https://doi.org/10.15407/intechsys.2025.01.024
UDC 681.513

О.О. БАРКАЛОВ, д-р техн. наук., професор,
Зеленогурський Університет,
вул. Ліцеальна, 9, Зелена Гура, Польща
https://orcid.org/orcid.org/0000-0002-4941-3979
A.Barkalov@iie.uz.zgora.pl

Л.О. ТІТАРЕНКО, д-р техн. наук., професор,
Зеленогурський Університет,
вул. Ліцеальна, 9, Зелена Гура, Польща,
Харківський національний університет радіоелектроніки,
пр. Науки, 14, м. Харків, 61166, Україна
https://orcid.org/0000-0001-9558-3322
L.Titarenko@iie.uz.zgora.pl

О.М. ГОЛОВІН, канд. техн. наук, старш. наук. співроб.,
Інститут кібернетики імені В.М. Глушкова НАН України,
просп. Акад. Глушкова, 40, м. Київ, 03187, Україна
https://orcid.org/0000-0002-0279-812X
o.m.golovin.1@gmail.com

О.В. МАТВІЄНКО, наук. співроб.,
Інститут кібернетики імені В.М. Глушкова НАН України,
просп. Акад. Глушкова, 40, м. Київ, 03187, Україна
https://orcid.org/0000-0003-1838-1422
avmatv@ukr.net

С.О. САБУРОВА, канд. техн. наук, доцент,
Харківський національний. університет радіоелектроніки,
пр. Науки, 14, м. Харків, 61166, Україна
https://orcid.org/0000-0001-6286-1648
sabsvet@gmail.com

ОПТИМІЗАЦІЯ ДВОРІВНЕВОЇ СХЕМИ АВТОМАТА МІЛІ У БАЗИСІ *FPGA*

Вступ. *Однією з найважливіших частин будь-якої цифрової системи є пристрій управління (ПУ), який координує взаємодію інших блоків системи. Зазвичай, схема ПУ визначається алгоритмом керування, а проєктування кожного ПУ починається від початку через унікальність алгоритму його роботи.*

Цитування: Баркалов О.О., Тітаренко Л.О, Головін О. М., Матвієнко О.В., Сабурова С.О. Оптимізація дворівневої схеми автомата Мілі у базисі *FPGA*. *Information Technologies and Systems*, Київ, 2025, Том 1 (1), 24–38. https://doi.org/10.15407/intechsys.2025.01.024

Від оптимальності характеристик ПУ залежить якість цифрової системи. Тому розробка ефективних методів оптимізації схем ПУ є настільки важливою. При синтезі схеми ПУ виникає ряд проблем оптимізації: зменшення площі мікросхеми ПУ, підвищення продуктивності, зниження енергоспоживання. Відомо, що вирішення першої з цих задач дає змогу покращити інші характеристики схеми.

Мета роботи: *розглянути проблему і запропонувати метод зменшення площі мікросхеми при реалізації схеми ПУ з використанням мікросхем FPGA (field-programtable logic array).*

Методи. *Об'єктами дослідження обрано мікросхеми FPGA та модель мікропрограмного автомата (МПА) Мілі. При реалізації схеми МПА в базисі FPGA використовують табличні елементи LUT (look-up table) і вбудовані блоки пам'яті (EMB). Оскільки домінуючим виробником мікросхем FPGA є AMD Xilinx, запропонований у статті метод орієнтований на FPGA цієї компанії.*

Результати. *Запропоновано спосіб зниження витрат при реалізації схеми МПА Мілі в базисі FPGA. Метод заснований на спільному використанні вбудованих блоків памяті EMB і елементів LUT. Граничним вважається випадок, коли розробник може використовувати лише один блок EMB. Для оптимізації схеми використовуються методи заміни входів МПА і подвійного кодування станів. Запропонований спосіб дозволяє зменшити кількість використовуваних елементів LUT до 18 %. Наведено приклад застосування запропонованого методу. Показано умови доцільності використання методу.*

Висновки. *Проведене дослідження показало, що є сенс модифікувати запропонований метод для випадку автоматів Мура.*

Ключові слова: *автомат Мілі, синтез, FPGA, EMB, LUT, кодування станів, заміна входів.*

Вступ

В наш час широко впроваджуються цифрові системи у різні сфери людської діяльності [1]. До цифрових систем належать послідовні пристрої [2], одним з важливих типів яких є пристрої управління (ПУ), які координують роботу інших блоків системи [3, 4]. Зазвичай, схему ПУ завжди визначає алгоритм управління, а проєктування кожного ПУ починається від самого початку через унікальність його алгоритму [5]. Якість системи багато в чому залежить від оптимальності показників ПУ [4]. Тому важливим є розроблення ефективних методів оптимізації схем ПУ.

Поведінка ПУ часто визначається з допомогою моделі мікропрограмного автомата (МПА) Мілі [2, 6]. Методи оптимізації схеми МПА багато в чому залежать від елементного базису [4]. Найбільш популярними для реалізації ПУ є мікросхеми *FPGA* (*Field-Programmable Logic Array*) [7, 8]. Так, наприклад, у літературі наводиться близько 1400 різних застосувань *FPGA* [9]. Тому ми вибрали базис *FPGA* та МПА Мілі як об'єкти дослідження у цій статті.

При реалізації схем МПА в базисі *FPGA* використовуються два типи конфігурованих логічних блоків (КЛБ): табличні елементи *LUT* (*look-up table*) і вбудовані блоки пам'яті *EMB* (*Embedded Memory Blocks*) [8, 10]. Блоки КЛБ різних виробників мають деякі відмінності. Нині домінуючим виробником мікросхем *FPGA* є фірма *AMD Xilinx* [11]. У зв'язку з цим запропонований у статті метод орієнтований на *FPGA* фірми *AMD Xilinx*.

