УДК 004.274

[1]**Баркалов А.А.,** д.т.н.,
[1]**Титаренко Л.А.,** д.т.н.,
[2]**Визор Я.Е.,** к.т.н.,
[3]**Матвиенко А.В.**

СТРУКТУРНАЯ РЕДУКЦИЯ В СОВМЕЩЕННЫХ АВТОМАТАХ

[1]Зеленогурский университет, Польша
[2]Национальный авиационный университет
[3]Институт кибернетики им. В.М. Глушкова НАН Украины

A.Barkalov@iie.uz.zgora.pl
L.Titarenko@iie.uz.zgora.pl
yaviz@ukr.net
matv@online.ua

Предложен метод синтеза совмещенного микропрограммного автомата, ориентированный на базис FPGA. Метод позволяет получить схему с минимальным числом элементов LUT. Оптимизация достигается за счет преобразования кодов состояний автомата в коды классов псевдоэквивалентных состояний. Приведен пример синтеза автомата с использования предложенного метода

Ключевые слова: совмещенный автомат, FPGA, LUT, EMB, синтез, структурная редукция

Введение

Микропрограммный автомат (МПА) предназначен для осуществления управления в цифровых системах [1,2]. При синтезе схем МПА возникает актуальная задача уменьшения аппаратурных затрат [3]. Решение этой задачи позволяет уменьшить площадь кристалла СБИС, занимаемую схемой МПА. В свою очередь это позволяет уменьшить энергию, потребляемую схемой [4], что особо важно в мобильных и автономных устройствах. Методы решения этой задачи во многом зависят от типа МПА и элементов базиса, используемого для реализации схемы автомата. В настоящей работе предлагается метод уменьшения аппаратурных затрат в схеме совмещенного МПА (СМПА), реализуемого в базисе СБИС типа *FPGA* (*field-programmable logic arrays*).

Особенностью СМПА является наличие выходных сигналов двух типов[1]. Выходные сигналы автомата Мили зависят от входных переменных и состояний, а автомата Мура – только от состояний [1,2]. Это позволяет использовать методы оптимизации МПА Мили и Мура для оптимизации схемы СМПА [5 – 7].

Базис *FPGA* [8,9] сейчас широко применяется для проектирования цифровых систем [10,11]. Для реализации схемы СМПА можно использовать логические элементы типа *LUT* (*look-up table*), программируемые триггера и блоки памяти *EMB* (*embedded memory blocks*). Для соединения элементов схемы и ее связи с другими схемами используется программируемая матрица межсоединений [8,9].

Метод структурной редукции [12] предполагает увеличение числа уровней в схеме МПА. Такой подход позволяет уменьшить число логических элементов по сравнению с одноуровневыми схемами. При этом каждый уровень может быть реализован с использованием различных логических элементов (гетерогенная реализация схемы МПА) [12]. Эта концепция идеально соответствует базису *FPGA*, в котором существуют элементы *LUT* и *EMB*. Например, элементы *LUT* целесообразно использовать для замены входных переменных, а *EMB* – для реализации функций возбуждения памяти. Предлагаемый в данной работе подход основан на замене входных переменных и

преобразовании кодов псевдоэквивалентных состояний (ПЭС) автомата Мура [13].

Особенности совмещенного МПА и FPGA

Математической моделью СМПА является 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.

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

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

$$a_s = \delta(a_m, X). \qquad (1)$$

Функции λ_1 и λ_2 имеют следующий вид:

$$y_n = \lambda_1(a_m, X). \qquad (2)$$

$$y_n = \lambda_2(a_m). \qquad (3)$$

Для реализации схемы СМПА состояния $a_m \in A$ необходимо закодировать двоичными кодами $K(a_m)$. Коды состояний хранятся в специальном регистре *RG*, состоящим из R триггеров с общими входами обнуления (*Start*) и синхронизации (*Clock*). Как правило, триггера имеют входы типа *D* [3]. R – число бит кода $K(a_m)$ находится в интервале $\lceil \log_2 M \rceil \le R \le M$. Будем рассматривать случай, когда

$$R = \lceil \log_2 M \rceil. \qquad (4)$$

Для кодирования состояний используются внутренние переменные $T_r \in T$, где $T=\{T_1,\ldots,T_R\}$. Для изменения содержимого RG используются функции возбуждения памяти, образующие множество $\Phi = \{D_1,\ldots, D_R\}$.

