УДК 004.274

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

Уменьшение числа *LUT*-элементов в схеме совмещенного автомата

Предложен метод синтеза совмещенного микропрограммного автомата в базисе *FPGA*, позволяющий получить схему с минимальным числом элементов *LUT* и встроенных блоков памяти *EMB*. Минимизация достигается путем замены части множества логических условий и соответствующего кодирования состояний автомата. Приведен пример применения метода.

Ключевые слова: совмещенный автомат, *FPGA*, *LUT*, *EMB*, синтез, граф-схема алгоритма.

Запропоновано метод синтезу суміщеного мікропрограмного автомата в базисі *FPGA*, що дозволяє отримати схему з мінімальним числом елементів *LUT* і вбудованих блоків пам'яті *EMB*. Мінімізація досягається шляхом заміни частини множини логічних умов та відповідного кодування станів автомата. Наведено приклад застосування методу.

Ключові слова: суміщений автомат, *FPGA*, *LUT*, *EMB*, синтез, граф-схема алгоритму.

Введение. Любая цифровая система включает устройство управления, для синтеза которого часто используется модель микропрограммного автомата (МПА) [1, 2]. Одна из моделей МПА – совмещенный автомат, в котором существуют выходные сигналы двух типов [3]. Выходные сигналы типа Мили формируются при переходе между состояниями. Выходные сигналы типа Мура существуют в течение такта работы МПА [2, 3].

Для реализации схем цифровых систем в настоящее время широко используются СБИС типа *FPGA* (*Field-Programmable Logic Arrays*) [4, 5]. Два типа логических элементов, входящих в *FPGA*, могут использоваться для реализации схемы МПА. Первый из них – логические элементы типа *LUT* (*Look-Up Table*), выходы которых могут быть связаны с входами триггеров. Элементы *LUT* имеют ограниченное число входов ($S \le 6$) и только один выход. Второй тип логических элементов – встроенные блоки памяти типа *EMB* (*Embedded Memory Blocks*). Их важная характеристика – реконфигурация, при которой меняется число выходов (t_F) и ячеек памяти (V). При этом общая емкость (V_0) есть константой:

$$V_0 = 2^{S_A} \cdot t_F,$$

где S_A – число адресных входов при данном количестве выходов t_F. Как правило, существуют следующие конфигурации *EMB*: 32K×1, 16K×2, 8K×4, 4K×8, 2K×16, 1K×32, 512×64 (битов) [4, 5]. Это определяет следующие пары вида S_A, t_F: 15, 1; 14, 2; 13, 4; 12, 8; 11, 16; 10, 32 и 9, 64.

При реализации МПА в базисе *FPGA* важно уменьшать площадь кристалла, занимаемого схемой. При этом улучшаются такие характеристики МПА, как время распространения сигналов и потребляемая мощность [6]. Один из подходов для решения этой задачи – замена элементов *LUT* блоками *EMB* [7–12]. Однако до сих пор никто не рассматривал эту задачу применительно к совмещенному МПА. В статье авторы предлагают одно из возможных решений и анализ условия его применения.

Особенности совмещенного МПА

Микропрограммный автомат задается шестикомпонентным вектором:

$$S = A, X, Y, \delta, \lambda, a_1,$$

где $A = \{a_1, ..., a_M\}$ – множество внутренних состояний, $X = \{x_1, \ldots, x_L\}$ – множество входных переменных, $Y = \{y_1, \ldots, y_N\}$ – множество выходных переменных, δ – функция переходов, λ – функция выходов, $a_1 \in A$ – начальное состояние. Функция δ служит для нахождения состояния перехода $a_s \in A$ на основе текущего состояния $a_m \in A$ и входных переменных:

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

Для автомата Мили функция λ определяет выходную переменную $y_n \in Y$:

$$y_n = \lambda(a_m, X). \quad (2)$$

Для автомата Мура выходные переменные определяются только внутренними состояниями:

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

В совмещенном МПА множество $Y = Y^1 U Y^2$, где Y^1 – множество выходных переменных типа Мили и Y^2 – множество выходных переменных типа Мура.

