DOI: 10.18372/2310-5461.49.15142
УДК 004.274

О. О. Баркалов, д-р техн. наук, проф.
Зеленогурський університет, Польща
orcid.org/0000-0002-4941-3979
e-mail: A. Barkalov@iie.uz.zgora.pl;

Л. О. Тітаренко, д-р техн. наук, проф.
Зеленогурський університет, Польща
Харківський національний університет радіоелектроніки
orcid.org/0000-0001-9558-3322
e-mail: L.Titarenko@iie.uz.zgora.pl;

О. М. Головін, канд. техн. наук
Інститут кібернетики ім. В.М. Глушкова НАН України
orcid.org/0000-0002-0279-812X
e-mail: o.m.golovin.1@gmail.com;

О. В. Матвієнко
Інститут кібернетики ім. В. М. Глушкова НАН України
orcid.org/0000-0003-1838-1422
e-mail: matv@online.ua;

МЕТОД ЗМЕНШЕННЯ КІЛЬКОСТІ ЕЛЕМЕНТІВ LUT В СХЕМІ КОМПОЗИЦІЙНОГО МІКРОПРОГРАМНОГО ПРИСТРОЮ УПРАВЛІННЯ З ПЕРЕТВОРЮВАЧАМИ АДРЕСИ

Вступ

Одним з головних блоків цифрових систем є пристрій управління [1; 2]. В даний час цифрові системи реалізуються з використанням різних надвеликих інтегральних схем [1]. При цьому важливо зменшувати площу схеми, яку займає пристрій управління [3], а також час затримки і споживану потужність. Як правило, зменшення площі схеми пристрою управління дозволяє поліпшити і інші його характеристики [1; 4], що свідчить про актуальність обраного напрямку досліджень.

Аналіз останніх досліджень і публікацій

Методи розв'язання цієї задачі багато в чому залежать від особливостей реалізованого алгоритму управління і елементного базису. Якщо алгоритм управління має лінійний характер, то пристрій управління доцільно реалізувати у вигляді композиційного мікропрограмного пристрою управління (КМПУ) [5]. Композиційний мікропрограмний пристрій управління є автоматом Мура, що складається зі схеми адресації, лічильника адреси мікрокоманд і пам'яті управління. В пам'яті управління зберігаються мікрокоманди (МК), що складаються з наборів мікрооперацій і спеціальних внутрішніх сигналів управління. Інформація про адреси переходів в МК відсутня, що зменшує необхідний обсяг пам'яті порівняно з традиційними мікропрограмними пристроями управління [5]. В даній статті розглядається задача синтезу схеми КМПУ в базисі надвеликих інтегральних схем типу FPGA (field-programmable logic arrays) [6; 7].

Вибір базису FPGA як об'єкта досліджень зумовлений їх широким застосуванням для реалізації різних цифрових систем [7; 8]. При цьому FPGA можуть використовуватися в мікросхемах типу SoC (System-on-a-Chip) [9], що реалізують складні цифрові системи. Ресурси FPGA в мікросхемах SoC досить обмежені через те, що основна площа кристала зайнята мікропроцесорами та іншими операційними блоками [9]. Зазначене обмеження підвищує актуальність проблеми зменшення площі схем пристрою управління, що використовують ресурси FPGA.

Одним з методів зменшення витрат апаратури в КМПУ є перетворення адрес мікрокоманд в коди виходів лінійних операторних ланцюгів (ЛОЛ) [5]. Таке перетворення дозволяє зменшити число входів схеми адресації мікрокоманд. Це особливо важливо при реалізації схеми пристрою управління в базисі FPGA в силу невеликого числа входів елементів табличного типу LUT (look-up table) [10]. У статті пропонується метод оптимізації схеми КМПУ, заснований на розбитті множини виходів ЛОЛ. Цей метод є адаптацією методу подвійного кодування станів автоматів Мілі [11] до особливостей КМПУ. Для завдання алгоритмів управління використовується мова граф-схем алгоритму (ГСА) [12; 13].

Усе наведене вище визначає ***основну мету*** дослідження, яка полягає у поліпшенні характеристик КМПУ завдяки врахуванню особливостей реалізованого алгоритму управління і елементного базису.

Реалізація КМПУ в базисі FPGA

Синтез КМПУ заснований на формуванні множити ЛОЛ [5].

Операційні ланцюги являють собою послідовності операторних вершин ГСА. Кожна операторна вершина містить набір мікрооперацій $Y_q \subseteq Y$, де $Y = \{y_1, ..., y_N\}$ — множина мікрооперацій (вихідних сигналів КМПУ).

Розглянемо ГСА Γ_1 (рис. 1). Множина вершин ГСА Γ_1 включає початкову вершину b_0, кінцеву вершину b_E, операторні вершини $b_1 - b_{20}$ і умовні вершини $b_{21} - b_{27}$.

В умовних вершинах записуються логічні умови (ЛУ) з множини $X = \{x_1, ..., x_L\}$. Для ГСА Γ_1 маємо $N = 7$ і $L = 6$.

Операційні вершини ГСА утворюють множину $B = \{b_1, ..., b_M\}$. Для ГСА Γ_1 маємо $M = 20$. Розглянемо ряд визначень [5], необхідних для подальшого викладу матеріалу.

