ISSN 1064-2307, Journal of Computer and Systems Sciences International, 2015, Vol. 54, No. 2, pp. 230–242. © Pleiades Publishing, Ltd., 2015.
Original Russian Text © A.S. Klimowicz, V.V. Solov'ev, 2015, published in Izvestiya Akademii Nauk. Teoriya i Sistemy Upravleniya, 2015, No. 2, pp. 68–80.

Structural Models of Finite-State Machines for Their Implementation on Programmable Logic Devices and Systems on Chip

A. S. Klimowicz and V. V. Solov'ev
Bialystok University of Technology, Bialystok, Poland
e-mail: valsol@mail.ru

Received March 11, 2014; in final form, August 22, 2014

Abstract—Digital systems based on programmable logic devices and systems on chip often use register buffers for the transmission of signals between functional units. In this paper, structural models of finite state machines (FSMs) that allow the use of input and output buffer triggers as memory elements of the finite state machine are considered. To this end, a new classification of structural models of finite state machines is proposed in which all finite state machines are divided into six classes A, B, C, D, E, and F. In the models C and D, output buffer triggers are used as memory elements; in the models E and F, input buffer triggers are used for this purpose. A definition of the set of internal states of FSMs in each class is given and synthesis methods for the FSMs of the classes C–F are proposed. The results of experiments showed that the use of FSM models of the classes C–F reduces the number of internal memory elements by 2.22% to 25.83% on the average and sometimes even by 100%.

DOI: 10.1134/S1064230715010074

INTRODUCTION

Due to the wide use programmable logic devices (PLDs), such as complex programmable logic devices (CPLD) and field programmable gate arrays (FPGAs) and for designing various sequential circuits in systems on chip (SoC), there is pressing need in finding new structural models of FSMs that effectively use the architectural capabilities of modern PLDs.

Presently, two models of FSMs—Mealy FSM [1] and Moore FSM [2]—are well studied. The architectures of modern PLDs typically include various memory units and distributed memory elements. For that reason, a large number of publications are devoted to the development of structural models of FSMs that are based or use FPGA memory. For example, in [3], structural models of FSMs implemented on memory arrays are proposed. In [4], the structure of FSMs based on FPGA memory is studied.

Some publications are devoted to hierarchical structural models of FSMs. For example, a structure for the implementation of hierarchical FSMs consisting of a combinational circuit, stack memory, and code transformer was considered in [5]. In [6], a structural model for the implementation of parallel hierarchical FSMs was proposed, which includes a combinational circuit and two stack memory units for FSMs and for internal states.

Some publications are devoted to the development of structural models of Moore FSMs as they are implemented on CPLDs and FPGAs. In [7], structural models for the implementation of Moore FSMs on CPLD macrocells were proposed. In [8], the structure of a microprogrammed Moore FSM implemented on logical Look Up Table (LUT) FPGA elements was proposed. In [9], the structure of the Moore FSMs implemented on FPGAs was considered and a method of their synthesis so as to minimize the number of logical LUT elements and memory units was proposed.

The structures of FSMs for the implementation of recursive computations were studied in [10–12]. In [10], the structure for the implementation of recursive algorithms for microprogrammed FSMs was proposed, which consists of a combinational circuit and two stack memory units—for FSMs and for modules. In [11], the general structure of an FSM for the implementation of recursive algorithms was described, which consists of a combinational circuit and three stack memory units—for modules, states, and data. In [12], two FSM structures (using explicit and implicit modules) for the implementation of recursive algorithms were considered.

There are some more interesting structural models of FSMs. For example, the structures of an asynchronous and pipeline asynchronous FSMs with enable signals for the time of internal state stabilization were described in [13]. In [14], the structure of a reconfigurable FSM which, in addition to two combinational circuits implementing the transition and output functions, contains three more combinational circuits was proposed. In [15], the structure of an FSM together with a data processing unit called FSM with data path (FSMD) was considered.

In PLD and SoC-based digital systems, input and output buffers (which are ordinary registers consisting of triggers) are often used to transmit data between units. The possibility to individually control each trigger or a group of triggers in the architectures of modern PLDs and SoCs makes it possible to use triggers of the input, output, and simultaneously input and output buffers of specific functional units as memory elements of FSMs. Since the input and output buffers are installed independently of the way the FSMs within the units are implemented, the implementation of FSM memory elements on input and (or) output buffer triggers does not require the use of additional PLD resources.

The possibility of using the values of input and output signals (vectors) as internal state codes in FSMs was studied in [16–18]. In [16], this problem was solved for completely determined FSMs and for FSMs with undetermined values of the input and output vectors. In [17], output signals of the Moore FSMs are used as a part of internal state codes; in [18], output vectors with undetermined values are used for this purpose.

In [19], structural models of PLD-based FSMs were proposed in which triggers at the outputs and in feedback loops of CPLD macrocells and FPGA logical elements employed as output buffer triggers are used as memory elements. In [20, 21], a classification of structural FSM models was proposed in which six basic and four combined classes were distinguished. In these models, triggers of the input and output buffers are used as memory elements. In [22], combined structural FSM models are considered. The book [23] is devoted to the synthesis of basic and combined structural models of PLD-based FSMs. In [24, 25], the combined model of Moore and Mealy FSMs is studied.

In the present paper, we investigate various structural FSM models in which triggers of the input and output buffers can be used as memory elements. A definition of the internal states of each class of these FSMs is given for the first time. An algorithm for the synthesis of class *C* FSMs is described and used to design methods for the synthesis of FSMs belonging to the classes *D*, *E*, and *F*. Experiments are aimed at finding out if it is possible to reduce the number of internal states for the FSMs of each class.

