Alexander BARKALOV, Larysa TITARENKO, Jacek BIEGANOWSKI
UNIWERSYTET ZIELONOGÓRSKI
INSTYTUT INFORMATYKI ELEKTRONIKI I TELEKOMUNIKACJI

Elementary chains modification for control unit optimization

Prof. dr hab. inż. Alexander A. BARKALOV

W latach 1976-1996 był pracownikiem dydaktycznym w Instytucie Informatyki Narodowej Politechniki Donieckiej. Współpracował aktywno z Instytutem Cybernetyki im. V.M. Glushkova w Kijowie, gdzie uzyskał tytuł doktora habilitowanego ze specjalnością informatyka. W latach 1996-2003 pracował jako profesor w Instytucie Informatyki Narodowej Politechniki Donieckiej. Od 2004 pracuje jako profesor na Wydziale Elektrotechniki, Informatykii Telekomunikacji Uniwersytetu Zielonogórskiego.

e-mail: a.barkalov@iie.uz.zgora.pl

Dr hab. Larysa TITARENKO

Dr hab. Larysa Titarenko, prof. UZ w 2004 roku obroniła rozprawę habilitacyjną i uzyskała tytuł doktora habilitowanego ze specjalnością telekomunikacja. W latach 2004-2005 pracowała jako profesor w Narodowym Uniwersytecie Radioelektronikiw Charkowie. Od 2005 roku pracuje na Wydziale Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego.

e-mail: l.titarenko@iie.uz.zgora.pl

Mgr inż. Jacek BIEGANOWSKI

Ukończył w roku 2003 studia na kierunku informatyka o specjalności inżynieria komputerowa. Od 2004 roku pracuje w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego na stanowisku asystenta. Jego zainteresowania związane są z systemami operacyjnymi, techniką cyfrową oraz algorytmami ewolucyjnymi.

e-mail: j.bieganowski@iie.uz.zgora.pl

Abstarct

The method of optimization of the hardware amount in addressing circuit of compositional microprogram control unit is proposed. Method is based on expansion of the microinstruction format by the field with code of the class of pseudoequivalent operational linear chains. Minimization is reached due to decreasing of the number of terms in system of Boolean functions describing the addressing circuit. An example of application of proposed method is shown.

Keywords: compositional microprogram control unit, code sharing, control microinstruction, PAL, synthesis, graph-scheme of algorithm.

Zastosowanie zmodyfikowanych łańcuchów elementarnych w optymalizacji jednostki sterującej

Streszczenie

W artykule przedstawiono metodę syntezy mikroprogramowanego układu sterującego z współdzieleniem kodów. Metoda jest zorientowana na zmniejszenie liczby makrokomórek PAL w części kombinacyjnej układu dzięki zastosowaniu zmodyfikowanych łańcuchów bloków operacyjnych. Proponowana modyfikacja polega na dodaniu do każdego łańcucha dodatkowych mikroinstrukcji z kodami klas łańcuchów pseudorównoważnych. W artykule przedstawiono także warunki jakie muszą być spełnione aby możliwe było zastosowanie proponowanej metody oraz analizę jej efektywności.

Słowa kluczowe: mikroprogramowana jednostka sterująca z współdzieleniem kodów, mikroinstrukcja sterująca, PAL, synteza, sieć działań.

1. Introduction

Control unit is one of the most important parts of any digital system [1]. If a control algorithm to be interpreted has the linear nature [2], it can be interpreted by Compositional Microprogram Control Unit (CMCU) [3]. Now Complex Programmable Logic Devices (CPLD) are widely used for implementation of combinational and sequential circuits [4, 5]. One of the main problems arose in the case of control unit implementation with CPLD is decrease of hardware amount, that is decrease of the number of Programmable Array Logic (PAL) macrocells required for implementation of a particular logic circuit [3, 4]. In case of CPLD, this task can be solved by decrease of the number of conjunctive terms in the sum-of-products of functions to be implemented [5]. The article propose solution for this problem in case of CMCU with code sharing and elementary Operational Linear Chains (OLC) [2, 3].