Граф переходов совмещенного автомата S_1 показан на рис. 1. Вершины графа соответствуют состояниям, а дуги – переходам между ними. Выходы типа Мура показаны рядом с вершинами, а выходы типа Мили – под дугами. Над дугами показаны входные сигналы, вызывающие переход. Как следует из рис. 1, $A = \{a_1,\ldots, a_4\}$, $X = \{x_1, x_2\}$, $Y^1 = \{y_1, y_2, y_4, y_6\}$, $Y^2 = \{y_3, y_4, y_5\}$. Это дает $M = 4$, $L = 2$, $N_1 = 4$ и $N_2 = 3$, где $N_1 = |Y^1|$ и $N_2 = |Y^2|$. Анализ множеств Y^1 и Y^2 позволяет получить соотношение $Y^1 \cap Y^2 = \varnothing$.

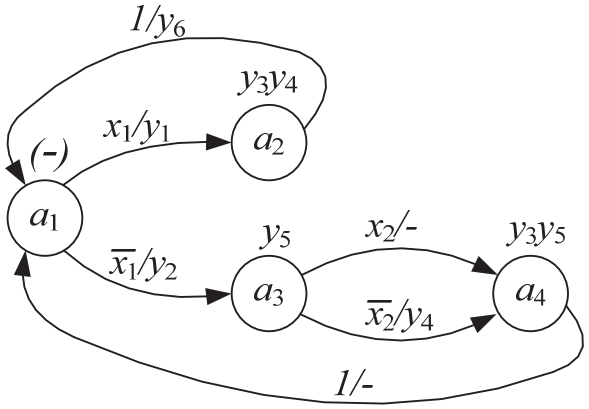


Рис. 1. Граф переходов совмещенного автомата S_1

Закодируем состояния $a_m \in A$ двоичными кодами $K(a_m)$, имеющими R разрядов:

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

Для кодирования состояний используем переменные $T_r \in T$, где $T = \{T_1,\ldots,T_R\}$.

Коды состояний хранятся в памяти МПА, которая обычно представляется регистром RG с D-входами [2]. Для переключения памяти используются функции возбуждения $D_r \in \Phi$, где $\Phi = \{D_1,\ldots, D_R\}$.

Для синтеза схемы совмещенного МПА необходимо получить функции δ и λ. Функция (1) определяется системой булевых функций (СБФ)

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

Системы (6) – (7) соответствуют функциям (2) – (3):

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

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

Системы (5) – (7) определяют структурную схему, приведенную на рис. 2.

Блок КС1 генерирует функции (5) – (6), блок КС1 – функции (7). Сигнал *Start* устанавливает в RG нулевой код начального состояния $a_1 \in A$. Импульс *Clock* вызывает переключение RG, что соответствует переходам МПА.

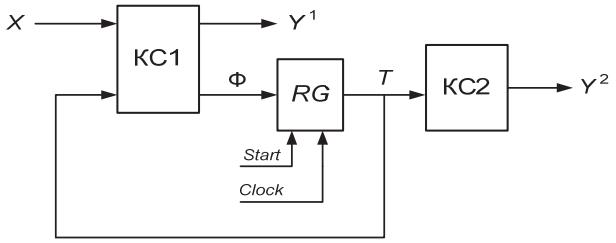
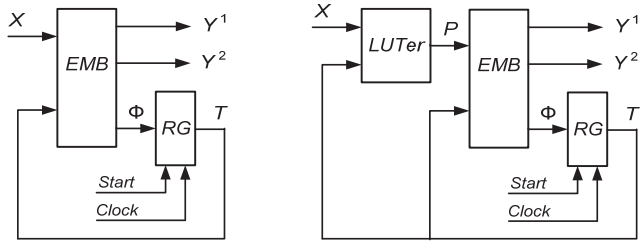


Рис. 2. Структурная схема совмещенного МПА

Реализация совмещенного МПА в базисе *FPGA*

Наилучшим с учетом аппаратурных затрат есть решение, при котором системы (5) – (7) реализуются на одном блоке *EMB*. Обозначим такую модель символом U_1 (рис. 3, *а*).