1. CLASSES OF FINITE STATE MACHINES

In engineering practice, two types of FSMs received widespread use—the Moore FSMs and the Mealy FSMs. The behavior of the Moore FSM is described by the equations

$$a_{t+1} = \varphi(z_t, a_t),$$
$$w_t = \psi(z_t, a_t),$$

where φ is the transition function, ψ is the output function, *t* are the machine's times ($t = 1, 2, 3, \ldots$), z_t is the input vector at the time *t*, w_t is the produced output vector, a_t is the current state of the FSM, and a_{t+1} is its next state.

The behavior of the Mealy FSM is described by the equations

$$a_{t+1} = \varphi(z_t, a_t),$$
$$w_t = \psi(a_t).$$

Structural models of Moore and Mealy FSMs are shown in Figs. 1a and 1b, where the combinational circuit CL_Φ implements the transition function, the combinational circuit CL_ψ implements the output function, the register RG implements the FSM memory, and CLK is the clock signal.

If each output vector w_t of a Moore FSM coincides with the code of its internal state a_t, then its behavior can be described by the equations

$$a_{t+1} = \varphi(z_t, a_t),$$
$$w_t = a_t.$$

The type of FSM is said to belong to the class *C* [26]. Similarly, the Mealy and Moore FSMs form the classes *A* and *B*, respectively. The structure of the class *C* FSM is shown in Fig. 1c. Its distinctive feature is the absence of the combinational circuit CL_ψ and the fact that all the outputs of the FSM are register outputs because they are formed at the outputs of the register RG.

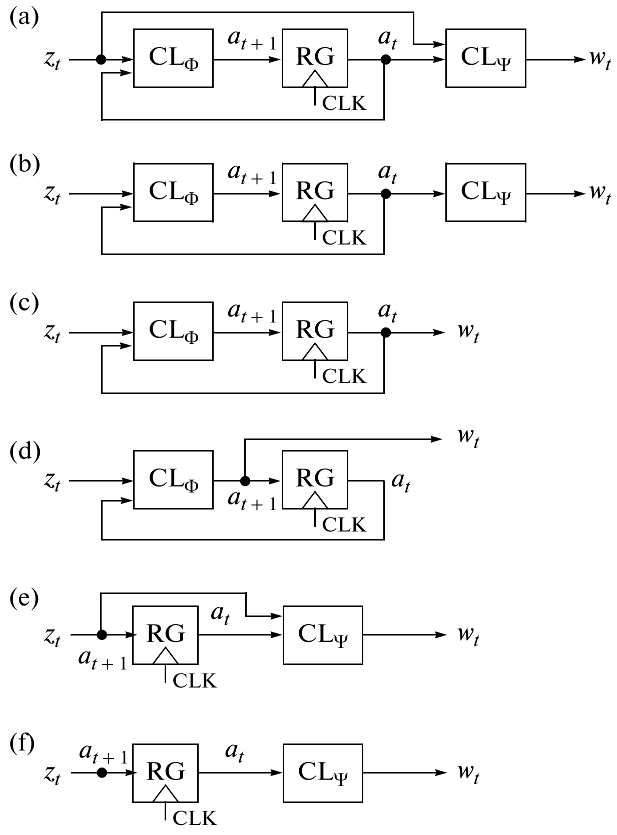

Fig. 1. Structural models of FSMs: (a) for class *A*, (b) for class *B*, (c) for class *C*, (d) for class *D*, (e) for class *E*, (f) for class *F*.

If each output vector w_t of a Mealy FSM coincides with the code of its next state a_{t+1}, then we have a class *D* FSM (Fig. 1d); its behavior is described by the equations

$$a_{t+1} = \varphi(z_t, a_t),$$
$$w_t = a_{t+1}.$$

Its distinctive feature is the absence of the combinational circuit CL$_\psi$, but its outputs are combinational because they are formed at the outputs of the combinational circuit CL$_\varphi$. The difference between the classes *C* and *D* FSMs is that in the class *C* FSMs the output vector w_t coincides with the code of its current state a_t, and in the case of the class *D* FSMs it determines the code of the next state a_{t+1}.

Note that in order to implement a class *C* FSM, it suffices to configure the outputs of the logical elements on which the output variables are implemented as register outputs; to implement a class *D* FSM, it is sufficient to configure the outputs of the logical elements as combinational ones with a trigger in the feedback loop. All modern PLDs make it possible to configure logical elements in this way; therefore, no special output buffers are needed for the implementation of the class *C* and *D* FSMs.

The codes of the FSM internal states can also be determined by sets of values of the input variables. If each input vector z_t of a Mealy FSM coincides with the code of the next state a_{t+1}, then we have the FSM of the class *E* (Fig. 1e) whose operation can be described by the equations

$$a_{t+1} = z_t,$$
$$w_t = \psi(z_t, a_t).$$

Similarly, if each input vector z_t of a Moore FSM coincides with the code of the next state a_{t+1}, then we have the FSM of the class *F* (Fig. 1f) whose operation can be described by the equations

$$a_{t+1} = z_t,$$
$$w_t = \psi(a_t).$$

A feature of the structures of the FSM classes *E* and *F* is the absence of the combinational circuit CL_φ. In the FSMs of the class *E*, the combinational circuit CL_ψ has two types of inputs—combinational and register ones; in the class *F* FSMs, it has only register inputs.

To implement a class *E* FSM on FPGA or SoC, the input buffer must admit two types of connections with the internal logic—a combinational and a register types; to implement a class *F* FSM, it is sufficient that the input FPGA or SoC buffer has the register type of connection with the internal logic. Note that modern FPGAs and SoCs make it possible to construct input buffers with such properties.

In practice, it is rarely possible to directly implement FSMs of the classes *C*–*F*. This is because the input or output vectors cannot always be used as codes of the FSM internal states. For this reason, additional internal memory elements similar to memory elements of the class *A* and *B* FSMs are often introduced to design FSMs of the classes *C*–*F*.

