КІБЕРНЕТИКА

А.А. БАРКАЛОВ, Л.А. ТИТАРЕНКО, А.В. БАЕВ, А.В. МАТВИЕНКО

УДК 004.274

СМЕШАННОЕ КОДИРОВАНИЕ НАБОРОВ МИКРООПЕРАЦИЙ В МИКРОПРОГРАММНОМ АВТОМАТЕ

Аннотация. Предложен метод уменьшения количества *LUT* в схеме микропрограммного автомата Мили, реализуемого в базисе FPGA. Метод основан на удалении из наборов микроопераций некоторых элементов для реализации на блоке памяти *EMB*. Такой подход позволяет уменьшить число уровней логики и межсоединений в схеме, реализуемой на элементах *LUT*. Предложен алгоритм разбиения множества микроопераций. Приведен пример синтеза и показаны результаты проведенных исследований.

Ключевые слова: автомат Мили, синтез, FPGA, *LUT*, *EMB*, кодирование наборов микроопераций.

ВВЕДЕНИЕ

При проектировании цифровых систем на СБИС актуальной задачей является уменьшение площади кристалла, занимаемой схемой устройства управления (УУ) [1, 2]. Решение этой задачи позволяет улучшить такие характеристики УУ, как быстродействие и потребление энергии [3]. Методы ее решения зависят от модели УУ и используемого элементного базиса [4, 5].

При проектировании УУ [6] часто применяют модель микропрограммного автомата (МПА) Мили, в котором выходные переменные (микрооперации) зависят от входных (логических условий) и внутренних переменных, кодирующих состояния [7]. Это приводит к многоуровневым схемам с большим количеством межсоединений, что отрицательно влияет на быстродействие и энергопотребление, особенно при реализации схем МПА в базисе FPGA (field programmable gate arrays) [8, 9].

Микросхемы FPGA широко используются при проектировании современных цифровых систем [10]. Поэтому актуальна разработка методов синтеза, ориентированных на этот базис, особенно методов синтеза МПА с нерегулярной структурой схем [7, 11], и отличающихся от методов, ориентированных на регулярные схемы сумматоров, счетчиков, умножителей и т.д. [6].

В настоящей работе предложен метод уменьшения аппаратурных затрат в схеме МПА Мили, реализуемой в базисе FPGA. Для задания автомата используется язык граф-схем алгоритмов (ГСА) [7].

1. СИНТЕЗ СХЕМЫ МПА МИЛИ ПО ГСА

Рассмотрим пример автомата, описанного ГСА Γ_1, которая на рис. 1 отмечена состояниями автомата Мили по правилам [7]. Из ГСА Γ_1 можно получить множества внутренних состояний $A = \{a_1, \ldots, a_M\}$, $M = 7$, логических условий $X = \{x_1, \ldots, x_L\}$, $L = 5$, и микроопераций $Y = \{y_1, \ldots, y_N\}$, $N = 9$.

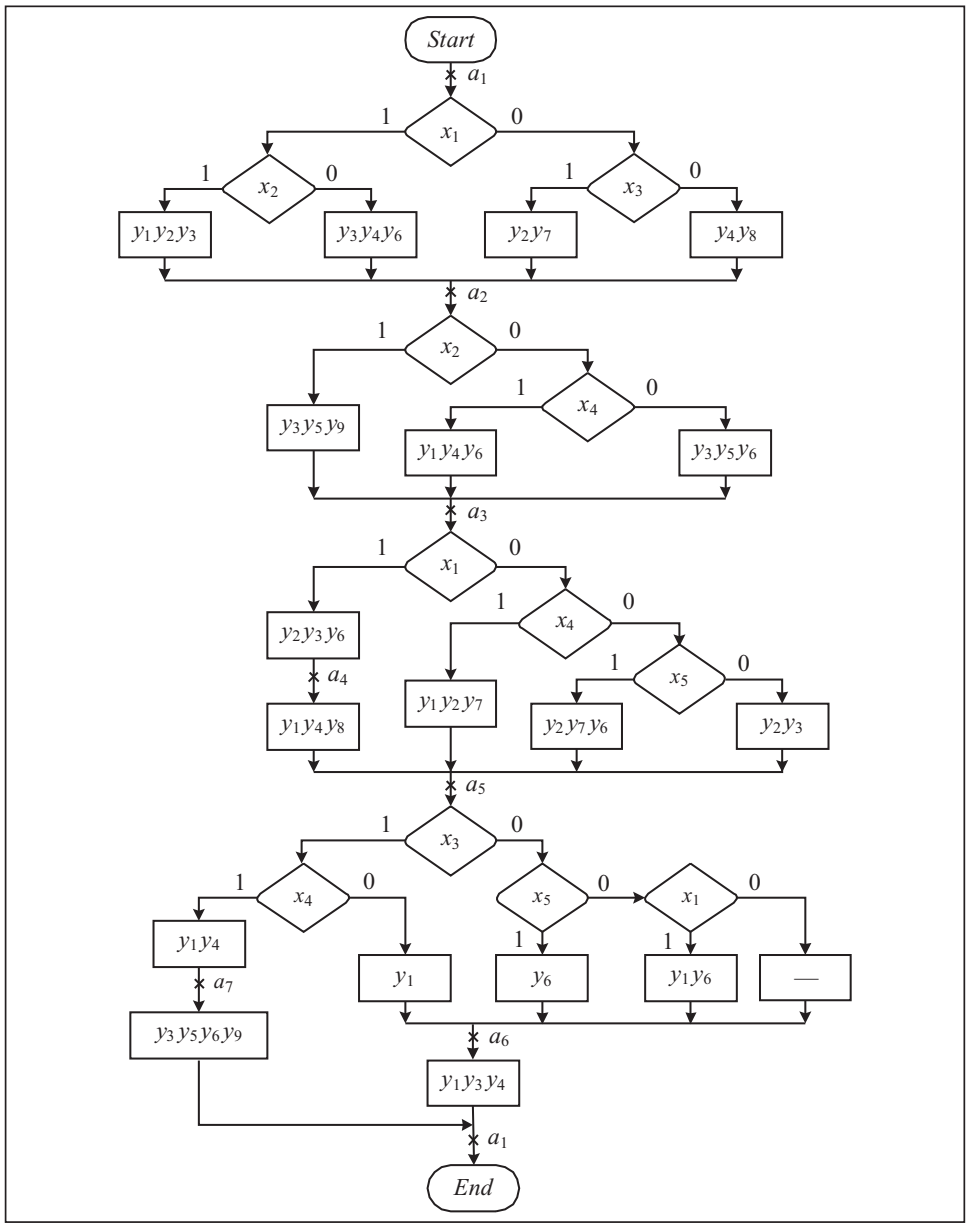

Рис. 1. Исходная ГСА Γ_1

Для синтеза схемы МПА необходимо получить некоторые системы булевых функций [6, 7]. Для этого следует закодировать состояния $a_m \in A$ двоичными кодами $K(a_m)$ разрядности R. При этом используются внутренние переменные, образующие множество $T = \{T_1, \dots, T_R\}$. Коды $K(a_m)$ хранятся в специальном регистре RG, содержимое которого можно изменить с помощью функций возбуждения памяти $\Phi = \{D_1, \dots, D_R\}$. Как правило, триггеры RG имеют информационные входы типа D [10]. Во многих практических случаях параметр $R = \lceil \log_2 M \rceil$. Здесь выражение $\lceil a \rceil$ обозначает ближайшее целое, равное a, если a целое или большее, чем a, если a дробное. Для ГСА Γ_1 имеем $R = 3$ и множества $T = \{T_1, T_2, T_3\}$ и $\Phi = \{D_1, D_2, D_3\}$.

