https://doi.org/10.15407/intechsys.2026.03.015
УДК 004.274

О.О. БАРКАЛОВ, д-р. техн. наук, професор,
Зеленогурський Університет,
вул. Ліцеальна, 9, Зелена Гура, Польща
https://orcid.org/ orcid.org/0000-0002-4941-3979
A.Barkalov@iie.uz.zgora.pl
Л.О. ТІТАРЕНКО, д-р. техн. наук, професор,
Зеленогурський Університет,
вул. Ліцеальна, 9, Зелена Гура, Польща
Харківський національний університет радіоелектроніки,
пр. Науки, 14, м. Харків, 61166, Україна
https://orcid.org/0000-0001-9558-3322
L.Titarenko@iie.uz.zgora.pl
О.В. МАТВІЄНКО, наук. співроб.,
Інститут кібернетики імені В.М. Глушкова НАН України,
просп. Акад. Глушкова, 40, Київ, 03187, Україна
https://orcid.org/0000-0003-1838-1422
avmatv@ukr.net

СИНТЕЗ КОМПОЗИЦІЙНОГО МІКРОПРОГРАМНОГО ПРИСТРОЮ КЕРУВАННЯ З ДВОМА ЯДРАМИ ЧАСТКОВИХ ФУНКЦІЙ

Запропоновано метод структурної декомпозиції схеми композиційного мікропрограмного пристрою управління (КМПУ) у базисі FPGA. Метод ґрунтується на виділенні двох ядер часткових функцій. Одне з ядер генерує функції, у яких число аргументів перевищує кількість входів елементів LUT. Друге ядро генерує функції, що базуються на кодуванні операторних лінійних ланцюгів. Запропоновано архітектуру КМПУ з двома ядрами часткових функцій та метод синтезу схеми КМПУ. Розглянуто приклад синтезу схеми КМПУ із двома ядрами. Розглянуто умови застосування цього підходу. Показано можливі шляхи вирішення оптимізаційних завдань, пов'язаних з дисбалансом характеристик алгоритму управління та елементів LUT.

Ключові слова: *часткова функція, ядро, КМПУ, LUT, EMB, синтез, структурна декомпозиція.*

Цитування: Баркалов О.О., Тітаренко Л.О., Матвієнко О.В. Синтез композиційного мікропрограмного пристрою керування з двома ядрами часткових функцій. *Information Technologies and Systems*, Київ, 2026, Том 3 (9), 15–29. https://doi.org/10.15407/intechsys.2026.03.015

Вступ

Композиційні мікропрограмні пристрої управління [1, 2] є ефективним засобом реалізації лінійних алгоритмів керування [2]. У процесі проєктування КМПУ постає проблема оптимізації характеристик його схеми. До таких характеристик відносяться апаратурні витрати, швидкодія та споживана потужність [3]. Методи вирішення проблеми залежать від особливостей елементного базису. У цій роботі розглядається реалізація схеми КМПУ в базисі *FPGA* (*field-programmablelogicalarray*) [4].

Вибір базису *FPGA* обумовлений широким його застосуванням під час реалізації цифрових систем [5]. На думку експертів [6] *FPGA* будуть популярними ще кілька десятиліть. Цим обумовлена актуальність розробки схем у базисі *FPGA*.

Пропонований метод може бути застосовано за необхідності функціональної декомпозиції [7, 8] схеми адресації мікрокоманд (МК). Ми розглядаємо ситуацію, коли частина схеми заснована на функціональній, а друга частина — на структурній декомпозиції [9]. Будь-яка декомпозиція цифрового пристрою пов'язана з формуванням систем часткових функцій (СЧФ) [10]. Ми пропонуємо використовувати два ядра СЧФ.

У статті розглядається КМПУ із загальною пам'яттю [1]. Для оптимізації схеми використовують метод перетворення адрес виходів операторних лінійних ланцюгів (ОЛЛ) [11] Алгоритм управління подано як граф схема алгоритму (ГСА) [12].

Особливості КМПУ та базису *FPGA*

Синтез КМПУ починається з формування ОЛЛ, що утворюють множину $C = \{\alpha_1, \ldots, \alpha_G\}$ [1]. Кожна ОЛЛ є вектором, який складається з операторних вершин ГСА Г. У кожній ОЛЛ $\alpha_g \in \tilde{N}$ вершини розташовуються у тому ж порядку, що і в ГСА Г. У ГСА Г виділяються множина операторних вершин $B = \{b_1, \ldots, b_M\}$ і множина умовних вершин, що складається з *NC* елементів. ГСА є лінійною в разі виконання умови [2]

$$M \geq 0{,}75(M + NC). \tag{1}$$

Кожна вершина $b_m \in B$ відповідає мікрокоманді Y_m. Мікрокоманди Y_m зберігаються у керувальній пам'яті (КП) КМПУ. Розрядність адрес мікрокоманд визначається як

$$R = \lceil \log_2 M \rceil. \tag{2}$$

Кожна умовна вершина містить одну логічну умову $x_l \in X$, де $X = \{x_1, \ldots, x_L\}$. В цьому разі може виконуватися умова $L < NC$. Кожна ОЛЛ має лише один вихід o_g (остання вершина в ОЛЛ). МК Y_m має адресу A_m розрядності R.

Адресація МК здійснюється наступним чином. Нехай ОЛЛ $\alpha_g \in \tilde{N}$ охоплює пару вершин $< b_i, b_j >$, де вихід вершини b_i пов'язаний з входом вершини b_j. У цьому разі маємо залежність між адресами A_i та A_j [2]:

$$A_j = A_i + 1. \qquad (3)$$

Як випливає з (3), для зберігання адрес необхідно використовувати лічильник адреси (ЛА). Виходи ЛА утворюють множину адресних змінних $T = \{T_1, \ldots, T_R\}$.

