Matrix Implementation of Moore FSM with Encoding of Collections of Microoperations

A. Barkalov, L. Titarenko, O. Hebda, K. Soldatov

***Abstract* — The method is proposed for reduction of hardware amount in logic circuit of Moore finite state machine. The method is oriented on customized matrix technology. It is based on representation of the next state code as a concatenation of code for class of collection of microoperations and code of the vertex. Such an approach allows elimination of dependence among states and microoperations. As a result, both circuits for generation of input memory functions and microoperations are optimized. An example of the proposed method application is given.**

***Index Terms* — Customized matrices, graph-scheme of algorithm, logic circuit, Moore FSM, pseudoequivalent states.**

I. Introduction

The model of Moore finite state machine (FSM) [1] is often used during the digital control systems realization [2, 3]. The development of microelectronics has led to appearance of different programmable logic devices [4], used for implementing FSM circuits. But in the case of mass production, they use ASIC (Application-Specified Integrated Circuits) [6]. In this case the circuit is implemented using customized matrices using the principle of distributed logic [7].

One of the important problems of FSM synthesis with ASIC is decrease of the chip area occupied by its logic circuit. One of the ways to solve this problem is optimal coding of FSM [2]. However this approach does not allow optimization of the circuit generated output signals. In this work some new optimization method is proposed. It is based on representation of the next state code as a concatenation of codes for class of pseudoequivalent states and vertex where this collection is generated. Such an approach allows reducing of hardware amount in both parts of FSM circuits and does not lead to speed loss. A control algorithm to be implemented is represented by the graph-scheme of algorithms [1].

Manuscript received October 9, 2009. Matrix Implementation of Moore FSM with Encoding of Collections of Microoperations.

A. Barkalov is with the Institute of Computer Engineering and Electronics, Zielona Góra, Poland (corresponding author to provide phone: 0-683282693; e-mail: A.Barkalov@iie.uz.zgora.pl).

L. Titarenko is with the Institute of Computer Engineering and Electronics, Zielona Góra, Poland (e-mail: L.Titarenko@iie.uz.zgora.pl).

O. Hebda is with the Institute of Computer Engineering and Electronics, Zielona Góra, Poland (e-mail: O.Shapoval@iie.uz.zgora.pl).

K. Soldatov is with the Institute of Computer Engineering Donetsk National Technical University, Donetsk, Ukraine (e-mail: k.soldatov@cs.donntu.donetsk.ua).

II. The General Aspects and the Basic Idea of Proposed Method

Let Moore FSM be represented by the structure table (ST) with columns [1]: a_m, $K(a_m)$, a_s, $K(a_s)$, X_h, Φ_h, h. Here a_m is an initial state of FSM; $K(a_m)$ is a code of state $a_m \in A$ of capacity $R = \lceil \log_2 M \rceil$, to code the states the internal variables from the set $T = \{T_1,..,T_R\}$ are used; a_s, $K(a_s)$, are a state of transition and its code respectively; X_h is an input, which determines the transition $\langle a_m, a_s \rangle$, and equal to conjunction of some elements (or their complements) of a logic conditions set $X = \{x_1,..,x_L\}$; Φ_h is a set of input memory functions for flip-flops of FSM memory, which are equal to 1 for memory switching from $K(a_m)$ to $K(a_s)$, $\Phi_h \subseteq \Phi = \{\varphi_1,..,\varphi_R\}$; $h = 1, \dots, H$ is a number of transition. In the column a_m a set of microoperations Y_q is written, which is generated in the state $a_m \in A$, where $Y_q \subseteq Y = \{y_1,..,y_N\}$, $q = 1,...,Q$. This table is a basis to form the system of functions

$$\Phi = \Phi(T, X), \quad (1)$$

$$Y = Y(T), \quad (2)$$

which determines an FSM logic circuit. Systems (1)-(2) describe the matrix model of Moore FSM U_1, shown in Fig.1.

