КОМП'ЮТЕРНІ ЗАСОБИ, МЕРЕЖІ ТА СИСТЕМИ

A.A. Barkalov, L.A. Titarenko, Y.E. Vizor, A.V. Matvienko

REALIZATION OF COMBINED FINITE STATE MASHINE WITH FPGAs

The method is proposed for realization of combined microprogrammed automation targeting FPGA. The method allows obtaining a circuit with minimum number of LUTs and EMBs.
Key words: combined FSM, FPGA, LUT, EMB, synthesis, graph-scheme of algorithm.

Запропоновано метод синтезу суміщеного мікропрограмного автомата, орієнтований на НВІС типу FPGA. Метод дозволяє отримати схему з мінімальним числом елементів LUT і блоків EMB.
Ключові слова: суміщений автомат, FPGA, LUT, EMB, синтез, граф-схема алгоритму.

Предложен метод синтеза совмещенного микропрограммного автомата в базисе СБИС типа FPGA. Метод позволяет получить схему с минимальным числом элементов LUT и блоков EMB.
Ключевые слова: совмещенный автомат, FPGA, LUT, EMB, синтез, граф-схема алгоритма.

УДК 004.274

А.А. БАРКАЛОВ, Л.А. ТИТАРЕНКО,
Я.Е. ВИЗОР, А.В. МАТВИЕНКО

РЕАЛИЗАЦИЯ СХЕМЫ СОВМЕЩЕННОГО АВТОМАТА В БАЗИСЕ FPGA

Введение. Одной из центральных частей практически любой цифровой системы является устройство управления [1]. Для синтеза схемы устройства управления часто используется модель микропрограммного автомата (МПА) [2]. В МПА могут существовать выходные сигналы двух типов. Выходные сигналы типа Мили формируются при изменении состояния МПА, а выходные сигналы типа Мура существуют в течение такта работы МПА [2, 3]. Автоматы с двумя типами выходных сигналов называют совмещенными МПА (СМПА) [1, 2].

В настоящее время для реализации цифровых систем широко используются СБИС типа FPGA (*field-programmable logic arrays*) [4, 5]. Для реализации схем МПА могут использоваться два типа логических элементов, входящих в *FPGA*. Первый из них – элементы типа *LUT* (*look-up table*), имеющие ограниченное число входов (не более восьми). Второй тип элементов – встроенные блоки памяти типа *EMB* (*embedded memory blocks*). Этим блокам присуще свойство изменения числа ячеек памяти (V) и их выходов (t_F) при постоянной общей емкости V_0: $V_0 = V \times t_F$.

Число ячеек памяти однозначно определяет число их входов (S_A):

$$S_A = \lceil \log_2 V \rceil. \qquad (1)$$

В формуле (1) функция $\lceil a \rceil$ определяет целое число, большее a, если a – дробное и равное a, если a – целое.

При реализации МПА в базисе *FPGA* важно уменьшать площадь кристалла, занимаемого схемой. В результате сокращается время распространения сигналов и потребляемая мощность [6].

Один из подходов к решению данной задачи – это замена элементов *LUT* блоками *EMB* [7 – 12]. В настоящей работе мы предлагаем одно из возможных решений этой задачи при синтезе схемы СМПА. Отметим, что в литературе практически отсутствуют методы синтеза, ориентированные на модель СМПА.

Особенности совмещенного МПА и FPGA. Для оптимизации характеристик схемы МПА необходимо учесть характеристики, как модели автомата, так и элементного базиса. Рассмотрим эти особенности.

Математическая модель СМПА – восьмикомпонентный вектор $S=<A, X, Y^1, Y^2, \delta, \lambda_1, \lambda_2, a_1>$.

Вектор S включает следующие компоненты: $A=\{a_1,\ldots, a_M\}$ – множество внутренних состояний; $X=\{x_1,\ldots,x_L\}$ – множество входных переменных; Y^1 – множество выходных переменных автомата Мили; Y^2 – множество выходных переменных автомата Мура; δ – функция переходов; λ_1 – функция выходов автомата Мили; λ_2 – функция выходов автомата Мура; $a_1 \in A$ – начальное состояние автомата.

