TEST TECHNOLOGY TECHNICAL COUNCIL (TTTC) OF THE IEEE COMPUTER SOCIETY
MINISTRY OF EDUCATION AND SCIENCE, YOUTH AND SPORT OF UKRAINE

KHARKOV NATIONAL UNIVERSITY
OF RADIOELECTRONICS

ISSN 1563-0064

RADIOELECTRONICS & INFORMATICS

Scientific and Technical Journal **Founded in 1997**

№ 4 (59), October – December 2012 **Published 4 times a year**

Certificate of the State Registration КВ № 12097-968 ПР 14.12.2006

CONTENTS

Implementing Control Units for Linear Algorithms

Alexander Barkalov, *IEEE member;* Larysa Titarenko; Alexander Miroshkin

***Abstract*—Two methods are proposed for reducing the number of LUT elements in logic circuits of compositional microprogram control units with code sharing. The methods are based on usage of free resources of embedded memory blocks for representing the codes of the classes of pseudoequivalent operational linear chains. It allows reducing the number of LUTs in the block of microinstruction addressing. The example of application and results of investigations are given.**

Keywords: compositional microprogram control unit, FPGA, LUT elements, embedded memory blocks, hardware reduction.

I. Introduction

As a rule, digital systems include control units responsible for interplay of all system blocks [1]. The behaviour of a control unit (CU) is determined by a control algorithm of a digital system. Such an algorithm can be represented as a graph-scheme of algorithm (GSA) [2]. One of the very important problems connected with design of CUs is a reduction of hardware amount required for implementing the CU's logic circuit [3]. Methods used for solution of this problem depend on peculiarities of both logic elements used for implementing logic circuits and control algorithms to be interpreted [2].

Now, the field-programmable gate arrays (FPGA) [4, 5] are widely used for implementing logic circuits of digital systems. In this article, we discuss FPGA chips including look-up table (LUT) elements and embedded memory blocks (EMB) [6].

The specific of LUT is the limited number of inputs (up to 6-8). It is known that to decrease the amount of LUTs in a circuit it is necessary to decrease the numbers of both arguments and product terms in a Boolean function to be implemented. The specific of EMBs is their ability for reconfiguration in the frames of particular size. For example, the configurations 16k×1, 8k×2, 4k×4, 2k×8, 1024×18, 512×36, and 256×72 exist for typical EMBs [4, 5]. An EMB targets implementing tabular functions. It is quite possible that either some cells, or outputs, or both are not used under implementing some systems of Boolean functions. There are a lot of researches devoted to FPGA-based design of control units [7-11].

Manuscript received November 8, 2012.

Alexander Barkalov: Uniwersytet Zielonogórski, Zielona Góra, Poland. A.Barkalov@iie.uz.zgora.pl

Larysa Titarenko: Uniwersytet Zielonogórski, Zielona Góra, Poland. L.Titarenko@iie.uz.zgora.pl

Alexander Miroshkin: Donetsk National Technical University, Ukraine. MiroshkinAN@gmail.com.

If a control algorithm is represented by a linear GSA, then a control unit can be implemented as a compositional microprogram control unit (CMCU) [12]. The positive feature of CMCU is usage of all recourses of FPGAs (both LUTs and EMBs). It allows obtaining logic circuits with minimum possible amount of LUTs [12].

In this article, some improvements are proposed for the CMCU with code sharing. They are based on specific of both Moore finite-state-machine (FSM) [3] and EMBs. Let us point out that the proposed approach can be used for any model of CMCU [12].

II. The model of CMCU with code sharing

Let a GSA Γ include a set of vertices B and a set of arcs E. Let $B = \{b_0, b_E\} \cup B_1 \cup B_2$ where b_0 is an initial vertex; b_E is a final vertex; B_1 is a set of operator vertices; B_2 is a set of conditional vertices. Operator vertices $b_m \in B_1$ include collections of microoperations $Y(b_m) \subseteq Y$, where $m=\overline{1,M}$, $M = |B_1|$, $Y = \{y_1, \ldots, y_N\}$ is a set of microoperations. Conditional vertices $b_q \in B_2$ contain elements of a set of logical conditions $X = \{x_1, \ldots, x_L\}$. Let us introduce some definitions.

Definition 1. An operational linear chain (OLC) α_g of GSA Γ is a finite vector of operator vertices $\alpha_g = \langle b_{g_1}, \ldots, b_{g_{F_g}} \rangle$ such that an arc $\langle b_{g_i}, b_{g_{i+1}} \rangle \in E$ corresponds to each pair of adjacent components of α_g $(i=\overline{1,F_g-1})$.

