doi.org/10.15407/usim.2019.05.012
УДК 004.274

О.О. БАРКАЛОВ, д-р техн. наук, проф., Університет Зеленогурський (Польща),
вул. Підгірна, 50, Зелена Гура, 65246, Польща,
a.barkalov@imei.uz.zgora.pl

Л.О. ТІТАРЕНКО, д-р техн. наук, проф., Університет Зеленогурський (Польща),
вул. Підгірна, 50, Зелена Гура, 65246, Польща,
L.Titarenko@iie.uz.zgora.pl, D_ts@nure.ua

Я.Є. ВІЗОР, канд. техн. наук, ст. наук. співробітник,
Ін-т кібернетики ім. В.М. Глушкова НАН України,
03187, м. Київ, просп. Акад. Глушкова, 40, Україна,
yaviz@ukr.neta

О.В. МАТВІЄНКО, наук. співробітник,
Ін-т кібернетики ім. В.М. Глушкова НАН України,
03187, м. Київ, просп. Акад. Глушкова, 40, Україна,
matv@online.ua

СИНТЕЗ ЧОТИРИРІВНЕВОЇ СХЕМИ СУМІЩЕНОГО АВТОМАТА

Запропоновано метод зменшення апаратурних витрат у схемі суміщеного автомата, що реалізовується в спільному базисі елементів LUT і блоків пам'яті EMB. Метод заснований на заміні логічних умов і розбитті множин логічних станів на класи. Кожен клас відповідає окремому блоку схеми. Такий підхід призводить до схем з регулярною структурою і чотирма логічними рівнями. Наведено приклад використання запропонованого методу. Показано умови його застосування.

Ключові слова: *суміщений автомат, синтез, LUT, EMB, розбиття, FPGA.*

Вступ

На сьогодні цифрові системи є невід'ємною частиною сучасної цивілізації [1] Це пов'язано з широким використанням вбудованих, автономних і мобільних пристроїв [1, 2]. Важливою частиною подібних систем є пристрій керування (ПК) [3, 4]. Ефективність ПК багато в чому визначає ефективність системи загалом. І тут важливе значення має зменшення витрат апаратури у схемі ПК. Вирішння цієї проблеми дає змогу зменшити енергоспоживання і час затримки схеми ПК.

Методи вирішення цього завдання багато в чому залежать від моделі, яка задає поведінку ПК, та елементного базису для реалізації схеми ПК [5, 6]. У цій статті ми розглядаємо задачу реалізації схеми суміщеного мікропрограмного автомата (СМПА) [7, 8] в базисі програмованих логічних інтегральних схем *FPGA* (*field-programmable logic arrays*) [9, 10]. Вибір моделі зумовлений її широким використанням для реалізації ПК сучасних цифрових систем [11]. Вибір базису зумовлений тим, що *FPGA* використовуються в багатьох цифрових системах [12, 13]. При цьому вважається, що вони

домінуватимуть у цій галузі найближчими десятиліттями [14].

Пропонований нами метод засновано на використанні структурної декомпозиції [13] схем СМПА. Для зменшення загальних витрат апаратури ми використовуємо й елементи табличного типу *LUT* (*look-up table*), й вбудовані блоки пам'яті *EMB* (*embedded memory blocks*). Для запровадження закону функціонування ПК ми використовуємо мову граф-схеми алгоритму (ГСА) [14].

Реалізація схеми СМПА в базисі FPGA

Модель СМПА об'єднує в собі риси автоматів Мілі і Мура [7, 8]. Множина станів $A = \{a_1,..., a_M\}$ формується, як для атомата Мура [11]. Множина вихідних змінних (мікрооперацій) Y є об'єднанням множин Y_A (мікрооперації автомата Мілі) і Y_B (мікрооперації автомата Мура). Мікрооперації $y_n \in Y_A$ записуються біля дуг ГСА, а мікрооперації $y_n \in Y_B$ — в операційних вершинах ГСА [11].

Для синтезу схеми СМПА необхідно побудувати пряму структурну таблицю (ПСТ). Цьому етапу передує кодування станів, коли кожному стану $a_m \in A$ ставиться у відповідність двійковий код $K(a_m)$ розрядності R:

$$R = |\log_2 M. \quad (1)$$

Для кодування станів використовуються внутрішні змінні, що утворюють множину $T = \{T_1,...,T_R\}$. Коди $K(a_m)$ зберігаються в спеціальному регістрі (*RG*), що складається з R двотактних тригерів. Ми розглядаємо випадок, коли ці тригери мають інформаційні входи типу D [6]. Для зміни вмісту *RG* використовуються функції збудження пам'яті, що утворюють множину $\Phi = \{D_1,..., D_R\}$.

У ПСТ є такі стовпці [11]: a_m — початковий стан СМПА; $K(a_m)$ — код стану $a_m \in A$; a_s — стан переходу; $K(a_S)$ — код стану $a_s \in A$; X_h — вхідний сигнал, що визначає перехід з $< a_m, a_S >$ і дорівнює кон'юнкції вхідних змінних із множини $X = \{x_1,..., x_L\}$; Y_{Ah} — набір мікрооперацій (НМО) автомата Мілі, що формується на переході $< a_m, a_S >$; Φ_h — набір функцій $D_r \in \Phi$, що дорівнюють одиниці для запису в *RG* коду $K(a_S)$; h — номер переходу ($h = \overline{1, H}$). У стовпці a_m записуються НМО автомата Мура, що формуються в цьому стані. Метод формування ПСТ щодо ГСА Г розглянуто, наприклад, в [11].

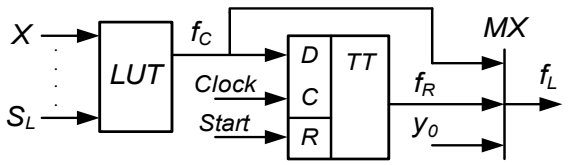

Рис. 1. Логічний елемент блоку LUTer

Схема СМПА задається системами булевих функцій (СБФ), що формуються на основі ПСТ:

$$\Phi = \Phi(T, X); \quad (2)$$

$$Y_A = Y_A(T, X); \quad (3)$$

$$Y_B = Y_B(T). \quad (4)$$

Кожна із систем (2)–(4) відповідає окремому блоку у схемі СМПА [7, 8].