Множества Y^1 и Y^2 образуют множество выходных переменных Y: $Y=Y^1 \cup Y^2$; $Y^1 \cap Y^2 = \varnothing$. При этом $|Y^1|=N_1$, $|Y^2|=N_2$ *и* $N_1+N_2=N$.

Функция δ определяет состояние перехода $a_s \in A$ на основе текущего состояния $a_m \in A$ и входных переменных:

$$a_s = \delta(a_m, X). \quad (2)$$

Функции λ_1 и λ_2 имеют следующий вид:

$$y_n = \lambda(a_m, X). \quad (3)$$
$$y_n = \lambda(a_m). \quad (4)$$

Для реальных устройств переменные $x_l \in X$ и $y_n \in Y$ – физические объекты, принимающие значения "0" и "1". Состояния $a_m \in A$ – абстрактные объекты. Для синтеза схемы МПА состояния $a_m \in A$ представляются двоичными кодами $K(a_m)$ разрядности R, где $\lceil \log_2 M \rceil \le R \le M$ **.** Коды состояний хранятся в регистре памяти *RG*. Как правило, триггера регистра *RG* имеют входы типа *D*.

Закодируем состояния $a_m \in A$ двоичными кодами $K(a_m)$ разрядности $R = \lceil \log_2 M \rceil$. Для этого используем внутренние переменные, образующие множество $T=\{T_1,\ldots,T_R\}$. Для задания кода состояния перехода (2) используем функции возбуждения памяти, образующие множество $\Phi = \{D_1, \ldots, D_R\}$.

Для синтеза схемы СМПА необходимо получить функции (2) – (4). Они определяются следующими системами булевых функций:

$$\Phi = \Phi(T,X); \quad (5)$$
$$Y^1 = Y^1(T,X); \quad (6)$$
$$Y^2 = Y^2(T). \quad (7)$$

Системы функций (5) – (7) определяют структурную схему СМПА (рис. 1). Блок КС1 генерирует функции (5) – (6), блок КС2 – функции (7). Сигнал *Start* заносит в *RG* код начального состояния $a_1 \in A$. Импульс *Clock* вызывает переключение *RG*, соответствующее функции перехода (2).

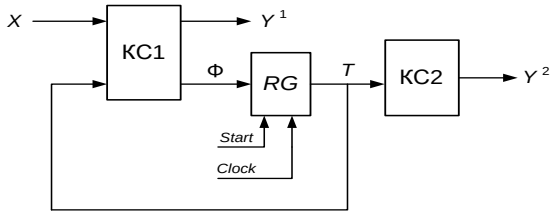

РИС. 1. Структурная схема совмещенного МПА

Особенность *FPGA* – наличие реконфигурируемых блоков *EMB*. Типичные конфигурации $V \times t_F$ – следующие: 32K×1, 16K×2, 8K×4, 4K×8, 2K×16, 1K×32 и 512×64 [4, 5]. Это определяет такие пары вида $<S_A, t_F>$: <15, 1>, <14, 2>, <13, 4>, <12, 8>>, <11, 16>, <10, 32> и <9, 64>. Таким образом, параметры *EMB* можно подбирать так, чтобы уменьшить число блоков в схеме СМПА.

Реализация совмещенного МПА в базисе *FPGA*. Существуют две тривиальные схемы совмещенного МПА в базисе *FPGA*. В автомате U_1 (рис. 2, а) схема реализуется на элементах *LUT*, образующих *LUTer*. В автомате U_2 (рис. 2, б) схема реализуется на одном блоке *EMB*.

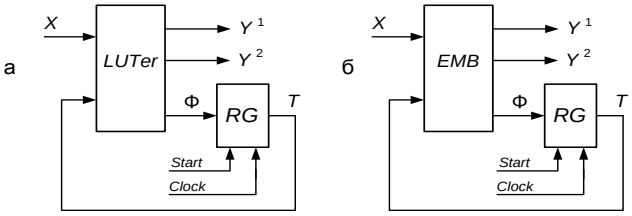

РИС. 2. Тривиальные схемы совмещенных МПА

В модели U_1 триггеры регистра *RG* распределены между элементами *LUT*. Поэтому регистр не существует, как отдельный блок. Если *EMB* является синхронным, то сигналы *Start* и *Clock* поступают на соответствующие входы *EMB*. Как правило, использование модели U_1 приводит к схемам, содержащим большое число уровней и межсоединений [6]. Применение модели U_2 приводит к схемам с наименьшей площадью, но эта модель может использоваться только при выполнении условия

$$2^{R+L} \cdot (N+R) \le V_0 . \qquad (8)$$

Условие (8), как правило, выполняется для относительно простых МПА. При его нарушении можно применить метод замены логических условий [7–12]. В этом случае множество X заменяется множеством дополнительных переменных $P=\{p_1,\ldots,p_G\}$, где $G << M$. Параметр G определяется, как максимум из мощностей множеств $X(a_m)\subseteq X$, определяющих переходы из состояний $a_m \in A$.