Definition 2. An operator vertex $b_m \in B^g$, where $B^g \subseteq B_1$ is a set of operator vertices from the OLC α_g, is called an input of OLC α_g if there is an arc $\langle b_t, b_m \rangle \in E$, where $b_t \notin B^g$.

Definition 3. An operator vertex $b_m \in B^g$ is called an output of OLC α_g if there is an arc $\langle b_m, b_t \rangle \in E$, where $b_t \notin B^g$.

Definition 4. Operational linear chains α_i and α_j are pseudoequivalent operational linear chains (POLC) if there are arcs $\langle b_i, b_t \rangle$, $\langle b_j, b_t \rangle \in E$, where $b_i\,(b_j)$ is the output of OLC $\alpha_i\,(\alpha_j)$.

Definition 5. A GSA Γ is called a linear GSA if the following condition takes place:

$$\frac{M}{G} \geq 2\,. \tag{1}$$

So, a GSA Γ is a linear GSA if the number of its operator vertices at least twice exceeds the minimum number of OLCs. If condition (1) takes place, then the model of CMCU can be used [12]. Let us point out that an arbitrary OLC α_g can have up to $F_g = |B^g|$ inputs and exactly one output, O_g. The inputs of OLC α_g form a set $I(\alpha_g) = \{I_g^1, I_g^2, ...\}$.

Let us use the approach [12] and find the partition C of the set B_1 such that $C = \{\alpha_1, …, \alpha_G\}$. Let G be the minimum possible number of OLCs for the GSA Γ. Let us encode each OLC $\alpha_g \in C$ by a binary code $K(\alpha_g)$ having R_1 bits:

$$R_1 = \lceil \log_2 G \rceil. \tag{2}$$

Let us encode each component $b_{g_i} \in B^g$ by a binary code $K(b_{g_i})$ having R_2 bits:

$$R_2 = \lceil \log_2 (F_{\max}) \rceil. \tag{3}$$

The value of $F_{\max}$ is determined as $F_{\max} = \max (F_1, …, F_G)$. Let us use the elements of a set τ for encoding of the OLCs, whereas the elements of the set T are used for encoding of the components ($|\tau| = R1$, $|T| = R2$).

The encoding of the components is executed in the natural order:

$$K(b_{g_{i+1}}) = K(b_{g_i}) + 1; (g = \overline{1, G}; i = \overline{1, F_g - 1}). \tag{4}$$

Now, an operator vertex $b_m \in B^g$ corresponds to the microinstruction MI_m having the address $A(MI_m)$ determined as

$$A(MI_m) = K(\alpha_g) * K(b_{g_i}). \tag{5}$$

In (5), the sign * means the concatenation, whereas the vertex b_m corresponds to the component b_{g_i} of OLC $\alpha_g \in C$. In address $A(MI_m)$, the codes of OLC and its components are included separately (in the different bits of the address). This approach is called a code sharing.

On the base of (5), the model of CMCU with code sharing (CMCU CS) can be obtained (Fig. 1).

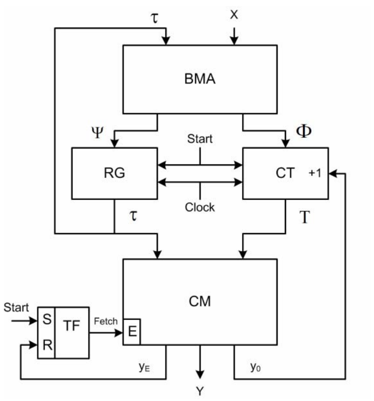


Fig. 1. Structure diagram of CMCU with code sharing

In the CMCU CS, a block of microinstruction addressing (BMA) implements systems of input memory functions for flip-flops of a register RG and a counter CT:

$$\begin{aligned} \Psi &= \Psi(\tau, X); \\ \Phi &= \Phi(\tau, X). \end{aligned} \tag{6}$$

This CMCU operates in the following manner. If Start = 1, then the process begins and zero codes are loaded into both RG and CT. At the same time, a flip-flop of fetching (TF) is set up. Now, there is Fetch = 1, and microinstructions can be fetched out the control memory (CM). Let in the instant t the contents of RG and CT form some address $A(MI_m)$ corresponding to the vertex $b_m \in B^g$. This microinstruction is fetched out the CM. If $b_m \neq O_g$, then a variable y_0 is generated causing incrementing the counter CT. It provides the mode of addressing (4). In the instant t+1the next microinstruction is fetched; it still corresponds to some component of the OLC α_g. If the output O_g is reached, then the variable y_0 is not generated. It allows loading both RG and CT from the outputs of BMA. Now, a transition is executed between the output of OLC α_g and an input of some other OLC (maybe, the same OLC α_g). The process is terminated when a variable y_E is generated. It corresponds to the situation $\langle O_g, b_E \rangle \in E$.