Для синтеза схемы СМПА необходимо получить функции (1) – (3)., которые определяются, соответственно, следующими системами булевых функций:

$$\Phi = \Phi(T,X); \qquad (5)$$

$$Y^1 = Y^1(T,X); \qquad (6)$$

$$Y^2 = Y^2(T). \qquad (7)$$

Системы функций (5) – (7) определяют структурную схему СМПА (Рис. 1).

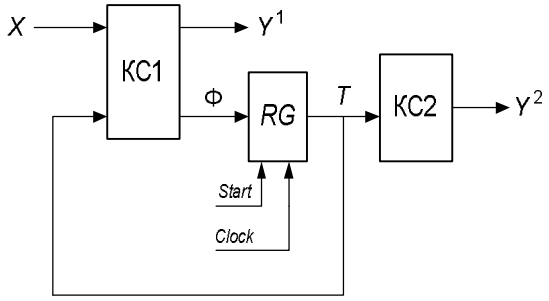

Рис.1. Структурная схема совмещенного МПА

В схеме на рис.1 блок КС1 реализует системы (5) и (6). Блок КС2 – функции (7). Сигнал *Start* обнуляет регистр *RG*, устанавливая код начального состояния $K(a_1)$. Сигнал *Clock* инициирует переключение *RG*, соответствующее функции (1).

Особенностью *FPGA* является наличие элементов памяти двух типов. Первый тип – элементы *LUT* (*look-up table*), имеющие S адресных входов и один выход. При этом параметр S относительно мал ($S \le 6$) [8, 9]. Выходы *LUT* могут быть связаны с входом триггера. Таким образом, регистр *RG* является распределенным. Второй тип *RAM* – элементы *EMB* (*embedded memory blocks*). Их важной характеристикой является реконфигурируе-

мость. При реконфигурации меняется число выходов (t_F) и адресных входов (S_A). При этом общая емкость (V_0) является константой:

$$V_0 = 2^{S_A} \times t_F . \qquad (8)$$

Типичными конфигурациями EMB являются следующие: 64K×1, 32K×2, 16K×4, 8K×8, 4K×16, 2K×32, 1K×64 (битов) [8, 9].

Здесь первый элемент пары определяет число ячеек памяти ($V = 2^{S_A}$), а второй – число выходов. Итак, для *EMB* следующие пары вида S_A, t_F : 16,1, 15,2,..., 10,64. Следовательно, *EMB* можно "настраивать" на системы (5) – (7). Это позволяет уменьшить число блоков памяти в схеме СМПА [5 – 7].

Реализация совмещенного МПА в базисе FPGA

Как показано в работах [5 – 7], существуют две тривиальные схемы совмещенного МПА в базисе *FPGA*. В первом случае (модель U_1) блоки КС1 и КС2 реализуются в виде блока *LUTer*. При этом под *LUTer* понимается схема, состоящая из элементов *LUT*. Недостатком модели U_1 является большое число уровней *LUT* и межсоединений между ними [3]. Во втором случае (модель U_2) блоки КС1 и КС2 реализуются на одном блоке *EMB*. Это приводит к схеме с наименьшей площадью, наибольшим быстродействием и наименьшей потребляемой энергией (по сравнению с другими возможными схемами) [11]. Однако эта модель может применяться только для достаточно простых автоматов, для которых выполняется условие

$$2^{R+L} \cdot (N_1 + N_2 + R) \le V_0 . \qquad (9)$$

При нарушении условия (9) необходимо использовать методы структурной редукции [12]. Наиболее часто используют метод замены входных переменных [10]. В этом случае множество X заменяется множеством дополнительных переменных $P=\{p_1,\ldots,p_G\}$, где $G << L$. Параметр G определяется, как минимум из $|X(a_m)|$, где $X(a_m) \subseteq X$ множество входных переменных, определяющих переходы из состояния $a_m \in A$.