2. Background of CMCU

Let a Graph-Scheme of Algorithm (GSA) Γ be used for representation of a control algorithm [6]. Let $B=\{b_0, b_E\}\cup E_1\cup E_2$ be a set of GSA vertices and A be a set of arks connected some of these vertices. Here b_0 the initial (start) vertex of GSA, b_E is the final vertex, E_1 is a set of operator vertices, where $M=|E_1|$, and E_2 is a set of conditional vertices. Each operator vertex $b_q \in E_1$ contains a collection of microoperations $Y(b_q)\subseteq Y$, where $Y=\{y_1,\ldots,y_N\}$ is a set of data-path microoperations [7]. Each conditional vertex $b_q \in E_2$ includes some logical condition $x_e \in X$, where $X=\{x_1,\ldots,x_L\}$ is a set of logical conditions. A GSA Γ is named linear GSA if the number M exceeds 75% of the total number of its vertices [3].

Let a set of elementary OLC $C_E=\{\alpha_1, \ldots, \alpha_G\}$ be formed for a linear GSA Γ, where OLC $\alpha_g \in C$ is a vector of operator vertices [2, 3]. Remind, that each pair of OLC adjacent components corresponds to some ark in the set A. Each elementary OLC $\alpha_g \in C_E$ has one input and one output O_g. Let each vertex $b_q \in E_1$ correspond to microinstruction MI_q with address $A(b_q)$ and let this address have R bits, where

$$R = \lceil \log_2 M \rceil. \quad (1)$$

Let F_g be the number of components in OLC $\alpha_g \in C_E$ and let $Q=\max(F_1, \ldots, F_G)$. Encode each OLC $\alpha_g \in C_E$ by binary code

$$R_1 = \lceil \log_2 G \rceil. \quad (2)$$

Encode each component of OLC $\alpha_g \in C_E$ by binary code $K(b_q)$ using variables $T_r \in T$, where $|T|=R_2$ and

$$R_2 = \lceil \log_2 Q \rceil. \quad (3)$$

The encoding of components should satisfy condition

$$K(b_{gi}) = K(b_{gi}-1)+1 \quad (i=\overline{1,Fg}). \quad (4)$$

If condition

$$R = R_1 + R_2 \quad (5)$$

holds, then linear GSA Γ can be interpreted by CMCU with code sharing [3] denoted further as U_1 (Fig. 1).

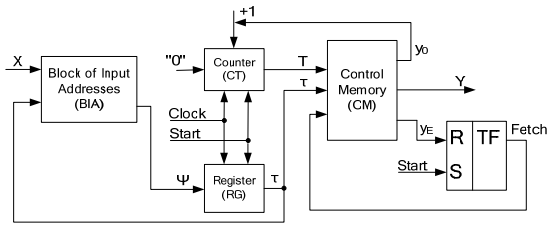

Fig. 1. Structural diagram of CMCU U_1
Rys. 1. Struktura mikroprogramowanego układu sterującego U_1

In CMCU U_1, a microinstruction address is represented as

$$A(b_q) = K(\alpha_g) * K(b_q), \tag{6}$$

where $b_q \in B_1$ and b_q is a component of OLC $\alpha_g \in C_E$, symbol * is a sign of concatenation [3]. This control unit operates in the following manner.

