ISSN 1064-2269, Journal of Communications Technology and Electronics, 2013, Vol. 58, No. 2, pp. 172–177. © Pleiades Publishing, Inc., 2013.
Original Russian Text © V.V. Solov'ev, 2013, published in Radiotekhnika i Elektronika, 2013, Vol. 58, No. 2, pp. 192–198.

PHYSICAL PROCESSES IN ELECTRON DEVICES

Implementation of Finite-State Machines Based on Programmable Logic ICs with the Help of the Merged Model of Mealy and Moore Machines

V. V. Solov'ev

Bialystok University of Technology, ul. Wiejska 45A, Bialystok, 15-351 Poland

Received September 3, 2010

Abstract—The problem concerning the synthesis of finite-state machines (FSMs) based on programmable logic ICs, in which FSM's output variables serve as the code (or part of the code) of internal states, has been examined. A solution to the problem is obtained with the help of the merged model of Mealy and Moore machines. A principal distinction between the proposed and well-known techniques is that an initial FSM undergoes no conversions related to an increase in the number of internal states and transitions thereof. The necessary conditions under which output variables can be used as the code of FSM's internal states are presented. The method for synthesizing the merged model *AC* of Mealy and Moore machines is described. Basic results of investigations, as well as promising directions of further studies concerned with the development of new structural models of FSMs, are discussed.

DOI: 10.1134/S106422691302006X

INTRODUCTION

Digital system designers have long been interested in the problem of applying the sets of input and/or output variables of finite-state machines (FSMs) as the part of the code of internal states [1–7]. In a number of cases, approaches of this kind can substantially reduce the implementation cost of FSMs and enhance their speed. With the advent of programmable logic ICs (PLICs) [8], such as complex programmable logic devices (CPLDs) and field programmable gate arrays (FPGAs) utilized as circuit components in digital systems, the development of efficient methods for synthesizing PLIC-based FSMs becomes a burning issue. For this purpose, six structural models of FSMs, which were called class *A*, *B*, *C*, *D*, *E*, and *F* machines and built around PLICs without any difficulties, have been proposed in [7].

The class *A* FSM is a classical Mealy machine [9], in which an output set depends on the input set and internal state of an FSM at each instant. The class *B* FSM is equivalent to a classical Moore machine [10], where each output set depends only on the FSM's internal state. Class *C*, *D*, *E*, and *F* FSMs are the modifications of class *A* and *B* machines. In the class *C* FSM, each output set of the Moore machine is identical to the code of the corresponding internal state. In the class *D* FSM, each output set of the Mealy machine corresponds to the code of the next internal state. The class *E* FSM is distinguished by the fact by that each input set of the Mealy machine coincides with the code of the next state. By analogy with the previous situation, the class *F* FSM is characterized by fact that each input set of the Moore machine coincides with the code of the next state. Class *C* and *D* machines make it possible to build FSM's memory elements around the flip-flops of macrocells (for CPLDs) and logic gates (for FPGAs) whereby its output functions are mechanized. In the class *E* and *F* machines, FSM's memory elements can be implemented utilizing the input buffers of PLICs.

In [7], the time analysis of structural models of FSMs was performed under the assumption of their maximum speed and the minimum clock cycle of an FSM (the minimum timing cycle), as well as the maximum frequency of FSM functioning, was determined. On the basis of time analysis, the authors of [7] have proposed the merged models *ADE*, *AD*, *AE*, and *BF* of FSMs.

The principal restriction according to which different classes of FSMs can be merged into one model is a complete similarity between the time diagrams of output signals in the merged models. The latter condition is appreciably diminishes the possibility of merging of different models. In practice, FSMs often have the properties of different models. However, owing to the aforementioned restriction, it is necessary to employ the most general models FSMs: Mealy machines [9] and Moore machines [10].