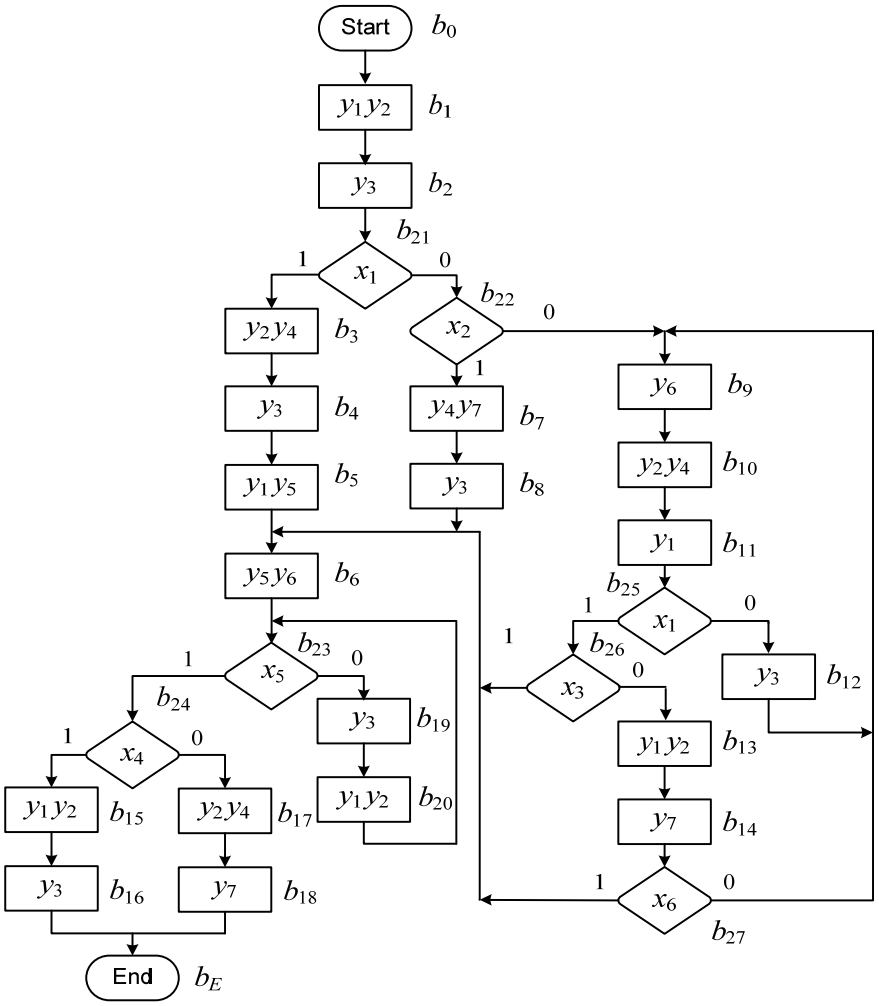

Рис. 1. Первинна ГСА Γ_1

Визначення 1. Кінцева послідовність операторних вершин $\alpha_g = \langle b_{g_1}, ..., b_{g_{F_g}} \rangle$ називається ЛОЛ ГСА Г, якщо для будь-якої пари сусідніх компонент вектора α_g існує дуга, де і — номер компоненти вектора α_g.

Визначення 2. Входом ЛОЛ α_g називається вершина, вхід якої пов'язаний з виходом початкової або умовної вершини, або з виходом операторної вершини, що не входить в ЛОЛ α_g.

Визначення 3. Виходом ЛОЛ α_g називається вершина, вихід якої пов'язаний з входом вершини, що не входить в ЛОЛ α_g.

Пам'ять управління КМПУ містить M_0 мікрокоманд, де $M_0 = M + 1$. Нульова адреса пам'яті управління відповідає початковій вершині b_0 [5]. Кожна операторна вершина $b_m \in B$ відповідає мікрокоманді MI_m з адресою A_m. Розрядність адреси МК визначається за формулою

$$R_Q = \lceil \log_2 Q \rceil. \quad (1)$$

Для реалізації схеми КМПУ необхідно знайти множину ЛОЛ $C = \{\alpha_1, ..., \alpha_G\}$, яка задовольняє таким умовам [5]:

1) кожна операторна вершина включена тільки в одну ЛОЛ;

2) усі операторні вершини включені в ЛОЛ $\alpha_g \in C$;

3) число елементів у множині C є мінімально можливим.

У праці [5] пропонується алгоритм побудови множини C, що задовольняє цим умовам.

До складу КМПУ входить лічильник адреси МК (ЛА), який є аналогом регістра кодів станів автомата Мура [14; 15]. Збільшення вмісту ЛА на одиницю відповідає переходу між МК, які є сусідніми компонентами ЛОЛ $\alpha_g \in C$. Таким чином, всередині ЛОЛ виконується природна адресація мікрокоманд [5].

Для будь-якої пари сусідніх компонент b_{gi}, $b_{gi} + 1$ ЛОЛ $\alpha_g \in C$ повинна виконуватися умова

$$A_{g_{i+1}} = A_{g_i} + 1, \quad (i = \overline{1, F_g - 1}) \quad (2)$$

При цьому мікрокоманда MI_0 має $A_0 = 0$. Це стартова адреса мікропрограми [5], що відповідає ГСА Г.

Нехай для ГСА Г сформовано множину ЛОЛ C і виконана природна адресація мікрокоманд.

Нехай також сформовано множину виходів $O(\Gamma) = \{O_1, ..., O_G\}$, що включає останні компоненти ЛОЛ $\alpha_g \in C$.

Тоді для реалізації алгоритму, представленого ГСА Г, можна використовувати КМПУ U_1 (рис. 2).

