DOI https://doi.org/10.15407/csc.2022.02.021
УДК 004.274

О.О. БАРКАЛОВ, доктор техн. наук, професор, Зеленогурський Університет (Польща), 65246, Зелена Гура, вул. Підгірна, 50, Польща, ORCID: https://orcid.org/0000-0002-4941-3979, A.Barkalov@iie.uz.zgora.pl

Л.О. ТІТАРЕНКО, доктор техн. наук, професор, Зеленогурський Університет (Польща), 65246, Зелена Гура, вул. Підгірна, 50, Польща; Харківський нац. ун-т радіоелектроніки, 61000, м. Харків, просп. Науки, 14, Україна, ORCID: https://orcid.org/0000-0001-9558-3322, L.Titarenko@iie.uz.zgora.pl

О.М. ГОЛОВІН, кандидат технічних наук, старший науковий співробітник, Ін-т кібернетики ім. В.М. Глушкова НАН України, 03187, м. Київ, просп. Академіка Глушкова, 40, Україна, ORCID: https://orcid.org/0000-0002-0279-812X, o.m.golovin.1@gmail.com

О.В. МАТВІЄНКО, наук. співробітник, Ін-т кібернетики ім. В.М. Глушкова НАН України, 03187, м. Київ, просп. Академіка Глушкова, 40, Україна, ORCID: https://orcid.org/0000-0003-1838-1422, matv@online.ua

ОПТИМІЗАЦІЯ СХЕМИ АВТОМАТА МІЛІ В БАЗИСІ FPGA

Запропоновано метод зменшення апаратурних витрат у схемі мікропрограмного автомата (МПА) Мілі, що реалізується в базисі EMB і LUT. Метод заснований на використанні EMB для реалізації блоку заміни логічних умов. Пропонується частину вихідних сигналів (мікрооперацій) реалізувати на EMB. Показано умови застосовності даного підходу. Наведено приклад синтезу МПА із застосуванням запропонованого методу. Всі етапи синтезу детально проаналізовано. Запропоновано низку альтернативних рішень і показано умови їхнього використання.

Ключові слова: *автомат Мілі, FPGA, EMB, LUT, структурна декомпозиція, синтез.*

Вступ

Нині різні цифрові системи проникли в усі сфери людської діяльності [1]. Очевидно, що зменшення апаратурних витрат у схемах цифрових систем у глобальних масштабах веде до колосальної економії. Цифрові системи складаються з комбінаційних і накопичувальних блоків [2, 3]. Для реалізації комбінаційних схем є стандартні рішення, що використовуються в різних системах автоматизованого проєктування (САПР) [1, 4]. Для багатьох типів накопичувальних блоків, наприклад, пристроїв управління (ПУ), немає стандартних бібліотечних рішень. Схеми ПУ необхідно щоразу проєктувати спочатку [5]. Як правило, ці схеми вирізняються нерегулярною структурою та складною системою міжз'єднань [1, 4]. При цьому характеристики ПУ суттєво впливають на якість цифрової системи [5].

Найчастіше для того, щоб задати поведінку ПУ, використовується модель мікропрограмного автомата (МПА) *Мілі* [6, 7]. І тут оптимізація ПУ зводиться до оптимізації схеми МПА. Методи оптимізації схеми МПА великою мірою залежать від особливостей елементного базису, що використовується для реалізації цифрових систем [1, 2]. Нині для реалізації цифрових систем широко використовують-

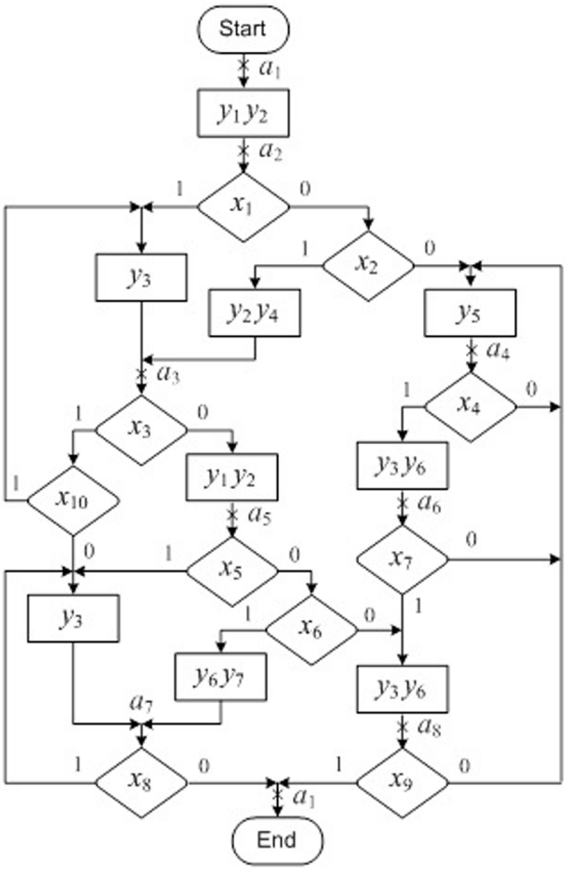

Рис.1. Помічена ГСА Γ_1

ся мікросхеми *FPGA* (*field-programmable logic array*) [8, 9]. У зв'язку з цим ми розглядаємо задачу оптимізації апаратурних витрат у схемі МПА *Мілі*, що реалізується в базисі *FPGA*.

Провідним виробником мікросхем *FPGA* є фірма *Xilinx* [10]. Тому запропонований нами підхід насамперед зорієнтовано на *FPGA* цієї фірми. У загальному випадку для реалізації схеми МПА можна використовувати такі ресурси *FPGA*: елементи *LUT* (*look-up table*), вбудовані блоки пам'яті *EMB* (*embedded memory blocks*) і блоки введення/виводу [1, 4].

Апаратурні витрати для схем на основі *FPGA* прийнято оцінювати площею кристала, яку займає схема [11, 12]. Ця площа є пропорційною числу елементів у схемі МПА [12]. У роботі ми також використовуємо цей підхід.

Реалізація автоматів FPGA в базисі FPGA

Автомат *Мілі* представляється вектором $S = =< A, X, Y, \delta, \lambda, a_1 >$ [6, 7]. Компонентами вектора S є множина внутрішніх станів $A = \{a, \ldots \ldots, a_M\}$, множина входів $X = \{x_1, \ldots, x_L\}$, множина виходів $Y = \{y_1, \ldots, y_N\}$, функція переходів δ: $A \times X \to A$, функція виходів λ: $A \times X \to Y$, початковий стан $a_1 \in A$. У випадку ПУ входи називаються логічними умовами (ЛУ), а виходи — мікрооперaціями (МО) [3, 6]. Ми також будемо використовувати цю термінологію.

