Кібернетика та комп'ютерні технології

УДК 004.274 DOI:10.34229/2707-451X.21.1.9

О.О. БАРКАЛОВ, Л.О. ТІТАРЕНКО, О.М. ГОЛОВІН, О.В. МАТВІЄНКО

ПОДВІЙНА АДРЕСАЦІЯ МІКРОКОМАНД У КМПК ІЗ ЗАГАЛЬНОЮ ПАМ'ЯТТЮ

Запропоновано метод зменшення апаратурних витрат у схемі композиційного мікропрограмного пристрою керування (КМПК), який реалізується в базисі FPGA. Метод заснований на перетворенні адрес мікрокоманд у коди виходів операційних лінійних ланцюгів (ОЛЛ). Для оптимізації схеми КМПК множина ОЛЛ розбивається на класи. Розбиття проводиться таким чином, що блок адресації мікрокоманд має точно два рівня елементів табличного типу. Керуюча пам'ять КМПК реалізується на вбудованих блоках пам'яті. В роботі розглянуто приклад синтезу схеми КМПК і виконаний аналіз запропонованого методу.

Ключові слова: *композиційний мікропрограмний пристрій керування, мікрокоманда, LUT, EMB, синтез.*

© О.О. Баркалов, Л.О. Тітаренко, О.М. Головін, О.В. Матвієнко, 2021

Вступ. Пристрій керування – один з важливих блоків практично будь-якої цифрової системи [1, 2]. Характеристики пристрою керування в значній мірі визначають характеристики системи в цілому [3]. Одна з основних характеристик пристрою керування – апаратурні витрати, які багато в чому визначають ціну і розмір системи в цілому. Методи зменшення витрат апаратури залежать від особливостей елементного базису і алгоритму керування, реалізованого пристроєм керування. У даній роботі розглядаються композиційні мікропрограмні пристрої керування (КМПК) [4], що реалізуються в базисі мікросхем FPGA (field-programmable logic arrays) [5, 6].

КМПК є автоматом Мура, в якому регістр станів замінений лічильником адреси мікрокоманд. КМПК призначений для реалізації лінійних алгоритмів керування [4]. Для формування вихідних сигналів (мікроопераций) у КМПК використовуються блоки пам'яті. Комбінаційна частина КМПК формує адреси мікрокоманд як функції логічних умов і вмісту лічильника адреси мікрокоманд.

Мікросхеми FPGA – один з найпоширеніших базисів, які використовуються для побудови цифрових систем [7, 8]. Для реалізації схем КМПК досить таких компонент FPGA, як логічні елементи табличного типу (LUT, look-up table), програмовані тригера, вбудовані блоки пам'яті EMB (embedded memory blocks) і програмовані міжпоєднання.

Основний недолік елементів LUT – дуже мале число входів S_L [6]. У роботах [9, 10] показано, що значення $S_L = 6$ забезпечує необхідний баланс між такими характеристиками елемента, як займана площа кристала, споживана потужність і швидкодія. Однак, сучасні цифрові системи можуть генерувати близько 50 логічних умов, що надходять у пристрій керування. Така невідповідність між характеристиками алгоритму керування та характеристиками елементів LUT призводить до багаторівневих схем пристроїв керування. Як відомо, чим більше елементів і міжпоєднань має схема, тим більше споживана нею енергія і займана площа кристала. Крім того, багаторівневі пристрої керування мають значно меншу швидкодію, ніж їх одноріневі аналоги [7].

Якщо отримані при проектуванні характеристики КМПК не відповідають технічному завданню, то необхідно оптимізувати ці характеристики.

У даній роботі пропонується метод подвійної адресації мікрокоманд в КМПК із загальною пам'яттю. Цей метод – адаптація дворівневого кодування станів автоматів Мілі [11], схеми яких реалізуються в базисі FPGA. Як показали наші дослідження, запропонований метод дозволяє отримати схему адресації мікрокоманд з двома рівнями логіки і регулярною системою міжпоєднань. Для завдання алгоритму керування ми використовуємо мову граф-схем алгоритму (ГСА) [3].

Реалізація КМПК в базисі FPGA. Для реалізації схеми КМПК необхідно знайти операторні лінійні ланцюги (ОЛЛ) [4], що складаються з операторних вершин ГСА. Розглянемо ГСА Γ_1 (рис. 1).

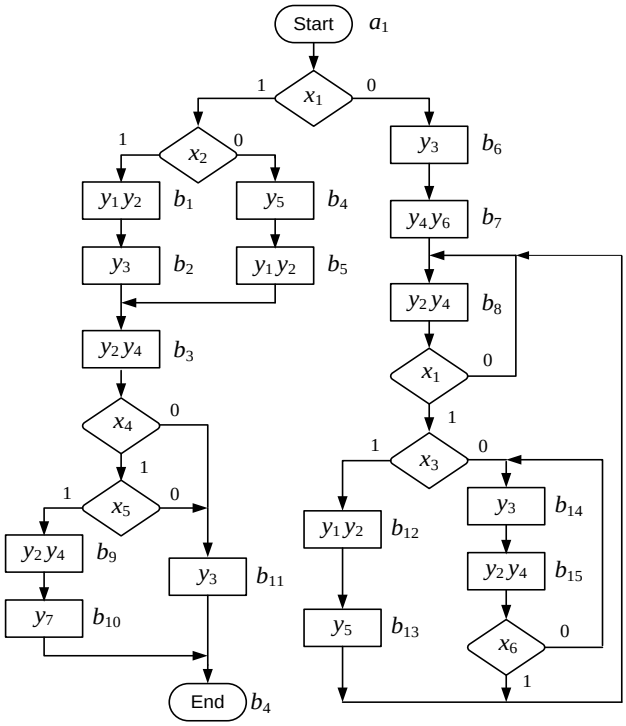

РИС. 1. Початкова ГСА Γ_1

Як і будь-яка ГСА, вона складається з початкової вершини (Start) b_0, кінцевої вершини (End) b_E, операторних вершин $b_1 - b_{15}$ й умовних вершин. В операційних вершинах записані набори мікрооперацій (НМО) $Y_q \subseteq Y$, де $Y = \{y_1,\ldots,y_N\}$ – множина мікрооперацій. В умовних вершинах записані логічні умови $x_l \in X$, де $X = \{x_1,\ldots,x_L\}$ – множина логічних умов. Операційні вершини утворюють множину $B = \{b_1,...,b_I\}$. Для ГСА Γ_1 маємо I = 15, L = 6 і N = 7. Розглянемо ряд визначень [4], необхідних для подальшого викладу матеріалу.

