set $U(a_i)$ is defined.
2. In the set $A$, find a state $a_i$ for which condition (2) are not satisfied. If such a state is found, then go to Step 3; otherwise, go to Step 7.
3. Put $Q := |U(a_i)|$. Introduce $Q$ new states $a_{i\_1},...,a_{i\_Q}$.
4. Determine the subsets $C(a_{i\_1}),...,C(a_{i\_Q})$ of transitions to the states $a_{i\_1},...,a_{i\_Q}$. Each subset $C(a_{i\_q})$ is assigned transitions which is initiated by the condition $X^q(a_m,a_i) \in U(a_i)$, $a_m \in B(a_i)$, $q = \overline{1,Q}$.
5. The subsets $P(a_{i\_1}),...,P(a_{i\_Q})$ of transitions from the states $a_{i\_1},...,a_{i\_Q}$ are determinated in the following way: $P(a_{i\_q}) := P(a_i)$ for all $q = \overline{1,Q}$.
6. Put $A := A \setminus \{a_i\}$, $A := A \cup \{a_{i\_1},...,a_{i\_Q}\}$, and $M := M+Q-1$; go to Step 2.
7. Stop.

## 5 State assignment of the class AE FSM

The main purpose of encoding the internal states when designing the class AE FSMs is to ensure the mutual orthogonality of these codes. To encode the internal states of a class AE FSM, a ternary matrix $W$ is constructed in which the rows correspond to the internal states and the columns correspond to the variables of the set $G$. A unit is put on intersection of the row $i$ and the column $j$ of the matrix $W$, if the input variable $x_j$ has the value 1 in the condition $X(a_i)$, a zero, if the input variable $x_j$ has the value 0 in the condition $X(a_i)$, and an undetermined value (the dash), if the variable $x_j$ does not

influence on the transition condition $X(a_i)$. Later, the rows of the matrix $W$ will determine the codes of the internal states of the class AE FSM.

To make the codes of internal states of the class AE FSM orthogonal it is necessary to solve the following task.

**Task 1**. To add in matrix $W$ the minimum number of the columns, which corresponding to the variables $e_1,...,e_R$, and to encode the rows of the matrix $W$ by binary values of the variables $e_1,...,e_R$ so that all the rows of the matrix $W$ were mutually orthogonal.

In order to solve the Task 1 and to encode the internal state of the class AE FSM the following algorithm is offered.

**Algorithm 2**.

1. The graph $H$ for the orthogonalization of the rows of the matrix $W$ is constructed. The vertices of $H$ correspond to the rows of $W$ (internal states of the FSM). Two vertices of $H$ are connected by an edge if the corresponding rows of $W$ are orthogonal.
2. The vertices connected to all other vertices (the rows of $W$ corresponding to these vertices are orthogonal to all other rows) are removed from $H$.
3. The graph $H$ is decomposed into the minimum number of complete subgraphs $H_1,...,H_T$ using Algorithm 3.
4. The subgraphs $H_1,...,H_T$ are encoded by binary codes of the minimum length $R = intlog_2T$ using Algorithm 5.
5. $R$ columns that correspond to the variables $e_1,...,e_R$ of the codes of the subgraphs $H_1,...,H_T$ are added to the matrix $W$. In row $i$ of $W$, the positions of the additional columns are filled by the code of the subgraph $H_t$, $t=\overline{1,T}$, containing the vertex $a_i$, $i=\overline{1,M}$. The other positions of the additional columns in $W$ are filled by zeros.
6. The contents of the row $i$ of $W$ is used as the code of the internal state $a_i$, $i=\overline{1,M}$.
7. Stop.

The decomposition of the graph $H$ into the minimum number of complete graphs $H_1,...,H_T$ (at Step 3 of Algorithm 2) is made by the following algorithm.

**Algorithm 3.**

1. Set $T := 0$.
2. Set $T := T + 1$. In the graph $H$, find a complete graph $H_T$ with the maximum number of vertices.
3. Remove the vertices of $H_T$ from the graph $H$.
4. If the set of vertices of $H$ is not empty, the go to Step 2; otherwise, go to Step 5.
5. Stop.

The maximal complete subgraph $H_t$, $t=\overline{1,T}$, at Step 2 of Algorithm 3 can be found using the following algorithm.