Для построения блока замены логических условий (ЗЛУ) необходимо найти систему функций

$$P = P(T,X). \tag{9}$$

Система (9) реализуется на LUT элементах, что определяет модель U_3 (рис. 3).

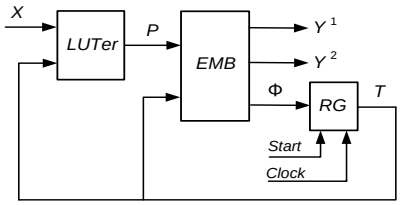

РИС. 3. Структурная схема СМПА U_3

В автомате U_3 $LUTer$ реализует систему (9), а EMB – систему (7) и системы

$$\Phi = \Phi(T,P); \tag{10}$$
$$Y^1 = Y^1(T,P). \tag{11}$$

Модель U_3 применима, если выполняется условие

$$2^{G+R} \cdot (N+R) \le V_0 . \tag{12}$$

Как показал анализ библиотеки [13] условие (12) выполняется для 82 % всех примеров.

Для уменьшения числа LUT в схеме U_3 необходимо уменьшить число аргументов в функциях $p_g \in P$. В настоящей работе предлагается метод решения этой задачи.

Основная идея предложенного метода. Метод, предложенный в данной работе, основан на кодировании классов псевдоэквивалентных состояний (ПЭС) [14]. Пусть в СМПА имеется I классов ПЭС. Закодируем класс $B_i \in \Pi_A$, где $\Pi_A = \{B_1, \ldots, B_I\}$ – разбиение множества A на классы ПЭС, двоичным кодом $K(B_i)$ разрядности R_1, где $R_1 = \lceil \log_2 I \rceil$. Используем для кодирования классов $B_i \in \Pi_A$ переменные $\tau_r \in \tau$, где $\tau=\{\tau_1,\ldots, \tau_{R1}\}$.

В дальнейшем мы покажем, что система (9) может быть заменена системой

$$P = P(\tau, X) . \tag{13}$$

Для реализации системы (13) необходимо преобразовать коды $K(a_m)$ в коды $K(B_i)$. Для этого требуется найти систему функций

$$V = V(T, P). \tag{14}$$

Система (14) включает R_1 функций.

Такой подход приводит к модели, включающей два регистра. Регистр RG хранит коды состояний $a_m \in A$. Регистр RG_1 хранит коды классов $B_i \in \Pi_A$. Обозначим эту модель символом U_4. Структурная схема модели U_4 показана на рис. 4.

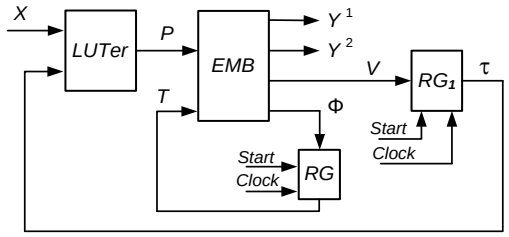

РИС. 4. Структурная схема автомата U_4

Модель U_4 применима при выполнении условия

$$2^{G+R} \cdot (N+R+R_1) \le V_0 . \tag{15}$$

Если условие (15) нарушается, то блок *EMB* заменяется блоком *EMBer*, включающим несколько блоков памяти. Такой вариант возможен, когда выполняется условие: $2^{G+R} \le V_0$.

Предлагаемый подход позволяет уменьшить число литералов и термов в системе функций (13) по сравнению с (9). Это происходит при выполнении условия

$$R_1 < R . \tag{16}$$

Анализ библиотеки [13] показал, что условие (15) выполняется для 100 % всех стандартных примеров. Это же справедливо и для условия (16).

