УДК 004.274

О.О. БАРКАЛОВ
Університет Зеленогурський, Зелена Гура, Польща; Донецький національний університет імені Василя Стуса, Вінниця, Україна,
e-mail: *A.Barkalov@iie.uz.zgora.pl*.

Л.О. ТІТАРЕНКО
Університет Зеленогурський, Зелена Гура, Польща; Харківський національний університет радіоелектроніки, Харків, Україна,
e-mail: *L.Titarenko@iie.uz.zgora.pl*.

А.В. БАЄВ
Донецький національний університет імені Василя Стуса;
фірма Peoly, Вінниця, Україна,
e-mail: *a.baev@donnu.edu.ua*.

О.В. МАТВІЄНКО
Інститут кібернетики ім. В.М. Глушкова НАН України, Київ, Україна,
e-mail: *avmatv@ukr.net*.

ОПТИМІЗАЦІЯ СХЕМИ КОМПОЗИЦІЙНОГО МІКРОПРОГРАМНОГО ПРИСТРОЮ КЕРУВАННЯ З РОЗДІЛЕННЯМ КОДІВ

Анотація. Запропоновано метод зменшення кількості елементів LUT у схемі композиційного мікропрограмного пристрою керування (КМПК) з розділенням кодів. Метод ґрунтується на подвійному кодуванні операторних лінійних ланцюгів. Кожний ланцюг має код як елемент загальної множини операторних лінійних ланцюгів і як елемент класу цієї множини. Такий підхід дає змогу отримати дворівневу схему адресації мікрокоманд. Керувальну пам'ять КМПК реалізовано на вбудованих блоках пам'яті EMB. Розглянуто приклад синтезу і наведено аналіз запропонованого методу.

Ключові слова: композиційний мікропрограмний пристрій керування, LUT, EMB, розділення кодів.

ВСТУП

Сучасні цифрові системи — це композиції комбінаційних і послідовних блоків [1, 2]. Одним із послідовних блоків є пристрій керування (ПК), який координує роботу інших блоків системи [3]. Як відомо, характеристики ПК значною мірою визначають якість цифрової системи [4, 5]. Це висвітлюється у багатьох роботах, пов'язаних з оптимізацією характеристик ПК.

Чимало сучасних цифрових систем реалізується в базисі НВІС типу FPGA (field-programmable logic arrays) [6, 7]. Щоб покращити характеристики ПК, у базисі FPGA необхідно зменшити площу кристала, яку займає схема ПК [8]. Зазвичай це скорочує час затримки (збільшує частоту роботи схеми) і зменшує споживану потужність [2, 9]. Останнє має вкрай важливе значення для мобільних і автономних систем [10].

Методи оптимізації характеристик ПК багато в чому залежать від особливостей реалізованого алгоритму керування. Якщо алгоритм є лінійним [11], тоді для його реалізації доцільно використовувати модель композиційного мікропрограмного пристрою керування (КМПК) [12, 13]. Цей ПК є автоматом Мура, в якому регістр кодів станів замінено лічильником адреси (ЛА) мікрокоманд. Такий КМПК містить схему адресації мікрокоманд, лічильник і керувальну пам'ять (КП) [11]. У КП зберігаються мікрокоманди (МК), що складаються з мікрооперацій (МО) і внутрішніх керувальних сигналів [11].

Мікросхеми FPGA найбільш придатні для реалізації схем КМПК. Блоки вбудованої пам'яті EMB (embedded memory block) [14, 15] доцільно використовувати для реалізації схеми КП. Схема адресації КП може бути реалізована на елементах табличного типу (LUT, look-up table) [14]. Елементи схеми поєднуються за допомогою програмованих міжз'єднань [9], ЛА розподілений між блоками CLB (configurable logic block) [14].

Вадою елементів LUT є вкрай обмежена кількість входів [5, 6]. Це зумовлює потребу у функціональній декомпозиції [16, 17] систем булівських функцій (СБФ), що являють собою комбінаційну частину КМПК. У разі застосування функціональної декомпозиції отримуємо багаторівневі схеми з нерегулярною структурою міжз'єднань. Такі схеми мають більшу споживану потужність і меншу швидкодію, ніж їхні однорівневі аналоги [9].

У запропонованій роботі розглядається метод зменшення кількості елементів LUT і їхніх рівнів у схемі КМПК з розділенням кодів [11]. Метод базується на знаходженні певного розбиття множини операторних лінійних ланцюгів (ОЛЛ). Метод є розвитком ідей подвійного кодування станів [18]. Для представлення керувальних алгоритмів використовується мова граф-схем алгоритмів (ГСА) [19]. Схема КМПК реалізується у вигляді композиції елементів LUT, блоків EMB, програмованих тригерів і міжз'єднань.

РЕАЛІЗАЦІЯ СХЕМ КМПК У БАЗИСІ FPGA

Методи синтезу КМПК базуються на формуванні множини ОЛЛ $C = \{\alpha_1, \ldots, \alpha_G\}$ [11]. Кожна ОЛЛ — це вектор, що містить лише операторні вершини ГСА Г. Операторні вершини відповідають мікрокомандам, що зберігаються в КП.

Множина вершин ГСА складається з M операторних вершин, початкової вершини b_0, фінальної вершини b_E та умовних вершин $x_l \in X = \{x_1, \ldots, x_L\}$, що містять логічні умови (ЛУ). Операторні вершини мають набори мікрооперацій (МО) $Y_q \subseteq Y$, де $Y = \{y_1, \ldots, y_N\}$ — множина мікрооперацій.

Наприклад, ГСА Γ_1 (рис. 1) містить $M = 19$ операторних і дев'ять умовних вершин. Аналіз ГСА Γ_1 дає змогу знайти параметри $N = 9$ і $L = 6$. Операторні вершини утворюють множину $B = \{b_1, \ldots, b_{19}\}$. Розглянемо низку визначень [11], необхідних для подальшого викладу матеріалу.

Означення 1. Операторним лінійним ланцюгом ГСА Г називається скінченна послідовність операторних вершин $\alpha_g = \langle b_{g1}, \ldots, b_{gF_g} \rangle$ така, що для будь-якої пари сусідніх компонент вектора α_g існує дуга $\langle b_{gi}, \ldots, b_{gi+1} \rangle$, де i — номер компоненти послідовності α_g.

Означення 2. Входом ОЛЛ α_g називається вершина, вхід якої зв'язаний з виходом початкової або умовної вершини, або з виходом операторної вершини, що не входить в ОЛЛ α_g.

Означення 3. Виходом ОЛЛ α_g називається вершина, вихід якої зв'язаний з входом вершини, що не входить в ОЛЛ α_g.