У цій статті ми розглядаємо задачу реалізації систем (2)–(4) в базисі *FPGA*. Для реалізації схеми ми використовуватимемо елементи *LUT* і *EMB*. Розгляньмо головні характеристики елементного базису.

Нехай *LUTer* означає блок, що складається з елементів *LUT*, програмованих тригерів і міжз'єднань. Таким чином, кожен *LUTer* складається з логічних елементів, наведених на рис. 1.

Елемент *LUT* реалізує довільну логічну функцію f_C, яка залежить від не більше, ніж S_L аргументів. Вихід f_C подається на вхід D двотактного тригера. Імпульс *Start* використовується для операції $f_R := 0$. Імпульс *Clock* ініціює операцію $f_R = f_C$. Вихід логічного елемента може бути або комбінаційним ($f_L = f_C$), або реєстровим ($f_L = f_R$). Для вибору типу виходу використовується мультиплексор MX і внутрішній сигнал y0: $f_L = y_0 f_C \vee \overline{y_0} f_R$.

Системи (2)–(4) визначають однорівневу схему СМПА [7]. Схема складається з блоків *LUTerA* (реалізує систему (3)), *LUTerB* (реалізує систему (4)) *і LUTerT* (реалізує систему (2)). Блок *LUTerT* включає регістр *RG*, розпо-

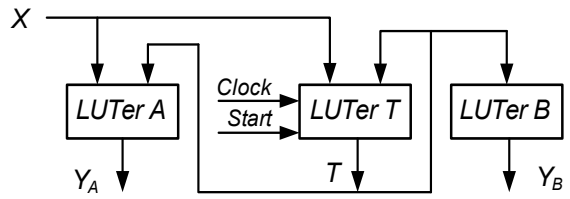

Рис.2. Однорівнева схема СМПА в базисі логічних елементів

ділений між логічними елементами. Тому блок має виходи $T_r \in T$ і входи *Clock* і *Start* (рис. 2). Позначмо цей СМПА символом U_1.

Сучасні *FPGA* мають значне число блоків *EMB* [12, 13]. Характерною рисою *EMB* є можливість зміни числа адресних входів (S_A) і розрядності комірок пам'яті (t_F) при збереженні постійної ємності (V_0):

$$V_0 = 2^{S_A} \times t_F. \quad (5)$$

Кожна пара $<S_A, t_F>$, яка задовольнить (5), визначає конфігурацію *EM*B. Для сучасних *FPGA* характерні наступні конфігурації блоків *EMB*: <15, 1>, <14, 2>, ..., <9, 64>. Таким чином, зменшення S_A на одиницю подвоює число виходів блоку t_F.

Наявність різних конфігурацій дає змогу узгодити параметри СМПА і блоку *EMB*. Нехай $|Y_A|=N_A$, $|Y_B|=N_B$, $|Y|=N=N_A+N_B$. Нехай виконується така умова:

$$2^{L+R} \times (N+R) \le V_0 . \quad (6)$$

При цьому СМПА реалізується у вигляді одного блоку *EMB*. Аналіз бібліотеки стандартних автоматів [16] показав, що умова (6) має місце для 68 відсотків всіх прикладів.

Оптимізація схеми СМПА в базисі FPGA

Елементи *LUT* мають вкрай обмежене число входів ($S_L \le 6$) [15, 16]. Аналіз [16] показує, що для більшості автоматів дотримується умова:

$$L+R>S_L. \quad (7)$$

У цьому разі схеми блоків *LUTerA* і *LUTerT* мають більше одного рівня елементів. Це призводить до нерегулярних схем із великим числом міжз'єднань. Це призводить до збільшення паразитних ємностей, відповідальних за подовження часу поширення сигналів і збільшення споживаної енергії [17]. При цьому для реалізації схеми використовуються складні методи функціональної декомпозиції [14].

Якщо умова (6) виконується, то для реалізації схеми СМПА потрібно кілька блоків *EMB*. Це призводить до значної надлишковості схеми. Крім того, завдання взагалі не має розв'язку, якщо виконується умова

$$2^{L+R} > V_0. \quad (8)$$

Аналіз бібліотеки [16] показав, що умова (8) виконується для 24 відсотків тестових прикладів. При цьому блоки *EMB* інтенсивно використовуються для реалізації різних блоків цифрових систем [18]. Як наслідок, розробник ПК може мати тільки обмежене число блоків *EMB*.

За виконання умов (7)–(8) найкращим рішенням є спільне використання елементів *LUT* і блоків *EMB* для реалізації схеми СМПА [7, 8]. Цей підхід пов'язано із застосуванням методів структурної декомпозиції [14]. При цьому найчастіше використовується метод заміни логічних умов (ЗЛУ) $x_l X$ [14].

Нехай $X(a_m)$ — множина логічних умов (ЛУ), яка визначає переходи зі стану $a_m \in A$. Знайдімо значення параметра G:

$$G = \max (|x(a_1)|, \dots, |x(a_M)|). \quad (9)$$

Параметр G визначає множину додаткових змінних $P = \{p_1,..., p_G\}$, що замінює множину X. Як показали дослідження [6], для реальних ГСА $G \le 3$.