а *б*

Рис. 3. Структурная схема автомата U_1 (*а*) и U_2 (*б*)

В МПА U_1 (как и в других моделях) регистр RG реализуется на элементах *LUT*, связанных с D-триггерами [4, 6]. Если блоки *EMB* синхронны, то регистр отсутствует. При этом импульсы *Start* и *Clock* поступают на соответствующие входы *EMB*.

Модель U_1 используется, если выполняется следующее условие:

$$2^{R+L} \ (N + R) \le V_0 \quad (8)$$

При этом содержимое *EMB* определяется таблицей со столбцами $K(a_m)$, X (определяет адрес ячейки), Ф, Y^1, Y^2 (содержимое ячейки), q (номер ячейки) – табл. 1.

Таблица 1. Таблица блока *EMB* автомата S_1

$K(a_m)$	X	Ф	Y^1	Y^2	q
T_1T_2	x_1x_2	D_1D_2	$y_1y_2y_6$	$y_3y_4y_5$	
00	00	10	010	000	1
00	01	10	010	000	2
00	10	01	100	000	3
00	11	01	100	000	4
01	00	00	001	110	5
01	01	00	001	110	6
01	10	00	001	110	7
01	11	00	001	110	*8*
10	00	11	001	001	*9*
10	01	11	000	001	10
10	10	11	001	001	11
10	11	11	000	001	12
11	00	00	000	101	13
11	01	00	000	101	14
11	10	00	000	101	15
11	11	00	000	101	16

Для автомата $S1$ необходимо $R = 2$, что определяет множество $T = \{T_1, T_2,\}$ и $\Phi = \{D_1, D_2,\}$. Поскольку $L + R = 4$, таблица *EMB* для автомата S_1 имеет 16 строк. При этом состояния $a_m \in A$ закодированы тривиальным образом: $K(a_1) = 00, \ldots, K(a_4) = 11$.

Фактически табл. 1 есть таблицей истинности функций (5) – (7). Переходы из любого состояния $a_m \in A$ задаются четырьмя строками. В общем случае, это число $H(a_m)$ определяется формулой:

$$H(a_m) = 2^L.$$

Связь между графом (рис. 1) и табл. 1 очевидна.

Анализ стандартных бенимарков [13] показал, что условие (8) выполняется для 76 процентов всех примеров. Если это условие нарушено, то в работах [7–12] предлагается использовать метод замены входных переменных [1].

В этом случае множество X заменяется множеством дополнительных переменных $P = \{P_1,\ldots,P_G\}$, где $G << L$. Параметр G определяется, как максимум из мощностей множеств $X(a_m) \subseteq X$, определяющих переход из состояний $a_m \in A$. Например, для автомата S_1 имеются следующие множества: $X(a_1) = \{x_1\}$, $X(a_2) = X(a_4) = \varnothing$ и $X(a_3) = \{x_2\}$. Это определяет параметр $G = 1$ и множество $P = \{P_1\}$.

Для замены $X \to P$ необходимо найти систему функций:

$$P = P(T, X). \qquad (9)$$

Система (9) реализуется на блоке *LUTer*, что определяет модель U_2 (рис. 3,*б*). Под термином *LUTer* принимаем схему, реализованную на элементах *LUT*.

В автомате U_2 блок *EMB* реализует систему (7) и

$$\Phi = \Phi(T, P);$$
$$Y^1 = Y^1(T, P).$$

Эта модель применима, если выполняется условие:

$$2^{G+R}(N+R) \le V_0. \qquad (10)$$

Как показал анализ библиотеки [13], эта модель применима для 82 процентов примеров.

Для уменьшения аппаратурных затрат в схеме U_2 необходимо уменьшить число элементов *LUT* в схеме блока *LUTer*. В статье предлагается подход, позволяющий решить эту задачу.