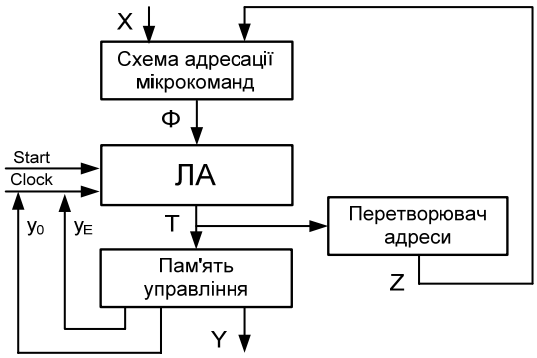

Рис. 2. Структурна діаграма КМПУ U_1
(з перетворювачем адреси мікрокоманд)

Блок адресації мікрокоманд формує множину функцій збудження пам'яті

$$\Phi = \Phi(Z, X). \quad (3)$$

Множина $\Phi = \{D_1, ..., D_R\}$ включає функції збудження пам'яті D_r, що надходять на входи ЛА. Множина Z включає R_A елементів $Z = \{z_1, ..., z_{R_A}\}$. Елементи множини Z використовуються для кодування виходів ЛОЛ кодами $K(O_g)$ розрядності

$$R_A = \lceil \log_2 G \rceil. \quad (4)$$

Коди $K(O_g)$ залежать від внутрішніх змінних $T_r \in T = \{T_1, ..., T_R\}$, що формуються на виході ЛА. Блок перетворювача адреси реалізує систему

$$Z = Z(T). \quad (5)$$

Внутрішні змінні використовуються і для адресації мікрокоманд, що зберігаються в пам'яті управління. Кожна МК може включати дві додаткові змінні. Змінна y_0 дозволяє збільшення вмісту ЛА на одиницю (ЛА: = ЛА + 1), або прийом адреси переходу (ЛА: = Ф). Мінлива $y_E = 1$ відповідає закінченню мікропрограми.

Метод синтезу КМПУ U_1, який називається КМПУ з перетворювачем адреси, детально розглянуто У працях [5; 16]. У цій роботі розглядаємо метод оптимізації характеристик КМПУ U_1 в базисі FPGA. Для реалізації схеми КМПУ використовуються різні блоки CLB (*configurable logic block*) і програмовані з'єднання [15]. У статті розглядаються два види CLB: логічні

елементи (ЛЕ) і вбудовані блоки пам'яті EMB (*embedded memory block*).

Логічний елемент складається з елемента LUT, що має S_L входів, програмованого тригера і мультиплексорів. На вихід ЛЕ може подаватися або комбінаційний вихід LUT, або вихід тригера. Як правило, ЛЕ включає ланцюги перенесення для організації лічильників [6; 10]. Елемент LUT реалізує довільну булеву функцію, яка залежить не більше ніж від S_L аргументів. Для реалізації схеми лічильника КМПУ будемо використовувати двотактні тригери типу D.

Основною особливістю LUT є обмежене число входів S_L (S_L <6) [10]. Відомо, що для реальних алгоритмів управління характерні такі параметри: L ≈ 50, R ≈ 10 [17]. Таким чином, функції (3) можуть мати до 60 аргументів. Така невідповідність між параметрами S_L і R + L призводить до багаторівневим схемами блоків адресації і перетворення адреси.

Відомо, що багаторівневі схеми мають ряд недоліків порівняно з їх одноріневими аналогами [17; 18]. Багаторівневі схеми займають велику площу кристала, мають меншу швидкодію і споживають більше електричної потужності. Для поліпшення характеристик багаторівневих схем необхідно в першу чергу зменшувати число елементів LUT в схемі [19].

Тригери лічильника розподіляються між елементами LUT, що формують функції $D_r \in Ф$. Таким чином, лічильник є розподіленим усередині блоку LUTerA, що реалізує систему (3). Ми використовуємо термін LUTer для схеми, що складається з логічних елементів. Очевидно, ці тригери ЛА мають загальні входи синхронізації (Clock) і обнуління (Start).

Для реалізації пам'яті управління будемо використовувати CLB типу EMB [10]. Блоки EMB мають постійний об'єм (V_0), а розрядність адреси (S_A) і виходів (t_F) осередків пам'яті може змінюватися [6; 7]. Пара <S_A, t_F> визначає конфігурацію EMB.

Блоки EMB доцільно використовувати, якщо існує конфігурація <S_0, t_0>, для якої виконується умова

$$S_0 = R. \tag{6}$$

При S_0 <R пам'ять управління представляється у вигляді багаторівневої схеми. Для сучасних мікросхем FPGA максимальне значення S_A = 15. Цього цілком достатньо для одноріневої реалізації пам'яті управління для складних алгоритмів управління [20].

Будемо розглядати випадок використання унітарних кодів мікрооперацій. З урахуванням розрядів, необхідних для зберігання змінних y_0 і y_E, число блоків EMB в пам'яті управління визначається виразом

$$n_E = \left\lceil \frac{N + 2}{t_0} \right\rceil. \tag{7}$$

Якщо n_E > 1, то будемо позначати пам'ять управління як EMBer.

Нехай елементи LUT використовуються для реалізації блоків адресації і перетворення адреси. Тоді архітектура КМПУ U_1 може бути представлена структурною діаграмою (рис. 3).

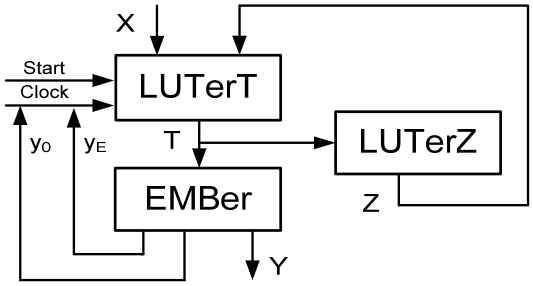

Рис. 3. Реалізація КМПУ U_1 в базисі FPGA

Блок LUTerT реалізує систему (3). Тригери лічильника ЛА розподілені між елементами LUT. Якщо y_E = 0 (виконання алгоритму триває) і y_0 = 1, то до вмісту ЛА додається одиниця. Для цього формується сигнал

$$C_1 = y_0 \overline{y_E} \text{Clock}. \tag{8}$$

Якщо y_E = 0 і y_0 = 0, то в ЛА записується інформація (3). Для цього формується сигнал

$$C_1 = \overline{y_0}\,\overline{y_E} \text{Clock}. \tag{9}$$

Якщо y_E = 1 (закінчення алгоритму), то $C_1 = C_2 = 0$ і функціонування КМПУ припиняється.