Визначення 1. Кінцева послідовність операторних вершин $\alpha = < b_{gi}, ..., b_{gFg} >$ називається операторним лінійним ланцюгом ГСА Γ, якщо для будь-якої пари сусідніх компонент α_g існує дуга $< b_{gi}, ..., b_{gi+1} >$, де i – номер компоненти вектора α_g.

Визначення 2. Входом операторного лінійного ланцюга α_g називається вершина, вхід якої пов'язаний з виходом початкової або умовної вершини, або з виходом операторної вершини, що не входить в операторний лінійний ланцюг α_g.

Визначення 3. Виходом операторного лінійного ланцюга α_g називається вершина, вихід якої пов'язаний з входом вершини, що не входить в операторний лінійний ланцюг α_g.

Кожна операторна вершина відповідає мікрокоманді, що зберігається в керуючій пам'яті [12]. Нехай ГСА Γ включає M операторних вершин, а керуюча пам'ять зберігає M_0 мікрокоманд. Якщо вихід початкової вершини пов'язаний з входом умовної вершини, то існує операторний лінійний ланцюг $\alpha_0 = < b_0 >$. В цьому випадку $M_0 = M + 1$. В іншому випадку $M_0 = M$.

Отже, вершина b_m відповідає мікрокоманді MI_m, що має адресу A_m, $(m \in \{0, ..., M\})$. Розрядність адреси мікрокоманди визначається як

$$R = \lceil \log_2 M_0 \rceil. \quad (1)$$

Побудуємо для ГСА Γ множину $C = \{\alpha_0, \alpha_1, ..., \alpha_G\}$ таку, що: 1) кожен елемент C є операторний лінійний ланцюг; 2) кожна операторна вершина включена тільки в один операторний лінійний ланцюг $\alpha_g \in C \setminus \{\alpha_0\}$; 3) число операторних лінійних ланцюгів є мінімальним. Для побудови множини C може бути використаний алгоритм [4].

Виконаємо природну адресацію мікрокоманд у межах кожного операторного лінійного ланцюга. При цьому, для кожної пари сусідніх компонент b_{gi}, b_{gi+1} виконується умова $A_{gi+1} = A_{gi} + 1$, де $g \in \{0, ..., G\}$, $i \in \{0, ..., F_g\}$.

В цьому випадку пристрій керування може бути представлений у вигляді КМПК U_1 (рис. 2).

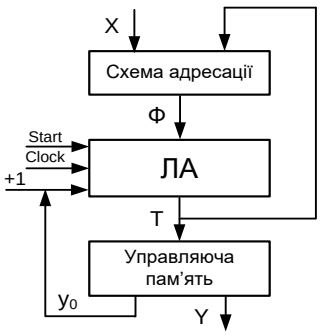

РИС. 2. Структурна схема КМПК U_1

Розглянемо значення символів T, Φ, y_0, *Clock*, *Start* та +1. Адреси A_m представляються векторами з елементів $T_r \in \{T_1, ..., T_R\}$, де T – множина внутрішніх змінних [3]. Для зміни вмісту лічильника адреси мікрокоманд (*ЛА* на рис. 2) використовуються функції збудження пам'яті $D_r \in \Phi = \{D_1, ..., D_R\}$, де Φ – множина функцій збудження пам'яті. За сигналом *Start* в *ЛА* записується початкова адреса мікропрограми. Вважатимемо, що ця адреса складається з нулів. Сигнал синхронізації *Clock* дозволяє зміну вмісту *ЛА*. Якщо $y_0 = 1$, то $ЛА := ЛА + 1$ (це безумовний перехід у межах одного операторного лінійного ланцюга). Якщо $y_0 = 0$, то $ЛА := < \Phi >$ (це відповідає переходу між виходом і входом операторного лінійного ланцюга $\alpha_g \in C$). Сигнал «+ 1» викликає збільшення вмісту *ЛА* при $y_0 = 1$.

Схема адресації формує системи функцій збудження пам’яті

$$\Phi = \Phi\,(\mathrm{T},\ \mathrm{X}). \tag{2}$$

Адреса переходу для моменту часу t, $(t = 0,\ 1,\ 2,\ ...)$ визначається виразом

$$A^{t+1} = \begin{cases} A^t + 1, \text{ якщо } y_0 = 1; \\ \Phi\,(T,\ X), \text{ якщо } y_0 = 0. \end{cases} \tag{3}$$

КМПК U_1 називається КМПК із загальною пам’яттю [4] і функціонує так. За сигналом *Start* початкова адреса мікропрограми завантажується в лічильник адреси мікрокоманд. Нехай в момент часу t в лічильнику адреси мікрокоманд знаходиться адреса A_t. Відповідна мікрокоманда вибирається з керуючої пам'яті (КП). Мікрооперації $y_n \in Y$ надходять в операційні блоки системи. Після виконання операції формуються значення логічних умов $x_l \in X$. Якщо виконана остання мікрокоманда прошивки, формується сигнал y_E (на рис. 2 не показаний). При цьому функціонування припиняється. В іншому випадку формується адреса переходу (3).