Для задання вектора S можна використовувати різні методи [7]: таблиці переходів, графи переходів, двійкові діаграми, графи I-НЕ (*and-not graphs*) [9]. У статті використовується мова граф-схем алгоритму (ГСА) [6]. Ця мова має гарну наочність, що важливо для ілюстрації запропонованого нами методу.

ГСА Г — це кінцевий орієнтований граф, що має 4 види вершин: початкову, кінцеву, операторну та умовну [6]. В операторних вершинах записуються набори мікрооперaцій, які виконуються в одному такті роботи МПА. В умовних вершинах записуються елементи множини ЛУ, що визначають переходи між вершинами. Наприклад, ГСА Γ_1 має 9 операторних та 10 умовних вершин (рис. 1). Стан автомата *Мілі* визначається згідно з правилами [13]. Аналіз ГСА Γ_1 дозволяє визначити такі властивості МПА Мілі: M=8, L=10, N=7.

Для синтезу схеми МПА по ГСА необхідно [6]:

1. закодувати стани $a_m \in A$ двійковими кодами $K(a_m)$,

2. побудувати пряму структурну таблицю (ПСТ),

3. побудувати системи булевих функцій (СБФ), що задають схему МПА. Ми розглядаємо випадок максимального кодування станів, коли число розрядів коду визначається формулою

$$R = \lceil \log_2 M \rceil . \qquad (1)$$

Для кодування станів використовуються внутрішні змінні, що утворюють множину $T = \{T_1, \ldots, T_R\}$. Ці змінні формуються на

виходах регістра кодів станів (*RG*). Для зміни вмісту *RG* використовуються функції збудження пам'яті, що утворюють множину $\Phi = \{D_1, \ldots, D_R\}$. Символ $D_r \in \Phi$ відображає той факт, що *RG* складається з тригерів типу *D* [3]. Зміни вмісту *RG* відбуваються під впливом імпульсу синхронізації *Clock*. Як правило, за сигналом *Start* в *RG* записується код початкового стану $a_1 \in A$.

Пряма структурна таблиця МПА має такі стовпці [6]: a_m — початковий стан; $K(a_m)$ — код початкового стану; a_s — стан переходу; $K(a_s)$ — код стану переходу; X_h — вхідний сигнал, що визначає $<a_m, a_s>$ та дорівнює кон'юнкції ЛУ $x_1 \in X$ (або їхніх заперечень); Y_h — набір МО, який формується на переході $<a_m, a_s>$; Φ_h — набір ФЗП, що дорівнюють 1 для запису в *RG* коду $K(a_s)$; h — номер переходу ($h \in \{0, ..., H\}$). Методику переходу от ГСА до ПСТ розглянуто, наприклад, у роботі [6]. На базі ПСТ формуються наступні СБФ:

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

$$Y = Y\ (T, X). \qquad (3)$$

Реалізація схеми МПА зводиться до технологічного відображення (*technology mapping*) систем (2)–(3) [14]. Цей етап визначається особливостями використовуваних логічних елементів. Ми розглядаємо технологічне відображення в базисі *LUT* та *EMB*.

Елемент *LUT* складається з q комірок пам'яті і має *SL* адресних входів. Кожна комбінація входів передає на вихід біт інформації з відповідної однорозрядної комірки пам'яті [1, 4]. Отже, *LUT* реалізує довільну функцію логіки алгебри, що має до *SL* аргументів. Вихід елемента *LUT* пов'язано із входом тригера. Така сполука утворює логічний елемент (рис.2).

На виході *LUT* формується змінна $f_C \in \{0, 1\}$, значення якої може змінюватися зі зміною вхідної комбінації. За сигналом *Clock* значення f_C записується в тригер ($f_R := f_C$). Значення змінної f_R може змінюватися лише за наявності синхронізації. Внутрішня змінна f_0 обирає комбінаційну (f_C) або регістрову (f_R) форму вихідного сигналу, яка формується на виході мультиплексора *MX*.

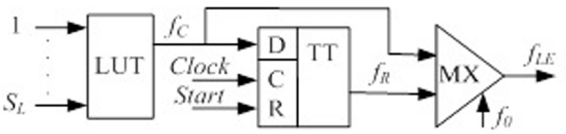

Рис.2. Структурна схема логічного елемента

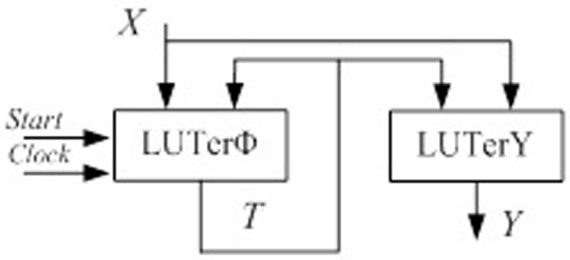

Рис.3. Структурна схема P_L автомата Мілі

Нехай $NA(f_i)$ означає число аргументів функції $f_i \in \Phi \cup Y$ та виконується умова

$$NA(f_1) \le S_L,\ (i \in \{1, \ldots, N+R\}). \qquad (4)$$

У цьому випадку для реалізації схеми будь-якої функції (2)–(3) достатньо одного елемента *LUT*. Схема МПА має $N+R$ елементів *LUT*, а коди станів зберігаються в регістрі *RG*, який розподілено між *LUT*, що реалізують СБФ (2). Умова (4) визначає однорівневий МП P_L МПА (рис. 3). Індекс *L* означає, що схему реалізовано на елементах *LUT*.

На рис.3 символ "*LUTer*" означає блок, що складається із логічних елементів (рис.2). У P_L МПА блок *LUTer* реалізує СБФ (2) і включає розподілений регістр кодів станів. Блок *LUTerY* реалізує СБФ (3).

Однорівневий автомат (P_L МПА) — найкращий варіант схеми на основі *LUT*. Однак таке рішення можливе лише для дуже простих автоматів. Число входів S_L є дуже малим ($S_L \le 6$ [10, 15]) у порівнянні з характеристиками реальних МПА. Наприклад, для автоматів середньої складності [6] характерні наступні характеристики: $L \approx 30$, $R \approx 6$, $N \approx 50$.

Характеристики схем МПА в базисі *FPGA* можна покращити, замінивши елементи *LUT* блоками *EMB*. Блок *EMB* має S_A входів і t_F ви-

ходів. Ємність пам'яті *EMB* (V_0) визначається як

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

Величина V_0 є постійною для блока *EMB*. Однак параметри S_A і t_F можуть змінюватися.

Пара $<S_A, t_F>$ визначає конфігурацію *EMB*. Наприклад, для *EMB* сімейства *Virtex*-7 [16] фірми *Xilinx* є такі пари $<S_A, t_F>$: <15, 1>, <14, 2>, <13, 4>, <12, 8>, <11, 16>, <10, 32> <9, 64>. Отже, *EMB* є дуже гнучким засобом реалізації СБФ [3, 12–14].