In this study, PLIC-based FSMs are implemented under the assumption of violation of an absolute similarity between the time diagrams of the output signals in the merged models. The sufficient condition is that output signals of merged models must be formed within the same FSM clock cycle. Such an abatement of the model merging condition defined in [7] makes it possible to combine class *A* and *C* FSMs. As a result,

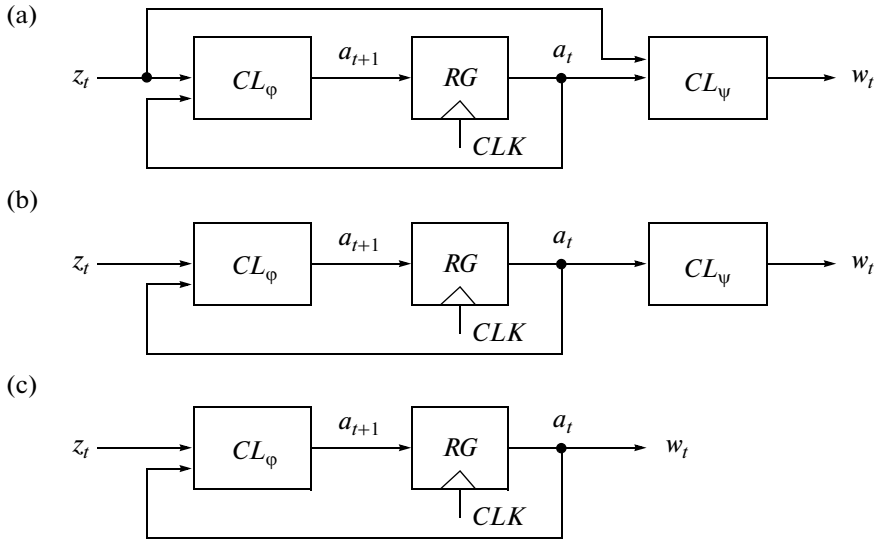

Fig. 1. Structures of the basic models of FSMs: class (a) *A*, (b) *B*, and (c) *C* machines.

the implementation cost can frequently be decreased without any reduction in FSM speed.

1. BASIC STRUCTURAL MODELS

In practice, sequential circuits are designed using two widespread types of FSMs: Mealy and Moore machines. The Mealy FSM behavior can be described by the equations

$$\begin{aligned} a_{t+1} &= \varphi(z_t, a_t); \\ w_t &= \psi(z_t, a_t), \end{aligned} \tag{1}$$

where φ is the transition function, ψ is the output function, z_t is the input set at instant t ($t = 1, 2, 3, ...$), w_t is the generated output set, a_t is the present state of an FSM, and a_{t+1} is the next state of an FSM.

The Moore FSM behavior is described by the equations

$$\begin{aligned} a_{t+1} &= \varphi(z_t, a_t); \\ w_t &= \psi(a_t). \end{aligned} \tag{2}$$

A distinctive feature of the Mealy FSM is the dependence of output set w_t on both input set z_t and internal state a_t. The output set w_t of the Moore FSM is determined only by internal state a_t. The structural models of Mealy and Moore machines are presented in Figs. 1a and 1b, where combinational circuits CL_φ and CL_ψ and register *RG* controlled by clock signal *CLK* mechanize a transition function, an output function, and an FSM memory, respectively.

If each output set w_t of a Moore machine coincides with the code of its internal state a_t, the FSM behavior can be defined as

$$\begin{aligned} a_{t+1} &= \varphi(z_t, a_t); \\ w_t &= a_t. \end{aligned} \tag{3}$$

The given type of an FSM is referred to as a class *C* machine [11]. By analogy, Mealy and Moore FSMs (Figs. 1a, 1b) are called class *A* and *B* machines, respectively. The structure of a Moore (class *C*) FSM is depicted in Fig. 1c. Its distinctive features are that such a machine have no combinational circuits CL_ψ and all outputs thereof are formed at the outputs of register *RG*. Note that the above structure is easily implementable with the help of PLICs because their output macrocells can be configured so as to have register outputs and feedback circuits.

2. PROPOSED TECHNIQUE

Let us consider the FSM whose sets of input and output variables and internal states are chosen as $X = \{x_1,..., x_L\}$, $Y = \{y_1, ..., y_N\}$, and $A = \{a_1, ..., a_M\}$, respectively.

In designing of FSMs, all transitions from state a_i, $a_i \in A$, are often accompanied by the formation of the same sets w_i of output variables. Let output set w_i be the code (or part of the code) of state a_i. When the FSM passes to state a_i, register *RG* stores the code of state a_i and set w_i is formed at its outputs. Hence, the memory element outputs, which are equal to unities in set w_i, can serve as FSM outputs. For this purpose, it will suffice to equip the PLIC's output macrocells used to implement the given memory elements with register outputs. Thus, both output and feedback variables can be generated by the same PLIC macrocells.