Якщо КМПК U_1 реалізується в базисі *FPGA*, то схема адресації і лічильник адреси мікрокоманд будуються на елементах *LUT* і тригерах, відповідно. Позначимо схему адресації символом *LUTerA*. В роботі [7] символом *LUTer* позначають блок, що складається з елементів *LUT*.

Для реалізації КП доцільно використовувати блоки *EMB*. Ці блоки мають можливість конфігурації [13 – 15]. Для *EMB* можна змінювати параметри S_A (число адресних входів) і t_F (число виходів даних пам’яті) при постійному обсязі.

Пара $< S_A,\ t_F >$ визначає конфігурацію *EMB*. Для сучасних блоків *EMB* [6] існують такі конфігурації: $R_0 - R_E$.

Блоки *EMB* можна використовувати для реалізації КП, якщо є конфігурація $< S_0,\ t_0 >$, для якої $S_0 = R$.

Розглянемо випадок, коли для кодування мікрооперацій використовуються унітарні коди. З урахуванням змінних y_0 і y_E, число блоків *EMB* в КП визначається як

$$n_{EMB} = \left\lceil \frac{N+2}{t_0} \right\rceil. \tag{4}$$

Позначимо символом *EMBer* блок, що складається з декількох *EMB*. З урахуванням усього вищевикладеного, КМПК U_1 можна представити структурною схемою (рис. 3).

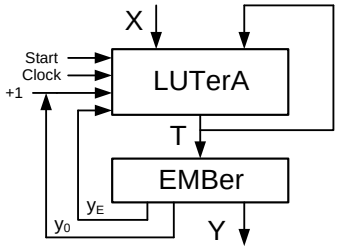

РИС. 3. Реалізація КМПК U_1 в базисі *FPGA*

Основна ідея пропонованого методу. В КМПК U_1 *LUTerA* реалізує систему функцій (2). Функція $D_r \in \Phi$ залежить від $NA(D_r)$ аргументів, що включають логічні умови $x_l \in X$ і внутрішні змінні $T_r \in T$. При виконанні умови

$$NA(D_r) > S_L, \ (r \in \{1, ..., R\}) \tag{5}$$

схема блоку *LUTerA* має кілька рівнів логічних елементів. Це призводить до зменшення швидкодії і збільшення споживаної енергії [16].

У цій роботі ми пропонуємо метод, що дозволяє гарантоване отримання дворівневої схеми блоку *LUTerA*. Запропонований метод використовує ідею подвійного кодування станів [11].

Нехай для деякої ГСА Γ отримано множину операторних лінійних ланцюгів $C = \{\alpha_1, ..., \alpha_G\}$. Кожний операторний лінійний ланцюг $\alpha_g \in C$ має тільки один вихід O_g. Сформуємо множину виходів $O(\Gamma)$, що включає виходи операторного лінійного ланцюга $\alpha_g \in C$. Функції (2) формуються тільки для виходів $O_g \in O(\Gamma)$. Таким чином, виходи $O_g \in O(\Gamma)$ – аналоги станів автомата Мура.

Очевидно, адреса $A(O_g)$ виходу $O_g \in O(\Gamma)$ аналогічна коду стану, який відзначає відповідну операторну вершину ГСА Γ. Адреси виходів формуються в процесі природної адресації компонент операторного лінійного ланцюга $\alpha_g \in C$. Тому ці адреси є фіксованими і не можуть призначатися довільно.

Побудуємо розбиття $\Pi_0 = \{O^1, ..., O^I\}$ множини $O(\Gamma)$, яке задовольняє наступній умові:

$$R_i + L_i \le S_L, \ (i \in \{1, \ldots, I\}), \tag{6}$$

де R_i – число внутрішніх змінних, необхідних для кодування виходів $O_g \in O^i$, а L_i означає число логічних умов, що визначають переходи з виходів $O_g \in O^i$.

Нехай $M_i = |O^i|$ тоді значення R_i визначається наступним чином:

$$R_i = \lceil \log_2(M_i + 1) \rceil, \quad (i = \overline{1, I}), \tag{7}$$

де 1 додається для урахування співвідношення $O_g \notin O^i$.

Адреси $A(O_g)$ зберігаються в лічильнику адреси мікрокоманд і представляються змінними $T_r \in T$. Використаємо для кодування елементів множини $O^i \in \Pi_0$ змінні $\tau_r \in \tau$. Потужність множини τ визначається наступним чином: $R_0 = R_1 + R_2 + ... + R_I$.

Кожен клас $O^i \in \Pi_0$ визначає множини τ^i, X^i і Φ^i. Множина $\tau^i \subseteq \tau$ включає додаткові змінні, які кодують виходи $O_g \in O^i$. Множина $X^i \subseteq X$ включає логічні умови, що визначають переходи з виходів $O_g \in O^i$. Множина $\Phi^i \subseteq \Phi$ включає функції збудження пам'яті, рівні 1 на переходах з виходів $O_g \in O^i$. Ці функції збудження пам'яті визначаються системою функцій:

$$\Phi^i = \Phi(\tau^i, X^i). \tag{8}$$

Для реалізації системи (8) використовується блок *LUTeri* $(i \in \{1,\ldots,I\})$. Очевидно, блок *LUTeri* реалізує часткові функції $D_r^i \in \Phi^i$. Для формування функції $D_r \in \Phi$ необхідно знайти диз'юнкцію

$$D_r = \bigvee_{i=1}^{I} D_r^i, \quad (r \in \{1,\ldots,R\}). \tag{9}$$

Формування функцій (9) виконує блок *LUTerT*. Блок *LUTerT* містить *R* тригерів, що утворюють розподілений лічильник адреси мікрокоманд. При виконанні умови $I \le S_L$ блок *LUTerT* включає рівно *R* елементів *LUT*. У цій роботі ми пропонуємо структурну схему КМПК U_2 (рис. 4).