Якщо виконується умова

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

то СБФ (2)–(3) реалізуються на одному блоці *EMB*. Назвімо таку схему P_E МПА *Мілі*, де «*E*» означає, що схему реалізовано на блоці пам'яті. Блоки *EMB* мають входи *Clock* і *Start* [10]. Через це для реалізації схеми МПА не використовується додатковий регістр.

Як показав аналіз бібліотеки [20], умова (6) виконується тільки для 67% усіх наявних в ній прикладів. В інших випадках потрібно представити МПА у вигляді мережі блоків *EMB*. Наприклад, для МПА, представленого ГСА Γ_1, маємо $L+R=13$. Через це один блок *EMB* може реалізувати лише тільки 4 функції, що відповідає парі <13, 4>. Отже, при використанні *FPGA Virtex*-7 необхідно $\left[(N+R)/4\right] = \left[(10+3)/4\right] = 4$ блоки *EMB*.

Як зазначено у [17], блоки *EMB* широко використовуються для реалізації операційних блоків цифрових систем. При цьому цілком можлива ситуація, коли у розробника є недостатня кількість блоків *EMB*. Саме така ситуація розглядається у цій статті.

Основна ідея запропонованого методу

Нехай умова (4) порушується для деякого МПА *Мілі* S_1 і тільки один блок *EMB* є у розпорядженні розробника схеми ПУ. При цьому схема МПА представляється у вигляді P_{EL} автомата (рис.4).

Як видно з рис.4, блок *EMB* реалізує СБФ $\Phi_E = \Phi_E(T, X_E)$ і $Y_E = Y_E(T, X_E)$. При цьому $\Phi_E \cap \Phi_L = \varnothing$ і $X_E \cap X_L = \varnothing$. Блок *LUTerTY* реалізує СБФ $\Phi_L = \Phi_L(T, X_L)$ і $Y_L = Y_L(T, X_L)$. У загальному випадку виконується умова $X_E \cap X_L = \varnothing$. Для реалізації на *EMB* необхідно вибрати t_E функцій $f_i \in \Phi \cup Y$, для яких різниці $NA(f_i) - S_F$ є максимальними [21, 22]. При цьому спрощується схема блоку *LUTerTY*.

Якщо умова (4) порушується хоча б для однієї з функцій $\Phi_L \cup Y_L$, то блок *LUTerTY* представляється у вигляді багаторівневої схеми. Для реалізації таких схем використовуються методи функціональної декомпозиції [14]. Основний недолік цих методів — відсутність можливості регулювати число рівнів схеми. Це унеможливлює точну апріорну оцінку швидкодії. Крім того, схеми, що базуються на функціональній декомпозиції, мають складну систему міжз'єднань.

Як зазначено в [23], у сучасних мікросхемах *FPGA* міжз'єднання споживають до 70% потужності. Зі збільшенням рівня інтеграції затримки на міжз'єднаннях починають відігравати домінуючу роль. Через це для схем на базі *FPGA* важливо спростити систему міжз'єднань. Цього можна досягти, використовуючи методи структурної декомпозиції (СД) [24]. Сенс СД зводиться до представлення схеми МПА у вигляді композиції окремих логічних блоків. Кожен блок реалізує функції, відмінні від СБФ (2)–(3). Як показали наші дослідження, СД дає змогу зменшити кількість рівнів схеми порівняно із схемами еквівалентних *PEL* автоматів Мілі.

Ми пропонуємо оптимізувати схему МПА *Мілі*, використовуючи метод заміни логічних умов [6]. При цьому множина X замінюється множиною додаткових змінних $P = \{p_1, \ldots, p_G\}$, де $|G| << L$.

Нехай для певного автомата *Мілі* виконано заміну ЛУ додатковими змінними ($X \to P$) та визначено величину параметра *G*. Нехай серед конфігурацій блока *EMB* є конфігурація $< S_A^0, t_F^0 >$, для якої виконується умова

$$S_A^0 = L + R; \quad (7)$$

$$t_F^0 \ge G. \quad (8)$$

У цьому разі ми пропонуємо виконати заміну ЛУ за допомогою блока *EMB*. Це приводить до $M_E P_{LE}$ МПА *Мілі*.

Якщо виконується умова

$$\Delta_t = t_F^0 - G > 0, \quad (9)$$

то доцільно використовувати Δ_t вільних виходів *EMB* для формування МО $y_n \in Y$. Для цього необхідно розбити множину Y на множини Y_E и Y_L, що не перетинаються. За допомогою *EMB* формуються МО $y_n \in Y_E$. Мікрооперації $y_n \in Y_L$ формуються схемою блока *LUTerT*.

Схема $M_E P_{LE}$ автомата представляється наступними СБФ:

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

$$Y_E = Y_E(T, X); \quad (11)$$

$$Y_L = Y_L(T, P); \quad (12)$$

$$\Phi = \Phi(T, P). \quad (13)$$

Системи (10)–(11) представляються таблицями істинності [6, 7]. Системи (12)–(13) представляються диз'юнктивними нормальними формами (ДНФ) [7]. Ці форми формуються за ПСТ $M_E P_{LE}$ автомата *Мілі*.

У цій роботі ми пропонуємо наступний метод синтезу $M_E P_{LE}$ МПА по ГСА Г:

1) формування множини станів A.
2) Заміна ЛУ $x_1 \in X$ змінними $p_g \in P$.
3) Розбиття множини Y на класи Y_E и Y_L.
4) Кодування станів $a_m \in A$.
5) Формування ПСТ автомата $M_E P_{LE}$.
6) Формування систем (12)–(13) за ПСТ.
7) Формування таблиці блока *EMB*.
8) Реалізація схеми МПА в заданому елементному базисі.

Позначмо символом $M_E P_{LE}(\Gamma_j)$ той факт, що автомат *Мілі* синтезується за ГСА Γ_j і має структурну схему, показану на рис.5. Розгляньмо, як виконуються окремі етапи запропонованого метода на прикладі синтезу МПА $M_E P_{LE}(\Gamma_1)$.

Приклад синтезу схеми $M_E P_{LE}$ автомата Мілі

Нехай у розпорядженні розробника ПУ є елементи *LUT* з S_L=5, і один блок *EMB* із конфігураціями <15, 1>, <14, 2>, <13, 4>, ..., <9, 64>. Перевіримо можливість використання моделі $M_E P_{LE}$ автомата для ГСА Γ_1.

Як було показано раніше, для ГСА Γ_1 маємо автомат *Мілі* з такими характеристиками: M=8, L=10, N=7. Використовуючи (1), знайдемо R=3, $T = \{T_1, T_2, T_3\}$ і $\Phi = \{D_1, D_2, D_3\}$. Отже, маємо $L+R$=13. З множини пар $<S_A, t_F>$ обираємо пару <13, 4>, що дає $S_A^0 = 13, t_F^0 = 4$. Тепер необхідно знайти величину G.