Since output set w_i is formed *at all* transitions from state a_i, set w_i is independent of input variables and depends only on internal state a_i. In other words, w_i is actually the output set of a Moore FSM. Moreover, since the code (or part of the code) of state a_i and output set w_i are identical, w_i is the output set of a class *C* FSM. Thus, the class *AC* FSM is a merged model.

Theorem. *If output set w_i is used as the code (or part of the code) of state a_i, it is required to satisfy the following necessary conditions*:

(i) *output set w_i must be formed only at FSM transitions from state a_i*;

(ii) *all output variables, which are equal to unities in set w_i, must be the output variables of a Moore FSM, i.e., independent of input variables*.

Proof. The violation of condition (1) can lead to an unforeseen FSM transition to state a_i, i.e., the degradation of deterministic behavior thereof. The violation of condition (2) makes it impossible to vary the Moore FSM's output variables by changing input variables, i.e., deteriorates the algorithm of FSM functioning.

The essence of the proposed technique consists in determining the sets of output variables such that are generated at all transitions under the corresponding conditions and satisfy the above theorem and performing the special encoding of FSM's internal states whereby the previously found output sets are used as the code (or part of the code) of internal states.

Note that the proposed approach somewhat differs from the known FSM synthesis methods. In [7], class C FSMs are synthesized via the method in which the merging of class A and class C machines is assumed to be unacceptable due to a dissimilarity between the time diagrams of output signals generated in the given machines. By contrast, the proposed technique relies on the merged model of class A and C machines.

Let the FSM clock cycle begin after its transition to a steady internal state, i.e., when the memory element outputs reaches true values.

In implementation of Mealy FSMs based on PLICs, output signals are commonly formed within delay time t_{PD} of PLIC's combinational circuits after the beginning of the clock cycle. In the proposed approach, Moore machine's output signals, which are determined by the code of an FSM internal state, are formed at the very beginning of the FSM clock cycle, i.e., pass ahead of Mealy machine's output signals by at least time t_{PD}. In the majority of practical applications, such formation of output signals is admissible.

In the method for synthesizing a class C FSM [7], a transition from the Mealy to Moore machine is obligatory. As a rule, this is due to the increased number of internal states and transitions of an FSM. In the proposed technique, an initial FSM undergoes no conversions caused by an increase in the number of internal states and transitions thereof.

3. METHOD FOR SYNTHESIZING THE MERGED MODEL *AC* OF FINITE-STATE MACHINES

Let A be a set of internal states in the certain FSM. The Moore machine's internal state is called FSM state a_i, $a_i \in A$, all transitions from which lead to the formation of the same set of output variables. For the given FSM, the collection of all states of a Moore machine generates the set A_B of Moore machine states, $A_B \subseteq A$. The remaining internal states of the FSM belong to the set A_A of Mealy machine states, $A_A = A \backslash A_B$.

Let $y(a_i)$ be the set of output variables whose unities are formed at transitions from state a_i, $a_i \in A$. For the Mealy FSM, the output variables of set Y_A are called the output variables whose unities are formed at transitions from Mealy FSM states belonging to set A_A:

$$Y_A = \{y(a_i) | a_i \in A_A\}. \quad (4)$$

The remaining FSM's output variables of set Y_B are called the Moore machine's output variables:

$$Y_B = Y \backslash Y_A. \quad (5)$$

Let us explain some features arising when the output variables of Mealy and Moore machines are implemented with the help of PLICs. The Mealy FSM's output variables of set Y_A depend directly on input variables and are asynchronous with respect to them, i.e., can vary under changes in the corresponding input variables. Hence, the Mealy FSM's output variables of set Y_A are always formed at the combinational outputs of PLICs.

The Moore FSM's output variables belonging to set Y_B depend directly on input variables, vary with a change in the FSM's internal state' and can serve as the code (or part of the code) of internal states. To implement the Moore machine's output variables, PLIC macrocells are equipped with register outputs, and macrocell feedbacks are used to transmit their variables to the FSM's combinational circuit inputs.

On the basis of the foregoing discussion, the algorithm for synthesizing the merged model *AC* of FSMs is formulated as follows.

Algorithm. 1. The set Y_B of Moore machine's output variables is found. If $Y_B = \varnothing$ ($\varnothing$ is the empty set), the method for synthesizing a Mealy machine [7] must be used and the algorithm passes to step 6.