The LUTs and latches are used for implementing logic circuits of BMA, RG, CT and TF, whereas the EMBs are used for implementing the control memory CM. If EMBs have some free recourses (cells, outputs or both), then we propose to use them for decreasing the number of LUT elements in the circuit of BMA.

III. THE MAIN IDEA OF PROPOSED METHOD

As shown in [12], an OLC $\alpha_g \in C$ is an equivalent of some state of Moore FSM. So, pseudoequivalent OLCs correspond to the pseudoequivalent states of Moore FSM [3]. It means that the table of transitions of CMCU CS can be reduced by replacing the pseudoequivalent OLCs by the corresponding class of POLC. It allows decreasing the number of product terms in the functions (6) and, therefore, the reduction of the amount of LUTs in the circuit of BMA. We proposed to keep the codes of classes of POLC in free recourses of EMBs. There are two possible approaches for usage of EMBs:

1. If there are enough free outputs, then the codes of classes of POLC can be included as a separate field in the microinstruction format. Let us call this approach as the expansion of microinstruction format (EMF-approach).
2. If there are enough free cells, then an additional microinstruction with the class code can be included into each OLC of a particular class. Let us call this approach as the modification of OLC (MOLC-approach).

Let us form a set $C_1 \subseteq C$. Let $\alpha_g \in C_1$ if $\langle O_g, b_E \rangle \notin E$. Let us find a partition $\Pi_C = \{B_1, …, B_I\}$ of the set C_1 by the classes of POLCs. It can be done in a trivial way, using the definition 4 from the section 2. Let us encode each class $B_I \in \Pi_C$ by a binary code $K(B_i)$ using R_3 bits, where:

$$R_3 = \lceil \log_2 I \rceil. \tag{7}$$

Let us use the variables $z_r \in Z$ for such an encoding, where $|Z| = R_3$. In this case the system (6) can be transformed in the following way:

$$\begin{aligned}\Psi &= \Psi(Z,X);\\ \Phi &= \Phi(Z,X).\end{aligned} \quad (8)$$

In the case of CMCU CS, the control memory should include M_0 cells. Each of these cells has t_0 bits:

$$M_0 = 2^{R_1+R_2}, \quad (9)$$

$$t_0 = \underline{N} + 2. \quad (10)$$

The value 2 is added to N to take into account the variables y_0 and y_E.

The FPGA chip includes EMBs having V_0 cells if the number of outputs $t_F = 1$. Let us point out that the value of t_F can be taken from some set of fixed values $O_F = \{1, 2, 4, 8, 9, 16, 18, 32, 36, 72\}$. Let us choose the value of $t_F^0 \in O_F$ such that the difference Δt is minimal:

$$\Delta t = t_F^0 - t_0 - R_3 \geq 0. \quad (11)$$

Now, if the condition

$$(V_0 / t_F^0) \geq M_0 \quad (12)$$

takes place, then the EMF-approach can be used. It results in the CMCU FCS (Fig. 2).

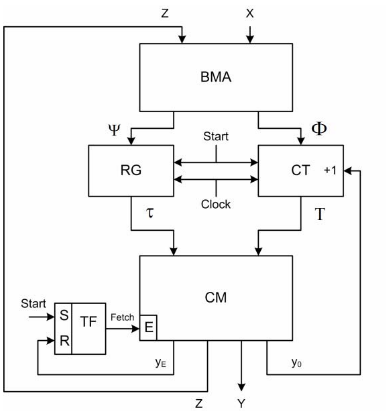


Fig. 2. Structure diagram of CMCU FCS

In the case of MOLC-approach, the number of required memory cells is determined as

$$M_1 = M + G. \quad (13)$$

Let the following condition take place for any OLC $\alpha_g \in C_1$:

$$F_g \leq 2^{R_2} - 1. \quad (14)$$

In this case, the introduction of additional microinstructions does not increase the value of R_2 in comparison with (3). Now, the value of t_F^0 is chosen from the following condition

$$\begin{aligned}&\Delta t = t_F^0 - t_0 \geq 0;\\ &\Delta t \to \min.\end{aligned} \quad (15)$$

If condition (14) takes place, then the MOLC-approach can be used leading to the CMCU MCS (Fig. 3).