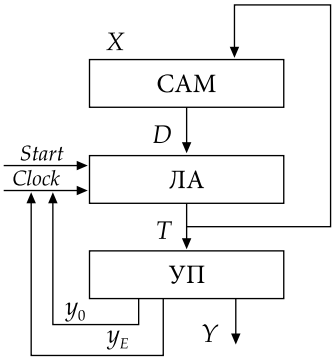

Рис. 1. Структурна схема КМПУ U_1

Будемо використовувати ЛА з інформаційними входами типу *D*. Для зміни адреси в ЛА, відмінної від (3), необхідно використовувати функції збудження пам'яті (ФЗП), що утворюють множину $D = \{D_1, \ldots, D_R\}$. При цьому адреса переходу, що відмінна від (3), визначається схемою адресації мікрокоманд (САМ). Ця схема реалізується з урахуванням СБФ:

$$D = D(T, X). \qquad (4)$$

Таким чином, у КМПУ використовуються два режими адресації. Для їх вибору використовується додаткова функція $y_O \notin Y$, де $Y = \{y_1, \ldots, y_N\}$ — множина мікрооперацій. Нехай $y_O = 1$ визначає режим (3), а $y_O = 0$ — режим (4).

Процес функціонування КМПУ починається під час надходження імпульсу *Start*. Зміна вмісту КП дозволяється за певним фронтом імпульсу синхронізації *Clock*. Функціонування завершується блокуванням сигналу *Clock* змінною y_E. На основі цієї інформації можна одержати КМПУ U_1 (рис. 1).

КМПУ U_1 отримав назву КМПУ із загальною пам'яттю [1, 2]. Схема функціонує в такий спосіб. За сигналом *Start* у ЛА записується початкова адреса мікропрограми. Схема САМ генерує СБФ (4). Адреси записуються до ЛА. Чергова мікрокоманда вибирається із КП. Сигнали y_o та y_E визначають режими адресації та зупинки.

Лічильник є стандартним пристроєм, і його реалізація не викликає труднощів. Схема КП будується на стандартних блоках пам'яті. Проблеми виникають під час реалізації блоку САМ. Ці проблеми пов'язані з тим, що САМ не є стандартним блоком. Його схема залежить від системи (4). У статті пропонується метод вирішення цієї проблеми за використання внутрішніх ресурсів мікросхем *FPGA*.

Блок САМ із лічильником утворюють послідовну схему [5]. Для її реалізації необхідні такі ресурси мікросхеми *FPGA*:

- елементи табличного типу *LUT* (*look-uptable*), що мають S_L входів та один вихід;
- програмовані синхронні тригери із входом скидання у 0;
- програмовані міжз'єднання.

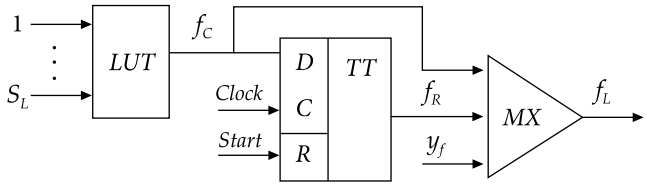

Рис. 2. Спрощена схема логічного елемента

Елементи *LUT* та тригера утворюють логічні елементи (ЛЕ). Декілька ЛЕ утворюють блок *CLB* (*ConfigurableLogicBlock*). Ускладі *CLB* є система швидких міжз'єднань [13]. З'єднання між *CLB* є повільнішими, ніж усередині *CLB*. На рис. 2 наведено схему ЛЕ.

ЛЕ має S_L входів і один вихід f_L. Елемент *LUT* зберігає таблицю істинності [14] довільної булевої функції, що має не більше S_L аргументів. Вихід *LUT* f_C пов'язаний із входом *D* тригера. За сигналом *Clock* комбінаційний сигнал f_C перетворюється на регістровий сигнал f_R. Вихід f_L може бути комбінаційним, або регістровим. Для вибору типу виходу використовується мультиплексор *MX* та сигнал y_f. Сигнал *Start* надходить на вхід скидання тригера у 0.

Таким чином, *CLB* може реалізовувати як комбінаційні схеми, так і схеми, що накопичують. Для реалізації швидких лічильників може бути використана внутрішня система прискорених переносів *CLB*.

Пам'ять КП реалізується за допомогою вбудованих блоків пам'яті *EMB* (*Embedded Memory Blocks*) [15, 16]. Ці блоки мають властивість реконфігурації [17]: число входів (S_A) та виходів (t_F) можна змінювати при збереженні постійної ємності V_0: $V_0 = 2^{S_A} t_F$.

Для реалізації пам'яті необхідно вибрати конфігурацію $< S_A, t_F >$, для якої

$$S_A = R. \tag{5}$$

Якщо такої конфігурації не існує, немає сенсу у використанні моделі КМПУ. При порушенні (5) КП повинна мати кілька рівнів блоків та дешифратор вибору рівня [16]. В результаті схема КП стає громіздкою, повільною та має значне енергоспоживання.

Якщо умова (5) виконується, схеми САМ і ЛА реалізуються на ЛЕ, а КП — на блоках *EMB*. Позначимо символом *LUTer* (*EMBer*) схему, що складається з кількох блоків *LUT* (*EMB*). Тоді отримаємо таку схему *F*-КМПУ U_1 (рис. 3).

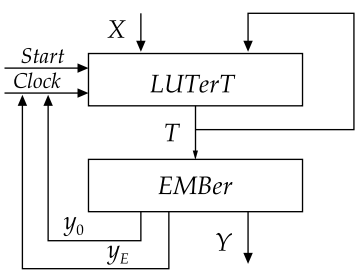

Рис. 3. Структурна схема *F*-КМПУ U_1

Позначання «*F*-КМПУ» підкреслює, що схема реалізована в базисі *FPGA*. Блок *LUTerT* містить блоки САМ і ЛА зі схеми на рис. 1

Кожна функція $D_r \in D$ подається диз'юнктивною нормальною формою (ДНФ), яка має NA_r аргументів. При

цьому L_r аргументів представляють логічні умови, а $NA_r \in L_r$ аргументів — адресні змінні $T_r \in T$. У статті розглядається випадок, коли для деяких ФЗП виконуються умови:

$$NA_r > S_L, \tag{6}$$

$$L_r \in S_L. \tag{7}$$

В цьому разі блок *LUTerT* має кілька рівнів елементів *LUT* та складну систему міжз'єднань. Такі схеми реалізуються за допомогою методів функціональної декомпозиції. Відомо, що ці методи призводять до схем з малою швидкодією та великим споживанням потужності. У цій статті пропонується метод покращення характеристик КМПУ в разі виконання умов (6), (7).