Метод ЗЛУ передбачає побудову таблиці ЗЛУ, рядки якої позначено змінними $p_g \in P$, а стовпці — станами $a_m \in A$. Якщо змінна $p_g = x_l$ для стану $a_m \in A$, то логічна умова (ЛУ) $x_l \in X$ записується на перетині рядка p_g і стовпця a_m. Ця таблиця є основою для формування СБФ:

$$P = P(T, X). \quad (10)$$

Використовуючи функції (10), системи (2)–(3) перетворюються до наступного вигляду:

$$\Phi = \Phi(T, P); \quad (11)$$

$$Y_A = Y_A(T, P). \quad (12)$$

Система (10) визначає блок ЗЛУ в схемі автомата. Як правило, для реалізації цього блоку використовуються елементи *LUT*, а інша частина схеми реалізується на блоці *EMB* [19]. Для цього має виконуватися умова:

$$2^{G+R} \times (N + R) \le V_0. \quad (13)$$

У цій статті ми розглядаємо випадок, коли умова (13) порушується. При цьому нехай виконується така умова:

$$2^{L+R} \times G \le V_0. \quad (14)$$

У цьому разі блок ЗЛУ можна реалізувати у вигляді одного блоку *EMB*. Решту блоків схеми реалізовано на елементах *LUT*. Це веде до моделі U_2 (рис. 3).

В автоматі U_2 блок *EMB* реалізує СБФ (10), а блок *LUTer*1 — системи (11)–(12). Блок *LUTer*2 є аналогічним до блоку *LUTerB* автомата U_1.

Аналіз бібліотеки [16] показує, що для ряду прикладів виконується умова:

$$G+R>S_L. \quad (15)$$

Для $S_L = 5$ (*Virtex*-5) умова (15) виконується для 28 відсотків тестових прикладів, для $S_L = 6$ (*Virtex*-6) — для 12 відсотків. При цьому схема блоку *LUTer*1 має кілька рівнів логіки.

Якщо виконується умова:

$$R>S_L, \quad (16)$$

то схеми блоків *LUTer*1 і *LUTer*2 є багаторівневими. Пропонований нами метод орієнтовано на автомати, що задовольняють умови (14)–(16).

Головна ідея пропонованого методу

Нехай для деякого СМПА знайдено множину A, виконано кодування станів і здійснено заміну ЛУ $x_l \in X$ додатковими змінними $p_g \in P$. Нехай для реалізації схеми використовуються блоки *EMB*, що задовольняють (13) і (14), і елементи *LUT*, що задовольняють (15)–(16). Знайдімо розбиття $\Pi_A = \{A^1, \ldots, A^K\}$ множини A на класи, що задовольняють умову:

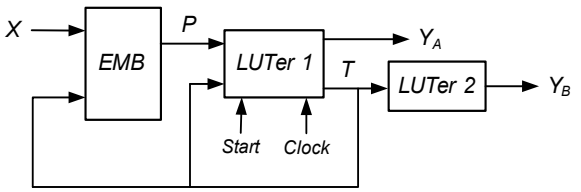

Рис. 3. Структурна схема СМПА U_2

$$R_k + G_k \le S_L \qquad (k = \overline{1, K}). \quad (17)$$

У (17) символ R_k означає число змінних $\tau_\rho \in \tau$, необхідних для кодування станів $a_m \in A^k$ кодами $C(a_m)$. Параметр G_k дорівнює числу змінних $p_g \in P$, що визначають переходи зі станів $a_m \in A^k$.

Параметр R_k визначається формулою

$$R_k = \lceil \log_2(|A^k| + 1) \rceil, \ (k = \overline{1, K}). \quad (18)$$

Одиниця в (18) додається для обліку відношення $a_m \notin A^k$. Параметр R_k визначає потужність множини $\tau^k \subseteq \tau$, де

$$|\tau| = R_A = R_1 + R_2 + \ldots + R_K. \quad (19)$$

Кожен клас $A^k \in \Pi_A$ визначає множини $P^k \subseteq P$, $\Phi^k \subseteq \Phi$, $Y_A^k \subseteq Y$. Множині P^k належать змінні $p_g \in P$, що визначають переходи з $a_m \in A^k$. Множина Φ^k включає функції $D_r \in \Phi$, рівні одиниці в кодах станів $a_S \in A$, таких, що є переходи $< a_m, a_S >$ і $a_m \in A^k$. Множина Y_A^k включає мікрооперації $y_n \in Y_A$, що формуються на переходах $a_m \in A^k$.

Шукатимемо розбиття Π_A, для якого виконуються умови:

$$| P^i \cap P^j | \to \min; \quad (20)$$

$$| A_T^i \cap A_T^J | \to \min; \quad (21)$$

$$| Y_T^i \cap Y_T^J | \to \min. \quad (22)$$

У виразах (20)–(21) для індексів i і j виконуються співвідношення: $i, j \in \{1, \ldots, K\}$, $i \ne j$. Символ A_T^k означає множину станів переходу зі станів класу $A^k \subseteq A$. Для розв'язання цього завдання можна використовувати метод із робіт [20, 21].

Ґрунтуючись на цих положеннях, ми пропонуємо структурну схему СМПА U_3. Ця схема має чотири рівні логіки (рис. 4).

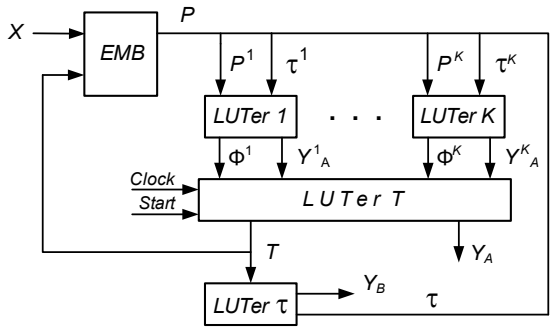

Рис. 4. Структурна схема СМПА U_3

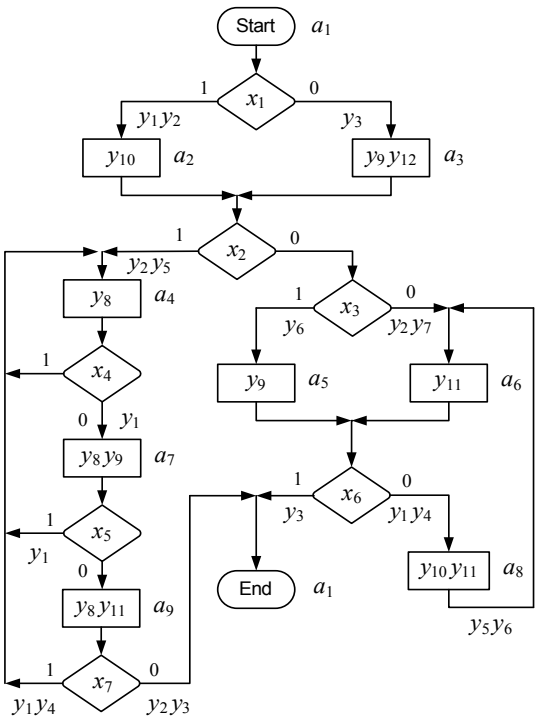

Рис. 5. Початкова ГСА Γ_1

T_3T_4 \ T_1T_2	00	01	11	10
00	a_1	a_4	*	a_2
01	a_5	a_7	*	a_3
11	*	*	*	*
10	a_6	a_9	*	a_8