Основная идея предлагаемого метода

Конфигурация блока *EMB* определяется парой $<S_A, t_F>$. При увеличении S_A на единицу число входов t_F уменьшается в два раза. Найдем конфигурацию $<S_{A0}, t_{F0}>$ такую, что имеет место следующее соотношение:

$$t_{F0} > N + R > t_{F0}/2.$$

Пусть при этом выполняется соотношение:

$$S_{A0} > G + R.$$

Итак, конфигурация $<S_{A0}, t_{F0}>$ позволяет реализовать схему МПА на одном *EMB*. При этом имеется Δ_S свободных адресных входов, где

$$\Delta_S = S_A - (G + R). \qquad (11)$$

В статье предлагается разбить множество X на два подмножества: $X = X^1 \bigcup X^2$, где $X = X^1 \bigcap X^2 = \varnothing$. Множество X^2 имеет мощность Δ_S, а множество $X^1 - (L - \Delta_S)$. В этом случае условия $x_1 \in X^1$ подлежат преобразованию $X^1 \to P$. Переменные $x_l \in X^2$ поступа-

ют непосредственно на входы блока *EMB*. Это определяет модель U_3, показанную на рис. 4.

В автомате U_3 блок *LUTer* реализует систему функций

$$P = P\,(T', X^1). \qquad (12)$$

При этом $T' \subseteq T$, что возможно при специальном кодировании состояний. Блок *EMB* реализует систему (7) и системы функций

$$Y^1 = Y^1(T, X^2, P);$$

$$\Phi = \Phi\,(T, X^2, P).$$

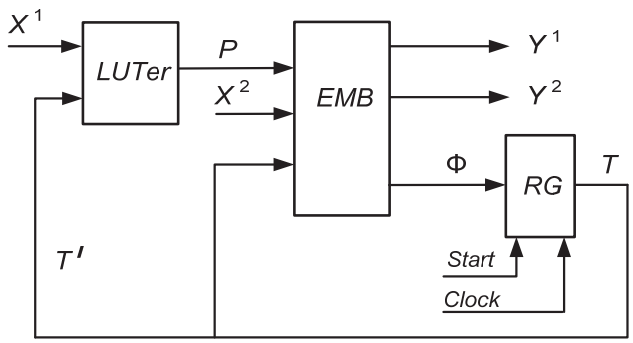


Рис. 4. Структурная схема автомата U_3

Такой подход позволяет уменьшить число литералов в системе (12) в сравнении с этим параметром для системы (9). Это приводит к уменьшению числа *LUT*-элементов в схеме *LUTer* автомата U_3 в сравнении с эквивалентным автоматом U_2. Автоматы считаются эквивалентными, если они синтезированы по одной и той же исходной граф-схеме алгоритма (ГСА).

Предлагается метод синтеза автомата U_3 по ГСА Г, включающей следующие этапы:

- формирование множества состояний A и множеств Y^1, Y^2;
- замена логических условий $x_l \in X$ переменными $p_g \in P$;
- разбиение множества логических условий;
- специальное кодирование состояний;
- формирование прямой структурной таблицы МПА;
- формирование таблиц элементов блока *LUTer*;
- формирование таблицы блока *EMB*;
- реализация схемы МПА в заданном элементном базисе.

Пример применения предложенного метода

Рассмотрим пример реализации этого метода для ГСА Γ_1 (рис. 5). Используем в качестве элементного базиса микросхему *FPGA* со следующими конфигурациями блоков *EMB*: $4K \times 1$, $2K \times 2$, $1K \times 4$, 512×8 и 256×16 (битов). Пусть *LUT*-элементы имеют число входов $S = 3$.