If Start=1, then zero address of the first microinstruction to be executed is loaded into register RG and counter CT. In the same time, flip-flop TF is set up and *Fetch*=1. Now microinstructions can be read out control memory (CM). Let contents of RG and CT form the address A(b_q), where b_q is a component of OLC $\alpha_q \in C_E$. If $b_q \notin O_g$, then special variable y_0 is generated by CM together with microoperations $y_n \in Y(b_q)$. If y_0=1, then content of CT is incremented. It corresponds to unconditional jump inside the OLC α_q. If $b_q = O_g$, then y_0=0 and block of input address (BIA) generates functions

$$\Psi = \Psi(\tau, X). \tag{7}$$

These functions form the code of next OLC to be interpreted. In the same time, counter CT is cleared. Contents of both CT and RG are changed using pulse Clock. If $\langle b_q,...,b_E\rangle \in A$, then special variable y_E is generated together with microoperations $y_n \in Y(b_q)$. It leads to reset of TF, *Fetch*=0 and operation of CMCU is terminated.

Let $\Pi_C = \{B_1, \ldots, B_I\}$ be a partition of the set C_E, where $B_i \in \Pi_C$ is a class of pseudoequivalent elementary OLC [2]. Remind, that OLC $\alpha_i, \alpha_j \in C_E$ are named pseudoequivalent OLC, if their outputs are connected with the input of the same vertex. Let condition

$$R_1 < R_3 \tag{8}$$

take place, where

$$R_3 = \lceil \log_2 I_E \rceil. \tag{9}$$

Here $I_E = |\Pi_E|$, $\Pi_E \subseteq \Pi_C$ and $B_i \in \Pi_E$ iff output of OLC $\alpha_g \in B_i$ is not connected with b_E. In this case we propose inserting of special control microinstructions [8] to decrease the hardware amount in block BIA.

3. Main idea of proposed metod

Let condition

$$2^{R_2} > F_g \tag{10}$$

take place for any OLC $\alpha_g \in C_1$, where C_1 is a set of elementary OLC from blocks $B_i \in \Pi_E$. Let us insert an additional control microinstruction MC_q in each OLC $\alpha_g \in C_1$. This microinstruction has field y_0=0 and field FB with binary code $K(B_i)$, where $\alpha_g \in B_i$. Let classes $B_i \in \Pi_E$ be encoded using variables $z_r \in Z$, where $|Z| = R_3$. In this case GSA Γ can be interpreted by CMCU U_2 (Fig. 2).

In CMCU U_2,block BIA generates functions

$$\Psi = \Psi(Z, X). \tag{11}$$

control memory CM is used to keep microoperations $y_n \in Y$, special variables y_0 and y_E, and code variables $z_r \in Z$. Principles of CMCU U_2 operation are clear from Fig. 2.

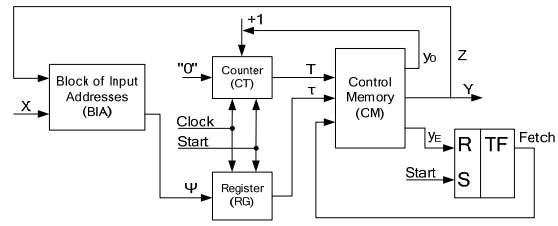

Fig. 2. Structural diagram of CMCU U_2
Rys. 2. Struktura mikroprogramowanego układu sterującego U_2

Let us point out that CMCU U_1 is a Moore finite-state-machine (FSM), where RG outputs are used as state variables [6]. The proposed approach permits decreasing of the number of terms in the system of input memory functions Ψ. It allows decrease of the number of PAL macrocells in logic circuit of block BIA for U_2 in comparison with U_1. The disadvantage of proposed method is increase of the cycles number required for execution of control algorithm represented by GSA Γ.

In this article we propose synthesis method for CMCU U_2, which includes the following steps:

1. Construction of sets C_E, C_1, Π_C and Π_E for GSA Γ.
2. Expansion of OLC $\alpha_g \in C_1$ by additional components.
3. Encoding of OLC $\alpha_g \in C_E$, their components and classes $B_i \in \Pi_E$.
4. Construction of control memory content.
5. Construction of transition table.
6. Implementation of CMCU logic circuit.