Для замены входных переменных (ЗВП) необходимо найти систему функций

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

Система (10) реализуется на *LUT* элементах, что определяет модель U_3 (Рис. 2).

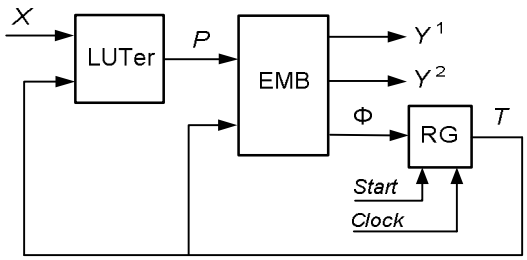

Рис. 2. Структурная схема СМПА U_3

В автомате U_3 блок *LUTer* реализует систему (10), а блок *EMB* – систему (7) и системы

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

$$Y^1 = Y^1(T,P) . \qquad (12)$$

Модель U_3 применима, если выполняется условие

$$2^{G+R} \cdot (N_1 + N_2 + R) \le V_0 . \qquad (13)$$

Как показал анализ библиотеки [14] условие (13) выполняется для 82% имеющихся в ней автоматов.

Для уменьшения числа элементов *LUT* в блоке *LUTer* необходимо уменьшить число аргументов в системе функций (10). В настоящей работе предлагается один из методов решения этой задачи. При этом алгоритм управления представляется в виде граф-схемы алгоритма (ГСА) [1].

Основная идея предложенного метода

Состояния $a_m \in A$ называются псевдоэквивалентными, если они отмечают вершины ГСА, которые связаны с входом одной и той же вершины [13]. Это определение позволяет найти разбиение $\Pi_A = \{B_1,\ldots, B_I\}$, где $B_i \in \Pi_A$ – класс ПЭС. Очевидно, что выполняется условие

$$I \le M. \qquad (14)$$

Закодируем классы $B_i \in \Pi_A$ двоичными кодами $K(B_i)$ разрядности R_1, где:

$$R_1 = \lceil \log_2 I \rceil. \qquad (15)$$

Используем для кодирования классов $B_i \in \Pi_A$ переменные $\tau_r \in \tau$, где $|\tau| = R_1$.

Пусть для данной ГСА Г и базиса FPGA выполняются следующие условия:

$$R_1 \le R. \qquad (16)$$

$$2^{G+R} \cdot (N + R + R_1) \le V_0. \qquad (17)$$

В этом случае мы предлагаем модель U_4 (Рис. 3).

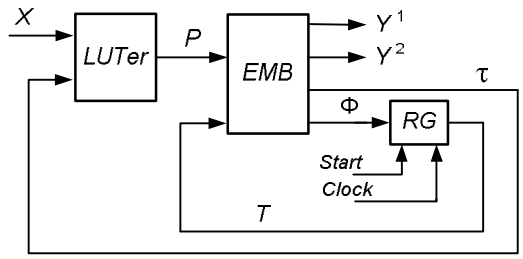

Рис. 3. Структурная схема СМПА U_4

В автомате U_4 блок *LUTer* реализует систему функций

$$P = P(\tau, X). \qquad (18)$$

Блок *EMB* автомата U_4 реализует системы функций (7), (11), (12) и систему

$$\tau = \tau(T). \qquad (19)$$

При выполнении условия (16) функции системы (18) имеют меньше аргументов, чем функции системы (10). Это приводит к уменьшению числа *LUT* элементов в блоке *LUTer* для автомата U_4 по сравнению с эквивалентным автоматом U_3. Отметим, что автоматы U_3 и U_4 являются эквивалентными, если они синтезируются по одной и той же ГСА Г.

Итак, схема СМПА U_4 содержит один блок *EMB* и меньше элементов *LUT* по сравнению с эквивалентным СМПА U_3. Это позволяет ожидать большего быстродействия и меньшей потребляемой энергии для U_4 относительно эквивалентного U_3.