Рис. 6. Коди станів автомата $U_3(\Gamma_1)$

У СМПА U_3 блок *EMB* реалізує систему (10), блоки *LUTer* 1 – *LUTer K* реалізують СБФ

$$\Phi^k = \Phi^k(\tau^k, P^k); \quad (23)$$

$$Y_A^k = Y_A^k(\tau^k, P^k). \quad (24)$$

Блок *LUTer T* формує мікрооперації y_n Y_A і внутрішні змінні $T_r \in T$, що є виходами тригерів розподіленого регістра. Блок *LUTer* реалізує систему (4) і СБФ

$$\tau = \tau(T). \quad (25)$$

У статті пропонується метод синтезу схеми СМПА U_3 за вихідною ГСА Г. Метод включає такі етапи:

1. Формування множини станів A.
2. Кодування станів $a_m \in A$, які оптимізують СБФ (4).
3. Заміна логічних умов $x_l \in X$ змінними $p_g \in P$.
4. Знаходження розбиття Π_A, що задовольняє умови (17), (20)–(22).
5. Кодування станів $a_m \in A^k$ кодами $C(a_m)$.
6. Формування таблиці *EMB*.
7. Формування систем функцій (23)–(24).
8. Формування функцій блоку *LUTer T*.
9. Формування функцій (4) і (25).
10. Реалізація схеми СМПА в заданому базисі.

Приклад синтезу автомата U_3

Нехай вираз $U_i(\Gamma_j)$ означає, що автомат U_i реалізується за ГСА Γ_j. Розглянемо приклад синтезу СМПА $U_3(\Gamma_1)$, де ГСА Γ_1 приведено на рис. 5. Із рис. 5 можна знайти множни: $A = \{a_1,..., a_9\}$, $X = \{x_1,..., x_7\}$, $Y_A = \{y_1,..., y_7\}$, $Y_B = \{y_{10},..., y_{15}\}$. Це дає такі параметри: $M = 9$, $L = 7$, $N_A = 7$, $N_B = 5$, $N = 12$.

Використовуючи (1), отримаємо $R = 4$. Це дає множини $T = \{T_1, ..., T_4\}$ і $\Phi = \{D_1, ... , D_4\}$. Використовуючи алгоритм з [20], закодуймо стани $a_m \in A$ (рис.6). Як буде видно з подальшого тексту, коди з рис. 6 дозволяють мінімізувати число літералів у СБФ (4).

Аналіз ГСА $_1$ показує, що $G = 2$. Це дає множину $P = \{p_1, p_2\}$. Виконаймо ЗЛУ так, як це показано в табл. 1.

З табл.1 маємо систему (26):

$$p_1 = A_1x_1 \vee (A_2 \vee A_3)x_2 \vee A_4x_4 \vee \vee (A_5 \vee A_6)x_6A_7x_5 \vee A_9x_7 \quad (26)$$
$$p_2 = x_3$$

Нехай елементи *LUT* мають S_L=4 входи. До класу A^k можуть входити до

$$NS_k = 2^{S-L_k} - 1, (k = \overline{1,K}) \quad (27)$$

станів автомата. Це дає наступні пари:
<G_k, NS_k>, <0, 15>, <1, 7>, <2, 3>, <3, 1>.

Використовуючи метод [24, 25], знайдімо наступне розбиття $\Pi_A = \{A^1, A^2\}$ з класами A^1={a_1, a_4, ... , a_7, a_9}, A^2 = {a_2, a_3, a_8}. Це дає множини P^1 = {p_1}, P^2 = {p_1, p_2}, Y_A^1 = {y_1, y_2, y_3, y_4}, Y_A^2 ={y_2, y_5, y_6, y_7}, A_T^1 = {a_2, a_3, a_4, a_7, a_8, a_9}, A_T^2={a_5, a_6}. При цьому $P^1 \cap P^2$={p_1}, $Y_A^1 \cap Y_A^2$={y_2}, $A_T^1 \cap A_T^2 = \varnothing$. Таким чином, для отриманого розбиття виконуються умови (17), (20) — (21).

Закодуймо стани $a_m \in A^k$ кодами $C(a_m)$. Із (18)–(19) маємо R_1=3, R_2=2, R_A=5. Це дає змогу отримати множини τ^1={τ_1, τ_2, τ_3}, τ^2={τ_4, τ_5}, τ={τ_1,...,τ_5}. Один із варіантів кодування станів показано в табл. 2.

Таблиця блоку *EMB* включає стовпці *T, X* (адреса комірки пам'яті), *P* (вміст комірки), *q* – номер рядка таблиці. Таблиця будується тривіально [18]. Нехай серед конфігурацій *EMB* є конфігурація <10, 1>. Оскільки p_2=x_3, для реалізації функції p_2 не потрібно ресурсів блоку *EMB*. Із формули p_1 (система (26) випливає, що на входи *EMB* мають надходити 10 змінних (T_1–T_4 і x_1, x_2, x_4, ... , x_7). Таким чином, для реалізації схеми ЗЛУ в цьому прикладі достатньо одного блока *EMB*.

Для формування СБФ (23)–(24) необхідно побудувати таблиці блоків *LUTer k*. Ці таблиці включають стовпці a_m, $C(a_m)$, a_s, $K(a_S)$, P_h^k, Y_{Ah}^k, Φ_h^k, h. Коди $K(a_s)$ беруться з рис. 6, коди $C(a_m)$ — з табл. 2. Для нашого прикладу *LUTer* 1 представлено табл. 3, а *LUTer* 2 — табл. 4.