4. Analysis of proposed method

Let us compare hardware amount of CMCU U_1 and U_2 using the probabilistic approach suggested in [8] and developed in [2].

There are three key points in such an approach:

1. Usage of classes of GSA instead of a particular graph-scheme of algorithm. Each class is characterized by parameter p_1, which is treated as probability of the event that a particular vertex is an operator vertex of GSA. In case of linear GSA, $p_1 \geq 0{,}75$.
2. Usage of matrix models of CMCU [6] instead of logic circuit implementations with standard CPLD chips. In this case the hardware amount is determined as chip area occupied by CMCU logic circuit.
3. Study of relations $S(U_2)/S(U_1)$, where $S(U_1)$ is an area of matrix realization for CMCU U_i(i=1,2). It is proved [2] that such relations are the same for both matrix realization and logic circuit implementation with standard CPLD.

The matrix realization of CMCU U_1 is shown in Fig. 3.

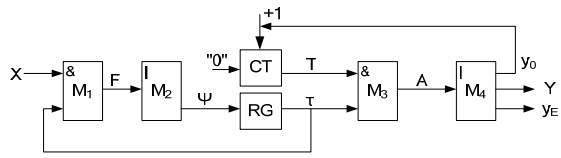

Fig. 3. Matrix realization of CMCU U_1
Rys. 3. Macierzowa realizacja układu CMCU U_1

Let us point out that some details such as *Start*, *Clock*, TF, *Fetch* are omitted in Fig. 1 because they are not important for further discussion. In Fig. 1, conjunctive matrix M_1 implements terms F corresponding to terms F_h(h=1,…,H) of CMCU transition table, disjunctive matrix M_2 implements input memory functions Φ, Ψ; conjunctive matrix M_3 implements variables $A_m \in A$, where A

is a set of CMCU addresses; disjunctive matrix M_4 implements microoperations $y_n \in Y$ and additional functions y_0, y_E.

Let K be the number of vertices in GSA Γ, then

$$R = \lceil \log_2 p_1 K \rceil. \tag{12}$$

Block BIA can be viewed as Moore FSM having

$$G = k_1 p_1 K \tag{13}$$

states, where $k_1 \leq 1$. Block BIA has R_1 outputs and $L+R_1$ inputs, where R_1 is determined by (2). The following formulae from [2]

$$L = (1 - p_1)K / 1.3; \tag{14}$$

$$H = 17.4 + 1.7 k_1 p_1 K. \tag{15}$$

can be used to estimate complexness of matrix realization (Fig. 3). Using (13)-(16), the areas $S(M_i)_1$, where i=1,…,4 can be found, namely

$$S(M_1)_1 = 2(L + R_1)H; \tag{16}$$

$$S(M_2)_1 = HR_1; \tag{17}$$

$$S(M_3)_1 = 2R2^R; \tag{18}$$

$$S(M_4)_1 = 2^R (N + 2) \cdot \tag{19}$$

The total area of CMCU U_1 matrix realization can be found as

$$S(U_1) = S(M_1)_1 + \ldots + S(M_4)_1. \tag{20}$$

In case of CMCU U_2, we have

$$S(M_1)_2 = 2(L + R_3)H_0; \tag{21}$$

$$S(M_2)_2 = H_0 R_1, \tag{22}$$

where

$$R_3 = \lceil \log_2 k_2 G \rceil. \tag{23}$$

In expresion (23) the coefficient $k_2 \leq 1$ determines the value of parameter I_E for a particular GSA Γ. The value of H_0 can be found using [2] as a following one:

$$H_0 = 4.4 + 1.1 k_2 G. \tag{24}$$

Let us point out that control memory of both U_1 and U_2 occupies the same area. It means that $S(M_3)_2 = S(M_3)_1$ and $S(M_4)_2 = S(M_4)_1$. Obviously, the total area needed for matrix realization of CMCU U_2 logic circuit is calculated using an expression similar to (20).