Ідея запропонованого методу

Побудуємо множину ОЛЛ $C = \{\alpha_1, \ldots, \alpha_G\}$ для деякої ГСА Г. Знайдемо множину виходів ОЛЛ $O(\Gamma)$. Визначимо число ЛУ L_g, що задають переходи з виходів ОЛЛ $\alpha_g \in \tilde{N}$. Якщо виконується умова:

$$L_g \in S_L, \tag{8}$$

вихід o_g розміщується у множину O_V, інакше — у множину O_T. Очевидно, що $O_V \cup O_T = O(\Gamma)$ і $O_V \cap O_T = \varnothing$. Закодуємо виходи $o_g \in O_V$ кодами $K(o_g)$. Число змінних, необхідних для кодування виходів, визначається виразом:

$$R_V = \lceil \log_2(|O_V| + 1) \rceil. \tag{9}$$

Для кодування виходів використовуються змінні $v_r \in V$, де $|V| = R_V$. У (9) одиниця додається для врахування співвідношення $o_g \notin O_V$.

Множина $O_V \subseteq O(\Gamma)$ визначає дві множини. Одна з них — множина X_V логічних умов, що задає переходи з виходів $o_g \notin O_V$. Друга — множина D_V, функцій збудження пам'яті, що формуються при переходах з виходів $o_g \notin O_V$. Клас O_V визначає блок *LUTerF*, схема якого задається СБФ:

$$D_V = D_V(V, X_V). \tag{10}$$

Множина O_T розбивається на класи, $O^k \in \Pi_O$, де $\Pi_O = \{O^1, \ldots, O^K\}$. Клас O^k складається з M_k виходів. Ці виходи кодуються частковими кодами $C(o_g)$, що мають R_k розрядів:

$$R_k = \lceil \log_2(M_k + 1) \rceil. \tag{11}$$

Код, що складається з R_k нулів, використовується для співвідношення $o_g \notin O^k$. Для кодування використовуються змінні $\tau_r \in \tau^k$, де $|\tau^k| = R_k$.

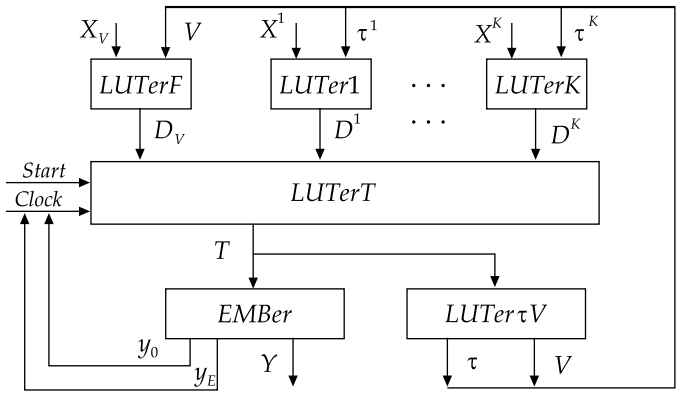

Рис. 4. Структурна схема *F*-КМПУ U_2

Кожен клас $O^k \in \Pi_O$ визначає множини D^k і X^k, які є аналогами множин D_V і X_V відповідно. Клас $O^k \in \Pi_O$ визначає блок *LUTerk*, що представляється СБФ:

$$D^k = D^k(\tau^k, X^k). \tag{12}$$

Остаточні значення ФЗП задаються СБФ:

$$D = D(D_V, D^1, D^2, ..., D^K). \tag{13}$$

Кожна функція $D_r \in D$ представлена системою диз'юнкцій $D_r = D_r^V \vee D_r^1 \vee D_r^2 \vee ... \vee D_r^K$.

Додаткові змінні $v_r \in V$ і $\tau_r \in \tau$ задаються наступними СБФ:

$$V = V(T); \tag{14}$$

$$\tau = \tau(T). \tag{15}$$

Система (13) визначає функціональний асемблер *LUTerT*. Функції (14) — (15) визначають блок перетворювача адрес МК *LUTerτV*. Мікрокоманди та додаткові функції y_0 та y_E зберігаються в КП, представленій блоком *EMBer*. Всі ці блоки визначають архітектуру *F*-КМПУ U_2 (рис. 4).

КМПУ U_2 функціонує у такій послідовності. Сигнал *Start* ініціює запис нульової адреси до ЛА. Ця адреса надходить до *EMBer*. Залежно від режиму адресації формується змінна $y_0 = 1$ або $y_0 = 0$. У першому випадку адреса визначається рівнянням (3). Для формування адреси переходу блок САМ не використовується. Якщо код виходу $y_0 = 0$, ОЛЛ формується блоком *LUTerτV*. В результаті один із блоків першого рівня логіки (*LUTerF, LUTer*1, …, *LUTerK*) формує часткові ФЗП, які надходять у блок *LUTerT* і перетворюються на функції $D_r \in D$. Ці функції надходять на входи тригерів лічильника всередині блоку *LUTerT*. За певним фронтом *Clock* функції $D_r \in D$ перетворюються на адресні змінні $T_r \in T$.

Функціонування продовжується до формування сигналу y_E. Цей сигнал блокує надходження сигналів синхронізації до ЛА. Після цього стан ЛА не змінюється (ФЗП не може змінити адресу).