**Algorithm 4.**

1. Find a vertex $a_i$ in $H$ with the greatest local degree.
2. Include $a_i$ into the graph $Ht$.

3. Among all the vertices of $H$ not included in $H_t$, find a node $a_i$ connected to all the nodes of the subgraph $H_t$. If several such nodes are found, choose a node with the greatest local degree among them.
4. If a vertex connected to all the vertices of the subgraph $H_t$ was found at Step 3, then go to Step 2; otherwise, go to Step 5.
5. Stop.

To encode the subgraphs $H_1,\ldots,H_T$ (Step 4 of Algorithm 2) the following algorithm is used to minimize the area of implementing the transition functions.

**Algorithm 5.**

1. Calculate the length $R$ of the codes of the subgraphs $H_1,\ldots,H_T$: $R = intlog_2T$.
2. Form the set $K$ of binary codes of length $R$.
3. The subgraph containing the initial state $a_1$ is encoded by the zero code from $K$.
4. If all the subgraphs $H_1,\ldots,H_T$ are encoded, then go to Step 5; otherwise, find among the not yet encoded subgraphs $H_1,\ldots,H_T$ a subgraph $H_t$ for which $\sum|C(a_i)|$=max for all $a_i \in H_t$.
   To encode the subgraph $H_t$, the code with the minimum number of unities is chosen in the set $K$. Go to Step 4.
5. Stop.

**Example.** Let us apply the proposed method for designing the FSM described by the state diagram shown in Fig. 4. The vertices correspond to the internal states $a_1,\ldots,a_5$ of this FSM, and the arcs correspond to the FSM transitions. Beside each arc, the value of the input vector that initiates the transition and, separated by a slash, the value of the output vector that formed on this transition are indicated. In this example, the FSM has five states, three input variables, and two output variables.

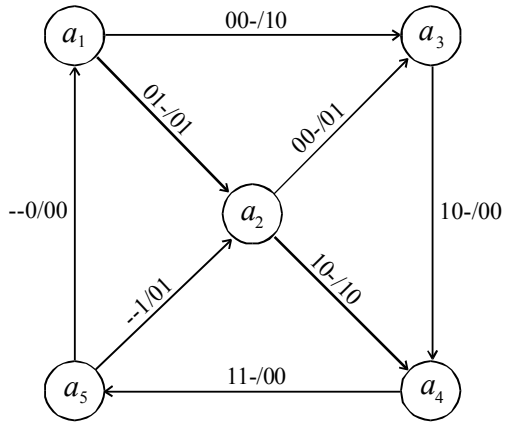


**Fig. 4.** The state diagram of the initial FSM

In this example, conditions (2) are violated for the state $a_2$ because $U(a_2)$ = {01-,--1}, i.e. $|U(a_2)| = 2$, therefore, $a_2$ is split into two states $a_{2\_1}$ and $a_{2\_2}$. The state diagram of the FSM obtained upon splitting the state $a_2$ is shown in Fig. 5.

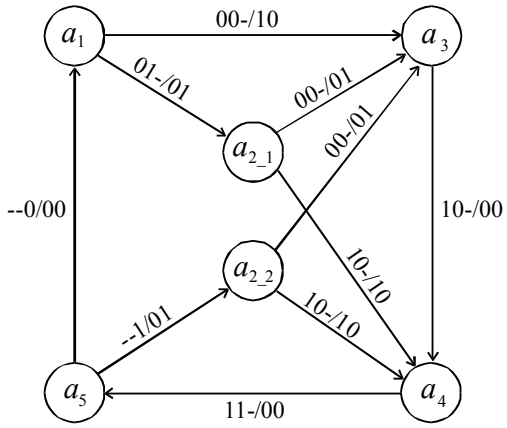


**Fig. 5.** The state diagram of the FSM after splitting the state $a_2$

The encoding of the internal states begins with constructing the matrix *W* (Table 1). Figure 5 shows the orthogonality graph *H* of the rows of *W*. The graph *H* is decomposed into two complete subgraphs $H_1$ and $H_2$. As the number of subgraphs *T* is equal 2, then one variable $e_1$ has enough for coding of two subgraphs. According to Step 4 of algorithm 5, the subgraph $H_1$ is encoded by binary code "1", and the subgraph $H_2$ is encoded by the code "0". The matrix *W* with an additional column $e_1$ for orthogonalization of rows is resulted in Table 2.