Блок LUTerZ реалізує систему (5). Блок EMBer реалізує функції

$$Y = Y(T). \tag{10}$$

$$y_0 = y_0(T). \tag{11}$$

$$y_E = y_E(T). \tag{12}$$

Основний недолік КМПУ U_1 полягає в тому, що при виконанні деяких умов блоки LUTerT і/або LUTerZ можуть мати кілька рівнів логіки. Якщо виконується умова

$$R > S_L, \tag{13}$$

то LUTerZ буде багаторівневим.

Якщо $NA(D_r)$ — число аргументів у функції $D_r \in Ф$, то блок LUTerT є багаторівневим за виконання умови

$$NA(D_r) > S_L. \tag{14}$$

У цій статті розглядається ситуація, коли умови (13)–(14) виконуються.

У цьому випадку нами пропонується КМПУ U_2, засноване на розбитті множини виходів О(Г).

Основна ідея пропонованого методу

Нехай для деякого алгоритму управління виконуються умови (13)–(14). У цьому випадку пропонуємо метод, що дозволяє:

а) гарантовано отримати дворівневу схему блоку LUTerT;

б) виключити блок LUTerZ.

Пропонований метод використовує ідею подвійного кодування станів [11].

Нехай для деякої ГСА Г сформовано множину виходів О(Г), що складається з виходів ЛОЛ $\alpha_g \in C$. Функції (3) необхідно формувати тільки для елементів цієї множини. Отже, виходи $O_g \in O$ (Г) є аналогами станів автомата Мура.

Адреси $A(O_g)$ виходів $O_g \in O$ (Г) аналогічні кодам станів автомата Мура. На відміну від кодів станів, адреси не можуть бути призначені довільно. Адреса виходу $A(O_g)$ залежить від адреси входу даної ЛОЛ $\alpha_g \in C$.

Знайдемо розбиття Π_0 множини О(Г) на класи $O^1, ..., O^I$, таке, що:

а) число класів є мінімальним;

б) для класу $O^i \in \Pi_0$ виконується умова

$$R_i + L_i \le S_L \quad (i \in \{1,\ldots,I\}). \qquad (15)$$

У виразі (15) символом R_i позначено число внутрішніх змінних $z_r \in Z$, необхідних для кодування виходів $O_g \in O^i$. Число вхідних змінних $x_l \in X$, що визначають переходи з виходів $O_g \in O^i$, позначено символом L_i.

Якщо $|O^i| = M_i$, то число кодуючих змінних визначається формулою

$$R_i = \left\lceil \log_2(M_i + 1) \right\rceil \quad (i \in \{1,\ldots,I\}). \qquad (16)$$

Одиниця у виразі (16) додається для кодування відносини $O_g \notin O^i$.

Адреси виходів ЛОЛ зберігаються в лічильнику і кодуються змінними $T_r \in T$. Для кодування множини $O^i \in \Pi_0$ використовуємо змінні $z_r \in Z$. Число елементів R_0 у множині Z визначається сумою потужностей множин Z_i (16):

$$R_0 = \sum_{i=1}^{I} R_i . \qquad (17)$$

Отже, множина $Z^i \subseteq Z$ включає додаткові змінні $z_r \in Z$, що кодують виходи $O_g \in O^i$. Крім того, кожен клас $O^i \in \Pi_0$ визначає множини $X^i \subseteq X$ і $\Phi^i \subseteq \Phi$. Множина $X^i \subseteq X$ включає ЛУ $x_l \in X$, що визначають переходи з виходів $O_g \in O^i$.

Множина $\Phi^i \subseteq \Phi$ включає змінні $D_r \in \Phi$, які дорівнюють одиниці на переходах з виходів $O_g \in O^i$. Змінні $z_r \in Z^i$ і $x_l \in X^i$ визначають функції збудження пам'яті

$$\Phi^i = \Phi^i(Z^i, X^i). \qquad (18)$$

Система (18) визначає блок LUTeri ($i \in \{1, ..., I\}$).

Кожен блок LUTeri визначає часткові функції $D_r^i = \Phi^i$. Результуюча система функції збудження пам'яті визначається диз'юнкціями:

$$D_r = \bigvee_{i-1}^{I} D_r^i \ (r \in \{1,...,R\}). \qquad (19)$$

Функції (19) реалізуються блоком LUTerT, до складу якого входить розподілений лічильник ЛА.

Для реалізації систем (10)–(12) використовується блок, що складається з n_E вбудованих блоків пам'яті. Нехай умова (6) виконується для деякої конфігурації $<s_0, t_0>$. Тоді блок EMBer складається із n_E блоків, де n_E визначається формулою (7). Блок EMBer має $n_E \times t = t_E$ виходів.

Нехай виконується така умова:

$$t_E \ge 2 + N + R_0. \qquad (20)$$

Тоді блок EMBer має достатньо виходів для представлення функцій (5). У цій статті ми пропонуємо структурну схему КМПУ U_2 (рис. 4).

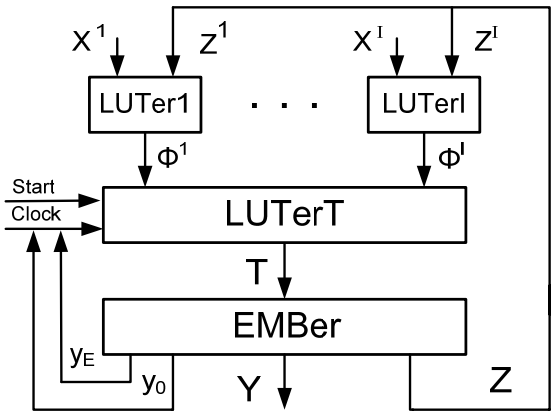

Рис. 4. Структурна схема КМПК U2

КМПУ U_2 містить три рівні логіки. Блоки LUTer1 — LUTerI і LUTerT реалізують схему адресації мікрокоманд. Блок EMBer реалізує функції (10)–(12), а також систему булевих функцій (5).