У статті пропонується метод синтезу схеми *F*-КМПУ U_2. Метод містить такі етапи:

1. Формування множини ОЛЛ за ГСА Г.
2. Розбиття множини O(Г) на множини O_V і O_T.
3. Природна адресація МК у межах кожної ОЛЛ $\alpha_g \in \tilde{N}$.
4. Кодування виходів $o_g \in O_V$ частковими кодами $K(o_g)$.
5. Формування таблиці блоку *LUTerF* та СБФ (10).
6. Розбиття множини виходів O_T на класи $O^k \in \Pi_O$.
7. Кодування виходів $o_g \in O^k$ частковими кодами $C(o_g)$.
8. Формування таблиці блоків $LUTer1 - LUTerK$ і СБФ (12).
9. Формування таблиці блока *LUTerT* і СБФ (13).
10. Формування таблиці блока *LUTerτV* і СБФ (14) — (15).
11. Формування таблиці блока *EMBer*.
12. Реалізація схеми КМПУ U_2 з використанням внутрішніх ресурсів мікросхеми *FPGA*.

Далі наведено приклад синтезу КМПУ U_2. Ми не розглядаємо крок остаточної реалізації, оскільки він вимагає використання таких пакетів САПР, як *Vivado* [18] або *Quartus* [19].

Приклад синтезу схеми КМПУ

Нехай алгоритм управління представлений ГСА Γ_1 (рис.5). Нехай для реалізації схеми використовуються елементи *LUT* із числом входів S_L = 4 та блоки *EMB* із конфігурацією < 5, 8 >. Такий блок *EMB* може зберігати до 32 МК, що мають не більше восьми розрядів.

Як випливає з рис. 5, ГСА Γ_1 має M = 18 операторних вершин. Таким чином, маємо множину $B = \{b_1, ..., b_{18}\}$. Використовуючи (2), отримаємо R = 5. Умова (5) виконується. Аналіз ГСА Γ_1 дозволяє знайти множини $X = \{x_1, ..., x_4\}$ і $Y = \{y_1, ..., y_6\}$, при цьому N + 2 = 8. Отже достатньо одного блоку *EMB* для реалізації схеми КП. Оскільки R = 5, маємо множини $T = \{T_1, ..., T_5\}$ і $D = \{D_1, ..., D_5\}$.

Використовуючи методику з [1], побудуємо множину ОЛЛ $C = \{\alpha_1, ..., \alpha_7\}$, де $\alpha_1 = < b_1, b_2 >$, $\alpha_2 = < b_3, b_4, b_5 >$, $\alpha_3 = < b_6, b_7, b_8 >$, $\alpha_4 = < b_9, b_{10}, b_{11} >$, $\alpha_5 = < b_{12}, b_{13}, b_{14} >$, $\alpha_6 = < b_{15}, b_{16} >$, $\alpha_7 = < b_{17}, b_{18} >$. Оскільки G = 7, то множина $O(\Gamma_1) = \{o_1, ..., o_7\}$. Ці виходи відповідають наступним вершинам: $o_1 = b_2$, $o_2 = b_5$, $o_3 = b_8$, $o_4 = b_{11}$, $o_5 = b_{14}$, $o_6 = b_{16}$, $o_7 = b_{18}$.

Для синтезу схеми використовуються *LUT*з S_L = 4. Очевидно, що умова (8) виконується для виходу o_1. З цього аналізу отримуємо множини $O_V = \{o_1\}$ і $O_T = \{O_2, ..., O_7\}$.

Використовуючи методику з [1], знайдемо адреси МК. Для нашого прикладу маємо: $A(b_1) = 00000$, $A(b_2) = 00001$, ..., $A(b_{18}) = 10001$.

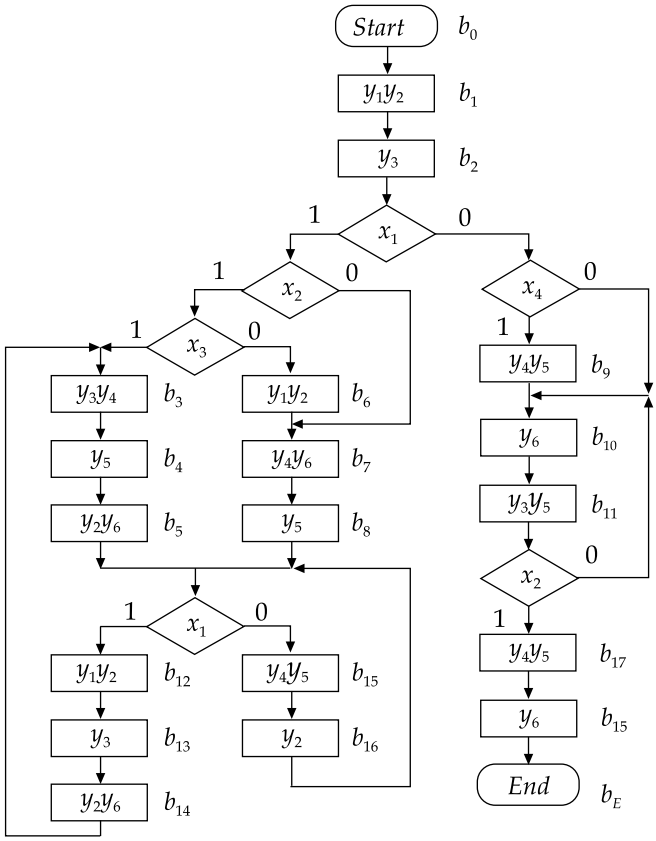

Рис. 5. Граф-схема алгоритма Γ_1

Вочевидь, умова (3) виконується для кожної пари сусідніх компонент всіх ОЛЛ $\alpha_g \in \tilde{N}$.

Оскільки $O_V = \{o_1\}$, то $M_V = |O_V| = 1$. З рівняння (9) отримаємо R_V=1, що дає $V = \{v_1\}$. Закодуємо вихід o_1 кодом $K(o_1) = 1$.