В настоящей работе предлагается метод синтеза автомата U_4 по граф-схеме алгоритма (ГСА) Г. Метод включает следующие этапы:

1) формирование множеств A, Π_A, Y_1 и Y_2;
2) замена переменных $x_l \in X$ переменными $p_g \in P$;
3) кодирование состояний $a_m \in A$;
4) кодирование классов $B_i \in \Pi_A$;
5) формирование прямой структурной таблицы СМПА;
6) формирование таблиц элементов блока *LUTer*;
7) формирование таблицы блока *EMB*;
8) реализация схемы СМПА в заданном элементном базисе.

Пример применения предложенного метода. Рассмотрим пример синтеза автомата $U_4(\Gamma_1)$, где символ $U_i(\Gamma_j)$ означает, что модель U_i используется применительно к ГСА Γ_j. Граф-схема Γ_1 показана на рис. 5.

На дугах ГСА Γ_1 показаны выходные переменные автомата Мили, в операторных вершинах – автомата Мура. Таким образом, $Y^1=\{y_1, \dots, y_5\}$ и $Y^2=\{y_6, \dots, y_9\}$, что дает $N_1= 5$, $N_2= 4$, и $N= 9$.

Состояния $a_m \in A$ – это состояния автомата Мура. Следовательно, каждая операторная вершина отмечается уникальной отметкой [1]. Будем отмечать одинаковыми отметками вершины, если 1) их выходы связаны с входом одной и той же вершины ГСА и 2) в этих вершинах нет переменных автомата Мура. Такой прием позволяет отметить состоянием a_6 три вершины (рис. 5) и уменьшить число состояний и переходов между ними по сравнению с методом отметки [1]. Для ГСА Γ_1 имеем $A=\{a_1, \dots, a_6\}$, что дает $M = 6$, $R = 3$, $T = \{T_1, T_2, T_3\}$ и $\Phi = \{D_1, D_2, D_3\}$.

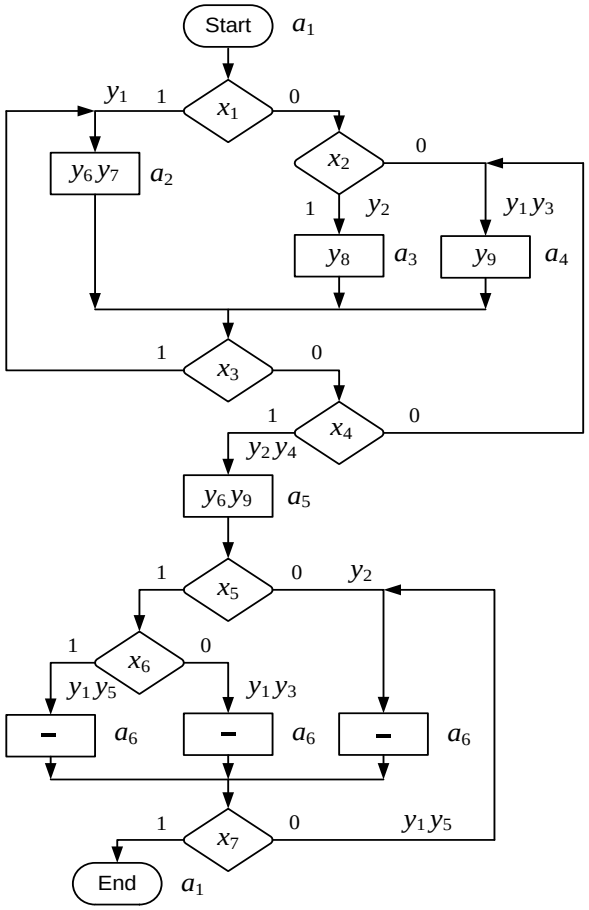

РИС. 5. Исходная ГСА Γ_1

Используем в качестве элементного базиса микросхему *FPGA* со следующими конфигурациями блоков *EMB*: 4K×1, 2K×2, 1K×4, 512×8 и 256×16 (битов). Пусть *LUT* элементы данной микросхемы имеют число входов S = 3.

Рассмотрим возможность реализации автомата $U_2(\Gamma_1)$ для выбранной микросхемы. Как следует из конфигурации 4K×1, емкость *EMB* V_0 = 4096 битов. Для ГСА Γ_1 имеем L= 7, R = 3 и N = 9. Проверим условие (8): $2^{10}\cdot(9+3) = 12\text{K} \le V_0 = 4\text{K}$.