Схема МПА Мили задается двумя системами функций:

$$\Phi = \Phi(T, X), \tag{1}$$

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

Таблица 1. Прямая структурная таблица МПА Мили

a_m	$K(a_m)$	a_s	$K(a_s)$	X_h	Y_h	Φ_h	h
a_1	000	a_2	001	$x_1 x_2$	$y_1 y_2 y_3$	D_3	1
		a_2	001	$x_1 \overline{x_2}$	$y_3 y_4 y_6$	D_3	2
		a_2	001	$\overline{x_1} x_3$	$y_2 y_7$	D_3	3
		a_2	001	$\overline{x_1}\,\overline{x_3}$	$y_4 y_8$	D_3	4
a_2	001	a_3	010	x_2	$y_3 y_5 y_9$	D_2	5
		a_3	010	$\overline{x_3} x_4$	$y_1 y_4 y_6$	D_2	6
		a_3	010	$\overline{x_3}\,\overline{x_4}$	$y_3 y_5 y_6$	D_2	7
a_3	010	a_4	011	x_1	$y_2 y_3 y_6$	$D_2 D_3$	8
		a_5	100	$\overline{x_1} x_4$	$y_1 y_2 y_7$	D_1	9
		a_5	100	$\overline{x_1}\,\overline{x_4} x_5$	$y_2 y_6 y_7$	D_1	10
		a_5	100	$\overline{x_1}\,\overline{x_4}\,\overline{x_5}$	$y_2 y_3$	D_1	11
a_4	011	a_5	100	1	$y_1 y_4 y_8$	D_1	12
a_5	100	a_7	110	$x_3 x_4$	$y_1 y_4$	$D_1 D_2$	13
		a_6	101	$x_3 \overline{x_4}$	y_1	$D_1 D_3$	14
		a_6	101	$\overline{x_3} x_5$	y_6	$D_1 D_3$	15
		a_6	101	$\overline{x_3}\,\overline{x_5} x_1$	$y_1 y_6$	$D_1 D_3$	16
		a_6	101	$\overline{x_3}\,\overline{x_5}\,\overline{x_1}$	—	$D_1 D_3$	17
a_6	101	a_1	000	1	$y_1 y_3 y_4$	—	18
a_7	110	a_1	000	1	$y_3 y_5 y_6 y_9$	—	19

Для получения систем (1), (2) необходимо построить прямую структурную таблицу (ПСТ) автомата Мили [7] (табл. 1), где a_m — исходное состояние; $K(a_m)$ — код состояния $a_m \in A$; a_s — состояние перехода; $K(a_s)$ — код состояния $a_s \in A$; X_h — конъюнкция логических условий, определяющая переход $\langle a_m, a_s \rangle$; Y_h — набор микроопераций (НМО), формируемый на переходе $\langle a_m, a_s \rangle$; Φ_h — набор функций возбуждения памяти, принимающих значение 1 для записи в RG кода $K(a_s)$; h — номер перехода, $h \in \{1, \ldots, H\}$, $H = 19$. Для безусловных переходов $X_h = 1$.

Закодируем состояния для описываемого примера тривиальным образом: $K(a_1) = 000, \ldots, K(a_7) = 110$. Эти коды приведены в ПСТ МПА Мили (см. табл. 1).

Функции (1), (2) зависят от термов F_h, соответствующих строкам ПСТ. Терм F_h определяется формулой $F_h = A_m X_h$, $h \in \{1, \ldots, H\}$, где A_m — конъюнкция внутренних переменных кода $K(a_m)$ из строки ПСТ.

Из табл. 1 можно, например, получить следующие минимизированные функции:

$$D_1 = \overline{T_1} T_2 \overline{T_3}\,\overline{x_1} \vee \overline{T_1} T_2 T_3 \vee T_1 \overline{T_2}\,\overline{T_3}; \;\; y_9 = \overline{T_1}\,\overline{T_2} T_3 x_2 \vee T_1 T_2 \overline{T_3}.$$

Системы (1), (2) определяют структурную схему МПА Мили U_1 (рис. 2). Здесь блок функций возбуждения памяти реализует функции (1), а блок микроопераций — систему (2). По импульсу *Start* в регистр RG записывается код начального состояния $a_1 \in A$. Импульс *Clock* разрешает изменение содержимого RG.

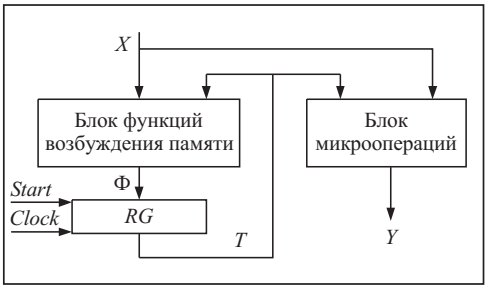

Рис. 2. Структурная схема МПА Мили U_1

2. РЕАЛИЗАЦИЯ СХЕМЫ МПА В БАЗИСЕ FPGA

Для реализации МПА можно использовать следующие компоненты FPGA: элементы табличного типа *LUT* (look-up table); встроенные блоки памяти *EMB* (embedded memory blocks); программируемые триггеры; матрицу программируемых межсоединений [12, 13]. Элементы *LUT* — это ОЗУ, имеющие S_L входов и один выход. Один *LUT* может реализовать произвольную булеву функцию от S_L аргументов. Выходы *LUT* либо связаны с входами триггеров с помощью специальных мультиплексоров [14], либо не связаны с ними.

Блок *EMB* — это реконфигурируемое ОЗУ, имеющее S_A входов и t_F выходов. Емкость V_0 блока *EMB* является константой, $V_0 = 2^{S_A} \times t_F$. Конфигурации блока *EMB* определяются парами $\langle S_A, t_F \rangle$. Обычно $V_0 = 32$К, и для такой емкости возможны следующие конфигурации: 32К×1, 16К×2, 8К×4, 4К×8, 2К×16, 1К×32 и 512×64 [15].