2. STATES OF FSMS OF THE CLASSES *A*–*F*

Let $X = \{x_1, ..., x_L\}$ be the set of input variables; $Y = \{y_1, ..., y_N\}$ be the set of output variables; $A = \{a_1, ..., a_M\}$ be the set of internal states; $D = \{d_1, ..., d_R\}$ be the set of excitation functions of memory elements; $E = \{e_1, ..., e_R\}$ be the set of feedback variables of the FSM, where $R = \mathrm{intlog}_2 M$ is the minimum number of code bits that are sufficient for the FSM internal state assignment; and $B(a_i)$ is the set of states the transitions from which end in the state a_i ($a_i \in A$).

Let the operation of the FSM be specified by a list of transitions. The list of transitions is a table consisting of four columns a_m, $X(a_m, a_s)$, a_s, and $Y(a_m, a_s)$. Each row in the list of transitions corresponds to one transition of the FSM. The column a_m contains the present state of the FSM, the column $X(a_m, a_s)$ contains the input vector that initiates this transition (the transition condition), the column a_s contains the next state, and the column $Y(a_m, a_s)$ contains the output vector formed in this transition. In the case of Moore FSMs, the last column in the transition list is denoted by $Y(a_m)$; it contains the output vectors formed in the states a_m. The transition condition in the column $X(a_m, a_s)$ can be written as a conjunction of the input variables or as a ternary vector. The output sets in $Y(a_m, a_s)$ can be written as a binary (for completely defined FSMs) or ternary (for incompletely defined FSMs) vector.

D e f i n i t i o n 1. The set A_A of states of the class *A* FSM is defined as the set of its internal states; i.e., $A_A = A$.

D e f i n i t i o n 2. The state a_m ($a_m \in A$) is a state of the class *B* FSM (Moore FSM), i.e., $a_m \in A_B$, where A_B is the set of states of the class *B* FSM, if identical output vectors are formed on all the transitions into the state a_m, i.e., if it holds that

$$Y(a_i, a_m) = Y(a_j, a_m) \quad \forall a_i,\ a_j \in B(a_m), \quad i \neq j. \tag{2.1}$$

In this case, the value of any output vector $Y(a_i, a_m)$ can be assigned to the state a_m and denoted by $Y(a_m)$.

D e f i n i t i o n 3. The state a_m is a state of the class *C* FSM, i.e., $a_m \in A_C$, where A_C is the set of states of the class *C* FSM, if this is a Moore FSM (i.e., all the FSM states satisfy conditions (2.1)) and the value of the output vector $Y(a_m)$ is formed in none of the states, except for a_m:

$$Y(a_i) \neq Y(a_m)\ \forall\ a_i \in A_B, i \neq m. \tag{2.2}$$

D e f i n i t i o n 4. The state a_m is a state of the class *D* FSM, i.e., $a_m \in A_D$, where A_D is the set of states of the class *D* FSM, if the same output vector $Y(a_i, a_m)$ is formed on all the transitions into the state a_m (i.e., condition (2.1) is satisfied) and the vector $Y(a_i, a_m)$ is not formed on the transitions into other FSM states, i.e., if it holds that

$$Y(a_s, a_j) \neq Y(a_i, a_m)\ \forall\ a_s, a_j \in A, j \neq m. \tag{2.3}$$

D e f i n i t i o n 5. The state a_m is a state of the class *E* FSM, i.e., $a_m \in A_E$, where A_E is the set of states of the class *E* FSM, if all the transitions into the state a_m are made under the action of the same input vector $X(a_i, a_m)$, i.e., if it holds that

$$X(a_i, a_m) = X(a_j, a_m) \forall\ a_i, a_j \in B(a_m), \quad i \neq j, \tag{2.4}$$

and the vector $X(a_i, a_m)$ does not affect the transitions into other states of the FSM, i.e., if it holds that

$$X(a_s, a_j) \neq X(a_i, a_m)\ \forall\ a_s, a_j \in A, j \neq m. \tag{2.5}$$

D e f i n i t i o n 6. The state a_m is a state of the class F FSM, i.e., $a_m \in A_F$, where A_F is the set of states of the class F FSM, if this is a Moore FSM (i.e., all the FSM states satisfy conditions (2.1)) and the state a_m satisfies conditions (2.4) and (2.5).

D e f i n i t i o n 7. An FSM belongs to the class h ($h \in \{B, C, D, E, F\}$) if

$$A_h = A. \tag{2.6}$$

Note that the sets A_B—A_F may overlap, i.e., the same state may simultaneously belong to FSMs of several classes of FSMs.

3. SYNTHESIS OF FSMS OF THE CLASSES *A–F*

To design FSMs of the classes A and B, all conventional synthesis methods can be used. When a class B FSM and FSMs of the classes C and F are synthesized, the initial FSM must be a Moore FSM. A Mealy FSM can be transformed into a Moore FSM by splitting its internal states as described in [27].

3.1. Synthesis of Class C FSMs

A Moore FSM belongs to the class C if each output vector uniquely determines the code of its internal state. However, this condition is often violated in practice. For example, this is the case when the number N of its output functions is less than the minimally required number of code bits R or when the same output vector is formed in different internal states. Moreover, the determinacy condition of the FSM behavior implies that the codes of the internal states must be mutually orthogonal. In all these cases, additional code bits must be added in order to differentiate between internal state codes, and the structure of the FSM must be supplemented with additional internal memory elements and the corresponding feedback loops.

In the general case, the synthesis of class C FSMs is reduced to a special internal state assignment. To encode the states of a class C FSM, the matrix G is constructed. The rows of G correspond to the internal states of the set A and its columns correspond to the output variables of the set Y. The element in row m and column j of the matrix G contains one if the variable y_j takes the value one in the output set $Y(a_m)$, it contains zero if the variable y_j is zero, and it contains a hyphen if the variable y_j is undefined (do not care). Now, the internal state assignment is reduced to the following problem.