Можливості кожного із типів КЛБ різні. Блок *EMB* дає змогу реалізувати систему булевих функцій (СБФ). Блок *LUT* реалізує довільну функцію логіки алгебри. При цьому найкращі результати досягаються за одночасного використання *EMB* та *LUT* для реалізації схем МПА [12, 13]. Такий підхід дозволяє зменшити площу *FPGA*, час циклу і споживану потужність порівняно з цими характеристиками гомогенної схеми [14]. Саме такий випадок розглядається у статті.

Реалізація схем МПА за допомогою ресурсів мікросхем *FPGA*

Автомат Мілі задається як вектор $S = <A, X, Y, \delta, \lambda, a_1>$ [3]. Тут $X = \{x_1, ..., x_L\}$ — множина внутрішніх станів, $Y = \{y_1, ..., y_N\}$ — множина виходів ($y_n, x_l \in \{0, 1\}$, $n = \overline{1, N}, l = \overline{1, L}$). Функція переходів δ ставить у відповідність кожній парі $<a_m, X_h>$ стан переходу $a_s \in A$. Тут X_h — вхідний сигнал, який дорівнює кон'юнкції деяких змінних $x_1 \in X$, і визначає перехід $<a_m, a_s>$. Функція виходів λ ставить у відповідність кожній парі $<a_m, X_h>$ вихідний сигнал $Y_h \subseteq Y$. Кількість переходів МПА дорівнює H. Автомат функціонує у дискретному часі, хід якого задається імпульсом *Clock*. У початковий момент $t = 0$ імпульс *Start* встановлює МПА у початковий стан.

Усі компоненти вектора S можуть бути подані як таблиця переходів [6]. Ця таблиця має стовпці a_m (вихідний стан), a_s (стан переходу), X_h (вхідний сигнал, який ініціює перехід $<a_m, a_s>$), Y_h (вихідний сигнал на цьому переході), h (номер переходу). Для синтезу схеми МПА необхідно [2]:

1. закодувати стани $a_m \in A$ МПА кодами розрядності R:

$$R = \lceil \log_2 M \rceil; \tag{1}$$

2. сформувати множини кодувальних змінних $T = \{T_1, ..., T_R\}$ та функцій збудження пам'яті (ФЗП) $D = \{D_1, ..., D_R\}$;

3. побудувати пряму структурну таблицю (ПСТ) автомата, куди входять коди станів $a_m, a_s \in A$ і ФЗП;

4. на базі ПСТ побудувати систему функцій:

$$D = D(T, X), \tag{2}$$

$$Y = Y(T, X). \tag{3}$$

Ми розглядаємо реалізацію СБФ (2)—(3) у базисі *FPGA*. Для реалізації використовуються такі внутрішні ресурси мікросхеми [14]: елементи *LUT*, блоки *EMB*, вбудовані мультиплексори, тригери, що програмуються, дерево синхронізації, програмовані входи—виходи. Розглянемо особливості цих елементів.

Елемент *LUT* є мережею однорозрядних осередків пам'яті *SRAM*, пов'язаних системою мультиплексорів [15]. Кожен *LUT* має S_L адресних входів та 2^{S_L} осередків пам'яті. Один *LUT* може реалізувати довільну

логічну функцію f_i з числом аргументів від 1 до S_L. Позначимо блок, що складається з елементів *LUT*, символом *LUTer*.

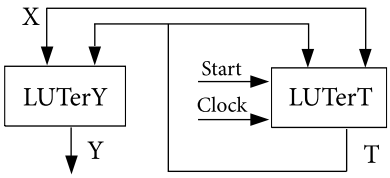

Рис. 1. Структурна схема МПА U_1

Блок КЛБ складається з кількох елементів *LUT*, виходи яких з'єднуються із входами тригерів. При цьому на вихід КЛБ можна передати або вихід *LUT* (комбінаційна логіка), або вихід тригера (пам'ять) [15]. Ці тригери використовуються в організації регістру станів (*RG*), який має входи типу *D* [14]. На Рис. 1 показана структурна схема МПА U_1, побудована тільки на елементах *LUT*.

У автоматі U_1 блок *LUTerY* реалізує СБФ (3), блок *LUTerT* — СБФ (2). Блок *LUTerT* містить також регістр *RG*, яким керують сигнали *Start* і *Clock*.

Нехай функція $f_i \in D \cup Y$ залежить від $NA(f_i)$ аргументів, де $NA(f_i) \le T + L$. Якщо виконується умова

$$NA(f_i) \le S_L, \tag{4}$$

то схема U_1 має точно $R+L$ елементів. Ця схема має один рівень логіки і є оптимальною за кількістю елементів і міжз'єднань, швидкодією, споживаною потужністю тощо [4]. У разі невиконання умови (4), характеристики схеми різко погіршуються. У цьому разі необхідно застосувати методи функційної [16] та/або структурної декомпозиції [4].

Блок *EMB* — це пам'ять, яка має S_A входів та t_F виходів. Число комірок пам'яті може змінюватися, що пов'язане зі зміною параметрів S_A та t_F. При цьому ємність пам'яті (V_0) не змінюється:

$$V_0 = 2^{S_A} \times t_F. \tag{5}$$

Кожна пара $<S_A, t_F>$, яка відповідає умові (5), визначає допустиму конфігурацію *EMB* [8, 12]. Нехай для певної конфігурації $<S_0, t_0>$ виконуються умови

$$S_0 \ge L + R; \tag{6}$$

$$t_0 \ge N + R. \tag{7}$$

Тоді схема МПА реалізується одним блоком *EMB*.