**Table 1.** The matrix *W* for state assignment of a class AE FSM

| | $g_1$ | $g_2$ | $g_3$ |
|---|---|---|---|
| $a_1$ | - | - | 0 |
| $a_{2\_1}$ | 0 | 1 | - |
| $a_{2\_2}$ | - | - | 1 |
| $a_3$ | 0 | 0 | - |
| $a_4$ | 1 | 0 | - |
| $a_5$ | 1 | 1 | - |

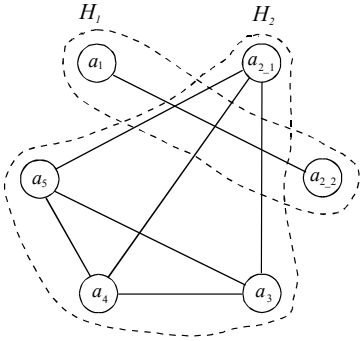


**Fig. 6.** The graph *H* for orthogonality of rows of the matrix *W*

**Table 2.** The matrix *W* after orthogonalization of rows

| | $g_1$ | $g_2$ | $g_3$ | $e_1$ |
|---|---|---|---|---|
| $a_1$ | - | - | 0 | 1 |
| $a_{2\_1}$ | 0 | 1 | - | 0 |
| $a_{2\_2}$ | - | - | 1 | 1 |
| $a_3$ | 0 | 0 | - | 0 |
| $a_4$ | 1 | 0 | - | 0 |
| $a_5$ | 1 | 1 | - | 0 |

The logical equations implemented by combinative circuit CL (fig. 3) for our example have the following view:

$$d_1 = g_1 g_2 \bar{e}_1;$$
$$y_1 = \bar{g}_3 e_1 \bar{x}_1 \bar{x}_2 + \bar{g}_1 g_2 \bar{e}_1 x_1 \bar{x}_2 + g_3 e_1 x_1 \bar{x}_2;$$
$$y_2 = \bar{g}_3 e_1 \bar{x}_1 x_2 + \bar{g}_1 g_2 \bar{e}_1 \bar{x}_1 \bar{x}_2 + g_3 e_1 \bar{x}_1 \bar{x}_2 + g_1 g_2 \bar{e}_1 x_3.$$

As it is possible to see from the resulted equations, the transition functions for the class AE FSM are very simple. The implementation cost of the given system of Boolean functions (it is traditionally defined as the number of inputs of the necessary for implementation gates) is equal 40, while at traditional implementation of the class A FSM, the implementation cost is equal 74. Thus, for the considered example usage of structural model of the class AE FSM, in comparison with the traditional approach, allowed to reduce the implementation cost by a factor of 1.85 or by 54 %.

The efficiency of the proposed method for designing the class AE FSMs was tested by implementing the FSM described in the example above in the FPGAs manufactured by Altera using the CAD tools Quartus Prime version 16.0. It is the synthesis parameters by default of CAD Quartus were used. The initial FSM of class A and the synthesized FSM of class AE have been described in language Verilog.

**Table 3.** Results of experimental researches at implementation on FPGA of the FSMs from the considered example

| **FPGA** | $LE_A$ | $F_{maxA}$ | $LE_{AE}$ | $F_{maxAE}$ | $LE_A/LE_{AE}$ | $F_{maxAE}/F_{maxA}$ |
|---|---|---|---|---|---|---|
| MAX II | 12 | 517.06 | 10 | 417.19 | 1.20 | 0.807 |
| MAX V | 12 | 261.57 | 10 | 260.15 | 1.20 | 0.996 |
| MAX 10 | 13 | 592.42 | 11 | 581.40 | 1.18 | 0.981 |
| Arria II | 8 | 1025.64 | 4 | 1022.49 | 2.00 | 0.997 |
| Cyclone IV E | 12 | 738.01 | 10 | 801.92 | 1.20 | 1.085 |
| Cyclone IV GX | 12 | 726.74 | 10 | 727.27 | 1.20 | 1.001 |
| Cyclone V | 6 | 640.20 | 4 | 640.20 | 1.50 | 1.000 |
| ***mid*** | | | | | 1.35 | 0.981 |

Tables 3 show the results of experiments for various FPGA families manufactured by the companies Altera, where $LE_A$ and $LE_{AE}$ are the numbers of the logical elements

(functional generators LUT — look-up table) used in the implementation of the class A FSM and the class AE FSM, $F_{\max A}$ and $F_{\max AE}$ are the maximum operation frequencies (in MHz) of these FSMs, $LE_A/LE_{AE}$ and $F_{maxAE}/F_{maxA}$ are the ratio of the corresponding parameters, and *mid* is the mean value of the parameter. The data in Tables 3 show that the proposed method for designing the class AE FSM reduced the implementation area of the FSM from the example on the Altera FPGAs by a factor of 1.35 on average and by a factor of 3.00 for the Arria II family. Thus the maximum operation frequency of the class AE FSM concedes to the maximum operation frequency of the class A FSM a little.