P r o b l e m. Add to the matrix G the minimum number R^* of columns corresponding to the additional internal variables $e_{N+1}, \ldots, e_{N+R^*}$ so that all the rows of G are mutually orthogonal.

This problem can be solved using the following algorithm.

A l g o r i t h m (for the orthogonalization of the rows of the matrix G).

1. Construct the graph H of matrix G row orthogonality. The vertices of H correspond to the rows of G. Two vertices i and j of H are connected by an edge if the rows i and j are mutually orthogonal (have different values in at least one column). Now, the problem is reduced to finding in the graph H the minimum number T of complete subgraphs $H_1, \ldots, H_T$ the vertices of which do not overlap and to assigning to these subgraphs binary codes determined by the values of the variables $e_{N+1}, \ldots, e_{N+R^*}$.

2. Remove from H all the vertices connected to all other vertices of the graph (the rows of G corresponding to these vertices are orthogonal to all other rows).

Set $t := 0$.

3. Set $t := t + 1$. Find a maximal complete subgraph H_t.

4. Remove the vertices of H_t from H. If the set of remaining vertices of H is empty, then go to Step 5; otherwise, go to Step 3.

5. Set $R^* = \mathrm{intlog}_2 T$. Each subgraph H_t ($t = \overline{1, T}$) is associated with the binary code determined by the values of the variables $e_{N+1}, \ldots, e_{N+R^*}$.

6. Add columns corresponding to the variables $e_{N+1}, \ldots, e_{N+R^*}$ to the matrix G. The content of these columns is determined by the codes of the corresponding subgraphs.

7. End.

As the code of a state a_m ($a_m \in A$) the content of the row of the matrix G is used.

E x a m p l e. Consider the synthesis of a class C FSM for the FSM with the graph shown in Fig. 2 and the transition list presented in Table 1. The initial FSM has five internal states $a_1, \ldots, a_5$, one input variable x_1, and three output variables $y_1, \ldots, y_3$.

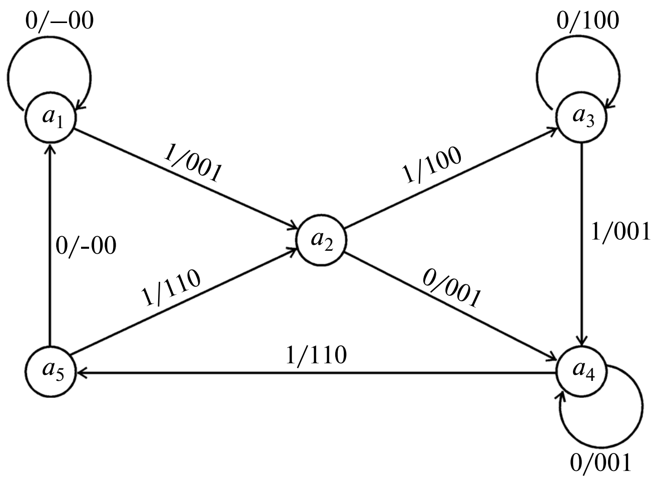

Fig. 2. Graph of the initial FSM.

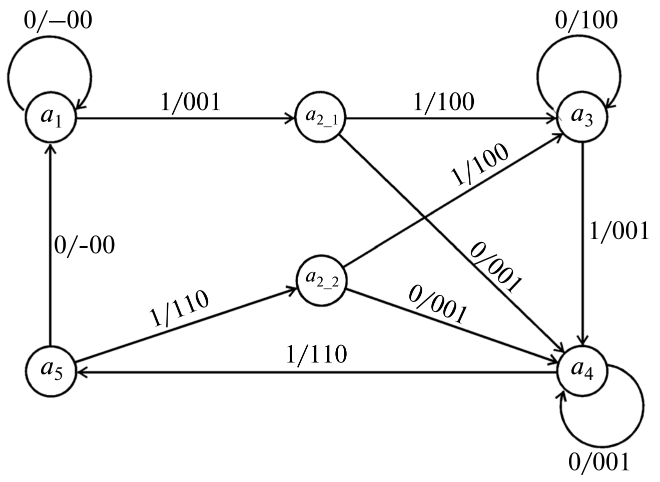

Fig. 3. Graph of the FSM upon splitting the state a_2.

A necessary condition for the synthesis of class C FSMs is the requirement that the initial FSM be a Moore FSM. For the FSM in Fig. 2, conditions (2.1) are satisfied for all the states except for a_2. For this reason, to obtain a Moore FSM from the Mealy FSM, the method described in [27] is used. As a result of applying this method, the state a_2 is split into two states a_{2_1} and a_{2_2}. The graph of the FSM obtained after splitting the state a_2 is shown in Fig. 3, and its list of transitions is shown in Table 2. Since the FSM was

Table 1. List of transitions of the initial FSM

a_m	$X(a_m, a_s)$	a_s	$Y(a_m, a_s)$
a_1	1	a_2	0 0 1
	0	a_1	– 0 0
a_2	1	a_3	1 0 0
	0	a_4	0 0 1
a_3	1	a_4	0 0 1
	0	a_3	1 0 0
a_4	1	a_5	1 1 0
	0	a_4	0 0 1
a_5	1	a_2	1 1 0
	0	a_1	– 0 0

Table 2. List of transitions upon splitting the state a_2

a_m	$X(a_m, a_s)$	a_s	$Y(a_m)$	$D(a_m, a_s)$
a_1	1	a_{2_1}	– 0 0	0 0 1 0
	0	a_1	– 0 0	– 0 0 0
a_{2_1}	1	a_3	0 0 1	1 0 0 1
	0	a_4	0 0 1	0 0 1 1
a_{2_2}	1	a_3	1 1 0	1 0 0 1
	0	a_4	1 1 0	0 0 1 1
a_3	1	a_4	1 0 0	0 0 1 1
	0	a_3	1 0 0	1 0 0 1
a_4	1	a_5	0 0 1	1 1 0 0
	0	a_4	0 0 1	0 0 1 1
a_5	1	a_{2_2}	1 1 0	1 1 0 1
	0	a_1	1 1 0	– 0 0 0

Table 3. The matrix G

	y_1	y_2	y_3	e_4
a_1	–	0	0	0
a_{2_1}	0	0	1	0
a_{2_2}	1	1	0	1
a_3	1	0	0	1
a_4	0	0	1	1
a_5	1	1	0	0

transformed into a Moore FSM, Table 2 contains the column $Y(a_m)$ instead of the column $Y(a_m, a_s)$, which contains the output vectors formed in the state a_m.