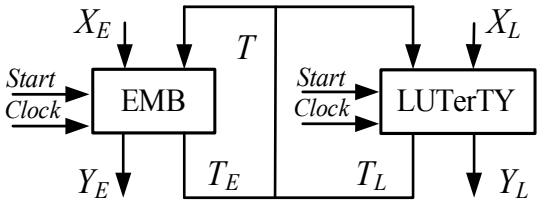

Рис. 4. Структурна схема *PEL* МПА Мілі

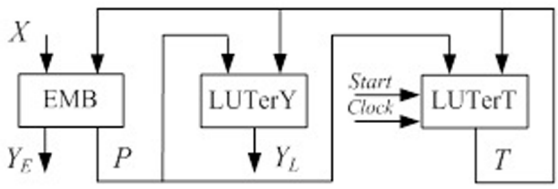

Рис.5. Структурна схема $M_E P_{LE}$ МПА Мілі

Нехай $X(a_m)$ — множина ЛУ, що визначають перехід зі стану $a_m \in A$. Нехай $L = |X(a_m)|$ — число елементів у множині $X(a_m) \subseteq X$. Число елементів у множині P визначається наступною формулою [7]:

$$G = \max(L_1, \ldots, L_M). \quad (14)$$

З аналізу ГСА Γ_1 випливають множини $X(a_1) = \varnothing$, $X(a_2)=\{x_1, x_2\}$, $X(a_3)=\{x_3, x_{10}\}$, $X(a_4)=\{x_4\}$, $X(a_5)=\{x_5, x_6\}$, $X(a_6)=\{x_7\}$, $X(a_7)=\{x_8\}$ і $X(a_8)=\{x_9\}$. Це дає наступні значення L_m: L_1=0, $L_4=L_6=L_7=L_8$=1, $L_2=L_3=L_5$=2. З (14) випливає, що G=2 і $P=\{p_1, p_2\}$.

Оскільки $t_F^0 = 4$, G=2, то з (9) знаходимо, що Δ_t=2. Отже дві мікрооперації можуть бути включені до множини Y_E. Побудуймо таблицю заміни ЛУ, використовуючи методику [6]. Таблиця має стовпці $a_m \in A$ та рядки $p_g \in P$. Якщо додаткова змінна $p_g \in P$ замінює ЛУ $x_1 \in X(a_m)$, то умова x_1 записується на перетині рядка p_g і стовпця a_m. Є алгоритми виконання заміни ЛУ. Однак, якщо використовується блок *EMB*, то заміна виконується у тривіальний спосіб. У прикладі таблицю заміни ЛУ представлено табл.1.

Таблиця 1. **Таблиця заміни ЛУ МПА $M_E P_{LE}(\Gamma_1)$**

a_m	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
p_1		x_1	x_3	x_4	x_5	——	x_8	——
p_2	——	x_2	x_{10}	——	x_6	x_7	——	x_9

Таблиця 2. **ПСТ автомата $P(\Gamma_1)$**

a_m	$K(a_m)$	a_s	$K(a_s)$	X^h	Y^h	Φ^h	h
a_1	000	a_2	001	1	$y_1 y_2$	D_3	1
A_2	001	a_3	010	x_1	y_3	D_2	2
		a_3	010		$y_2 y_4$	D_2	3
		a_4	011		y_5	$D_2 D_3$	4
A_3	010	a_3	010		y_3	D_2	5
		a_7	110		y_3	$D_1 D_2$	6
		a_5	100		$y_1 y_2$	D_1	7
A_4	011	a_6	101	x_4	$y_3 y_6$	$D_1 D_3$	8
		a_4	011		y_5	$D_2 D_3$	9
A_5	100	a_7	110		y_3	$D_1 D_2$	10
		a_7	110		$y_6 y_7$	$D_1 D_2$	11
		a_8	111		$y_3 y_6$	$D_1 D_2 D_3$	12
A_6	101	a_8	111		$y_3 y_6$	$D_1 D_2 D_3$	13
		a_5	100		y_5	D_1	14
A_7	110	a_7	110		y_3	$D_1 D_2$	15
		a_1	000				16
A_8	111	a_1	000				17
		a_4	011		y_5	$D_2 D_3$	18

Оскільки заміна $X \to P$ реалізується блоком *EMB*, то система (10) представляється таблицею істинності. Тому немає потреби формувати ДНФ функцій $p_g \in P$. У цьому випадку $R+G=5$, тобто виконується умова

$$R+G \le S_L. \qquad (15)$$

Тому для реалізації будь-якої функції $f_1 \in \in \Phi \cup Y$ необхідним є тільки один *LUT*. Це означає, що розбиття множини Y на класи Y_E і Y_L можна виконати у тривіальний спосіб. Нехай це розбиття дає множини $Y_E = \{y_1, y_2\}$ і $Y_L=\{y_3, ..., y_7\}$. У загальному випадку необхідно знайти ДНФ функцій (12). Потім для кожної функції визначається число елементів *LUT* у схемі, що формує цю функцію. Звичайно, множина Y_E включає функції, схеми для яких мають найбільшу кількість елементів.

Кодування станів є одним із найважливіших етапів синтезу [14]. Є безліч методів кодування, спрямованих на оптимізацію апаратурних витрат [24]. Одним із найкращих вважається алгоритм *JEDI*, що розповсюджується із системою *SIS* [1, 4]. Однак у нашому випадку виконується умова (15), тому результат кодування станів не впливає на число елементів у схемі МПА. Закодуймо стани у тривіальний спосіб: $a_m \in A$ $K(a_1)$=000, $K(a_2)$=001, ..., $K(a_8)$=111. Пряма структурна таблиця $M_E P_{LE}$ автомата має стовпці a_m, $K(a_m)$, a_s, $K(a_s)$, P_h, Y_h, Φ_h, h. Отже, ПСТ P МПА і ПСТ $M_E P_{LE}$ МПА відрізняються лише одним стовпцем. Для знаходження ДНФ

Таблиця 3. **ПСТ автомата $M_E P_{LE}(\Gamma_1)$**

a_m	$K(a_m)$	a_s	$K(a_s)$	P^h	X^h	Φ^h	h
a_1	000	A_2	001	1		D_3	1
a_2	001	A_3	010	x_1	y_3	D_2	2
		A_3	010		y_4	D_2	3
		A_4	011		y_5	$D_2 D_3$	4
a_3	010	A_3	010		y_3	D_2	5
		A_7	110		y_3	$D_1 D_2$	6
		A_5	100			D_1	7
a_4	011	A_6	101	x_4	$y_3 y_6$	$D_1 D_3$	8
		A_4	011		y_5	$D_2 D_3$	9
a_5	100	A_7	110		y_3	$D_1 D_2$	10
		A_7	110		$y_6 y_7$	$D_1 D_2$	11
		A_8	111		$y_3 y_6$	$D_1 D_2 D_3$	12
a_6	101	A_8	111		$y_3 y_6$	$D_1 D_2 D_3$	13
		A_5	100		y_5	D_1	14
a_7	110	A_7	110		y_3	$D_1 D_2$	15
		A_1	000				16
a_8	111	A_1	000				17
		A_4	011		y_5	$D_2 D_3$	18

функцій $y_h \in Y_E$ необхідно побудувати ПСТ P МПА. Для знаходження функцій (12)–(13) необхідно перетворити цю ПСТ на ПСТ $M_E P_{LE}$ МПА. У нашому прикладі це табл. 2 і табл. 3. Ми використовуємо символ $P(\Gamma_1)$ для позначення того, що P автомат реалізується за ГСА Γ_1.