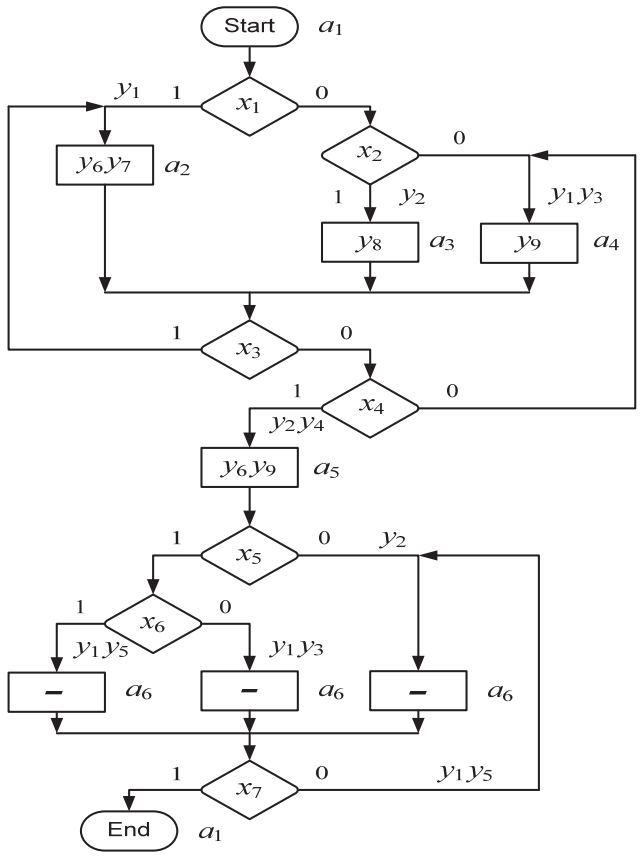


Рис. 5. Исходная ГСА Γ_1

На дугах ГСА Γ_1 показаны выходные переменные $y_n \in Y^1$, в операторных вершинах – переменные $y_n \in Y^2$. Анализ ГСА Γ_1 дает множества $Y^1 = \{y_1,\ldots, y_5,\}$, и $Y^2 = \{y_6,\ldots, y_9\}$. Таким образом $N_1 = 5$, $N_2 = 4$ и $N = 9$.

Состояния $a_m \in A$ – состояния автомата Мура, т.е. каждая операторная вершина отмечается уникальной отметкой [1]. Авторы предлагают отмечать одинаковыми состояниями вершины, если: их выходы связаны с входом одной и той же вершины ГСА Г и в этих вершинах нет переменных $y_n \in Y^2$.

Такой подход позволяет отметить состоянием a_6 три вершины (рис. 5). Это приводит к

уменьшению числа состояний и числа переходов из них. Для ГСА Γ_1 имеем $A = \{a_1,\ldots, a_6\}$, что дает $M = 6$, $R = 3$, $T = \{T_1, T_2, T_3\}$ и $\Phi = \{D_1, D_2, D_3\}$.

Обозначим символом $U_i(\Gamma_j)$ тот факт, что автомат U_i синтезируется по ГСА Γ_j. Рассмотрим возможность реализации автомата $U_1(\Gamma_1)$ для данной микросхемы *FPGA*. Очевидно, $V_0 = 4096$ (это следует, например, из конфигурации $4K \times 1$). Для ГСА Γ_1 имеем $L = 7$, $R = 3$ и $N = 9$. Проверим условие (8):

$$2^{10} \times (9+3) = 12K > V_0 = 4K .$$

Видимо, для реализации систем (5) – (7) необходимо три блока *EMB*.

Таким образом, целесообразно проверить автомат $U_2(\Gamma_1)$. Обозначим символом $X(a_m)$ множество логических условий, определяющих переходы из состояния $a_m \in A$. Для ГСА Γ_1 имеем: $X(a_1) = \{x_1, x_2\}$, $X(a_2) = X(a_3) = X(a_4) = \{x_3, x_4\}$, $X(a_5) = \{x_5, x_6\}$, $X(a_6) = \{x_7\}$. Следовательно, число переменных G определяется следующим образом:

$$G = \max(L_1,\ldots, L_M),$$

где $L_m = |X(a_m)|$, $m \in \{1, \ldots, M\}$.