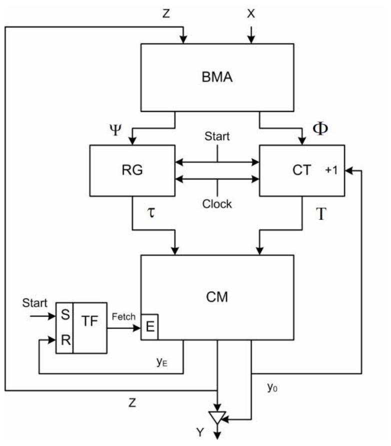


Fig. 3. Structure diagram of CMCU MCS

Let us point out that the EMF-approach is more preferable. It does not require additional (idle) cycles of CMCU. So, it is necessary to start from the model of CMCU FCS. If this model cannot be used, then the model of CMCU MCS should be tried. Let us discuss the case when both models can be used and, moreover, only one EMB is enough for implementing the control memory. In other cases, the proposed methods need some modifications. The modifications are not complex, and, because of it, they are beyond the scope of this article.

The proposed design methods include the following steps:

1. Constructing the sets C, C_1, Π_C for a given GSA Γ.
2. Encoding of OLC $\alpha_g \in C$ and their components.
3. Encoding of the classes $B_i \in \Pi_C$.
4. Constructing the content of control memory.
5. Constructing the table of transitions of CMCU and finding the system (8).
6. Implementing the logic circuit using given FPGA chip.

The step 1 is executed using the methods from [12].As a result, the number G of OLC $\alpha_g \in C$ is minimal. The partition Π_C is formed using the definition 4.

The encoding of OLC should be executed in a way minimizing the number of terms in (8). The well-known methods [1] can be used to solve this problem. The components of OLC $\alpha_g \in C$ are encoded in a trivial way. The first component of any OLC has the code whose decimal equivalent is equal to zero. The codes of the second components are equal to 1, the third – to 2 and so on. This style of encoding satisfies to (4). The codes of classes do not affect the number o LUTs in the circuit of BMA.

The content of CM is represented by the table having the fields $A(MI_m)$, $Y(b_m)$, y_0, y_E, $K(B_i)$. In the case of CMCU FCS, the fields $Y(b_m)$ and $K(B_i)$ require different bits. In the case of CMCU MCS, these fields share the same bits of EMB. The number of required bits is determined as max $(N+2; R_3)$.

To construct the table of CM, it is necessary to transform the initial GSA Γ [12]. If a vertex $b_m \in B^g$ is not the output of OLC $\alpha_g \in C$, then the variable y_0 is introduced into this vertex. If $\langle b_m, b_E\rangle \in E$, then the variable y_E is introduced into the vertex $b_m \in B_1$.

The table of transitions is constructed on the base of generalized formulae of transitions [12]:

$$B_i \rightarrow \bigvee_{h=1}^{H_i} X_h b_m; (i=\overline{1,I}). \qquad (16)$$

In (16), X_h is a conjunction of logical conditions determining the transition from the output of any OLC $\alpha_g \in B_i$ to the operator vertex b_m; H_i is the number of transitions from this output. The system (16) leads to the table of transitions having the following columns: B_i, $K(B_i)$, b_m, $A(MI_m)$, X_h, Ψ_h, Φ_h, h. Here $\Psi_h \subseteq \Psi$ is a set of input memory functions for the RG; $\Phi_h \subseteq \Phi$ is a set of input memory functions for the CT; h is a number of transitions. The system (8) is constructed as the following:

$$D_r = \bigvee_{h=1}^{H} C_{rh} B_h X_h; (r=\overline{1,R_2+R_3}). \qquad (17)$$

In (17), C_{rh} is the Boolean variable equal to 1 iff the function D_r is written in the h-th row of the table, B_h is a conjunction of variables $z_r \in Z$ corresponding the code $K(B_i)$ for the h-th row of the table $(h=\overline{1,H})$.

The last step is reduced to implementation of the logic circuit of CMCU using some standard tools [4, 5].

