УДК 004.274

А.А. Баркалов, Л.А. Титаренко, Я.Е. Визор, А.В. Матвиенко

Уменьшение аппаратурных затрат в совмещенных автоматах

Предложен метод синтеза совмещенного микропрограммного автомата, ориентированный на базис *FPGA*, позволяющий получить схему с минимальным числом элементов *LUT*. Оптимизация достигается преобразованием кодов состояний автомата в коды классов псевдоэквивалентных состояний. Приведен пример синтеза автомата с использованием данного метода.

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

Запропоновано метод синтезу суміщеного мікропрограмного автомата, орієнтований на базис *FPGA*, який дозволяє отримати схему з мінімальним числом елементів *LUT*. Оптимізація досягається перетворенням кодів стану автомата у коди класів псевдоеквівалентних станів. Подено приклад синтезу автомата з використанням даного методу.

Ключові слова: суміщений автомат, *FPGA*, *LUT*, *EMB*, синтез, структурна редукція.

Введение. Микропрограммный автомат (МПА) предназначен для осуществления управления в цифровых системах [1, 2], при синтезе схем которого 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 особенно важно в мобильных и автономных устройствах. Методы решения этой задачи во многом зависят от типа МПА и элементного базиса, используемого для реализации схемы автомата. В статье предлагается метод уменьшения аппаратурных затрат в схеме совмещенного МПА (СМПА), реализуемого в базисе СБИС типа *FPGA* (*Field-Programmable Logic Arrays*).

Особенность СМПА – 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] широко применяется для проектирования цифровых систем [10, 11]. Для реализации схемы СМПА можно использовать логические элементы типа *LUT* (*Look-up Table*), программируемые триггеры и блоки памяти *EMB* (*Embedded Memory Blocks*). Для соединения элементов схемы и ее связи с другими схемами используется программируемая матрица межсоединений [8, 9].

Постановка задачи

Метод структурной редукции [12] предполагает увеличение числа уровней в схеме МПА. Такой подход позволяет уменьшить число логических элементов в сравнении с одноуровневыми схемами. При этом каждый уровень может быть реализован с использованием различных логических элементов (гетерогенная реализация) [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 подход основан на замене входных переменных и преобразовании кодов псевдоэквивалентных состояний (ПЭС) автомата Мура [13].

Особенности совмещенного МПА и микросхем *FPGA*

Математической моделью СМПА есть восьмикомпонентный вектор

$$S = < A, X, Y^1, Y^2, \delta, \lambda_1, \lambda_2, a_1 > .$$

Вектор S включает следующие компоненты: $A = \{a_1, \dots, a_M\}$ – множество внутренних со-

стояний; $X=\{x_1,\dots,x_L\}$ – 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 СМПА.

Y^1 и Y^2 образуют множество выходных переменных Y. При этом $Y^1 \bigcup Y^2 = Y$ 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:

$$a_S = \delta(a_m, X). \quad (1)$$

Функции λ_1 и λ_2 имеют следующий вид:

$$y_n = \lambda_1(a_m, X); \quad (2)$$

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

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

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

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

При синтезе схемы СМПА необходимо найти функции (1) – (3). Они определяются, соответственно, следующими системами:

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

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

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

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

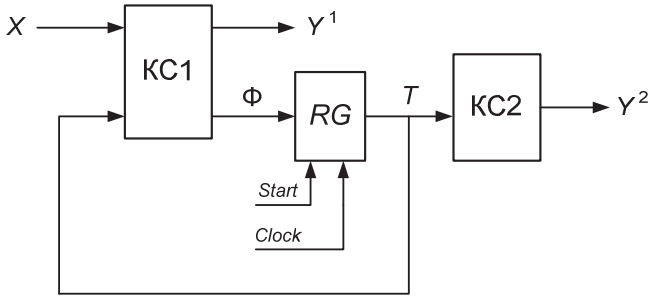

Рис. 1. Структурная схема СМПА

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

Особенность *FPGA* – наличие элементов памяти двух типов. Первый тип – элементы *LUT*, имеющие *S* адресных входов и один выход. При этом параметр *S* относительно мал ($S \le 6$) [8, 9]. Выход элемента *LUT* может быть связан с входом триггера. Таким образом, регистр *RG* распределен. Второй тип памяти – элементы встроенной памяти *EMB*. Их важная характеристика – способность реконфигурации. При реконфигурации меняется число выходов (t_F) и адресных входов (S_A). Однако общая емкость памяти (V_0) – константа:

