Optimization of Control Unit with Code Sharing

Aleksander A. Barkalov, *Member, IEEE,* Larysa A. Titarenko, Aleksander N. Miroshkin

***Abstract* — The new design method for compositional microprogram control units with code sharing is proposed. The method targets on reduction in the number of PAL macrocells in the combinational part of control unit. Some additional control microinstructions containing codes of the classes of pseudoequivalent chains are used for operational linear chains modification. Proposed method is illustrated by an example. Various graph-scheme of algorithm (GSA) research results are illustrated with the diagrams. Most desirable GSA characteristics for using proposed method were obtained.**

***Index Terms* — Circuit synthesis, flow graphs, logic devices, minimization methods.**

I. Introduction

A control unit (CU) is one of the important blocks of any digital system [1]. The problem of hardware amount reduction is an important problem connected with implementation of logic circuits of CUs [2]. Peculiarities of a control algorithm to be implemented as well as logic elements in use should be taken into account to solve this problem. In this article we propose a method of this problem solution in case when a linear control algorithm is implemented using complex programmable logic devices (CPLD). We discuss the case when macrocells of programmable array logic (PAL) and embedded memory blocks (EMB) are used in a CPLD chip [3, 4]. In a linear algorithm there are more than 75% of operator vertices [5]. The compositional microprogram control units (CMCU) [5] are widely used for interpretation of linear algorithms. An approach based on existence of pseudoequivalent operational linear chains (POLC) is proposed in [6, 7] for optimization of CMCU with code sharing [5]. But this approach does not decrease the hardware amount for a block of microoperations. The development of this approach is proposed in this article, which is based on coding of collections of microoperations [2].

The aim of this research is CMCU logic circuit optimization due to introduction in the format of microinstruction the special fields with codes of classes of POLCs and collections of microoperations.

The task of research is development of synthesis method allowing decrease for the numbers of macrocells PAL and blocks EMB in the logic circuit of CMCU. A control algorithm is represented by a graph-scheme of algorithm (GSA) [8, 9].

Manuscript received March 7, 2009. Optimization of Control Unit with Code Sharing.

A. A. Barkalov is with University of Zielona Gora, Poland. E-mail: A.Barkalov@iie.uz.zgora.pl

L. A. Titarenko is with University of Zielona Gora, Poland. E-mail: L.Titarenko@iie.uz.zgora.pl

A. N. Miroshkin is with Donetsk National Technical University, Donetsk, Ukraine. MiroshkinAN@gmail.com

II. Analysis of CMCU with Code Sharing

Let a control algorithm to be interpreted be represented by a graph-scheme of algorithm (GSA) Γ [9]. Let this GSA be characterized by the set of vertices $B = \{b_0, b_E\} \cup E_1 \cup E_2$ and the set of arcs E, where b_0 is an initial vertex, b_E is a final vertex, E_1 is a set of operator vertices, 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 $Y = \{y_1, ..., y_N\}$ is a set of data-path microoperations. Each conditional vertex $b_q \in E_2$ contains some element $x_l \in X$, where $X = \{x_1,...,x_L\}$ is a set of logical conditions (input signals). A GSA Γ is named a linear GSA [5] if the number of its operator vertices exceeds 75% of the total their number in the GSA.

Let the set $C = \{\alpha_1, ..., \alpha_G\}$ be constructed for GSA Γ, where $\alpha_g \in C$ is an operational linear chain (OLC) [5]. Any component b_{g_i} of OLC $\alpha_g \in C$ belongs to the set E_1 $(i = 1, ..., F_g)$. Each pair of adjacent components b_{g_i}, $b_{g_{i+1}}$ corresponds to the arc $< b_{gi}, b_{gi+1} > \in E$, where $i = 1, ..., F_g - 1$, $g = 1, ..., G$. Each OLC $\alpha_g \in C$ has only one output O_g and the arbitrary number of inputs. Formal definitions of OLC, its input and output can be found in [5]. Each vertex $b_q \in E_1$ corresponds to microinstruction MI_q kept in the cell of control memory (CM) with address A_q. It is enough

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

bits for microinstruction addressing, where $M = |E_1|$. Let each OLC $\alpha_g \in C$ include F_g components and $Q = \max(F_1, ..., F_G)$. Let each OLC $\alpha_g \in C$ be encoded by binary code $K(\alpha_g)$ having

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

bits and variables $\tau_r \in \tau$ be used for such an encoding, where $|\tau| = R_1$. Let each component $b_q \in E_1$ be encoded by binary code $K(b_q)$ having

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

bits and variables $T_r \in T$ be used for this encoding, where $|T| = R_2$. The encoding of components is executed in such a manner that condition

$$K(b_{gi+1}) = K(b_{gi}) + 1 \quad (4)$$

takes place for each OLC $\alpha_g \in C$ $(i = 1, \ldots, F_g - 1)$. If condition

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

takes place, then the model of CMCU with code sharing U_1 can be used for interpretation of GSA Γ (Fig. 1).