В настоящей работе предлагается метод синтеза СМПА U_4 по ГСА Г. Метод включает следующие этапы:

1. Формирование множеств A, Π_A, Y^1 и Y^2.

2. Формирование множества P.

3. Кодирование состояний $a_m \in A$ и классов $B_i \in \Pi_A$.

4. Формирование таблицы входных переменных.

5. Формирование прямой структурной таблицы СМПА U_4.

6. Формирование таблиц элементов блоков *LUTer* и *EMB*.

7. Реализация схемы СМПА в заданном элементном базисе.

Пример применения предложенного метода

Рассмотрим пример синтеза СМПА $U4$ по ГСА Γ_1 (Рис. 4). Для отметки состояний мы использовали подход, предложенный в [5 – 7]. Операторные вершины отмечаются одинаковыми состояниями, если: 1) их выходы связаны с входом одной и той же вершины ГСА и 2) в этих операторных вершинах нет выходных переменных $y_n \in Y^2$.

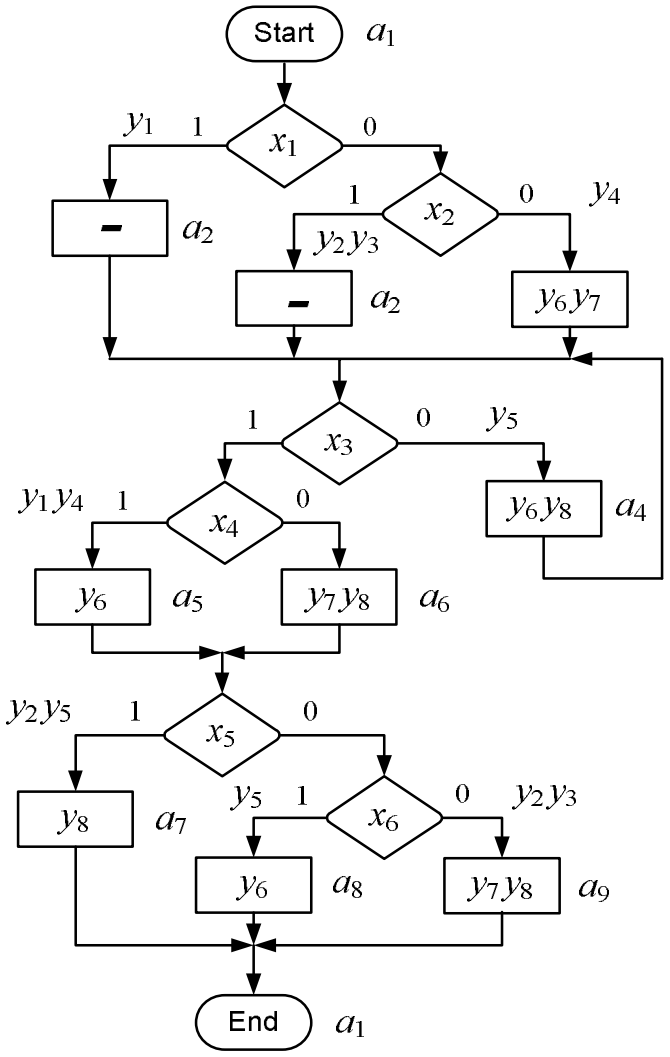

Рис. 4. Отмеченная ГСА Γ_1

Из рис. 4 можно найти следующие множества: $A=\{a_1,..., a_6\}$, $X=\{x_1,..., x_6\}$, $Y=\{y_1,..., y_8\}$, $Y^1=\{y_1,..., y_5\}$ и $Y^2=\{y_6, y_7, y_8\}$. Это дает следующие параметры: M=9, L=6, N=8, N_1= 5, N_2=3. Из (4) имеем R=4, что дает множества T = $\{T_1,..., T_4\}$ и $\Phi=\{D_1,..., D_4\}$.

Используя определение ПЭС [13], можно найти множество $\Pi_A = \{B_1,\ldots, B_4\}$, где $B_1=\{a_1\}$, $B_2=\{a_2, a_3, a_4\}$, $B_3=\{a_5, a_6\}$ и $B_4=\{a_7, a_8, a_9\}$. Итак, I=4, что дает R_1=2. Для ГСА Γ_1 выполняются условия (14) и (16).