В рассматриваемом примере имеем $G = 2$, что определяет множество $P = \{p_1, p_2\}$. Построим таблицу замены логических условий (табл. 2). В ней переменная $x_l \in X$, заменяемая в состоянии $a_m \in A$ переменной $a_g \in P$, записывается на пересечении строки p_g и столбца a_m.

Таблица 2. Таблица замены логических условий

p_g \ a_m	a_1	a_2	a_3	a_4	a_5	a_6
p_1	x_1	x_3	x_3	x_3	x_5	x_7
p_2	x_2	x_4	x_4	x_4	x_6	–

Условие (10) справедливо для автомата $U_2(\Gamma_1)$. Проверим возможность использования модели $U_3(\Gamma_1)$. Поскольку $R + N = 12$, необходимо выбрать конфигурацию 256×16. Это дает $S_A = 8$. Так как $G + R = 5$, из (11) можно получить $\Delta_S = 3$. Таким образом, три переменные $x_l \in X$ подавать непосредственно на адресные входы *EMB*, т.е. модель $U_3(\Gamma_1)$ может быть использована.

Представим множество X в виде объединения множеств $X^1 = \{x_1, x_2, x_6, x_7\}$, и $X^2 = \{x_3, x_4, x_5\}$. Теперь таблица замены логических условий имеет следующий вид (табл. 3)

Таблица 3. Таблица замены логических условий автомата $U_3(\Gamma_1)$

p_g \ a_m	a_1	a_2	a_3	a_4	a_5	a_6
p_1	x_1	x_3	x_3	x_3	x_5	x_7
p_2	x_2	x_4	x_4	x_4	x_6	–

Из табл. 3 следует, что $p_2 = x_2$. Таким образом, имеем множества $P = \{p_1\}$, $X^1 = \{x_1, x_6, x_7\}$, $X^2 = \{x_2, x_3, x_4, x_5\}$.

Специальное кодирование состояний заключается в выборе таких кодов $K(a_m)$, чтобы функции (12) имели минимальное число литералов. Закодируем состояния так, как это показано в карте Карно (рис. 6). В этой карте знак «*» означает, что данный код может быть использован для минимизации функций (12).

T_1 \ T_2T_3	00	01	11	10
0	a_1	a_2 *	*	a_5
1	a_6	a_3 *	*	a_4 *

Рис. 6. Коды состояний автомата $U_3(\Gamma_1)$

Из рис. 6 можно получить уравнение:

$$P_1 = \overline{T}_1\overline{T}_2 x_1 \vee T_1 x_7 \vee T_2 x_6 . \qquad (13)$$

При использовании кодов из рис. 6 и табл. 2 можно получить следующие уравнения:

$$P_1 = \overline{T}_1\overline{T}_2\overline{T}_3 x_1 \vee \overline{T}_2 T_3 x_3 \vee T_1 T_2 x_3 \vee \overline{T}_1 T_2 x_5 , \qquad (14)$$

$$P_2 = \overline{T}_2\overline{T}_3 x_2 \vee \overline{T}_2 T_3 x_4 \vee T_1 x_4 \vee \overline{T}_1 T_2 x_6 . \qquad (15)$$

В нашем примере *LUT*-элементы имеют $S = 3$. Для реализации (13) требуется четыре элемента и схемы имеют два уровня. Для реализации (14) – (15) требуется 13 элементов и схема имеет четыре уровня. Таким образом, применение предложенного подхода позволяет в три раза уменьшить аппаратурные затраты и в два раза увеличить быстродействие блока *LUTer*.

Фрагмент прямой структурной таблицы (ПСТ) автомата $U_3(\Gamma_1)$ приведен в табл. 4.

