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

Synthesis of control unit using code sharing and chain modifications

Prof. dr hab. inż. Alexander A. BARKALOV

Prof. Alexander A. Barkalov w latach 1976-1996 był pracownikiem dydaktycznym w Instytucie Informatyki Narodowej Politechniki Donieckiej. Współpracował aktywnie 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, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego.

e-mail: a.barkalov@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 programowaniem, systemami operacyjnymi, techniką cyfrową oraz algorytmami ewolucyjnymi.

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

Dr hab. Larysa TITARENKO, prof. UZ

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

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

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. Modyfikacja łańcuchów 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: mikroinstrukcja, mikroprogramowy układ sterujący, CPLD, PAL.

Abstract

The paper presents a method of synthesis of compositional microprogram control unit with code sharing. The method is oriented on decreasing the number of PAL macrocells in the combinational part of the device by modification of the operational linear chains. Some control microinstructions are added into each chain with classes codes of the peudoequivalent operational linear chains. The article also presents conditions which should be met to apply given method and an analyze of its efficiency.

Keywords: microistruction, control memory, compositional microprogram control unit, CPLD, PAL.

Introduction

One of the most important blocks of digital systems is the control unit [1] which can be implemented as a compositional microprogram Control Unit (CMCU) [2]. Nowadays Complex Programmable Logic Devices (CPLD) [3, 4] are widely used for implementation of logic circuits. The problem of decreasing of hardware amount in the circuit of a control unit is still a task of great importance [3]. In the case of CPLD, this problem can be solved by decreasing of the number of terms of Sum-of-Products (SoP) functions to be implemented [3, 4]. In this article we suggest a possible solution for this problem applicable for CMCU with code sharing [5].

Background of CMCU with code sharing

Let a control algorithm be represented by graph-scheme of algorithm (GSA) [6] with the set of vertices $B=\{b_0,b_E\}\cup E_1\cup E_2$ and the set of arcs E. Let b_0 be the initial vertex, b_E be the final vertex, E_1 be the set of operator vertices and E_2 be the set of conditional vertices. The operator vertex $b_q \in E_1$ contains a collection of microoperations $Y(b_q) \subseteq Y$, where $Y=\{y_1,...,y_N\}$ is a set of microoperations. Conditional vertex $b_q \in E_2$ contains some element $x_e \in X$, where $X=\{x_1,...,x_L\}$ is a set of logic conditions. A GSA Γ is called a linear GSA [2] if the number $M=|E_1|$ of its operator vertices exceeds 75% of the value $|B|$.

Let set $C = \{\alpha_1,...,\alpha_G\}$ be formed for linear GSA Γ, where $\alpha_g \in C$ is an Operational Linear Chain (OLC).

An OLC $\alpha_g \in C$ is a sequence of operator vertices, such that each pair of its adjacent components corresponds to some arc from set E. Each OLC $\alpha_g \in C$ has only one output O_g and arbitrary number of inputs. Formal definitions of OLC and their input and outputs can be found in [2, 5]. Let each vertex $b_q \in E_1$ correspond to the microinstruction MI_q with address $A(b_q)$ and let this address have R bits, where

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

Let each OLC $\alpha_g \in C$ include F_g components, and let $G=|C|$. Let $Q=\max(F_1, \ldots, F_G)$. Encode each OLC $\alpha_g \in C$ by binary code $K(\alpha_g)$ using variables $\tau_r \in \tau$, where $|\tau|=R_1$ and

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

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

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

The encoding of components should be executed in such a manner that condition

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

is met for each OLC $\alpha_g \in C$. If condition

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

holds, then GSA Γ can be interpreted by CMCU with code sharing U_1 (Fig. 1).

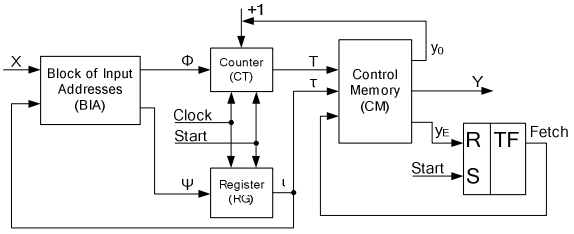

Fig. 1. Structural diagram of CMCU U_1

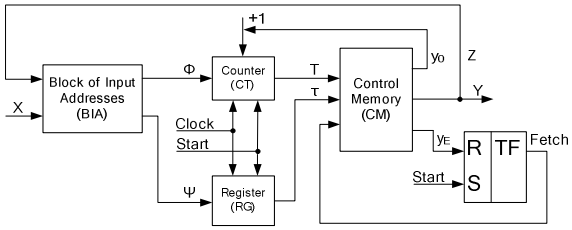

Fig. 2. Structural diagram of CMCU U_2

In CMCU U_1 an address of microinstruction corresponding to component b_q of OLC $\alpha_g \in C$ is represented as