IV. An example of application of proposed methods

Let some GSA Γ_1 include $M = 17$ operator vertices. Let these vertices form the set $C = \{\alpha_1, \ldots, \alpha_8\}$ where $\alpha_1 = \langle b_1, b_2\rangle$, $\alpha_2 = \langle b_3, b_4, b_5\rangle$, $\alpha_3 = \langle b_6, b_7\rangle$, $\alpha_4 = \langle b_8, b_9, b_{10}\rangle$, $\alpha_5 = \langle b_{11}, b_{12}\rangle$, $\alpha_6 = \langle b_{13}, b_{14}\rangle$, $\alpha_7 = \langle b_{15}, b_{16}\rangle$ and $\alpha_8 = \langle b_{17}\rangle$. It means $G = 8$, condition (1) takes place and the model of CMCU can be used.

Let $\alpha_8 \notin C_1$, $L = 4$, $N = 6$ and $\Pi_C = \{B_1, \ldots, B_4\}$, where $B_1 = \{\alpha_1, \alpha_6\}$, $B_2 = \{\alpha_2, \alpha_3, \alpha_5\}$, $B_3 = \{\alpha_4\}$, $B_4 = \{\alpha_7\}$.Because there is $G = 8$, then $R_1 = 3$ and $\tau = \{\tau_1, \tau_2, \tau_3\}$. It can be found that $F_{\max} = 3$; it means that $R_2 = 2$ and $T = \{T_1, T_2\}$. Let us encode the OLC $\alpha_g \in C$ in a trivial way: $K(\alpha_1) = 000$, $K(\alpha_2) = 001$, …, $K(\alpha_8) = 111$. The first components at any OLC $\alpha_g \in C$ have the code 00, the second components have the code 01, the third components have the code 10 and the fourth components have the code 11. Let us point out that in the discussed example the fourth components are added into some OLCs of CMCU MCS.

The addresses of microinstructions can be found from Table I. In this table, the symbols $(b_{18}) - (b_{24})$ denote additional vertices introduced for CMCU MCS.

Let the following system of generalized formulae of transitions can be obtained after analysis of the GSA Γ_1:

$$B_1 \rightarrow x_1 b_3 \vee \bar{x}_1 x_2 b_8 \vee \overline{x_1 x_2} b_6; \quad B_2 \rightarrow x_4 b_{15} \vee \bar{x}_4 b_{17}; \qquad (18)$$
$$B_3 \rightarrow x_3 b_{11} \vee \bar{x}_3 b_{13}; \quad B_4 \rightarrow x_5.$$

TABLE I
ADDRESSES OF MICROINSTRUCTIONS

OLC	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8
$\tau_3\tau_2\tau_1$	000	001	010	011	100	101	110	111
T_2T_1								
00	b_1	b_3	b_6	b_8	b_{11}	b_{13}	b_{15}	(b_{17})
01	b_2	b_4	b_7	b_9	b_{12}	b_{14}	b_{16}	–
10	(b_{18})	b_5	(b_{20})	b_{10}	(b_{22})	(b_{23})	(b_{24})	–
11	–	(b_{19})	–	(b_{21})	–	–	–	–

Let us encode the classes $B_i \in \Pi_C$ in a trivial way: $K(B_1) = 00$, …, $K(B_4) = 11$. Using these codes and the system (18), the table of transitions can be constructed (Table II).

TABLE II
TABLE OF TRANSITIONS OF CMCU

B_i	$K(B_i)$	b_m	$A(MI_m)$	X_h	Ψ_h	Φ_h	h
B_1	00	b_3	00100	x_1	D_1	–	1
		b_8	01100	$\bar{x}_1 x_2$	$D_2 D_1$	–	2
		b_6	01000	$\bar{x}_1 \bar{x}_2$	D_2	–	3
B_2	01	b_{15}	11000	x_4	$D_3 D_2$	–	4
		b_{17}	11100	$\bar{x}_4$	$D_3 D_2 D_1$	–	5
B_3	10	b_{11}	10000	x_3	D_3	–	6
		b_{13}	10100	$\bar{x}_3$	$D_3 D_1$	–	7
B_4	11	b_5	00110	1	D_1	D_4	8

The addresses of microinstructions A(IMm) are taken from Table 1 using the expression (5). For example, $b_5 \in B^2$ and $K(\alpha_2) = 001$. Therefore, $A(MI_5) = K(\alpha_2)*K(b_5) = 00110$.