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

Для современных *EMB* существуют следующие конфигурации: 64*K*×1, 32*K*×2, 16*K*×4, 8*K*×8, 4*K*×16, 2*K*×32, 1*K*×64 (битов) [8, 9]. Здесь первый элемент пары определяет число ячеек памяти ($V = 2^{S_A}$), а второй – число выходов блока. Итак, для *EMB* имеются следующие пары вида S_A, t_F : 16,1, 15,2, ..., 10,64. Следовательно, *EMB* можно *настраивать* на системы (5) – (7), что позволяет уменьшить число блоков памяти в схеме СМПА [5–7].

Реализация СМПА в базисе *FPGA*

Как показано в работах [5–7], существуют две тривиальные схемы СМПА 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*. Недостаток модели U_1 – значи-

тельное 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 (в сравнении с другими возможными схемами) [11]. Однако эта модель может применяться только для достаточно простых автоматов, для которых выполняется условие

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

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

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

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

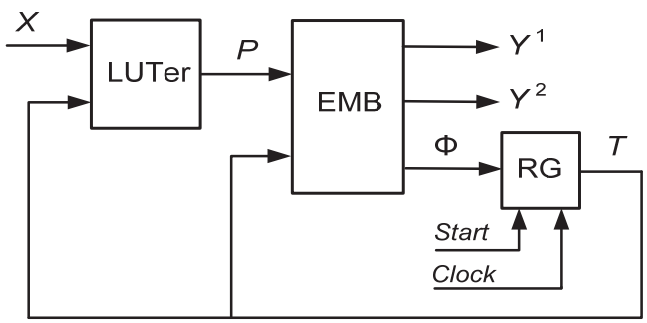

Рис. 2. Структурная схема СМПА U_3

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

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

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

Модель U_3 применима, если выполняется условие

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

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

Для уменьшения числа элементов *LUT* в блоке *LUTer* необходимо уменьшить число аргументов в системе функций (10). В статье 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 ими отмечены вершины ГСА, выходы которых 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

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

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

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

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

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

$$R_1 \le R; \quad (16)$$

$$2^{G+R}(N_1 + R) > V_0; \quad (17)$$

$$2^{G+R_1}(N_1 + R) \le V_0; \quad (18)$$

$$R \le S. \quad (19)$$

Условие (16) свидетельствует, что использование классов ПЭС приводит к уменьшению числа аргументов в функциях $p_g \in P$. Условие (17) показывает, что даже при ЗВП функции $y_n \in Y^1$ и $D_r \in \Phi$ нельзя реализовать на одном блоке *EMB*. Условие (18) показывает, что при ЗВП и кодировании классов ПЭС системы $y_n \in Y^1$ и $D_r \in \Phi$ реализуются на одном блоке *EMB*. Условие (19) свидетельствует, что любая из функций $y_n \in Y^2$ реализуется на одном элементе *LUT*.

В случае выполнения условий (16) – (19) мы предлагаем модель U_4 (рис. 3).

Как следует из схемы (рис. 3), системы функций (10) – (12) преобразовываются, соответственно, в системы

$$P = P(\tau, X). \quad (20)$$

$$\Phi = \Phi(\tau, P). \quad (21)$$

$$Y^1 = Y^1(\tau, P). \quad (22)$$

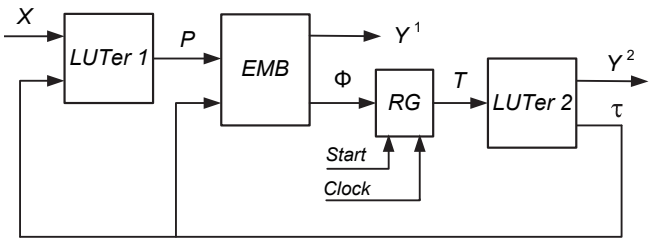

Рис. 3. Структурная схема СМПА U_4

Система (20) реализуется блоком *LUTer*1, системы (21) и (22) – блоком *EMB*. Блок *LUTer*2 реализует систему (7) и

$$\tau = \tau(T). \quad (23)$$

Такой подход позволяет использовать только один блок *EMB*. Кроме того, число элементов *LUT* в блоке *LUTer*1 будет меньше, чем в блоке *LUTer* эквивалентного автомата U_3. Отметим, что автоматы U_3 и U_4 – эквивалентны, если они синтезируются по одной и той же ГСА.

