Alexander BARKALOV, Larysa TITARENKO, Olena HEBDA
UNIVERSITY OF ZIELONA GORA, ul. Podgórna 50, 65-246 Zielona Góra

Matrix implementation of Moore FSM with expansion of coding space

Prof. dr hab. inż. Alexander 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.

e-mail: A.Barkalov@iie.uz.zgora.pl

Mgr inż. Olena HEBDA

Olena Hebda studied at National Aerospace University 'Kharkiv Aviation Institute' from 2003 till 2009 and she 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.

e-mail: O.Shapoval@weit.uz.zgora.pl

Dr hab. inż. Larysa TITARENKO

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.

e-mail: L.Titarenko@iie.uz.zgora.pl

Abstract

The proposed method is targeted on reduction of hardware amount in logic circuit of Moore finite-state machine implemented with customized matrices. The method is based on using more than minimal amount of variables in codes of FSM internal states. The method includes two stages of state encoding. The second stage is connected with recoding of states inside each class of pseudoequivalent states. An example is given for proposed method application.

Keywords: Moore FSM, graph-scheme of algorithm, pseudoequivalent states, customized matrices, logic circuit.

Macierzowa implementacja automatu Moore'a z rozszerzeniem przestrzeni kodowania

Streszczenie

Zaproponowana metoda jest zorientowana na redukcję zasobów sprzętowych potrzebnych do implementacji skończonego automatu stanu typu Moore'a implementowanego w układach o strukturze macierzowej. Metoda wykorzystuje dwuetapowe kodowanie stanów, w którym liczba zmiennych jest większa od minimalnej. W pierwszym etapie realizowane jest optymalne kodowanie stanów dla klas stanów pseudorównoważnych. Poszczególne stany są reprezentowane jako pojedynczy unikalny interwał boolowskiej przestrzeni kodów. Etap ten jest konieczny do zoptymalizowania układu realizującego funkcje wejść. W drugim etapie zamieniana jest kolejność stanów w ramach poszczególnych klas stanów pseudorównoważnych, co pozwala na optymalizację powierzchni macierzy implementującej funkcje wyjść. Proponowana metoda może zostać użyta w układach CPLD z komórkami PAL i PLA oraz w układach FPGA. W artykule przedstawiono także przykład zastosowania proponowanej metody.

Słowa kluczowe: automat typu Moore'a, sieć działań, stany pseudorównoważne, układ logiczny.

1. Introduction

The model of Moore finite state machine (FSM) [1] is often used during the digital control systems realization [3, 6]. The development of microelectronics has led to appearance of different programmable logic devices [3], which are used for implementing FSM logic circuit. But in the case of mass production of microelectronics products, they widely use so called customized circuits called ASIC (Application-Specified Integrated Circuits) [5]. In the case of ASIC, the logic circuits of FSM are designed, as a rule, using so called matrix structures. In these customized matrices, the principle of distributed logic is used [4].

One of the important problems of FSM synthesis with matrix structures is the decrease in chip space occupied by FSM logic circuit. One of the ways to solve this problem is optimal coding of FSM internal states [2]. However, this approach does not allow optimization of the output signals generation part of FSM circuit.

In this work the optimization method is proposed. It is based on encoding of state codes using more than minimal amount of variables. These codes can be rearranged to optimize the matrix area for the part of the circuit implementing the system of microoperations. Such an approach allows reducing of hardware amount in FSM circuits and does not lead to speed loss. The reducing of hardware is connected with compressing the FSM table of transitions. The fewer rows this table includes, the less amount of terms the implemented systems of Boolean functions includes, too. In this article a control algorithm to be implemented is represented by a graph-scheme of algorithm (GSA) [1].