Використовуючи табл.2, можна отримати ДНФ функцій $y_1, y_2 \in Y_E$. Ці ДНФ формуються на основі термів

$$F_h = A_m X_h . \qquad (16)$$

У (16) символ A_m означає кон'юнкцію внутрішніх змінних, які відповідають коду $K(a_m)$ [6]. Для нашого прикладу маємо:

$$y_1 = F_1 \vee F_7 = \overline{T_1}\,\overline{T_2}\,\overline{T_3} \vee \overline{T_1}\,T_2\,\overline{T_3}\,\overline{x_3}\,;$$
$$y_2 = F_1 \vee F_3 \vee F_7 = \qquad (17)$$
$$= \overline{T_1}\,\overline{T_2}\,\overline{T_3} \vee \overline{T_1}\,\overline{T_2}\,T_3\,\overline{x_1}\,x_2 \vee \overline{T_1}\,\overline{T_2}\,T_3\,\overline{x_3}.$$

Система (17) має бути перетворена на таблицю істинності. Ця таблиця включатиме $L+R=13$ аргументів.

Розгляньмо, як здійснюється перехід від табл.2 до табл.3. У рядку 1 стовпця Y_h табл.2 записано набір $\{y_1, y_2\}$. Оскільки $y_1, y_2 \in Y_E$, в рядку 1 стовпця Y_h табл.3 записано знак "-", який відповідає порожньому набору МО. Далі, для $h=3$ (табл.2) $X_h = \overline{x_1} x_2$. Як випливає з табл.1, $x_1=p_1$ і $x_2=p_2$. Тому для рядка $h=3$ (табл.3) записується кон'юнкція $P_h = \overline{p_1} p_2$. Інші рядки стовпців P_h и Y_h (табл.3) заповнено на основі аналогічного аналізу.

З табл.3 можна знайти терми

$$F_h = A_m P_h , \; h = \overline{1, H}. \qquad (18)$$

Терми (18) входять у ДНФ функцій (12) – (13). Наприклад, з табл.3 можна знайти такі ДНФ:

$$D_1 = \left|F_6 \vee F_7\right| \vee F_8 \vee \left|F_{10} \vee F_{11} \vee F_{12} \vee F_{13} \vee F_{14} \vee\right|$$
$$\vee F_{15} = \overline{T_1} T_2 \overline{T_3}\,\overline{p_2} \vee \overline{T_1}\,T_2\,\overline{T_3} p_1 \vee T_1\,\overline{T_2} \vee T_1 T_2\,\overline{T_3} p_1, \quad (19)$$
$$y_4 = F_3 = \overline{T_1}\,\overline{T_2}\,T_3\,\overline{p_1} p_2.$$

Таблиця 4. **Фрагмент таблиці *EMB* автомата $M_E P_{LE}(\Gamma_1)$**

a_m	$K(a_m)$			X										P		Y		h
	T_1	T_2	T_3	x_{10}	x_9	x_8	x_7	x_6	x_5	x_4	x_3	x_2	x_1	p_1	p_2	y_1	y_2	
a_2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1025
a_2	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1026
a_2	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1027
a_2	0	0	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1028
a_2	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1029
a_2	0	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	1030
a_2	0	0	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	1031
a_2	0	0	1	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1032
a_2	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1033
a_2	0	0	1	0	0	0	0	0	0	1	0	0	1	0	1	0	0	1034

Таблиця блоку *EMB* має стовпці a_m, $K(a_m)$, X (адрес комірки пам'яті), P, Y_E (вміст комірки пам'яті), h (номер комірки). Число рядків H_0 цієї таблиці визначається формулою

$$H_0 = 2^{L+R}. \qquad (20)$$

У цьому прикладі $H_0 = 2^{13} = 8192$ (рядки). Ця таблиця будується на основі таблиці заміни ЛУ і ПСТ P автомата. Переходи зі стану $a_m \in A$ представляються частиною таблиці, що має H_m рядків:

$$H_m = 2^L. \qquad (21)$$

У нашому прикладі $H_0 = 2^{10}=1024$. Наприклад, перші 10 рядків для станів $a_2 \in A$ показані в табл.4.

Табл.4 починається рядком 1025, оскільки перші 1024 рядки задають $p_g \in P$ і $y_n \in Y_E$ для стану $a_1 \in A$. Якщо $x_1=x_2=0$, то це відповідає рядку 4 (табл.2). Якщо $x_1=0$ і $x_2=1$, то це відповідає рядку 3 (табл.2). Якщо $x_1=1$ і $x_2=0$, то це відповідає рядку 2 (табл.2). Для рядків 1025–048 $p_1=x_1$ і $p_2=x_2$ (що випливає з табл.1). У рядку 3 (табл.2) $y_2=1$. Тому $y_2=1$ в рядках табл.4, для яких $x_1=0$ і $x_2=1$ (1027, 1030, 1034, тощо). У стовпці y_1 записано нуль для всіх рядків 1025–048, оскільки $y_2=0$ при всіх переходах зі стану $a_2 \in A$ (табл.2).

Схема МПА $M_E P_{LE}(\Gamma_1)$ складається з одного блоку *EMB* і $R + L - |Y_E| = 8$ елементів *LUT*. Для синтезованого МПА виконується умова (5). Тому для реалізації функцій $f_i \in \Phi \cup Y_L$ необхідним є лише один рівень елементів *LUT* (рис.6).

У схемі (рис.6) елементи *LUT*1–*UT*5 реалізують *LUTerY*, *LUT*6–*UT*8 реалізують *LUTerT*. Виходи *EMB* та елементів *LUT* утворюють шини, що відповідають виходам на рис.5.

Останній етап запропонованого методу пов'язано із застосуванням стандартних пакетів САПР, таких як *Vivado* [25] фірми *Xilinx*. Ми не розглядаємо цей крок у нашому прикладі.

Аналіз імовірних альтернатив

Запропонований метод дає економію апаратурних засобів, якщо виконується умова (15). Інакше, схеми блоків *LUTerY* та *LUTerT* будуть багаторівневими. Для цього варіанта ми пропонуємо три альтернативних рішення.