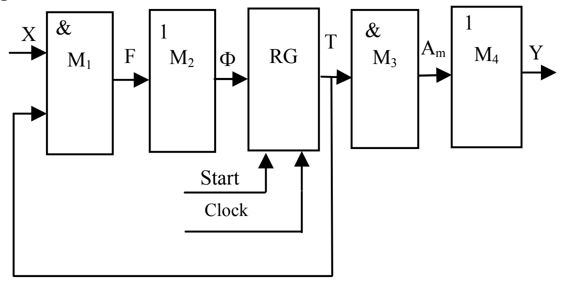


Fig. 1. Matrix implementation of FSM U_1

In FSM U_1 the conjunctive matrix M_1 implements the system of terms $F = \{F_1,...,F_H\}$; the disjunctive matrix M_2 implements the system (1); the conjunctive matrix M_3 implements the terms A_m ($m = 1,\dots,M$) corresponding to FSM states; the disjunctive matrix M_4 implements functions (2). The register RG keeps state codes. The matrices M_1 and M_2 forms the block of input memory functions (BIMF) whereas the matrices M_3 and M_4 the block of microoperations (BMO). The area of BIMF can be decreased using the approach of optimal state encoding [8]. It permits to decrease the number of terms in system (1) up to H_0, where H_0 is the number of transitions for equivalent Mealy FSM. The area of BMO can be decreased due to refined state encoding [9]. It is possible some outcome of encoding, when the matrix M_4 is absent. But these methods cannot be used together. In this article the method is proposed permitting mutual area decrease for both blocks of FSM.

One of Moore FSM features is existence of pseudoequivalent states [2], which are the states with the same transitions by the effect of the same inputs. Such states correspond to the control algorithm operator vertices [1], outputs of which are connected with an input of the same vertex.

Let $\Pi_A = \{B_1,..,B_I\}$ be a partition of a set A on classes of pseudoequivalent states. Let us code classes $B_i \in \Pi_A$ by binary codes $K(B_i)$ having R_B bits, where

$$R_B = \lceil \log_2 I \rceil. \qquad (3)$$

Let initial GSA Γ include Q different collections of microoperations (CMO) $Y_q \subseteq Y$. Let us code set Y_q with binary code $K(Y_q)$ having R_Y bits, where

$$R_Y = \lceil \log_2 Q \rceil. \qquad (4)$$

Let $E_1 = \{b_1,\dots,b_D\}$ be a set of operator vertices from GSA Γ. Let us use the following relation α on this set E_1

$$b_i \alpha b_j \leftrightarrow Y(b_i) = Y(b_j). \qquad (5)$$

In (5), the symbols $Y(b_i), Y(b_j) \subseteq Y$ stand for collections of MO from vertices b_i and b_j ($i, j \in \{1,...,D\}$).The relation α determines the partition $\Pi_\alpha = \{C_1,...,C_\eta\}$. Let us encode each vertex $b_q \in C_j$ by the binary code $K(b_q)$ having

$$R_\alpha = \lceil \log_2 G \rceil \qquad (6)$$

bits. In (6), $G = \max(|C_1|,...,|C_\eta|)$. Let us use variables $z_r \in Z_1$ for this encoding, where $|Z_1| = R_\alpha$. In this case, the code for state $a_m \in A$ can be represented as:

$$K(a_m) = K(Y_q) * K(b_q), \qquad (7)$$

where $b_q \in E_1$ is the operator vertex marked by state $a_m \in A$, $Y_q = Y(b_q)$, and * is the sign of concatenation.

Let us construct the system

$$B = B(A), \qquad (8)$$

which describes the dependence among the classes $B_i \in \Pi_A$ from the states $a_m \in A$. Each function $B_i \in B$ is represented as the following

$$B_i = \bigvee_{i=1}^{I} C_{im} A_m (i = 1,...,I), \qquad (9)$$

where the symbol C_{im} stands for Boolean variable equal to 1, $a_m \in B_i$. The proposed matrix implementation of Moore FSM U_2 is shown in Fig. 2.