Table 2 is the base for constructing the system (8). In the discussed case, this system is the following one:

$$D_1 = F_1 \vee F_2 \vee F_5 \vee F_7 \vee F_8; \quad D_2 = F_2 \vee F_3 \vee F_4 \vee F_5; \qquad (19)$$
$$D_3 = F_4 \vee F_5 \vee F_6 \vee F_7; \quad D_4 = F_8,$$

where $F_1 = \overline{z_1 z_2} x_1$, $F_2 = \overline{z_1 z_2 x_1} x_2$, $F_3 = \overline{z_1 z_2 x_1 x_2}$, …, $F_8 = z_1 z_2$.

Let $Y(b_3) = \{y_1, y_3, y_0\}$, $Y(b_4) = \{y_4, y_0\}$, $Y(b_5) = \{y_5\}$, $Y(b_6) = \{y_2, y_0\}$, $Y(b_7) = \{y_3, y_6\}$. Using addresses from Table I, the following fragment of content of control memory can be created for CMCU FCS (Table III).

Because the relation $\alpha_2 \in B_2$ takes place, the code $K(B_2) = 01$ is placed into the cell with address 00110. This cell corresponds to the output of OLC α_2. This very code is placed into the cell corresponding to the output of OLC α_3.

In the case of CMCU MFS, the second and the third bits of microinstruction are used either as microoperations y_1, y_2 or variables z_2, z_1 (Table IV).

TABLE III
PART OF CONTROL MEMORY FOR CMCU FCS

Address					Microinstruction										b_m	α_i
τ_3	τ_2	τ_1	T_2	T_1	y_E	y_1	y_2	y_3	y_4	y_5	y_6	y_0	z_2	z_1		
0	0	1	0	0	0	1	0	1	0	0	0	1	0	0	b_3	
0	0	1	0	1	0	0	0	0	1	0	0	1	0	0	b_4	α_2
0	0	1	1	0	0	0	0	0	0	1	0	0	**0**	**1**	b_5	
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	–	
0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	b_6	
0	1	0	0	1	0	0	0	1	0	0	1	0	**0**	**1**	b_7	α_3
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	–	
0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	–	

TABLE IV
PART OF CONTROL MEMORY FOR CMCU MCS

Address					Microinstruction								b_m	α_i
τ_3	τ_2	τ_1	T_2	T_1	y_E	y_1 z_2	y_2 z_1	y_3	y_4	y_5	y_6	y_0		
0	0	1	0	0	0	1	0	1	0	0	0	1	b_3	
0	0	1	0	1	0	0	0	0	1	0	0	1	b_4	
0	0	1	1	0	0	0	0	0	0	1	0	0	b_5	α_2
0	0	1	1	1	0	**0**	**1**	0	0	0	0	0	b_{19}	
0	1	0	0	0	0	0	1	0	0	0	0	1	b_6	
0	1	0	0	1	0	0	0	1	0	0	1	0	b_7	α_3
0	1	0	1	0	0	**0**	**1**	0	0	0	0	0	b_{20}	
0	1	0	1	1	0	0	0	0	0	0	0	0	–	

We do not show the logic circuits of these CMCUs. But we developed CAD tools allowing synthesis of proposed models of CMCU. Our CAD tools use Xilinx ISE WebPack to produce a final implementation.

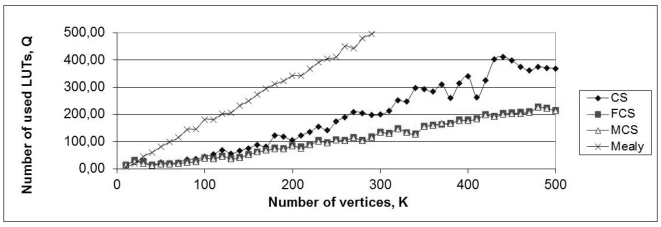
V. EXPERIMENTAL RESULTS

Our CAD system is based on the following principles. An initial GSA is represented in the XML format. One of its blocks generates VHDL-description of a given model of CMCU, together with data using for programming EMBs. This information is transferred into the system Xilinx ISE WebPack. Next, the implementation of a logic circuit is executed. The initial GSAs are generated by a special generator, which is the part of CAD tools:

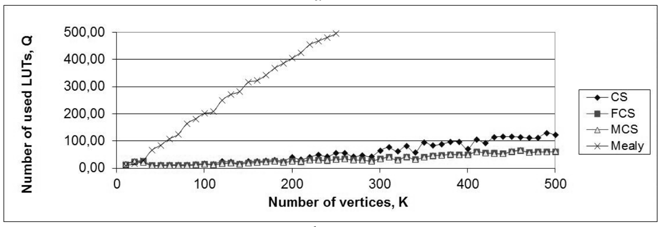
– the number of vertices K is changed from 10 to 500;
– the part of operator vertices is changed from 50 % to 90 %;
– the number of microoperations $N = 15$;
– the number of logical conditions $L = 5$.

