Synthesis of Moore finite state machine with nonstandard presentation of state codes

Alexander Barkalov, Larysa Titarenko, Olena Hebda

Abstract: The method is proposed for reduction of hardware amount in logic circuit of Moore finite state machine. The method is oriented on CPLD technology. It is based on representation of the next state code as a concatenation of codes for class of pseudoequivalent states and collection of microoperations. 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.

Key words: Moore FSM, CPLD, pseudoequivalent states, classes, PAL macrocells, hardware reduction

1. 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], one of which are CPLD (complex programmable logic devices) [6 — 8]. The base of CPLD is a macrocell PAL (programmable array logic), the cells are connected by the programmable array of interconnections. One of the important problems of FSM synthesis on CPLD is a minimization of macrocells' number of its logical circuit. One of the ways to solve this problem is optimal states' coding [2]. However this approach does not allow optimization of the output signals generation circuit. In this work an 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 collection of microoperations. Such an approach allows reducing of hardware amount in FSM circuits and does not lead to speed loss.

2. THE GENERAL ASPECTS AND THE BASIC IDEA OF A 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 $K(a_s)$, $\Phi_h \subseteq \Phi = \{\varphi_1,..,\varphi_R\}$; $h = 1, \ldots, H$ 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

$$\Phi = \Phi(T, X), \tag{1}$$

$$Y = Y(T), \tag{2}$$

which determines an FSM logic circuit. Systems (1)-(2) describe the model of Moore FSM U_1, shown in Fig.1. In this model a block of input memory functions (BIM) realizes the system (1), a register RG keeps states' codes, block of microoperations (BMO) realizes the system (2). To minimize the amount of macrocells PAL in BIM, the optimal states' coding method can be used [2]. It allows decreasing the amount of terms in system (1) up to H_0.

Here H_0 is the number of transitions of equivalent Mealy FSM. Optimization for the block BMO can be done by the use of refined state coding [2]. Meanwhile, the number of macrocells can be decreased up to N; it corresponds to situation, when every function $y_n \in Y$ is realized on single macrocell. For optimal and refined states' coding, some well- known algorithms [3] can be used. But both methods cannot be used at the same time. That is why states' coding allows optimizing either BIM or BMO.

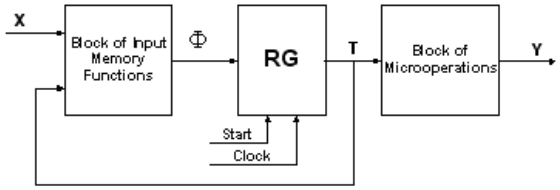


Fig. 1. Structural diagram of Moore FSM U_1

In this paper a method for reduction of hardware amount in both logic circuits of Moore finite state machine is proposed.

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

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

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

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

Let an operator vertex b_t of GSA Γ correspond to a state $a_m \in B_i$ and let a collection of microoperations Y_q be written in this vertex. Then the code of state $a_m \in A$ can be represented as a concatenation of codes

$$K(a_m) = K(B_i) * K(Y_q), \quad (5)$$

where * is a concatenation sign.

Such a representation allows presenting Moore FSM as a model U_2 (Fig.2).

In FSM U_2 the block BIM realizes functions (6) and the block BMO functions (7).

$$\Phi = \Phi(\tau, X), \quad (6)$$

$$Y = Y(Z). \quad (7)$$

Here all elements of the set $\tau = \{\tau_1,...,\tau_{R_B}\}$ are used to code the classes $B_i \in \Pi_A$, and all elements of the set $Z = \{z_1,...,z_{R_Y}\}$ are used to code the collections of microoperations.

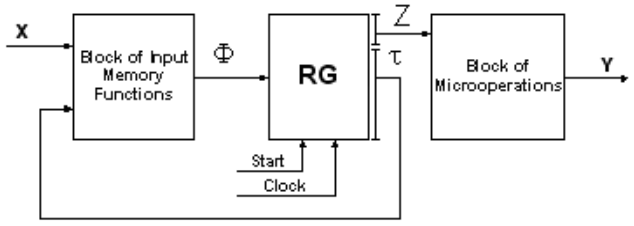


Fig. 2. Structural diagram of Moore FSM U_2

Offered approach has the next advantages:

1. Codes of classes do not depend on codes of microoperations (and vice versa). It allows a classes coding oriented on simplification of the BMO logic circuit.