У КМПК міститься $M_0 = M + 1$ мікрокоманд, які складаються з мікрооперацій $y_n \in Y$ і спеціальних змінних y_0 (безумовний перехід всередині в ОЛЛ) і y_E (закінчення алгоритму). Домовимося використовувати унітарне кодування МО [11], тоді для представлення МК потрібно N розрядів. Комірка КП з нульовою адресою відповідає вершині b_0, що й пояснює різницю між M і M_0.

Кожній вершині $b_m \in B$ відповідає МК MI_m з адресою A_m. Мікрокоманди відповідають станам еквівалентного автомата Мура [11], а адреси MI_m є аналогами кодів станів. Розрядність адреси МК визначається $R = \lceil \log_2 M_0 \rceil$.

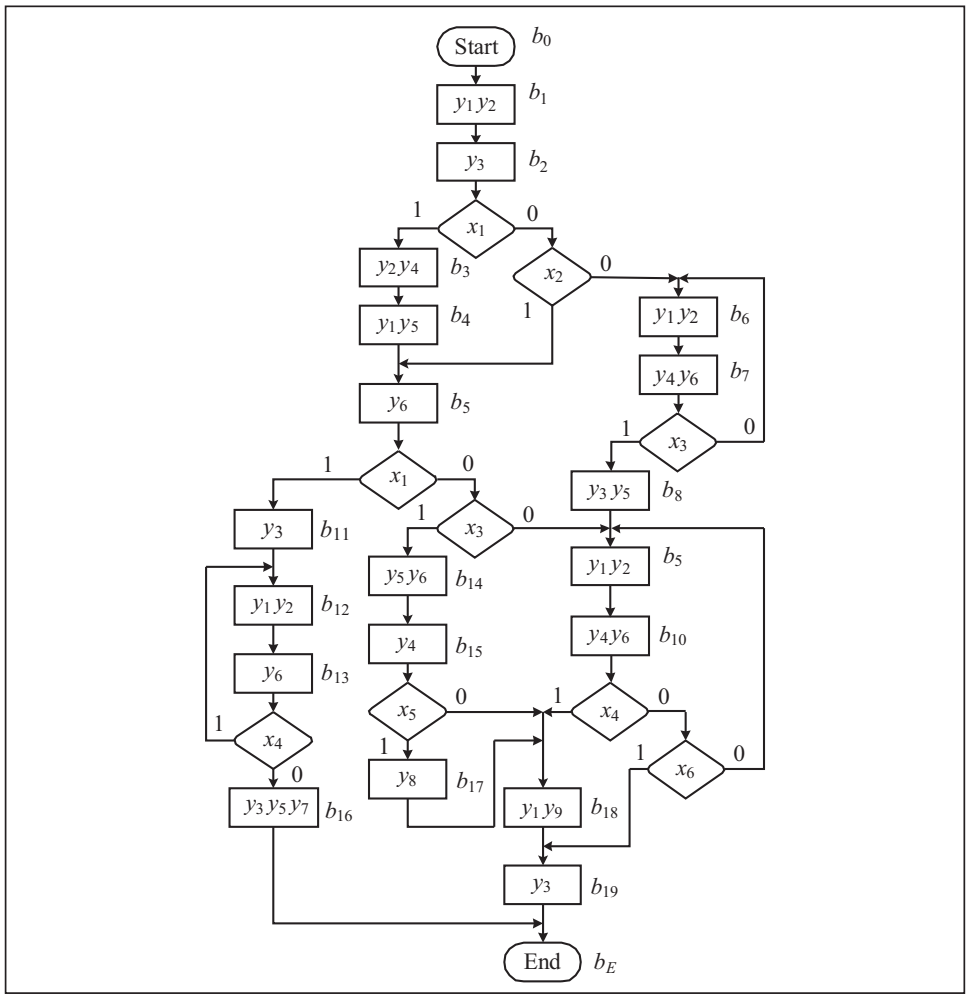

Рис. 1. Початкова граф-схема алгоритму Γ_1

Нехай для ГСА Г сформовано множину $C = \{\alpha_1, \dots, \alpha_G\}$, що задовольняє такі умови [11]:

— кожна вершина $b_m \in B$ належить лише одній ОЛЛ $\alpha_g \in C$;

— усі операторні вершини належать ОЛЛ;

— кількість ОЛЛ G є мінімально можливою.

Для адресації МК використовується природний порядок [20]. До того ж для будь-якої пари сусідніх компонент b_{gi}, b_{gi+1} будь-якої ОЛЛ $\alpha_g \in C$ виконується умова

$$A_{gi+1} = A_{gi} + 1 \ (i = \overline{1, F_g - 1}). \tag{1}$$

Адреси МК зберігаються в лічильнику, що складається з R тригерів типу D [20]. Адреси представляються змінними з множини $T = \{T_1, \dots, T_R\}$. Для зміни вмісту ЛА використовуються функції збудження пам'яті з множини $\Phi = \{D_1, \dots, D_R\}$. Якщо $y_0 = 1$, вміст ЛА збільшується на 1 (ЛА = ЛА +1). Якщо $y_0 = 0$, вміст ЛА визначається функціями $D_r \in \Phi$ (ЛА = Ф). Якщо $y_E = 1$, функціонування КМПК припиняється. За сигналом *Start* в ЛА записується нульова адреса. За сигналом *Clock* дозволяється зміна вмісту ЛА. Ці принципи визначають структурну схему КМПК U_1 (рис. 2), який називають КМПК із загальною пам'яттю.

Блок адресації МК формує функції

$$\Phi = \Phi(T, X). \qquad (2)$$

Ці функції мають нерегулярний характер [11]. У реальних пристроях кожна з функцій (2) може залежати від 30 до 40 аргументів [19].

Для реалізації схеми КП доцільно використовувати блоки EMB. Ці блоки мають можливість конфігурації [5, 6], тобто одночасної зміни кількості адресних входів (S_A) і виходів даних (t_F) за постійної ємності (V_0). Параметр V_0 визначається як $V_0 = 2^{S_A} \times t_F$.

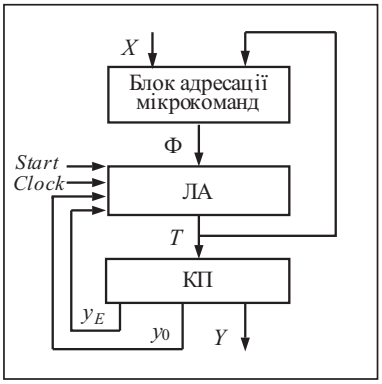

Рис. 2. Структурна схема КМПК U_1

Пара $\langle S_A, t_F \rangle$ визначає конфігурацію EMB [5]. Для сучасних FPGA [14] характерними є блоки EMB ємністю 32Kb і конфігураціями $32\text{K} \times 1$, $16\text{K} \times 2$, $8\text{K} \times 4$, $4\text{K} \times 8$, $2\text{K} \times 16$, $1\text{K} \times 32$, 512×64, 256×128 біт. Блоки EMB можна налаштувати з урахуванням характеристик КП.