Очевидно, для реализации систем (5) – (7) необходимо 3 блока памяти *EMB* с V_0 = 4K.

Проверим возможность реализации автомата $U_4(\Gamma_1)$. Найдем множество $\Pi_A = \{B_1,\ldots, B_I\}$. Как отмечено в [14], состояния a_m , $a_s \in A$ являются псевдоэквивалентными, если выходы отмеченных ими вершин связаны с входом одной и той же вершины ГСА Г. Анализ ГСА Γ_1 дает множество Π_A с классами $B_1=\{a_1\}$, $B_2=\{a_2, a_3, a_4\}$, $B_3=\{a_5\}$ и $B_4=\{a_6\}$. Таким образом, I= 4 и R_1= 2, что дает $\tau = \{\tau_1, \tau_2\}$.

Построим множества $X(a_m)\subseteq X$. Для ГСА Γ_1 имеем $X(a_1)=\{x_1, x_2\}$, $X(a_2)=X(a_3)=X(a_4)=\{x_3, x_4\}$, $X(a_5)=\{x_5, x_6\}$ и $X(a_6)=\{x_7\}$. Очевидно, множество $X(a_m)$ можно заменить множеством $X(B_i)$, если $a_m \in B_i$. Для Γ_1 имеем $X(B_1)=\{x_1, x_2\}$, $X(B_2)=\{x_3, x_4\}$, $X(B_3)=\{x_5, x_6\}$ и $X(B_4)=\{x_7\}$.

Число переменных G определяется формулой $G = max(L_1, \ldots , L_M)$, где $L_m=|X(a_m)|$, $a_m \in A$. В рассматриваемом примере имеем G = 2, что определяет множество $P=\{p_1, p_2\}$. В табл. 1 замены логических условий (ЗЛУ) переменная $x_l \in X$, заменяемая в состоянии $a_m \in A$ переменной $p_g \in P$, записывается на пересечении строки p_g и столбца B_i, где $a_m \in B_i$.

ТАБЛИЦА 1. Таблица замены логических условий

p_g / B_i	B_1	B_2	B_3	B_4
p_1	x_1	x_3	x_5	x_7
p_2	x_2	x_4	x_6	—

Закодируем состояния $a_m \in A$ и классы $B_i \in \Pi_A$. В данном случае исход кодирования не влияет на число элементов *LUT* и блоков *EMB*. Поэтому закодируем состояния и классы тривиальным образом: $K(a_1)$ = 000, ... , $K(a_6)$= 101, $K(B_1)$= = 00, ... , $K(B_4)$ = 11. Очевидно, $V = \{D_4, D_5\}$.

Проверим условие (15) для конфигурации 256×16, где S_A = 8 и t_F = 16. Так как $G+R = 5 < 5$ и $N+R+R_1 = 14 < t_F$, то модель $U_4(\Gamma_1)$ может быть использована. При этом схема включает только один блок *EMB*.

Прямая структурная таблица (ПСТ) автомата $U_4(\Gamma_j)$ должна задать системы (7), (10), (11) и (14). Эта таблица включает следующие столбцы: a_m – исходное состояние СМПА; $K(a_m)$ – код состояния $a_m \in A$; a_S – состояние перехода; $K(a_S)$ – код состояния $a_S \in A$; P_h – входной сигнал, определяющий переход $< a_m, a_S >$; Y^1_h – выходные переменные, формируемые на переходе $< a_m, a_S >$; Φ_h – функции возбуждения памяти *RG* для изменения ее содержимого из $K(a_m)$ в $K(a_S)$; B_i, $K(B_i)$

– класс B_i , где $a_S \in B_i$, и его код, соответственно; V_h – функции возбуждения памяти RG1 для изменения ее содержимого из $K(B_k)$, где $a_m \in B_k$ в $K(B_i)$; h – номер перехода. Кроме того, в столбце a_m записываются переменные $y_n \in Y^2$, формируемые в состоянии $a_m \in A$. Фрагмент ПСТ приведен в табл. 2.