Побудуємо формулу переходу для виходу o_1 [12]:

$$b_2 \to x_1x_2x_3b_3 \vee x_1x_2\overline{x_3}b_6 \vee x_1\overline{x_2}b_7 \vee \overline{x_1}x_4b_9 \vee \overline{x_1}\,\overline{x_4}b_{10}\,. \qquad (16)$$

Ця формула визначає таблицю блоку *LUTerF*. Таблиця має стовпці o_g, $K(o_g)$, b_m, $A(b_m)$, X_h, D_h, h. Стовпець b_m містить вершини формули переходу, стовпець $A(b_m)$, складається з адрес МК. Стовпець X_h включає кон'юнкції логічних умов, що визначають переходи. Стовпець D_h складається з ФЗП, що дорівнюють одиниці для запису в ЛА адреси $A(b_m)$. Для прикладу блок *LUTerF* поданий табл.1.

Число рядків у табл.1 дорівнює числу термів у формулі (16). Принцип формування інформації в стовпцях b_m — D_h є очевидним.

Таблиця блоку *LUTerF* є основою формування СБФ (10). Щоб підкреслити, що ці часткові ФЗП формуються блоком *LUTerF*, ми використовуємо верхній індекс «0». Для нашого прикладу з табл. 1

отримані такі функції:

$$D_1^0 = 0; \quad D_2^0 = v_1\overline{x_1}; \quad D_3^0 = v_1 x_1;$$
$$D_4^0 = v_1 x_1 \overline{x_2} \vee v_1 \overline{x_1 x_4}; \quad D_5^0 = v_1 x_1 x_2 \overline{x_3}.$$

Для пошуку розбиття Π_O використовується методика з робіт [20, 21]. Розбиття виконується так, щоб для кожного класу $O^k \in \Pi_O$ виконувалася умова $R_k + L_k \le S_L$. Крім того, кількість класів K сумісних виходів має бути мінімальною.

Для нашого прикладу отримуємо розбиття $\Pi_O = \{O^1, O^2\}$ із числом класів K = 2. Кожен клас має M_k = 3 виходів: $O^1 = \{o_2, o_3, o_6\}$ і $O^2 = \{o_4, o_5, o_7\}$. Використовуючи (11), отримаємо $R_1 = R_2 = 2$, $\tau^1 = \{\tau_1, \tau_2\}$, $\tau^2 = \{\tau_3, \tau_4\}$, $\tau = \{\tau_1, \ldots, \tau_4\}$.

Закодуємо співвідношення $o_g \notin O^k$ кодом 00, а виходи — природним способом: $C(o_2) = C(o_4) = 01$, $C(o_3) = C(o_5) = 10$ и $C(o_6) = C(o_7) = 11$. Використовуючи ці часткові коди, адреси МК та формули переходу між вершинами ГСА Γ_1, отримаємо таблиці блоків *LUTer*1 та *LUTer*2.

З ГСА Γ_1 маємо такі дві системи формул переходів:

$$o_2, o_3, o_6 \to x_1 b_{12} \vee \overline{x_1} b_{15}, \tag{17}$$

$$o_4 \to x_2 b_7 \vee \overline{x_2} b_{10}; \; o_5 \to b_3; \; o_7 \to b_E. \tag{18}$$

Система (17) визначає таблицю блоку *LUTer*1 (табл. 2). Система (18) визначає таблицю блоку *LUTer*2 (табл. 3). Ці таблиці мають такі самі стовпці, як і табл. 1, тільки стовпець $K(o_g)$ замінюється стовпцем $C(o_g)$.

З табл. 2 формується наступна СБФ:

$$D_1^1 = 0; \quad D_2^1 = \tau_1 \vee \tau_2; \quad D_3^1 = \tau_2 \overline{x_1} \vee \tau_1 \overline{x_1};$$
$$D_4^1 = \tau_1 x_1 \vee \tau_2 x_1; \quad D_5^1 = D_2^1, \tag{19}$$

Таблиця 1. **Таблиця блока *LUTerF***

o_g	$K(o_g)$	b_m	$A(b_m)$	X_h	D_h	h
o_1	1	b_3	00010	$x_1 x_2 x_3$	D_4	1
		b_6	00101	$x_1 x_2 \overline{x_3}$	$D_3 D_5$	2
		b_7	00110	$x_1 \overline{x_2}$	$D_3 D_4$	3
		b_9	01000	$\overline{x_1} x_4$	D_2	4
		b_{10}	01001	$\overline{x_1 x_4}$	$D_2 D_4$	5

Таблиця 2. **Таблиця блока *LUTer*1**

o_m	$C(o_m)$	b_m	$A(b_m)$	X_h	D_h	h
o_2	01	b_{12}	01011	x_1	$D_2 D_4 D_5$	1
		b_{15}	01101	$\overline{x_1}$	$D_2 D_3 D_5$	2
o_3	10	b_{12}	01011	x_1	$D_2 D_4 D_5$	3
		b_{15}	01101	$\overline{x_1}$	$D_2 D_3 D_5$	4
o_6	11	b_{12}	01011	x_1	$D_2 D_4 D_5$	5
		b_{15}	01101	$\overline{x_1}$	$D_2 D_3 D_5$	6

а з табл. 3 — СБФ:

$$D_1^2 = 0; \quad D_2^2 = D_5^2 = \overline{\tau_3}\tau_4\overline{x_2};$$
$$D_3^2 = \overline{\tau_3}\tau_4 x_2; \quad D_4^2 = \tau_3\overline{\tau_4}. \tag{20}$$

У системах (19)—(20) верхній індекс відповідає номеру блока *LUTerk*. Система (19) використовується для реалізації блока *LUTer*1, система (20) —блока *LUTer*2.

Блок *LUTerT* представляється таблицею зі стовпцями $O_V, O^1, ..., O^K$ та рядками $D_1 - D_R$. Якщо $D_r^k \neq \varnothing$, то на перетині рядка D_r та стовпця O^k (або O_V) записується одиниця. У цьому прикладі блок *LUTerT* поданий табл. 4.