2. Triple matrix **W** is constructed. The rows of matrix **W** correspond to the FSM's internal states belonging to set A, and its columns contain the Moore machine's output variables of set Y_B. For the Moore machine's internal states of set A_B, its output variables defined by set Y_B, which are formed at transitions from the corresponding states, are written in the corresponding rows of matrix **W**. For the Mealy machine's internal states of set A_A, indefinite values (hyphens) are written in the corresponding rows of matrix **W**.

3. The problem of redefining the rows of matrix **W** by means of a binary code of minimum length R is solved so as to obtain the mutual orthogonality of all rows in matrix **W**.

4. The codes of FSM's internal states are found. Each state a_i, $a_i \in A$, is associated with code $K(a_i)$ equal to the row i of matrix **W**.

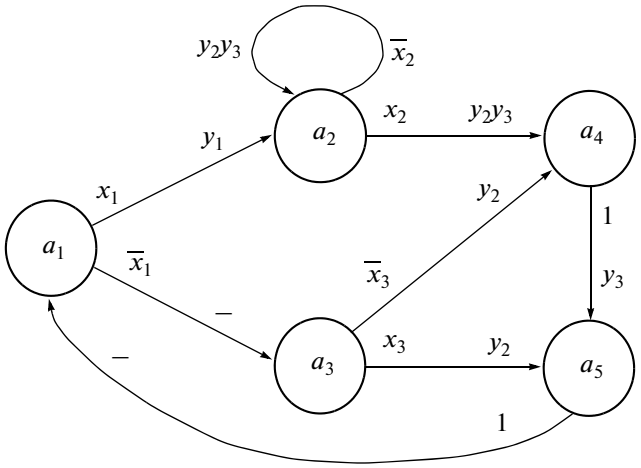

Fig. 2. Graph of the FSM chosen in the example.

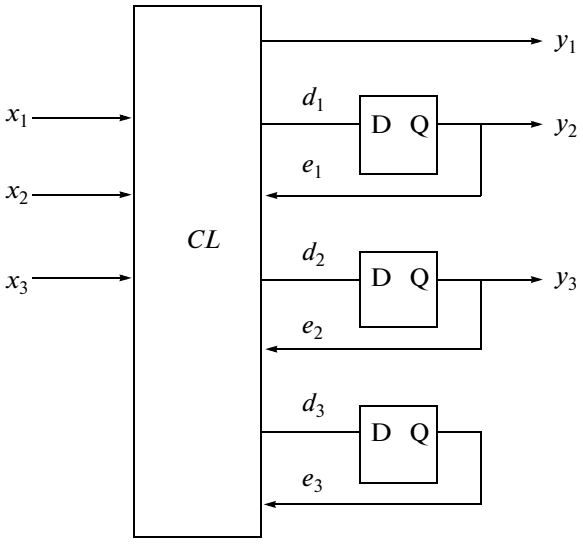

Fig. 3. Structural model *AC* of the PLIC-based FSM from the example. Here, *CL* is the FSM's combinational circuit; x_1, x_2, x_3 (y_1, y_2, y_3) are the FSM's input (output) variables; d_1, ..., d_3 are the functions of excitation of memory elements (transition functions); and e_1, ..., e_3 are the feedback variables.

5. Logic equations for the transition and output functions of an FSM are constructed.

6. End.

The problem of redefining the rows of matrix **W** by means of a binary code of minimum length R (step 3 in the algorithm) can be solved via the method described in [7]. The essence of the method is to construct the orthogonality graph of the rows in matrix **W**, calculate the minimum number of nonintersecting full subgraphs in the orthogonality graph, and encode the last binary codes of minimum length.

Example. Let us consider synthesis of the merged model *AC* of an FSM with the graph shown in Fig. 2. The given FSM is characterized by input variables x_1, x_2, and x_3, output variables y_1, y_2, and y_3, and five internal states (a_1, ..., a_5). The list of FSM transitions is illustrated in Table 1. Note that unity in column $X(a_m, a_s)$ designates an unconditional transition and hyphen in column $Y(a_m, a_s)$ corresponds to zero output set. According to the given FSM, we obtain

$$A_A = \{a_1\} \quad \text{and} \quad A_B = \{a_2, a_3, a_4, a_5\},$$

because only transitions from state a_1 generate the different output variables ($y_1 = 1$ if $x_1 = 1$, and the output set is zero if $x_1 = 0$).

