УДК 681.3

А.А. Баркалов, Л.А. Титаренко, А.С. Лаврик

Модели устройства управления с общей памятью, ориентированные на ПЛИС

Предложены метод уменьшения аппаратурных затрат в логической схеме композиционного микропрограммного устройства управления, реализуемого в базисе *CPLD* и пример его применения.

A method of the hardware amount reduction in a logical scheme of a compositional microprogram control unit implemented in the CPLD basis and an example of its application are suggested.

Запропоновано метод зменшення апаратурних витрат у логічній схемі композиційного мікропрограмного пристрою керування при реалізації в базисі *CPLD* та приклад його застосування.

Введение. Для реализации схем устройств управления (УУ) часто используется модель автомата Мура [1]. Одна из важных задач, возникающих при реализации схем УУ, – уменьшение аппаратурных затрат. Это может быть площадь кристалла, занимаемого схемой, или число макроячеек, используемое для реализации схемы [2]. Решение этой задачи базируется на использовании особенностей реализуемого алгоритма управления и элементного базиса [3]. Так, для реализации линейных алгоритмов управления целесообразно использовать модель композиционного микропрограммного устройства управления (КМУУ) [4]. По своей сути, КМУУ есть автомат Мура, для хранения состояний которого используется счетчик. Один из популярных базисов, используемых для реализации схем УУ, – это программируемые логические интегральные схемы (ПЛИС) типа *CPLD* (*Complex Programmable Logic Devices*) [5, 6]. Подобные ПЛИС основаны на макроячейках программируемой матричной логики (ПМЛ). Особенность макроячеек ПМЛ – достаточно большое число входов (порядка нескольких десятков), ограниченное число промежуточных термов (не более десяти) и один выход [7–8]. Эти особенности вызывают необходимость разработки методов синтеза схем УУ, в которых минимизируется число термов в каждой из реализуемых функций. Одним из путей решения этой задачи является использование нескольких источников кодов состояний автомата Мура [9–10]. В КМУУ аналогом состояния есть операторная линейная цепь (ОЛЦ) [4]. В статье предлагается обобщенная модель КМУУ с общей памятью, основанная на использовании нескольких источников кодов ОЛЦ. При этом алгоритм управления представляется в виде граф-схемы алгоритма (ГСА) [11]. Отметим, что предложенный метод может использоваться и для других моделей КМУУ.

Базовая модель КМУУ с общей памятью

Пусть ГСА Г представлена множествами вершин B и дуг E, соединяющих эти вершины. При этом $B = \{b_0, b_E\} \cup B_1 \cup B_2$, где b_0 – начальная вершина ГСА, b_E – конечная вершина ГСА, B_1 – множество операторных вершин, где $|B_1| = M$, B_2 – множество условных вершин. В вершинах $b_q \in B_1$ записаны наборы микроопераций $Y(b_q) \subseteq Y$, где $Y = \{y_1, \ldots, y_N\}$ – множество микроопераций. В вершинах $b_q \in B_2$ записаны элементы множества логических условий $X = \{x_1, \ldots, x_L\}$. Введем ряд определений [4–5], необходимых для дальнейшего изложения.

Определение 1. Операторной линейной цепью ГСА Г называется конечная последовательность операторных вершин $\alpha_g = \left\langle b_{g1}, \ldots, b_{gF_g} \right\rangle$, такая, что для любой пары соседних компонент кортежа a_g существует дуга $\langle b_{g1}, b_{gi+1} \rangle \in E$, где i – номер компоненты кортежа α_g $\left(i = \overline{1, F_g - 1}\right)$, G – число ОЛЦ в ГСА Г.

Определение 2. Операторная вершина $b_q \in D^g$, где $D^g \subseteq B_1$ – множество операторных вершин, входящих в ОЛЦ a_g, называется входом ОЛЦ a_g, если существует дуга $\langle b_t, b_q \rangle \in E$, где $b_q \notin D^g$.

Определение 3. Операторная вершина $b_q \in D^g$ называется выходом ОЛЦ a_g, если существует дуга $\left\langle b_t, b_q \right\rangle \in E$, где $b_t \notin D^g$.

Определение 4. Операторные линейные цепи α_i, α_j называются псевдоэквивалентными ОЛЦ, если существуют дуги $\langle b_i, b_t \rangle, \langle b_j, b_t \rangle \in E$, где b_i, b_j – соответственно выходы ОЛЦ α_i, α_j.

Каждая ОЛЦ α_g имеет произвольное число входов, образующих множество $I(\alpha_g)=\{I_g^1, I_g^2, ..\}$. Любая ОЛЦ a_g имеет только один выход, обозначаемый символом O_g. Пусть для ГСА Г найдено разбиение $C=\{\alpha_1, \ldots, \alpha_G\}$ множества B_1 на ОЛЦ. Пусть множество C включает минимально возможное число ОЛЦ, для чего используется методика [4].

Определение 5. Граф-схема алгоритма Г называется линейной, если выполняется условие

$$\frac{M}{G} \geq 2. \qquad (1)$$

Таким образом, ГСА является линейной, если число ее операторных вершин, хотя бы в два раза превосходит минимальное число ее ОЛЦ. При выполнении условия (1) для реализации УУ целесообразно использовать модель КМУУ [4].

Отметим, что каждая вершина $b_q \in B_1$ соответствует микрокоманде MI_q, хранимой в управляющей памяти (УП) по адресу $A(b_q)$. Для адресации микрокоманд достаточно

$$R = \lceil \log_2 M \rceil \qquad (2)$$

бит, где $M = |B_1|$. Используем для адресации переменные $T_r \in T$, где $|T| = R$.