ТАБЛИЦА 2. Фрагмент ПСТ автомата $U_4(\Gamma_1)$

a_m	$K(a_m)$	a_S	$K(a_S)$	P_h	Y^1_h	Φ_h	B_i	$K(B_i)$	V_h	h
a_2 $(y_6 y_7)$	001	a_2	001	p_1	$y_3\, y_5$	D_3	B_2	01	D_5	4
		a_5	100	$\bar{p}_1\, p_2$	$y_2\, y_3$	D_1	B_3	10	D_4	5
		a_4	011	$\bar{p}_1\, \bar{p}_2$	$y_1\, y_5$	$D_2\, D_3$	B_2	01	D_5	6

Для формирования схемы блока *LUTer* необходимо:

- получить уравнения для функций $p_g \in P$;
- выполнить декомпозицию уравнений и найти уравнения для каждого элемента *LUT*;
- построить таблицу содержимого каждого элемента *LUT*.

Например, используя коды $K(B_i)$, из табл. 1 получаем следующее уравнение:

$$p_1 = \overline{\tau_1}\,\overline{\tau_2} x_1 \vee \overline{\tau_1}\tau_2 x_3 \vee \tau_1 \overline{\tau_2} x_5 \vee \tau_1 \tau_2 x_7 \,. \qquad (17)$$

Число букв в (17) превышает S=3. Разложим (17) по переменной τ_1:

$$p_1 = \overline{\tau_1}\left(\overline{\tau_2} x_1 \vee \tau_2 x_3\right) \vee \tau_1 \left(\overline{\tau_2} x_5 \vee \tau_2 x_7\right) = \overline{\tau_1} A \vee \tau_1 B \,. \qquad (18)$$

Очевидно, части уравнения (18), обозначенные A и B, могут быть реализованы на одном элементе *LUT* каждая. Это дает схему для p_1 (рис. 6).

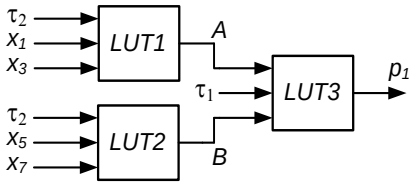

РИС. 6. Схема для функции P_1

Отметим, что для автомата $U_3(\Gamma_1)$ схема для P_1 реализуется на пяти элементах *LUT* с S = 3 и имеет три уровня. Схема для P_2 автомата $U_4(\Gamma_1)$ содержит 3 элемента *LUT* и два уровня. Схема для P_2 автомата $U_3(\Gamma_1)$ содержит 4 элемента и имеет 3 уровня логики. Таким образом, блок *LUTer* для $U_3(\Gamma_1)$ содержит 9 элементов *LUT* и имеет 3 уровня логики, а блок *LUTer* для $U_4(\Gamma_1)$ имеет 6 элементов

LUT и 2 уровня логики. Следовательно, предложенный метод позволяет в 1,5 раза уменьшить число элементов *LUT* с $S = 3$ и время распространения сигнала по сравнению с методом, использующим модель $U_3(\Gamma_1)$. Этап 3 для блока *LUTer* в данной статье не рассматривается в силу его тривиальности.

Таблица блока *EMB* строится на основе ПСТ. Она имеет столбцы $K(a_m)$, P (адрес ячейки памяти) и Ф, Y^1, Y^2, V (содержимое ячейки), q. В общем случае эта таблица имеет H строк, где $H = 2^{G+R}$.

В рассматриваемом примере $H = 32$. Остальные 224 ячейки блока *EMB* не используются. Переходы из каждого состояния задаются с помощью $H(a_m)$ строк, где $H(a_m) = 2^G$. В нашем примере $H(a_m) = 4$.

Таблица блока *EMB* является таблицей истинности, соответствующей ПСТ. Переменные $y_n \in Y^2$ записываются во всех строках, соответствующих состоянию $a_m \in A$, в котором они формируются.

Для выбранного базиса имеем $S_A = 8$. Пусть ПСТ автомата $U_4(\Gamma_1)$ занимает ячейки памяти с адресами <000$T_1T_2T_3$ p_1 p_2>. Тогда строке 5 табл. 2, например, соответствует ячейка с адресом <00000101>. Содержимое ячейки памяти определяется вектором <D_1, D_2, D_3, y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, D_4, D_5, 0, 0>. Так, строке 5 соответствует вектор <1000101010011000>. Аналогично находятся адреса и содержимое всех ячеек памяти. Мы не показываем таблицу *EMB* для данного примера в силу ее громоздкости.