Блоки EMB є ефективним засобом для реалізації КП, якщо виконується умова

$$(S_A = R \wedge (S_A + 1) > R) = 1. \qquad (3)$$

Отже, блок EMB має $t_A = V_0 / 2^{S_A}$ виходів, а для реалізації КП потрібно n_E блоків:

$$n_E = \left\lceil \frac{N+2}{t_A} \right\rceil. \qquad (4)$$

Якщо серед пар $\langle S_A, t_F \rangle$ немає конфігурації, що задовольняє (3), пам'ять буде багаторівневою. У цьому разі різко зростає споживана потужність і зменшується швидкодія КМПК.

Блок адресації МК доцільно реалізовувати на елементах LUT. Останній має $S_L \le 6$ входів [14] і здатний виконувати довільну функцію, що має не більше S_L аргументів. Значення $S_L = 6$ вважається оптимальним [21] для забезпечення балансу між займаною площею кристала, швидкодією і споживаною потужністю.

Однак аналіз бібліотеки [22] показує, що функції (2) можуть мати до 25 аргументів. Тож схема блока адресації КМПК U_1 може містити кілька рівнів елементів LUT.

Для зменшення кількості елементів зворотного зв'язку в [11] запропоновано метод поділу кодів. Особливість методу полягає у кодуванні ОЛЛ і природній адресації МК.

Нехай ГСА Г має G ОЛЛ $\alpha_g \in C$. Поставимо у відповідність ОЛЛ $\alpha_g \in C$ код $K(\alpha_g)$ розрядності

$$R_C = \lceil \log_2 G \rceil. \qquad (5)$$

Для кодування ОЛЛ використаємо змінні $T_r \in T_C$, де $|T_C| = R_C$.

Поставимо у відповідність кожній вершині $b_m \in B$ код $K(b_m)$, що ідентифікує цю вершину як елемент ОЛЛ $\alpha_g \in C$. Нехай $F_{\max}$ — максимальна кількість компонент в ОЛЛ для деякої ГСА Г. Тоді для кодування вершин достатньо R_M змінних:

$$R_M = \lceil \log_2 F_{\max} \rceil. \qquad (6)$$

Для кодування вершин використовуємо елементи множини T_M ($|T_M| = R_M$).

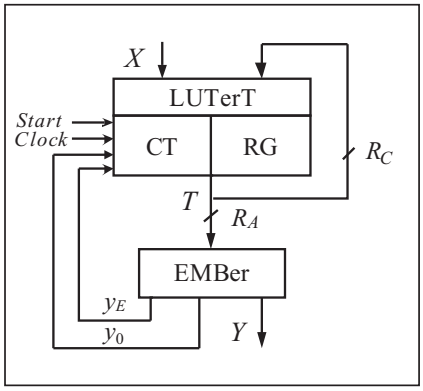

Рис. 3. Структурна схема КМПК U_2

Адреса A_m мікрокоманди MI_m, що відповідає вершині b_q, яка входить в ОЛЛ $\alpha_g \in C$, визначається конкатенацією кодів $K(\alpha_g)$ і $K(b_q)$:

$$A_m = K(\alpha_g) * K(b_q), \qquad (7)$$

де * — операція конкатенації. Вочевидь, адреса МК має $R_A = R_C + R_M$ розрядів.

Кодування вершин здійснюється так, що виконується умова

$$K(b_{gi+1}) = K(b_{gi}) + 1 \ \ (g = \overline{1, G}). \qquad (8)$$

Таким чином, умови (1) і (8) тотожні.

Позначимо LUTer блок, що складається з елементів LUT, і EMBer — блок, що складається з блоків EMB. Тоді КМПК U_2 з розділенням кодів має структурну схему, показану на рис. 3.

У КМПК U_2 блок LUTerT реалізує систему (2). Однак лише R_C елементів множини T є аргументами функцій (2). Код $K(\alpha_g)$ зберігається в RG, а код $K(b_q)$ — у ЛА. Обидва блоки (RG і CT) розподілені між елементами LUT. Керувальна пам'ять КМПК U_2 реалізована у вигляді блока EMBer. Основною перевагою КМПК U_2 є те, що функції (2) мають менше аргументів, ніж у рівнозначного КМПК U_1 [11].

Нехай функція $D_r \in \Phi$ має NA_r аргументів. Якщо виконується умова

$$NA_r > S_L \ \ (r \in \{1, \ldots, R_A\}), \qquad (9)$$

тоді блок LUTerT має кілька рівнів логіки.

У цій статті запропоновано метод зменшення кількості елементів LUT у схемі блока LUTerT за виконання умови (9). Метод є адаптацією методу подвійного кодування станів [18]. У разі виконання певних умов схема блока LUTerT має точно два рівні елементів LUT.

ОСОБЛИВІСТЬ ЗАПРОПОНОВАНОГО МЕТОДУ

Нехай умова (9) виконується для деяких функцій з СБФ (2). У цьому разі пропонується перетворити блок LUTerT, використовуючи ідеї з [18].

Знайдемо множину ОЛЛ $C(\Gamma) \subseteq C$ таку, що $\alpha_g \in C(\Gamma)$, якщо вихід O_g не зв'язаний дугою з входом фінальної вершини. Наприклад, ОЛЛ $\alpha_g \notin C(\Gamma)$, якщо їхніми виходами є вершини b_{16}, b_{19} (див. рис. 1).

Знайдемо розбиття Π_C множини $C(\Gamma)$ на класи $C^1, \ldots, C^K$, яке задовольняє такі умови:

— кількість класів K є мінімально можливою;

— для класів $C^k \in \Pi_C$ виконується умова

$$R_k + L_k \le S_L \ \ (k \in \{1, \ldots, K\}), \qquad (10)$$

де R_k — кількість змінних, необхідних для кодування ОЛЛ $\alpha_g \in C^k$; L_k — кількість ЛУ, що визначають переходи з виходів ОЛЛ $\alpha_g \in C^k$; S_L — кількість входів елементів LUT, які використовуються для реалізації схеми блока LUTerT.