Коли умова (7) порушується, схема МПА має один рівень, і складається з $n_1 = \lceil (N + R) / t_0 \rceil$ блоків *EMB*. Якщо розробник має у своєму розпорядженні менше ніж n_1 блоків, то схема реалізується в гетерогенному базисі. Якщо умова (6) порушується, схема МПА також реалізується в базисі *EMB* і *LUT* [12].

Розглянемо випадок, коли (6) виконується, (7) порушується і необхідна кількість елементів *EMB* відсутня. Нехай маємо граничний

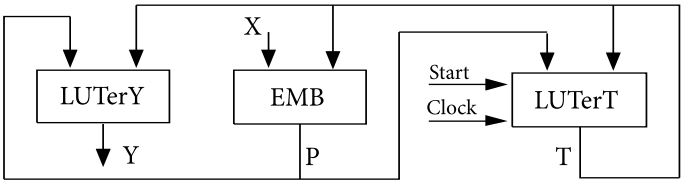

Рис. 2. Структурна схема МПА U_1

випадок, коли є лише один блок *EMB*. Ми пропонуємо використовувати цей блок для заміни вхідних змінних [4, 13].

Нехай переходи із кожного стану $a_m \in A$ залежать від L_m входів $x_1 \in X$. Знайдемо число G:

$$G = \max(L_1, ..., L_M), \tag{8}$$

яке визначає кількість додаткових змінних $p_g \in P$, що замінюють входи.

Нехай виконується умова

$$2^{R+L} \cdot G \le V_0. \tag{9}$$

У цьому випадку блок *EMB* може реалізувати СБФ:

$$P = P(T, X) \tag{10}$$

і реалізувати заміну множини X множиною $P = \{p_1, ..., p_G\}$.

Заміна $X \to P$ веде до заміни СБФ (2) — (3) системами

$$D = D(T, P); \tag{11}$$

$$Y = (T, P). \tag{12}$$

Системи (11 — 12) реалізуються з допомогою елементів *LUT*. Це веде до МПА U_2 (Рис. 2). У МПА U_2 блок EMB реалізує СБФ (10), *LUTerT* — СБФ (11) та *LUTerY* — СБФ (12).

Якщо умова (4) виконується для функцій $f_i \in D \cup Y$, то схема U_2 складається з $R + N$ елементів *LUT*. У разі порушення цієї умови схема має багаторівневу структуру. Це веде до погіршення показників схеми. У цій роботі ми розглядаємо саме таку ситуацію.

Основна ідея запропонованого методу

Якщо блоки *LUTerY* та/або *LUTerT* є багаторівневими схемами, ми пропонуємо використовувати подвійне кодування станів [4]. За певних умов такий підхід веде до дворівневих схем цих блоків та регулярної системи міжз'єднань.

Нехай для деякого МПА Мілі виконано заміну входів та визначено параметр G. Знайдемо розбиття множини A на мінімальну кількість класів K: $\Pi_A = \{A^1, ..., A^K\}$. Клас $A^k \in \Pi_A$ містить M_k станів. Параметр M_k вибирається так, щоб виконувалася умова

$$G + \lceil \log_2(M_k + 1) \rceil \le S_L. \tag{13}$$

Другий додаток лівої частини (13) є розрядністю кодів станів $C(a_m) \neq \neq K(a_m)$ всередині класу $A^k \in \Pi_A$ Зрозуміло, що виконується умова $R_k = S_L - G$. При цьому число класів K визначається, як $K = \left\lceil M / 2^{R_k} \right\rceil$.

Розбиття Π_A будується тривіальним способом, оскільки будь-які M_k станів можуть бути включені до будь-якого класу. Клас $A^k \in \Pi_A$ визначає множину Y^k виходів $y_n \in Y$, що формуються при переходах зі станів $a_m \in A$. Для зменшення кількості елементів *LUT* необхідно формувати розбиття множини станів так, щоб мінімізувати сумарне число однакових виходів у різних множинах $Y^k \subseteq Y$. Це можна зробити з допомогою способу [4].

Крім того, кожен клас $A^k \in \Pi_A$ визначає множину $D^k \subseteq D$. Ця множина включає ФЗП, які формуються при переходах зі станів $a_m \in A^k$. Необхідно формувати Π_A так, щоб однакові ФЗП входили в мінімальне число множин $D^k \subseteq D$. Задля вирішення цієї задачі також застосуємо метод подвійного кодування станів [4].

Якщо $a_m \in A^k$, для формування кодів $C(a_m)$ використовуються змінні з множин τ^k. Ці множини утворюють множину $\tau = \tau^1 \cup \tau^2 \cup ... \cup \tau^K$. У загальному випадку множина τ містить R_A елементів: $R_A = R_1 + R_2 + ... + R_K$.

Цілком можливо, що переходи зі станів $a_m \in A^k$ залежать тільки від частини додаткових змінних, тобто від елементів множини $P_K \subseteq P$. Змінні з $p_g \in P^k$ і $\tau_r \in \tau^k$ використовуються для формування часткових функцій:

$$D^k = D^k(\tau^k, P^k); \tag{14}$$

$$Y^k = Y^k(\tau^k, P^k). \tag{15}$$

Часткові функції (14)—(15) необхідно перетворити на функції $D_r \in D$ та $y_n \in Y$. Ці функції є диз'юнкцією часткових функцій:

$$D_r = D_r^1 \vee D_r^2 \vee ... \vee D_r^K, \tag{16}$$

$$y_n = y_n^1 \vee y_n^2 \vee ... \vee y_n^K. \tag{17}$$

Для формування змінних $\tau_r \in \tau$ необхідно перетворити коди $K(a_m)$ на коди $C(a_m)$. Для цього потрібно реалізувати СБФ

$$\tau = \tau(T). \tag{18}$$

Системи (10), (14)—(18) визначають пропонований у цій роботі МПА U_3 (Рис. 3).