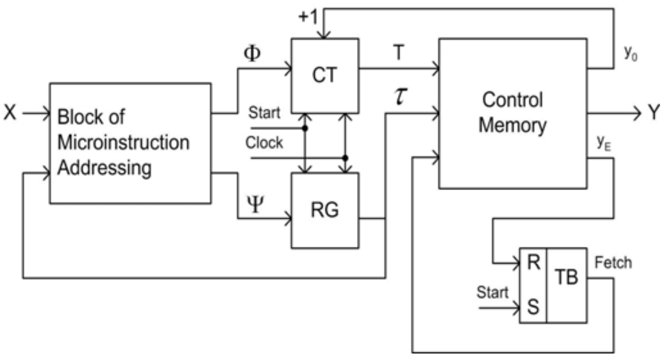

Fig. 1. Structural diagram of CMCU U_1

In CMCU U_1, a block of microinstruction addressing (BMA) implements the system of input memory functions for counter CT and register RG:

$$\Phi = \Phi(\tau, X); \quad \Psi = \Psi(\tau, X). \quad (6)$$

Let us point out that in the case of CMCU U_1 an address of microinstruction is represented as the following one:

$$A(b_q) = K(\alpha_g) * K(b_q), \quad (7)$$

where b_q is a component of OLC $\alpha_g \in C$ and "*" is a sign of concatenation. The CMCU U_1 operates in the following order.

If $Start = 1$, then an initial address (all zeros) is loaded into RG and CT. In the same time a flip-flop TF is set up which causes $Fetch = 1$, then microinstructions can be read out of control memory. Each cell of CM keeps microoperations $y_n \in Y$ and special variables y_0 and y_E. If $y_0 = 1$, then a current content of CT is incremented, otherwise both CT and RG are loaded from BMA. The first case corresponds to transition from any OLC component except of its output. The second case corresponds to transition from an OLC output. If $y_E = 1$, then the flip-flop TF is reset, signal $Fetch = 0$ and operation of CMCU is terminated. It corresponds to transition from the vertex $b_q \in E_1$, where $< b_q, b_E > \in E$. Pulse **Clock** is used for timing of CMCU.

Let us point out that OLC $\alpha_i, \alpha_j \in C$ are pseudoequivalent OLC [5] if their outputs are connected with input of the same vertex of GSA Γ. The hardware amount in logic circuit of BMA can be decreased due to introduction of a special block for transforming the OLC codes into the codes of the classes of pseudoequivalent OLC named as a code transformer (TC) [5]. But the TC consumes some resources of the chip in use.

In this article we propose to use free cells of CM for this transformation. To reduce the number of EMB in the control memory, we propose to use the maximum encoding of collections of microoperations [2].

III. Main idea of proposed method

Let $C_1 \subset C$ be a set of OLC such that their outputs are not connected with the vertex b_E. Let us find the partition $P_C = \{B_1, \ldots B_I\}$ of the set C_1 by the classes of POLC. Let us encode classes $B_i \in \Pi_C$ by binary codes $K(B_i)$ having R_B bits, where

$$R_B = \lceil \log_2 I \rceil. \quad (8)$$

Let us use variables $v_r \in V$ for this encoding, where $|V| = R_B$.

In the process of CMCU synthesis, an initial GSA Γ is transformed and additional variables y_0 and y_E are introduced in its operational vertices. Thus, the initial set Y is transformed in the set $Y_C = Y \cup \{y_0, y_E\}$. Let the set Y_C includes Q_1 different collections of microoperations (CMO). Let us encode each collection Y_q by a binary code $K(Y_q)$ having R_Y bits, where

$$R_Y = \lceil \log_2 Q_1 \rceil. \quad (9)$$

Let us use variables $z_r \in Z$ for this encoding, where $|Z| = R_Y$. In this case the control memory includes two blocks [5], namely a block of micromemory (BMM) and a block of microoperation (BMO). The BMM generates functions