2. The general aspects and 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 $a_m \in A$ having the capacity $R = \lceil \log_2 M \rceil$ (to code the states the internal variables from the set $T=\{T_1,\dots,T_R\}$ 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 it is equal to conjunction of some elements (or their complements) of a set of logic conditions $X = \{x_1,..,x_L\}$; Φ_h is a set of input memory functions for flip-flops of FSM memory, which are equal to 1 to change the content of the memory from $K(a_m)$, to $K(a_s)$. $\Phi_h \subseteq \Phi = \{\varphi_1,..,\varphi_R\}$; h=1,…H is a number of transition. In the column a_m a set of microoperations Y_q is written, these microoperations are generated in the state $a_m \in A$ ($Y_q \subseteq Y = \{y_1,..,y_N\}$, $q = 1,...,Q$). This table is a basis to form the following systems of functions:

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

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

These systems specify an FSM logic circuit. In the case of matrix implementation, systems (1)-(2) determine the model of Moore FSM U_1, shown in Fig.1.

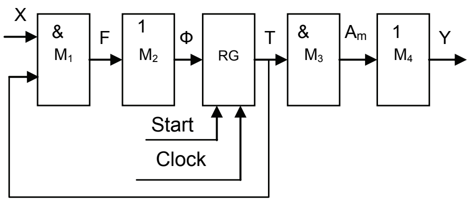

Fig. 1. Matrix circuit of Moore FSM U_1
Rys. 1. Macierzowy układ automatu Moore'a U_1

In the FSM U_1, a conjunctive matrix M_1 implements the system of terms $F=\{F_1,\ldots,F_H\}$; a disjunctive matrix M_2 implements system (1); a conjunctive matrix M_3 implements terms A_m ($m=1,\ldots,M$), corresponding to FSM states; a disjunctive matrix M_4 implements system (2). The register RG keeps state codes; it is controlled by signals Start (clearing) and Clock (changing content depending on functions Φ). The matrices M_1 and M_2 determine the block of input memory functions (BIM), whereas the matrices M_3 and M_4 determine the block of microoperations (BMO). The method of optimal state encoding [2] may be used for reducing the number of terms in system (1) up to H_0. Here the symbol H_0 stands for the number of transitions for the equivalent Mealy FSM. An area of BMO may be decreased due to refined state encoding [2]. As an extreme solution, it permits to specify each microoperation $y_n \in Y$ by a single term of M_3 and the matrix M_4 is absent. Unfortunately, the mentioned methods cannot be applied simultaneously. It means that it is possible to decrease either a chip area for BIM, or for BMO. In this article we propose a method, which allows optimization for both blocks, BIM and BMO.

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 $\Pi_A = \{B_1,\ldots,B_I\}$ 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

$$R_B = \lceil \log_2 I \rceil . \tag{3}$$

In the general case the following condition takes place:

$$R_B < R_A . \tag{4}$$

In this article we propose to represent the states by binary codes $K(a_m)$ in such a way that it allows represent the codes $K(B_i)$ of classes $B_i \in \Pi_A$ using variables $T_r \in T' \subset T$. It results in Moore FSM U_2. The structures of both U_1 and U_2 are the same. But in U_2 the block BIM implements functions

$$\Phi = \Phi(T', X) . \tag{6}$$

Such an approach guarantees the reduction of the number of terms in system Φ up H_0. It leads to reducing area of matrix M_1, as well as the number of inputs for matrix M_2. Unfortunately, this approach does not permit terms reduction in system (2).

Let us represent the coding space as a Karnaugh map having I_K columns, where

$$I_K = 2^{R_B} . \tag{7}$$

Let $|B_i| = m_i$ and $M_O = \max(m_1,\ldots,m_I)$, then the map should have

$$O_K = 2^{R_O} \tag{8}$$

rows, where

$$R_O = \lceil \log_2 M_O \rceil . \tag{9}$$

If condition (8) is true, then the states for any class $B_i \in \Pi_A$ are placed in the same column of the map and the code $K(B_i)$ is determined by the values of variables marking this column. If the following condition

$$R_B + R_O > R_A \tag{10}$$

takes place, then the expansion of coding space occurs.