Принцип функціонування МПА U_3 зрозумілий із попереднього матеріалу. Блок *EMB* реалізує СБФ (10). Блоки *LUTer1 — LUTerK* реалізують СБФ (14)—(15). Блок *LUTerYT* перетворює часткові функції, реалізуючи СБФ (16)—(17). Цей блок також містить прихований регістр *RG*, який зберігає коди $K(a_m)$. Тригери *RG* управляються імпульсами *Start* та *Clock*. Блок *LUTer*τ перетворює змінні $T_r \in T$ на змінні $\tau_r \in \tau$. З цією метою він реалізує СБФ (18).

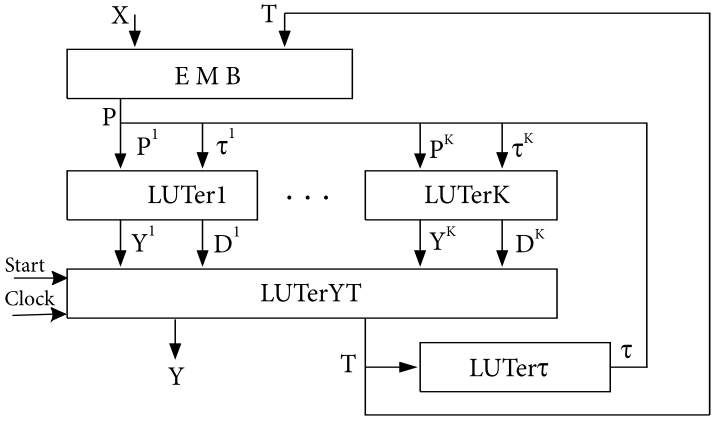

Рис. 3. Структурна схема МПА U_3

У кожному такті лише один із блоків *LUTerK* є активним. На виходах інших блоків другого рівня логіки формуються нульові значення часткових функцій. Це досягається надходженням на входи блоків тих кодів, що відповідають співвідношенню $a_m \notin A^k$. З цією метою використовуються часткові коди, що містять лише нулі.

У кожному такті лише один із блоків *LUTerK* є активним. На виходах інших блоків другого рівня логіки формуються нульові значення часткових функцій. Це досягається надходженням на входи блоків тих кодів, що відповідають співвідношенню $a_m \notin A^k$. З цією метою використовуються часткові коди, що містять лише нулі.

Нехай пропонований автомат U_3 задано таблицею переходів. Тоді метод його синтезу містить такі етапи:

1) кодування станів кодами $K(a_m)$;
2) заміна змінних $x_l \in X$ елементами множини P;
3) формування таблиці *EMB*;
4) формування розбиття Π_A з мінімальним числом K;
5) формування таблиць блоків *LUTer*1 — *LUTerK*;
6) формування СБФ (14) — (15);
7) формування таблиці блока *LUTerYT* та СБФ (16) — (17);
8) формування таблиці *LUTer*τ та СБФ (18);
9) реалізація схеми МПА у заданому базисі.

Приклад синтезу МПА U_3 із застосуванням пропонованого методу

Нехай для реалізації схеми МПА використовується блок *EMB* з такими конфігураціями: <12,1>, <11,2>,..., <8,16>. Крім того, розробник має елементи *LUT* з $S_L = 5$. Розглянемо приклад використання моделі U_3 для синтезу МПА S_1 (табл. 1).

Як випливає з табл. 1 автомат S_1 характеризується параметрами $M = 9, L = 7, N = 12$. Ці характеристики визначають множини $A = \{a_1, ..., a_9\}$,

$X = \{x_1, ..., x_7\}$ і $Y = \{y_1, ..., y_{12}\}$. З (1) маємо $R = 4$, що дає множини $T = \{T_1, ..., T_4\}$ і $D = \{D_1, ..., D_4\}$. Отже виконується співвідношення $R \le S_L$, блок *LUTer*τ складається з R_A елементів *LUT* і є однорівневим, а коди станів $K(a_m)$ не впливають на число елементів *LUT* в схемі блока *LUTer*τ.

Виконаємо кодування станів тривіальним способом: код стану дорівнює двійковому еквіваленту його індексу, що дає коди $K(a_1) = 0000$, $K(a_2) = 0001$,..., $K(a_9) = 1000$. Зазначимо, що стани можна закодувати так, щоб оптимізувати число літералів СБФ (18). Такий підхід зменшує кількість міжз'єднань у схемі.

Для заміни $X \to P$ застосуємо алгоритм [4]. Відповідно до табл. 1, максимальне значення $L_m = 2$. Використовуючи (8), отримаємо $G = 2$ та $P = \{p1, p2\}$. Перевіримо можливість використання *EMB* для заміни $X \to P$.

Маємо $R + L = 11$ та $G = 2$. Серед конфігурацій *EMB* є пара <11,2>, що ідеально підходить для автомата S_1. Це випливає з умов (5–7). Таблицю заміни $X \to P$ можна побудувати без будь-якої оптимізації. Для нашого випадку заміну подано у табл. 2.