$$Z = Z(T, \tau), \quad (10)$$

and the BMO generates variables

$$Y_C = Y_C(Z). \quad (11)$$

In this article we propose to include the fields $K(B_i)$ and $K(Y_q)$ in the microinstruction format. These microinstructions include R_I bits, where

$$R_I = R_B + R_Y. \quad (12)$$

Both BMM and BMO are implemented using EMBs having t outputs. Assume that each EMB includes q words and

$$q \geq \max(M, Q_1). \quad (13)$$

The block BMM has R_Y outputs and it is implemented using n_1 blocks EMB, where

$$n_1 = \left\lceil \frac{R_Y}{t} \right\rceil. \quad (14)$$

In this case, there are R_3 free bits in the word of BMM, where

$$R_3 = n_1 t - R_Y. \quad (15)$$

These free bits can be used for keeping of some part V^1 of the code $K(B_i)$.

All bits of $K(B_i)$ are generated by the BMM if the following condition takes place:

$$R_3 \geq R_B . \quad (16)$$

Otherwise, the block of code transformer (BCT) is used to generate the rest of the bits, R_4, where

$$R_4 = R_B - R_3 . \quad (17)$$

These bits form a part V^2 of the code $K(B_i)$. This approach leads to a CMCU U_2 (Fig. 2).

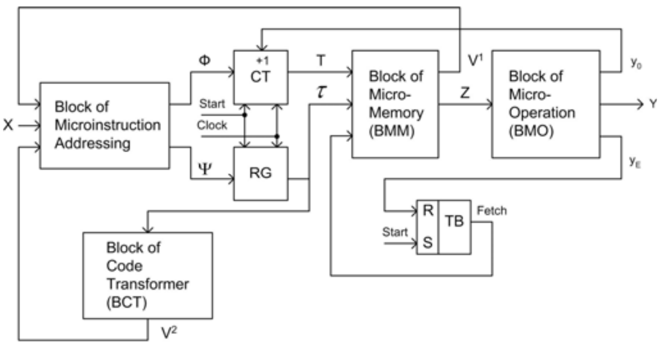

Fig. 2. Srtuctural diagram of CMCU U_2

In CMCU U_2, the block BMA implements functions

$$\Phi = \Phi(V, X), \quad (18)$$

$$\Psi = \Psi(V, X), \quad (19)$$

and the block BCT implements functions

$$V^2 = V^2(\tau). \quad (20)$$

The following conditions take places:

$$V^1 \cup V^2 = V , \quad (21)$$

$$V^1 \cap V^2 = \varnothing . \quad (22)$$

Functions of other blocks have been already discussed. Let us point out that logic circuits of BMA, CT, RG and TF are implemented using PAL macrocells, whereas circuits of BMM and BMO using EMBs. Logic circuits of BCT can be implemented using either PAL macrocells or EMBs.

In this article the following synthesis method is proposed for the CMCU U_2:

1. Construction of sets C, C_1 and P_C for GSA Γ.
2. Encoding of OLCs, their components and classes.
3. Encoding of collections of microoperations $Y_q \subseteq Y_C$.
4. Construction of control memory contents for blocks BMM and BMO.
5. Construction of CMCU transition table.
6. Construction of BCT table.
7. Logic synthesis of CMCU logic circuit.

IV. Application of Proposed Method

Let a GSA Γ_1 be represented by the sets $C = \{\alpha_1, ..., \alpha_8\}$, where $\alpha_8 \notin C_1$, and $\Pi_C = \{B_1, ...B_5\}$, where $B_1 = \{\alpha_1\}$, $B_2 = \{\alpha_2, \alpha_3\}$, $B_3 = \{\alpha_4, \alpha_5\}$, $B_4 = \{\alpha_6\}$, $B_5 = \{\alpha_7\}$, $\alpha_1 = < b_1, b_2, b_3 >$, $\alpha_2 = < b_4, ..., b_7 >$, $\alpha_3 = < b_8, b_9 >$, $\alpha_4 = < b_{10}, b_{11}, b_{12} >$, $\alpha_5 = < b_{13}, ..., b_{16} >$, $\alpha_6 = < b_{17}, ..., b_{19} >$, $\alpha_7 = < b_{20}, b_{21} >$, $\alpha_8 = < b_{22}, b_{23}, b_{24} >$. Therefore, we can get the following values and sets: number of OLC $G = 8$, for their encoding we use $R_1 = 3$ variables from the set $\tau = \{\tau_1, \tau_2, \tau_3\}$, maximum OLC length is $Q = 4$ vertexes, for their encoding $R_2 = 2$ variables from the set $T = \{T_1, T_2\}$ is enough, number of operational vertices in the GSA $M = 24$, $R = 5$ bits are necessary for their encoding. Hence, condition (5) takes place and there is possibility to use the code sharing. It is enough $R_B = 3$ variables for encoding of the classes $B_i \in \Pi_C$. It means that $V = \{v_1, v_2, v_3\}$.