Нехай ОЛЛ $\alpha_g \in C$ закодовані кодами $K(\alpha_g)$, що мають R_C розрядів. Ці коди представлені змінними $T_r \in T_C$. Закодуємо ОЛЛ $\alpha_g \in C^k$ кодами $C(\alpha_g)$, які ідентифікують ОЛЛ $\alpha_g \in C$ як елемент множини $C^k \subseteq C$. Будемо використо-

вувати змінні $\tau_r \in \tau^k$ для кодування ОЛЛ $\alpha_g \in C^k$. До того ж кількість таких змінних визначається формулою

$$R_k = \left\lceil \log_2 (|C^k| + 1) \right\rceil. \tag{11}$$

Умовимося, що нульовий код, який складається із змінних $\tau_r \in \tau^k$, відповідає відношенню $\alpha_g \in C^k$. Цим пояснюється наявність одиниці в (11).

Нехай перші R_1 елементів утворюють множину τ^1, наступні R_2 елементів — множину τ^2 і так далі. Тоді для кодування елементів усіх класів $C^k \in \Pi_C$ потрібно $R_0 = R_1 + R_2 + \ldots + R_K$ змінних і множина $\tau = \tau^1 \cup \tau^2 \cup \ldots \cup \tau^K$ має R_0 елементів.

Змінні $T_r \in T_C$ зберігаються в RG. Змінні $\tau_r \in \tau$ слід формувати на основі змінних $T_r \in T_C$. Для цього треба реалізувати на елементах LUT систему

$$\tau = \tau(T_C). \tag{12}$$

Кожен клас $C^k \in \Pi_0$ визначає блок LUTerk, який реалізує СБФ

$$\Phi^k = \Phi^k(\tau^k, X^k) \ \ (k \in \{1, \ldots, K\}). \tag{13}$$

Як випливає з (10), для реалізації будь-якої функції $D_r^k \in \Phi^k$ достатньо одного елемента LUT.

Блок LUTerk формує часткові функції D_r^k. Для формування адреси (7) необхідні функції $D_r \in \Phi$. Вони формуються як диз'юнкції часткових функцій

$$D_r = \bigvee_{k=1}^{K} D_r^k \ \ (r = \overline{1, R}). \tag{14}$$

Для формування функцій (14) необхідний блок LUTerOR.

Запропонований у цій статті КМПК U_3 (рис. 4) містить блоки LUTer1 ... LUTerK, LUTerOR і EMBer. Регістр RG і лічильник CT розподілені між елементами LUT блока LUTerOR.

Блок LUTerτ реалізує СБФ (12), функції якої використовуються для кодування ОЛЛ $\alpha_g \in C$. Блок LUTerOR формує коди $K(\alpha_g)$ і $K(b_q)$, що утворюють адреси МК (7). Блок EMBer зберігає мікрокоманди, які відповідають операторним вершинам початкової ГСА Г.

У кожний момент часу лише один блок LUTerk є «активним». На його виходах формуються функції (13). Решта блоків першого рівня перебувають у стані «очікування». На їхніх виходах формуються лише нулі. Це дає змогу зменшити споживану потужність у порівнянні з КМПК U_2.

За умови

$$K \le S_L \tag{15}$$

блок LUTerOR складається з $R_C + R_M$ елементів LUT. У цьому разі схема блока LUTerOR є однорівневою. Таким чином, за виконання умов (10) і (15) схема формування адреси МК у КМПК U_3 має тільки два рівні елементів LUT.

Далі пропонується метод синтезу КМПК U_3, який складається з таких кроків:

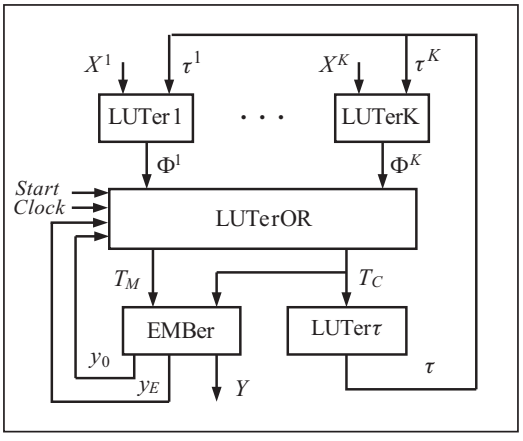

Рис. 4. Структурна схема КМПК U_3

1) формування множини ОЛЛ для ГСА Г;
2) кодування ОЛЛ кодами $K(\alpha_g)$;
3) кодування операторних вершин кодами $K(b_q)$;
4) формування розбиття Π_C множини $C(\Gamma)$;
5) формування таблиць блоків LUTer1 ... LUTerK;
6) формування СБФ (13), (14);
7) формування таблиці блока LUTerτ і СБФ (12);
8) формування таблиці блока EMBer;
9) реалізація схеми КМПК у заданому елементному базисі.

Позначимо $U_i(\Gamma_j)$ той факт, що модель U_i синтезується за ГСА Γ_j. Розглянемо приклад синтезу КМПК, якщо використовуються елементи LUT з $S_L = 5$.

ПРИКЛАД СИНТЕЗУ КМПК $U_3(\Gamma_1)$

Для формування множини C з мінімальною кількістю класів використовуємо метод [18]. Це дає змогу знайти множину $C = \{\alpha_1, \dots, \alpha_8\}$, де $\alpha_1 = \langle b_1, b_2 \rangle$, $\alpha_2 = \langle b_3, b_4, b_5 \rangle$, $\alpha_3 = \langle b_6, b_7 \rangle$, $\alpha_4 = \langle b_8, b_9, b_{10} \rangle$, $\alpha_5 = \langle b_{11}, b_{12}, b_{13} \rangle$, $\alpha_6 = \langle b_{14}, b_{15} \rangle$, $\alpha_7 = \langle b_{16} \rangle$, $\alpha_8 = \langle b_{17}, b_{18}, b_{19} \rangle$. Отже, маємо $G = 8$.

З виразу (5) отримуємо $R_C = 3$. Закодуємо ОЛЛ $\alpha_g \in C$ тривіальним способом: $K(\alpha_1) = 000, \dots, K(\alpha_8) = 111$.

Аналіз ОЛЛ $\alpha_g \in C$ дає змогу знайти $F_{\max} = 3$. Використовуючи (6), знаходимо $R_M = 2$, що дає множину $T_M = \{T_4, T_5\}$. Оскільки для адресації МК потрібно $\lceil \log_2 M \rceil = 5$ розрядів коду [11], у розглянутому прикладі не збільшується кількість входів КП у порівнянні з КМПК U_1. Це найкраща із можливих ситуацій [11].