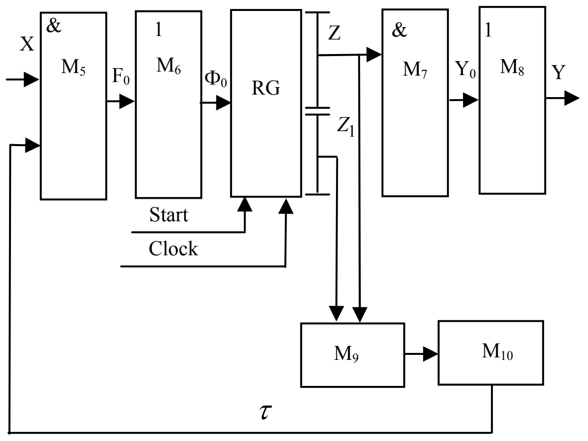


Fig. 2. Matrix implementation of FSM U_2

In FSM U_2, the matrix M_5 implements the system of terms F_0 corresponding to rows of transformed table of transitions and depending on logical conditions $x_l \in X$ and additional variables $\tau_r \in \tau$, used for encoding the classes $B_i \in \Pi_A$, where $|\tau| = R_B$. The matrix M_6 implements the input memory functions

$$\Phi_0 = \Phi_0(\tau, X), \qquad (10)$$

The system (10) includes $R_Y + R_\alpha$ functions; it is the number of flip-flops from RG. The matrix M_7 implements terms Y_0, entering the system $y_n \in Y$ and depending from variables $z_r \in Z$, where $|z| = R_Y$. The matrix M_8 implements functions $y_n \in Y$, depending on terms $\Delta_q \in Y_0$. The matrix M_9 implements the terms A_0 from (9), whereas the matrix M_{10} functions $\tau_r \in \tau$, used for encoding classes $B_i \in \Pi_A$, where $|\tau| = R_B$.

Matrices M_5 and M_6 form the block BIMF, the matrices M_7 and M_8 form the block BMO implementing the functions

$$Y = Y(Z). \qquad (11)$$

Matrices M_9 and M_{10} form the block of code transformer (BCT) generating functions

$$\tau = \tau(z, z_1). \quad (12)$$

There are some positive features in the proposed method. Now codes of collections of microoperations do not depend on state codes. It allows encoding of collections $Y_q \subseteq Y$ minimizing the area of BMO. The number of rows in the table of transitions for FSM U_2 is always equal to H_0. It allows such their encoding that diminishes the area occupied by BIMF. As it was mentioned, it is enough

$$R_A = \lceil \log_2 M \rceil \quad (13)$$

variables for state encoding in case of FSM U_1. The main drawback of U_2 is increase of the number of inputs for BIMF if the following condition is true:

$$R_Y + R_\alpha > R_A. \quad (14)$$

Besides, the model U_2 includes the block BCT, which requires some area of the chip. But these drawbacks are compensated by area decrease for blocks BIMF and BMO in comparison with the model U_1.

III. Proposed Synthesis Method for Moore FSM

In this work a method of Moore FSM U_2 synthesis using a GSA Γ is proposed. The method includes the next stages:

1. Marking of the GSA Γ and creation of the state set A.
2. Partition of the set A on classes of pseudoequivalent states.
3. Coding of microoperation collections $Y_q \subseteq Y$.
4. Construction of the partition Π_α and encoding of operator vertices $b_q \in E_1$.
5. Encoding the classes $B_i \in \Pi_A$.
6. Construction of transformed table of transitions.
7. Construction of system (12) by the table of BCT.
8. Implementation of matrices M_5 – M_{10}.

For the first step implementation the known method [1] is used, when every operator vertex is marked by a unique state. The second step is trivially done by the use of pseudoequivalent states' definition [2]. Remind, that states a_m, $a_s \in A$ are named pseudoequivalent, if marked by them operator vertices of GSA are connected with the input of the same vertex.

The main goal of the third step is maximum decrease for the number of terms in system Y_0. In the best case, each microoperation $y_n \in Y$ is represented by a single term and the matrix M_8 is absent [1]. The fourth step is executed on the base of (5). The codes of states $a_m \in A$ are determined using the formula (7). Classes $B_i \in \Pi_A$ are encoded in such a manner that the number of terms in (12) is maximally decreased. It is reduced to the well-known task of symbolic encoding [3].

The transformed table of transitions includes the columns B_i, $K(B_i)$, a_s, $K(a_s)$, X_k, Φ_k, h. Here $\Phi_h \subseteq \Phi_0$ is a collection of input memory functions equal to 1 to write the code $K(a_s)$ into the register; $h = 1,\ldots,H_0$ is the number of transition. The table of BCT includes the columns a_m, $K(a_m)$, B_i, $K(B_i)$, τ_m, m. Here $\tau_m \subseteq \tau$ is the collection of variables equal to 1 into the code $K(B_i)$ from the m-th line of the table, where $m = 1,\ldots,M$. The last step is discussed in the proposed example.