To encode the internal states of a class C FSM, the matrix G shown in Table 3 (columns y_1, y_2, and y_3) is constructed and the orthogonalization algorithm for the rows of G is executed. At Step 1 of this algorithm, the graph H of the matrix G row orthogonality (Fig. 4) is constructed. Next, the graph H is decomposed into the minimal number of complete subgraphs H_1 and H_2 (Fig. 5). To encode the subgraphs H_1

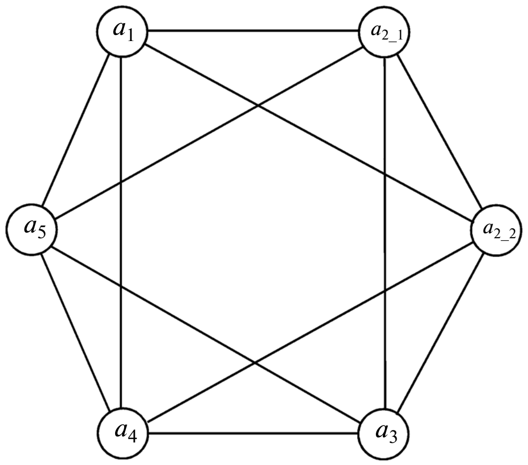

Fig. 4. Orthogonality graph H for the rows of the matrix G.

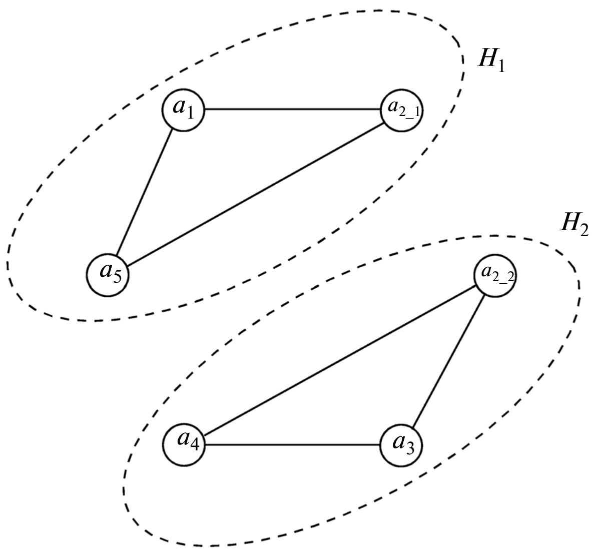

Fig. 5. Decomposition of the graph H into the minimal number of complete subgraphs.

and H_2, one internal variable e_4 is sufficient. The results of encoding subgraphs are presented in Table 3 (the column e_4); the subgraph H_1 is assigned the code 0 and H_2 is assigned the code 1.

Upon the orthogonalization, the rows of G determine the following codes of the internal states: "–000" for a_1, "0010" for a_{2_1}, "1101" for a_{2_2}, "1001" for a_3, "0011" for a_4, and "1100" for a_5. The column $D(a_m, a_s)$ is added to Table 2 that contains the values of the excitation functions of the memory elements d_1, ..., d_4 formed on the transitions from the state a_m into the state a_s. Now, using the internal state codes and the list of transitions from Table 2, we can write the logical equations of the FSM combinational part:

$$d_1 = x_1\bar{e}_1\bar{e}_2e_3 + x_1e_1e_2\bar{e}_3 + \bar{x}_1e_1\bar{e}_2\bar{e}_3e_4,$$

$$d_2 = x_1\bar{e}_1\bar{e}_2e_3e_4 + x_1e_1e_2\bar{e}_3\bar{e}_4,$$

$$d_3 = x_1\bar{e}_2\bar{e}_3\bar{e}_4 + \bar{x}_1\bar{e}_1\bar{e}_2e_3 + \bar{x}_1e_1e_2\bar{e}_3e_4 + x_1e_1\bar{e}_2\bar{e}_3e_4,$$

$$d_4 = \bar{e}_1\bar{e}_2e_3\bar{e}_4 + e_1\bar{e}_3e_4 + \bar{x}_1\bar{e}_1\bar{e}_2e_3e_4 + x_1e_1e_2\bar{e}_3\bar{e}_4.$$

An implementation of the FSM is shown in Fig. 6. Thus, to construct the C class FSM on PLD, four logical elements are needed—three for the output variables y_1, ..., y_3 and one for the implementation of the excitation function of the internal memory element corresponding to the variable e_4. For comparison, the implementation of the class A Mealy FSM in this example without using the models proposed above requires six logical elements—three for the output and three variables for the implementation of the excitation functions of the memory elements.

3.2. Synthesis of Class D FSMs

An FSM belongs to the class D if each output vector uniquely determines the code of its next state. In the synthesis of class D FSMs, the same problems as in the synthesis of class C FSMs arise—the output sets do not always determine the codes of the transition states, the number of output functions can be less than the minimum number R of code bits, different transition states are associated with identical output sets, and the state codes must be mutually orthogonal.

Therefore, the synthesis method of the class D FSMs is similar to the method used to synthesize class C FSMs. The difference is in the construction of the matrix G. Namely, each row m of G is assigned the content of the output vector $Y(a_i, a_m)$, where $a_i \in B(a_m)$ and $a_m \in A_D$. The further synthesis is performed using the algorithm described above.