The value of R_0 can be decreased if the following is condition true:

$$I_K > I . \tag{11}$$

In this case up to $(I_K - I)$ classes B_i can be placed in adjacent cells of the map. Let us discuss the following example. Let the following partition $\Pi_A = \{B_1,\ldots,B_5\}$ be obtained for some FSM S_1: $B_1 = \{a_1\}$, $B_2 = \{a_2,a_3,a_4,a_5,a_6\}$, $B_3 = \{a_7,a_8,a_9,a_{10}\}$, $B_4 = \{a_{11},a_{12},a_{13},a_{14},a_{15}\}$ and $B_5 = \{a_{16},a_{17}\}$. Now, we have $R_B = 3$, $M_O = 5$ and $R_o = 3$. Therefore, it is enough $R_B + R_O = 6$ variables to encode the states. But the map includes $I_K = 8$ columns and condition (11) is true. Obviously, those classes $B_i \in \Pi_A$ should be transformed having the maximum number of states. Let us represent the class B_2 as $B_2^1 = \{a_2,\ldots,a_5\}$ and $B_2^2 = \{a_6\}$, whereas the class B_4 as $B_4^1 = \{a_{11},\ldots,a_{14}\}$ and $B_4^2 = \{a_{15}\}$. Now there are $M_O = 4$, $R_O = 2$ and variables $T_1,\ldots,T_5$ are used for state encoding (Fig. 2). As follows from Fig. 2, we have one code for each class $B_i \in \Pi_A$, namely: $K(B_1) = {*}00$, $K(B_2) = 0{*}1$, $K(B_3) = 010$, $K(B_4) = 11{*}$, $K(B_5) = 10{*}$. In this cased there is $R_A = 5$, so there is no code space expansion.

$T_1T_2T_3$ / T_4T_5	000	001	011	010	110	111	101	100
00	a_1	a_2	a_6	a_7	a_{11}	a_{15}	a_{16}	*
01	*	a_3	*	a_8	a_{12}	*	a_{17}	*
11	*	a_4	*	a_9	a_{13}	*	*	*
10	*	a_5	*	a_{10}	a_{14}	*	*	*

Fig. 2. The codes of states for Moore FSM S_1
Rys. 2. Kody stanów automatu Moore'a S_1

3. Proposed design method

In this article we propose the following design method for Moore FSM U_2:

1. Construction of the marked GSA and the set of states A. There are no difficulties here, because the well-known approach [1] is used.

2. Construction of the classes of pseudoequivalent states. States $a_i, a_j \in A$ are called pseudoequivalent states if the vertices marked by them are connected with input of the same vertex of GSA Γ [2]. The set Π_A is formed in a trivial way using this definition.

3. Prime state encoding. This step is connected with finding values of I_K and O_K. Here the possibility of decreasing R_O under keeping R_B is analyzed. This step is finished by placement of states $a_m \in B_i$ inside the columns of Karnaugh map having size $I_K * O_K$. It allows finding the set $T' \subset T$ and codes $K(B_i)$.

4. Secondary state encoding. This step is connected with optimizing system (2). Let us represent its equations as the following ones:

$$y_n = \bigvee_{m=1}^{M} C_{nm} A_m (n = 1,...,N). \qquad (12)$$

In (12) the symbol C_{nm} stands for the Boolean variable equal to 1 if the microoperation $y_n \in Y$ is formed in the state $a_m \in A$. States $a_m \in B_i$ are rearranged in the column $K(B_i)$ to decrease the number of terms in system (12). The "don't care" cells of the map are used for this optimization. Unfortunately, we have no an algorithm for solution of this problem.