Таблиця 1. **Таблиця переходів автомата Мілі S_1**

a_m	a_s	X_h	Y_h	H	A^K
a_1	a_2	x_1	y_1y_4	1	A^1
	a_3	$\overline{x_1}x_2$	y_2y_5	2	
	a_4	$\overline{x_1}\overline{x_2}$	y_8y_{11}	3	
a_2	a_5	x_3	y_1y_6	4	A^1
	a_6	$\overline{x_3}x_4$	y_7	5	
	a_7	$\overline{x_3}\overline{x_4}$	y_3y_{11}	6	
a_3	a_8	x_5	y_{12}	7	A^1
	a_9	$\overline{x_5}$	y_4y_5	8	
a_4	a_9	x_5x_6	y_4	9	A^1
	a_5	$x_5\overline{x_6}$	$y_{11}y_{12}$	10	
	a_6	$\overline{x_5}$	y_1y_5	11	
a_5	a_1	1	y_6y_7	12	A^1
a_6	a_7	x_7	y_3y_9	13	A^2
	a_8	$\overline{x_7}$	y_8y_{10}	14	
a_7	a_1	1	–	15	A^2
a_8	a_9	x_1x_7	y_1y_6	16	A^1
	a_2	$x_1\overline{x_7}$	y_3y_7	14	
	a_4	x_1	y_6y_7	18	
a_9	a_3	x_6	y_9y_{10}	19	A^2
	a_7	$\overline{x_6}$	y_8	20	

Таблицю блока *EMB* побудовано з урахуванням табл. 2. Вона містить такі стовпці: T_1, T_R, x_L, …, x_1 — адреси комірок пам'яті, p_1, p_2 — вміст комірки, q — десятковий еквівалент адреси комірки. Перші 12 рядків таблиці *EMB* для МПА S_1 подано в табл. 3.

У загальному випадку таблиця блоку *EMB* має H_E рядків, де $H_E = 2^{R+L}$. У нашому випадку $H_E = 2^{11} = 2048$. При цьому $q \in \{0, ..., 2047\}$. Переходи з кожного стану подані в таблиці, яка має H_m рядків, де $H_m = 2^L$. В нашому прикладі $H_m = 128$.

У табл. 3 наведено лише 12 зі 128 рядків, що відтворюють переходи зі стану $a_1 \in A$. У табл. 3 додано стовпець h для того, щоб показати зв'язок між таблицями.

Для нашого прикладу $G = 2$ і з (13) випливає, що граничним значенням $R_k \in S_L - G = 3$. Таким чином, кожен блок $A^k \in \Pi_A$ може містити до 7 станів. Побудуємо це розбиття так, щоб кожен із двох блоків містив максимальну кількість унікальних виходів $y_n \in Y$. З цією метою використаємо метод із [4].

Використання [4] дає розбиття $\Pi_A = \{A^1, A^2,\}$, де $A^1 = \{a_1, ..., a_5, a_8,\}$ і $A^2 = \{a_6, a_7, a_9,\}$. З табл. 1 маємо множини Y^1 і Y^2: $Y^1 = \{y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_{11}, y_{12}\}$, $Y^2 = \{y_3, y_9, y_{10}, y_8\}$. Перетин Y^1 і Y^2 дає множину $\{y_3, y_8\}$, отже множина Y^1 містить 8 унікальних виходів, а множина Y^2 — два. Унікальні виходи реалізуються на елементах *LUT* блоків *LUTerk*, загальні — на елементах *LUT* блоку *LUTeYT*. У найгіршому випадку для реалізації кожної вихідної функції необхідно 3 елементи *LUT*

Таблиця 2. **Заміна входів МПА S_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_3	x_5	x_5	–	x_7	–	x_1	–
p_2	x_2	x_4	–	x_6	–	–	–	x_7	x_6

Таблиця 3. **Фрагмент таблиці блока *EMB* автомата S_1**

T_1	T_2	T_3	T_4	x_7	x_6	x_5	x_4	x_3	x_2	x_1	p_1	p_2	Q	H
0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	0	0	1	1	0	1	1
0	0	0	0	0	0	0	0	0	1	0	0	1	2	2
0	0	0	0	0	0	0	0	0	1	1	1	1	3	1
0	0	0	0	0	0	0	0	1	0	0	0	0	4	3
0	0	0	0	0	0	0	0	1	0	1	1	0	5	1
0	0	0	0	0	0	0	0	1	1	0	0	1	6	2
0	0	0	0	0	0	0	0	1	1	1	1	1	7	1
0	0	0	0	0	0	0	1	0	0	0	0	0	8	3
0	0	0	0	0	0	0	1	0	0	1	1	0	9	1
0	0	0	0	0	0	0	1	0	1	0	0	1	10	2
0	0	0	0	0	0	0	1	0	1	1	1	1	11	1

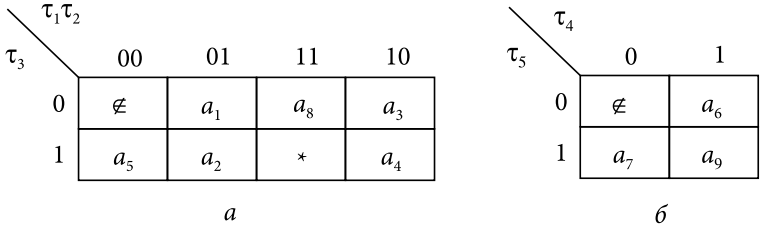

Рис. 4. Коди станів $C(a_m)$ для МПА S_1

(це необхідно також і для y_3 та y_8), Таким чином, у найгіршому випадку необхідно витратити $3N = 36$ елементів *LUT*. Завдяки використанню методу [4] ця частина схеми вимагає лише $(N - 2) + 3 \times 2 = 16$ елементів *LUT*.

Як випливає з (13), число кодуючих змінних визначається як $[\log_2(H_k+1)]$. У нашому випадку маємо $R1 = 2$ та $R2 = 2$. Це дає $\tau^1 = \{\tau_1, \tau_2, \tau_3\}$, $\tau^2 = \{\tau_4, \tau_5,\}$ і $\tau = \{\tau_1, ..., \tau_5,\}$. Закодуємо стани, як показано на рис. 4, *а* (для A^1) і рис. 4, *б* (для A^2).

Таблиця блока *LUTerk* будується з урахуванням таблиці переходів і кодів $C(a_m)$, $K(a_m)$. Таблиця містить такі стовпці: a_m, $C(a_m)$, a_s, $K(a_s)$, P_h^k, Φ_h^k, Y_h^k, h. Блок *LUTer*1 подано у табл. 4, а блок *LUTer*2 — у табл. 5.