Для реализации схем регистров в базисе FPGA используются программируемые триггеры. Матрица межсоединений предназначена для организации связей между элементами схемы МПА.

Обозначим *LUTer* схему, состоящую из элементов *LUT*, а *EMBer* — схему, состоящую из блоков *EMB*. Существуют две тривиальные схемы автомата U_1 (рис. 3).

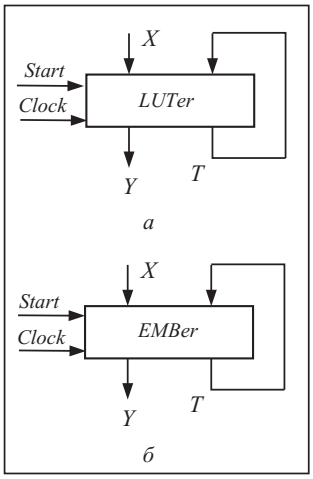

Рис. 3. Тривиальные реализации автомата U_1

На рис. 3, *а* системы (1), (2) реализуются на элементах *LUT*. Регистр *RG* распределен между этими элементами. Однако *LUT* имеют ограниченное число входов ($S_L \le 6$) [14]. Анализ библиотеки [16] показывает, что функции (1), (2) могут зависеть от 20 и более аргументов. Такой дисбаланс приводит к необходимости применять методы функциональной декомпозиции [17, 18], в результате чего увеличивается число элементов, количество уровней логики и межсоединений (по сравнению с одноуровневыми схемами).

На рис. 3, *б* системы (1), (2) реализуются на блоках *EMB*. Эти блоки допускают возможность синхронизации [14], поэтому регистра *RG* в явном виде здесь не существует. Анализ библиотеки [16] показал, что для 82 % всех примеров достаточно одного блока *EMB*. В остальных случаях это число варьируется от двух до восьми. Как отмечается в [19, 20], блоки *EMB* широко используются для реализации схем различных блоков цифровых систем и для реализации УУ может быть выделен только один блок *EMB* [13]. Поэтому для достаточно сложных МПА целесообразно совместное использование *LUT* и *EMB*. Как правило, *LUT* применяются для реализации схемы замены логических условий [7, 17]. Это приводит к МПА U_2 (рис. 4).

В автомате U_2 (см. рис. 4) блок *LUTer* реализует систему функций $P = P(T, X)$. Переменные $p_g \in P$ заменяют логические условия $x_i \in X$. При этом $|P| << L$, что позволяет уменьшить количество блоков *EMB* по сравнению с эквивалентным автоматом U_1. Будем называть автоматы эквивалентными, если они синтезируются по одной и той же ГСА Г.

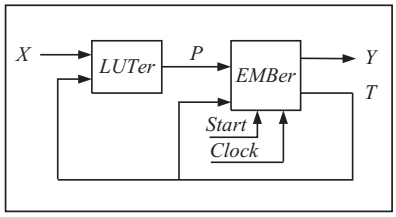

Рис. 4. Структурная схема автомата U_2

Отметим, что для реализации схемы *LUTer* может понадобиться применение функциональной декомпозиции. Кроме того, при

ограниченном количестве *EMB*, доступных для реализации схемы МПА, часть функций $y_n \in Y$ и $D_r \in \Phi$ необходимо реализовать на элементах *LUT*.

Метод замены логических условий относится к методам структурной декомпозиции [18]. В настоящей работе для реализации схемы МПА предлагается использовать метод кодирования НМО и модифицировать известный подход [17] для уменьшения числа элементов *LUT* в схеме автомата.

3. ОСНОВНАЯ ИДЕЯ ПРЕДЛАГАЕМОГО МЕТОДА

Поставим в соответствие каждому из Q наборов микроопераций $Y_q \subseteq Y$ двоичный код $K(Y_q)$ разрядности $R_Q = \lceil \log_2 Q \rceil$. Для кодирования НМО используем переменные $z_r \in Z$, где $|Z| = R_Q$. Тогда микрооперации $y_n \in Y$ представляются системой функций, зависящих от переменных $z_r \in Z$:

$$Y = Y(Z). \tag{3}$$

Как правило, выполняется неравенство $Q << L + R$, из чего следует, что для реализации системы (3) требуется гораздо меньше элементов *LUT*, чем для реализации системы (2). Однако при этом некоторые ресурсы FPGA необходимо использовать для реализации системы $Z = Z(T, X)$.

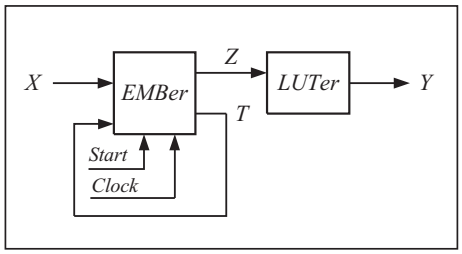

Рис. 5. Структурная схема МПА U_3

При кодировании НМО устройство управления можно представить как МПА U_3 (рис. 5), в котором блок *EMBer* реализует функции $z_r \in Z$ и $D_r \in \Phi$ (выходы блока *EMBer* соответствуют внутренним переменным МПА). Блок *LUTer* реализует систему функций (3).

Пусть разработчик схемы УУ использует только один блок *EMB*. В этом случае для реализации модели U_1 должно выполняться условие

$$2^{L+R} \times (R + N) \le V_0, \tag{4}$$

для модели U_2 — условие

$$2^{G+R} \times (R + N) \le V_0, \tag{5}$$

а для модели U_3 — условие

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

Рассмотрим случай, когда условия (4), (5) нарушаются, а условие (6) выполняется. Если, кроме того, выполняется условие $R_Q > S_L$, то можно использовать модель U_3. Однако блок *LUTer* является многоуровневой схемой. Отметим, что при $R_Q > S_L$ блок *LUTer* состоит из N элементов *LUT*.

Рассмотрим случай, когда блок *EMBer* реализуется на одном блоке *EMB*. Пусть при выполнении условия (6) блок *EMB* имеет конфигурацию $t_F = t_1$.

Пусть выполняется следующее условие:

$$\Delta_t = t_1 - (R + R_Q) \ge 0. \tag{7}$$

Обозначим V множество НМО, которое включает Y_i и Y_j такие, что $Y_i = Y_j \cup \{y_n\}$. Тогда удаление y_n из Y_i приводит к равенству $Y_i = Y_j$, а число НМО множества V уменьшается на единицу. Если $I(y_n)$ — число пар НМО $\langle Y_j, Y_i \rangle$ таких, что удаление МО y_n из наборов $Y_i \subseteq Y$ ведет к равенству $Y_i = Y_j$,

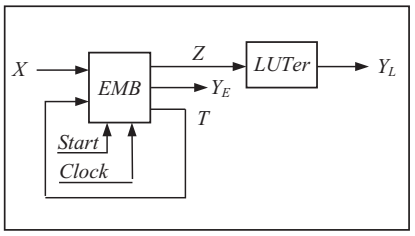

Рис. 6. Структурная схема МПА U_4

то это позволяет уменьшить количество НМО на $I(y_n)$. Вследствие удаления y_n множество V преобразуется во множество V_1, имеющее Q_1 элементов, $Q_1 = Q - I(y_n)$. Для кодирования НМО $Y_q \in V_1$ необходимо R_1 бит, $R_1 = \lceil \log_2 Q_1 \rceil$.