Let us encode OLC $\alpha_g \in C$ and their components in the following way: $K(\alpha_1) = 000$, …, $K(\alpha_8) = 111$, $K(B_1) = 000$, …, $K(B_5) = 100$. To satisfy the condition (4), let the first component of each OLC $\alpha_g \in C$ have code 00, the second 01, the third 10, and the fourth 11. It leads to microinstruction addresses $A(b_q)$ shown in Table 1.

TABLE 1

Microinstruction Addresses for CMCU $U_2(\Gamma_1)$

Address	000	001	010	011	100	101	110	111
00	b_1	b_4	b_8	b_{10}	b_{13}	b_{17}	b_{20}	b_{22}
01	b_2	b_5	b_9	b_{11}	b_{14}	b_{18}	b_{21}	b_{23}
10	b_3	b_6	*	b_{12}	b_{15}	b_{19}	*	b_{24}
11	*	b_7	*	*	b_{16}	*	*	*

From Table 1 we can derive, for example, that $A(b_5) = 00101$, $A(b_{15}) = 10010$, and so on. Replacement of vertices by corresponding collections of microoperations in Table 1 results in the content of control memory (Table 2).

TABLE 2

Control Memory Content for CMCU $U_2(\Gamma_1)$

Address	000	001	010	011	100	101	110	111
00	y_0, y_1, y_2	y_0, y_3, y_5	y_0, y_1, y_2	y_0, y_3, y_6	y_0, y_3, y_5	y_0, y_1, y_2	y_0, y_3, y_6	y_0, y_3, y_9
01	y_0, y_3, y_9	y_0, y_3, y_9	y_1, y_7	y_0, y_3, y_9	y_0, y_3, y_9	y_0, y_3, y_5	y_8	y_0, y_3, y_6
10	y_4	y_0, y_3, y_6	*	y_8	y_0, y_1, y_2	y_4	*	y_1, y_2, y_E
11	*	y_8	*	*	y_1, y_7	*	*	*

Obviously, collections of microoperations are taken from the GSA Γ_1, but we do not show it. As follows from Table 2, the control memory includes $Q_1 = 8$ collections of microoperations, namely: $Y_1=\{y_0,y_1,y_2\}$, $Y_2=\{y_0,y_3,y_9\}$, $Y_3=\{y_4\}$, $Y_4=\{y_0,y_3,y_5\}$, $Y_5=\{y_0,y_3,y_6\}$, $Y_6=\{y_1,y_7\}$, $Y_7=\{y_8\}$, $Y_8=\{y_1,y_2,y_E\}$. They can be encoded using $R_Y=3$ variables, therefore $Z=\{z_1,z_2,z_3\}$.

Let EMB in use have $t=2$ outputs, then number of used EMB $n_1=2$. Number of non-used bits $R_3=1$. It means that one bit of the code $K(B_i)$ can be generated by the block BMM. Let variables $v_r \in V$ be devided between V^1 and V^2 in the following way: $V^1=\{v_1\}$, $V^2=\{v_2,v_3\}$.

It is enough to replace the collections in Table 1 by their codes to specify the block BMM. Each output of OLC $\alpha_g \in B_i$ is complemented by value of the first bit of code $K(B_i)$. In our example, the block BMM is represented by Table 3, and the variable v_1 is included in the output of OLC α_7.

TABLE 3
CONTENT OF BLOCK BMM FOR CMCU $U_2(\Gamma_1)$

Address	000	001	010	011	100	101	110	111
00	000	001	000	011	100	000	100	001
01	001	011	101 _	100	001	011	110 $\underline{v_1}$	100
10	010 _	100	*	110 _	000	010 _	*	111
11	*	110 _	*	*	101 _	*	*	*

The block BMO is specified by a table with columns $K(Y_q)$, Y_q, q. This table is constructed in a trivial way (Table 4).