Одне рішення — кодування Q наборів МО $Y_q \subseteq Y$ двійковими кодами $K(Y_q)$ розрядності

$$R_Q = \lceil \log_2 Q \rceil. \qquad (22)$$

Для кодування використовуються елементи множини $Z = \{z_1, \ldots, z_{R_Q}\}$. Якщо виконується умова

$$2^{L+R} \cdot (G + R_Q) \le V_0, \qquad (23)$$

то МПА *Мілі* може бути представлений у вигляді $M_E P_L Y_L$ автомата (рис.7).

У $M_E P_L Y_L$ автоматі блок *EMB* формує СБФ (10) і

$$Z = Z(T, X). \qquad (24)$$

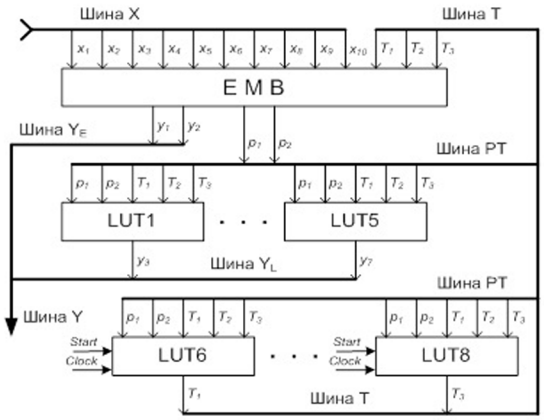

Рис. 6. Логічна схема автомата $M_E P_{LE}$ (Г1)

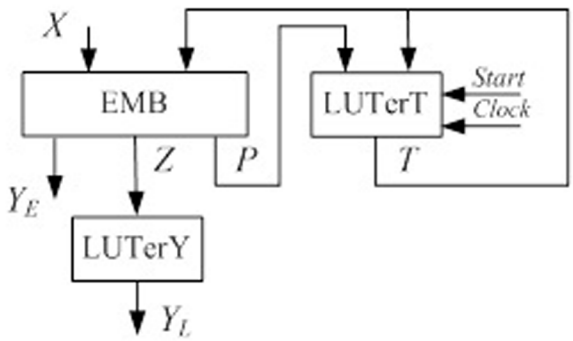

Рис.8. Структурна схема $M_E P_L Y_{EL}$ автомата Мілі

Блок *LUTerT* реалізує СБФ (13), а *LUTerY* — СБФ

$$Y = Y(\boldsymbol{Z}). \qquad (25)$$

Ця структура має кілька варіантів, що залежать від поєднання характеристик *EMB* і МПА. Нехай серед конфігурацій *EMB* є така, для якої виконується умова (7) та

$$t_F^0 > G + R_Q . \qquad (26)$$

У цьому випадку $\Delta_t = t_F^0 - (G + R_Q)$ МО утворюють множину $Y_E \in Y$. Мікрооперації $y_n \in Y_E$ формуються блоком *EMB*, а МО $y_n \in Y_L = Y \setminus Y_E$ — блоком *LUTerY*. Це призводить до $M_E P_L Y_{EL}$ автомату *Мілі* (рис.8).

Якщо видалення Y_E з Y призводить до зменшення величини R_Q, то число МО, що реалізуються блоком *EMB*, збільшується. Формування множин Y_E і Y_L доцільно виконувати з використанням методики [22].

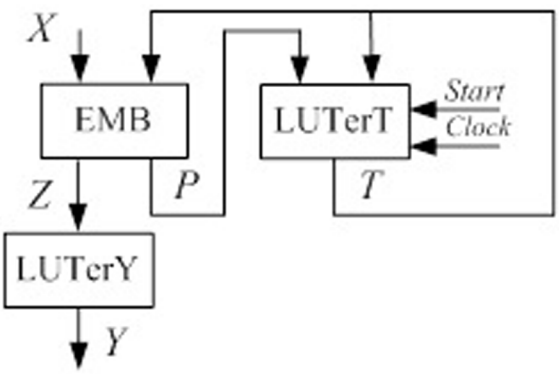

Рис.7. Структурна схема $M_E P_L Y_L$ МПА Мілі

Якщо для цієї конфігурації виконується умова (7), а умова (8) порушується, то множину P слід розділити на множини P_E і P_L, що не перетинаються. Це призводить до $M_{EL} P_L$ автомата Мілі (рис.9).

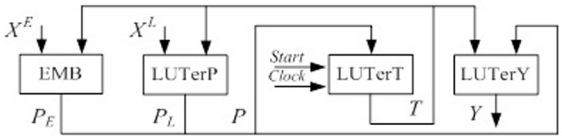

Рис.9. Структурна схема $M_{EL} P_L$ МПА Мілі

Як видно з рис.9, систему (10) представлено у вигляді двох систем:

$$P_E = P_E(T, X^E); \qquad (27)$$

$$P_L = P_L(T, X^L). \qquad (28)$$

Система (27) реалізується на *EMB*, система (28) — на *LUTerP*. Функції інших блоків є зрозумілими з попереднього тексту.

Методи синтезу цих МПА багато в чому є подібними до методів синтезу $M_E P_{EL}$ МПА *Мілі*. Однак є відмінності, пов'язані з особливостями структурних діаграм. У подальших дослідженнях плануємо розробити методи синтезу схем МПА рис.7–рис.9.

Висновки

Запропонований метод належить до групи методів реалізації схем МПА у змішаному елементному базисі. Цей метод використовується, якщо ресурсів блоків пам'яті *EMB*, що ви-

користовуються, недостатньо для реалізації схеми МПА. Для дослідження запропонованого методу використовувалася бібліотека стандартних автоматів [20]. Дослідження здійснювалися для мікросхем сімейства *Virtex*-7 [16] з використанням САПР *Vivado* [25] фірми *Xilinx*. Дослідження показали, що з одного блоку *EMB* наш метод дає виграш за кількістю елементів *LUT*. Порівняння здійснювалися із схемами, отриманими з використанням методів *Auto*, *One-hot* та *Sequential* системи *Vivado*. У методах, що використовують *Vivado*, немає заміни вхідних змінних. Цим значною мірою пояснюється виграш нашого підходу. У подальших дослідженнях ми плануємо розробити методи синтезу автоматів *Мілі*, схеми яких (рис.7–рис.9) запропоновано у статті. Крім того, ми збираємося перевірити ефективність запропонованого підходу для оптимізації схем МПА Мура та суміщених МПА [26, 27].