Таблиці 4 і 5 є основою для формування СБФ (14)–(15). Кожна з функцій реалізується на одному елементі *LUT*. Оптимізація кожної з функцій має сенс, якщо тільки деякі аргументи виключаються з усіх її термів [4]. Так блок *LUTer*2 може бути поданий такими СБФ:

Таблиця 4. **Таблиця блока LUTer1**

a_m	$C(a_m)$	a_s	$K(a_s)$	P_h^1кор	Φ_h^1кор	y_h^1 кор	h
a_1	010	a_2	0001	p_1	D_4	$y_1 y_4$	1
		a_3	0010	$\overline{p_1} p_2$	D_3	$y_2 y_5$	2
		a_4	0011	$\overline{p_1}\overline{p_2}$	$D_3 D_4$	$y_8 y_{11}$	2
a_2	011	a_5	0100	p_1	D_2	$y_1 y_6$	4
		a_6	0101	$\overline{p_1} p_2$	$D_2 D_4$	y_7	5
		a_7	0110	$\overline{p_1}\overline{p_2}$	$D_2 D_3$	$y_3 y_{11}$	6
a_3	100	a_8	0111	p_1	$D_2 D_3 D_4$	y_{12}	7
		a_9	1000	p_1	D_1	$y_4 y_5$	8
a_4	101	a_3	0010	$p_1 p_2$	D_3	y_4	9
		a_5	0100	$p_1 \overline{p_2}$	D_2	$y_{11} y_{12}$	10
		a_6	0101	p_1	$D_2 D_4$	$y_1 y_5$	11
a_5	100	a_1	0000	1	–	$y_6 y_7$	12
a_8	110	a_9	1000	$p_1 p_2$	D_1	$y_1 y_6$	13
		a_2	0001	$p_1 \overline{p_2}$	D_4	$y_3 y_7$	14
		a_4	0011	$\overline{p_1}$	$D_3 D_4$	$y_6 y_7$	15

$$D_2^2 = \tau_4 \overline{\tau_5} \vee \tau_4 \tau_5 p_2; \quad D_3^2 = \tau_4; \quad D_4^2 = \tau_4 \overline{\tau_5}\,\overline{p_1}.$$

Як видно, тут функції D_2^2 і D_4^2 залежать від трьох змінних, а функція D_3^2 — від однієї. Тому для реалізації функції D_3^2 не потрібний окремий *LUT* (це просто вихід тригера, що формує змінну τ_4).

$$y_3^2 = \tau_4 \overline{\tau_5} p_1; \quad y_8^2 = \tau_4 \overline{\tau_5}\,\overline{p_1} \vee \tau_4 \tau_5 \overline{p_2};$$

$$y_9^2 = \tau_4 \overline{\tau_5} p_1 \vee \tau_4 \tau_5 p_2 = y_9; \quad y_{10}^2 = \tau_4 \overline{\tau_5}\,\overline{p_1} \vee \tau_4 \tau_5 p_2 = y_{10}.$$

У цій СБФ лише функція y_3^2 залежить від $G + R = 4$ змінних. У рівняннях для y^2_9 і y^2_{10} ми підкреслили, що ці функції не потребують елементів *LUT* блоку *LUTerYT*. Подібним способом можна знайти системи (14)—(15) для блоку *LUTer*1.

Таблиця блока *LUTerYT* будується тривіальним способом. Вона містить стовпці $D_1,..., D_R, y_1,..., y_N$ і рядки *LUTer*1, ..., *LUTerK*. Якщо якась функція формується блоком *LUTerk*, то це відзначається символом «+» у відповідній клітині. Інакше там стоїть знак «–». Для нашого прикладу блок *LUTerYT* задано табл. 6.

З табл. 6 маємо СБФ блоку *LUTerYT*

$$D_1 = D_1^1; \quad D_2 = D_2^1 \vee D_2^2; \quad D_3 = D_3^1 \vee D_3^2; \quad D_4 = D_4^1 \vee D_4^2;$$

$$y_1 = y_1^1; \quad y_2 = y_2^1; \quad y_3 = y_3^1 \vee y_3^2; \quad y_4 = y_4^1; \quad y_5 = y_5^1; \quad y_6 = y_6^1;$$

$$y_7 = y_7^1; \quad y_8 = y_8^1 \vee y_8^2; \quad y_9 = y_9^1; \quad y_{10} = y_{10}^1; \quad y_{11} = y_{11}^1; \quad y_{12} = y_{12}^1.$$

З табл. 6 і цієї СБФ випливає, що *LUTerYT* складається з 5-ти *LUT*. У загальному випадку необхідно $R + N = 16$ *LUT*. Економія числа *LUT* у 3,2 рази пов'язана з видом розбиття Π_A.

Таблиця 5. **Таблиця блока *LUTer*2**

a_m	$C(a_m)$	a_s	$K(a_s)$	P_h^2	Φ_h^2	y_h^1	h
a_6	10	a_7	0100	p_1	$D_2 D_3$	$y_3 y_9$	1
		a_8	0001	$\overline{p_1}$	$D_2 D_3 D_4$	$y_8 y_{10}$	2
a_7	01	a_1	0100	1	–	–	2
a_9	11	a_3	0010	p_2	D_3	$y_9 y_{10}$	4
		a_7	0110	$\overline{p_2}$	$D_2 D_3$	y_8	5

Таблиця 6. **Таблиця блока *LUTerYT***

Блок	Функція															
	D_1	D_2	D_3	D_4	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}
*LUTer*1	+	+	+	+	+	+	+	+	+	+	+	+	–	–	+	+
*LUTer*2	–	+	+	+	–	–	+	–	–	–	–	+	+	+	–	–

T_3T_4 \ T_1T_2	00	01	11	10
00	τ_2 (1)	τ_3 (5)	*	$\tau_4\tau_5$ (9)
01	$\tau_2\tau_3$ (2)	τ_4 (6)	*	*
11	$\tau_1\tau_3$ (4)	$\tau_1\tau_2$ (8)	*	*
10	τ_1 (3)	τ_5 (7)	*	*

Рис. 5. Карта Карно для блока *LUTer*т

Таблиця блока *LUTer*т ставить у відповідність кожному коду $K(a_m)$ код $C(a_m)$. Задамо цю таблицю як карту Карно для R_A функцій (рис. 5).