Удаление y_n может привести к выполнению условия

$$R_1 < R_Q. \tag{8}$$

При этом в блоке *EMB* появляется свободный выход, который используется для генерации МО y_n.

Далее следует найти МО y_m, удаление которой из НМО $Y_q \in V_1$ приводит к множеству V_2, имеющему Q_2 наборов. Для их кодирования необходимо R_2 бит, $R_2 = \lceil \log_2 Q_2 \rceil$.

Выполнение условия $R_1 < R_2$ означает, что вследствие удаления y_m происходит дальнейшее уменьшение разрядности кодов НМО. Этот процесс можно продолжать до тех пор, пока удаление любой из оставшихся МО не будет приводить к уменьшению разрядности кодов НМО.

Пусть удаленные МО образуют множество Y_E. Очевидно, что они реализуются блоком *EMB*. Остальные МО образуют множество $Y_L = Y \setminus \{Y_E\}$ и генерируются блоком *LUTer*.

Предлагаемый подход позволяет получить два вида кодов. Блок *EMB* генерирует унитарные коды МО $y_n \in Y_E$, а блок *LUTer* — максимальные коды [6, 10] НМО $Y_q \in V_L$, где V_L — множество наборов МО, состоящих из $y_n \in Y_L$. Назовем такое кодирование смешанным.

Рассмотренная структура порождает МПА U_4 (рис. 6).

В автомате U_4 используется только один блок *EMB*, функции $y_n \in Y_E$ представляются в виде (2), а функции $y_n \in Y_L$ — в виде (3).

Множество Y_L образует V_L наборов МО. Для их кодирования необходимо R_L переменных, $R_L = \lceil \log_2 Q_L \rceil$, а параметр Q_L определяется как $Q_L = |V_L|$.

Приведенный подход дает наилучшие результаты при выполнении условия

$$R_L \le S_L. \tag{9}$$

В этом случае для реализации каждой МО $y_n \in Y_L$ достаточно одного элемента *LUT*.

Предлагаемый метод синтеза автомата U_4 включает следующие этапы:

1) формирование множества состояний A по ГСА Г;
2) кодирование состояний $a_m \in A$;
3) формирование множества НМО V;
4) формирование множеств Y_E и Y_L;
5) кодирование наборов микроопераций;
6) формирование таблиц блоков *EMB* и *LUTer*;
7) реализация схемы автомата в заданном базисе.

Далее предлагается метод разбиения множества Y (п. 4 метода синтеза автомата U_4), приводится пример синтеза и результаты исследований.

4. РАЗБИЕНИЕ МНОЖЕСТВА МИКРООПЕРАЦИЙ

На рис. 7 представлена блок-схема предлагаемого алгоритма разбиения множества Y. На первом этапе все МО находятся во множестве Y_L, а все НМО — в V_L (вершина *1*). Алгоритм имеет не более N шагов (параметр k), а каждый шаг — не более N циклов (параметр i).