ЛІТЕРАТУРА

1. *Sklyarov V., Skliarova I., Barkalov A., Titarenko L.* Synthesis and optimization of FPGA-based systems. Berlin: Springer, 2014. 432 p.
2. *Czerwinski R., Kania D.* Finite state machines logic synthesis for complex programmable logic devices. Berlin: Springer, 2013. 172 p.
3. *Соловьев В.В.* Проектирование цифровых схем на основе программируемых логи-ческих интегральных схем. Москва: Горячая линия – ТЕЛЕКОМ, 2001. 636 с.
4. *Maxfield C.* The design warrior's guide to FPGAs. Orlando: Academic Press, 2004. 542 p.
5. *Skliarova I., Sklyarov V., Sudnitson A.* Design of FPGA-based circuits using hierarchical finite state machines. Tallinn: TUT Press, 2012. 240 p.
6. *Baranov S.* Logic synthesis for control automata. Dordrecht: Kluwer Academic Publishers, 1994. 312 p.
7. *DeMicheli G.* Synthesis and optimization of digital circuits. New York: McGraw-Hill, 1994. 576 p.
8. *Kuon I., Tessier R., Rose J.* FPGA Architecture: Survey and Challenges. Foundations and Trends in Electronic Design Automation. 2008. vol. 2. N 2, pp. 135–253.
9. *Sass R., Schmidt A.* Embedded System Design with platform FPGAs: Principles and Practices. Amsterdam: Morgan Kaufmann Publishers, 2010–409 pp.
10. UG473 (v1.14) July 3, 2019. URL: www.xilinx.com.
11. *Rafla N.I., Gauba I.* A Reconfigurable Pattern Matching Hardware Implementation using On-Chip RAM-Based FSM, in 2010 53rd IEEE International Midwest Symposium on Circuits and Systems. IEEE, August 2010, pp. 49–52.
12. *Tiwari A., Tomko K.* Saving power by mapping finite state machines into embedded memory blocks in FPGAs. Proc. Design, Automation and Test in Europe Conference and Exhibition (Paris, France, 6–20 Feb. 2004). 2004. Vol. 2. P. 916–921.
13. *Глушков В. М.* Синтез цифровых автоматов. М.: Физматгиз, 1962–476 с.
14. *Kubica M., Opara A., Kania D.* Technology Mapping for LUT-based. FPGA. Berlin: Springer, 2021.
15. Intel® FPGAs and Programmable Devices. https://www.intel.com/content/www/us/en/ prod-ucts/programmable. html.
16. *Yang S.* Logic synthesis and optimization benchmarks user guide. Version 3.0. Techn. Rep. Microelectronics Center of North Carolina, 1991. 43 p.
17. *Sklyarov V.* Synthesis and Implementation of RAM-based Finite States Machines in FPGAs. in Proceeding of Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing. Villach: Springer-Verlag, 2000, pp. 718–727
18. *Senhaji-Navarro R., Garcia-Vargas I., Jimenes-Moreno G., Civit-Balcells A., Guerra-Gutierres P.* "ROM-based FSM implementation using input multiplexing in FPGA devices", Electronics Letters, vol. 40, N 20, pp. 1249–1251, 2004.
19. *Senhaji-Navarro R., Garcia-Vargas I., Guisado L.J.* "Performance evaluation of RAM-based implementation of Finite States Machines in FPGAs ", in 2012 19th IEEE International Conference on Electronics, Circuits, and Systems (ICECS 2012), Seville, Spain, Dec 2012, pp. 225–228.
20. *Yang S.* Logic synthesis and optimization benchmarks user guide. Version 3.0. Techn. Rep. Microelectronics Center of North Carolina, 1991. 43 p.
21. *Barkalov A., Titarenko L., Mielcarek K.* Hardware reduction for LUT–based Mealy FSMs. International Journal of Applied Mathematics and Computer Science. 2018, pp. 595–607.

22. *Barkalov A., Titarenko L., Mielcarek K.* Improving characteristics of LUT–based Mealy FSMs. International Journal of Applied Mathematics and Computer Science, 2020, 30(4), pp. 745–759.
23. *Ruiz-Rosero, J.; Ramirez-Gonzalez, G.; Khanna, R.* Field Programmable Gate Array Applications–A Scientometric Review. Computation 2019, 7(4), 63.
24. *Barkalov O., Titarenko L., and Barkalov Jr.*, A Structural Decomposition as a tool for the optimization of an FPGA–based implementation of a Mealy FSM. Cybernetics and Systems Analysis 2012, Vol. 48, N 2, pp. 313–322.
25. Vivado Design Suite. https://www.xilinx.com/products/design-tools/vivado.html,2020, acceed: January, 2020
26. *Баркалов А.А., Титаренко Л.А., Визор Я.Е., Матвиенко А.В.* Оптимальное кодиро-вание состояний в совмещенном автомате. Управляющие системы и машины. 2016. № 6. С. 34–39.
27. *Баркалов А.А., Титаренко Л.А., Визор Я.Е., Матвиенко А.В.* Уменьшение аппара-турных затрат в совмещенных автоматах. Управляющие системы и машины. 2017. № 4. С. 43–50.

Надійшла 14.01.2022

REFERENCES

1. *Sklyarov, V., Skliarova, I., Barkalov, A., Titarenko, L.*, 2014. “Synthesis and optimization of FPGA-based systems”. Berlin: Springer, 432 p. https://doi.org/10.1007/978-3-319-04708-9.
2. *Czerwinski, R., Kania, D.*, 2013. Finite state machines logic synthesis for complex programmable logic devices. Berlin: Springer, 172 p.
3. *Soloviev, V.V.*, 2001. Proyektirovaniye tsifrovykh skhem na osnove programmiruyemykh logicheskikh integralnykh skhem. [Design of digital circuits based on programmable logic integrated circuits], Hotline – TELECOM, Moscow, 636 p. (In Russian).
4. *Maxfield, C.*, 2004. The design warrior’s guide to FPGAs. Orlando: Academic Press, 542 p.
5. *Skliarova, I., Sklyarov, V., Sudnitson, A.*, 2012. Design of FPGA-based circuits using hierarchical finite state machines. Tallinn: TUT Press, 240 p.
6. *Baranov, S.*, 1994. Logic synthesis for control automata. Dordrecht: Kluwer Academic Publishers, 312 p. DOI: 10.1007/978-1-4615-2692-6_6.
7. *DeMicheli, G.,* 1994. Synthesis and optimization of digital circuits. New York: McGraw-Hill, 576 p.
8. *Kuon, I., Tessier, R., Rose, J.*, 2008. “FPGA Architecture: Survey and Challenges. Foundations and Trends”. Electronic Design Automation. vol. 2. No 2, pp. 135–253. DOI:10.1561/1000000005.
9. *Sass, R., Schmidt, A.*, 2010. Embedded System Design with platform FPGAs: Principles and Practices. Amsterdam: Morgan Kaufmann Publishers, 409 p.
10. UG473 (v1.14) July 3, 2019. [Online] Available at: <www.xilinx.com> [Accessed 2 May. 2021].
11. *Rafla, N.I., Gauba, I.A.*, 2010. “Reconfigurable Pattern Matching Hardware Implementation using On-Chip RAM-Based FSM”. 53rd IEEE International Midwest Symposium on Circuits and Systems. IEEE, August 2010, pp. 49–52. DOI: 10.1109/MWSCAS.2010.5548558.
12. *Tiwari, A., Tomko, K.*, 2004. “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). Vol. 2, pp. 916–921. DOI: 10.1109/DATE.2004.1269007.
13. *Glushkov, V.M.*, 1962. Cintez tsifrovykh avtomatov. M.: Fizmatgiz, 476 p. (In Russian).
14. *Kubica, M., Opara, A., Kania, D.*, 2021. Technology Mapping for LUT-based. FPGA. Berlin: Springer, 2021 DOI: 10.1007/978-3-030-60488-2_1.
15. Intel® FPGAs and Programmable Devices. [Online] Available at: <https://www.intel.com/content/www/us/en/ products/programmable.html> [Accessed 2 May. 2021].
16. *Yang, S.*, 1991. Logic synthesis and optimization benchmarks user guide. Version 3.0. Techn. Rep. Microelectronics Center of North Carolina, 43 p.
17. *Sklyarov, V.*, 2000. “Synthesis and Implementation of RAM-based Finite States Machines in FPGAs”. Proceeding of Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing. Villach: Springer-Verlag, pp 718–727.
18. *R. Senhaji-Navarro, I.Garcia-Vargas, G. Jimenes-Moreno, A. Civit-Balcells, and P. Guerra-Gutierres*, "ROM-based FSM implementation using input multiplexing in FPGA devices", Electronics Letters, vol. 40, N 20, pp. 1249–1251, 2004. DOI: 10.1049/el:20046007.
19 *Senhaji-Navarro, R. Garcia-Vargas, I. Guisado, L.J.*, 2012. "Performance evaluation of RAM-based implementation of Finite States Machines in FPGAs ", 19th IEEE International Conference on Electronics, Circuits, and Systems (ICECS 2012), Seville, Spain, Dec 2012, pp. 225–228.