Запропонований в нашій статті метод синтезу КМПУ U_2 включає такі етапи:

1. Формування множини ЛОЛ для ГСА Г.

2. Виконання природної адресації мікрокоманд.

3. Формування множини виходів О(Г).

4. Формування розбиття Π_0, що задовольняє умову (15).

5. Кодування виходів $O_g \in O(\Gamma)$ кодами $K(O_g)$.

6. Формування таблиць блоків LUTer1-LUTerI.

7. Формування систем функцій (18).

8. Формування систем функцій (19).

9. Формування таблиці блоку EMBer.

10. Реалізація схеми КМПУ в заданому базисі.

Нехай символ $U_i(\Gamma_j)$ означає, що алгоритм управління, представлений ГСА Γ_j, реалізується з використанням моделі КМПУ U_i.

Розглянемо приклад синтезу КМПУ U_2 (Γ_1) для випадку $S_L = 5$.

Приклад синтезу КМПУ U_2

Використовуємо метод із праці [5] для формування множини $C=\{\alpha_1,...,\alpha_G\}$. Застосування цього методу в розглянутому прикладі дає множина $C = \{\alpha_1,...,\alpha_8\}$, де $\alpha_1 = < b_0, b_1, b_2 >$, $\alpha_2 = < b_3, b_4, b_5, b_6 >$, $\alpha_3 = < b_7, b_8 >$, $\alpha_4 = < b_{12}, b_9, b_{10}, b_{11} >$, $\alpha_5 = < b_{13}, b_{14} >$, $\alpha_6 = < b_{15}, b_{16} >$, $\alpha_7 = < b_{17}, b_{18} >$, $\alpha_8 = < b_{19}, b_{20} >$. Отже, $G = 8$ для ГСА Γ_1.

Для ГСА Γ_1 маємо $M_0 = 21$. Застосовуючи вираз (1), отримаємо R=5, що дають множини $T = \{T_1,...,T_5\}$ та $\Phi = \{D_1,...,D_5\}$. Для виконання природної адресації МК можна застосовувати методи [5; 18]. Один з можливих варіантів показано на рис. 5.

T_4T_5 \ $T_1T_2T_3$	000	001	010	011	100	101	110	111
00	b_0	b_{20}^+	b_{17}	*	b_6^+	*	b_{11}^+	*
01	b_1	b_{15}	b_{18}	b_3	b_7	b_{12}	b_{13}	*
10	b_2^+	b_{16}^+	*	b_4	b_8^+	b_9	b_{14}^+	*
11	b_{19}	*	*	b_5	*	b_{10}	*	*

Рис. 5. Адреси мікрокоманд КМПУ U_2 (Γ_1)

Як видно з визначення (3), виходи ЛОЛ є останніми компонентами векторів $\alpha_g \in C$. Отже, множина $O(\Gamma)$ формується тривіальним чином. У нашому прикладі отримано множину $O(\Gamma_1) = \{b_2, b_6, b_8, b_{11}, b_{14}, b_{16}, b_{18}, b_{20}\}$.

Для формування множини Π_0 можна використовувати метод із праці [11]. Цей метод дозволяє мінімізувати кількість блоків $O^i \in \Pi_0$. У кращому випадку виконується умова

$$I \le S_L. \qquad (21)$$

Якщо умова (21) порушується, то блок LUTerT має кілька рівнів елементів LUT.

Відзначимо, що пошук розбиття Π_0 виконується на множині $O'(\Gamma) \subseteq O(\Gamma)$. Якщо вершина $b_q = O_g$ та існує дуга $< b_q, b_E >$, то $O_g \notin O'(\Gamma)$.

Таким чином, множина $O'(\Gamma)$ містить тільки виходи, із яких немає безумовних переходів в вершину b_E.

У разі Γ_1, маємо множину $O'(\Gamma) = \{O_1, O_2, O_3, O_4, O_5, O_8\}$, де $O_1 = b_2$, $O_2 = b_6$, $O_3 = b_8$, $O_4 = b_{11}$, $O_5 = b_{14}$ і $O_8 = b_{20}$. Застосування методу [11] дозволяє отримати розбиття $\Pi_0 = \{O^1, O^2\}$, де $O^1 = \{O_1, O_3, O_4\}$ та $O^2 = \{O_2, O_5, O_8\}$.

Аналіз ГСА Γ_1 і множин O^1, O^2 дозволяє отримати множини $X^1 = \{x_1, x_2, x_3\}$ і $X^2 = \{x_4, x_5, x_6\}$. Оскільки $M_1 = M_2 = 3$, то з виразу (16) маємо $R_1 = R_2 = 2$. З (17) можемо отримати $R_0 = 4$, що дає множину $Z = \{z_1, ..., z_4\}$. Нехай $Z^1 = \{z_1, z_2\}$ і $Z^2 = \{z_3, z_4\}$.

Очевидно, що умова (15) виконується для класів O^1 і O^2, тому коди $K(O_g)$ не впливають на число елементів LUT в блоках LUTer1–LUTerI. Залишимо код O_0 для співвідношення $O_g \notin O^i$.

Закодуємо інші виходи так:

$K(O_1) = K(O_2) = 01$;

$K(O_3) = K(O_5) = 10$;

$K(O_4) = K(O_8) = 11$.