Пояснімо принцип заповнення таблиці блоків *LUTer k*. Розгляньмо рядок 1 табл. 3, який задає перехід a_1 в a_2 під дією x_1=1 (це випливає

Таблиця 1. **ЗЛУ автомата $U_3(\Gamma_1)$**

p_g \ a_m	a_1	a_2	a_3	a_4	A_5	a_6	a_7	a_8	a_9
p_1	x_1	x_2	x_3	x_4	X_5	x_6	x_5	–	x_7
p_2	–	x_3	x_3	–	–	–	–	–	–

Таблиця 2. **Коди станів a_m A^k**

a_m	$\tau_1\tau_2\tau_3$	$\tau_4\tau_5$	a_m	$\tau_1\tau_2\tau_3$	$\tau_4\tau_5$	a_m	$\tau_1\tau_2\tau_3$	$\tau_4\tau_5$
a_1	001	00	a_4	010	00	a_7	101	00
a_2	000	01	a_5	011	00	a_8	000	11
a_3	000	10	a_6	100	00	a_9	110	00

Таблиця 3. ***LUTer*1**

a_m	$C(a_m)$	a_s	$K(a_s)$	P_h^1	Y_{Ah}^1	Φ_h^1	h
a_1	001	a_2	0100	p_1	y_1y_2	D_1	1
		a_3	1001	$\overline{p_1}$	y_3	D_1D_4	2
a_4	010	a_2	1000	p_1	y_2y_3	D_1	3
		a_7	0101	$\overline{p_1}$	y_1	D_2D_4	4
a_5	011	a_1	0000	p_1	y_3	-	5
		a_8	1010	$\overline{p_1}$	y_1y_4	D_1D_3	6
a_6	100	a_1	0000	p_1	y_3	-	7
		a_8	1010	$\overline{p_1}$	y_1y_4	D_1D_1	8
a_7	101	a_2	1000	p_1	y_1	D_1	9
		a_9	0110	$\overline{p_1}$	-	D_2D_3	10
a_9	111	a_2	1000	p_1	y_1y_4	D_1	11
		a_1	0000	$\overline{p_1}$	y_2y_3	-	12

Таблиця 4. ***LUTer*2**

a_m	$C(a_m)$	a_s	$K(a_s)$	P_h^2	Y_{Ah}^2	Φ_h^2	h
a_2	01	a_4	0100	p_1	y_2y_5	D_2	1
		a_6	0001	$\overline{p_1}p_2$	y_6	D_4	2
		a_7	0010	$\overline{p_1}\,\overline{p_2}$	y_2y_7	D_3	3
a_2	10	a_4	0100	p_1	y_2y_5	D_2	4
		a_6	0001	$\overline{p_1}p_2$	y_6	D_4	5
		a_7	0010	$\overline{p_1}\,\overline{p_2}$	y_2y_7	D_3	6
a_2	11	a_2	0010	1	y_5y_6	D_3	7

з ГСА Γ_1). З табл. 1 випливає, що $p_1=x_1$. Отже, у стовпці P_h^1 рядка 1 записано змінну p_1. На цьому переході формується набір $y_1 y_2$. Цей набір записано в рядку 1 стовпця Y_{Ah}^1 . З табл. 2 маємо $C(a_1)$=001, з рис. 6 — $K(a_2)$=1000. Оскільки T_1=1, в стовпці Φ_h^1 рядка 1 записано D_1. Решта рядків таблиці заповнюються в аналогічний спосіб.

Із табл. 3 можна отримати функції (23)–(24) з індексом 1. Наприклад, можна отримати такі функції:

$$y_2^1 = \overline{\tau_1}\overline{\tau_2}\tau_3 p_1 \vee \overline{\tau_1}\tau_2\overline{\tau_3}p_1 \vee \tau_1\tau_2\overline{\tau_3}\,\overline{p_1};$$
$$D_2^1 = \overline{\tau_1}\tau_2\overline{\tau_3}\,\overline{p_1} \vee \tau_1\overline{\tau_2}\tau_3\overline{p_1}. \quad (28)$$

Із табл. 2 можна отримати функції (23)–(24) з індексом 2. Наприклад, можна отримати функції:

$$y_2^2 = \overline{\tau_4}\tau_5(p_1 \vee \overline{p_2}) \vee \tau_4\overline{\tau_5}(p_1 \vee \overline{p_2}); \quad (29)$$
$$D_2^2 = \overline{\tau_4}\tau_5 p_1 \vee \tau_4\overline{\tau_5}p_1.$$

Функції блоку *LUTerT* формуються на основі функцій (23)–(24). Це є тривіальним процесом і в нашому прикладі дає наступні рівняння:

$$y_1 = y_2^1; y_1 = y_2^1 \vee y_2^2; y_3 = y_3^1; y_4 = y_4^1;$$
$$y_5 = y_5^2; y_6 = y_6^2; y_7 = y_7^2; y_1 = y_2^1; \quad (30)$$
$$D_1 = D_1^1; D_2 = D_2^1 \vee D_2^2;$$
$$D_3 = D_3^1 \vee D_3^2; D_4 = D_4^1 \vee D_4^2. \quad (31)$$

Як випливає із системи (30) для реалізації системи (3) в нашому прикладі потрібен тільки один *LUT* для реалізації схеми блоку *LUTerT*. Решта мікрооперацій $y_n \in Y_A$ формуються блоками *LUTer* 1 і *LUTer* 2. Зменшення числа *LUT* пов'язано з виконанням умови (7).

Як випливає з (31) для реалізації системи (2) досить 3 елементи *LUT*. Функція D_1 формується блоком *LUTer* 1.

Для формування системи (4) необхідно:

1. Побудувати систему $y_n = f_n(A_m)$, де $y_n \in Y_B$, A_m — кон'юнкція змінних $T_r \in T$, що відповідає коду $K(a_m)$.

