УДК 004.274

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

Оптимальное кодирование состояний в совмещенном автомате

Предложен метод синтеза совмещенного микропрограммного автомата, ориентированный на СБИС типа *FPGA*. Метод позволяет получить схему с минимальным числом элементов *LUT* и блоков *EMB*. Оптимизация достигается путем представления кодов псевдоэквивалентных состояний обобщенными интервалами кодирующего пространства. Приведен пример синтеза с использованием предложенного метода.

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

Запропоновано метод синтезу суміщеного мікропрограмного автомата, орієнтований на НВІС типу *FPGA*. Метод дозволяє отримати схему з мінімальним числом елементів *LUT* та блоків *EMB*. Оптимізація досягається шляхом представлення кодів псевдоеквівалентних станів узагальненими інтервалами кодуючого простору. Наведено приклад синтезу з використанням запропонованого методу.

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

Введение. Устройство управления – одна из важнейших частей любой цифровой системы [1]. Для синтеза схемы устройства управления часто используется модель совмещенного микропрограммного автомата (СМПА) [2]. В СМПА существуют выходные сигналы двух типов. Выходные сигналы автомата Мили зависят от входных переменных и состояний СМПА. Выходные сигналы автомата Мура – только от состояний [2, 3]. Отметим, что в литературе практически отсутствуют формальные методы синтеза СМПА.

В настоящее время для реализации цифровых систем широко используются СБИС типа *FPGA* (*field-programmable logic arrays*) [4, 5]. Для реализации схем СМПА 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 сохранении 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 выражении (1) функция $\lceil a \rceil$ определяет целое число, большее a, если a – дробное и равное a, если a – целое.

Одна из существенных задач, связанных с синтезом схем автоматов, это уменьшение площади кристалла, занимаемого схемой [6]. В случае *FPGA* решение этой задачи приводит к уменьшению времени задержки распространения сигналов и потребляемой мощности. Кроме того, необходимо уменьшать число межсоединений в схеме, что приводит к уменьшению паразитных емкостей [7]. Один из путей решения этой задачи – замена элементов *LUT* встроенными блоками памяти *EMB* [8–12]. При этом важно минимизировать число блоков *EMB* в схеме. В статье предлагается один из подходов к решению этой задачи, позволяющий уменьшить число адресных входов блока *EMB*. Метод основан на использовании псевдоэквивалентных состояний (ПЭС) автомата Мура [13].

Особенности совмещенного микропрограммного автомата и базиса *FPGA*

Для оптимизации характеристик схемы СМПА необходимо учесть особенности как модели автомата, так и элементного базиса. Рассмотрим эти особенности.

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: $A = \{a_1, \ldots, a_M\}$ – множество внутренних состояний; $X = \{x_1, \ldots, x_L\}$ – множество вход-

ных 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. Для этих множеств справедливы следующие отношения: $Y = Y^1 \bigcup Y^2$; $Y^1 \bigcap Y^2 = \varnothing$. Введем следующие обозначения: $\left|Y^1\right| = N_1$, $\left|Y^2\right| = N_2$ 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:

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

Функции λ_1 и λ_2 имеют следующий вид:

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

$$y_n = \lambda\left(a_m\right). \quad (4)$$

Для реальных устройств переменные $x_1 \in X$ и $y_n \in Y$ – физические объекты, принимающие значения «0» и «1». В то же время состояния $a_m \in A$ есть абстрактными объектами. Для синтеза схемы микропрограммного автомата (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, триггеры 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 кодирования состояний 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). Эти функции определяются, соответственно, следующими системами булевых функций:

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

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

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

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

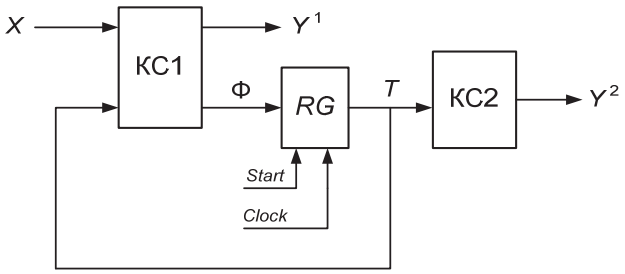


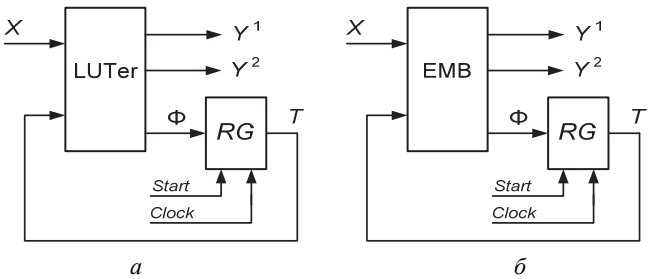
Рис. 1. Структурная схема совмещенного МПА

В этой схеме блок КС1 генерирует функции (5) – (6), блок КС2 – функции (7). Сигнал *Start* устанавливает в *RG* код начального состояния $a_1 \in A$. Импульс *Clock* вызывает переключение *RG*, соответствующее функции перехода (2).