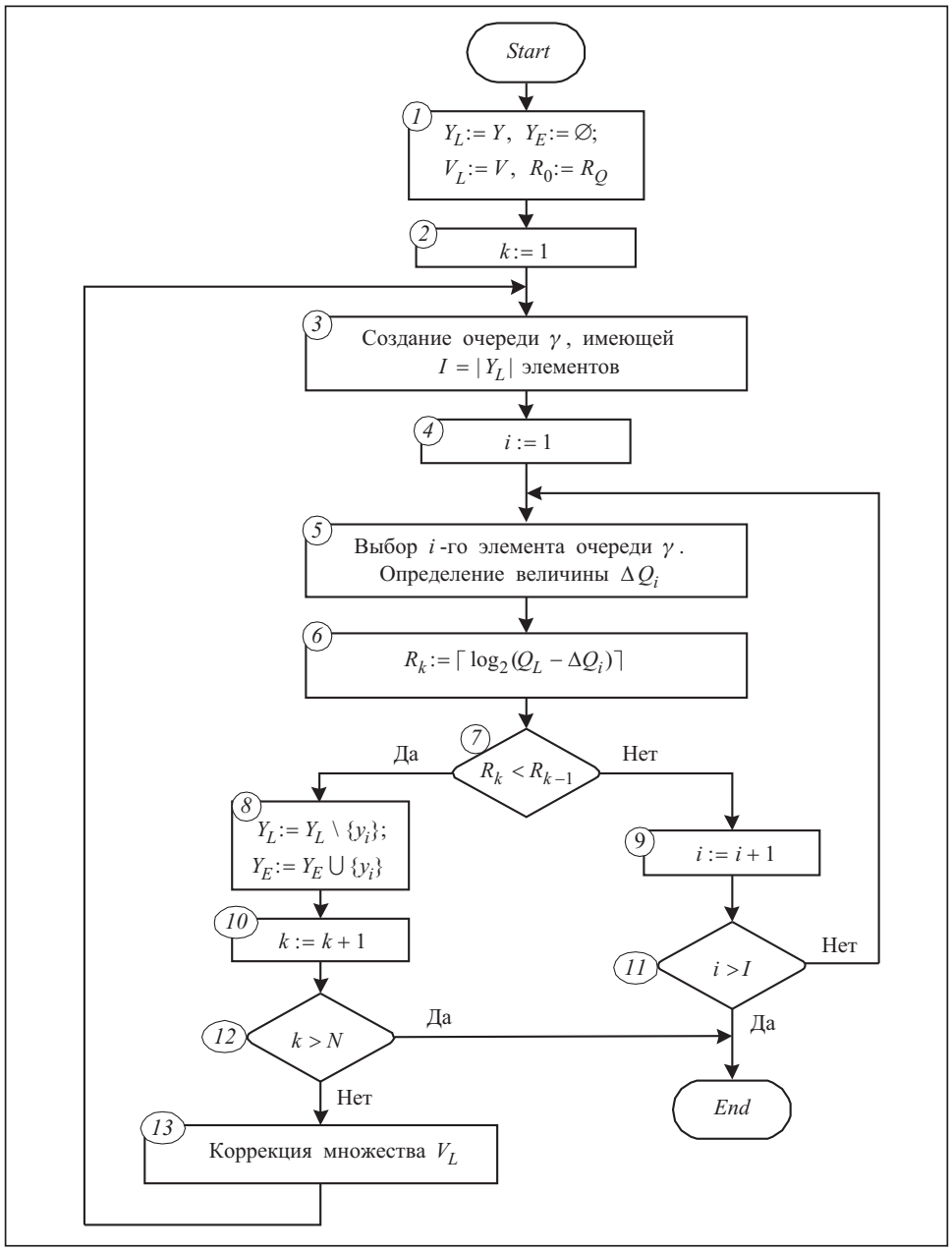

Рис. 7. Блок-схема алгоритма разбиения множества МО

Основная идея метода — поиск микроопераций $y_n \in Y_L$, включение которых во множество Y_E максимально уменьшает количество НМО, остающихся во множестве Y_L. Для этого на каждом шаге алгоритма формируется очередь γ микроопераций $y_n \in Y_L$. Элементы очереди упорядочиваются в порядке уменьшения параметра $Q(y_n)$, который равен количеству НМО $Y_q \in V_L$, содержащих МО $y_n \in Y_L$. Количество элементов в очереди (вершина *3*) определяет максимальное число циклов на шаге k, $k \in \{1, \ldots, N\}$.

В каждом цикле выполняется выбор того элемента очереди (вершина *5*), для которого определяется величина ΔQ_i, равная количеству НМО, исключаемых из V_L вследствие включения МО $y_n \in Y_L$ во множество Y_E. Далее определяется чис-

ло переменных R_k, достаточное для кодирования НМО $Y_q \in V_L$ (вершина *6*):

$$R_k = \lceil \log_2 (Q_L - \Delta Q_i) \rceil. \qquad (10)$$

Если исключение *i*-го элемента очереди γ не приводит к уменьшению разрядности кода $K(Y_q)$ (выход «Нет» из блока *7*), то осуществляется переход к следующему элементу очереди (вершина *9*). Если очередь исчерпана (выход «Да» из блока *11*), то алгоритм завершается. В противном случае (выход «Нет» из блока *11*) выбирается следующий элемент очереди γ.

Если разрядность кода $K(Y_q)$ уменьшается (выход «Да» из блока *7*), то *i*-й элемент очереди включается в Y_E и исключается из Y_L (вершина *8*). Параметр k увеличивается на 1 (вершина *10*). Если все МО y_n проверены (выход «Да» из блока *12*), то алгоритм завершается. В противном случае осуществляется коррекция множества V_L (вершина *13*) и создается новая очередь (переход к вершине *3*).

Проиллюстрируем этот алгоритм на примере ГСА Γ_1, анализ которой показывает, что ее операторные вершины содержат 18 НМО (табл. 2). Имеем $V = \{Y_1, \ldots, Y_{18}\}$, $Q = 18$, $R_Q = 5$, что определяет множество $Z = \{z_1, \ldots, z_5\}$.

Разбиение множества $\bar{Y}$ приведено в табл. 3. Здесь в столбце y_n показаны микрооперации $y_n \in Y$, в столбцах $Q(y_n)$ и ΔQ_i — величины, используемые в блок-схеме алгоритма (см. рис. 7), знаки +, $\oplus$ и – обозначают соответственно, что МО y_n выбрана для анализа, включена во множество Y_E, в дальнейшем не рассматривается. В строке Y^1 приведены МО $y_n \in Y_E$ в порядке их включения во множество Y_E. Информация из столбца $Q(y_n)$ позволяет создать очередь $\gamma = \{y_6, y_1, y_2, y_3, y_4, y_5, y_7, y_8, y_9\}$. Включение y_6 в Y_E уменьшает количество НМО на $\Delta Q_i = 6$. Из (10) имеем $R_1 = 4$. Условие (8) выполняется и МО y_6 включается в Y_E.

Таблица 2. Наборы микроопераций для ГСА Γ_1

q	Y_q	q	Y_q	q	Y_q
1	—	7	$y_1 y_4 y_6$	13	$y_1 y_4 y_8$
2	$y_1 y_2 y_3$	8	$y_3 y_5 y_6$	14	$y_1 y_4$
3	$y_3 y_4 y_6$	9	$y_2 y_3 y_6$	15	y_1
4	$y_2 y_7$	10	$y_1 y_2 y_7$	16	y_6
5	$y_4 y_8$	11	$y_2 y_7 y_6$	17	$y_1 y_6$
6	$y_3 y_5 y_9$	12	$y_2 y_3$	18	$y_3 y_5 y_6 y_9$

Таблица 3. Разбиение множества микроопераций

y_n	$Q(y_n)$	ΔQ_i	R_1	$Q(y_n)$	ΔQ_i	R_2	$Q(y_n)$	ΔQ_i	R_3
y_1	7			5 +	4	3 $\oplus$	–	–	–
y_2	7			4			2 +	0	3 –
y_3	6			5			4 +	0	3 –
y_4	5			4			3 +	1	3 –
y_5	3			2			2 +	0	3 –
y_6	8 +	6	4 $\oplus$	–	–	–	–	–	–
y_7	3			2			1 +	0	3 –
y_8	2			2			1 +	1	3 –
y_9	2			1			1 +	1	3 –
Y^1	y_6			y_1			–		

Таблица 4. Наборы МО после исключения y_6

q	Y_q	q	Y_q	q	Y_q
1	—	5	$y_4 y_8$	9	$y_2 y_3$
2	$y_1 y_2 y_3$	6	$y_3 y_5 y_9$	10	$y_1 y_2 y_7$
3	$y_3 y_4$	7	$y_1 y_4$	11	$y_1 y_4 y_8$
4	$y_2 y_7$	8	$y_3 y_5$	12	y_1

Таблица 5. Наборы МО после исключения y_1

q	Y_q	q	Y_q	q	Y_q	q	Y_q
1	—	3	$y_3 y_4$	5	$y_4 y_8$	7	y_4
2	$y_2 y_3$	4	$y_2 y_7$	6	$y_3 y_5 y_9$	8	$y_3 y_5$

Рассмотрим множество $V_L = \{Y_1, \ldots, Y_{12}\}$. Полученные НМО $Y_q \in V_L$ приведены в табл. 4. Далее формируется очередь $\gamma = \{y_1, y_2, y_3, y_4, y_5, y_7, y_8, y_9\}$. Включение МО y_1 в Y_E позволяет исключить четыре НМО из V_L, что дает $R_2 = 3$. Следовательно, условие (7) выполняется и МО y_1 включается в Y_E.

Рассмотри множество $V_L = \{Y_1, \ldots, Y_8\}$. Полученные НМО $Y_q \in V_L$ показаны в табл. 5.