Выводы. Предложенный в работе метод позволяет реализовать схему СМПА с использованием одного блока *EMB* и минимального числа элементов *LUT*. Первое достигается благодаря замене логических условий дополнительными переменными, второе – за счет кодирования классов псевдоэквивалентных состояний. Основываясь на результатах [7 – 12], можно утверждать, что в этом случае схема СМПА занимает минимально возможную площадь кристалла и потребляет минимальную энергию. Кроме того, уменьшение числа уровней логики в схеме блока *LUTer* позволяет повысить быстродействие схемы по сравнению с известными подходами [7 – 12].

Анализ библиотеки [13] показал, что применение предложенного метода позволяет получить схемы с одним блоком *EMB* для 100 % стандартных примеров. Отметим, что можно уменьшить число требуемых выходов *EMB* до $R+N$, если использовать метод оптимального кодирования состояний [14]. Число элементов *LUT* можно уменьшить, разбив множество входных переменных [15].

Дальнейшее направление наших исследований связано с адаптацией подходов [14, 15] к особенностям совмещенного автомата.

1. *Baranov S.* Logic Synthesis for Control Automata. Dordrecht: Kluwer Academic Publishers, 1994. 312 p.
2. *DeMicheli G.* Synthesis and Optimization of Digital Circuits. New York: McGraw-Hill, 1994. 636 p.
3. *Соловьев В.В.* Проектирование цифровых схем на основе программируемых логических интегральных схем. М.: Горячая линия – ТЕЛЕКОМ, 2001. 636 с.

4. *Skliarova I., Sklyarov V., Sudnitson A.* Design of FPGA-based circuits using Hierarchical Finite State Machines. Tallinn: TUT Press, 2012. 240 p.
5. *Грушницкий Р.И., Мурсаев А.Х., Угрюмов Е.П.* Проектирование систем с использованиием микросхем программируемой логики. СПб: БХВ. Петербург, 2002. 608 с.
6. *Sklyarov V., Skliarova I., Barkalov A., Titarenko L.* Synthesis and Optimization of FPGA-based Systems. Berlin: Springer, 2014. 432 p.
7. *Cong J. and Yan. K.* Synthesis for FPGAs with Embedded Memory Blocks. *Proceeding of the 2000 ACM/SIGDA 8th International Symposium on FPGAs.* 2000. P. 75 – 82.
8. *Garcia–Vargas L., Senhadji–Navarro R., M. Civit–Balcells A. and Guerra–Gutierrezz P.* ROM–Based Finite State Machine Implementation in Low Cost FPGAs. *IEEE International Simposium on Industrial Electronics, Vigo.* 2007. P. 2342–2347.
9. *Nowicka M., Łuba T. and Rawski V.* FPGA–based decomposition of boolean functions: algorithms and implementations. *Advanced Computer Systems*. 1999. P. 502 – 509.
10. *Rawski M., Selvaraj H. and Łuba T.* An application of functional decomposition in ROM–based FSM implementation in FPGA devices. *Journal of System Architecture* 51(6–7). 2005. P. 424 – 434.
11. *Rawski M., Tomaszewicz P., Borowski G. and Łuba, T.* Logic Synthesis Method of Digital Circuits Designed for Implementation with Embedded Memory Blocks on FPGAs. *Design of Digital Systems and Devises. LNEE 70, Springer, Berlin.* 2011. P. 121 – 144.
12. *Tiwari A. and Tomko K.* Saving power by mapping finite state machines into embedded memory blocks in FPGAs, Proceedings of Design Automation and Test in Europe. 2004. Vol. 2. P. 916 – 921.
13. *Yang S.* Logic Synthesis and optimization benchmarks user guide. *Microelectronics Center of North Carolina.* 1991. 43 p.
14. *Баркалов А.А.* Принципы оптимизации логической схемы микропрограммного автомата Мура. *Кибернетика и системный анализ.* 1998. № 1. С. 65 – 72.
15. *Barkalov A., Titarenko L., Kolopenczyk M.* EMB–based design of Mealy FSM. *Proceedings of 12th IFAC Conference on programmable devices and embedded systems.* 2013. P. 215–220.

Получено 10.07.2016