Для формування таблиць блоків LUTer1 і LUTer2, побудуємо системи формул переходу [15] для виходів $O_g \in O'(\Gamma_1)$:

$$O_1 = x_1 b_3 \vee \overline{x_1} x_2 b_7 \vee \overline{x_1 x_2} b_9;$$

$$O_3 = b_6;$$

$$O_4 = x_1 x_3 b_6 \vee x_1 \overline{x_3} b_{13} \vee \overline{x_1} b_{12}. \qquad (22)$$

$$O_2 = x_5 x_4 b_{15} \vee x_5 \overline{x_4} b_{17} \vee \overline{x_5} b_{19};$$

$$O_5 = x_6 b_6 \vee \overline{x_6} b_9;$$

$$O_8 = x_5 x_4 b_{15} \vee x_5 \overline{x_4} b_{17} \vee \overline{x_5} b_{19}. \qquad (23)$$

Таблиці блоків LUTerI мають такі стовпці: O_g — вихід ЛОЛ відповідної формули переходу; $K(O_g)$ — код виходу; b_q — вершина ГСА, в яку відбувається перехід; A_q — адреса МК, відповідної b_q; X_h — кон'юнкція ЛУ $x_l \in X$, що визначає перехід із O_g у b_q; Φ_h — набір функцій збудження пам'яті, рівних одиниці при формуванні з ЛА адреси A_q; h — номер переходу.

Із системи (22) маємо таблицю блоку LUTer1, з системи (23) — таблицю блоку LUTer2. Адреси мікрокоманд беруть із карти Карно (рис. 5). Кон'юнкції X_h йдуть безпосередньо з систем (22)–(23). Блок LUTer1 поданий у табл. 1, блок LUTer2 у табл. 2.

Зв'язок між цими таблицями, адресами мікрокоманд і кодами виходів очевидний.

Таблиця 1

Таблиця блоку LUTer1

O_1	01	b_3	01101	x_1	$D_2 D_3 D_5$	1
		b_7	10001	$\overline{x_1}x_2$	$D_1 D_5$	2
		b_9	10110	$\overline{x_1}\,\overline{x_2}$	$D_1 D_3 D_4$	3
O_3	10	b_6	10000	1	D_1	4
O_4	11	b_6	10000	x_1x_3	D_1	5
		b_{13}	11001	$x_1\overline{x_3}$	$D_1 D_2 D_5$	6
		b_{12}	10101	$\overline{x_1}$	$D_1 D_3 D_3$	7

Таблиця 2

Таблиця блоку LUTer2

O_g	$K(O_g)$	b_q	A_q	X_h	Φ_h	h
O_2	01	b_{15}	00101	x_5x_4	$D_3 D_5$	1
		b_{17}	01000	$x_5\overline{x_4}$	D_2	2
		b_{19}	00011	$\overline{x_5}$	$D_4 D_5$	3
O_5	10	b_6	10000	x_6	D_1	4
		b_9	10110	$\overline{x_6}$	$D_1 D_3 D_4$	5
O_8	11	b_{15}	00101	x_5x_4	$D_3 D_5$	6
		b_{17}	01000	$x_5\overline{x_4}$	D_2	7
		b_{19}	00011	$\overline{x_5}$	$D_4 D_5$	8

Системи (18) формуються на основі табл. 1 і табл. 2.

Із табл. 1 (з урахуванням мінімізації) отримуємо систему булевих функцій:

$$\begin{aligned}
&D_1^1=\overline{z_1}z_2\,\overline{x_1}\vee z_1;\ D_2^1=\overline{z_1}z_2x_1\vee z_1z_2x_1\overline{x_3};\\
&D_3^1=z_2x_1\vee\overline{z_1}z_2\,\overline{x_2};\\
&D_4^1=\overline{z_1}z_2\,\overline{x_1}\,\overline{x_2};\\
&D_5^1=\overline{z_1}z_2x_1\vee\overline{z_1}z_2x_2\vee z_1z_2\overline{x_3}\vee z_1z_2\overline{x_1}.
\end{aligned}\qquad (24)$$

Із табл. 2 маємо таку систему булевих функцій:

$$\begin{aligned}
&D_1^2=z_3\overline{z_4};\ D_2^2=z_4x_5\overline{x_4};\\
&D_3^2=z_4x_5x_4;\\
&D_4^2=z_4\overline{x_5}\vee z_3\overline{z_4}\,\overline{x_6};\ D_5^2=z_4x_4\vee z_4\overline{x_5}.
\end{aligned}\qquad (25)$$

Система (19) формується тривіальним чином. У розглянутому прикладі всі функції $D_r\in\Phi$ мають вигляд

$$D_r=D_r^1\vee D_r^2. \qquad (26)$$

Нехай для реалізації блоку EMBer використовуються блоки EMB з конфігурацією $S_A = 5$, $t_F = 8$. Оскільки $S_A = R$, EMB доцільно використовувати для реалізації пам'яті управління КМПУ.

Очевидно, для реалізації блоку EMBer необхідно

$$n_{EMB}=\left\lceil\frac{2+N+R_0}{t_F}\right\rceil. \qquad (27)$$

У нашому випадку $N = 7$, $R_0 = 4$, $t_F = 8$, що дає $n_{EMB} = 2$.

Ці два блоки мають загальні входи $T_1 – T_5$.

Кожен рядок таблиці блоку EMBer відповідає одній МК. Таблиця має M_0 рядків, що містять таку інформацію:

$$MI_m, A_m, y_0, y_E, Y, Z, m.$$

Для нашого прикладу блок EMBer представлений у табл. 3. Таблиця блоку EMBer заповнюється так:

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

2. Якщо вершина b_q не є виходом ЛОЛ, то в комірку пам'яті для відповідної МК необхідно записати таку інформацію:

$$y_0 = 1 \text{ і } y_E = 0.$$

3. Якщо вихід ЛОЛ пов'язаний з входом вершини b_E, то $y_E = 1$.

4. Якщо вихід ЛОЛ $\alpha_g \in C$ не пов'язаний з вершиною b_E, то в стовпчику Z записується код $K(O_g)$. Крок 10 запропонованого методу пов'язаний з використанням САПР типу Vivado [21] і не розглядається в нашому прикладі.