IV. Example of Application for Proposed Method

Let the symbol $U_i(\Gamma_j)$ means that the GSA Γ_j is interpreted by the model U_i ($i = 1,2$). Let us discuss the example of design for Moore FSM $U_2(\Gamma_1)$, where GSA Γ_1 is shown in Fig. 3.

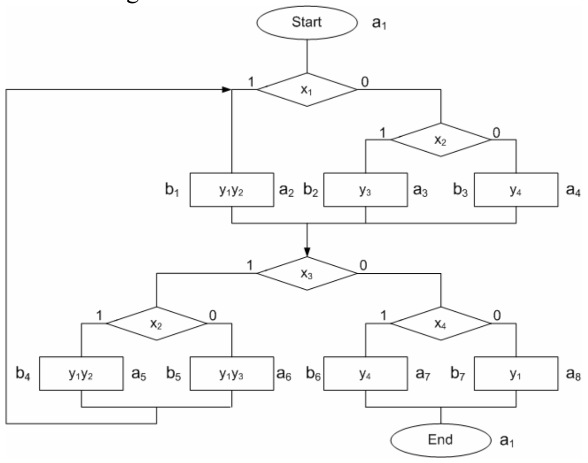
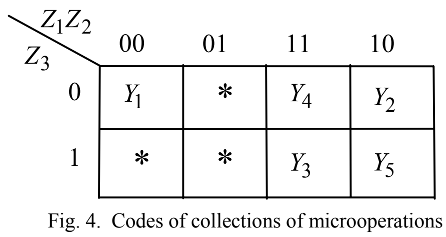


Fig. 3. Initial graph-scheme of algorithm Γ_1

It can be found from GSA Γ_1, that $A = \{a_1,\ldots,a_8\}$, $M = 8$, and $R_A = 3$. There is the partition $\Pi_A = \{B_1,\ldots,B_4\}$, where $B_1 = \{a_1\}$, $B_2 = \{a_2, a_3, a_4\}$, $B_3 = \{a_5, a_6\}$, $B_4 = \{a_7, a_8\}$. It gives us $I = 4$, $R_B = 2$, $\tau = \{\tau_1, \tau_2\}$. There are five different collections of microoperations in GSA Γ_1: $Y_1 = 0$, $Y_2 = \{y_1, y_2\}$, $Y_3 = \{y_3\}$, $Y_4 = \{y_4\}$, $Y_5 = \{y_1, y_3\}$. To encode them, it is enough $R_Y = 3$ variables from the set $Z = \{z_1, z_2, z_3\}$. Let us encode the collections $Y_q \subseteq Y$ as it is shown in Fig. 4.



Z_3 \ Z_1Z_2	00	01	11	10
0	Y_1	*	Y_4	Y_2
1	*	*	Y_3	Y_5

Fig. 4. Codes of collections of microoperations $U_2(\Gamma_1)$

The following system of equations can be obtained using Fig. 3 and Fig.4:

$$
\begin{aligned}
&y_1 = Y_2 \vee Y_5 = z_2\bar{z}_3 = \Delta_1;\\
&y_2 = Y_2 = \bar{z}_1 z_2 \bar{z}_3 = \Delta_2;\\
&y_3 = Y_3 \vee Y_5 = z_1 = \Delta_3;\\
&y_4 = Y_4 = \bar{z}_1 z_3 = \Delta_4;
\end{aligned} \qquad (15)
$$

The partition Π_α includes four classes: $C_1 = \{b_1, b_4, b_7\}$, $C_2 = \{b_2\}$, $C_3 = \{b_3, b_6\}$, $C_4 = \{b_5\}$. It gives $G = 3$, $R_\alpha = 2$, $Z_1 = \{z_4, z_5\}$. There is the following system (8) in our example:

$$
\begin{aligned}
&B_1 = A_1; B_2 = A_2 \vee A_3 \vee A_4;\\
&B_3 = A_5 \vee A_6; B_4 = A_7 \vee A_8.
\end{aligned} \qquad (16)
$$

Let us encode the vertices $b_q \in E_1$ in such a manner that the state codes (7) are determined from Fig.5.