З карти Карно (Рис. 5) маємо таку СБФ:

$$\tau_1=\overline{T_2}T_3 \vee T_2T_3T_4;\ \ \tau_2=\overline{T_1}\,\overline{T_2}\,\overline{T_3} \vee T_2T_3T_4;$$
$$\tau_3=\overline{T_2}T_3 \vee T_2\overline{T_3}\,\overline{T_4};\ \ \tau_4=T_1\overline{T_3} \vee T_2\overline{T_3}\,\overline{T_4}; \qquad (19)$$
$$\tau_5=T_1 \vee T_2T_3\overline{T_4}.$$

Як видно з (19), функції τ_1 і τ_3 мають по 3 аргументи.

Оцінимо число *LUT* у схемі МПА S_1. Як випливає з табл. 4 блок *LUTer* — складається з 14 елементів *LUT*, а блок *LUTer*2 — з 6. З табл. 6 випливає, що блок *LUTerYT* складається з 5 елементів. З системи (19) випливає, що *LUTer*т складається з 5 елементів. Отже, схема МПА S_1 з урахуванням моделі U_3 складається з 20 елементів *LUT*.

Висновки

Зменшення площі схеми МПА можна досягти збільшенням функційних можливостей застосовних елементів. У разі застосування схем, що реалізуються в базисі *FPGA*, зменшення площі можна досягти переходом від блоків *LUT* до блоків *EMB* [14]. Такий перехід призводить до зменшення часу циклу та споживаної потужності [4, 14]. Однак блоки *EMB* також мають обмеження за кількістю входів та виходів. Відомо, що багаторівневі схеми, які складаються з блоків *EMB*, є вкрай неефективними [4]. Тому виникає завдання реалізації схеми МПА у гетерогенному базисі блоків *EMB* та елементів *LUT*.

У нашій статті розглядається граничний випадок, коли лише один блок *EMB* може бути використаний, і його можливостей не вистачає для реалізації схеми МПА. Пропонований метод заснований на такій ідеї: блок *EMB* виконує заміну входів МПА додатковими змінними $p_g \in P$, а інші СБФ реалізуються на елементах *LUT*. Для оптимізації цієї частини схеми використовуємо метод подвійного кодування станів [4].

Наші дослідження показали, що такий підхід дає змогу значно (до 18 %) зменшити кількість *LUT* порівняно з відомими методами. Дослідження проводилися з використанням стандартних МПА [18] і САПР *Vivado* [19]. Зазначимо, що такий підхід призводить до значного зменшення числа *LUT* лише у разі виконання умови (9). Однак для 12 % автоматів ця умова порушується [18]. Таким чином, умова (9) визначає доцільність застосування запропонованого методу.

Метою подальших досліджень є модифікація запропонованого підходу з урахуванням особливостей автомата Мура [20, 21]. Крім того, необхідно розробити метод оптимізації схеми МПА Мілі за ситуації, коли можливостей одного блока *EMB* недостатньо для реалізації схеми заміни входів.

ЛІТЕРАТУРА / REFERENCES

1. Marwedel P. *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things.* Springer International Publishing, 3rd ed., 2018.
2. Minns P., Elliot I. *FSM-based digital design using Verilog HDL.* John Wiley and Sons, 2008. https://doi.org/10.1002/9780470987629
3. Baranov S. *Finite State Machines and Algorithmic State Machines.* Amazon, 2018.
4. Barkalov A., Titarenko L., Mielcarek K., Chmielewski S. Logic Synthesis for FPGA-Based Control Units — Structural Decomposition in Logic Design. *Lecture Notes in Electrical Engineering,* Springer, Berlin, 2020, Vol. 636. https://doi.org/10.1007/978-3-030-38295-7
5. Gazi O., Arli A. *State Machines using VHDL: FPGA Implementation of Serial Communication and Display Protocols.* Springer, Berlin, 2021. https://doi.org/10.1007/978-3-030-61698-4
6. De Micheli G. *Synthesis and Optimization of Digital Circuits.* McGraw-Hill, 1994.
7. Trimberg S. Three ages of FPGA: A retrospective on the first thirty years of FPG Atechnology. *IEEE Proceedings,* 2015, Vol. 103 (3), 318–331. https://doi.org/10.1109/JPROC.2015.2392104
8. Wolf W. *FPGA-Based System Design.* Prentice Hall PTR, Upper Saddle River, New Jersey, USA, 2004.
9. Ruiz-Rosero, Juan, Gustavo Ramirez-Gonzalez, Rahul Khanna. Field Programmable Gate Array Applications—A Scientometric Review. *Computation 7,* 2019, Vol. 4, 63 p. https://doi.org/10.3390/computation7040063
10. Maxfield C. *FPGAs: Instant access.* Newnes, 2008.
11. Xilinx. URL: https://www.amd.com/en.html [Accessed Jan. 2025]
12. Garcia-Vargas I., Senhadji-Navarro R., Jimnez-Moreno G., Civit-Balcells A., GuerraGutierrezz P. ROM-based finite state machines implementation in low cost FPGAs. *IEEE Intern. Simp. on Industrial Electronics (ISIE'07) (Vigo, 2007),* 2007, 2342–2347. https://doi.org/10.1109/ISIE.2007.4374972
13. Garcia-Vargas L., Senhaji-Navarro R. Finite state machines with input multiplexing: A performance study. *IEEE Transactions on CAD of Integrated Circuits and Systems,* 2015, Vol. 34 (5), 867–871. https://doi.org/10.1109/TCAD.2015.2406859
14. Sklyarov V., Skliarova I., Barkalov A., Titarenko L. *Synthesis and optimization of FPGA-based systems.* Berlin: Springer, 2014, 432 p. https://doi.org/10.1007/978-3-319-04708-9
15. Virtex-7 FPGAs. URL: https://docs.amd.com/v/u/en-US/7-series-product-selectionguide [Accessed Jan. 2025]
16. Czerwinski R., Kania D. *Finite state machines logic synthesis for complex pro gra mmable logic devices.* Berlin: Springer, 2013, 172 p. https://doi.org/10.1007/978-3-642-36166-1