Вочевидь, перший елемент кожного ОЛЛ повинен мати нульову адресу. Адресація виконується тривіальним способом. У табл. 1 наведено коди ОЛЛ $K(\alpha_g)$ і коди операторних вершин $K(b_q)$ для розглянутого прикладу. Конкатенація цих кодів дає адресу MI_m. Наприклад, для ОЛЛ $\alpha_3 \in C$ маємо адреси $A_6 = 01000$, $A_7 = 01001$.

Зазначимо, що вершина b_0 має нульову адресу. Тому вона потрапляє в ОЛЛ з кодом 000. Якщо введення b_0 в ОЛЛ збільшує параметр $F_{\max}$, доцільно ввести в множину C додаткову ОЛЛ $\alpha_0 = \langle b_0 \rangle$.

Крок 4 запропонованого методу значною мірою визначає якість схеми адресації МК: кількість блоків LUTerk, рівнів у схемі LUTerOR, а також міжз'єднань [18]. Для розв'язання цього завдання використовується метод [18]. Розбиття Π_C шукають для множини $C(\Gamma)$. У цьому разі ОЛЛ $\alpha_7, \alpha_8 \notin C(\Gamma_1)$.

Застосування методу [18] дає змогу отримати розбиття $\Pi_C = \{C^1, C^2\}$ з класами $C^1 = \{\alpha_1, \alpha_2, \alpha_3\}$ і $C^2 = \{\alpha_4, \alpha_5, \alpha_6\}$. З ГСА Γ_1 можна отримати множини $X^1 = \{x_1, x_2, x_3\}$ і $X^2 = \{x_4, x_5, x_6\}$. Отже, $L_1 = L_2 = 3$. Використовуючи (11), от-

Таблиця 1. Адреси мікрокоманд КМПК U_3 (Γ_1)

α_g	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8
$K(\alpha_g)$ / $K(b_q)$	000	001	010	011	100	101	110	111
00	b_0	b_3	b_6	b_8	b_{11}	b_{14}	b_{16}	b_{17}
01	b_1	b_4	b_7	b_9	b_{12}	b_{15}	—	b_{18}
10	b_2	b_5	—	b_{10}	b_{13}	—	—	b_{19}
11	—	—	—	—	—	—	—	—

римуємо $R_1 = R_2 = 2$, що визначає множини $\tau^1 = \{\tau_1, \tau_2\}$, $\tau^2 = \{\tau_3, \tau_4\}$, $\tau = \{\tau_1, ..., \tau_4\}$ і $R_0 = 4$. Таким чином, умова (10) виконується для кожного класу Π_C множини $C(\Gamma_1)$.

Використаємо коди 00 для співвідношень $\alpha_g \notin C^1$ $(\tau_1 = \tau_2 = 0)$ і $\alpha_g \notin C^2$ $(\tau_3 = \tau_4 = 0)$. Вочевидь, коди $C(\alpha_g)$ не впливають на кількість елементів LUT у схемах блоків LUTer1, LUTer2. Тож ОЛЛ $\alpha_g \in C^k$ можна закодувати довільно. Нехай ОЛЛ мають такі коди: $C(\alpha_3) = C(\alpha_6) = 11$, $C(\alpha_2) = C(\alpha_5) = 10$, $C(\alpha_3) = C(\alpha_6) = 11$.

Таблиця блока LUTerK має стовпчики $\alpha_g, C(\alpha_g), b_q, A(b_q), X_h^k, \Phi_h^k, h$. У стовпчику X_h^k записується кон'юнкція змінних $x_l \in X^k$, що визначає перехід з виходу ОЛЛ $\alpha_g \in C^k$ у вершину $b_q \in B$. У стовпчику Φ_h^k записуються функції $D_r \in \Phi^k$, що формують у RG і ЛА адресу МК MI_q, яка відповідає вершині $b_q \in B$.

Для формування таблиць блоків LUTer1 ... LUTerK необхідно знайти системи формул переходів (СФП) [20] для виходів ОЛЛ $\alpha_g \in C^k$. У розглянутому прикладі, використовуючи ГСА Γ_1, можна отримати такі СФП:

$$\alpha_1 \to x_1 b_3 \vee \overline{x_1} x_2 b_5 \vee \overline{x_1}\,\overline{x_2} b_6; \tag{16}$$

$$\alpha_2 \to x_1 b_{11} \vee \overline{x_1} x_3 b_{14} \vee \overline{x_1}\,\overline{x_3} b_9;\ \alpha_3 \to x_3 b_8 \vee \overline{x_3} b_6;$$

$$\alpha_4 \to x_4 b_{18} \vee \overline{x_4} x_6 b_{19} \vee \overline{x_4}\,\overline{x_6} b_9; \tag{17}$$

$$\alpha_5 \to x_4 b_{12} \vee \overline{x_4} b_{16};\ \alpha_6 \to x_5 b_{17} \vee \overline{x_5} b_{18}.$$

Система (16) дає змогу побудувати таблицю блока LUTer1 (табл. 2), а система (17) — таблицю блока LUTer2 (табл. 3). Адреси мікрокоманд беруть з табл. 1. Зв'язок між системами (16), (17) і таблицями блоків LUTer1, LUTer2 є очевидним.

Для синтезу схем блоків LUTer1, LUTer2 необхідно знайти системи функцій (13). З табл. 2 маємо таку СБФ (з урахуванням мінімізації):

$$D_1^1 = \tau_1 \overline{\tau_2} x_1 \vee \tau_1 \overline{\tau_2} x_3;\ D_2^1 = \overline{\tau_1} \tau_2 \overline{x_1}\,\overline{x_2} \vee \tau_1 \overline{\tau_2}\,\overline{x_1}\,\overline{x_3} \vee \tau_1 \tau_2;$$

$$D_3^1 = \overline{\tau_1} \tau_2 x_1 \vee \tau_1 \overline{\tau_2}\,\overline{x_1} \vee \tau_1 \tau_2 x_3; \tag{18}$$

$$D_4^1 = \overline{\tau_1} \tau_2 \overline{x_1} x_2;\ D_5^1 = \tau_1 \overline{\tau_2}\,\overline{x_1}\,\overline{x_3}.$$

Таблиця 2. Таблиця блока LUTer1 КМПК $U_3(\Gamma_1)$

α_g	$C(\alpha_g)$	b_q	$A(b_q)$	X_h^1	Φ_h^1	h
α_1	01	b_3	00100	x_1	D_3^1	1
		b_5	00110	$\overline{x_1} x_2$	$D_3^1 D_4^1$	2
		b_6	01000	$\overline{x_1}\,\overline{x_2}$	D_2^1	3
α_2	10	b_{11}	10000	x_1	D_1^1	4
		b_{14}	10100	$\overline{x_1} x_3$	$D_1^1 D_3^1$	5
		b_9	01101	$\overline{x_1}\,\overline{x_3}$	$D_2^1 D_3^1 D_5^1$	6
α_3	11	b_8	01100	x_3	$D_2^1 D_3^1$	7
		b_6	01000	$\overline{x_3}$	D_2^1	8