Особенностью *FPGA* – наличие реконфигурируемых блоков *EMB*. Типичными конфигурациями $V \times t_F$ есть следующие: $32K \times 1$, $16K \times 2$, $8K \times 4$, $4K \times 8$, $2K \times 16$, $1K \times 32$ и 512×64 (битов) [4, 5], что определяет следующие пары вида S_A, t_F: 15, 1, 14, 2, 13, 4, 12, 8, 11, 16, 10, 32 и 9, 64. Эту особенность можно учесть следующим образом: параметры *EMB* можно подбирать так, чтобы уменьшить число блоков в схеме СМПА.

Реализация совмещенного микропрограммного автомата 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*.



а *б*

Рис. 2. Тривиальные схемы совмещенных МПА

Отметим, что в модели U_1 триггеры регистра *RG* распределены между элементами *LUT*. Поэтому регистр не существует как отдельный

блок. Если *EMB* есть синхронным, то сигналы *Start* и *Clock* поступают на соответствующие входы *EMB*. Однако структуры U_1 и U_2 имеют более общий характер. Как правило, использование модели U_1 приводит к схемам, содержащим большое число уровней и межсоединений [6]. Использование модели U_2 приводит к схемам с наименьшей площадью. Однако эта модель может использоваться только при выполнении условия:

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

При нарушении условия (8) системы (5) – (7) реализуются на нескольких блоках *EMB*. Для уменьшения числа блоков *EMB* может быть использован метод замены логических условий [7–12].Однако это приводит к появлению дополнительной подсхемы, реализуемой на элементах *LUT*. В свою очередь это ухудшает динамические характеристики схемы СМПА.

В статье предлагается использовать оптимальное кодирование ПЭС [13]. Такой подход позволяет уменьшить число адресных входов *EMB* без использования метода замены логических условий.

Основная идея предложенного метода

Пусть СМПА задан граф-схемой алгоритма (ГСА) Γ_1 (рис. 3).

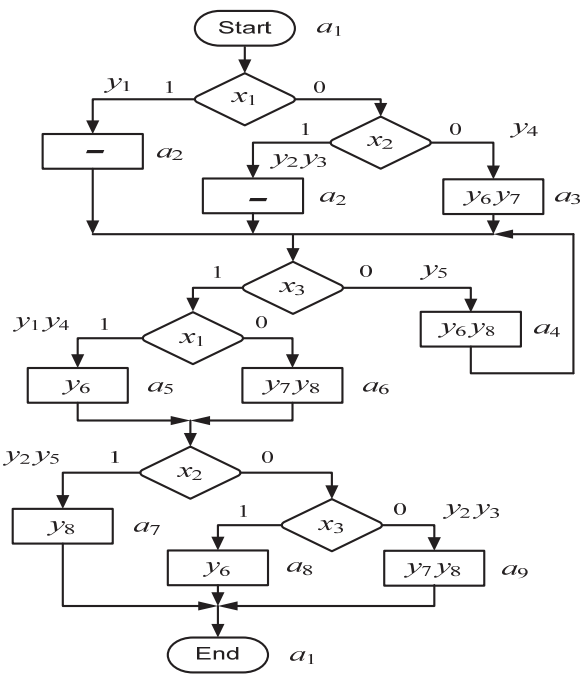


Рис. 3. Отмеченная ГСА Γ_1

Пока не будем обсуждать метод отметки состояний. Состояния a_s, $a_m \in A$ называются псевдоэквивалентными, если они соответствуют операторным вершинам ГСА, выходы которых связаны с входом одной и той же вершины. Например, состояния a_2, a_3 и a_4 (рис. 3) отвечают этому определению.

Найдем разбиение $\Pi_A = \{B_1,...,B_I\}$ множества A на классы ПЭС. Закодируем состояния $a_m \in B_i$ так, чтобы они входили в один обобщенный интервал пространства кодирования. Рассмотрим этот интервал как код класса B_i.