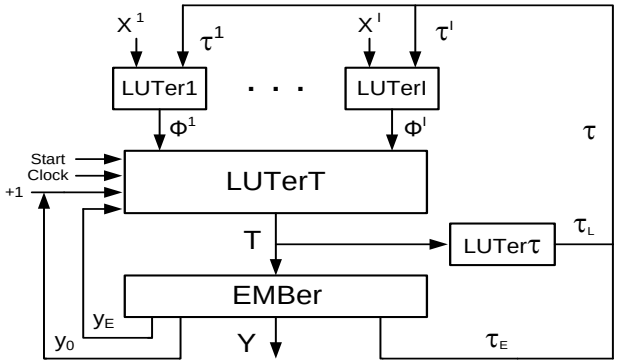

РИС. 4. Структурна схема КМПК U_2

Блоки *LUTer1, ..., LUTerI, LUTerT* замінюють блок *LUTerA*. Для формування кодів виходів $K(O_g)$ необхідно реалізувати систему функцій $\tau = \tau(T)$. В КМПК U_2 для цих цілей використовується блок *EMBer* і додатковий блок *LUTer*τ.

Блок *EMBer* складається з n_{EMB} блоків *EMB*, що мають у сумі $t_E = t_0 \times n_{EMB}$ виходів. При цьому R_E виходів є вільними, де $R_E = t_E - (N+2)$. Ми пропонуємо використовувати ці виходи для реалізації функцій $\tau_r \in \tau_E$.

Множина τ розбивається на непересічні множини τ_E і τ_L. Функції $\tau_r \in \tau_L$ реалізуються блоком *LUTer*τ. Якщо $R_E = 0$, то $\tau_L = \tau$. Якщо $R_E \ge R_0$, то $\tau_E = \tau$ і $\tau_L = \varnothing$.

Метод синтезу КМПК U_2 включає такі етапи:

1. Формування множини операторних лінійних ланцюгів $C = \{\alpha_1,\ldots,\alpha_G\}$ для вихідної ГСА Γ.
2. Природна адресація мікрокоманд у межах кожного операторного лінійного ланцюга.
3. Формування множини виходів операторних лінійних ланцюгів $O(\Gamma)$.
4. Формування розбиття Π_0, що задовольняє умові (6).
5. Кодування виходів $O_g \in O(\Gamma)$ кодами $K(O_g)$.
6. Формування таблиць блоків *LUTer1 – LUTerI*.
7. Формування систем функцій (8).
8. Формування систем функцій (9).

9. Розбиття множини τ на блоки τ_E і τ_L.
10. Формування таблиці блоку *EMBer*.
11. Формування таблиці блоку *LUTer*τ.
12. Реалізація схеми КМПК в базисі *LUT* і *EMB*.

Зазначимо, що код $K\left(O_g\right)$ можна розглядати як додаткову адресу мікрокоманди, відповідну виходу $O_g \in O\left(\Gamma\right)$. Цим і пояснюється назва нашої статті.

Позначимо символом $U_i\left(\Gamma_j\right)$ той факт, що КМПК U_i реалізується по ГСА Γ_j. Розглянемо приклад синтезу КМПК $U_2\left(\Gamma_1\right)$. При цьому використовуємо елементи *LUT* з $S_L = 5$.

Приклад синтезу КМПК U_2. Для формування операторних лінійних ланцюгів використаємо метод з [4]. Застосування цього методу до *ГСА* Γ_1 дає множину $C = \{\alpha_0,\ \alpha_1,...,\alpha_7\}$, де $\alpha_0 = < b_0 >$, $\alpha_1 = < b_1, b_2, b_3 >$, $\alpha_2 = < b_4, b_5 >$, $\alpha_3 = < b_6, b_7, b_8 >$, $\alpha_4 = < b_9, b_{10} >$, $\alpha_5 = < b_{11} >$, $\alpha_6 = < b_{12}, b_{13} >$, $\alpha_7 = < b_{14}, b_{15} >$. Отже, для ГСА Γ_1 маємо $G = 8$. Виходячи з визначення (2), виходи операторних лінійних ланцюгів є останні компоненти ланцюгів. Це дає множину виходів $O(\Gamma_1) = \{b_0, b_3, b_5, b_8, b_{10}, b_{11}, b_{13}, b_{15}\}$. Зазначимо, що перетворенню $T \to \tau$ підлягають тільки виходи, для яких відсутні безумовні переходи в вершину b_E. В даному випадку, вершини b_{10} і b_{11} мають такі переходи.

Оскільки вершина b_0 входить в операторний лінійний ланцюг α_0, параметр $M_0 = 16$. Використовуючи (1), отримаємо $R = 4$, що дає множини $T = \{T_1, ..., T_4\}$ і $\Phi = \{D_1, ..., D_4\}$. В даному випадку адресація мікрокоманд виконується тривіально: $A_0 = 0000$, $A_1 = 0001,...,$ $A_{15} = 1111$. У загальному випадку для виконання природної адресації необхідно використовувати метод [4].

Крок 4 – це найбільш важливий етап синтезу. Результат його виконання визначає число елементів *LUT* у блоках *LUTer*1, ..., *LUTerI*, *LUTerT*. Для вирішення цього завдання використано алгоритм з [11]. При цьому виходи $O_g \in O\left(\Gamma_1\right)$ розглядаються як стани, що і дозволяє використати метод [11].

Розбиття Π_0 виконується на множині $O^{'}(\Gamma) = \{O_0, O_1, O_2, O_3, O_6, O_7\}$, де $O_0 = b_0$, $O_1 = b_3$, $O_2 = b_5$, $O_3 = b_8$, $O_6 = b_{13}$ і $O_7 = b_{15}$. Для розглянутого прикладу, елементи *LUT* мають 5 входів.