Последние три столбца табл. 3: $Q(y_n)$, ΔQ_i, R_3, показывают, что разрядность кодов НМО $Y_q \in V_L$ не изменяется, поэтому алгоритм завершает свою работу. В результате применения предложенного алгоритма имеем $Y_E = \{y_1, y_6\}$, $Y_L = \{y_2, y_3, y_4, y_5, y_7, y_8, y_9\}$, $Q_i = 8$, $R_L = 3$, $Z = \{z_1, z_2, z_3\}$. Следовательно, разрядность кодов НМО уменьшена на 2.

5. ПРИМЕР СИНТЕЗА АВТОМАТА СО СМЕШАННЫМ КОДИРОВАНИЕМ МО

Рассмотрим пример синтеза МПА U_4 по ГСА Γ_1.

Пусть в используемой микросхеме FPGA имеется только три входа S_L элементов *LUT*, а среди конфигураций блоков *EMB* — конфигурация $S_A = 8$ и $t_F = 8$.

Как следует из ГСА Γ_1, автомат Мили характеризуется следующими параметрами: $L = 5$, $R = 3$, $N = 9$, $R_Q = 5$. Поскольку $S_A = 8$ и $t_F = 8$, то $V_0 = 2^8 \times 8 = 2048$ бит.

Для автомата U_1 (Γ_1) имеем $2^{L+R} \times (N + R) = 2^8 \times 12 = 3072$ бит. Для автомата U_2 (Γ_1) имеем $G = 4$ (это следует из [10] и в настоящей статье подробно не объясняется). Итак, $2^{G+R} \times (R + R_Q) = 2^7 \times 12 = 1536$ бит. Так как $R_Q = 5$, то $2^{L+R} \times (R + R_Q) = 2^8 \times 8 = 2048$ бит. Очевидно, в данном случае можно использовать модели U_2 и U_3. Это следует из условий (4)–(6).

Поскольку $t_L = t_F = 8$, $R = 3$, $R_Q = 5$, условие (7) выполняется, следовательно, модель U_4 (Γ_1) можно использовать.

Этапы 1–4 предложенного метода синтеза для МПА U_4 выполнены. Коды состояний $K(a_m)$ показаны в ПСТ (см. табл. 1). Множества Y_E и Y_L найдены в разд. 4.

Условие (9) выполняется, поэтому кодирование НМО $Y_q \in V_L$ можно провести произвольно. Применив алгоритм кодирования [19], получим коды $K(Y_q)$ (рис. 8).

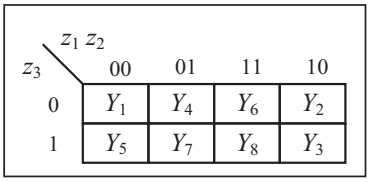

Рис. 8. Кодирование наборов микроопераций МПА U_4 (Γ_1)

Как следует из карты Карно (рис. 8), $y_3 = z_1$, значит, МО $y_3 \in Y_L$ генерируется блоком *EMB*. Таким образом, блок *LUTer* в данном случае состоит из $|Y_L| - 1 = 6$ элементов *LUT*. Отметим, что блок *LUTer* автомата U_4 (Γ_1) имеет один уровень логики.

Таблица блока *EMB* автомата Мили U_4 (Γ_1) строится на основе ПСТ автомата Мили (см. табл. 1) и имеет следующие столбцы (табл. 6): $K(a_m)$, X — адреса ячейки памяти, Y_E, Z, Ф — содержание соответствующих ячеек памяти, g — номер ячейки памяти, h — номер перехода. В общем случае таблица имеет G_{EMB} строк, $G_{EMB} = 2^{L+R}$, а переходы из состояния $a_m \in A$ описываются с помощью $H(a_m)$ строк, $H(a_m) = 2^L$. В рассматриваемом примере имеем $G_{EMB} = 256$ и $H(a_m) = 32$.

В столбце Y_E указываются МО $y_n \in Y_E$, формируемые на переходе $\langle a_m, a_s \rangle$. В столбце Z указываются коды $K(Y_q)$ НМО $Y_q \in V_L$, формируемых на переходе $\langle a_m, a_s \rangle$. В этом примере номера НМО выбираются из табл. 5, а их коды — из карты Карно (рис. 8).

Поясним заполнение табл. 6. Здесь строка $g = 1$ соответствует строке $h = 4$ из табл. 1, где в столбце Y_h записан набор $y_4 y_8$, соответствующий НМО Y_5 из табл. 5. Поэтому столбец Z для $g = 1$ содержит код 001 из рис. 8. Строка $g = 4$ соответствует строке $h = 1$, в которой записан НМО $y_1 y_2 y_3$ (см. табл. 1). Так как $y_1 \in Y_E$, столбец Y_E содержит код 10 (строка $g = 4$). После исключения y_1 имеем набор $y_2 y_3 = Y_2$ (см. рис. 8) с кодом $K(Y_2) = 100$, который находится в столбце Z (строка $g = 4$). При всех переходах из $a_1 \in A$ формируется $D_3 = 1$. Поэтому в столбце Ф находится код 001 (строки g от 1 до 8). Аналогично заполняются все 256 строк таблицы блока *EMB* автомата U_4 (Γ_1).

Таблица блока *LUTer* (табл. 7) содержит столбцы $K(Y_q)$, Y_L и q, а также имеет количество строк $Q = 8$. Коды НМО $Y_q \in V_L$ выбирают из рис. 8, а содержимое строки для столбца Y_L — из табл. 5. Отметим, что МО y_3 в табл. 7 не существует, так как $y_3 = z_1$.

На последнем этапе 7 предлагаемого метода применяются стандартные средства САПР для решения задач размещения, трассировки и формирования потоков «bit-stream» [10]. (В настоящей статье этот этап не рассматривается.)

Таблица 6. Фрагмент таблицы блока *EMB* автомата U_4 (Γ_1)

$K(a_m)$	X	Y_E	Z	Ф	g	h
$T_1 T_2 T_3$	$x_5 x_4 x_3 x_2 x_1$	$y_1\ y_6$	$z_1\ z_2\ z_3$	$D_1 D_2 D_3$		
0 0 0	0 0 0 0 0	0 0	0 0 1	0 0 1	1	4
0 0 0	0 0 0 0 1	0 1	1 0 1	0 0 1	2	2
0 0 0	0 0 0 1 0	0 0	0 0 1	0 0 1	3	4
0 0 0	0 0 0 1 1	1 0	1 0 0	0 0 1	4	1
0 0 0	0 0 1 0 0	0 0	0 1 0	0 0 1	5	3
0 0 0	0 0 1 0 1	0 1	1 0 1	0 0 1	6	2
0 0 0	0 0 1 1 0	0 0	0 1 0	0 0 1	7	3
0 0 0	0 0 1 1 1	1 0	1 0 0	0 0 1	8	1

Таблица 7. Блок *LUTer* автомата U_4 (Γ_1)

$K(Y_q)$	Y_L	q	$K(Y_q)$	Y_L	q
$z_1 z_2 z_3$	$y_2 y_4 y_5 y_7 y_8 y_9$		$z_1 z_2 z_3$	$y_2 y_4 y_5 y_7 y_8 y_9$	
0 0 0	0 0 0 0 0 0	1	0 0 0	1 0 0 0 0 0	2
0 0 1	0 1 0 0 1 0	5	0 0 1	0 1 0 0 0 0	3
0 1 0	1 0 0 1 0 0	4	0 1 0	0 0 1 0 0 1	6
0 1 1	0 1 0 0 0 0	7	0 1 1	0 0 1 0 0 0	8

6. ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ ПРЕДЛОЖЕННОГО МЕТОДА

Для анализа эффективности предложенного метода использованы стандартные тестовые примеры (СТП) из библиотеки LGSynth 93 [16]. Библиотека включает 48 СТП, заданных в формате KISS2.