TABLE 4
CONTENT OF BLOCK BMO FOR CMCU $U_2(\Gamma_1)$

$K(Y_q)$	Y_q	q	$K(Y_q)$	Y_q	q
000	y_0,y_1,y_2	1	100	y_0,y_3,y_6	5
001	y_0,y_3,y_9	2	101	y_1,y_7	6
010	y_4	3	110	y_8	7
011	y_0,y_3,y_5	4	111	y_1,y_2,y_E	8

To construct the table of transitions for CMCU U_2, it is necessary to construct the system of generalized formulae of transitions [4] for classes $B_i \in \Pi_C$. Let the following system exist for our example:

$$\begin{aligned} &B_1 \to x_1b_4 \vee \bar{x}_1b_8;\\ &B_2 \to x_3b_{10} \vee \bar{x}_3x_4b_{13} \vee \bar{x}_3\bar{x}_4b_{17};\\ &B_3 \to x_2b_{17} \vee \bar{x}_2x_3b_{20} \vee \bar{x}_2\bar{x}_3b_{18};\\ &B_4 \to b_{20};\ B_5 \to x_1b_{22} \vee \bar{x}_1b_{11}. \end{aligned} \tag{23}$$

Such a system is the base for construction of CMCU transition table including the following columns: B_i, $K(B_i)$, b_q, $A(b_q)$, X_h, Φ_h, Ψ_h, h. The purpose of each column is clear from Table 5. The number of such a table rows H is determined by the number of terms in system of generalized formulae of transitions. In our case we have $H=11$.

TABLE 5
FRAGMENT OF TRANSITIONS TABLE FOR CMCU $U_2(\Gamma_1)$

B_i	$K(B_i)$	b_q	$A(b_q)$	X_h	Φ_h	Ψ_h	h
		b_{17}	10100	x_2	–	D_1D_3	6
B_3	010	b_{20}	11000	$\bar{x}_2x_3$	–	D_1D_2	7
		b_{18}	10101	$\bar{x}_2\bar{x}_3$	D_5	D_1D_3	8

This fragment describes the transitions for class B_3, starting from the sixth term of system (23). The table of transitions is used to derive functions (18)-(19), having the following terms

$$F_h=\left(\bigwedge_{r=1}^{R_B} v_r^{l_{rh}}\right)\cdot X_h\ (h=1,\dots,H). \tag{24}$$

In system (24), the symbol l_{rh} stands for value of the bit r of code $K(B_i)$ from the line h of the table: $l_{rh}\in\{0,1\}$, $v_r^0=\bar{v}_r$, $v_r^1=v_r$ $(r=1,..,R_B)$. For example, the following system can be derived from Table 5:

$$D_1=F_6 \vee F_7 \vee F_8=\bar{v}_1v_2\bar{v}_3;$$
$$D_2=F_6 \vee F_8=\bar{v}_1v_2\bar{v}_3x_2 \vee \bar{v}_1v_2\bar{v}_3\bar{x}_2\bar{x}_3;$$
$$D_3=F_8=\bar{v}_1v_2\bar{v}_3\bar{x}_2\bar{x}_3.$$

The table of BCT includes columns α_g, $K(\alpha_g)$, B_i, $K(B_i)$, V_g^2. In our example, Table 6 represents the block BCT.

TABLE 6
SPECIFICATION OF BLOCK BCT FOR CMCU $U_2(\Gamma_1)$

α_g	$K(\alpha_g)$	B_i	$K(B_i)$	V_g^2	g
α_1	000	B_1	000	–	1
α_2	001	B_2	001	v_3	2
α_3	010	B_2	001	v_3	3
α_4	011	B_3	010	v_2	4
α_5	100	B_3	010	v_2	5
α_6	101	B_4	011	v_2v_3	6
α_7	110	B_5	100	–	7
α_8	111	B_6	–	–	8

Remind that the variable v_1 is generated by the block BMM. In the same time, there is no code $K(B_6)$ because $\alpha_8 \notin C_1$. Obviously, this table specifies blocks EMB. If the logic circuit of BCT is implemented using PAL macrocells, then Table 6 corresponds to Karnaugh maps for function $v_r \in V^2$. To optimize system (20), we should encode OLC $\alpha_g \in C_1$ in the optimal way. The well-known method

ESPRESSO [1], for example, can be used for such an encoding. We do not discuss this task in our article.

Implementation of the logic circuit of CMCU U_2 is reduced to implementation of systems (18)-(19) using PAL macrocells, and tables similar to Table 3, Table 4, and Table 6 using EMB. To solve this task, a designer can use either standard tools [4] or some known methods [8]. We do not discuss this step also.