З табл. 4 можна отримати наступну СБФ:

$$D_1 = 0; \quad D_2 = D_2^V \vee D_2^1 \vee D_2^2; \quad D_3 = D_3^V \vee D_3^1 \vee D_3^2;$$
$$D_4 = D_4^V \vee D_4^1 \vee D_4^2; \quad D_5 = D_5^V \vee D_5^1 \vee D_5^2.$$

Ця СБФ описує схему блока *LUTerT*.

Для формування СБФ (14)—(15) необхідно побудувати таблицю блоку *LUTerτV*. Цей блок перетворює адреси МК на часткові коди $K(o_g)$ і $C(o_g)$. Таблиця має стовпці o_g, $A(b_m)$, $K(o_g)$, $C(o_g)$, V_g, τ_g, g. Стовпець V_g містить змінні $v_r \in V$, рівні одиниці у коді $K(o_g)$. Стовпець τ_g містить змінні $\tau_r \in \tau$, рівні одиниці у коді $C(o_g)$. Для нашого прикладу блок *LUTerτV* поданий табл. 5.

Таблиця 3. **Таблиця блока *LUTer*2**

o_m	$C(o_m)$	b_m	$A(b_m)$	X_h	D_h	h
o_4	01	b_7	00110	x_2	D_3D_4	1
		b_{10}	01001	$\overline{x}_2$	D_2D_5	2
o_5	10	b_3	00010	1	D_4	3
o_7	11	b_E	00000	1	—	4

Таблиця 4. **Таблиця блока *LUTerτT***

D_r	O_V	O^1	O^2
D_1	0	0	0
D_2	1	1	1
D_3	1	1	1
D_4	1	1	1
D_5	1	1	1

Таблиця 5. **Таблиця блока *LUTerτV***

o_g	$A(b_m)$	$K(o_g)$	$C(o_g)$	V_g	τ_g	g
o_1	00001	1	00	v_1	—	1
o_2	00100	0	01	—	τ_2	2
o_3	00111	0	10	—	τ_1	3
o_4	01010	0	01	—	τ_4	4
o_5	01101	0	10	—	τ_3	5
o_6	01111	0	11	—	$\tau_1\tau_2$	6
o_7	10001	0	11	—	$\tau_3\tau_4$	7

З табл. 5 маємо СБФ:

$$v_1 = \overline{T_1 T_2 T_3 T_4}; \tag{21}$$

$$\begin{aligned} &\tau_1 = \overline{T_1} T_3 T_4 T_5; \quad \tau_3 = \overline{T_1} T_2 T_3 \overline{T_4} T_5 \vee T_1 \overline{T_2 T_3 T_4} T_5; \\ &\tau_2 = \overline{T_1 T_2} T_3 \overline{T_4 T_5} \vee \overline{T_1} T_2 T_3 T_4 T_5; \quad \tau_4 = \overline{T_1} T_2 \overline{T_3} T_4 \overline{T_5} \vee T_1 \overline{T_2 T_3 T_4} T_5. \end{aligned} \tag{22}$$

Як випливає із систем (21) – (22), функції v_1, τ_1 мають по 4 змінних. Інші функції залежать від п'яти змінних. Тому схема блока *LUTerτV* має два рівні елементів *LUT*.

Таблиця блока *EMBer* містить стовпці *Адреса*, b_m, y_0, y_E, y_1, …, y_N. Заповнення таблиці проводиться на основі аналізу ГСА Г та множини $C = \{\alpha_1, ..., \alpha_G\}$. Якщо вершина b_m не є виходом ОЛЛ, за адресою $A(b_m)$ записується y_0. Якщо вершина b_m пов'язана з вершиною b_E, за адресою $A(b_m)$ записується y_E. Для нашого прикладу фрагмент вмісту КП наведено у табл. 6.

Після цього кроку отримано всю інформацію для реалізації схеми КМПУ. Зазначимо, що для системи (22) необхідно виконати функціональну декомпозицію функцій τ_2 – τ_4.

Аналіз запропонованого методу

Основною метою запропонованого методу є зменшення кількості елементів *LUT* та їх міжз'єднань у блоці адресації МК. Метод застосовується, якщо для ГСА Г виконується умова (1). При цьому ГСА є лінійною [1] і потрібна декомпозиція функцій збудження пам'яті, що визначається умовою (6).

Ми пропонуємо використовувати два ядра часткових функцій відповідно до умови (8) У результаті одне ядро засноване на функціональній декомпозиції [8]. Питання оптимізації цієї частини схеми адресації ми не розглядаємо. Для вирішення цієї проблеми є ефективні методи.

Ми пропонуємо метод оптимізації ядра, для якого умова (8) порушується. Як альтернатива функціональный декомпозиції пропонується використовувати структурну декомпозицію [9]. Як відомо [22], структурна декомпозиція призводить до схем, характеристики

Таблиця 6. **Фрагмент таблиці блока *EMB***

Адреса	b_m	y_O	y_E	y_1	y_2	y_3	y_4	y_5	y_6
0000	b_1	1	0	1	1	0	0	0	0
0001	b_2	0	0	0	0	1	0	0	0
0010	b_3	1	0	1	0	1	1	0	0
0011	b_4	1	0	0	0	0	0	1	0
0100	b_5	0	0	0	1	0	0	0	1

яких краще, ніж у еквівалентних схем, заснованих на функціональній декомпозиції.

Як випливає з рис.4, запропонована організація КМПУ має три рівні елементів *LUT*. Розглянемо шляхи оптимізації блоків КМПУ U_2.

Перший рівень схеми складається з K блоків елементів *LUT*. Для синтезу цього блока виконується розбиття багатьох виходів ОЛЛ на класи сумісних виходів. Відповідно до [22], кожна часткова функція реалізується одним елементом *LUT*.

Єдиний шлях оптимізації схеми блока *LUTerk* — зменшення числа функцій, що генерує цей блок. Переходи з виходів ОЛЛ відбуваються або у входи ОЛЛ, або в кінцеву вершину b_E. Для цього необхідно збільшити кількість нульових розрядів адрес входів. Цей підхід є напрямом наших подальших досліджень.