2. Виконати мінімізацію системи, врахувавши коди стану автомата.

Система $y_n = f_n(A_m)$ будується за ГСА Г. Для цього виконується аналіз вмісту операторних вершин. У нашому прикладі маємо наступну СБФ:

$$y_8 = A_4 \vee A_7 \vee A_9 = T_2;$$
$$y_9 = A_3 \vee A_5 \vee A_7 = T_4;$$
$$y_{10} = A_2 \vee A_8 = T_1\overline{T}_4; \quad (32)$$
$$y_{11} = A_6 \vee A_8 \vee A_9 = T_3;$$
$$y_{12} = A_3 = T_1 T_4.$$

Аналіз системи (32) показує, що функції y_8, y_9, y_{11} формуються як виходи блоку *LUTerT*. Тому для реалізації системи (4) в цьому прикладі достатньо тільки два елементи *LUT*.

Система (25) формується на основі аналізу кодів $K(a_m)$ і $C(a_m)$. Наприклад, змінна τ_1=1 в кодах $C(a_6)$, $C(a_7)$ і $C(a_9)$ (табл.2). Отже, функція τ_1 визначається як $A_6 \vee A_7 \vee A_9$. Використовуючи коди $K(a_m)$ з рис. 6, отримаємо $\tau_1 = \overline{T}_1 T_3 \vee T_2 T_4$. В аналогічний спосіб можна отримати інші рівняння системи (25). Зазначмо, що стан $a_m \in A$ можна закодувати так, щоб мінімізувати число літералів у системах (4) і (25). Однак ми не розглядаємо цю задачу.

Для реалізації схеми в базисі *FPGA* необхідно використовувати стандартні промислові пакети [9, 10]. Отримані бітові потоки (*bit streams*) заносяться в елементи схеми за допомогою програматорів. У статті не розглядається цей етап для нашого прикладу.

Висновок

Запропонований метод засновано на спільному використанні блоку пам'яті *EMB* та елементів *LUT*. Метод може бути використано, якщо: 1) виконується умова (14) і 2) існує розбиття Π_A, що задовольняє умови (17) і

$$K \le S_L. \quad (33)$$

Якщо при цьому виконується умова

$$R \le S_L, \quad (34)$$

то схема СМПА U_3 має чотири структурних рівні, кожен із яких включає тільки один рівень логіки. Якщо умова (34) порушується, тобто справедливою є умова (16), то число рівнів у блоці *LUTer* можна зменшити завдяки кодуванню станів $a_m \in A$.

За виконання умов (21)–(22) частина функцій $y_n \in Y_A$ і $D_r \in \Phi$ реалізується на блоках *LUTer k*. При цьому число елементів *LUT* в блоці *LUTerT* буде меншим, ніж N_1+R.

Запропонована модель веде до схем із регулярними зв'язками. Так, змінні $x_i \in X$ пов'язано тільки з блоком *EMB,* змінні $p_g \in P$ і $\tau_r \in \tau$ — тільки з блоками *LUTer k*. Це спрощує завдання розміщення та трасування [10] під час реалізації схеми СМПА. Позитивною рисою U_3 є те, що сигнали *Clock* і *Start* пов'язано тільки з одним блоком схеми. Це дає змогу уникнути проблем, пов'язаних із так званим перекосом синхронізації [13].

Аналіз бібліотеки [16] показав, що запропонований метод доцільно використовувати для 68 відсотків тестових прикладів. Водночас для 76 відсотків прикладів може використовуватися й модель U_2. Для 24 відсотків прикладів виконується умова (8) і може використовуватися тільки модель U_3. Наші дослідження з використанням *FPGA Virtex*-6 [10] показали, що мірою зростання параметрів *R, L, N* збільшується виграш щодо апаратури, швидкодії та споживаної енергії для автоматів U_3 у порівнянні з U_2.

Подальші напрямки наших досліджень пов'язано з розробкою подібних структур СМПА з використанням: кодування наборів мікрооперацій автомата Мілі; кодування термів систем (2)–(3) і перетворенням кодів псевдоеквівалентних станів автомата Мура.

ЛІТЕРАТУРА

1. *Gajski D., Abdi S.*, *Gerstlager A.*, *Schirner G.* Embedded System Design. New York: Springer, 2009, 416 p.

2. *Marwedel P.* Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems. Berlin: Springer, 2018, 259 p.

3. *Баркалов А.А., Титаренко Л.А., Визор Я.Е., Матвиенко А.В., Горина В.В.* Уменьшение числа LUT элементов в схеме совмещенного автомата. УСиМ. 2016, №3. С. 16–22.

4. *Баркалов А.А., Титаренко Л.А., Визор Я.Е., Матвиенко А.В.* Реализация схемы совмещенного микропрограммного автомата в базисе FPGA. Проблеми інформатизації та управління: Зб. наук. праць. Нац. авіаційний ун-т. Київ, 2015. Вип. 3(51). С. 5–13.

5. *Соловьев В.В.* Проектирование цифровых схем на основе программируемых логических интегральных схем. М.: Горячая линия – ТЕЛЕКОМ, 2001. 636 с.

6. *Skliarova I., Sklyarov V., Sudnitson A.* Design of FPGA–based circuits using Hierarchical Finite State Machines. Tallinn: TUT Press, 2012. 240 p.

7. *Баркалов А.А., Титаренко Л.А., Визор Я.Е., Матвиенко А.В.* Структурная редукция в совмещенных автоматах. Проблеми інформатизації та управління: Зб. наук. праць. Нац. авіаційний ун-т. Київ, 2017 Вип. 1-2 (57-58). С. 14–19.

8. *Баркалов А.А., Титаренко Л.А., Визор Я.Е., Матвиенко А.В.* Кодирование выходных переменных в совмещенном автомате. Комп'ютерні засоби, мережі та системи. К.: Ін-т кібернетики ім. В.М.Глушкова НАН України, 2018. №17. С. 73–80.

9. *www.altera.com*.