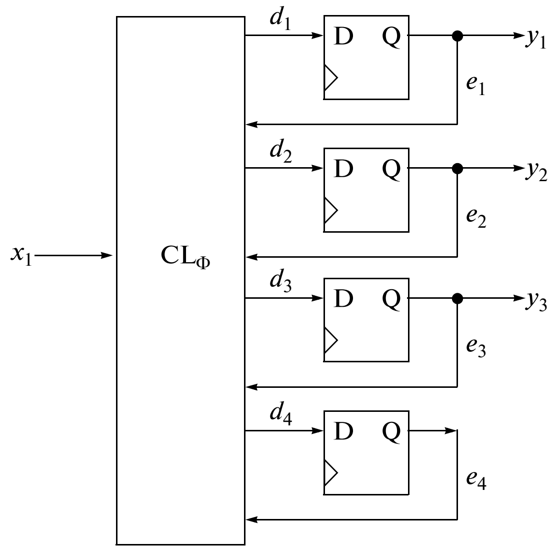

Fig. 6. Implementation of the FSM.

3.3. Synthesis of Class E and F FSMs

A Mealy (Moore) FSM belongs to the class E (F) if its each input vector coincides with the code of the next state. However, the sets of input variable values cannot always be the codes of internal states. For example, this is the case when the number L of the input variables is less then R or when transitions to different internal states are initiated by the same input vector. On the other hand, not every set of input vectors can be used to encode the internal states of the FSM; indeed, the condition that the behavior of the FSM must be deterministic requires that the codes of different states must by mutually orthogonal. In all these cases, the structure of the FSM belonging to the class E and F must be complemented with additional code bits and additional internal memory elements.

The internal state assignment for the class E and F FSMs is similar to that for the class C and D FSMs. The difference is in the construction of the matrix G. The rows of G correspond to the internal states of the set A, and the columns correspond to the input variables of the set X. Let $X(a_i)$ be the condition of transition into the state a_i ($a_i \in A_E$). The element in the row i and column j of G has the value 1 if the variable x_j of the input vector $X(a_i)$ is equal to 1; it has the value 0 if x_j is equal to 0, and it has the value hyphen if the value of x_j is undefined. The matrix G for the FSM of the class F is similar but the set A_E is replaced with A_F.

To encode the internal states of the FSMs of the classes E and F, the algorithm for the orthogonalization of the rows of the matrix G is used in which the number N of the output variables is replaced with the number L of input variables. Upon the execution of this algorithm, the rows of the matrix G determine the internal state codes of the FSMs of the classes E and F.

R e m a r k 1. Conditions (2.2), (2.3), (2.5), and (2.6) are very stringent, and they are rarely satisfied in the design of FSMs. For this reason, in order to better use the input and output vectors as codes (or part of code) of internal states in the synthesis of FSMs of the classes C–F, conditions (2.2), (2.3), (2.5), and (2.6) may be dropped. In this case, the rows of the matrix G corresponding to the internal states for which these conditions are violated are assigned undefined values '–'.

4. EXPERIMENTAL RESULTS

Experiments were performed on the benchmarks of MCNC [28]. First, the potential possibility of using the input and output vectors as a code (or part of code) of internal states of FSMs was checked. To this end, according to conditions (2.1)–(2.5), the sets A_A–A_F of the internal states of the FSMs of classes A–F were specified. The set A_A is the set of states of the initial FSM, A_B is the set of states of the FSM obtained by applying the method described in [27], A_C is the subset of states of A_B satisfying condition (2.2), A_D is the subset of states of A_A satisfying conditions (2.1) and (2.3), A_E is the subset of states of A_A satisfying conditions (2.4) and (2.5), and A_F is the subset of states of A_B satisfying conditions (2.4) and (2.5).

The experimental results are presented in Table 4, where L is the number of input variables; N is the number of output variables; M_A–M_F are the numbers of the elements in the sets A_A–A_F, respectively; and

Table 4. The number of internal states of class *A*–*F* FSMs

Name	L	N	M_A	M_B	M_C	M_D	M_E	M_F
bbara	4	2	10	12	2	0	0	0
bbsse	7	7	16	24	7	1	0	5
bbtas	2	2	6	9	3	0	0	0
beecount	3	4	7	10	3	0	0	1
cse	7	7	16	29	7	0	0	0
dk14	3	5	7	26	1	1	0	0
dk15	3	5	4	17	6	0	0	0
dk16	2	3	27	75	0	0	0	0
dk17	2	3	8	16	0	0	0	0
dk27	1	2	7	10	0	0	0	0
dk512	1	3	15	24	0	0	0	0
donfile	2	1	24	24	0	0	0	0
ex1	9	19	18	78	50	1	0	14
ex2	2	2	19	23	0	0	0	0
ex3	2	2	10	13	2	1	0	0
ex4	6	9	14	18	8	2	0	7
ex5	2	2	9	13	0	0	0	0
ex6	5	8	8	14	12	2	0	4
ex7	2	2	10	14	0	0	0	0
keyb	7	2	19	20	1	0	1	1
lion	2	1	4	5	0	0	0	0
lion9	2	1	9	11	0	0	0	0
mc	3	5	4	8	8	0	0	2
modulo12	1	1	12	12	0	0	0	0
planet	7	19	48	95	37	1	2	20
pma	8	8	24	49	5	1	12	13
s1	8	6	20	20	20	20	0	0
s1488	8	19	48	168	3	0	0	42
s1494	8	19	48	168	3	0	1	39
s1a	8	6	20	20	0	0	0	0
s208	11	2	18	37	3	0	0	2
s27	4	1	6	6	0	0	0	0
s298	3	6	218	332	0	0	0	0
s386	7	7	13	23	5	1	0	3
s420	19	2	18	37	3	0	0	2
s510	19	7	47	73	2	0	7	20
s8	4	1	5	5	0	0	0	0
s820	18	19	25	70	1	0	1	18
s832	18	19	25	70	1	0	1	18
sand	11	9	32	84	18	2	0	19
shiftreg	1	1	8	16	0	0	0	0
see	7	7	16	24	7	1	0	5
styr	9	10	30	57	13	3	0	2
tav	4	4	4	27	7	0	0	6
tbk	6	3	32	60	0	0	0	0
tma	7	6	20	38	8	1	4	4
train11	2	1	11	13	0	0	0	0
train4	2	1	4	5	0	0	0	0
mid	6.0	5.9	21.0	42.0	5.13	0.79	0.60	5.15