## 6 Experimental study

The synthesis method of the class AE FSM was researched at implementation on FPGA for the FSM benchmarks MCNC [6]. For this purpose to each benchmark of the FSM the considered synthesis method was applied. Both finite state machines, the initial class A FSM and synthesized the class AE FSM, were described in language Verilog. Then standard implementation on FPGA of FSMs by means of CAD Quartus II version 13.1 was fulfilled. It is the synthesis parameters by default of CAD Quartus were used. As criteria of optimization the implementation cost ($C$), defined by the number of used logical elements LUT, and the maximum operation frequency ($F$) was considered.

In Table 4 and 5, the parameter relations are presented for eleven FSM benchmarks for which usage of the synthesis method of the class AE FSM is the most effective. Here relation $C_A/C_{AE}$ shows in how many time the synthesis method of the class AE FSM, in comparison with the class A FSM, improves the implementation cost; $F_{AE}/F_A$ – the frequency.

The analysis of Table 4 and 5 shows that the synthesis method of the class AE FSM allows to reduce a implementation cost by a factor of 3.00 for the benchmark shiftreg for all FPGA families, a time delay by a factor of 1.60 for the benchmark lion for family Cyclone, and an operation frequency by a factor of 2.93 for the benchmark train4 for family Stratix.

An average improving is of a cost by a factor from 1.19 (Cyclone, Stratix) to 1.39 (Arria, Stratix II, (III), of a time delay – from 0.97 (Cyclone II) to 1.05 (Cyclone), of a frequency – from 1.92 (Cyclone II) to 1.10 (Stratix).

**Table 4.** Results of experimental researches of the synthesis method of the class AE FSM for families Arria GX and Cyclone

| FSM | Arria GX | | Cyclone | | Cyclone II | | Cyclone III | |
|---|---|---|---|---|---|---|---|---|
| | $C_A/C_{AE}$ | $F_{AE}/F_A$ | $C_A/C_{AE}$ | $F_{AE}/F_A$ | $C_A/C_{AE}$ | $F_{AE}/F_A$ | $C_A/C_{AE}$ | $F_{AE}/F_A$ |
| dk15 | 0.86 | 1.55 | 0.71 | 1.45 | 0.71 | 1.18 | 0.71 | 1.06 |
| dk16 | 1.31 | 0.85 | 0.63 | 0.80 | 0.55 | 0.74 | 0.55 | 0.69 |
| dk17 | 1.67 | 1.06 | 0.96 | 0.70 | 0.96 | 0.91 | 0.96 | 1.12 |
| dk27 | 1.33 | 1.05 | 1.25 | 0.97 | 1.25 | 0.99 | 1.25 | 1.04 |
| dk512 | 1.42 | 0.82 | 0.68 | 0.91 | 0.78 | 0.82 | 0.78 | 0.77 |
| ex5 | 0.79 | 0.61 | 0.64 | 0.67 | 0.64 | 0.68 | 0.64 | 0.69 |
| lion | 1.00 | 0.65 | 0.71 | 0.54 | 0.77 | 0.63 | 0.77 | 0.64 |
| lion9 | 1.43 | 0.81 | 1.82 | 1.16 | 1.82 | 1.08 | 1.82 | 0.91 |
| shiftreg | 3.00 | 0.62 | 3.00 | 0.55 | 3.00 | 0.75 | 3.00 | 0.64 |
| train4 | 0.75 | 1.77 | 1.00 | 1.60 | 1.00 | 1.45 | 1.00 | 1.56 |
| train11 | 1.71 | 1.49 | 1.71 | 1.56 | 1.92 | 0.94 | 1.92 | 0.84 |
| ***mid*** | 1.39 | 1.03 | 1.19 | 0.99 | 1.22 | 0.92 | 1.22 | 0.91 |
| ***max*** | 3.00 | 1.77 | 3.00 | 1.60 | 3.00 | 1.45 | 3.00 | 1.56 |