Недостаток автомата U_4 – наличие блока *LUTer*2. Однако при выполнении условия (19) блок *LUTer*2 состоит из $R_1 + N_2$ элементов *LUT*. Анализ библиотеки [14] показал, что этот недостаток практически компенсируется уменьшением числа элементов *LUT* в блоке *LUTer*1 (в сравнении с блоком *LUTer*).

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

- формирование множеств A, Π_A, Y^1 и Y^2;
- формирование множества P;
- кодирование состояний $a_m \in A$ и классов $B_i \in \Pi_A$;
- формирование таблицы входных переменных;
- формирование прямой структурной таблицы СМПА U_4;
- формирование таблиц элементов блоков *LUTer*1, *EMB* и *LUTer*2;
- реализация схемы СМПА в заданном элементном базисе.

Пример применения предложенного метода

Рассмотрим пример синтеза СМПА *U*4 по ГСА Γ_1 (рис. 4). Для отметки состояний использован подход из работ [5–7]. Операторные вершины отмечаются одинаковыми состояниями, если:

- их выходы связаны с входом одной и той же вершины ГСА;
- в этих операторных вершинах нет выходных переменных $y_n \in Y^2$.

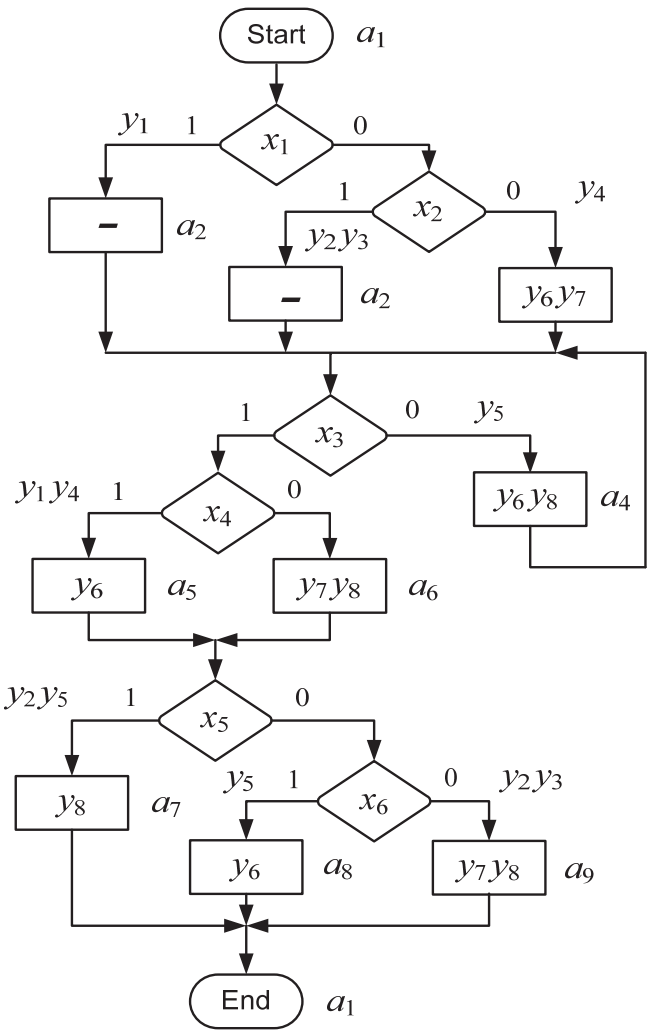

Рис. 4. Пример синтеза СМПА *U*4 по ГСА Γ_1

Из ГСА Γ_1 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 $R_1 = 2$. Очевидно, что 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 блока *EMB* имеется конфигурация с $S_A = 4$ и $t_F = 10$. Тогда для ГСА Γ_1 и данного блока памяти выполняются следующие условия: $G + R_1 = 4 \le S_A$ и $N_1 + R = 9 \le t_F$. Следовательно, системы (21) – (22) реализуются на одном блоке *EMB*. Пусть для элементов *LUT* число входов $S = 4$. В этом случае условие (19) выполняется. Таким образом, модель U_4 может быть применена в случае ГСА Γ_1 и данного элементного базиса.