Таблиця 3. Таблиця блока LUTer2 КМПК $U_3(\Gamma_1)$

α_g	$C(\alpha_g)$	b_q	$A(b_q)$	X_h^2	Φ_h^2	h
α_4	01	b_{18}	11101	x_4	$D_1^2D_2^2D_3^2D_5^2$	1
		b_{19}	11110	$\overline{x_4}x_6$	$D_1^2D_2^2D_3^2D_4^2$	2
		b_9	01101	$\overline{x_4}\,\overline{x_6}$	$D_2^2D_3^2D_5^2$	3
α_5	10	b_{12}	10010	x_4	$D_1^2D_4^2$	4
		b_{16}	11000	$\overline{x_4}$	$D_1^2D_2^2$	5
α_6	11	b_{17}	11100	x_5	$D_1^2D_3^2D_3^2$	6
		b_{18}	11101	$\overline{x_5}$	$D_1^2D_2^2D_3^2D_5^2$	7

З табл. 3 з урахуванням мінімізації маємо СБФ

$$D_1^2=\overline{\tau_3}\tau_4x_4\vee\overline{\tau_3}\tau_4x_6\vee\tau_3;\ D_2^2=\tau_4\vee\tau_3\overline{\tau_4}\,\overline{x_4};\ D_3^2=\tau_4;$$
$$D_4^2=\overline{\tau_3}\tau_4\overline{x_4}x_6\vee\tau_3\overline{\tau_4}x_4;\ D_5^2=\overline{\tau_3}\tau_4x_4\vee\overline{\tau_3}\tau_4\overline{x_6}\vee\tau_3\tau_4\overline{x_5}. \qquad (19)$$

Для синтезу блока LUTerOR необхідно знайти диз'юнкції однойменних функцій з СБФ (18), (19). У разі КМПК $U_3(\Gamma_1)$ блок LUTerOR представляється СБФ $D_r=D_r^1\vee D_r^2$ $(r\in\{1,\dots,5\})$.

Таблиця блока LUTerτ має такі стовпчики: $\alpha_g, K(\alpha_g), C(\alpha_g), \tau_g, g$. У стовпчику τ_g записуються змінні $\tau_r\in\tau$, рівні одиниці в коді $C(\alpha_g)$. У розглянутому прикладі блок LUTerτ представлено табл. 4.

З табл. 4 формується СБФ

$$\tau_1=\overline{T_1}\,\overline{T_2}T_3\vee\overline{T_1}T_2\overline{T_3};\ \tau_2=\overline{T_1}\,\overline{T_3};\ \tau_3=T_1\overline{T_2};\ \tau_4=\overline{T_1}T_2T_3\vee\overline{T_1}\,\overline{T_2}\,\overline{T_3}. \qquad (20)$$

Наприклад, змінна τ_2 записана в рядках 1 і 3. Склеюючи кон'юнкції $\overline{T_1}\,\overline{T_2}\,\overline{T_3}$ і $\overline{T_1}T_2T_3$, отримуємо формулу для τ_2. Аналогічно отримаємо інші формули. Отже, СБФ (20) є системою (12) для розглянутого прикладу.

Блок EMBer представляється таблицею, що має такі стовпчики: MI_m, A_m, y_0, y_E, Y, m. Для мікрокоманд, які не відповідають виходам ОЛЛ, змінна $y_0=1$. Для мікрокоманд, що відповідають операторним вершинам, виходи яких з'єднано з входом фінальної вершини b_E ГСА Г, змінна $y_E=1$. Для розглянутого прикладу блок EMBer представлено табл. 5

Оскільки $R_C+R_M=5$, табл. 5 повинна мати 32 рядки. Однак рядки 20–32 цієї таблиці будуть містити лише нулі. Тому в табл. 5 показано лише 19 рядків, які містять одиниці в деяких стовпчиках.

Таблиця 4. Таблиця блока LUTerτ КМПК $U_3(\Gamma_1)$

α_g	$K(\alpha_g)$	$C(\alpha_g)$	τ_g	g
α_1	000	0100	τ_2	1
α_2	001	1000	τ_1	2
α_3	010	1100	$\tau_1\tau_2$	3
α_4	011	0001	τ_4	4
α_5	100	0010	τ_3	5
α_6	101	0011	$\tau_3\tau_4$	6

Таблиця 5. Таблиця блока EMBer КМПК $U_3(\Gamma_1)$

MI_m	A_m	y_0	y_E	Y									m
				y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	
MI_1	00000	1	0	1	1	0	0	0	0	0	0	0	1
MI_2	00001	0	0	0	0	1	0	0	0	0	0	0	2
MI_3	00100	1	0	0	1	0	1	0	0	0	0	0	3
MI_4	00101	1	0	1	0	0	0	1	0	0	0	0	4
MI_5	00110	0	0	0	0	0	0	0	1	0	0	0	5
MI_6	01000	1	0	1	1	0	0	0	0	0	0	0	6
MI_7	01001	0	0	0	0	0	1	0	1	0	0	0	7
MI_8	01100	1	0	0	0	1	0	1	0	0	0	0	8
MI_9	01101	1	0	1	1	0	0	0	0	0	0	0	9
MI_{10}	01110	0	0	0	0	0	1	0	1	0	0	0	10
MI_{11}	10000	1	0	0	0	1	0	0	0	0	0	0	11
MI_{12}	10001	1	0	1	1	0	0	0	0	0	0	0	12
MI_{13}	10010	0	0	0	0	0	0	0	1	0	0	0	13
MI_{14}	10100	1	0	0	0	0	0	1	1	0	0	0	14
MI_{15}	10101	0	0	0	0	0	1	0	0	0	0	0	15
MI_{16}	10101	0	1	0	0	1	0	1	0	1	0	0	16
MI_{17}	11000	1	0	0	0	0	0	0	0	0	1	0	17
MI_{18}	11101	1	0	1	0	0	0	0	0	0	0	1	18
MI_{19}	11110	0	1	0	0	1	0	0	0	0	0	0	19

Останній крок запропонованого методу пов'язаний із застосуванням стандартних САПР типу Vivado [23] або Quartus [24]. На цьому кроці розв'язуються задачі розміщення і трасування, формуються таблиці елементів LUT і блоків EMB, формуються бітові потоки для програмування міжз'єднань і елементів [2, 5, 6]. Ці питання детально вивчені, наприклад, у [2] і виходять за межі цієї роботи.