**Table 5.** Results of experimental researches of the synthesis method of the class AE FSM for families MAX II and Stratix

| FSM | MAX II | | Stratix | | Stratix II | | Stratix III | |
|---|---|---|---|---|---|---|---|---|
| | $C_A/C_{AE}$ | $F_{AE}/F_A$ | $C_A/C_{AE}$ | $F_{AE}/F_A$ | $C_A/C_{AE}$ | $F_{AE}/F_A$ | $C_A/C_{AE}$ | $F_{AE}/F_A$ |
| dk15 | 0.70 | 1.18 | 0.71 | 1.07 | 0.86 | 1.02 | 0.86 | 0.66 |
| dk16 | 0.63 | 0.83 | 0.63 | 0.73 | 1.31 | 0.77 | 1.31 | 0.82 |
| dk17 | 1.39 | 0.98 | 0.96 | 0.75 | 1.67 | 1.21 | 1.67 | 1.02 |
| dk27 | 1.25 | 1.20 | 1.25 | 0.95 | 1.33 | 0.89 | 1.33 | 1.00 |
| dk512 | 0.70 | 0.89 | 0.68 | 0.98 | 1.42 | 1.03 | 1.42 | 0.78 |
| ex5 | 0.62 | 0.77 | 0.64 | 0.63 | 0.79 | 0.48 | 0.79 | 0.68 |
| lion | 0.71 | 0.77 | 0.71 | 0.84 | 1.00 | 0.66 | 1.00 | 0.54 |
| lion9 | 1.82 | 0.82 | 1.82 | 0.75 | 1.43 | 0.95 | 1.43 | 0.87 |
| shiftreg | 3.00 | 1.14 | 3.00 | 0.31 | 3.00 | 0.66 | 3.00 | 0.71 |
| train4 | 1.00 | 0.95 | 1.00 | 2.93 | 0.75 | 1.77 | 0.75 | 1.91 |
| train11 | 1.77 | 0.97 | 1.71 | 2.15 | 1.71 | 1.74 | 1.71 | 1.65 |
| ***mid*** | 1.24 | 0.95 | 1.19 | 1.10 | 1.39 | 1.02 | 1.39 | 0.97 |
| ***max*** | 3.00 | 1.20 | 3.00 | 2.93 | 3.00 | 1.77 | 3.00 | 1.91 |

## Conclusions

The considered method of synthesis of the class AE FSM showed the high efficiency at minimization of implementation cost of FSMs for various FPGA families, by a factor of 1.19–1.39 on average and by a factor of 3.00 for certain families. Besides, in certain cases the method allows to increase the FSM performance (by a factor of 2.93 for benchmark train4 for family Sratix). An application of the given method is the most effective for FSMs with the many of input variables, especially when the transitions in the various states are initiated by different transition conditions.

The proposed method for the minimization of FSMs based on the use of the structural model of the class AE FSM is universal because it is applicable to Mealy FSMs (i.e., to arbitrary FSMs), does not change the operational algorithm of the FSM, and is efficient for all FPGA families. Therefore, this method can be recommended for inclusion in industrial CAD tools in order to minimize the area of implementation. The given method can be used not only at implementation of FSMs on FPGA, but also on other an element basis, for example on ASIC (Application Specific Integrated Circuit). We see perspective a direction of the further researches when values of input and output variables of the FSM are shared for states assignment.

**Acknowledgements**. The present study was supported by a grant S/WI/1/2013 from Bialystok University of Technology and founded from the resources for research by Ministry of Science and Higher Education.

## References

1. McCluskey E. J.: Reduction of Feedback Loops in Sequential Circuits and Carry Leads in Iterative Networks. Information and Control. **2**, 99–118 (1963)
2. Pomeranz I., Cheng K.T.: STOIC: State Assignment Based on Output/Input Functions. IEEE Trans. on CAD. **8**, P. 613–622 (1993)
3. Forrest J.: ODE : Output Direct State Machine Encoding. In: European Design Automation Conference (EURO-DAC'95). pp. 600-605 Brighton. UK (1995)
4. Solovjev V.: Synthesis of Sequential Circuits on Programmable Logic Devices Based on New Models of Finite State Machines. In: Euromicro Symposium on Digital Systems Design (DSD'2001). pp. 170-173. Warsaw. Poland (2001)
5. Solov'ev V. V.: Minimization of Mealy Finite-State Machines by Using the Values of the Output Variables for State Assignment. Journal of Computer and Systems Sciences International. **1**, 96–104 (2017)
6. Yang S.: Logic synthesis and optimization benchmarks user guide. Version 3.0. Microelectronics Center of North Carolina (MCNC). North Carolina. USA (1991)