Схема блока *LUTerT* є оптимальною у разі виконання умови:

$$K \le S_L. \tag{23}$$

У цьому випадку блок *LUTerT* складається з R елементів *LUT*, де значення R визначається формулою (2).

Якщо умова (23) порушується, то схема блока *LUTerT* стає багаторівневою. Кількість рівнів можна зменшити, якщо кожна функція $D_r \in D$ складається з $K(D_r)$ часткових функцій, і виконується умова $K(D_r) \le S_L$. У разі виконання цієї умови для всіх ФЗП схема блока *LUTerT* має один рівень. У подальших дослідженнях ми плануємо розробити метод, що дозволяє виконати цю умову для якомога більшої кількості ФЗП $D_r \in D$.

У разі виконання умови $R > S_L$ схема блока *LUTer*т має кілька рівнів. З такою ситуацією ми мали справу в прикладі.

Для покращення характеристик блока *LUTerтV* необхідно зменшувати кількість аргументів у функціях (14), (15). Цього можна досягти за допомогою урахування надлишковості блока *EMB*. За виконання умови $\Delta_A = 2^{S_A} - M > 0$ в КП є Δ_A осередків, у яких не записані МК. Цей факт можна врахувати під час виконання адресації. Розробка такого методу також є спрямуванням наших подальших досліджень.

Висновки

Запропонований у роботі метод можна застосувати, якщо виникає проблема декомпозиції функцій схеми адресації мікрокоманд КМПУ із загальною пам'яттю [1]. Ми розглядаємо ситуацію, коли схема КМПУ реалізується у базисі *FPGA*. При цьому всі комбінаційні блоки реалізуються за допомогою елементів *LUT*.

Необхідність у запропонованому методі виникає, якщо для виходів деяких ОЛЛ виконується умова (8). Для таких виходів слід застосовувати методи функціональної декомпозиції. У цій роботі ми пропонуємо один із можливих шляхів оптимізації схеми КМПУ.

У статті пропонується гетерогенний підхід до формування часткових функцій. Для цієї мети у схемі виділяються два ядра часткових функцій. Одне ядро засноване на функціональній декомпозиції. Друге ядро засноване на адаптації методів подвійного кодування станів [21] до особливостей КМПУ. Практично виходи ОЛЛ виконують функції станів, а адреси МК — функції кодів станів.

У процесі синтезу КМПУ із двома ядрами часткових функцій виникає низка оптимізаційних завдань. Ця необхідність викликається дисбалансом між: 1) числом класів розбиття множини ОЛЛ та числом входів елемента *LUT*, а також 2) числом адресних змінних та числом входів елемента *LUT*. Ми проаналізували ці ситуації та визначили відповідні їм умови. У подальших дослідженнях ми маємо намір розробити метод зменшення впливу зазначених факторів.

Окрім КМПУ із загальною пам'яттю відомі й інші архітектури КМПУ [1]. Крім подвійного кодування станів відомі інші підходи до структурної декомпозиції схем. У подальших наших дослідженнях планується розробка структур та методів синтезу схем КМПУ з іншою архітектурою, що ґрунтується на різних відомих методах структурної декомпозиції.

ПОВІДОМЛЕННЯ:

1) Внесок авторів: Баркалов О.О.: основна ідея, методологія, оригінальна чернетка; Тітаренко Л.О.: узагальнення, концептуалізація, методологія; Матвієнко О.В.: формальний аналіз, ресурси, написання, рецензування та редагування; 2) ШІ та інші інструменти під час написання статті не використовувалися; 3) Конфлікти інтересів відсутні ;4) Гранти або фінансова підтримка не використовувалися.

ЛІТЕРАТУРА / REFERENCES

1. Barkalov A., Titarenko L. Logic Synthesis for Compositional Microprogram Control Units. *Lectures Notes in Electrical Engineering*, Springer, 2008, Vol. 22, 272 p. https://doi.org/10.1007/978-3-540-69285-0
2. Barkalov A. Microprogram control unit as composition of automate with programmable and hardwired logic. *Automatics and Computer Science,* 1983, Vol. 17, 36–41.
3. Feng W., Greene J., Mishchenko A. Improving FPGA Performance with a S44 LUT Structure. *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (NewYork,NY,USA), FPGA'18*, 2018, Association for Computing Machinery, 61–66. https://doi.org/10.1145/3174243.317427
4. Kuon I., Tessier R., Rose J. FPGA Architecture: Survey and Challenges. *Foundations and Trends in Electronic Design Automation*, 2008, Vol 2 (2), 135–253. https://doi.org/10.1561/1000000005
5. DeMicheli G. *Synthesis and optimization of digital circuits*. New York: McGraw-Hill, 1994, 576 p.
6. Ruiz-Rosero J., Ramirez-Gonzalez G., Khanna R. Field Programmable Gate Array Applications—A Scientometric Review. *Computation*, 2019, Vol. 7 (4), Article 63. https://doi.org/10.3390/computation7040063
7. Czerwinski R., Kania D. Synthesis method of high speed finite state machines. *Bulletin of the polish academy of sciences. Technical sciences*, 2010, Vol. 58 (4), 635–644. https://doi.org/10.2478/v10175-010-0067-6