Коды состояний $a_m \in A$ не влияют на число блоков *EMB*. Поэтому закодируем состояния тривиальным образом: $K(a_1) = 0000$, ..., $K(a_9) = 1000$. Для данного примера имеем множество $\tau = \{\tau_1, \tau_2\}$. В общем случае классы ПЭС можно закодировать так, чтобы уменьшить число литералов в функциях (20). Анализ ГСА Γ_1 показывает, что в ее условных вершинах нет одинаковых переменных $x_l \in X$. Поэтому, классы $B_i \in \Pi_A$ могут быть закодированы произвольным образом. Итак, закодируем классы ПЭС для нашего примера тривиальным образом: $K(B_1) = 00$, ..., $K(B_4) = 11$.

В общем случае таблица ЗВП имеет строки $p_1, \ldots, p_G$ и столбцы $B_1, \ldots, B_I$. 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. Замена входных переменных

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:

$$p_1 = \overline{\tau_1}\,\overline{\tau_2} x_1 \vee \overline{\tau_1}\tau_2 x_3 \vee \tau_1 \overline{\tau_2} x_5; \qquad (24)$$
$$p_2 = \overline{\tau_1}\,\overline{\tau_2} x_2 \vee \overline{\tau_1}\tau_2 x_4 \vee \tau_1 \overline{\tau_2} x_6.$$

Прямая структурная таблица (ПСТ) автомата U_4 имеет следующие столбцы: B_i – класс ПЭС, включающий текущее состояние $a_m \in A$; $K(B_i)$ – код класса $B_i \in \Pi_A$; a_s – состояние перехода; $K(a_s)$ – код состояния $a_s \in A$; P_h – набор переменных $p_g \in P$, определяющий переход a_m, a_S; $Y^1{}_h$ – выходные переменные $y_n \in Y^1$, формируемые на переходе a_m, a_s; Φ_h – набор функций возбуждения памяти, равных единице для переключения регистра *RG* из $K(a_m)$ в $K(a_s)$; h – номер перехода ($h = \overline{1, H}$).

В рассматриваемом примере $H = 10$ (табл. 2).

Таблица 2. ПСТ СМПА *U*4

B_i	$K(a_m)$	a_S	$K(a_S)$	P_h	$Y^1{}_h$	Φ_h	h
B_1	00	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
B_2	01	a_5	0100	$p_1 p_2$	$y_1 y_4$	D_2	4
		a_6	0101	$p_1 \overline{p_2}$	–	$D_2 D_4$	5
		a_4	0011	$\overline{p_1}$	y_5	$D_3 D_4$	6
B_3	10	a_7	0110	p_1	$y_2 y_5$	$D_2 D_3$	7
		a_8	0111	$\overline{p_1} p_2$	y_5	$D_2 D_3 D_4$	8
		a_9	1000	$\overline{p_1}\,\overline{p_2}$	$y_2 y_3$	D_1	9
B_4	11	a_1	0000	1	–	–	10

Поясним принцип заполнения столбца P_h. Класс B_2 состоит из состояний a_4, a_5, a_6. Конъюнкция $x_3 x_4$ определяет переходы $\langle a_2, a_5\rangle$, $\langle a_3, a_5\rangle$ и $\langle a_4, a_5\rangle$. Эти переходы задаются строкой четыре ПСТ (табл. 2). Из табл. 1 следует, что для класса $B_2 \in \Pi_A$ имеются равенства $p_1 = x_3$ и $p_2 = x_4$. Поэтому конъюнкция $x_3 x_4$ заменяется конъюнкцией $p_1 p_2$. Аналогично заполняются все строки табл. 2.

Для построения таблиц истинности для элементов *LUT* блока *LUTer* необходимо проанализировать систему (20). Пусть $V(p_g)$ – множество переменных, входящих в функцию $p_g \in P$. Если $S \ge |V(p_g)|$, то для реализации формулы $p_g \in P$ достаточно одного элемента *LUT*. В

противном случае функцию p_g необходимо преобразовать с использованием правил функциональной декомпозиции [2, 3].