In the case of state a_1, we obtain $y(a_1) = \{y_1\}$. Hence, the set $Y_A = \{y_1\}$ is found from (4). The set $Y_B = \{y_2, y_3\}$ is determined from (5). Thus, output variables y_2 and y_3 are the Moore machine's output variables and can serve as the part of the code of states belonging to set A_B.

Matrix **W** obtained in step 3 of the algorithm is presented in the first two columns of Table 2. Output variables y_2 and y_3 correspond to transition functions d_1 and d_2. To orthogonalize all rows of matrix **W**, it will suffice to employ one binary variable corresponding to transition function d_3 (the third column in Table 2). The values of rows from matrix **W** make it possible to determine the following codes of FSM's internal states:

$$K(a_1) = (--1);$$
$$K(a_2) = (110);$$
$$K(a_3) = (100);$$
$$K(a_4) = (010);$$
$$K(a_5) = (000).$$

The structural model of the PLIC-based FSM is depicted in Fig. 3. The FSM's memory elements are

Table 1. List of transitions of the FSM chosen in the example

a_m	$X(a_m, a_s)$	a_s	$Y(a_m, a_s)$
a_1	x_1	a_2	y_1
a_1	$\overline{x}_1$	a_3	–
a_2	x_2	a_4	y_2, y_3
a_2	$\overline{x}_2$	a_2	y_2, y_3
a_3	x_3	a_5	y_2
a_3	$\overline{x}_3$	a_4	y_2
a_4	1	a_5	y_3
a_5	1	a_1	–

Note: a_m is the state corresponding to the beginning of transition (the present state), $X(a_m, a_s)$ is the set of input variables initiating the given transition (the transition condition), a_s is the transition state (the next state), $Y(a_m, a_s)$ is the set of output variables taking unit values at the given transition, unity in column $X(a_m, a_s)$ designates an unconditional transition, and hyphen in column $Y(a_m, a_s)$ corresponds to zero output set.

Table 2. Matrix **W** from the example

	d_1	d_2	d_3
	y_2	y_3	
a_1	–	–	1
a_2	1	1	0
a_3	1	0	0
a_4	0	1	0
a_5	0	0	0

Table 3. List of transitions in the merged model *AC* of the FSM from the example

a_m	$K(a_m)$ $e_1e_2e_3$	$X(a_m, a_s)$	a_s	$K(a_s)$ $d_1d_2d_3$	$Y(a_m, a_s)$	$Y(a_m)$
a_1	- - 1	x_1	a_2	1 1 0	y_1	–
a_1	- - 1	$\overline{x}_1$	a_3	1 0 0	–	–
a_2	1 1 0	x_2	a_4	0 1 0	–	y_2, y_3
a_2	1 1 0	$\overline{x}_2$	a_2	1 1 0	–	y_2, y_3
a_3	1 0 0	x_3	a_5	0 0 0	–	y_2
a_3	1 0 0	$\overline{x}_3$	a_4	0 1 0	–	y_2
a_4	0 1 0	1	a_5	0 0 0	–	y_3
a_5	0 0 0	1	a_1	- - 1	–	–

Note: Quantities a_m, $X(a_m, a_s)$, and a_s are defined in Table 1; $Y(a_m, a_s)$ are the Mealy machine's output variables of set Y_A; $Y(a_m)$ are the Moore machine's output variables of set Y_B; $K(a_m)$ is the code of FSM state a_m characterized by feedback variables e_1, e_2, and e_3; and $K(a_s)$ is the code of transition state a_s that specify the values of transition functions d_1, d_2, and d_3.

implemented using the D flip-flops of PLIC's output macrocells. Mealy machine's output variable y_1 (Fig. 3) is formed at the PLIC's combinational output. Moore machine's output variables y_2 and y_3 are formed at the PLIC's register outputs, and the feedback circuits of the corresponding output macrocells are utilized to transmit the values of feedback variables e_1 and e_2. In fact, feedback variables e_1 and e_2 are interpreted as output functions y_2 and y_3. Moreover, one PLIC macrocell with a register output is used to implement additional transition function d_3. Note that additional transition functions can be implemented with the help of buried PLIC macrocells that are not connected to external terminals.

For the above example, the list of transitions of merged model *AC* is presented in Table 3. Note that the Moore machine's output variables (Table 3, column $Y(a_m)$) are completely determined by the code $K(a_m)$ of FSM state a_m.