2. The number of rows of transition table of FSM U_2 is always equal to H_0 and does not depend on the state coding method.

To code the states of FSM U_1 is enough to have

$$R_A = \lceil \log_2 M \rceil \quad (8)$$

bits. The evident disadvantage of FSM U_2 is an increasing of the number of functions Φ, which is determined as a sum of R_B and R_Y. Taking into account that all modern CPLD have registered outputs [6, 7], offered method does not increase the number of macrocells, demanded to flip-flop realization.

3. Synthesis method of Moore FSM with extension of state codes

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 set A.
2. Partition of the set A on classes of pseudoequivalent states.
3. Coding of the classes $B_i \in \Pi_A$.
4. Coding of microoperation collections $Y_q \subseteq Y$.
5. Construction of the transformed structure table of Moore FSM.
6. Construction of the Boolean functions for the system Φ.
7. Construction of the BMO table.
8. Construction of the Boolean functions for the system Y.
9. The synthesis of FSM logic circuit with given logic elements.

For the first step of the implementation algorithm the known method [1] is used, when every operator vertex is marked by a unique state. Let us point out that both start and final vertices are marked by the same state, which is the initial state of FSM. This state always should be encoded by all zeros.

The second step is trivially done by the use of pseudoequivalent states' definition [2]. Remind, that states $a_m, a_s \in A$ are called pseudoequivalent states, if marked by them operator vertices of GSA are connected with the input of the same vertex.

The coding of classes $B_i \in \Pi_A$ does not effect on the length of structure table, because it is guarantee equal to its minimal possible amount, H_0. That is why the class codes $K(B_i)$ can be chosen arbitrary. The coding method for collections of microoperations depends on logic elements used for the block BMO realization. If PLA cells are used, then coding is targeted on their number reduction.

If macrocells PAL are used for implementation, then the coding is oriented on the minimization of the number of terms for functions $y_n \in Y$. The known algorithms ESPRESSO [3] can be used to solve this problem.

If embedded memory blocks are used for implementation, then coding is oriented on the decrease for the number of terms in functions Φ. The frequency approach [1] can be used in this case: the more operator vertices contain the set Y_q, the less 1's its code includes.

The transformed ST includes the columns B_i, $K(B_i)$, a_s, $K(a_s)$, X_h, Φ_h, h. The purpose of each column is obvious. The table of block BMO includes the columns $K(Y_q)$, Y_q, q, where q is the number of the table line.

The steps 6-9 are thoroughly discussed in the literature [2] and they are beyond the scope of our article.

4. EXAMPLE OF APPLICATION OF PROPOSED METHOD

Let us discuss an example for a Moore FSM U_2 (Γ_1) synthesis, where a GSA Γ_1 is shown in Fig.3. The symbol $U_i(\Gamma_j)$ stands for interpretation of a GSA Γ_j with an FSM model U_i. As follows from Fig.3, the set A includes $M=8$ elements distributed among $I=4$ classes $B_1=\{a_1\}$, $B_2=\{a_2,a_3,a_4\}$, $B_3=\{a_5,a_6\}$, $B_4=\{a_7,a_8\}$.