Анализ ГСА Γ_1 показывает, что G = 2 (переходы из состояний $a_m \in A$ зависят от не более, чем двух переменных $x_l \in X$). Таким образом, имеем множество $P=\{p_1, p_2\}$.

Пусть среди конфигураций *EMB* имеется конфигурация 6, 16 с S_A=6 и t_F=16. Тогда для данного примера имеем: $G+R= 6 \le S_A$ и $N+R_1+R_2=14 \le t_F$. Это значит, что модель U_4 может быть использована для ГСА Γ_1 на рис. 4 и данного элементного базиса.

Закодируем состояния $a_m \in A$ тривиальным образом. Это дает коды $K(a_1)=0000$, ... , $K(a_9)=1000$. В общем случае классы $B_i \in \Pi_A$ можно закодировать так, чтобы уменьшить число аргументов и термов в системе (18). Это возможно, если переходы из состояний a_m для разных классов $B_i \in \Pi_A$ зависят от одинаковых входных переменных [11]. Поскольку в нашем случае это не так, закодируем классы $B_i \in \Pi_A$ тривиальным образом:

$K(B_1)$=00, $K(B_2)$=01, $K(B_3)$=10 и $K(B_4)$=11. Отметим, что для данного примера имеем $\tau = (\tau_1, \tau_2)$.

В общем случае таблица ЗВП имеет строки $p_1, \dots, p_G$ и столбцы $B_1, \dots, B_I$. Если переменная $x_l \in X$ заменяется переменной $p_g \in P$ для состояния $a_m \in B_i$, то на пересечении строки p_g и столбца B_i записывается переменная $x_l \in X$. В рассматриваемом примере ЗВП представлена в Табл.1.

Таблица 1. Таблица ЗВП автомата U_4

B_1	B_1	B_2	B_3	B_4
$K(B_i)$	00	01	10	11
p_1	x_1	x_3	x_5	—
p_2	x_2	x_4	x_6	—

Из табл.1 можно найти функции (18). Они имеют следующий вид:

$$p_1 = \overline{\tau}_1\overline{\tau}_2 x_1 \vee \overline{\tau}_1\tau_2 x_3 \vee \tau_1\overline{\tau}_2 x_5;$$
$$p_2 = \overline{\tau}_1\overline{\tau}_2 x_2 \vee \overline{\tau}_1\tau_2 x_4 \vee \overline{\tau}_1\overline{\tau}_2 x_6. \quad (20)$$

Прямая структурная таблица (ПСТ) автомата U_4 имеет следующие столбцы: a_m – текущее состояние автомата; $K(a_m)$ – 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 ; $Y^1{}_h$ – выходной сигнал автомата Мили, формируемый на переходе a_m, a_S ; τ_h – переменные $\tau_h \in \tau$, равные единице в коде $K(B_i)$, где $a_m \in B_i$; Φ_h – набор функций возбуждения памяти $D_r \in \Phi$, равных единице для загрузки в RG кода $K(a_S)$; h – номер перехода. Кроме того, в столбце a_m записывается набор функций $y_n \in Y^2$, формируемый в состоянии $a_m \in A$. В рассматриваемом примере ПСТ имеет H=21 строку. Часть ПСТ представлена в Табл.2.

Столбец P_h ПСТ заполняется следующим образом. Переход a_1, a_2 определяется термами x_1 (строка 1) и $\overline{x_1}x_2$ (строка 2). Из Табл.1 следует, что $x_1=p_1$ и $x_2=p_2$. Следовательно, в первой строке ПСТ записывается терм p_1, а во второй – $\overline{p_1}p_2$. Остальные строки Табл.2 формируются аналогично.