$Z_4Z_5 \backslash Z_1Z_2Z_3$	000	001	011	010	110	111	101	100
00	a_1	*	a_4	a_2	*	a_3	*	*
01	*	*	*	a_5	a_6	*	*	*
11	*	*	*	*	*	*	*	*
10	*	*	a_8	a_7	*	*	*	*

Fig. 5. State codes for Moore FSM $U_2(\Gamma_1)$

The following codes can be found from the Karnaugh map (Fig. 5):

$$
\begin{aligned}
&B_1 = \bar{z}_2; B_2 = z_2\bar{z}_4\bar{z}_5;\\
&B_3 = z_5; B_4 = z_4.
\end{aligned} \qquad (17)
$$

Let us encode the classes $B_i \in \Pi_A$ in the following manner $K(B_1) = 01$, $K(B_2) = 00$, $K(B_3) = 10$, $K(B_4) = 11$. The following system can be derived from these codes:

$$
\begin{aligned}
&\tau_1 = B_3 \vee B_4 = z_5 \vee z_4;\\
&\tau_2 = B_1 \vee B_4 = \bar{z}_2 \vee z_4.
\end{aligned} \qquad (18)
$$

The system (18) determines the block BCT, where the matrix M_9 is absent.

Let us construct the system of generalized formulae of transitions for GSA Γ_1:

$$
\begin{aligned}
&B_1 \to x_1a_2 \vee \bar{x}_1x_2a_3 \vee \bar{x}_1\bar{x}_2a_4;\\
&B_2 \to x_3x_2a_5 \vee x_3\bar{x}_2a_6 \vee \bar{x}_3x_4a_7 \vee \bar{x}_3\bar{x}_4a_8;\\
&B_3 \to a_2; B_4 \to a_1.
\end{aligned} \qquad (19)
$$

This system together with state codes from Fig. 5 leads to the transformed table of transitions for FSM $U_2(\Gamma_1)$, having $H_0 = 9$ lines (Table 1).

TABLE I

Transformed Table of Transitions for FSM $U_2(\Gamma_1)$

B_i	$K(B_i)$	a_s	$K(a_s)$	X_h	Φ_h	h
B_1	01	a_2	01000	x_1	D_2	1
		a_3	11100	$\bar{x}_1x_2$	$D_1D_2D_3$	2
		a_4	01100	$\bar{x}_1\bar{x}_2$	D_2D_3	3
B_2	00	a_5	01001	x_3x_2	D_2D_5	4
		a_6	11001	$x_3\bar{x}_2$	$D_1D_2D_5$	5
		a_7	01010	$\bar{x}_3x_4$	D_2D_4	6
		a_8	01110	$\bar{x}_3\bar{x}_4$	$D_2D_3D_4$	7
B_3	10	a_2	01000	1	D_2	8
B_4	11	a_1	00000	1	-	9

This table is used to derive the system (10). For example, the following functions can be found from Table 1: $D_1 = F_2 \vee F_5 = \bar{\tau}_1\tau_2\bar{x}_1x_2 \vee \bar{\tau}_1\bar{\tau}_2x_3\bar{x}_2$; $D_2 = F_2 \vee ... \vee F_8$; $D_3 = F_2 \vee F_3 \vee F_7$; $D_4 = F_6 \vee F_7$; $D_5 = F_4 \vee F_5$. In the case of matrix implementation, there is no need in minimizing these functions. The table for BCT is absent on our example because the system (18) determines the functions (12). Let us find the areas for matrices M_5 – M_{10}, determined as the product for the numbers of inputs and outputs of the matrix. From system of functions we can find the following areas of matrices: $S(M_5) = 2(4+2)*9 = 108$, $S(M_6) = 9*5 = 45$, $S(M_7, M_8) = 5*4 = 20$ and $S(M_9, M_{10}) = 3*2 = 6$. Thus, it is necessary 179 area units [1] to implement the logic circuit of Moore FSM $U_2(\Gamma_1)$. It can be found for FSM $U_1(\Gamma_1)$ that $H = 19$, $S(M_1) = 2(4+3)*19 = 166$, $S(M_2) = 19*3 = 57$, $S(M_3) = 2*3*7 = 42$ and $S(M_4) = 7*4 = 28$. It means that logic circuit of FSM $U_1(\Gamma_1)$ occupies 293 area units. Besides, the circuit of $U_1(\Gamma_1)$ has 4 levels of logic, whereas the circuit for $U_2(\Gamma_1)$ only three (because functions τ and Y are generated in the same time). Thus, application of proposed method for encoding of collections of microoperations with state code presentation in the form (7) allows area decrease for 1,7 times Moore FSM.