Для данного примера элементы *LUT* имеют $S = 4$. Рассмотрим систему (20). Для функции p_1 имеем $|V(p_1)| = 5 > S$. Следовательно, уравнение p_1 необходимо преобразовать. Преобразуем его следующим образом:

$$p_1 = \overline{\tau}_1\left(\overline{\tau}_2 x_1 \vee \tau_2 x_3\right) \vee \tau_1\left(\overline{\tau}_2 x_5\right) = A \vee B. \qquad (25)$$

Уравнение (25) соответствует схеме, включающей два элемента *LUT* (рис. 5). Например, элемент *LUT*2 представляется таблицей истинности (табл. 3). Аналогично преобразовывается уравнение для $p_2 \in P$.

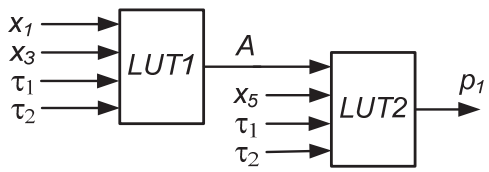

Рис. 5. Реализация функции p_1 при $S = 4$

Таблица 3. Значения элемента *LUT*2

$A\,\tau_1\tau_2 x_5$	p_1	$A\,\tau_1\tau_2 x_5$	p_1
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

Таблица блока *EMB* строится на основе ПСТ. Она включает следующие столбцы: $K(B_i)$, P (адрес ячейки памяти), Y^1, Ф (содержимое ячейки памяти), q – номер ячейки памяти ($q = \overline{1,Q}$). Параметр Q определяется как $Q = 2^{R_1+G}$ Переходы из состояний $a_m \in A$ задаются при помощи $H(B_i)$ строк таблицы, где $H(B_i) = 2^G$.

В рассматриваемом примере $Q = 16$ и $H(B_i) = 4$. В табл. 4 представлены первые восемь строк таблицы блока *EMB* для нашего примера.

Табл. 4 задает переходы из состояний, входящих в классы B_1 и B_2. Первые четыре строки соответствуют переходам для класса B_1, следующие четыре строки – для класса B_2. Столбец h добавлен, чтобы показать соответствие между табл. 2 и 4. Содержимое столбцов Y^1 и Ф берется из столбцов ${Y^1}_h$ и Ф исходной ПСТ (табл. 2).

Таблица 4. Фрагмент блока *EMB* автомата U_4

$K(B_i)$	P	Y^1	Ф	q	h
$\tau_1\tau_2$	p_1p_2	$y_1y_2y_6y_4y_5$	$D_1D_2D_3D_4$		
0 0	0 0	0 0 0 1 0	0 0 1 0	1	3
0 0	0 1	0 1 1 0 0	0 0 0 1	2	2
0 0	1 0	1 0 0 0 0	0 0 0 1	3	1
0 0	1 1	1 0 0 0 0	0 0 0 1	4	1
0 1	0 0	0 0 0 0 1	0 0 1 1	5	6
0 1	0 1	0 0 0 0 1	0 0 1 1	6	6
0 1	1 0	0 0 0 0 0	0 1 0 1	7	5
0 1	1 1	1 0 0 1 0	0 1 0 0	8	4

Таблица блока *LUTer*2 объединяет таблицы для функций $y_n \in Y^2$ и $\tau_r \in \tau$, она содержит столбцы: $K(a_m)$ – адрес ячейки памяти; Y^2, τ – содержимое ячейки памяти; q – номер ячейки ($q = \overline{1,Q_1}$). Параметр Q_1 определяется как $Q_1 = 2^R$. В рассматриваемом примере имеем $Q_1 = 16$ (табл. 5). Следующие комбинации не содержат полезной информации, поэтому они не показаны в табл. 5.

Таблица блока *LUTer*2 заполняется следующим образом. Например, строка $q = 4$ соответствует состоянию $a_4 \in B_2$. Из ГСА Γ_1 следует, что в состоянии a_4 формируются выходные функции $(y_6, y_8) \in Y^2$. Это соответствует коду 101 в столбце Y^2 строки четыре. Так как $K(B_2) = 01$, то в столбце τ этой строки находится код. Аналогичным образом заполняются все строки табл. 5.

Последний этап предлагаемого метода связан с применением стандартных пакетов [8, 9], используемых для реализации электронных схем в базисе *FPGA*. В статье этот этап не рассматривается.