Для ГСА Γ_1 имеем разбиение Π_A с классами $B_1 = \{a_1\}$, $B_2 = \{a_2, a_3, a_4\}$, $B_3 = \{a_5, a_6\}$ и $B_4 = \{a_7, a_8, a_9\}$. При числе состояний $M = 9$ имеем $R = 4$. Этот параметр определяет множества $T = \{T_1,... ...,T_4\}$ и $\Phi = \{D_1,..., D_3\}$. Закодируем состояния $a_m \in A$ так, как показано на карте Карно (рис. 4).

$T_3\,T_4$ \ $T_1\,T_2$	00	01	11	10
00	a_1	a_2	a_7	a_5
01	*	a_3	a_8	a_6
11	*	*	*	*
10	*	a_4	a_9	*

Рис. 4. Коды состояний СМПА для ГСА Γ_1

Из карты Карно имеем коды $K(B_i)$ классов ПЭС: $K(B_1) = 00{**}$, $K(B_2) = 01{**}$, $K(B_3) = 10{**}$ и $K(B_4) = 11{**}$. Таким образом, классы $B_i \in \Pi_A$ определяются только частью множества внутренних переменных, образующих множество $T = \{T_1, T_2\}$. Пусть выполняется условие

$$R_1 < R , \qquad (9)$$

где $R_1 = |T'|$. В этом случае мы предлагаем структурную схему СМПА U_3 (рис. 5).

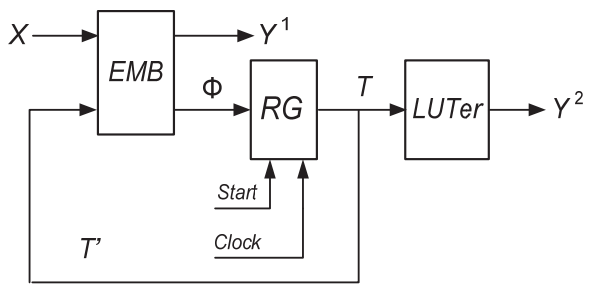


Рис. 5. Структурная схема СМПА U_3

В автомате U_3 блок *EMB* реализует системы функций

$$\Phi = \Phi(T', X);$$
$$Y^1 = Y^1(T', X).$$

Блок *LUTer* реализует систему (7). Наличие этого блока – недостаток в сравнении с автоматом U_2. Однако этот блок значительно проще, чем блок замены логических условий.

В статье предлагается метод синтеза СМПА U_3 по ГСА Г. Метод включает следующие шаги:

1. Формирование множеств A, Π_A, Y^1 и Y^2.
2. Оптимальное кодирование состояний $a_m \in A$.
3. Формирование прямой структурной таблицы (ПСТ) СМПА.
4. Формирование таблицы блока *EMB*.
5. Формирование таблицы блока *LUTer*.
6. Реализация схемы СМПА в заданном элементном базисе.

Пример применения предложенного метода

Рассмотрим пример синтеза автомата U_3 по ГСА Γ_j. Отметим, что этапы 1 и 2 уже выполнены. Построим прямую структурную таблицу автомата, включающую 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 $a_s \in A$; X_h – входной сигнал, определяющий переход из $a_m \in B_i$ в $a_s \in A$; Y^1_h – выходные сигналы автомата Мили, формируемые на переходе a_m, a_S ; Φ_h – функции возбуждения памяти, равные 1 для переключения регистра *RG* из $K(a_m)$ в $K(a_S)$; h – номер строки ПСТ ($h = \overline{1, H}$). В рассматриваемом примере $H = 9$ (табл. 1).

Таблица 1. Прямая структурная таблица СМПА U_3

B_i	$K(B_i)$	a_S	$K(a_S)$	X_h	$Y^1{}_h$	Φ_h	h
B_1	00**	a_2	0100	x_1	y_1	D_2	1
		a_2	0100	$x_1 x_2$	$y_2 y_3$	D_2	2
		a_3	0101	$x_1 x_2$	y_4	$D_2 D_4$	3
B_2	01**	a_5	1000	$x_3 x_1$	$y_1 y_4$	D_1	4
		a_6	1001	$x_3 x_1$	–	$D_1 D_4$	5
		a_4	0110	x_3	y_5	$D_2 D_3$	6
B_3	10**	a_7	1100	x_2	$y_2 y_5$	$D_1 D_3$	7
		a_8	1101	$x_2 x_3$	y_5	$D_1 D_2 D_4$	8
		a_9	1110	$x_2 x_3$	$y_2 y_3$	$D_1 D_2 D_3$	9

Переходы из состояний $a_7, a_8, a_9 \in B_4$ не включены в табл. 1, так как они не связаны с формированием функций $D_r \in \Phi$ и $y_n \in Y^1$. Эти переходы происходят автоматически по сигналу синхронизации *Clock*.