On the basis of Table 3, it is possible to construct the following logic equations for functions mechanized by combinational circuit *CL*:

$$y_1 = e_3x_1;$$

$$d_1 = e_3 + e_1e_2\overline{e}_3\overline{x}_2;$$

$$d_2 = e_3x_1 + e_1e_2\overline{e}_3 + e_1\overline{e}_2\overline{e}_3\overline{x}_3;$$

$$d_3 = \overline{e}_1\overline{e}_2\overline{e}_3.$$

To implement the given circuit, four PLIC macrocells are required, one of which can be buried. In the traditional approach, an initial FSM is built around six PLIC macrocells, three of which are used to mechanize output functions and three others are intended for transition functions. Moreover, the implemented functions became substantially less intricate. If the implementation cost of logic functions is measured by the number of gate inputs, the implementation cost of merged model *AC* is 24 in the example discussed above, and the corresponding value for all transition and output functions of an FSM is 46 in the traditional approach.

Thus, the proposed approach offers an advantage over the traditional technique because, in implementation of FSMs, not only the number of PLIC macrocells diminishes but also the FSM's combinational part can be simplified due to the enhanced complexity of the synthesis method without any additional expenses. As the final result, an increase in FSM speed is stimulated.

CONCLUSIONS

In implementation of the same finite-state machine based on programmable logic ICs, merging of two structural models—a class *A* Mealy machine and a class *C* Moore machine—is demonstrated to be possible if an absolute coincidence between the time diagrams of output signals is violated. In this case, Moore machine's output signals are formed before Mealy machine's output signals within the same clock cycle and advance time t_{PD} is the delay time of PLIC's combinational part. The theorem that specifies the conditions under which the FSM's output set can serve as the code (or part of the code) of an internal state has been proved. The proposed technique for synthesizing merged model *AC* makes it possible to implement PLIC-based FSMs having the lower cost and higher speed than traditional ones without any additional expenses. The use of the proposed technique is illustrated with the example where the implementation cost of the FSM's combinational part diminishes from 46 to 24 and the number of PLIC macrocells reduces from six to four. It can be assumed that a decrease in the implementation cost of the FSM's combinational part also diminishes the signal delay, i.e., enhances the FSM speed. The concrete values of a decrease in implementation cost and an

increase in FSM speed depend on the features of each FSM.

A promising direction of further investigations is the development of new merged structural models and FSM synthesis methods under violation of the condition of an absolute similarity between the time diagrams of output signals. Another promising direction is the development of new structural models of FSMs, which allow the efficient utilization of the properties of modern PLICs with progressive architectural capabilities.

ACKNOWLEDGMENTS

This study was supported in part by the Bialystok University of Technology, Poland, grant no. W/WI/11/07.

REFERENCES

1. E. J. McCluskey, Inform. Control, No. 6, 99 (1963).
2. I. Pomeranz and K.-T. Cheng, in *Proc. 29th ACM/IEEE Design Automation Conf. (DAC'92), Los Alamitos, June, 1992* (IEEE, New York, 1992), p. 573.
3. I. Pomeranz and K-T. Cheng, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **12**, 1123 (1993).
4. V. V. Solov'ev, Vesti Akad. Nauk Belarusi, Ser. Fiz.-Tekh. Nauk, No. 1, 68 (1994).
5. J. Forrest, in *Proc. European Design Automation Conf. with EURO-VHDL, Los Alamitos, Calif., 1995* (IEEE, New York, 1995), p. 600.
6. V. Solov'ev, in *Proc. EUROMICRO Symp. on Digital Systems Design (DSD'2001), Warsaw, Poland, Sept. 4–6, 2001* (IEEE Computer Society, Los Alamitos, Calif., 2001), p. 170.
7. V. V. Solov'ev and A. Klimovich, *Logic Design of Digital Systems Based on Programmable Logical Integrated Circuits* (Goryachaya Liniya-Telekom, Moscow, 2008) [in Russian].
8. V. V. Solov'ev and A. G. Vasil'ev, *Programmable Logical Integrated Circuits and Their Application* (Belorus. Nauka, Minsk, 1998) [in Russian].
9. G. H. Mealy, Bell Syst. Tech. J. **34**, 1045 (1955).
10. E. F. Moore, *Automata Studies*, Ed. by C. Shannon and J. McCarthy, (Princeton Univ. Press, Princeton, 1956).
11. *Programmable Logic* (Intel, 1994).