Таблица 4. Фрагмент ПСТ автомата $U_3(\Gamma_1)$

a_m	$K(a_m)$	a_S	$K(a_S)$	X_h	P_h	Φ_h	Y_h	h
$a_1(-)$	000	a_2	001	1	P_1	D_3	y_4	1
		a_3	101	x_2	$\overline{P_1}$	$D_1\,D_3$	y_2	2
		a_4	110	$\overline{x}_2$	$\overline{P_1}$	$D_1\,D_2$	$y_1\,y_3$	3
$a_2(y_6\,y_7)$	001	a_2	001	x_3	1	D_3	–	4
		a_5	010	$\overline{x}_3 x_4$	1	D_2	$y_2\,y_4$	5
		a_4	110	$\overline{x}_3\overline{x}_4$	1	$D_1\,D_2$	$y_1\,y_3$	6

ПСТ автомата $U_3(\Gamma_1)$ имеет следующие столбцы: a_m – текущее 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, P_h – входной сигнал, определяющий переход из a_m, в a_S; Φ_h – функции возбуждения памяти, равные единице для ее переключения из $K(a_m)$ в $K(a_S)$; Y_h – выходные переменные, формируемые при переходе из a_m, в a_S; h – номер перехода. Кроме того, в столбце a_m указывается формируемый в нем выходной сигнал.

В табл. 4 показаны переходы для состояний a_1 *и* a_2. Отметим, что ПСТ автомата $U_3(\Gamma_1)$ имеет $H = 21$ строку. Если переход не зависит от переменных $x_l \in X$ ($p_g \in P$), то в соответствующем столбце записывается единица. Подобная таблица – основа для формирования содержимого блока *EMB*.

Блок *LUTer* задается таблицами, каждая из которых соответствует одному элементу *LUT*. Например, конъюнкция $\overline{T_1}\,\overline{T_2}x_1$ уравнения (13) соответствует элементу *LUT*1. Этот элемент задается табл. 5.

Таблица 5. Таблица элемента *LUT*1

$T_1\,T_2\,x_1$	p_1	q	$T_1\,T_2\,x_1$	p_1	q
0 0 0	0	1	1 0 0	0	5
0 0 1	1	2	1 0 1	0	6
0 1 0	0	3	1 1 0	0	7
0 1 1	0	4	1 1 1	0	8

Терм $\overline{T_1}\,\overline{T_2}x_1$ соответствует ячейке с номером $q = 2$. Во второй строке переменная $p_1 = 1$, в остальных строках $p_1 = 0$. Аналогичным образом строятся таблицы для программирования каждого *LUT*-элемента.

Таблица блока *EMB* строится на основе ПСТ. Она имеет столбцы $K(a_m)$, X^2, P (адрес ячейки), Ф, Y^1, Y^2 (содержимое ячейки), q. Таблица блока *EMB* имеет H строк, где:

$$H = 2^{R+G+\Delta_S}.$$

В рассматриваемом примере $H = 256$. Переходы из каждого состояния задаются при помощи $H(a_m)$ строк, где:

$$H(a_m) = 2^{G+\Delta_S}.$$

В рассматриваемом примере $H(a_m) = 32$.

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

Для рассматриваемого примера адреса ячеек соответствуют вектору $<T_1\ T_2\ T_3\ x_2\ x_3\ x_4\ x_5\ p_1>$, а их содержимое – вектору $<D_1\,D_2\,D_3\,y_1\,y_2\,y_3\,y_4\,y_5\,y_6\,y_7\,y_8\,y_9>$. Например, четвертая строка табл. 4 определяется векторами <001∗1∗∗∗> и <001010101100>, где символ «∗» показывает, что значение переменной несущественно. Таким образом, четвертая строка ПСТ соответствует 16 ячейкам *EMB*. Мы не показываем таблицу *EMB* в силу ее громоздкости.

Заключение. Предложенный метод позволяет реализовать схему совмещенного МПА с использованием минимального числа блоков *EMB* и элементов *LUT*. Первое достигается путем замены логических условий, второе – путем замены только части множества логических условий. При этом, основываясь на результатах [7–12], можно утверждать, что схема МПА занимает минимально возможную площадь кристалла и потребляет минимальную энергию.

Анализ библиотеки [13] показывает, что для 82 процентов примеров схема реализуется с использованием только одного *EMB*. Для остальных 18 процентов примеров требуется от двух до трех блоков *EMB*. Для оптимизации этих примеров мы предлагаем использовать методы из статьи [14].