Таблица 2. Фрагмент ПСТ автомата U_4

a_m	$K(a_m)$	a_S	$K(a_S)$	P_h	$Y^1{}_h$	τ_h	Φ_h	h
a_2 (—)	0000	a_2	0001	p_1	y_1	—	D_4	1
		a_2	0001	$\overline{p_1}p_2$	$y_2 y_3$	—	D_4	2
		a_3	0010	$\overline{p_1}\overline{p_2}$	y_4	—	D_3	3
a_2 (—)	0001	a_5	0100	$p_1 p_2$	$y_1 y_4$	τ_2	D_2	4
		a_6	0101	$\overline{p_1}\overline{p_2}$	—	τ_2	$D_2 D_4$	5
		a_4	0011	$\overline{p_1}$	y_5	τ_2	$D_3 D_4$	6
a_3 ($y_6 y_7$)	0010	a_5	0100	$p_1 p_2$	$y_1 y_4$	τ_2	D_2	7
		a_6	0101	$\overline{p_1}\overline{p_2}$	—	τ_2	$D_2 D_4$	8
		a_4	0011	$\overline{p_1}$	y_5	τ_2	$D_3 D_4$	9

Для построения таблицы элементов LUT блока $LUTer$ необходимо проанализировать систему (18). Пусть $V(p_g)$ – множество переменных, входящих в функцию $p_g \in P$. Если $S \geq |V(p_g)|$, то для реализации формулы $p_h \in P$ достаточно одного элемента LUT. В противном случае функцию p_h необходимо преобразовать с использованием правил функциональной декомпозиции [2, 3].

Пусть для данного примера S=4. Рассмотрим систему уравнений (20). Для функции p_1 имеем $|V(p_1)|$ = 5 S. Следовательно, уравнение необходимо преобра-

зовать. Преобразуем его следующим образом:

$$p_1 = \overline{\tau_1}\left(\overline{\tau_2} x_1 \vee \tau_2 x_3\right) \vee \vee \tau_1\left(\overline{\tau_2} x_5\right) = A \vee B \quad . \qquad 21)$$

Уравнение (21) соответствует схеме (Рис.5), включающей два элемента *LUT*. Например, *LUT*2 представляется таблицей истинности (Табл.3). Аналогичным образом преобразовывается уравнение для $p_2 \in P$.

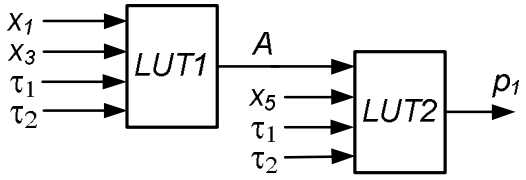

Рис. 5. Реализация функции p_1 при S=4

Таблица блока *EMB* строится на основе ПСТ. Она включает следующие столбцы: $K(a_m)$, P (адрес ячейки памяти), Y^1, Y^2, τ, Φ (содержимое ячейки памяти), q – номер ячейки памяти ($q = \overline{1, Q}$). Параметр Q определяется следующим образом: $Q = 2^{R+G}$.

Таблица 3. Таблица элемента LUT2

A τ1 τ2 x5	p1	A τ1 τ2 x5	p1
0 0 0 0	0	1 0 0 0	1
0 0 0 1	0	1 0 0 1	1
0 0 1 0	0	1 0 1 0	1
0 0 1 1	0	1 0 1 1	1
0 1 0 0	0	1 1 0 0	1
0 1 0 1	1	1 1 0 1	1
0 1 1 0	0	1 1 1 0	1
0 1 1 1	0	1 1 1 1	1

Переходы из состояний $a_m \in A$ задаются при помощи $H(a_m)$ строк таблицы, где $H(a_m)=2^G$.

В рассматриваемом примере Q=64 и $H(a_m)$=4. В табл.4 представлен фрагмент таблицы блока *EMB* для нашего примера.

Табл.4 задает переходы из состояния $a_3 \in A$, имеющего код 0010. Столбец h добавлен, чтобы показать соответствие между табл.2 и табл.4. Содержимое столбца Y^1 берется из столбца $Y^1{}_h$ ПСТ. Содержимое столбца Y^2 берется из столбца a_m ПСТ. Содержимое столбцов τ и Φ берется из столбцов ПСТ τ_h и Φ_h соответственно.