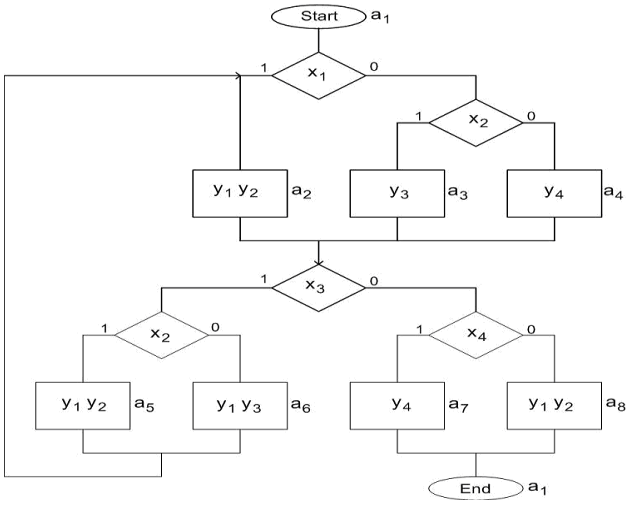


Fig. 3. Initial GSA Γ_1

It is enough $R_B=2$ variables $\tau_r\in\tau$ for coding of the classes $B_i\in\Pi_A$. Therefore, $\tau=\{\tau_1,\tau_2\}$. There are $Q=5$ different collections of microoperations in the operator vertices of the GSA Γ_1, namely $|Y_1|=0$, $Y_2=\{y_1,y_2\}$, $Y_3=\{y_3\}$, $Y_4=\{y_4\}$, $Y_5=\{y_1,y_3\}$. It is enough $R_Y=3$ variables for coding of these collection, it means that $Z=\{z_1,z_2,z_3\}$. Let us use PAL macrocells for implementation of the block BMO. In this case the coding approach is used to decrease the number of terms in the system (7). But it is enough only one PAL macrocell for implementation of any function $y_n\in Y$ in our example. Because of it, let us use the frequency approach for collection coding to decrease the complexity of the block BMO.

Let n_q be the number of vertices including the collection $y_q\in Y$. In our example, there are $n_1=1$, $n_2=3$, $n_3=1$, $n_4=2$, $n_5=1$. It means that the collection can be coded in the following manner: $K(Y_2)=000$, $K(Y_4)=001$, $K(Y_1)=010$, $K(Y_3)=100$, $K(Y_5)=101$. Let us use the frequency approach for encoding of the classes $B_i\in\Pi_A$: The more states the class includes, the more zeros its code includes. The following codes can be found in our case: $K(B_1)=11$, $K(B_2)=00$, $K(B_3)=01$, $K(B_4)=10$.

Now the expanded codes of states (5) can be found. For example, we can get that

$$K(a_1)=K(B_1)*K(y_1)=11010,$$

$$K(a_2)=K(B_2)*K(y_2)=00000$$

and so on. These codes are used in the column a_s of transformed table (Table 1).

Table 1

Specification of block of input memory functions

B_i	$K(B_i)$	a_s	$K(a_s)$	X_h	Φ_h	h
B_1	11	a_2	00000	x_1	-	1
		a_3	00100	$\overline{x_1}x_2$	D_3	2
		a_4	00001	$\overline{x_1}\,\overline{x_2}$	D_5	3
B_2	00	a_5	01000	x_3x_2	D_2	4
		a_6	01101	$x_3\overline{x_2}$	$D_2D_3D_5$	5
		a_7	10001	$\overline{x_3}x_4$	D_1D_5	6
		a_8	10000	$\overline{x_3}\,\overline{x_4}$	D_1	7
B_3	01	a_2	00000	1	-	8
B_4	10	a_1	11010	1	$D_1D_2D_5$	9

The terms F_h corresponding to the lines of this table are determines as

$$F_h=\bigwedge_{r=1}^{R_B}\tau_r^{l_{rh}}\cdot X_h(h=1,...,H_0), \qquad (9)$$

where $l_{rh}\in\{0,1\}$ is the value of the bit r for the code $K(B_i)$ from the line h of the table.

The system (6) is derived from this table. For example, the following equation $D_1=F_6\vee F_7\vee F_9=\overline{\tau_1}\,\overline{\tau_2}x_3\vee\tau_1\overline{\tau_2}$ can be found after some minimization. Let us point out that the transformed ST of Moore FSM U_1 (Γ_1) includes $H=19$ lines, that is two times more than in the case of FSM U_2 (Γ_1). The table of block BMO includes $Q=5$ lines (Table 2).

This table can be used as $N=4$ Karnaugh maps with insignificant input assignments 011, 110 and 111. After minimizing, the system (7) is following one: $y_1=\overline{z_1}\,\overline{z_2}\,\overline{z_3}\vee z_1z_3$; $y_2=\overline{z_1}\,\overline{z_2}\,\overline{z_3}$; $y_3=z_1$; $y_4=\overline{z_1}z_3$.

As it mentioned before, there are no problems with logic circuit design on the base of systems (6) — (7). The design can be done using, for example, WebPack package of Xilinx. The steps of design are standard ones. First of all, the FSM model should be created using some hardware description language, for example, Verilog or VHDL. Our investigations show that a language in use does not affect the final result of design. Merely, the first of these languages, Verilog, is more used in USA, whereas the second in Europe. Next, the systems of equations should be used and the model should be tuned for a particular control algorithm. It means that such important parameters of FSM as the number of state variables, number of microoperations and so on should replace their symbols in a model. But all these procedures are standard ones for an up-to-day designer of digital devices with programmable logic devices. Because of it, we do not discuss this problem in our article.