20. *Yang, S.*, 1991. Logic synthesis and optimization benchmarks user guide. Version 3.0. Techn. Rep. Microelectronics Center of North Carolina, 43 p.
21. *Barkalov, A., Titarenko, L., Mielcarek, K.*, 2018. "Hardware reduction for LUT–based Mealy FSMs". International Journal of Applied Mathematics and Computer Science, pp. 595–607.
22. *Barkalov, A., Titarenko, L., Mielcarek, K.*, 2020. "Improving characteristics of LUT–based Mealy FSMs". International Journal of Applied Mathematics and Computer Science, 30(4), pp. 745–759.
23. *Ruiz-Rosero, J., Ramirez-Gonzalez, G. Khanna, R.*, 2019. Field Programmable Gate Array Applications–A Scientometric Review. Computation, 7(4), p. 63.
24. *Barkalov, O., Titarenko, L., Barkalov, Jr.*, 2012. "A Structural Decomposition as a tool for the optimization of an FPGA–based implementation of a Mealy FSM". Cybernetics and Systems Analysis, Vol. 48, N 2, pp. 313–322. https://doi.org/10.1007/s10559-012-9410-2.
25. Vivado Design Suite. [Online] Available at: <https://www.xilinx.com/products/design-tools/vivado.html> [Accessed 2 Jan. 2020].
26. *Barkalov, A.A., Titarenko, L.A., Vizor, Ya.E., Matviienko, A.V.,* 2016. "Optimalnoye kodirovaniye sostoyaniy v sovmeshchennom avtomate" ["Optimal coding of states in a combined automaton"], Upravlyayushchiye sistemy i mashiny, 6, pp. 34–39. DOI: 10.15407/usim.2016.06.034 (In Russian).
27. *Barkalov, A.A., Titarenko, L.A., Vizor, Ya.E., Matvienko, A.V.*, 2017. "Umensheniye apparaturnykh zatrat v sovmeshchennykh avtomatakh" ["Reduction of hardware costs in combined machines"], Upravlyayushchiye sistemy i mashiny, 4, pp. 43–50. DOI: 15407/usim.2017.04.043. (In Russian).

Received 14.01.2022

O.O. Barkalov, Doctor of Technical Sciences, Professor, Computer Science and Telecommunications University of Zielona Gora (Poland),
str. Podgorna, 50, Zielona Gora, 65-246, Poland, A.Barkalov@iie.uz.zgora.pl
L.O. Titarenko, Doctor of Technical Sciences, Professor, Computer Science and Telecommunications University of Zielona Gora (Poland),
str. Podgorna, 50, Zielona Gora, 65-246, Poland, L.Titarenko@iie.uz.zgora.pl
O.M. Golovin, Ph.D., Senior Researcher, Institute of Cybernetics of NAS of Ukraine,
03187, Kyiv, Glushkov Avenue, 40, Ukraine, o.m.golovin.1@gmail.com
O.V. Matvienko, Researcher Associate, Institute of Cybernetics of NAS of Ukraine,
03187, Kyiv, Glushkov Avenue, 40, Ukraine, matv@online.ua

OPTIMIZATION OF THE MEALY AUTOMATON CIRCUIT IN THE FPGA BASIS

Introduction. The ubiquitous penetration of digital systems into all spheres of human activity has made obvious the desire to reduce hardware costs. If for the implementation of combinational circuits there are standard solutions used in various CAD systems, then for many types of accumulating blocks, which include control devices (CU), there are no standard library solutions. CU circuits, which are a sequential circuit, must be designed first each time because, as a rule, these circuits differ in an irregular structure and a complex interconnection system. The characteristics of the control device have a significant impact on the quality of the digital system, and therefore the issue of designing a control device is a responsible process that, as a rule, requires solving the problem of reducing hardware costs. Methods for solving this problem depend on the features of both the architecture of the control device and the elemental basis.

Purpose. The main goal of this work is to reduce hardware costs and power consumption of control devices of digital systems by taking into account features of an element base of the control device and rational organization of micro-command addressing. FPGA (field-programmable logic array) microcircuits, widely used for the implementation of modern digital systems, were chosen as an elementary basis.

Methods. To evaluate the effectiveness of solving the problem, we used the methods of set theory, synthesis of automata, and software simulation using Xilinx Vivado CAD.

Results. The paper proposes a method for reducing hardware costs in the microprogram Mealy automaton (MPA) scheme implemented on an EMB and LUT basis. The method is based on the use of EMB to implement the logical condition replacement block. It is proposed to implement a part of the output signals (micro-operations) on EMB. The applicability conditions for this approach are shown. An example of the synthesis of MPA using the proposed method is given. All stages of the synthesis are analyzed in detail. Some alternative solutions are proposed and the conditions for their use are shown.

Conclusion. Studies have shown that when using one EMB block, our method gives a gain in the number of LUT elements. Comparisons were made with schemes obtained using the Auto, One-hot, and Sequential methods of the *Vivado* system.
Keywords: *Mealy machine, FPGA, EMB, LUT, structural decomposition, synthesis.*