Таблица блока *EMB* строится на основе ПСТ. Она включает столбцы: $K(B_i)$, X (адрес ячейки памяти), Y^1, Ф (содержимое ячейки памяти), q – номер ячейки памяти, начиная с единицы ($q = \overline{1, Q}$). В общем случае выполняется условие: $Q = 2^{R1+L}$.

В нашем примере $Q = 32$. Переходы из состояний $a_m \in B_i$ задаются при помощи $H(B_i)$ строк, где $H(B_i) = 2^L$.

В нашем примере $H(B_i) = 8$.

В табл. 2 представлен фрагмент таблицы блока *EMB* для нашего примера.

Этот фрагмент соответствует переходам из состояния $a_1 \in B_1$. Столбец h добавлен, чтобы показать соответствие между табл. 1 и 2.

Таблица 2. Таблица блока *EMB* автомата U_3

$K(B_i)$	X	Y^1	Ф	q	h
$T_1 T_2$	$x_1 x_2 x_3$	$y_1 y_2 y_3 y_4 y_5$	$D_1 D_2 D_3 D_4$		
0 0	0 0 0	0 0 0 1 0	0 1 0 1	1	3
0 0	0 0 1	0 0 0 1 0	0 1 0 1	2	3
0 0	0 1 0	0 1 1 0 0	0 1 0 0	3	2
0 0	0 1 1	0 1 1 0 0	0 1 0 0	4	2
0 0	1 0 0	1 0 0 0 0	0 1 0 0	5	1
0 0	1 0 1	1 0 0 0 0	0 1 0 0	6	1
0 0	1 1 0	1 0 0 0 0	0 1 0 0	7	1
0 0	1 1 1	1 0 0 0 0	0 1 0 0	8	1

Таблица блока *LUTer* (табл. 3) объединяет N_2 таблицы для функций $y_n \in Y^2$. Эта таблица имеет столбцы $K(a_m)$, Y^2, q, где $q = \overline{1, Q_1}$. Параметр Q_1 определяется, как $Q_1 = 2^R$.

В рассматриваемом примере $Q_1 = 16$.

Таблица 3. Таблица блока *LUTer* автомата U_3

$K(a_m)$	Y^2	q	m	$K(a_m)$	Y^2	q	m
$T_1 T_2 T_3 T_4$	$y_6 y_7 y_8$			$T_1 T_2 T_3 T_3$	$y_6 y_7 y_8$		
0 0 0 0	0 0 0	1	1	1 0 0 0	0 0 1	9	5
0 0 0 1	0 0 0	2	–	1 0 0 1	0 0 0	10	6
0 0 1 0	0 0 0	3	–	1 0 1 0	0 0 0	11	–
0 0 1 1	0 0 0	4	–	1 0 1 1	0 0 0	12	–
0 1 0 0	0 0 0	5	2	1 1 0 0	0 0 0	13	7
0 1 0 1	0 0 0	6	3	1 1 0 1	0 0 0	14	8
0 1 1 0	0 0 0	7	4	1 1 1 0	0 0 0	15	9
0 1 1 1	0 0 0	8	–	1 1 1 1	0 0 0	16	–

Столбец m добавлен в табл. 3, чтобы показать соответствие между строками табл. 3 и состояниями $a_m \in A$.

Последний этап предлагаемого метода связан с использованием стандартных пакетов для реализации схем в базисе *FPGA*. В статье этот этап авторы не рассматривают.

Анализ предложенного метода

При выполнении условия (9) уменьшаются требования к числу адресных входов *EMB*. При этом блок *EMB* реализуется в виде одного блока памяти при выполнении условия

$$2^{R+L1} \cdot (N_1 + R) \le V_0 .$$

Анализ библиотеки [14] показал, что для 92 процентов автоматов выполняется условие (9).

Пусть S_L – число входов элементов *LUT*. Автомат U_3 целесообразно использовать при выполнении условия

$$R \le S_L . \qquad (10)$$

При выполнении условия (10) для реализации блока *LUTer* требуется не более N_2 элементов *LUT*. Схема блока *LUTer* будет иметь только один уровень. Если условие (10) не выполняется, то необходимо сравнить предлагаемый метод с методами, основанными на замене логических условий.

Если условие (10) нарушается, то упрощение блока *LUTer* возможно путем соответствующего кодирования состояний. Например, закодируем состояния $a_m \in A$ так, как показано на рис. 6.