Table 5. The number of internal memory elements of FSMs

Name	R_A	R_D	R_E	R_{AD}, %	R_{AE}, %	R_B	R_C	R_F	R_{BC}, %	R_{BF}, %
bbsse	4	4	4	0	0	5	4	5	20	0
bbtas	3	3	3	0	0	4	3	4	25	0
beecount	3	3	3	0	0	4	3	4	25	0
dk15	2	2	2	0	0	5	4	5	20	0
ex1	5	5	5	0	0	7	5	6	29	14
ex4	4	4	4	0	0	5	4	4	20	20
ex6	3	3	3	0	0	4	1	4	75	0
mc	2	2	2	0	0	3	0	3	100	0
planet	6	6	6	0	0	7	6	7	14	0
pma	5	5	4	0	20	6	6	6	0	0
s1	5	0	5	100	0	5	0	5	100	0
s1488	6	6	6	0	0	8	8	7	0	13
s1494	6	6	6	0	0	8	8	7	0	13
s510	6	6	6	0	0	7	7	6	0	14
s820	5	5	5	0	0	7	7	6	0	14
s832	5	5	5	0	0	7	7	6	0	14
see	4	4	4	0	0	5	4	5	20	0
tma	5	5	4	0	20	6	5	6	17	0
mid	4.39	4.11	4.28	5.56	2.22	5.72	4.89	5.39	25.83	5.67

mid is the average value of the corresponding parameter. The analysis of Table 4 shows that when the Mealy FSMs are replaced with Moore FSMs, the number of states increases from 21.0 to 42.0 on the average, i.e., it doubles. The average number of states in the class C FSMs is 5.13 (or 12.21%), in the class D FSMs it is 0.79 (or 3.76%), in the class E FSMs it is 0.60 (or 2.86%), and in the class F FSMs it is 5.15 (or 12.26%). Thus, the most potential for using the values of the input vectors as the code (or part of code) of internal states has the class F, while the class C has the most potential for using the output vectors for this purpose.

We also analyzed the efficiency of using the proposed structural models for the synthesis of FSMs. As the efficiency indicator, we used the number of additional code bits (internal memory elements) that must be added to encode the states of the FSM of the corresponding class. It is clear that all the memory elements of the class A and B FSMs are internal; for the classes C–F, some memory elements can be in the input (for the classes E and F) or output (for the classes C and D) buffers.

The experimental results are presented in Table 5, where R_A, R_D, R_E, R_B, R_C, and R_F are the numbers of internal memory elements of the FSMs of the classes A, D, E, B, C, and F, respectively; $R_{AD}(\%)$ shows how much the use of the class D FSM model reduces the number of internal memory elements compared with the use of the class A FSM on a percentage basis; $R_{AD}(\%) = [(R_A - R_D)/R_A] \cdot 100\%$, R_{AE}, R_{BC}, and $R_{BF}(\%)$ are interpreted similarly; and *mid* is the average value of the corresponding parameter.

The analysis of Table 5 shows that the use of the class D FSMs for the these example reduces the number of internal memory elements by 5.56% on the average (and by 100% for certain examples); the use of class E FSMs by 2.22% (by 20% for certain examples); the use of class C FSMs by 25.832% (by 100% for certain examples); and the use of class F FSMs by 5.68% (by 20% for certain examples).

It should be emphasized that the use of the class C FSM model for the synthesis in examples *ex6* and *mc* made it possible to obtain less internal memory elements than in the models of Mealy FSMs (classes A, D, and E). In other words, the use of Moore FSM instead of Mealy FSM and the use of a class C FSM sometimes reduces the number of internal memory elements compared with Mealy FSM models.

R e m a r k 2. The efficiency of FSM models of the classes C–F is not limited to the reduction of the number of internal memory elements and the number of memory element excitation functions. In fact, the combinational part of the FSM is simplified, which ultimately reduces the implementation cost, the delays on the critical path, and the power consumption.

CONCLUSIONS

A theoretical justification of the use of triggers of input and output buffers as memory elements of the FSM is given. To this end, a new classification of FSM structural models in which all FSMs are subdivided into six classes *A*, *B*, *C*, *D*, *E*, and *F* is proposed. Models in the classes *A* and *B* are the conventional models of Mealy and Moore FSMs; in the class *C* and *D* FSMs, output buffer triggers are used as memory elements; and in the class *E* and *F* FSMs, input buffer triggers are used as memory elements.

Definitions of states of each class of FSMs and the definition of FSMs of each class are given. A general method for the synthesis of FSMs is described using the class *C* as an example. It provides a basis for the synthesis of FSMs of the classes *D*, *E*, and *F*.

The experimental results demonstrated the high efficiency of the proposed structural models of FSMs. The use of the models of the classes *C*–*F* reduces the number of internal memory elements by 2.22–25.38% on the average and by 100% for certain examples.

The further study in this field can be in the direction of construction of combined models of FSMs and elaboration of the available synthesis methods.

ACKNOWLEDGMENTS

This work was supported by the Bialystok University of Technology (Poland), grant no. S/WI/1/2013.