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

where * is a sign of concatenation. The block of input addresses (BIA) generates input memory functions

$$\begin{aligned} \Phi &= \Phi(\Gamma, X), \\ \Psi &= \Psi(\Gamma, X), \end{aligned} \tag{7}$$

used to load component code into counter CT and code of OLC into register RG respectively. Control memory CM keeps microoperations $y_n \in Y$ and special variables y_0 and y_E. If y_0=1, then the current content of CT is incremented, otherwise both CT and RG are loaded from BIA. The first case corresponds to transition from any component of α_g except its output. The second case corresponds to transition from OLC output. If y_E =1, then flip-flop TF is cleared, signal *Fetch*=0 and the operation of CMCU is terminated. It corresponds to vertex b_E of GSA. Pulse *Start* is used to load zero codes into both RG and CT which corresponds to address of first microinstruction. In the same time flip-flop TF is set up, *Fetch*=1 and microinstructions can be read out CM. Pulse *Clock* is used for timing of CMCU.

Let $\Pi_C=\{B_1,\ldots, B_I\}$ be the partition of set C by classes of pseudoequivalent OLC. Recall that OLC $\alpha_i, \alpha_j \in C$ are pseudoequivalent OLC if their outputs are connected with the input of the same vertex [2]. Let condition

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

take place, where

$$R_3 = \lceil \log_2 I_0 \rceil. \tag{9}$$

Here $I_0=|\Pi_0|$, $\Pi_0 \in \Pi_C$ and $B_i \in \Pi_0$ if output O_g is not connected with the final vertex of GSA Γ.

In this case we propose to decrease the block BIA hardware amount using special control microinstructions [7].

The main idea of proposed method

Let condition

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

take place for all OLC $\alpha_g \in C_1$, where $C_1 \subseteq$ C is a set of OLC from classes $B_i \in \Pi_0$. Let us insert an additional component MC_g in each OLC $\alpha_g \in C_1$. This component corresponds to the control microinstruction with y_0=0 and field FB which contains code $K(B_i)$ of class $B_i \in \Pi_0$, where $\alpha_g \in B_i$. Let classes $B_i \in \Pi_0$ 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 implements systems

$$\Phi = \Phi(Z, X); \tag{11}$$

$$\Psi = \Psi(Z, X). \tag{12}$$

Other elements of U_2 have the same meaning as their counterparts for U_1. Functions (11)-(12) are generated when the concatenation of contents of RG and CT represents an address of the control microinstruction. In this case the data-path of the controlled digital system is in idle state. It can be achieved if synchronization of data-path is controlled by variable y_0.

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

- construction of sets C, C_1, Π_C and Π_0 for GSA Γ,
- including additional components into OLC $\alpha_g \in C_1$,
- encoding of OLC $\alpha_g \in C$, their components and classes $B_i \in \Pi_C$,
- construction of control memory content,
- construction of transition table of CMCU U_2,
- implementation of CMCU logic circuit.

Analysis of proposed method

Let us compare hardware amount of CMCU U_1 and U_2 using the probabilistic approach suggested in [9] 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.

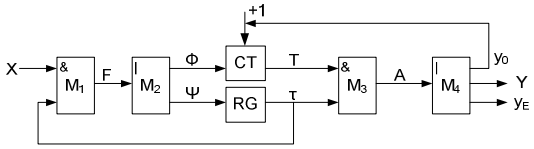

Fig. 3. Matrix realization of CMCU U_1

Let us point out that some details such as *Start*, *Clock*, TF, *Fetch* are omitted in Fig. 3 because they are not important for further discussion. In Fig. 3, 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

$$R = \lceil \log_2 p_1 K \rceil. \tag{13}$$

Block BIA can be viewed as Moore FSM having

$$M_1 = k_1 p_1 K \qquad (14)$$

states, where M_1 is equal to the total number of inputs for all OLC $\alpha_g \in C_1$; $k_1 \le 1$. This block has R outputs and $L+R_1$ inputs, where