Исследования проводились с помощью САПР K2F [20], DEMAIN [21] и SIS [6]. Из системы SIS использован алгоритм кодирования состояний JEDI, признанный одним из лучших в настоящее время [5, 10].

Для синтеза схемы МПА, ориентированного на микросхему XC5VLX30FF324 фирмы Xilinx, применялась САПР ISE14.1 этой фирмы [15]. Такой выбор обусловлен тем, что с помощью указанной микросхемы проводились исследования в [8, 20], что позволяет сравнить результаты, полученные в настоящей статье, с результатами использования других методов.

Исследовались модели U_1–U_4 из библиотеки [16]. Для реализации схем U_1 (Γ_j) применялся алгоритм JEDI ($U_{1\mathrm{J}}$) и система DEMAIN ($U_{1\mathrm{D}}$), основанная на сбалансированной функциональной декомпозиции [11]. Для автоматов U_2, U_3 результаты выбирались из [17]. Во всех случаях сравнивалось число элементов *LUT* в схеме, требуемое для реализации блока *LUTer*.

Отметим, что микросхема XC5VLX30FF324 относится к семейству Virtex-5. Она включает 32 блока памяти, из которых использовался только один. Элементы *LUT* имеют $S_L = 5$ входов.

Как уже отмечалось, 82 % СТП реализуются на одном блоке *EMB*. В табл. 8 приведены характеристики СТП, для реализации схем которых необходимо совместное использование *LUT* и *EMB*.

В табл. 9 показаны элементы *LUT*, необходимые для реализации блока *LUTer* в СТП из табл. 8. В строке Σ приведено суммарное количество элементов в блоке *LUTer* для каждой модели, а в строке % — процентное соотношение количества *LUT* для различных моделей МПА. При этом суммарное количество *LUT* для модели U_4 принято равным 100 %.

Таблица 8. Характеристики стандартных тестовых примеров

СТП	L	N	M	R	R_Q	G
*ex*1	9	19	20	5	7	5
planet	17	19	48	6	7	5
*planet*1	17	19	48	6	7	5
*S*1488	12	22	61	6	8	4
*S*1494	14	24	68	7	8	4
sand	11	27	32	5	6	3
styr	14	31	42	6	7	4
kirkman	12	16	16	4	8	5

Таблица 9. Суммарное количество элементов в блоке *LUTer*

СТП	U_{1J}	U_{1D}	U_2	U_3	U_4
*ex*1	20	24	22	36	16
planet	32	38	34	38	14
*planet*1	32	38	34	38	14
*S*1488	24	32	28	42	18
*S*1494	26	36	30	44	21
sand	30	38	32	52	22
styr	31	42	34	60	26
kirkman	26	36	28	34	12
Σ	221	282	242	344	143
%	154	197	169	240	100

Как видно из табл. 9, для данных СТП наилучшие характеристики имеет автомат U_4, затем модель U_1, базирующаяся на алгоритме JEDI. Наихудшие показатели у модели U_3, основанной на кодировании НМО.

Сравнение столбцов N (см. табл. 8) и U_4 (см. табл. 9) позволяет сделать следующие выводы. Количество элементов *LUT* в блоке *LUTer* меньше числа микроопераций в СТП. Это означает, что часть микроопераций реализована на блоке *EMB*. Схемы блока *LUTer* для U_4 одноуровневые, т.е. смешанное кодирование НМО позволило уменьшить число разрядов кода $K(Y_q)$ до $S_L = 5$.

Для всех остальных моделей блок *LUTer* содержит более одного уровня логики. Следовательно, предложенный подход позволяет уменьшить число элементов *LUT*, их уровней и межсоединений в блоке *LUTer* по сравнению с другими исследуемыми моделями. Как известно [8, 22], приблизительно 60 % энергии используют межсоединения. Поэтому следует ожидать, что схемы, основанные на модели U_4, будут потреблять меньше энергии, чем схемы, базирующиеся на моделях U_1–U_3. Поскольку блок *LUTer* в U_4 имеет только один уровень, предложенный метод позволяет получать схемы с более высоким быстродействием, чем для моделей U_1–U_3.

Естественно, что эти выводы справедливы только для данной микросхемы FPGA и автоматов из [16]. Однако результаты исследований показывают, что предложенный подход может улучшить основные характеристики устройств управления.

ЗАКЛЮЧЕНИЕ

Описанный в работе метод относится к методам структурной декомпозиции схем МПА [10] и является развитием идей, предложенных в [17]. Метод позволяет улучшить характеристики схемы МПА, основанной на кодировании наборов микроопераций. Уменьшение аппаратурных затрат обеспечивается реализацией некоторых микроопераций блоком *EMB*. Это дает возможность уменьшить количество наборов, используемых блоком *LUTer* для реализации остальных микроопераций.

Метод можно применять, если разработчик схемы устройства управления использует ограниченное количество блоков встроенной памяти. В работе рассмотрен случай, когда для этого имеется только один блок *EMB*.

Как показали исследования, предложенный метод позволяет получать одноуровневые схемы блоков *LUTer* для стандартных примеров [16], что приводит к уменьшению числа элементов *LUT* и их межсоединений по сравнению с методами, применение которых ведет к многоуровневым схемам *LUTer*. Таким образом, смешанное кодирование наборов микроопераций может рассматриваться

как альтернатива известным методам синтеза схем микропрограммных автоматов в базисе FPGA.

В работах [23–27] предлагаются различные методы синтеза схем совмещенных МПА. Как известно [7], последние имеют выходные сигналы типа Мура и Мили. Поэтому дальнейшим направлением исследований будет разработка методов синтеза совмещенных МПА со смешанным кодированием наборов микроопераций.