10. *www.xilinx.com*

11. *Baranov S.* Logic Synthesis for Control Automata. Dordrecht: Kluwer Academic Publishers, 1994. 312 p.

12. *Grout I.* Digital Systems Design with FPGAs and CPLDs. Amsterdam: Elseveit, 2008. 784 p.

13. *C. Maxfield C.* The Design Warrior's Guide to FPGAs. – Orlando: Academic Press, 2004. 542 p.

14. *Rawski, M.*, *Tomaszewicz P., Borowski G.*, *uba T.* (2011). Logic Synthesis Method of Digital Circuits Designed for Implementation with Embedded Memory Blocks on FPGAs, *Design of Digital Systems and Dev44ises. LNEE* 70, Springer, Berlin. P. 121–144.

15. *Wisniewski R.*, *Gomes L.*, *Costa A.* Dynamic Partial Reconfiguration of Concurrent Control Systems Implemented in FPGA Devices.-IEEE Transactions on industrial informatics. 2017, V. 13, № 4. pp. 1734–1741.

16. *Yang S.* Logic Synthesis and optimization benchmarks user guide. Microelectronics Center of North Carolina. 1991, 43 p.

17. *Garcia–Vargas L.*, *Senhadji–Navarro R.M.*, *Civit–Balcells* A., *Guerra–Gutierrezz P.* (2007). ROM–Based Finite State Machine Implementation in Low Cost FPGAs, *IEEE International Simposium on Industrial Electronics*, Vigo, pp. 2342–2347.

18. *Barkalo A., Titarenko L.* Logic Synthesis for FSM – based Control Units. Berlin: Springer, 2009. 233 p.

19. *Das N.*, *Ptija P.* FPGA Implementation of Reconfigurable FSMs with Input Multiplexing Architecture using Hungarian Method.-International Journal of Reconfigurable Computing. 2018, 14 (5), pp. 1–15

20. *Barkalov A*, *Titarenko L.*, *Mielcatek K.* Twofold state assignment for FPGA – based Mealy FSMs. In: Proceedings of International Conference MOCAST-18. Thessaloniki, Greece. New York, IEEE Explore, 2018, pp. 1–4.

21. *Barkalov A*, *Titarenko L.*, *Mielcatek K.* Hardware Reduction for LUT-based Mealy FSMs. Int. J. of Applied Mathematics and Computer Science. 2018, 28 (3), pp. 28–41.

Надійшла 12.09.2019

REFERENCES

1. Gajski, D., Abdi, S., Gerstlager, A., Schirner, G., 2009. Embedded System Design, New York: Springer, 416 p.

2. Marwedel, P., 2018. Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems, Berlin: Springer, 259 p.

3. Barkalov, A.A., Titarenko, L.A., Vizor, Ya.Ye., Matvienko, A.V., Gorina, V.V., 2016. "Synthesis of Combined Finite State Machine with FPGAs", Upr. sist. mas., 3, pp. 16–22. (In Russian).

4. Barkalov, A.A., Titarenko, L.A., Vizor, Ya.Ye., Matviyenko, A.V., 2015. "Realizatsiya skhemy sovmeshchennogo mikroprogrammnogo avtomata v bazise FPGA", Problemy informatyzatsiyi ta upravlinnya: Zb. nauk. prats. Natsionalnyy aviatsiynyy universytet, Kyiv, 3 (51), pp. 5–13. (In Russian).

5. Solovyev, V.V., 2001. Proyektirovaniye tsifrovykh skhem na osnove programmiruyemykh logicheskikh integralnykh skhem. M.: Goryachaya liniya – TELEKOM, 636 p. (In Russian).

6. Skliarova, I., Sklyarov, V., Sudnitson, A., 2012. Design of FPGA–based circuits using Hierarchical Finite State Machines, Tallinn: TUT Press, 240 p.

7. Barkalov, A.A., Titarenko, L.A., Vizor, Ya.Ye., Matviyenko, A.V., 2017. "Strukturnaya reduktsiya v sovmeshchennykh avtomatakh", Problemy informatyzatsiyi ta upravlinnya: Zb. nauk. prats. Natsionalnyy aviatsiynyy universytet, Kyiv, 1 (57-58). pp. 12–19 (In Russian).

8. Barkalov, A.A., Titarenko, L.A., Vizor, Ya.Ye., Matviyenko, A.V., 2018. "Kodirovaniye vykhodnykh peremennykh v sovmeshchennom avtomate", Komputerni zasoby, merezhi ta systemy, K.: In-t kibernetyky im. V.M.Hlushkova NAN Ukrayiny, 17, pp. 73–80.

9. Intel FPGas and Programmable Device, [online]. Available at: <www.altera.com> [Accessed 05 Jan. 2019].

10. Xilinx, [online]. Available at: <www.xilinx.com> [Accessed 10 Feb. 2019].

11. Baranov, S., 1994. Logic Synthesis for Control Automata, Dordrecht: Kluwer Academic Publishers, 312 p.

12. Grout, I., 2008. Digital Systems Design with FPGAs and CPLDs, Amsterdam: Elseveit, 784 p.

13. Maxfield, C., 2004. The Design Warrior's Guide to FPGAs, Orlando: Academic Press, 542 p.

14. Rawski, M., Tomaszewicz, P., Borowski, G. and uba, T., 2011. "Logic Synthesis Method of Digital Circuits Designed for Implementation with Embedded Memory Blocks on FPGAs, Design of Digital Systems and Devises", LNEE 70, Springer, Berlin, pp. 121–144.

15. Wisniewski, R., Gomes, L., Costa, A., 2017. "Dynamic Partial Reconfiguration of Concurrent Control Systems Implemented in FPGA Devices", IEEE Transactions on industrial informatics, 13 (4), pp. 1734–1741.

16. Yang, S., 1991. Logic Synthesis and optimization benchmarks user guide, Microelectronics Center of North Carolina, 43 p.

17. Garcia–Vargas, L., Senhadji–Navarro, R., M. Civit–Balcells, A. and Guerra–Gutierrezz, P., 2007. "ROM–Based Finite State Machine Implementation in Low Cost FPGAs", IEEE International Simposium on Industrial Electronics, Vigo., pp. 2342–2347.