Використовуючи [11], отримаємо розбиття $\Pi_0 = \{O^1, O^2\}$, де $O^1 = \{O_0, O_2, O_3\}$ і $O^2 = \{O_1, O_6, O_7\}$. З ГСА Γ_1 маємо множини $X^1 = \{x_1, x_2, x_3\}$ і $X^2 = \{x_4, x_5, x_6\}$. З умови (7) маємо $R = R_2 = 2$. Це дає множини $\tau^1 = \{\tau_1, \tau_2\}$, $\tau^2 = \{\tau_3, \tau_4\}$ і $\tau = \{\tau_1, ..., \tau_4\}$. Таким чином, отримуємо $R_0 = 4$.

Оскільки умова (6) виконується для обох класів розбиття Π_0, то коди входів не впливають на число елементів *LUT* у блоках *LUTer*1 і *LUTer*2. Тому виходи $O_g \in O^{'}\left(\Gamma_1\right)$ можна закодувати довільним чином. Залишимо код 00 для співвідношення $O_g \notin O^i$. Закодуємо виходи наступним чином: $K(O_0) = K(O_1) = 01$, $K(O_2) = K(O_6) = 10$ і $K(O_3) = K(O_7) = 11$.

Для формування таблиць блоків *LUTer*1 і *LUTer*2 побудуємо системи формул переходу для виходів $O_g \in O^{'}\left(\Gamma_1\right)$. Формули переходу задають зв'язок між виходами і входами операторних лінійних ланцюгів [4].

Блок *LUTer*1 задається наступною системою формул переходу:

$$b_0 \to x_1 x_2 b_1 \vee x_1 \overline{x_2} b_4 \vee \overline{x_1} b_6; \quad b_5 \to b_3; \quad b_8 \to x_1 x_3 b_{12} \vee x_1 \overline{x_3} b_{14} \vee \overline{x_1} b_8. \tag{10}$$

Блок *LUTer*2 задається наступною системою формул переходу:

$$b_3 \to x_4 x_5 b_9 \vee x_4 \overline{x_5} b_{11} \vee \overline{x_4} b_{11}; \quad b_{13} \to b_8; \quad b_{15} \to x_6 b_8 \vee \overline{x_6} b_{14}. \tag{11}$$

Таблиці блоків *LUTer*1, *LUTer*2 мають такі стовпці: O_g – вихід операторного лінійного ланцюга для відповідної формули переходу; $K\left(O_g\right)$ – код виходу O_g; b_q – вершина, до якої відбувається перехід; A_q – адреса мікрокоманди, відповідної вершини b_q; X_h – кон'юнкція логічних умов, що визначає перехід з O_g в b_q; Φ_h – набір функцій збудження пам'яті, рівних 1 для завантаження в лічильник адреси мікрокоманд A_q; h – номер переходу.

Із системи (10) маємо таблицю блоку *LUTer*1 (табл. 1), з системи (11) – таблицю блоку *LUTer*2 (табл. 2).

ТАБЛИЦЯ 1. Таблиця блоку *LUTer*1

O_g	$K(O_g)$	b_q	A_q	X_h	Φ_h	h
O_0	01	b_1	0001	$x_1 x_2$	D_4	1
		b_4	0100	$x_1 \overline{x_2}$	D_2	2
		b_6	0110	$\overline{x_1}$	$D_2 D_3$	3
O_2	10	b_3	0011	1	$D_3 D_4$	4
O_3	11	b_{12}	1100	$x_1 x_3$	$D_1 D_2$	5
		b_{14}	1110	$x_1 \overline{x_3}$	$D_1 D_2 D_3$	6
		b_8	1000	$\overline{x_1}$	D_1	7

ТАБЛИЦЯ 2. Таблиця блоку *LUTer*2

O_g	$K(O_g)$	b_q	A_q	X_h	Φ_h	h
O_1	01	b_9	1001	$x_4 x_5$	$D_1 D_4$	1
		b_{11}	1011	$x_4 \overline{x_5}$	$D_1 D_3 D_4$	2
		b_{11}	1011	$\overline{x_4}$	$D_1 D_3 D_4$	3
O_6	10	b_8	1000	1	D_1	4
O_7	11	b_8	1000	x_6	D_1	5
		b_{14}	1110	$\overline{x_6}$	$D_1 D_2 D_3$	6

З табл. 1 і 2 формуються системи (8). Так з табл. 1 маємо (з урахуванням мінімізації):

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

$$D_3^1=\overline{\tau_1}\tau_2\overline{x_1}\vee\tau_1\overline{\tau_2}\vee\tau_1\tau_2x_2\overline{x_3};\quad D_4^1=\overline{\tau_1}\tau_2x_1x_2\vee\tau_1\overline{\tau_2}.$$

З табл. 2 отримуємо систему функцій для *LUTer*2:

$$D_1^2=\tau_3\vee\tau_4;\quad D_2^2=\tau_3\tau_4\overline{x_6};$$
$$D_3^2=\overline{\tau_3}\tau_4\overline{x_5}\vee\overline{\tau_3}\tau_4\overline{x_4}\vee\tau_3\tau_4\overline{x_6};\quad D_4^2=\overline{\tau_3}\tau_4.$$

Система (9) формується тривіальним чином. В даному випадку вона має такий вигляд:

$$D_r=D_r^1\vee D_r^2,\quad (r\in\{1,...,4\}).$$

Таким чином, кожен з блоків *LUTer*1, *LUTer*2 і *LUTerT* складається з чотирьох елементів *LUT*. В цілому ця частина схеми містить 12 елементів *LUT*.