REFERENCES

1. G. H. Mealy, "Method for synthesizing sequential circuits," Bell Syst. Tech. J. **34**, 1045–1079 (1955).
2. E. F. Moore, "Gedanken-experiments on sequential machines," in *Automata Studies*, Ed. by C. Shannon and J. McCarthy (Princeton Univ. Press, Princeton, 1956), pp. 129–153.
3. L. D. Coraor, P. T. Hulina, and O. A. Morean, "A general model for memory-based finite-state machines," IEEE Trans. on Computers **C-36**, 175–184 (1987).
4. R. Senhadji-Navarro, I. Garcia-Vargas, and J. L. Guisado, "Performance evaluation of RAM-based implementation of finite state machines in FPGAs," *Proc. of the 19th IEEE Int. Conf. on Electronics, Circuits and Systems (ICECS), Seville, Spain, 2012*, pp. 225–228.
5. V. Sklyarov, "Hierarchical finite-state machines and their use for digital control," IEEE Trans. Very Large Scale Integration Syst. **7**, 222–228 (1999).
6. V. Sklyarov and I. Skliarova, "Synthesis of parallel hierarchical finite state machines," in *Proc. of the 21st Iranian Conf. of Electrical Engineering (ICEE), Mashhad, Iran, 2013*, pp. 1–8.
7. A. Barkalov, L. Titarenko, and S. Chmielewski, "Reduction in the number of PAL macrocells in the circuit of a Moore FSM," Int. J. Appl. Math. Comput. Sci. **17**, 565–575 (2007).
8. A. Barkalov, L. Titarenko, and J. Bieganowski, "Reduction in the number of LUT elements for control units with code sharing," Int. J. Appl. Math. Comput. Sci. **20**, 751–761 (2010).
9. A. Barkalov, L. Titarenko, R. V. Malhava, and K. A. Soldatov, "Hardware reduction in FPGA-based Moore FSM," J. Circuits, Syst. Comput. **22** (3), 1–20 (2013).
10. V. Sklyarov, "FPGA-based implementation of recursive algorithms," Microprocessors and Microsystems, Special Issue on FPGAS: Applications and Designs **28** (5–6), 197–211 (2004).
11. S. Ninos and A. Dollas, "Modeling recursion data structures for FPGA-based implementation," in *Proc. of the Int. Conf. on Field Programmable Logic and Applications (FPL), Heidelberg, Germany, 2008*, pp. 11–16.
12. D. Mihhailov, V. Sklyarov, I. Skliarova, and A. Sudnitson, "Hardware implementation of recursive algorithms," in *Proc. of the 53rd IEEE Int. Midwest Symposium on Circuits and Systems (MWSCAS), Seattle, 2010*, pp. 225–228.
13. M.-D. Shieh, C.-L. Wey, and P. D. Fisher, "Model of asynchronous finite state machines and their pipelined structures," *Proc. of the 35th Midwest Symposium on Circuits and Systems, Washington, 1992*, Vol. 1, pp. 659–662.
14. M. Koster and J. Teich, "(Self-)reconfigurable finite state machines: theory and implementation," in *Proc. of the Design, Automation and Test in Europe Conference and Exhibition (DATE 2002), Paris, 2002*, pp. 559–566.
15. P. P. Chu, *FPGA Prototyping by VHDL Examples* (Wiley-InterScience, Hoboken, USA, 2008).
16. I. Pomeranz and K.-T. Cheng. "STOIC: state assignment based on output/input functions," IEEE Trans. on CAD **12**, 613–622 (1993).
17. J. Forrest, "ODE: output direct state machine encoding," in *Proc. of the European Design Automation Conference (EURO-DAC'95), Brighton, Great Britain, 1995*, pp. 600–605.
18. S. Mitra, L. J. Avra, and E. J. McCluskey, "An output encoding problem and a solution technique," IEEE Trans. on CAD **18**, 761–768 (1999).

19. V. Solovjev and M. Chyzy, "Models of the finite state machines," in *Proc. of the Sixth Int. Conf. on Methods and Models in Automation and Robotics (MMAR 2000), Miedzyzdroje, Poland, 2000*, Vol. 2, pp. 909–914.
20. V. V. Solov'ev, *Designing Digital Systems Based on Programmable Logic Devices* (Goryachaya liniya—Telekom, Moscow, 2001) [in Russian].
21. V. Solov'ev, "Synthesis of sequential circuits on programmable logic devices based on new models of finite state machines," in *Proc. of the EUROMICRO Symp. on Digital Systems Design (DSD'2001), Poland, Warsaw, 2001*, pp. 170–173.
22. A. Klimowicz and Solovjev, "Synthesis of sequential circuits on programmable logic devices based on common model of finite state machine," in *Proc. of East-West Design & Test Conference (EWDTC'03), Alushta, Ukraine, 2003*, pp. 136–139.
23. V. V. Solov'ev and A. Klimowicz, *Logical Design of Digital Systems Based on Programmable Logic Devices* (Goryachaya liniya—Telekom, Moscow, 2008) [in Russian].
24. A. Klimowicz and V. Salauyou, "The synthesis of combined Mealy and Moore machines structural model using values of output variables as codes of states," in *Proc. of the 15th EUROMOCRO Symp. on Digital System Design (DSD 2012), Izmir, Turkey, 2012,* pp. 789–794.
25. V. V. Solov'ev, "Implementation of finite-state machines based on programmable logic ICs with the help of the merged model of Mealy and Moore machines," J. Commun. Technol. Electron. **58**, 172–177 (2013).
26. *Programmable Logic* (Intel, Santa Clara,1994).
27. A. S. Klimovich and V. V. Solov'ev, "Transformation of a Mealy finite–state machine into a Moore finite–state machine by splitting internal states," J. Comput. Syst. Sci. Int. **49**, 900–908 (2010).
28. S. Yang, "Logic synthesis and optimization benchmarks user guide. Version 3.0, Technical Report of the Microelectronics Center of North Carolina (1991).

Translated by A. Klimontovich