Для уменьшения числа *LUT*-элементов необходимо соответствующим образом разбить множество входных переменных. Это будет направлением наших дальнейших исследова-

ний. В частности предлагается адаптировать методы [15] 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. – N.Y.: 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 использованием микросхем программируемой логики. – СПб: БХВ. – Петербург, 2002. – 608 с.
6. *Synthesis* and Optimization of FPGA-based Systems / V. Sklyarov, I. Skliarova, A. Barkalov et al. – Berlin: Springer, 2014. – 432p.
7. *Cong J.*, *Yan K.* Synthesis for FPGAs with Embedded Memory Blocks // Proc. of the 2000 ACM/SIGDA 8th Int. Symp. on FPGAs. – 2000. – P. 75–82.
8. *ROM-Based* Finite State Machine Implementation in Low Cost FPGAs / L. Garcia–Vargas, R. Senhadji–Navarro, M. Civit–Balcells et al. // IEEE Int. Simp. on Industrial Electronics, Vigo. – 2007. – P. 2342–2347.
9. *Nowicka M.*, *Łuba T.*, *Rawski V.* FPGA-based decomposition of boolean functions: algorithms and implementations // Advanced Comp. Syst. – 1999. – P. 502–509.
10. *Rawski M.*, *Selvaraj H.*, *Łuba T.* An application of functional decomposition in ROM-based FSM implementation in FPGA devices // J. of System Architecture. – 2005. – **51**(6–7) – P. 424–434.
11. *Logic* Synthesis Method of Digital Circuits Designed for Implementation with Embedded Memory Blocks on FPGAs / M. Rawski, P. Tomaszewicz, G. Borowski et al. // Design of Digital Systems and Devises. LNEE 70. – Berlin: Springer, 2005. – P. 121–144.
12. *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.
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 // 12th IFAC Conf. on programmable devices and embedded systems. – 2013. – P. 215–220.

Поступила 13.05.2016

Тел. для справок: +38 044 526-2504, 406-7829 (Киев)
E-mail: *A.Barkalov@iie.uz.zgora.pl*, *L.Titarenko@iie.uz.zgora.pl*, *yaviz@ukr.net*, *matv@online.ua*

UDC 004.274

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

Synthesis of Combined Finite State Machine with FPGAs

Keywords: combined FSM, FPGA, LUT, EMB, synthesis, graph-scheme of algorithm.

A method for synthesis of combined finite state machine (CFSM) with FPGA is proposed. An analysis of CFSM's peculiarities is given. The main feature of CFSM is an existence of two types of the output signals. Mealy outputs depend on the both inputs and states. Moore outputs depend only on the states. The known methods of CFSM design and conditions for their application and the method of the logical conditions replacement is thoroughly analyzed.

It allows using embedded memory locks (EMB) for implementing some part of CFSM circuit. It is shown that the situations are possible when not all address inputs of an EMB are used. The suggested method is based on using these free address inputs. It is proposed to connect a part of logical conditions with unused address inputs of EMB blocks. It allows diminishing for the number of look-up table (LUT) elements in the circuit of logical conditions replacement in comparison with known methods of CFSM design.

It is proposed to replace some part of the logical conditions set by additional variables. A design method based on such partial replacement is proposed. The method allows obtaining a CFSM circuit with the minimum number of table elements LUTs and memory blocks EMBs. Some additional optimization are possible for the replacement block of the logical conditions due to a special state assignment.

The main idea of the special state assignment is reduced the assignment neighbor codes for states with transitions depending on the same logical conditions. It allows diminishing the number of the literals in functions implemented of the block of the logical conditions replacement. An example of the studied method application is shown.

The proposed method allows obtaining a circuit required minimum chip space and consuming minimum power in comparison with the known design methods. The conducted researches are based on some library of standard GSAs. The investigations show that for the majority of standard GSAs the proposed method produces the circuits with a single EMB.

●