Нехай для реалізації блоку *EMBer* використовуються блоки *EMB* з конфігурацією $S_A=t_F=4$. Оскільки $S_A=R$, такі блоки пам'яті можна використовувати в нашому прикладі. З формули (4) знайдемо $n_{EMB}=4$ (так як t_0 = 4 і N + 2 = 9). Це дає $t_E=4\times3=12$ і $R_E=3$. Отже, три елементи множини τ можна включити до множини τ_E. Нехай $\tau_L=\{\tau_1\}$ і $\tau_E=\{\tau_2,\tau_3\}$.

Таблиця блоку *EMBer* має M_0 рядків і наступні колонки: MI_m, A_m, y_0, y_E, Y, τ_E, m. Для нашого прикладу блок *EMBer* наведений в табл. 3.

ТАБЛИЦЯ 3. Таблиця блоку *EMBer КМПК* $U_2(\Gamma_1)$

MI_m	A_m	y_O	y_E	y_1	y_2	y_3	y_4	y_5	y_6	y_7	τ_2	τ_3	τ_4	m
MI_0	0000	0	0	0	0	0	0	0	0	0	1	0	0	1 +
MI_1	0001	1	0	1	1	0	0	0	0	0	0	0	0	2
MI_2	0010	1	0	0	0	1	0	0	0	0	0	0	0	3
MI_3	0011	0	0	0	1	0	1	0	0	0	0	0	1	4 +
MI_4	0100	1	0	0	0	0	0	1	0	0	0	0	0	5
MI_5	0101	0	0	1	1	0	0	0	0	0	0	0	0	6 +
MI_6	0110	1	0	0	0	1	0	0	0	0	0	0	0	7
MI_7	0111	1	0	0	0	0	1	0	1	0	0	0	0	8
MI_8	1000	0	0	0	1	0	1	0	0	0	1	0	0	9 +
MI_9	1001	1	0	0	1	0	1	0	0	0	0	0	0	10
MI_{10}	1010	0	1	0	0	0	0	0	0	1	0	0	0	11 −
MI_{11}	1011	0	1	0	0	1	0	0	0	0	0	0	0	12 −
MI_{12}	1100	1	0	1	1	0	0	0	0	0	0	0	0	13
MI_{13}	1101	0	0	0	0	0	0	1	0	0	0	1	0	14 +
MI_{14}	1110	1	0	0	0	1	0	0	0	0	0	0	0	15
MI_{15}	1111	0	0	0	1	0	1	0	0	0	0	1	1	16 +

Таблиця заповнюється наступним чином:

1. У стовпчиках y_1-y_N рядки m записуються мікрооперації з операторної вершини, відповідної мікрокоманди MI_m.

2. Якщо вершина b_q не є виходом операторного лінійного ланцюга, то $y_0 = 1$ для відповідної мікрокоманди.

3. Якщо вихід операторного лінійного ланцюга пов'язаний з входом кінцевої вершини, то $y_E = 1$ (ці виходи відмічені знаком « − » у стовпці *m*).

4. Якщо вихід операторного лінійного ланцюг пов'язаний з входом кінцевої вершини, то $y_E = 1$ (ці виходи відмічені знаком « − » у стовпці *m*).

5. Якщо вихід операторного лінійного ланцюга $\alpha_g \in C$ не пов'язана з b_E (такі виходи відмічені знаком « + » у стовпці *m*), то в стовпцях τ_E записується код $K(O_g)$. Якщо $\tau_r \in \tau_L$, то ці розряди коду $K(O_g)$ формуються схемою *LUTerτ*.

Таблиця блоку *LUTerτ* має стовпці: O_g, A_g, $K(O_g)$, τ_L, g. Для нашого прикладу блок *LUTerτ* наведений у табл. 4.

ТАБЛИЦЯ 4. Таблиця блоку *LUTerτ* КМПК $U_2(\Gamma_1)$

O_g	A_g	$K(O_g)$	τ_L	g
O_2	0101	1000	τ_1	1
O_3	1000	1100	τ_1	2

Для інших виходів операторного лінійного ланцюга маємо $\tau_1 = 0$. З табл. 4 формується рівняння

$$\tau_1 = \overline{T}_1 T_2 \overline{T}_3 \overline{T}_4 \vee T_1 \overline{T}_2 \overline{T}_3 \overline{T}_4. \qquad (12)$$

Оскільки $R < S_L = 5$, рівняння (12) реалізується на одному елементі *LUT* і блок *LUTerτ* для даного прикладу складається з одного елемента *LUT*.

Для вирішення завдань, пов'язаних з етапом 12, необхідно використовувати стандартні САПР [17]. Цей етап не розглядається в нашому прикладі.

Аналіз запропонованого методу. В основі запропонованого методу лежить заміна блоку *LUTerA* (рис. 3) композицією блоків *LUTer1 – LUTerI* і *LUTerτ*. Природно, така заміна має сенс тільки в тому випадку, якщо вона призводить до зменшення числа елементів *LUT* (і їх рівнів) у схемі КМПК.

Якщо умова (5) порушується для всіх функцій $D_r \in \Phi$, то блок *LUTerA* має тільки *R* елементів *LUT*. Однак ця умова може порушуватися лише для дуже простих алгоритмів керування. Для більшості FPGA [5] маємо $S_L = 6$. В роботі [10] показано, що для практичних прикладів переходи залежать у середньому від трьох логічних умов. Тому умова (5) виконується вже для $R = 3$.

Зазначимо, що запропонований підхід не змінює організації лічильника. Для КМПК U_1 лічильник повинен мати *R* інформаційних входів типу *D*, входи обнулення (Reset), керування операцією *ЛА*:=*ЛА*+1 (C_1) і керування операцією *ЛА*:= $\Phi(C_2)$. Для виконання цих операцій на входи C_1 або C_2 необхідно подати сигнал *Clock*.