V. Conclusion

The proposed method of state code presentation targets on area decrease under implementation of Moore FSM logic circuit with customized matrices. This approach allows decreasing the number of terms in the system of input memory functions up to corresponding value of the equivalent Mealy FSM. Besides, this method permits decreasing the number of terms in the system of microoperations due to the lack of dependence among the state codes and codes of collections of microoperations.

Investigation for effectiveness of proposed method was conducted on the standard examples [10]. It shows that the proposed method permits to decrease the average chip area occupied by FSM circuit up to 52% in comparison with the standard FSM implementation. In the same time, it was the increase for FSM performance in 86% of examples. The further direction of our research is application of proposed method for case of FPGA.

References

[1] Baranov S., *Logic Synthesis for Control Automata*, Kluwer Academic Publishers, Boston, 1994.

[2] A. Barkalov, L. Titarenko, *Logic Synthesis for FSM-Based Control Units.* Springer - Berlin Verlag Heidelberg, Lectures Notes in Electrical Engineering, 2009, №53.

[3] DeMicheli *G. Synthesis and Optimization of Digital Circuits*, McGraw-Hill, NY, 1994.

[4] Maxfield C. *The Design Warrior's Guide to FPGAs*, Elseveir, Amsterdam, 2004.

[5] Smith M. *Application-Specific Integrated Circuits*, Addison-Wesley, Boston, 1997.

[6] Nababi Z. Embedded *Core Design with FPGA*, McGraw-Hill, NY, 2008.

[7] Yang S. *Logic Synthesis and Optimization Benchmarks user guide. Technical report*, №1991 – IWLS-UG-Saryang.-Microelectronics center of North Carolina.

Prof. dr hab. inż. Alexander BARKALOV, prof. UZ. Prof. Alexander A. Barkalov worked in Donetsk National Technical University (DNTU) from 1976 till 1996 as a tutor . He cooperated actively with Kiev Institute of Cybernetics (IC) named after Victor Glushkov. He got his degree of doctor of technical sciences (Informatics) in 1995 from IC. From 1996 till 2003 he worked as a professor of DNTU. From 2003 he has been working as a professor on Department of Electrotechnics, Informatics and Telecommunications of University of Zielona Gora. The area of his scientific activity is design methods for control units implemented with FPGA, CPLD, and ASIC.

Dr hab. inż. Larysa TITARENKO, prof. UZ. Prof. Larysa Titarenko has got her degree of doctor of technical sciences (Telecommunications) in 2005 from Kharkov National University of Radioelectronics (KNURE). Till September, 2003 she worked as a professor of KNURE. From 2005 she has been working as a professor on Department of Electrotechnics, Informatics and Telecommunications of University of Zielona Gora. The area of her scientific activity is design methods for control units implemented with FPGA, CPLD, and ASIC, as well as their application in telecommunications.

MA Olena Hebda. MA Olena Hebda studied at National aerospace university 'Kharkiv Aviation Institute' from 2003 till 2009 and has received higher education on speciality "Biotechnical and Medical Apparatuses and Systems" and qualification of the research engineer (electronics, telecommunications). In 2009 she has started PhD study on Department of Electrotechnics, Informatics and Telecommunications of University of Zielona Gora. The area of her scientific activity is design methods for control units implemented with FPGA, CPLD, and ASIC.

MA Kirill Soldatov. MA Kirill Soldatov studied at Donetsk National Technical University (DNTU) from 200 till 2010 and has received higher education on speciality "Computer systems and networks". In 2010 he has started PhD study on the chair of computer engineering of DNTU. The area of his scientific activity is design methods for control units implemented with FPGA, CPLD, and ASIC.