Let us point out that the control memory of CMCU $U_1(\Gamma_1)$ includes 32*12=384 bits (if $t = 2$), and CMCU transition table includes 17 lines. In the CMCU $U_2(\Gamma_1)$, the BMM includes 32*4=128 bits, the BMO requires 8*11=88 bits, and the BCT consumes 8*2=16 bits. Therefore, the control memory of CMCU $U_2(\Gamma_1)$ uses 232 bits of memory, and its transition table has $H = 11$ lines. It means that the CMCU $U_2(\Gamma_1)$ requires 1.5 times less of the memory, and its block BMA includes 1.54 times less amount of terms.

V. Conclusion

In this article we propose the method oriented on decrease for the number of macrocells in the logic circuit of CMCU. The method is based on including the field with code of class of pseudoequivalent OLC into the microinstruction format. The size of CMCU control memory is decreased too due to maximal encoding of collections of microoperations. To decrease the number of macrocells in the block of microinstruction addressing, the special code transformer is used. It transforms OLC codes into codes of their classes. This block can be absent it condition (16) takes place. In this case, the transformation is executed by CMCU block of micromemory.

But such an approach leads to the CMCU U_2 with less performance than this characteristic of CMCU with code sharing. Let us point out that reduction of the number of the macrocells in logic circuit can result in decrease of its levels. It can compensate the negative effect of the memory splitting by two blocks. We made some examples of synthesis using the standard package WebPack. The results show that the number of macrocells is decreased up to 30%, and the number of required memory blocks are decreased up to 50%. Comparison is given for CMCU U_1 and U_2. In the same time, the number of levels in logic circuit of CMCU U_2 is decreased up to 2-3. Let us remind, that the proposed method can be applied only for linear GSA, when condition (5) takes place.

The scientific novelty of proposed method is determined by use of the classes of pseudoequivalent OLC and free resources of EMB for decreasing the number of macrocells in block of microinstruction addressing. Besides, application of encoding of collections of microoperations allows decrease for required memory resources. The practical significance of the method is determined by decrease for the number of macrocells and EMB in CMCU logic circuit, It allows to design the circuits with less amount of hardware in comparison with known control units oriented on linear GSAs.

References

[1] De Micheli G. Synthesis and Optimization of Digital Circuits. – NY: McGraw-Hill, 1994. – 636 pp.

[2] Barkalov A.A., Wegrzyn M. Design of Control Units with Programmable Logic. – Zielona Gora: UZG Press, 2006. – 150 pp.

[3] Macrocell Configurations in CoolRunner XPLA3 CPLDs. http://www.xilinx.com/support/documentation/application_notes/xapp335.pdf

[4] Kania D. Two-level logic synthesis on PALs // Electronic Letters. – 1999, № 7. – pp. 879 – 890.

[5] Barkalov A., Titarenko L. Logic Synthesis for Compositional Microprogram Control Units – Berlin : Springer, 2008. – 272 pp.

[6] Barkalov A.A., Kovalyov S.A., Bieganowski J., Miroshkin A.N. Synthesis of control unit with code sharing and modified linear chains. Machinebuilding and Technosphere XXI // Proc. of XV Int. Scientific Conf. Sevastopol 15-20 September 2008. – Donetsk: DonNTU, 2008. Vol. 4. – P. 54-59. (in Russian).

[7] Barkalov A.A., Krasichkov A.A., Miroshkin A.N. Control Device Synthes with code devising and modification of operator line chains. Sc. Trans. of Donetsk National Technical University. Series "Informatics, Cybernetics and Calculate Techniques". Issue 9 (132) – Donetsk: DonNTU. – 2008. – P. 183-187. (in Russian).

[8] Solovjev V.V. Digital circuit design with CPLD. – Moscow: Hot Line-Telecom, 2001. – 636 pp. (in Russian)

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

Aleksander A. Barkalov – Doctor of Science, Professor of DonNTU (Ukraine), Professor of University of Zielona Gora, Poland.
Dr. Barkalov's scientific interests: digital control units, SoPC Address: Campus A, Budynek Dydaktyczny / A-2
prof. Z. Szafrana str. 2, 65-516 Zielona Gora
E-mail: A.Barkalov@iie.uz.zgora.pl

Larysa A. Titarenko - Doctor of Science, Professor of Kharkiv National Univercity of Radioelectonics (KNURE), Professor of University of Zielona Gora, Poland.
Dr. Titarenko's scientific interests: Digital, adaptive and spatial-time processing of signals in telecommunication. Management and control in communication networks Research of modern digital telecommunication systems and nets.

Aleksander N. Miroshkin – Assistant of Donetsk National Technical University.
Scientific interests: digital control units.