Последний этап предлагаемого метода связан с использованием стандартных пакетов для реализации схем в базисе *FPGA*. При этом таблицы блоков *LUTer* и *EMB* преобразовываются в битовые потоки (bit-stream). В данной статье этот этап не рассматривается.

Таблица 4. Фрагмент таблицы блока *EMB* автомата U_4

$K(a_m)$	P	Y^1	Y^2	τ	Φ	q	h
$T_1 T_2 T_3 T_4$	$p_1 p_2$	$y_1 y_2 y_6 y_4 y_5$	$y_6 y_7 y_8$	$\tau_1 \tau_2$	$D_1 D_2 D_3 D_4$		
0 0 1 0	0 0	0 0 0 0 1	1 1 0	0 1	0 0 1 1	9	9
0 0 1 0	0 1	0 0 0 0 1	1 1 0	0 1	0 0 1 1	10	9
0 0 1 0	1 0	0 0 0 0 0	1 1 0	0 1	0 1 0 1	11	8
0 0 1 0	1 1	1 0 0 1 0	1 1 0	0 1	0 1 0 0	12	7

Заключение

Предложенный в работе метод позволяет уменьшить число элементов *LUT* в схеме СМПА по сравнению с известными методами. Это достигается за счет преобразования кодов состояний МПА в коды классов псевдоэквивалентных состояний. Такой подход позволяет уменьшить число адресных входов в блоке замены входных переменных.

Метод целесообразно использовать, если замена входных переменных позволяет получить схемы с одним блоком *EMB*. Анализ библиотеки [14] показал, что к этому классу относится 82% стандартных автоматов. Кроме того, замена кодов $K(a_m)$ кодами $K(B_i)$ позволяет в среднем на 38% уменьшить число элементов *LUT* в блоке *LUTer*.

Дальнейшее направление наших исследований связано с адаптацией подходов [3, 11] к особенностям совмещенного автомата.

Список литературы

1. Baranov S. Logic Synthesis for Control Automata. – Dordrecht: Kluwer Academic Publishers, 1994. – 312 pp.

2. DeMicheli G. Synthesis and Optimization of Digital Circuits. – New York: McGraw-Hill, 1994. – 636 pp.

3. Sklyarov V., Skliarova I., Barkalov A., Titarenko L. Synthesis and Optimization of FPGA-based Systems. –Berlin: Springer, 2014. – 432pp.

4. Tiwari A. and Tomko, K. (2004). Saving power by mapping finite state machines into embedded memory blocks in FPGAs, Proceedings of Design Automation and Test in Europe, Vol. 2, pp. 916–921.

5. А.А. Баркалов, Л.А. Титаренко, Я.Е. Визор, А.В. Матвиенко. Синтез совмещенного микропрограммного автомата в базисе FPGA. – Комп'ютерні засоби, мережі та системи. Збірник наукових праць. Ін-т кібернетики ім. В.М.Глушкова НАН України. – Київ, 2015 Випуск 14,– С. 32-39.

6. А.А. Баркалов, Л.А. Титаренко, Я.Е. Визор, А.В. Матвиенко. Реализация схемы совмещенного микропрограммного автомата в базисе FPGA. – Проблеми інформатизації та управління. Збірник наукових праць. Національний авіаційний університет. – Київ, 2015 Випуск 3(51), – С 5-13.

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

8. www.altera.com.

9. www.xilinx.com.

10. Barkalov A., Titarenko L. Logic Synthesis for FSM-based Control Units. – Berlin: Springer, 2009. – 233pp.

11. Barkalov A., Titarenko L., Kolopenczyk M., Mielcarek K., Bazydlo G. Logic Synthesis for FPGA-based Finite State Mashines. – Berlin: Springer, 2016. – 280pp.

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

13. Баркалов А.А. Принципы оптимизации логической схемы микропрограммного автомата Мура // Кибернетика и системный анализ. – 1998, №1. – стр. 65-72.

14. Yang S. Logic synthesis and optimization benchmarks user guide. Microelectronics Center of North Carolina. – 1991, 43 pp.

Статью представлено в редакцию 10.06.2017