Таблица 5. Значения блока *LUTer*2 автомата U_4

$K(a_m)$	Y^2	τ	q
$T_1T_2T_3T_4$	$y_6y_7y_8$	$\tau_1\tau_2$	
0 0 0 0	0 0 0	0 0	1
0 0 0 1	0 0 0	0 1	2
0 0 1 0	1 1 0	0 1	3
0 0 1 1	1 0 1	0 1	4
0 1 0 0	1 0 0	1 0	5
0 1 0 1	0 1 1	1 0	6
0 1 1 0	0 0 1	1 1	7
0 1 1 1	1 0 0	1 1	8
1 0 0 0	0 1 1	1 1	9

Заключение. Предложенный метод позволяет уменьшить число элементов *LUT* в схеме СМПА в сравнении с известными методами. Это достигается преобразованием 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 использовать только один блок *EMB* для реализации систем функций возбуждения памяти и входных переменных автомата Мили. Анализ библиотеки [14] показал, что к этому классу относятся 18 процентов стандартных автоматов. Кроме того, число адресных входов элементов *LUT* должно быть достаточным для реализации в виде одного элемента любой функции из множества $Y^2 \bigcup \tau$. Как показали исследования, замена состояний классами ПЭС позволяет компенсировать наличие блока *LUTer*2 уменьшением числа элементов *LUT* в блоке ЗВП. Таким образом, эквивалентные автоматы U_3 и U_4 имеют одинаковое число элементов *LUT*, но в автомате U_4 используется только один блок *EMB*.

Дальнейшие исследования связаны с адаптацией подходов [3, 11] к особенностям совмещенного автомата.

1. *Baranov S.* Logic Synthesis for Control Automata. – Dordrecht: Kluwer Acad. Publ., 1994. – 312 p.
2. *DeMicheli G.* Synthesis and Optimization of Digital Circuits. – New York: McGraw-Hill, 1994. – 636 p.
3. *Synthesis* and Optimization of FPGA-based Systems / V. Sklyarov, I. Skliarova, A. Barkalov et al. – Berlin: Springer, 2014. – 432 p.
4. *Tiwari A.*, *Tomko K.* Saving power by mapping finite state machines into embedded memory blocks in FPGAs // Proc. of Design Automation and Test in Europe, 2004. – **2**. – P. 916–921.
5. *Синтез* совмещенного микропрограммного автомата в базисе *FPGA* / А.А. Баркалов, Л.А. Титаренко, Я.Е. Визор и др. – Комп'ютерні засоби, мережі та системи: Зб. наук. праць. Ін-т кібернетики ім. В.М. Глушкова НАН України. – 2015. – **14**. – С. 32–39.
6. *Реализация* схемы совмещенного микропрограммного автомата в базисе *FPGA* / А.А. Баркалов, Л.А. Титаренко, Я.Е. Визор и др. – Проблеми інформатизації та управління: Зб. наук. праць. Національний авіаційний університет. – Київ, 2015. – **3**(51). – С. 5–13.
7. *Уменьшение* числа *LUT* элементов в схеме совмещенного автомата / А.А. Баркалов, Л.А. Титаренко, Я.Е. Визор и др. // УСиМ. – 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. – 233 p.
11. *Logic* Synthesis for FPGA-based Finite State Mashines / A. Barkalov, L. Titarenko, M. Kolopenczyk et al. – Berlin: Springer, 2016. – 280 p.
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. – С. 65–72.
14. *Yang S.* Logic synthesis and optimization benchmarks user guide. – Microelectronics Center of North Carolina. – 1991. – 43 p.

Поступила 29.06.2017

Тел. для справок: +38 044 526-2504, 526-3069 (Киев)
E-mail: A*A.Barkalov@iie.uz.zgora.pl*, *yaviz@ukr.net*, *matv@online.ua*

UDC 004.274

A.A. Barkalov[1], L.A. Titarenko[2], Y.E. Vizor[3], A.V. Matvienko[4]

[1] Doctor of Technical Sciences, Professor of Institute of Computer Engineering and Electronics, Institute of Informatics and Electronics. Zielenogorski University, ul. Podgorna, 50, Zielona Gora, 65-246, POLAND, a.barkalv@imei.uz.zgora.pl

[2] Doctor of Technical Sciences, Professor of Institute of Computer Engineering and Electronics, Institute of Informatics and Electronics. Zielenogorski University, ul. Podgorna, 50, Zielona Gora, 65-246, POLAND.