5. Construction of transformed structure table. This table is constructed using the system of generalized formulae of transitions [2], which is the following one:

$$B_i \to \bigvee_{h=1}^{H_0} C_{ih} X_h a_s (i = 1,...,I). \qquad (13)$$

In (13), the Boolean variable $C_{ih} = 1$, if the term $X_h a_s$ belongs to the system of transitions for the class $B_i \in \Pi_A$. This system is constructed using initial GSA Γ. The transformed structure table includes the columns B_i, $K(B_i)$, a_S, $K(a_S)$, X_k, Φ_k, h. The rows of this table correspond to the terms $F_h \in F$:

$$F_h = (\bigwedge_{r=1}^{R_B} T_r^{l_{ri}}) * X_h (h = 1,...,H_0). \qquad (14)$$

In (14), the first part is determined by the code $K(B_i)$ from the row h, whereas the value of $l_{ri} \in \{0,1,*\}$ is the value of code bit r; $T_r^0 = \overline{T_r}, T_r^1 = T_r, T_r^* = 1$, where $r = 1,\ldots,R_B$.

These terms (14) belong to functions $D_r \in \Phi$, where:

$$D_r = \bigvee_{h=1}^{H_0} C_{rh} F_h (r = 1,..., R_B + R_O). \qquad (15)$$

In (15) the Boolean variable C_{rh}=1, if the row h contains function $D_r \in \Phi$, (h=1,…,H_0).

6. Implementation of FSM matrix circuit. Let us analyze the matrices from the circuit shown in Fig. 1. The matrix M_1 implements terms (14); it has I_1=2(L+R_B) inputs and O_1=H_0 outputs. The matrix M_2 implements functions (15); it has I_2=H_0 inputs and O_2=R_B+R_O. The matrix M_3 implements terms (12); it has I_3 inputs and O_3 outputs depending on outcome of the secondary encoding stage ($I_3 \le 2(R_B + R_O), O_3 \le M$). If some functions $y_n \in Y$ are represented by a single term, then they are formed directly by the matrix M_3. It decreases the number of inputs for M$_4$ from N–1 to 0. The lower bound corresponds to the situation when all microoperations are formed by the matrix M_3. In the general case the matrix M_4 has $I_4 \le M$ inputs and $O_4 \le N$ outputs.

4. Conclusion

The proposed method of expansion of encoding space targets on decrease in the chip space occupied by the matrix circuit of Moore FSM. This approach guarantees the decrease for the number of terms in the system of input memory functions of Moore FSM up to this value of the equivalent Mealy FSM. It allows decrease for the chip space used for implementing microoperations. But to make such a decrease, it is necessary to work out the corresponding method. It is reduced to change the cells of Karnaugh map occupied by states of FSM. Of course, there are many variants for this changing and an efficient method should do them in the right direction.

To investigate the effectiveness of proposed method, we used around 50 test FSMs from the library [7]. We made both primary and secondary state encodings by hand. Next, we calculated the space of matrix implementations using both arbitrary state codes and our code approach. Our investigations show that the proposed approach allows the chip space decrease up to 78%. In 42% of tests, we witnessed the increase of FSM's performance. It can be explained by absence of the matrix M_4 due to successful secondary encoding, when each microoperation is represented by single term. The further direction of our research is development of the secondary encoding method. Next, we are going to check how this method can be applied for the cases when FSM logic circuits are implemented using some standard elements as PAL and PLA, as well as the programmable logic devices using these types of logic as its macrocells.

5. References

[1] Baranov S.: Logic and System Design of Digital Systems. Tallinn: TUT Press, 2008.

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

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

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

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

[6] Solovjov V., Klimowicz A.: Logic Design of Digital Systems on the base of programmable logic devices. M.: Hot line-Telecom, 2008.

[7] Yang S.: Logic Synthesis and Optimization Benchmarks user guide. Technical report. Microelectronics center of North Carolina, 1991.

otrzymano / received: 07.05.2010
przyjęto do druku / accepted: 04.06.2010 *artykuł recenzowany*