Таблиця 3

Таблиця блоку EMBer КМПУ U_2 ($Г_1$)

MI_m	A_m	y_O	y_E	Y							Z				m
				y_1	y_2	y_3	y_4	y_5	y_6	y_7	z_1	z_2	z_3	z_4	
MI_0	00000	1	0	0	0	0	0	0	0	0	0	0	0	0	1
MI_1	00001	1	0	1	1	0	0	0	0	0	0	0	0	0	2
MI_2	00010	0	0	0	0	1	0	0	0	0	0	1	0	0	3
MI_{19}	00011	1	0	0	0	1	0	0	0	0	0	0	0	0	4
MI_{20}	00100	0	0	1	1	0	0	0	0	0	0	0	1	1	5
MI_{15}	00101	1	0	1	1	0	0	0	0	0	0	0	0	0	6
MI_{16}	00110	0	1	0	0	1	0	0	0	0	0	0	0	0	7
MI_{17}	01000	1	0	0	1	0	1	0	0	0	0	0	0	0	8
MI_{18}	01001	0	1	0	0	0	0	0	0	1	0	0	0	0	9
MI_3	01101	1	0	0	1	0	1	0	0	0	0	0	0	0	10
MI_4	01110	1	0	0	0	1	0	0	0	0	0	0	0	0	11
MI_5	01111	1	0	1	0	0	0	1	0	0	0	0	0	0	12
MI_6	10000	0	0	0	0	0	0	1	1	0	0	0	0	0	13
MI_7	10001	1	0	0	0	0	1	0	0	1	0	0	0	0	14
MI_8	10010	0	0	0	0	1	0	0	0	0	1	0	0	0	15
MI_{12}	10101	1	0	0	0	1	0	0	0	0	0	0	0	0	16
MI_9	10110	1	0	0	0	0	0	0	1	0	0	0	0	0	17
MI_{10}	10111	1	0	0	1	0	1	0	0	0	0	0	0	0	18
MI_{11}	11000	0	0	1	0	0	0	0	0	0	1	1	0	0	19
MI_{13}	11001	1	0	1	1	0	0	0	0	0	0	0	0	0	20
MI_{14}	11010	0	0	0	0	0	0	0	0	1	0	0	1	0	21

Висновки

Методи синтезу схем КМПУ багато в чому залежать від особливостей елементного базису. При реалізації схеми КМПУ в базисі FPGA доцільно використовувати два типи логічних блоків CLB: блоки типу LUT і блоки типу EMB. Елементи LUT використовуються для реалізації нерегулярної схеми адресації мікрокоманд. Блоки пам'яті EMB є найкращим засобом для реалізації керуючої пам'яті. У даній роботі пропонується метод оптимізації схеми КМПУ з перетворювачем адреси мікрокоманд. Цей метод можна розглядати як адаптацію методу подвійного кодування станів [11] до особливостей КМПУ.

Метод доцільно використовувати, якщо блок адресації мікрокоманд представляється багаторівневої схемою. Це можливо, якщо число аргументів у функціях $D_r \in Ф$ перевищує число входів елементів LUT. Якщо розрядність адреси МК перевищує число входів S_L, то схема блоку перетворювача адреси також є багаторівневою. Аналіз стандартних автоматів з бібліотеки [22] і базису Virtex 7 [23] показав, що цей метод можна застосовувати для 68% елементів бібліотеки.

Крім того, ми запропонували використовувати EMB для реалізації перетворювача адрес. Це дозволяє зменшити число елементів LUT і з'єднань. На підставі досліджень [11] можна стверджувати, що запропонований підхід дозволить значно поліпшити характеристики КМПУ порівняно з КМПУ U_1. Існує дуже багато різних моделей КМПУ [5; 16; 18].

Кожна з моделей підходить для реалізації керуючих алгоритмів з певними характеристиками.

У своїх подальших дослідженнях ми плануємо адаптувати запропонований підхід до інших моделей КМПУ. Другим напрямком наших досліджень є застосування методів оптимізації суміщених автоматів [24, 25, 26] для поліпшення характеристик КМПУ.

ЛІТЕРАТУРА

[1] Czerwinski R., Kania D. Finite state machines logic synthesis for complex programmable logic devices. Berlin: Springer, 2013. 172 pp.

[2] Skliarova I., Sklyarov V., Sudnitson A. Design of FPGA-based circuits using hierarchical finite state machines. Tallinn: TUT Press, 2012. 240 pp.

[3] DeMicheli G. Synthesis and optimization of digital circuits. New York: McGraw-Hill, 1994. 576 pp.

[4] 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. pp. 916–921.

[5] Баркалов А.А., Титаренко Л.А. Синтез композиционных микропрограммных устройств управления. Харьков: Коллегиум, 2007. 304 с.

[6] Maxfield C. The design warrior's guide to FPGAs. Orlando: Academic Press, 2004. 542 pp.

[7] Grout I. Digital systems design with FPGAs and CPLDs. Amsterdam: Elsevier, 2008. 784 p.

[8] Грушницкий Р.И., Мурсаев А.Х., Угрюмов Е.П. Проектирование систем с использованием микросхем программируемой логики. СПб: БХВ-Петербург, 2002. 608 с.

[9] Garcia-Vargas I., Senhadji-Navarro R., Jimenez-Moreno G., Civit-Balcells A., Guerra-Gutierrez P. ROM-based finite state machines implementation in low-cost FPGAs. *IEEE Intern. Simp. on Industrial Electronics* (ISIE'07) (Vigo, 2007). 2007. P. 2342–2347.

[10] White paper FPGA architecture. URL: www.altera.com. (data access 25.09.2020)

[11] 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. Pp. 424–434.

[12] Baranov S. Logic synthesis for control automata. Dordrecht: Kluwer Academic Publishers, 1994. 312 pp.

[13] Sklyarov V., Skliarova I., Barkalov A., Titarenko L. Synthesis and optimization of FPGA-based systems. Berlin: Springer, 2014. 432 pp.

[14] 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. Pp. 1–21.

[15] Barkalov A., Titarenko L. Logic synthesis for FSM-based control units. Berlin: Springer, 2009. 233 pp.