СПИСОК ЛИТЕРАТУРЫ

1. Соловьев В.В. Проектирование цифровых схем на основе программируемых логических интегральных схем. Москва: Горячая линия — ТЕЛЕКОМ, 2001. 636 с.
2. Грушницкий Р.И., Мурсаев А.Х., Угрюмов Е.П. Проектирование систем с использованием микросхем программируемой логики. СПб: БХВ-Петербург, 2002. 608 с.
3. 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 Exhibitio*n (Paris, France, 6–20 Feb. 2004). 2004. Vol. 2. P. 916–921.
4. Skliarova I., Sklyarov V., Sudnitson A. Design of FPGA-based circuits using hierarchical finite state machines. Tallinn: TUT Press, 2012. 240 p.
5. Czerwinski R., Kania D. Finite state machines logic synthesis for complex programmable logic devices. Berlin: Springer, 2013. 172 p.
6. DeMicheli G. Synthesis and optimization of digital circuits. New York: McGraw-Hill, 1994. 576 p.
7. Baranov S. Logic synthesis for control automata. Dordrecht: Kluwer Academic Publishers, 1994. 312 p.
8. Garcia-Vargas I., Senhadji-Navarro R., Jiménez-Moreno G., Civit-Balcells A., Guerra-Gutierrezz P. ROM-based finite state machines implementation in low cost FPGAs. *IEEE Intern. Simp. on Industrial Electronics* (ISIE'07) (Vigo, 2007). 2007. P. 2342–2347.
9. Rawski M., Tomaszewicz P., Borowski G., Łuba T. Logic synthesis method of digital circuits designed for implementation with embedded memory blocks on FPGAs. In: *Design of Digital Systems and Devises. Lecture Notes in Electrical Engineering.* Adamski M., Barkalov A., Wegrzyn M. (Eds.). Vol. 79. Berlin: Springer, 2011. P. 121–144.
10. Sklyarov V., Skliarova I., Barkalov A., Titarenko L. Synthesis and optimization of FPGA-based systems. Berlin: Springer, 2014. 432 p.
11. Rawski M., Selvaraj H., Łuba T. An application of functional decomposition in ROM-based FSM implementation in FPGA devices. *Journal of System Architecture*. 2005. Vol. 51, Iss. 6–7. P. 424–434.
12. Maxfield C. The design warrior's guide to FPGAs. Orlando: Academic Press, 2004. 542 p.
13. Grout I. Digital systems design with FPGAs and CPLDs. Amsterdam: Elsevier, 2008. 784 p.
14. White paper FPGA architecture. URL: www.altera.com.
15. UG473 (v1.14) July 3, 2019. URL: www.xilinx.com.
16. Yang S. Logic synthesis and optimization benchmarks user guide. Version 3.0. Techn. Rep. Microelectronics Center of North Carolina, 1991. 43 p.
17. Barkalov A., Titarenko L., Chmielewski S. Mixed encoding of collections of output variables for LUT-based FSMs. *Journal of Circuits, Systems and Computers.* 2019. Vol. 28, N 8. P. 1–21.
18. Nowicka M., Łuba T., Rawski M. FPGA-based decomposition of boolean functions: Algorithms and implementations. *Proc. of the 6th International Conference on Advanced Computer Systems* (Szczecin, 1999). P. 502–509.
19. Barkalov A., Titarenko L. Logic synthesis for FSM-based control units. Berlin: Springer, 2009. 233 p.
20. Kolopienczyk M., Titarenko L., Barkalov A. Design of EMB-based Moore FSMs. *Journal of Circuits, Systems and Computers*. 2017. Vol. 26, N 7. P. 1–23.
21. DEMAIN. URL: www.zpt.tele.pw.edu.pl/oprogramowanie/demain.html.
22. 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, Iss. 5. P. 867–871.

23. Баркалов А.А., Титаренко Л.А., Ефименко К.Н. Оптимизация схем композиционных микропрограммных устройств управления, реализуемых на ПЛИС. *Кибернетика и системный анализ*. 2011. № 1. С. 179–188.

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

25. Баркалов А.А., Титаренко Л.А., Визор Я.Е., Матвиенко А.В. Оптимальное кодирование состояний в совмещенном автомате. *Управляющие системы и машины.* 2016. № 6. С. 34–39.

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

27. Баркалов А.А., Титаренко Л.А., Визор Я.Е., Матвиенко А.В. Уменьшение аппаратурных затрат в совмещенных автоматах. *Управляющие системы и машины.* 2017. № 4. С. 43–50.

Надійшла до редакції 11.06.2019

О.О. Баркалов, Л.О. Тітаренко, А.В. Баєв, О.В. Матвієнко

ЗМІШАНЕ КОДУВАННЯ НАБОРІВ МІКРООПЕРАЦІЙ У МІКРОПРОГРАМНОМУ АВТОМАТІ

Анотація. Запропоновано метод зменшення кількості *LUT* у схемі мікропрограмного автомата Мілі, який реалізується в базисі FPGA. Метод ґрунтується на вилученні з наборів мікрооперацій деяких елементів для реалізації на блоці пам'яті *EMB*. Такий підхід дає змогу зменшити кількість рівнів логіки і міжз'єднань у схемі, яка реалізується на елементах *LUT*. Запропоновано алгоритм розбиття множини мікрооперацій. Наведено приклад синтезу і результати проведених досліджень.

Ключові слова: автомат Мілі, синтез, FPGA, *LUT*, *EMB*, кодування наборів мікрооперацій.

A.A. Barkalov, L.A. Titarenko, A.V. Baiev, A.V. Matviienko

MIXED ENCODING OF COLLECTIONS OF MICROOPERATIONS FOR MICROPROGRAMMED AUTOMATA

Abstract. A method is proposed for reducing the number of *LUT*s in the circuit of a microprogrammed Mealy FSM. The method is based on elimination of some elements from the sets of microoperations for their implementation by EMB. This approach reduces logic levels and interconnections for a circuit implemented with *LUT*s. An algorithm is proposed for searching a partition of the set of micro-operations. An example of synthesis is given as well as results of the investigations.

Keywords: Mealy FSM, synthesis, FPGA, *LUT*, *EMB*, encoding of collections of micro-operations.

Баркалов Александр Александрович,
доктор техн. наук, профессор Университета Зеленогурского (Польша); профессор Донецкого национального университета имени Василия Стуса, Винница, e-mail: A.Barkalov@iie.uz.zgora.pl.

Титаренко Лариса Александровна,
доктор техн. наук, профессор Университета Зеленогурского (Польша); профессор Харьковского национального университета радиоэлектроники, e-mail: L.Titarenko@iie.uz.zgora.pl.

Баев Артем Викторович,
кандидат физ.-мат. наук, ведущий научный сотрудник Донецкого национального университета имени Василия Стуса, Винница; руководитель направления по искусственному интеллекту, фирма Peoly, Винница, e-mail: a.baev@donnu.edu.ua.

Матвиенко Александр Владимирович,
научный сотрудник Института кибернетики им. В.М. Глушкова НАН Украины, Киев,
e-mail: avmatv@ukr.net.