АНАЛІЗ ЗАПРОПОНОВАНОГО МЕТОДУ

Виконаємо аналіз розглянутого прикладу і знайдемо кількість елементів LUT у схемі КМПК $U_3(\Gamma_1)$ для LUT з $S_L = 5$.

Для блоків LUTer1, LUTer2 кількість елементів LUT визначається кількістю рівнянь у системах (18) і (19). Кожна з цих систем має $R_C + R_M = 5$ рівнянь, для яких виконується умова (10). Рівняння для функції D_3^2 складається з однієї літери. Тож ця функція генерується на виході тригера. Отже, блоки LUTer1 і LUTer2 мають дев'ять елементів LUT.

Кількість елементів LUT можна зменшити, якщо для деякої функції $D_r^k \in \Phi^k$ виконується умова

$$D_r^k = 0 \ \ (r = \overline{1, R_C + R_M};\ k = \overline{1, K}). \tag{21}$$

Для виконання (21) необхідно відповідним чином закодувати ОЛЛ $\alpha_g \in C$.

Наприклад, для деякого блока LUTerk є переходи на входи в ОЛЛ $\alpha_3, \alpha_4, \alpha_5$. Якщо $R_C = 3$, ОЛЛ потрібно закодувати таким чином: $K(\alpha_3) = 001$, $K(\alpha_4) = 010$, $K(\alpha_5) = 011$. Вочевидь, $T_1 = 0$ для всіх кодів, що відповідає виконанню умови (21) для функції D_1^k. Алгоритм такого кодування ОЛЛ заплановано розробити в подальших роботах.

Для КМПК $U_3(\Gamma_1)$ маємо $K = 2$. Умова (15) виконується і блок LUTerOR складається з п'яти елементів LUT. Зазначимо, що за виконання умови (21) кількість міжз'єднань блоків LUTer1 ... LUTerK і LUTerOR зменшується. Це відбувається відносно максимально можливої кількості міжз'єднань, що визначається добутком K і $R_C + R_M$.

Для розглянутого прикладу виконується умова

$$R_C \le S_L. \tag{22}$$

Отже, блок LUTerτ має $R_0 = 4$ елементи LUT. Як випливає з рівнянь (20), функції τ_2 і τ_3 залежать від двох аргументів. Тож елементи LUT для функцій τ_1 і τ_4 мають три міжз'єднання, а для функцій τ_2 і τ_3 — два.

За виконання умови (22) максимальна кількість міжз'єднань визначається добутком $R_C \times R_0$. У наведеному прикладі $R_C \times R_0 = 12$, а з (20) випливає, що існує 10 міжз'єднань.

Як відомо, кількість міжз'єднань впливає на швидкодію і споживану потужність схеми [21]. Таким чином, кодування ОЛЛ треба виконувати так, щоб зменшити кількість міжз'єднань.

Обчислення показує, що для реалізації схеми КМПК $U_3(\Gamma_1)$ потрібно 18 елементів LUT з $S_L = 5$. Кількість блоків EMB визначається виразом (4).

У кращому разі блок адресації мікрокоманд має два рівні елементів LUT. Аналіз бібліотеки стандартних прикладів [22] показав, що для мікросхем сім'ї Virtex-7 [25] схема адресації має два рівні логіки. Елементи LUT Virtex-7 мають $S_L = 6$. Цього достатньо, щоб блок LUTerτ мав тільки один рівень логіки.

Блоки EMB мікросхем Virtex-7 мають такі параметри, що для реалізації блока EMBer пристрою U_3 достатньо одного блока EMB. Зазначимо, що майже для всіх прикладів [22] виконувалася умова

$$t_F \ge R_0 + N + 2. \tag{23}$$

За виконання умови (23) блок LUTerτ може бути вилучений з КМПК. Система (12) реалізується блоком EMBer, що визначає схему КМПК U_4 (рис. 5).

Методи синтезу КМПК U_3 і U_4 збігаються, але для КМПК U_4 не потрібен етап формування таблиці блока LUTerτ. Таблиця блока EMBer КМПК U_4 містить стовпчик τ з кодами $C(\alpha_g)$. У цьому разі немає впливу кодів $C(\alpha_g)$ на кількість елементів LUT у блоці LUTerτ.

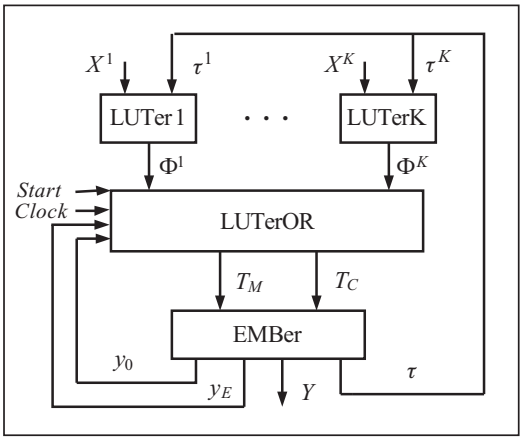

Рис. 5. Структурна схема КМПК U_4

Зазначимо, що КМПК U_3 базується на існуванні розбиття Π_C, для якого виконується умова (10). Остання виконується для всіх стандартних прикладів [22] для $S_L = 6$. Якщо умова (10) порушується, тоді запропонований у роботі метод не можна застосува-

ти. У цьому разі схема блока адресації мікрокоманд реалізується з використанням методів функціональної декомпозиції [16].

Якщо умови (10) і (22) виконуються, кожен блок LUTer в КМПК має лише один рівень логіки. Це дозволяє отримати схеми КМПК U_3 з швидкодією кращою, ніж для рівнозначного КМПК U_2.

ВИСНОВКИ

У разі реалізації схем КМПК у базисі FPGA доцільно використовувати елементи LUT і блоки пам'яті EMB. Елементи LUT реалізують схему адресації мікрокоманд, а блоки EMB — керувальну пам'ять, що зберігає мікрокоманди.

У цій роботі запропоновано метод зменшення кількості елементів LUT у схемі адресації мікрокоманд КМПК, який ґрунтується на ідеї подвійного кодування станів мікропрограмних автоматів [18] для оптимізації схеми КМПК з розділенням кодів.

Наведений метод доцільно використовувати, якщо схема адресації МК має більше двох рівнів логіки. За виконання умови (10) метод гарантує отримання схеми адресації МК з двома рівнями логіки. До того ж існує можливість зменшення кількості з'єднань між блоками схеми. Аналіз стандартних автоматів бібліотеки [22] і базису Virtex-7 [25] показав, що запропонований підхід можна застосовувати для 100 % тестових прикладів.