$$R_1 = \lceil \log_2 k_2 M_1 \rceil. \qquad (14)$$

In (14) coefficient k_2 determines parameter G for particular GSA ($k_2 \le 1$). The following formula from [2] can be used to find the values of parameters needed to find a matrix realization area:

$$L = (1 - p_1)K / 1.3; \qquad (15)$$

$$H = 17,4 + 1,7 k_1 p_1 K. \qquad (16)$$

Using formula (13)-(16) we can find areas $S(M_i)_1$, where i=1,…,4, namely

$$S(M_1)_1 = 2(L + R_1)H; \qquad (17)$$

$$S(M_2)_1 = HR; \qquad (18)$$

$$S(M_3)_1 = 2R \cdot 2^R; \qquad (19)$$

$$S(M_4)_1 = 2^R \cdot (N + 2). \qquad (20)$$

Sum of areas $S(M_1)$-$S(M_4)$ gives area $S(U_1)$ needed for implementation of CMCU U_1. In case of CMCU U_2 we have

$$S(M_1)_2 = 2(L + R_3)H_0 \qquad (21)$$

$$S(M_2)_2 = H_0 R; \qquad (22)$$

$$S(M_3)_2 = S(M_3)_1; \quad S(M_4)_2 = S(M_4)_1. \qquad (23)$$

Value of R_3 can be calculated as

$$R_3 = \lceil \log_2 k_3 k_2 M_1 \rceil, \qquad (24)$$

where coefficient $k_3 \le 1$ determines parameter I_0 for particular GSA. Using results from [2] we can find

$$H_0 = 4,4 + 1,1 k_3 k_2 M_1. \qquad (25)$$

Some results of our probabilistic experiments are shown in Fig. 4. The figure presents comparison of estimated area of circuit U_1 and U_2 ($\eta = S(U_1)/S(U_2)$). Values k_1=0.75, k_2=0.75, k_3=0.6 were obtained from analysis of our set of benchmark GSAs.

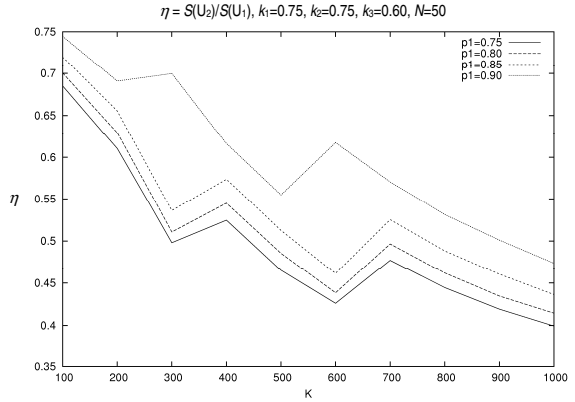

Fig. 4. Influence of parameter p_1 on function η

As one can see from Fig.4 the method can be used with any linear flow-chart (p_1>0.75). Area of CMCU U_2 can be reduced up to 50% if the value of parameter p_1, is close to 1.0.

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 [8].

Conclusion

The method proposed can be used if additional control microinstruction can be inserted without increasing control memory size. It means that condition (10) should take place otherwise, equation (5) will be violated and

$$R_1 + R_2 = R + 1 \qquad (26)$$

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

If condition (10) takes place, the total number of PAL macrocells in logic circuit of block BIA can be decreased up to 50 % with use of 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 resulting digital system. Each control microinstruction corresponds to idle cycle of system data-path. It means that operational linear chains modification can be applied if resulting performance satisfies initial technical requirements.

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

References

[1] De Micheli G. *Synthesis and Optimization od 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. 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 finite states machines for telecommunication systems*, 2007, vol. 148, CNURE, Kharkiv.

[6] S. I. Baranov, *Logic synthesis of Control Automata*. Kluwer Academic-Publishers, 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] http://www.xilinx.com

[9] G. Novikow *About one approach for finite-state-machines research*. Contr. Syst. Mach., 1974, No. 2., pp. 70–75, (in Russian).

Title: Synteza jednostki sterującej z dzieleniem kodów oraz modyfikacją łańcuchów operacyjnych

Artykul recenzowany