8. Kubica M., Opara A., Kania D. *Technology Mapping for LUT- based. FPGA*. Springer, 2021. https://doi.org/10.1007/978-3-030-60488-2
9. Barkalov A., Titarenko L., Mielcarek K., Chmielewski S. Logic Synthesis for FPGA–Based Control Units. *Structural Decomposition in Logic Design*. *Lecture Notes in Electrical Engineering*, Springer, 2020, Vol. 636. https://doi.org/10.1007/978-3-030-38295-7
10. Grout I. *Digital systems design with FPGAs and CPLDs*. Elsevier, Amsterdam, 2008, 784 p. https://doi.org/10.1016/B978-0-7506-8397-5.X0001-3
11. Barkalov O., Titarenko L., Saburova S., Golovin, O., V. Matvienko O. Optimization of a composite microprogram control device scheme with a basic architecture. *Cybernetics and Computer Technologies,* 2024, Issue 4, 121–133. https://doi.org/10.34229/2707-451X.24.4.11
12. Baranov S. *Logic synthesis for control automata*. Dordrecht: Kluwer Academic Publishers, 1994, 312 p. https://doi.org/10.1007/978-1-4615-2692-6
13. Chapman K. Multiplexer Design Techniques for Datapath Performance with Minimized Routing Resources. *Xilinx All Programmable*, 2014. URL: https://docs.amd.com/api/khub/documents/9pOn~3NV8ApAbwglqp6MJQ/content [Accessed Jan. 2022]
14. Trimberg S. Three ages of FPGA: A retrospective on the first thirty years of FPGA technology. *IEEE Proceedings 103*, 2015, Vol. 3, 318–331. https://doi.org/10.1109/JPROC.2015.2392104
15. Tiwari A., Tomko K. Saving power by mapping finite state machines into embedded memory blocks in FPGAs. *Design, Automation and Test in Europe Conference and Exhibition,* Paris, France, 2004, Vol. 2, 916–921. https://doi.org/10.1109/DATE.2004.1269007
16. Senhaji-Navarro R., Garcia-Vargas I., Jimenes-Moreno G., Civit-Balcells A., Guerra-Gutierres P. ROM-based FSM implementation using input multiplexing in FPGA devices. *Electronics Letters*, 2004, Vol. 40 (20), 1249–1251. https://doi.org/10.1049/el:20046007
17. Xilinx. URL: https://www.amd.com/en/products/adaptive-socs-and-fpgas/fpga.html#overview [Accessed Jan. 2025]
18. Vivado Design Suite. URL: https://www.xilinx.com/products/design-tools/vivado.html,2020 [Accessed Jan. 2025]
19. Quartus II. URL: https://www.intel.com/content/www/us/en/products/details/fpga/development-tools/quartus-prime/resource.html [Accessed Jan. 2025]
20. Barkalov O., Titarenko L., Mielcarek K. Hardware reduction for LUT based Mealy FSMs. *International Journal of Applied Mathematics and Computer Science*, 2018, Vol. 28 (3), 595–607. https://doi.org/10.2478/amcs-2018-0046
21. Barkalov A., Titarenko L., Mielcarek K. Hardware reduction for LUT–based Mealy FSMs. *International Journal of Applied Mathematics and Computer Science,* 2018, Vol. 28 (3), 595 – 607. https://doi.org/10.2478/amcs-2018-0046
22. Opara A.; Kubica M.; Kania D. Decomposition Approaches for Power Reduction. IEEE Access 2023, 11, 29417–29429

Отримано/Received 18.05.2026
Прийнято/Accepted 11.07.2026
Опубліковано/Published 25.08.2026

O.O. BARKALOV, DSc (Engineering), Professor,
University of Zielona Góra,
ul. Licealna 9, 65-417, Zielona Góra, Poland
https://orcid.org/0000-0002-4941-3979; A.Barkalov@iie.uz.zgora.pl
L.O. TITARENKO, DSc (Engineering), Professor,
University of Zielona Góra,
9, Licealna Str., Zielona Gora, 65-417, Poland
Kharkiv National University of Radioelectronics,
14, Science ave., Kharkiv, 61166, Ukraine
https://orcid.org/0000-0001-9558-3322; L.Titarenko@iie.uz.zgora.pl
O.V. MATVIIENKO, Research Associate,
Glushkov Institute of Cybernetics of the NAS of Ukraine,
40, Hlushkova Akad. ave., Kyiv, 03187, Ukraine
https://orcid.org/0000-0003-1838-1422; avmatv@ukr.net

SYNTHESIS OF A COMPOSITIONAL MICROPROGRAM
CONTROL UNIT WITH TWO CORES OF PARTIAL FUNCTIONS

Introduction. Digital systems consist of combinational and sequential blocks. The most important sequential blocks include control units (CU). The circuits of CUs are not standard library components in CAD systems, so designing a particular CU circuit is a more labor-intensive process than implementing systems from common blocks such as registers, counters, arithmetic blocks, and logic blocks.

The quality of a digital system depends on the optimal characteristics of the CU. When synthesizing a CU circuit, it is necessary to solve a number of optimization problems: reducing the chip area occupied by the CU, increasing performance, and reducing power consumption. It is believed that solving the first of these problems allows improving other characteristics of the CU circuit.

Purpose of the work. The main focus of this article is the development of effective methods for optimizing the parameters of CU circuits when implemented on an FPGA (field-programmable gate array) basis.

Methods. A method for structural decomposition of a Compositional Microprogram Control Unit (CMCU) circuit in an FPGA is proposed. The method is based on the identification of two cores of partial functions. One core is based on the functional decomposition and generates functions for which the number of arguments exceeds the number of LUT inputs. The second core is based on the adaptation of twofold state assignments methods to the specifics of CMCU. This core generates functions based on the encoding of operator linear chains.

Results. An CMCU architecture with two cores of partial functions and a method for synthesizing the CMCU circuit are proposed. The FPGA implementation of the CMCU circuit used look-up tables (LUTs) and embedded memory blocks (EMBs). Since AMD Xilinx is the dominant FPGA chip manufacturer, the method proposed in this article is tailored to FPGAs from this company.

An example of synthesizing a CMCU circuit with two cores is presented. The conditions for applying this approach are discussed. Possible solutions to optimization problems associated with imbalances in the characteristics of the control algorithm and LUT elements are demonstrated.

Conclusion. During the synthesis of a CMCU with two cores of partial functions, a number of optimization problems arise. This is necessitated by an imbalance between: 1) the number of classes in the operator linear chains set partitioning and the number of LUT element inputs, and 2) the number of address variables and the number of LUT element inputs. This article analyzes these situations and identifies the corresponding conditions. In our future research, we intend to develop a method for mitigating the impact of these factors.

Keywords*: partial function, kernel, CMCU, LUT, EMB, synthesis, structural decomposition.*