Для реалізації схеми КМПК необхідно мати два коди для кожної ОЛЛ. Для переходу від коду ОЛЛ як елемента множини до коду ОЛЛ як елемента деякого класу розбиття потрібно мати блок перетворення кодів. Однак перетворення виконується паралельно з формуванням мікрооперацій і не впливає на швидкодію КМПК. За виконання умови (23) перетворювач кодів доцільно реалізувати як частину керувальної пам'яті.

Для складних алгоритмів керування можлива ситуація, коли не існує розбиття Π_C, що задовольняє умову (10). У цьому разі модель U_3 не може бути використана. У подальших дослідженнях планується розробити метод синтезу КМПК для такого поєднання ГСА та FPGA. Крім того, планується застосувати цей підхід для синтезу нейроподібних мереж [28].

СПИСОК ЛІТЕРАТУРИ

1. Kubica M., Kania D. Technology mapping oriented to adaptive logic modules. *Bulletin of Polish Academy of Sciences*. 2019. Vol. 67, N 5. P. 947–956.
2. Sklyarov V., Skliarova I., Barkalov A., Titarenko L. Synthesis and optimization of FPGA-based systems. Berlin: Springer, 2014. 432 p.
3. Соловьев В.В. Проектирование цифровых схем на основе программируемых логических интегральных схем. Москва: Горячая линия — ТЕЛЕКОМ, 2001. 636 с.
4. Rawski M., Tomaszewicz P., Borowski G., Luba T. Logic synthesis method of digital circuits designed for implementation with embedded memory blocks on FPGAs. *Design of Digital Systems and Devises. Lecture Notes in Electrical Engineering.* Adamski M., Barkalov A., Wegrzyn M. (Eds.). Vol. 79. Berlin: Springer, 2011. P. 121–144.
5. Maxfield C. The design warrior's guide to FPGAs. Orlando: Academic Press, 2004. 542 p.
6. Grout I. Digital systems design with FPGAs and CPLDs. Amsterdam: Elsevier, 2008. 784 p.
7. Rawski M., Selvaraj H., Luba T. An application of functional decomposition in ROM-based FSM implementation in FPGA devices. *Journal of System Architecture.* 2005. Vol. 51, Iss. 6–7. P. 424–434.
8. Czerwinski R., Kania D. Finite state machines logic synthesis for complex programmable logic devices. Berlin: Springer, 2013. 172 p.
9. Mishchenko A., Chattarejee S., Bryton R. Improvements to technology mapping for LUT-based FPGAs. *IEEE Transactions on CAD*. 2006. Vol. 27, N 2. P. 240–253.

10. Kilts S. Advanced FPGA design: Architecture, implementation and optimization. Willey-IEEE Press, 2007.
11. Баркалов А.А., Титаренко Л.А. Синтез композиционных микропрограммных устройств управления. Харьков: Коллегиум, 2007. 304 с.
12. Баркалов А.А., Титаренко Л.А., Ефименко К.Н. Оптимизация схем композиционных микропрограммных устройств управления, реализуемых на ПЛИС. *Кибернетика и системный анализ*. 2011. № 1. С. 179–188.
13. Баркалов А.А., Титаренко Л.А. Преобразование кодов в композиционных микропрограммных устройствах управления. *Кибернетика и системный анализ.* 2011. № 5. С. 107–118.
14. White paper FPGA architecture. URL: www.altera.com.
15. Tiwari A., Tomko K. Saving power by mapping finite state machines into embedded memory blocks in FPGAs. *Proc. Design, Automation and Test in Europe Conference and Exhibition* (Paris, France, 6–20 Feb. 2004). 2004. Vol. 2. P. 916–921.
16. Scholl C. Functional decomposition with application to FPGA synthesis. Kluwer Academic Publishers, Boston, 2001.
17. Nowicka M., Luba T., Rawski M. FPGA-based decomposition of Boolean functions: Algorithms and implementations. *Proc. of the 6th International Conference on Advanced Computer Systems* (Szczecin, 1999). P. 502–509.
18. Barkalov A., Titarenko L., Mielcatek K. Hardware reduction for LUT-based Mealy FSMs. *International Journal of Applied Mathematics and Computer Science.* 2018. Vol. 28, N 3. P. 595–607.
19. Baranov S. Logic synthesis for control automata. Dordrecht: Kluwer Academic Publishers, 1994. 312 p.
20. Barkalov A., Titarenko L. Logic synthesis for FSM-based control units. Berlin: Springer, 2009. 233 p.
21. Kuon I., Tessier R., Rose J. FPGA Architecture: Survey and challenges — found trends. *Electrical Design Automation.* 2008. N 2. P. 135–253.
22. Yang S. Logic synthesis and optimization benchmarks user guide. Version 3.0. Techn. Rep. Microelectronics Center of North Carolina, 1991. 43 p.
23. Vivado Design Suite. URL: https://www.xilinx.com/products/design-tools/vivado.html.
24. Intel® Quartus® Prime Software Suite. URL: https://www.intel.com/content/www/us/en/software/programmable/quartus-prime/overview.html.
25. Virtex-7 FPGAs. URL: https://www.xilinx.com/products/silicon-devices/fpga/virtex-7.html.
26. Barkalov A., Titarenko L., Chmielewski S. Mixed encoding of collections of output variables for LUT-based FSMs. *Journal of Circuits, Systems and Computers*. 2019. Vol. 28, N 8. P. 1–21.
27. Garcia-Vargas L., Senhaji-Navarro R. Finite state machines with input multiplexing: A performance study. *IEEE Transactions on CAD of Integrated Circuits and Systems.* 2015. Vol. 34, Iss. 5. P. 867–871.
28. Opanasenko V.N., Kryvyi S.L. Synthesis of neural-like networks on the basis of conversion of cyclic Hamming codes. *Cybernetics and Systems Analysis*. 2017. Vol. 53, N 4. P. 627–635. https://doi.org/10.1007/s10559-017-9965-z.

A.A. Barkalov, L.A. Titarenko, A.V. Baiev, A.V. Matviienko

OPTIMIZATION OF CMCU WITH CODE SHARING

Abstract. The article proposes a method for reducing the number of LUT elements in the circuit of a compositional microprogram control unit (CMCU) with code sharing. The method is based on two-fold encoding of operator linear chains (OLC). Each chain has a code as an element of the OLC set and as a class element of this set. This approach allows obtaining a two-level microinstruction addressing circuit. The control memory of the CMCU is implemented on the embedded memory blocks. The article considers an example of synthesis and provides an analysis of the proposed method.

Keywords: compositional microprogram control unit, LUT, EMB, code sharing.

Надійшла до редакції 01.02.2021