For each GSA, the following control units were implemented: CMCU with code sharing, CMCU FCS, CMCU MCS, and Mealy FSM. The experimental results are shown on diagrams. Each point on the diagrams is an average result obtained for five different GSAs with similar parameters.

The numbers of LUTs required for implementing logic circuits of different control units are shown on Fig. 4. The results for GSAs with 70 % of operator vertices are shown on Fig. 4, a. The results for GSAs with 90 % of operator vertices are shown on Fig. 4, b. Analysis of Fig. 4 shows that the proposed models of CMCU require fewer amounts of LUTs than both Mealy FSM and the base model of CMCU CS. Moreover, the growth in the number of operator vertices leads to increasing the hardware amount for Mealy FSM. But it has quite opposite effect in the cases of CMCU.



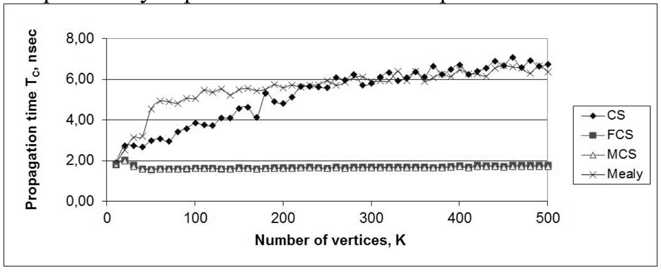
a



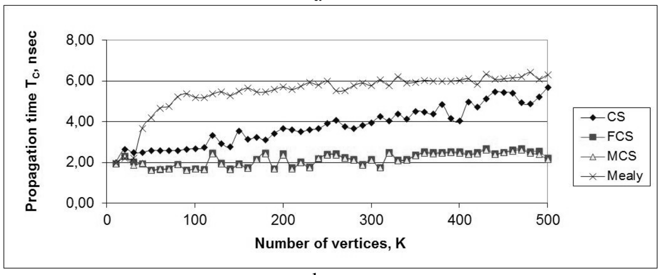
b

Fig. 4. Number of LUT elements in logic circuits of control units

The temporal characteristics of different control units are shown in Fig. 5. As in previous case, results for GSAs with 70 % of operator vertices are shown in Fig. 5, a. Results for GSAs with 90 % of operator vertices are shown in Fig. 5, b. Both diagrams show minimal possible propagation time T_C for the control units under investigation. The analysis if Fig. 5 shows that the proposed models provide higher performance in comparison with both Mealy FSM and CMCU CS. It is interesting that the propagation time does not practically depend on the number of operator vertices.



a



b

Fig. 5. Minimal propagation time for different control units

So, if the CMCU FCS and MCS require the same amount of EMBs, their characteristics (number of LUT elements and propagation time) are practically identical. Obviously, a control algorithm's execution requires more cycles in CMCU MCS than in the case of equivalent CMCU FCS. It is connected with existence of additional microinstructions in the control memory of CMCU MCS. So, if there are such conditions that both proposed models can be used, then the model of CMCU FCS is more preferable.

Let us point out that results of investigation are obtained for the FPGA Spartan-3 by Xilinx. If other chips are used, the results can be different. But the tendency remains.

VI. Conclusion

As the results of investigations show, the proposed methods allow decreasing the hardware amount (in average) to 40% in comparison with known design methods.

One of the results of investigation is obtaining the formula showing the hardware amount required for implementing CMCU with code sharing and proposed modifications. Let us point out that this formula is correct for FPGA chips having LUT elements with four inputs (for example, for Spartan-3 family by Xilinx). The formula is the following:

$$Q = (-0.026P_1^2 + 2.56P_1 - 10.11) \cdot K \qquad (20)$$

In (20), Q is the number of LUTs in a logic circuit, K is the number of vertices in the GSA Γ, P_1 is a part of operator vertices in a GSA Γ ($0.5 \le P_1 \le 1$). Let us point out that the expression (18) is correct for $L = 5$. If similar formulae include L as a variable, then they can be used for preliminary estimation of hardware amount in the case of an arbitrary GSA.