Some results of our probabilistic experiments are shown in Fig. 4.

As it follows from Fig. 4, the area of CMCU U_2 can be reduced up to 50% in comparison with U_1. Values of parameters k_1, k_2, k_3 were obtained from analysis of our benchmark GSAs. Plot on left proves that proposed method can be used with any linear GSA. If p_1 is closer to 1 the size of CMCU U_2 is smaller. Plot on right shows influence of microinstruction size on area of CMCU U_2, best results were obtained for microinstructions containing 10 microoperations. This results will be checked using our own software, which includes VHDL models of CMCU U_1 and U_2. The method ESPRESSO [1] is used for encoding of OLC and their classes. To generate a logic circuit of CMCU, standard package WebPack was used [9].

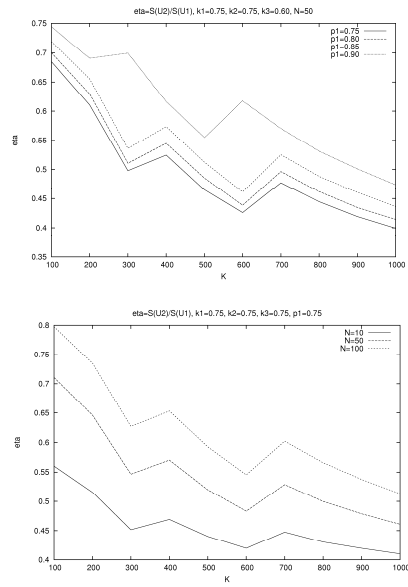

Fig. 4. Influence of parameter p_1 (left) and N (right) on function η
Rys. 4. Wpływ parametru p_1 (po lewej) i parametru N (po prawej) na funkcję η

5. Conclusion

The proposed method can be used if additional control microinstructions can be inserted without increase of resulted control memory size. It means that condition (10) should take place, otherwise equality (5) will be violated and

$$R_1 + R_2 = R + 1. \tag{25}$$

It means that the control memory size for CMCU U_2 will be twice more than its counterpart for CMCU U_1.

If condition (10) takes place, our statistic method shows that the total number of PAL macrocells in logic circuit of block BIA can be decreased up to 50 % using the proposed method. The best results were obtained for graph-scheme of algorithms with more than 500 vertices.

Disadvantage of this method is slowing down of resulted digital system. Each control microinstruction corresponds to idle cycle of system data-path. It means that operational linear chains modification can be applied if resulted performance satisfies initial technical requirements.

Further development of our researches is verification of proposed method application in case of control units implemented with FPGA.

6. References

[1] De Micheli G. Synthesis and Optimization of Digital Circuits. McGraw Hill, NY, 1994. – 578 pp.
[2] M. Adamski, A. Barkalov. Architectural and sequential synthesis of digital devices, ISBN: 83-7481-039-4. University of Zielona Góra, Zielona Góra, 2006. – 199 pp.
[3] D. Kania. Logic synthesis for PAL matrix programmable structures Zeszyty Naukowe Politechniki Śląskiej, Gliwice, 2004. – 240 str.
[4] V. V. Solovjev. Design of digital systems using the programmable logic integrated circuits. Hot Line-Telecom, Moscow, 2001, (in Russian). – 636 pp.
[5] A. Barkalov, L. Titarenko, Design of control units with programmable logic devices. Wydaw. Komunikacji i Łączności, 2007
[6] S. I. Baranov, Logic synthesis of Control Automata. Kluwer AcademicPublishers, 1994.
[7] A. Barkalov, L. Titarenko, and J. Bieganowski, Synthesis of control unit with modified operational linear chains, PAK, 2007, No. 5, pp. 5–17.
[8] Novikow G. About one approach for finite-state-machines research. Contr. Syst. Mach. – 1974. – No. 2. – pp. 70–75 (in Russian).
[9] www.xilinx.com

Artykuł recenzowany