Якщо $y_E = 1$, то операції не виконуються ($C_1 = C_2 = 0$). При $y_E = 0$ і $y_0 = 1$, необхідно збільшити вміст лічильника адреси мікрокоманд ($C_1 = 1$). Якщо $y_E = 0$ і $y_0 = 0$, у лічильник адреси мікрокоманд завантажується інформація Ф ($C_2 = 1$). Цей алгоритм визначає систему рівнянь:

$$C_1 = \overline{y_E} y_0 Clock; \quad C_2 = \overline{y_E}\,\overline{y_0} Clock,$$

для реалізації якої потрібні 2 елементи *LUT*. На рис. 5 показана структурна схема лічильника, загальна для КМПК U_1 і U_2.

Запропонований метод не збільшує вимоги до числа блоків *EMB* у порівнянні з їх числом у КМПК U_1. В обох випадках число блоків визначається формулою (4). При цьому, чим менше різниця $R_0 - R_E$, тим менше елементів містить блок *LUTerτ*.

Як показали дослідження [11], подвійне кодування станів дозволяє значно покращити характеристики схем автоматів. Очевидно, такого ж покращення характеристик можна очікувати і при застосуванні подвійної адресації мікрокоманд.

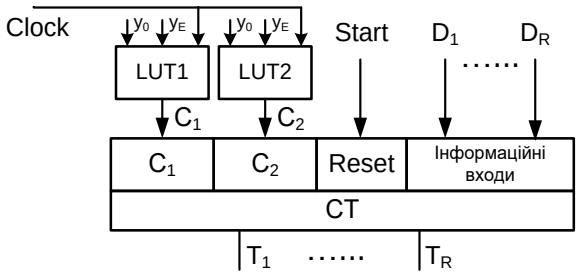

РИС. 5. Структурна схема КМПК U_2

Висновок. Композиційні мікропрограмні пристрої керування містять блоки формування адреси мікрокоманд [4]. Ці блоки реалізують функції переходів відповідного автомата Мура. Отже, для покращення характеристик блоків формування адреси мікрокоманд можна використовувати методи оптимізації автоматів Мура.

У роботі пропонується метод подвійної адресації мікрокоманд. Даний метод – це адаптація метода подвійного кодування станів [11]. Метод розрахований на реалізацію схеми КМПК у базисі FPGA. При цьому блок адресації реалізується на елементах *LUT*, а керуюча пам'ять – на блоках *EMB*.

Метод доцільно використовувати, якщо блок адресації представляється багаторівневою схемою. Схема стає багаторівневою, якщо число елементів у функціях збудження пам'яті перевищує число входів елементів *LUT*. Аналіз бібліотеки [18] і FPGA Virtex-7 [19] показав, що умова (5) виконується для 68 % стандартних прикладів.

Грунтуючись на дослідженнях [11], можна стверджувати, що запропонований метод дозволить зменшити апаратурні витрати (число елементів *LUT* і їх міжпоєднань), час затримки і споживану енергію у порівнянні з КМПК U_1. При цьому, чим складніше алгоритм керування, тим більший виграш дає перехід від КМПК U_1 до КМПК U_2.

У цій роботі ми розглянули найбільш просту модель КМПК, в якій лічильник використовується для зберігання адрес мікрокоманд і кодів станів. Однак, відомі й інші структури КМПК [20, 21]. Тому подальший напрямок наших досліджень пов'язаний із застосуванням запропонованого методу для інших моделей КМПК. Крім того, ми досліджуємо можливість застосування методів оптимізації суміщених автоматів [22 – 24] для оптимізації схем КМПК.