The time Clock for proposed models is in the interval [1.7 nsec; 2.5 nsec]. As our investigations show, this interval is equal to [5 nsec; 6 nsec] for Mealy FSM. Moreover, this characteristic for CMCU depends only on the type of FPGA. In the case of Mealy FSM, delays increase with increasing the numbers of vertices in a control algorithm.

So, the proposed models of control units allow designing logic circuits with better hardware and timing characteristics in comparison with known models. Let us point out that they can be used only if a control algorithm is represented by a linear graph-scheme of algorithm.

References

[1] DeMicheli G., Synthesis and Optimization of Digital Circuits, McGraw-Hill, New York, 1994.

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

[3] Barkalov A., Titarenko L., Logic Synthesis for FSM-Based Control Units, Springer, Berlin, 2009.

[4] Spartan-3 FPGA Data Sheets, www.xilinx.com/support/documentation/spartan-3.htm (Oct 22, 2012)

[5] Embedded Memory in Altera FPGAs, www.altera.com/technology/memory/embedded/mem-embedded.html (Oct 22, 2012)

[6] Navabi Z., Embedded Core Design with FPGAs, McGraw-Hill, New York, 2007.

[7] R. Senhadji-Navarro, I. García-Vargas, G. Jiménez-Moreno, A. Civit-Ballcels, ROM-based FSM implementation using input multiplexing in FPGA devices, Electronics Letters (2004), 40 (20), 1249-1251.

[8] M. Rawski, H. Selvaraj, T. Łuba, An application of functional decomposition in ROM-based FSM implementation in FPGA devices, Journal of Systems Architecture (2005), 51 (6-7), 424-434.

[9] V. Sklyarov, Synthesis and Implementation of RAM-Based Finite State Machines in FPGAs, 10th International Conference «Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing» Proceedings, FPL 2000 Villach, Austria (August 27–30, 2000), 718-727.

[10] A. Tiwari, K.A. Tomko, Saving power by mapping finite-state machines into embedded memory blocks in FPGAs, Proceedings - Design, Automation and Test in Europe Conference and Exhibition (2004), 2, 916-921.

[11] E. Garcia, Creating Finite State Machines Using True Dual-Port Fully Synchronous SelectRAM Blocks, Xcell Journal (2000), Issue 38, 36-38.

[12] Barkalov A., Titarenko L., Logic Synthesis for Compositional Microprogram Control Units, Springer, Berlin, 2008.

Alexander A. Barkalov received Doctor of Technical Sciences degree in Computer Science from Institute of Cybernetics named after V.M. Glushkov (Kiev, Ukraine). From 2003 he is a Professor of Computer Engineering at the Department of Informatics and Electronics, University of Zielona Gora, Poland. His current research interests include theory of digital automata, especially the methods of synthesis and optimization of control units implemented with field-programmable logic devices. Address: Campus A, Budynek Dydaktyczny / A-2 prof. Z. Szafrana str. 2, 65-516 Zielona Gora. E-mail: A.Barkalov@iie.uz.zgora.pl

Alexander N. Miroshkin, PhD student in Computer Science from Donetsk National Technical University (Donetsk, Ukraine). His current research interests include theory of digital automata. Address: Donetsk National Technical University, Ukraine. MiroshkinAN@gmail.com.

Larysa A. Titarenko received the M.Sc. (1993), PhD (1996) and Doctor of Technical Sciences (2005) degree in Telecommunications from Kharkov National University of Radioelectronics, Ukraine. From 2007 she is a Professor of Telecommunications at the Institute of Informatics and Electronics, University of Zielona Gora, Poland. Her current research interests include theory of telecommunication systems, theory of antennas and theory of digital automata and its applications. Address: Campus A, Budynek Dydaktyczny / A-2 prof. Z. Szafrana str. 2, 65-516 Zielona Gora. E-mail: A.Barkalov@iie.uz.zgora.pl

Camera-ready was prepared in Kharkov National University of Radio Electronics
Lenin Ave, 14, Kharkov, 61166, Ukraine
Approved for publication: 27.12.2013. Format 60×84 1/8.
Relative printer’s sheets: 9,2. Circulation: 300 copies.
Published by SPD FL Stepanov V.V.

Рекомендовано Вченою радою Харківського національного
університету радіоелектроніки (протокол № 7 від 27.03.2012)
61166, Харків, просп. Леніна, 14.
Підписано до друку 27.12.2013. Формат 60×841/8.
Умов. друк. арк. 9,2. Тираж 300 прим. Ціна договірна.
Віддруковано у ФОП Степанов В.В.