$T_3\,T_4$ \ $T_1\,T_2$	00	01	11	10
00	a_1	a_2	a_8	a_5
01	*	a_4	a_7	a_6
11	*	*	a_9	*
10	*	a_3	*	*

Рис. 6. Уточненные коды состояний автомата U_3

Пусть A_m – конъюнкция переменных $T_r \in T$, соответствующая коду $K(a_m)$ состояния $a_m \in A$. Из ГСА Γ_1 можно получить систему (7), которая в данном случае имеет следующий вид:

$$y_6 = A_3 \vee A_4 \vee A_5 \vee A_8 ;$$
$$y_7 = A_3 \vee A_6 \vee A_9 ;$$
$$y_8 = A_4 \vee A_6 \vee A_7 \vee A_9$$

Используя коды из рис. 6, и несущественные кодовые комбинации (обозначены символом *), получим следующую систему уравнений:

$$y_6 = \overline{T}_1 T_4 \vee \overline{T}_1 T_3 \vee T_1 T_3$$
$$y_7 = T_3 \vee \overline{T}_2 T_4 \qquad (11)$$
$$y_8 = T_4$$

Анализ системы (11) показывает, что для ее реализации достаточно двух элементов *LUT*, имеющих $S_L = 3$ входа. При этом функция y_8 является выходом T_4 регистра *RG*.

Отметим, что состояния $a_m \in B_i$ могут переставляться только в пределах своего обобщенного интервала. Это позволяет получить минимальную величину параметра R_1.

Заключение. Предложенный в статье метод позволяет уменьшить число блоков *EMB* и элементов *LUT* в схеме СМПА, что достигается путем соответствующего кодирования состояний. При этом первичным есть оптимальное кодирование состояний, позволяющее уменьшить число адресных входов *EMB*. Оптимизация блока *LUTer* необходима при нарушении условия (10). Коды состояний $a_m \in B_i$ можно изменять без изменения кода класса $K(B_i)$. Полученные таким образом коды состояний мы назвали уточненными.

Анализ библиотеки [14–15] показал, что данный метод позволяет получить схемы с одним блоком *EMB* для 82 процентов всех примеров. При этом условие (10) нарушается в 46 процентах всех примеров. Перестановка кодов состояния позволила уменьшить число элементов *LUT* для 60 процентов этих примеров.

Дальнейшее направление наших исследований связано с адаптацией подхода [16] к особенностям совмещенного автомата. В основе метода из [16] находится расщепление множе-

ства логических условий. Это позволяет уменьшить число элементов *LUT* в блоке замены логических условий в сравнении с известными методами замены[11, 12].

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. *Соловьев В.В.* Проектирование цифровых схем на основе программируемых логических интегральных схем. – М.: ТЕЛЕКОМ, 2001. – 636 с.
4. *Skliyarova 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. – 432 p.
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 A. 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 Computer Systems, 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. – N 51(**6–7**). – P. 424–434.
11. *Logic* Synthesis Method of Digital Circuits Designed for Implementation with Embedded Memory Blocks on FPGAs, Design of Digital Systems and Devises. LNEE 70 / M. Rawski, P. Tomaszewicz, G. Borowski et al. – Berlin: Springer, 2011. – 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. *Реализация* схемы совмещенного автомата в базисе *FPGA* / А.А. Баркалов, Я.Е. Визор, А.В. Матвиенко и др. // Комп'ютерні засоби, мережі та системи: Зб. наукових праць. – Інститут кібернетики ім. В.М. Глушкова НАН України. – К., 2016. – № 15. – С. 32–40.
16. *Barkalov A.*, *Titarenko L.*, *Kolopenczyk M.* EMB-based design of Mealy FSM. – Proc. of 12th IFAC Conf. on programmable devices and embedded syst. – 2013. – P. 215–220.

Поступила 09.04.2015

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

UDC 004.274

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

An Optimal State Assignment for the Combined Automation

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

An offer in-process method allows to decrease the number of EMB blocks and LUT elements in the chart of the combined microprogram automat. It is arrived at due to the corresponding encryption of the states. Thus, the optimal encryption of the states is primery, allowing to decrease the number of address entrances of EMB. The optimization of LUT block is needed at violation of corresponding terms. The codes of the states can be changed without the change of code of class of K (Bi). The codes of the states have got the specified features.

The analysis of the used library showed that this method allows to get charts with one block of EMB for 82% of all examples. Thus, corresponding condition is violated in 46% of all examples. Transposition of the codes of the state allows to decrease the number of LUT elements for 60% of these examples.

Further direction of researches is related to adaptation of the new features of the combined automat. It allows to decrease the number of LUT elements in the block of replacement of the logical terms as compared to the well-known replacement methods.

●