18. Barkalov, A., Titarenko, L., 2009. Logic Synthesis for FSM – based Control Units, Berlin: Springer, 233 p.

19. Das, N., Ptija, P., 2018. "FPGA Implementation of Reconfigurable FSMs with Input Multiplexing Architecture using Hungarian Method", International Journal of Reconfigurable Computing, 14 (5). pp. 1–15.

20. Barkalov, A., Titarenko, L., Mielcatek, K., 2018. "Twofold state assignment for FPGA – based Mealy FSMs". In: Proceedings of International Conference MOCAST-18, Thessaloniki, Greece, New York, IEEE Explore, pp. 1–4.

21. Barkalov, A., Titarenko, L., Mielcatek, K., 2018. "Hardware Reduction for LUT – based Mealy FSMs", International Journal of Applied Mathematics and Computer Science, 28 (3), pp. 28–41.

Received 12.09.2019

A.A. Barkalov, Doctor in Techn. Sciences, Professor, University of Zielona Gora (Poland),
Podgorna str., 50, Zielona Gora, 65246, Poland,
a.barkalov@iie.uz.zgora.pl

L.A. Titarenko, Doctor of Technical Sciences, Professor of the University of Zelenogorsk (Poland),
Podgorna str., 50, Zielona Gora, 65246, Poland
L.Titarenko@iie.uz.zgora.pl, D_ts@nure.ua

Y.E. Visor, Ph.D., Senior Researcher, Institute of Cybernetics of NAS of Ukraine,
03187, Kiev, Glushkov Avenue, 40, Ukraine,
yaviz@ukr.net

O.V. Matvienko, Researcher Associate, Institute of Cybernetics of NAS of Ukraine,
03187, Kiev, Glushkov Avenue, 40, Ukraine,
matv@online.ua

SYNTHESIS OF A FOUR-LEVEL SCHEMA OF A COMBINED AUTOMATON

Introduction. A method is proposed targeting hardware decrease in the circuit of combined automation, implemented with LUTs and EMBs. The method is based on replacement of logical conditions and partition of the set of states by classes. Each class corresponds to a single block of the circuit. This approach leads to circuits with regular structure and four levels of logic.

Purpose. The proposed model leads to schemes with regular connections. This simplifies the placement and tracing tasks when implementing the SMPA scheme. A positive feature of the proposed model is the fact that the Clock and Start signals are associated with only one block of the circuit. This avoids the problems associated with the so-called distortion synchronization.

Results. Analysis of a special library showed that the proposed method is advisable to use for 68% of test cases. Moreover, for 76% of the examples, the original model can be used. For 24% of the examples, if the corresponding condition is met, only the proposed model can be used. Our studies using Virtex-6 FPGAs showed that as the parameters R, L, N increase, the gain in equipment, speed and energy consumption for the proposed machines compared to the original ones is increased An example is shown for using of proposed method. The conditions of its application are shown.

Conclusion. Further areas of our research are related to the development of similar SMPA structures using: coding of sets of microoperations of the Mealy automaton; the coding of the terms of systems; and the conversion of codes of pseudo-equivalent states of the Moore automaton.

Keywords*: combined automaton, synthesis, LUT, EMB, partition, FPGA.*

А.А. Баркалов, д-р техн. наук, профессор, Университет Зеленогурский (Польша),
ул. Подгорная, 50, 65-246, Зеленая Гура (Польша),
a.barkalov@iie.uz.zgora.pl

Л.А. Титаренко, д-р техн. наук, профессор, Университет Зеленогурский (Польша),
ул. Подгорная, 50, 65-246, Зеленая Гура (Польша),
L.Titarenko@iie.uz.zgora.pl, D_ts@nure.ua

Я.Е. Визор, канд. техн. наук, ст. научн. сотрудник, Институт кибернетики им. В.М. Глушкова
НАН Украины, 03187, г. Киев, пр. Акад. Глушкова, 40, Украина,
yaviz@ukr.net

А.В. Матвиенко, научн. сотрудник, Институт кибернетики им. В.М. Глушкова
НАН Украины, 03187, г. Киев, пр. Акад. Глушкова, 40, Украина,
matv@online.ua

СИНТЕЗ ЧЕТЫРЕХУРОВНЕВОЙ СХЕМЫ СОВМЕЩЕННОГО АВТОМАТА

Вступление. Предложен метод уменьшения аппаратурных затрат в схеме совмещенного автомата, реализуемой в совместном базисе элементов *LUT* и блоков памяти *EMB*. Метод основан на замене логических условий и разбиении множества логических состояний на классы. Каждый класс соответствует отдельному блоку схемы. Такой подход приводит к схемам с регулярной структурой и четырьмя логическими уровнями.

Цель. Предложенная модель приводит к схемам с регулярными связями. Это упрощает задачи размещения и трассировки при реализации схемы СМПА. Положительной чертой предложенной модели является тот факт, что сигналы *Clock* и *Start* связаны только с одним блоком схемы. Это позволяет избежать проблем, связанных с так называемым перекосом синхронизации.

Результаты. Анализ специальной библиотеки показал, что предложенный метод целесообразно использовать для 68 процентов тестовых примеров. При этом для 76 процентов примеров может быть использована исходная модель. Для 24 процентов примеров, при выполнении соответствующего условия, может быть использована только предложенная модель. Наши исследования с использованием *FPGA Virtex*-6 показали, что по мере роста параметров *R*, *L*, *N* увеличивается выигрыш в быстродействии и потребляемой энергии используемой аппаратуры с применением предложенных автоматов в сравнении с исходными. Приведен пример использования предложенного метода и показаны условия его применения

Выводы. Дальнейшие направления наших исследований связаны с разработкой подобных структур СМПА с использованием: кодирования наборов микроопераций автомата Мили; кодирования термов систем и преобразованием кодов псевдоэквивалентных состояний автомата Мура.

Ключевые слова: совмещенный автомат, синтез, *LUT*, *EMB*, разбиение, *FPGA*.