Table 2
Specification of block BMO

$K(Y_q)$	000	001	010	100	101
Y_q	y_1y_2	y_4	-	y_3	y_1y_3
q	2	4	1	3	5

5. CONCLUSION

The proposed method of state codes presentation is oriented on decrease for the number of macrocells PAL in the logic circuit of Moore FSM. This approach allows decrease for the number of terms in the system of input memory functions up to corresponding value of equivalent Mealy FSM. Our researches show that the proposed method always is more effective than the classical method of Moore FSM design. In average, the number of PAL macrocells is up to 38% less than for the classical approach.

This reduction is not accompanied with decrease for FSM performance, because there are no new blocks in the FSM model. Moreover, the reduction for macrocells is often connected with decrease for the number of levels in the FSM logic circuit. It results in increase for FSM performance, as well as for controlled digital system.

Our researches show that this approach can be applied successfully in the following three cases:

1. Moore FSM is implemented using FPGA chips. In this case look-up table (LUT) elements are used for implementation of the circuit for input memory functions. It is known that LUT have very limited number of inputs (up to 6). Our approach permits decreasing the number of feedback variables, because state variables are replaced by class variables used to encode classes of pseudoequivalent states. In the same time the block of microoperations is implemented using embedded memory blocks. Using our approach, the number of address inputs for memory blocks can be decreased. In this case we need fewer amounts of memory blocks for implementing BMO than in a classical case.

2. Moore FSM is implemented with CPLD chips based on programmable array logic (PAL) technology. In this case PAL cells are used for implementation of both blocks of FSM. These cells have limited number of terms per cell. Our approach permits decrease for the number of terms in the sum-of-product for each input memory function to be implemented. It permits decreasing the total number of cells in the circuit of BIM. Moreover, the codes of sets of microinstructions for each state do not depend on the states codes or codes of classes of pseudoequivalent states. It means that we can choose such codes of these sets that the number of PAL cells in BMO is minimal. The well-known algorithm ESSPRESSO can be used to solve this problem.

3. Moore FSM is implemented with CPLD chips based on programmable logic array (PLA) technology. In this case PLA cells are used for implementation of both blocks of FSM. In this case one cell can implement more than one function. These cells have limited number of terms, too. Our approach permits to decrease the number of terms for system of input memory functions to be implemented and the needed hardware amount.

LITERATURE

[1] Baranov S. *Logic Synthesis for Control Automata.* Boston: Kluwer Academic Publishers, 1994. 312 pp,

[2] *A. Barkalov, L. Titarenko*, "Logic Synthesis for FSM-Based Control Units. Springer", Berlin Verlag Heidelberg, *Lectures Notes in Electrical Engineering*, 2009, №53. 233 pp.

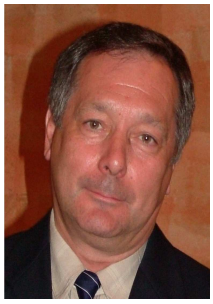
[3] DeMicheli G. *Synthesis and Optimization of Digital Circuits*. NJ: McGraw-Hill, 1994. 636 pp..

[4] Maxfield C. *The Design Warrior's Guide to FPGAs.* Amsterdam: Elseveir, 2004. 541 pp.

[5] Altera Corporation Webpage http// www.altera.com

[6] Atmel Corporation Webpage http// www.atmel.com

[7] Xilinx Corporation Webpage http// www.xilinx.com



prof. dr hab. Inż. Alexander BARKALOV, prof. UZ
University of Zielona Gora
Department of Electrotechnics, Informatics and Telecommunications,
Institute Informatics and Electronics
Licealna str.9 , Zielona Gora, 65-417,
Poland
+48 068 243 58 94
e-mail: A.Barkalov@iie.uz.zgora.pl

dr hab. Inż. Larysa TITARENKO, prof. UZ
University of Zielona Gora
Department of Electrotechnics, Informatics and Telecommunications
Institute Informatics and Electronics
Licealna str.9 , Zielona Gora, 65-417,
Poland
+48 068 243 58 94
e-mail: L.Titarenko@iie.uz.zgora.pl

mgr inż. Olena Hebda
University of Zielona Gora
Department of Electrotechnics, Informatics and Telecommunications
Institute Informatics and Electronics
Licealna str.9 , Zielona Gora, 65-417,
Poland
+48 068 243 58 94
e-mail: O.Shapoval@weit.uz.zgora.pl