[3] Doctor of Technical Sciences, V.M. Glushkov Institute of Cybernetics of National Academy of Sciences of Ukraine, Kyiv, 03187, Glushkov ave., 40, Ukraine, +38 (044) 526-25-04, E-mail: yaviz@ukr.net.

[4] Researcher Associate, V.M. Glushkov Institute of Cybernetics of National Academy of Sciences of Ukraine, Kyiv, 03187, Glushkov ave., 40, Ukraine, +38 (044) 526-30-69, E-mail: matv@online.ua.

Reducing the Hardware Amount for the Combined Automata

Keywords: combined FSM, FPGA, LUT, EMB, synthesis, structural reduction.

Introduction. The proposed method allows to reduce the number of LUT elements in the scheme of a combined microprogram automatic machine in comparison with the known methods.

Purpose. This is achieved by transforming the codes of the states of the microprogram automaton into the class codes of the pseudoequivalent states. This approach reduces the number of input addresses in the block of the input variables replacement.

Purpose. It is advisable to use this method if the replacement of input variables allows the use of only one EMB block for the implementation of the memory excitation function systems and input variables of the Mili machine. The analysis shows that this class includes 18% of standard machines. In addition, the number of address inputs of LUTs must be sufficient to implement as one element of any function from the set.

Conclusion. As our studies have shown, the replacement of the states by pseudoequivalent state classes makes it possible to compensate the presence of the LUTer2 block due to the decrease in the number of LUT elements in the block for the input variables replacement.

The direction for our further research is connected with the adaptation of these approaches to the combined automatic machine features.

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. *Sklyarov V*., *Skliarova I.*, *Barkalov A*., *Titarenko L*. Synthesis and Optimization of FPGA-based Systems,Berlin: Springer, 2014, 432 p.
4. *TiwariA.*, *Tomko K.* Saving power by mapping finite state machines into embedded memory blocks in FPGAs, Proceedings of Design Automation and Test in Europe, Vol. 2, 2004, P. 916–921.
5. *Barkalov A.A*., *Titarenko L.A*., *Vizor Ja.E.*, *Matvienko A.V.* Sintez sovmeshhennogo mikroprogrammnogo avtomata v bazise FPGA, Komp'juterni zasobi, merezhi ta sistemi, Zbirnik naukovih prac', In-t kibernetiki im. V.M. Glushkova NAN Ukraïni, Kiïv, 2015, N 14, P. 32–39. (In Russian).
6. *Barkalov A.A.*, *Titarenko L.A.*, *Vizor Ja.E.*, *Matvienko A.V.* Realizacija shemy sovmeshhennogo mikroprogrammnogo avtomata v bazise FPGA. Problemi informatizaciï ta upravlinnja, Zbirnik naukovih prac'. Nacional'nij aviacijnij universitet, Kiïv, 2015, Vipusk 3(51), P. 5–13. (In Russian).
7. *Barkalov A.A*., *Titarenko L.A*., *Vizor Ja.E*., *Matvienko A.V*., *Gorina V.V.* Umen'shenie chisla LUT jelementov v sheme sovmeshhennogo avtomata, Upr. sist. mas., 2016, N3, P. 16–22. (In Russian).
8. *www*.altera.com.
9. *www*.xilinx.com.
10. *Barkalov A.*, *Titarenko L*. Logic Synthesis for FSM-based Control Units, Berlin: Springer, 2009, 233 p.
11. *Barkalov A.*, *Titarenko L.*, *Kolopenczyk M.*, *Mielcarek K*., *Bazydlo G*. Logic Synthesis for FPGA-based Finite State Mashines, Berlin: Springer, 2016, 280p.
12. *Solov'ev V.V.* Proektirovanie cifrovyh shem na osnove programmiruemyh logicheskih integral'nyh shem, M.: Gorjachaja linija – TELEKOM, 2001, 636 p. (In Russian).
13. *Barkalov A.A.* Principy optimizacii logicheskoj shemy mikroprogrammnogo avtomata Mura, Kibernetika i sistemnyj analiz, 1998, №1, P. 65–72. (In Russian).
14. *Yang S.* Logic synthesis and optimization benchmarks user guide. Microelectronics Center of North Carolina, 1991, 43 p.

●