Список літератури

1. Соловьев В.В. Проектирование цифровых схем на основе программируемых логических интегральных схем. М: Горячая линия ТЕЛЕКОМ, 2001. 636 с.
2. DeMicheli G. Synthesis and optimization of digital circuits. New York: McGraw-Hill, 1994. 576 p.
3. Baranov S. Logic synthesis for control automata. Dordrecht: Kluwer Academic Publishers, 1994. 312 p.
4. Баркалов А.А., Титаренко Л.А. Синтез композиционных микропрограммных устройств управления. Харьков: Коллегиум, 2007. 304 с.
5. Maxfield C. The design warrior's guide to FPGAs. Orlando: Academic Press, 2004. 542 p.
6. White paper FPGA architecture. www.altera.com (звернення 10.03.2021)
7. Sklyarov V., Skliarova I., Barkalov A., Titarenko L. Synthesis and optimization of FPGA-based systems. Berlin: Springer, 2014. 432 p. https://doi.org/10.1007/978-3-319-04708-9
8. Grout I. Digital systems design with FPGAs and CPLDs. Amsterdam: Elsevier, 2008. 784 p.
9. Garcia-Vargas I., Senhadji-Navarro R., JimБnez-Moreno G., Civit-Balcells A., Guerra-Gutierrezz P. ROM-based finite state machines implementation in low cost FPGAs. IEEE International Symposium on Industrial Electronics. (ISIE'07) (Vigo, 2007). 2007. P. 2342–2347. https://doi.org/10.1109/ISIE.2007.4374972
10. Skliarova I., Sklyarov V., Sudnitson A. Design of FPGA-based circuits using hierarchical finite state machines. Tallinn: TUT Press, 2012. 240 p.
11. Barkalov A., Titarenko L., Mielcatek K. Hardware reduction for LUT-based Mealy FSMs. *International Journal of Applied Mathematics and Computer Science*. 2018. **28** (3). P. 595–607. https://doi.org/10.2478/amcs-2018-0046
12. Barkalov A., Titarenko L. Logic synthesis for FSM-based control units. Berlin: Springer, 2009. 233 p.
13. Грушницкий Р.И., Мурсаев А.Х., Угрюмов Е.П. Проектирование систем с использованием микросхем программируемой логики. СПб: БХВ-Петербург, 2002. 608 с.
14. Tiwari A., Tomko K. Saving power by mapping finite state machines into embedded memory blocks in FPGAs. Proceedings Design, Automation and Test in Europe Conference and Exhibition. (Paris, France, 6–20 Feb. 2004). 2004 . **2**. P. 916–921. https://doi.org/10.1109/DATE.2004.1269007
15. 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. **34** (5). P. 867–871. https://doi.org/10.1109/TCAD.2015.2406859
16. Rawski M., Selvaraj H., Luba T. An application of functional decomposition in ROM-based FSM implementation in FPGA devices. *Journal of System Architecture*. 2005. **51** (6–7). P. 424–434. https://doi.org/10.1016/j.sysarc.2004.07.004
17. Vivado Design Suite. https://www.xilinx.com/products/design-tools/vivado.html (звернення 10.03.2021)
18. Yang S. Logic synthesis and optimization benchmarks user guide. Version 3.0. Techn. Rep. Microelectronics Center of North Carolina. 1991. 43 p.
19. Virtex-7 FPGAs. https://www.xilinx.com/products/silicon-devices/fpga/virtex-7.html (звернення 10.03.2021)
20. Баркалов А.А., Титаренко Л.А., Ефименко К.Н. Оптимизация схем композиционных микропрограммных устройств управления. *Кибернетика и системный анализ*. 2011. № 1. С. 179–188. http://nbuv.gov.ua/UJRN/KSA_2011_47_1_17
21. Баркалов А.А., Титаренко Л.А. Преобразование кодов в композиционных микропрограммных устройств управления. *Кибернетика и системный анализ*. 2011. № 5. С. 107–118. http://nbuv.gov.ua/UJRN/KSA_2011_47_5_11.
22. Баркалов А.А., Титаренко Л.А., Визор Я.Е., Матвиенко А.В. Оптимальное кодирование состояний в совмещенном автомате. *Управляющие системы и машины*. 2016. № 6. С. 34–39. https://doi.org/10.15407/usim.2016.06
23. Баркалов А.А., Титаренко Л.А., Визор Я.Е., Матвиенко А.В. Синтез совмещенного автомата в базисе ASIC. *Cybernetics and Computer Technologies*. 2020. № 2. С. 78–85. https://doi.org/10.34229/2707-451X.20.2.8
24. Баркалов А.А., Титаренко Л.А., Визор Я.Е., Матвиенко А.В. Уменьшение аппаратурных затрат в совмещенных автоматах. *Управляющие системы и машины*. 2017. № 4. С. 43–50. https://doi.org/10.15407/usim.2017.04.043

Одержано 11.02.2021

Баркалов Олександр Олександрович,
доктор технічних наук, професор Університету Зеленогурського (Польща),
https://orcid.org/0000-0002-4941-3979
A.Barkalov@iie.uz.zgora.pl

Тітаренко Лариса Олександрівна,
доктор технічних наук,професор Університету Зеленогурського (Польща),
професор Харківського національного університету радіоелектроніки,
https://orcid.org/0000-0001-9558-3322
L.Titarenko@iie.uz.zgora.pl

Головін Олександр Миколайович,
кандидат технічних наук, старший науковий співробітник
Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
https://orcid.org/0000-0002-0279-812X
o.m.golovin.1@gmail.com

Матвієнко Олександр Володимирович,
науковий співробітник Інституту кібернетики імені В.М. Глушкова НАН України, Київ.
https://orcid.org/0000-0003-1838-1422
avmatv@ukr.net

UDC 004.274

A.A. Barkalov [1], L.A. Titarenko [1, 2], O.M Golovin [3], A.V. Matvienko [3 *]

Twofold Addressing of Microinstructions in CMCU with Common Memory

[1] *University of Zielona Gora, Poland*
[2] *Kharkiv National University of Radio Electronics, Ukraine*
[3] *V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine, Kyiv*
[*] *Correspondence: avmatv@ukr.net*

Introduction. Control unit (CU) is one of the most important blocks of practically any digital system. Its characteristics largely determine the characteristics of a system as a whole. As a rule, to synthesize CUs, the models of Mealy and Moore finite state machines (FSMs) are used.

The article is devoted to compositional microprogram control units (CMCUs). A CMCU is a Moore FSM in which a state register is replaced by a microinstruction address counter. The choice of CMCU is an optimal solution for implementing linear control algorithms. When developing FSM circuits, it is necessary to optimize such characteristics as the performance and hardware amount. The methods of optimization depend strongly on logic elements used. Nowadays, FPGA chips are one of the most common logic elements for implementing digital systems. To implement the CMCU circuit, it is enough to use look-up table (LUT) elements, programmable flip-flops, embedded memory blocks, and programmable interconnections.

The purpose of the article. In the article, there is proposed a CMCU design method improving such characteristics of CU as the number of logic levels and regularity of programmable interconnections.

The main drawback of LUT is a small number of inputs. Modern digital systems can generate signals of logical conditions entering the control unit, the number of which is tens of times greater than the number of LUT inputs. Such a discrepancy between the characteristics of the control algorithm and the number of inputs of the LUT elements leads to multi-level control circuits with an irregular structure of programmable interconnections, and is the reason for a decrease in performance and an increase in chip area and power consumption.

Results. A method for double addressing of microinstructions in CMCU with shared memory is proposed. The method is an adaptation of the two-fold state assignment of Mealy FSMs, the circuits of which are implemented with FPGAs. The proposed method makes it possible to obtain a microinstruction addressing circuit with two logic levels and a regular interconnection system. The paper considers an example of the synthesis of the CMCU circuit and analyzes the proposed method.

Conclusions. The proposed method allows reducing hardware amount (the number of LUTs and their interconnections), time of delay and power consumption. Moreover, the more complex the control algorithm, the greater the benefit the proposed method gives.

Keywords: compositional microprogram control unit, microinstruction, LUT, EMB, synthesis.