17. Tiwari A., Tomko K. Saving power by mapping finite state machines into em bedded memory blocks in FPGAs. *Proc. Design, Automation and Test in Europe Conference and Exhibition (Paris, France, 6-20 Feb. 2004)*, 2004, Vol. 2, 916–921. https://doi.org/10.1109/DATE.2004.1269007
18. Yang S. *Logic synthesis and optimization benchmarks user guide. Version 3.0. Techn. Rep.* Microelectronics Center of North Carolina, 1991, 43 p.
19. Vivado. URL: https://www.amd.com/en/products/software/adaptive-socs-and-fpgas/vivado.html [Accessed Jan. 2025]
20. Barkalov A. and Titarenko L. *Logic synthesis for FSM-based control units.* Lecture notes in electrical engineering, Springer-Verlag, Berlin, 2009, Vol. 53. https://doi.org/10.1007/978-3-642-04309-3_3
21. Barkalov A., Titarenko L., Kolopienczyk M., Mielcarek K., Bazydlo G. *Logic synthesis for FPGA-based Finite State Machines.* Studies in Systems, Decision and Control, Springer International Publishing, Cham Heidelberg, 2015, Vol. 38, 280 p. https://doi.org/10.1007/978-3-319-24202-6

Received 06.02.2025

O.O. BARKALOV, DSc (Engineering), Professor,
University of Zielona Góra,
9, Licealna Str., Zielona Góra, 65-417, Poland
https://orcid.org/orcid.org/0000-0002-4941-3979
A.Barkalov@iie.uz.zgora.pl

L.O. TITARENKO, DSc (Engineering), Professor,
University of Zielona Góra,
9, Licealna Str., Zielona Góra, 65-417, Poland
Kharkiv National University of Radio Electronics,
14, Science ave., Kharkiv, 61166, Ukraine
https://orcid.org/orcid.org/0000-0001-9558-3322
L.Titarenko@iie.uz.zgora.pl

O.M. GOLOVIN, PhD (Engineering),
Senior Research Associate,
Glushkov Institute of Cybernetics of the NAS of Ukraine,
40, Hlushkova Akademika ave., Kyiv, 03187, Ukraine
https://orcid.org/orcid.org/0000-0002-0279-812X
o.m.golovin.1@gmail.com

O.V. MATVIIENKO, Research Associate,
Glushkov Institute of Cybernetics of the NAS of Ukraine,
40, Hlushkova Akademika ave., Kyiv, 03187, Ukraine
https://orcid.org/orcid.org/0000-0003-1838-1422
avmatv@ukr.net

S.O. SABUROVA, PhD (Engineering),
Associate Professor,
Kharkiv National University of Radio Electronics,
14, Science ave., Kharkiv, 61166, Ukraine
https://orcid.org/orcid.org/0000-0001-6286-1648
sabsvet@gmail.com

OPTIMIZATION OF THE TWO-LEVEL MEALY MACHINE CIRCUIT IN THE FPGA BASIS

Introduction. One of the most important parts of any digital system is a control unit (CU), which coordinates the interaction of other blocks of the system. As a rule, the CU circuit is determined by the control algorithm, and the design of each CU starts from scratch due to the uniqueness of its operating algorithm.

The purpose of the paper is the quality of a digital system depends on the optimality of the CU characteristics. Therefore, the development of effective methods for optimizing CU circuits is so important. When synthesizing a CU circuit, a number of optimization problems arise: reducing the chip area occupied by the CU, increasing the performance reducing the power consumption. It is known that solving the first of these problems allows improving other characteristics of the circuit. This paper considers the problem of reducing the chip area when implementing the CU circuit using FPGA (field-programmable logic array) chips.

Methods. The FPGA chips and the Mealy finite state machine (FSM) model are selected as the objects of study in the article. When implementing the FSM circuit with FPGAs, LUT (look-up table) elements and embedded memory blocks (EMB) are used. Since the dominant manufacturer of FPGA chips is AMD Xilinx, the method proposed in the article is oriented towards FPGA of this company.

The article proposes a method for reducing hardware costs when implementing the Mealy FSM circuit in the FPGA basis. The method is based on the joint use of the EMB embedded memory blocks and LUT elements. The limiting case is considered when the developer can use only one EMB block. To optimize the circuit, the methods of replacing the FSM inputs and double state coding are used. An example of applying the proposed method is given.

Results. The proposed method allows reducing the number of LUT elements used by up to 18%. The conditions for the feasibility of using the method are shown.

Conclusions. It makes sense to modify the proposed method for the case of Moore FSMs.

Keywords: *Mealy FSM, synthesis, FPGA, EMB, LUT, state coding, input replacement.*