[16] Баркалов А. А., Титаренко Л. А. Преобразование кодов в композиционных микропрограммных устройствах управления. *Кибернетика и системный анализ*. 2011. № 5. С. 107–118.

[17] Соловьев В. В. Проектирование цифровых схем на основе программируемых логических интегральных схем. Москва: Горячая линия — ТЕЛЕКОМ, 2001. 636 с.

[18] Баркалов А. А., Титаренко Л. А., Ефименко К. Н. Оптимизация схем композиционных микропрограммных устройств управления. *Кибернетика и системный анализ*. 2011. № 1. С. 179–188.

[19] 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.

[20] Kolopienczyk M., Titarenko L., Barkalov A. Design of EMB-based Moore FSMs. *Journal of Circuits, Systems and Computers*. 2017. Vol. 21, N 7. P. 1–23.

[21] Vivado Design Suite. https://www.xilinx.-com/products/design-tools/vivado.html. (data access 25.09.2020)

[22] Yang S. Logic synthesis and optimization benchmarks user guide. Version 3.0. Techn. Rep. Microelectronics Center of North Carolina, 1991. 43 p.

[23] Virtex-7 FPGAs. https://www.xilinx.com/products/silicon-devices/fpga/virtex-7.html. (data access 25.09.2020)

[24] 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.

[25] Баркалов А. А., Титаренко Л. А., Визор Я. Е., Матвиенко А. В. Уменьшение числа LUT элементов в схеме совмещенного автомата. *Управляющие системы и машины*. 2016. № 3. С. 16–22.

[26] 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.

Баркалов О. О., Титаренко Л. О., Головін О. М., Матвієнко О. В.

МЕТОД ЗМЕНШЕННЯ КІЛЬКОСТІ ЕЛЕМЕНТІВ LUT В СХЕМІ КОМПОЗИЦІЙНОГО МІКРОПРОГРАМНОГО ПРИСТРОЮ УПРАВЛІННЯ З ПЕРЕТВОРЮВАЧАМИ АДРЕСИ

У сучасних цифрових системах, які реалізуються з використанням різноманітних надвеликих інтегральних схем, важливим питанням залишається зменшування площі схеми, яку займає пристрій управління, а відтак і зниження часу затримки і обсягів споживаної потужності. Як правило, зменшення площі схеми пристрою управління дозволяє поліпшити і інші його характеристики. Методи розв'язання цієї задачі в деякій мірі залежать від особливостей реалізованого алгоритму управління і елементного базису.

В статті пропонується метод зменшення числа елементів LUT (Look-Up-Table) в схемі композиційного мікропрограмного пристрою управління (КМПУ), заснований на перетворенні адрес виходів лінійних операторних ланцюгів (ЛОЛ) в коди виходів. Таке перетворення дозволяє зменшити число входів схеми адресації мікрокоманд, що особливо важливо при реалізації схеми пристрою управління в базисі FPGA в силу невеликого числа входів елементів табличного типу LUT. Виходи ЛОЛ кодуються як елементи деяких класів розбиття множини виходів. Цей підхід дозволяє перейти від багаторівневої схеми адресації мікрокоманд до дворівневої. Пам'ять управління і блок перетворення адреси реалізуються на вбудованих блоках пам'яті. Запропонований метод є адаптацією методу подвійного кодування станів автоматів Мілі до особливостей КМПУ. Метод доцільно використовувати, якщо блок адресації мікрокоманд представляється багаторівневою схемою. Це можливо, якщо число аргументів у функціях збудження пам'яті перевищує число входів елементів LUT. Якщо розрядність адреси мікрокоманди перевищує число входів LUT, то схема блоку перетворювача адреси також є багаторівневою. Аналіз стандартних автоматів з бібліотеки і базису Virtex 7 показав, що цей метод можна застосовувати для 68% елементів бібліотеки. В статті розглянуто приклад синтезу схеми КМПУ із застосуванням запропонованого методу.

Ключові слова: композиційний микропрограмний пристрій управління; LUT; EMB; синтез.

Barkalov A. A., Titarenko L. A., Golovin O. M., Matvienko A. V.

METHOD OF REDUCING THE NUMBER OF LUT ELEMENTS IN THE SCHEME OF A COMPOSITION MICROPROGRAM CONTROL UNIT WITH ADDRESS TRANSFORMERS

In advance digital systems, implemented using various VLSI circuits, an important issue remains to reduce the area of the circuit, which is occupied by the control device, and, consequently, reduce the delay time and the amount of power consumption. As a rule, reducing the area of the circuit of the control device allows improving its other characteristics. Methods for solving this problem to some extent depend on the features of the implemented control algorithm and element basis.

The article proposes a method for reducing the number of LUT (Look-Up-Table) elements in the scheme of a composition microprogram control unit (CMCU), based on converting the addresses of the outputs of linear operator circuits (LOC) into output codes. This transformation reduces the number of inputs of the addressing scheme of microinstructions, which is especially important when implementing the circuit of the control device on the FPGA basis due to the small number of inputs of elements of the tabular type LUT. The outputs of the LOC are encoded as elements of some classes of partitioning the set of outputs. This approach allows you to move from a multi-level microinstruction addressing scheme to a two-level one. The control memory and the address conversion block are implemented on embedded memory blocks. The proposed method is an adaptation of the method of double coding of states of Miles automata to the features of CMCU. The method is advisable to use if the microinstruction addressing unit is represented by a multi-level scheme. This is possible if the number of arguments in the memory excitation functions exceeds the number of LUT inputs. If the bit width of the microinstruction address exceeds the number of LUT inputs, then the address converter block circuit is also multilevel. Analysis of standard automata from the library and the Virtex 7 basis showed that this method can be applied to 68% of the library elements. The article considers an example of the synthesis of a CMCU circuit using the proposed method.

Keywords: Composition microprogram control unit; LUT; EMB; synthesis.

Стаття надійшла до редакції 27.01.2021 р.
Прийнято до друку 24.02.2021 р.