Выполним адресацию микрокоманд так, чтобы для каждой пары соседних компонент ОЛЦ $\alpha_g \in C$ выполнялось условие:

$$A(b_{qi+1}) = A(b_{qi}) + 1. \qquad (3)$$

В выражении (3) подразумевается, что $q = \overline{1,G}$ и $i = \overline{1, F_g - 1}$. В этом случае для реализации УУ по ГСА Г может быть использована модель КМУУ с общей памятью (рис. 1). В дальнейшем эта модель обозначается символом U_1.

Рассмотрим принцип функционирования КМУУ U_1. По сигналу *Start* в счетчик *CT* заносится нулевой адрес первой микрокоманды исполняемой микропрограммы. Одновременно триггер выборки ТВ устанавливается в единичное состояние. При этом $Fetch = 1$, что разрешает выборку команд из УП. Пусть в момент времени t ($t = 1, 2, \ldots$) из УП считана микрокоманда MI_q. Если $b_q \neq O_g$, то одновременно с микрооперациями $y_n \in Y(b_q)$ формируется переменная y_0. Если $y_0 = 1$, то к содержимому *CT* прибавляется единица. Эта операция соответствует режиму (3). Если $b_q = O_g$, то выход ОЛЦ $\alpha_g \in C$ достигнут. При этом $y_0 = 0$, что позволяет запись в *CT* адреса перехода с выхода схемы формирования адреса (СФА). Блок СФА формирует функции возбуждения триггеров *CT*:

$$\Phi = \Phi(T, X). \qquad (4)$$

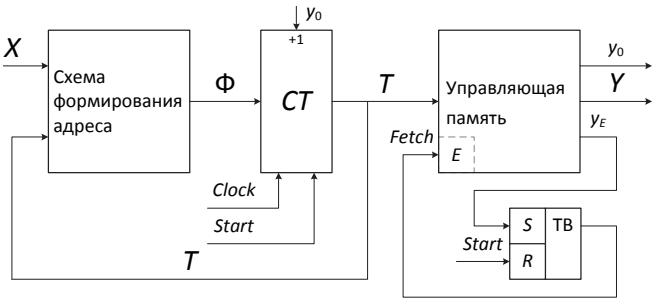


Рис. 1. Базовая модель КМУУ с общей памятью

Если в момент времени t выбрана микрокоманда MI_q, такая, что $\langle O_g, b_E \rangle \in E$, то формируется переменная y_E. Если $y_E = 1$, то $Fetch = 0$ и выборка микрокоманд прекращается.

Основной недостаток U_1 – число строк H_1 таблицы переходов определяется числом переходов эквивалентного автомата Мура. Этот автомат строится по блочной ГСА [4], в которой каждая ОЛЦ заменена одним блоком. Этот принцип иллюстрируется на рис. 2.

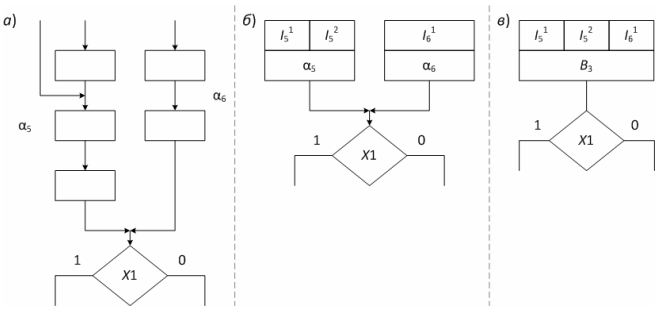


Рис. 2. Различные представления блоков ГСА

Пусть в ГСА Г существуют псевдоэквивалентные ОЛЦ (ПОЛЦ) $\alpha_5, \alpha_6 \in B_3$, где B_3 – не-

который класс ПОЛЦ (рис. 2,*а*). Схема СФА формирует функции (4) только при достижении выходов ОЛЦ. Поэтому ОЛЦ α_5 и α_6 могут быть заменены блоками с соответствующим числом входов. Как уже отмечалось, ОЛЦ – аналог состояния автомата Мура, что хорошо видно на рис. 2,*б*. Этот фрагмент ГСА вносит в систему (4) четыре терма. Если заменить ОЛЦ α_5 и α_6 блоком B_3 (рис. 2,*в*), то число переходов (а значит и число термов) уменьшится до двух. Таким образом, классы B_i соответствуют состояниям автомата Мили.

Предлагается метод, позволяющий гарантированно уменьшить число переходов КМУУ до величины этого параметра в эквивалентном автомате Мили. Метод основан на избыточности макроячеек ПМЛ по входам, а также избыточности блоков ППЗУ по выходам. Отметим, что ППЗУ используются для реализации управляющей памяти.

Основная идея предлагаемого метода

Пусть для ГСА Г с использованием методики [4] получено множество ОЛЦ $C=\{\alpha_1,\ldots,\alpha_G\}$. Построим множество $C_1 \subseteq C$, при этом, если $\langle Q_g, b_E\rangle \in E$, то $\alpha_g \notin C_1$. Найдем разбиение $\Pi_C = \{B_1,\ldots,B_I\}$ множества C_1 на классы псевдоэквивалентных ОЛЦ. Выполним адресацию (3) таким образом, чтобы максимально возможное число классов $B_i \in \Pi_C$ представлялось одним обобщенным интервалом R-мерного булева пространства. Назовем такую адресацию оптимальной.

Представим множество Π_C в виде $\Pi_C = \Pi_A \cup \cup \Pi_B$, где $\Pi_A \cap \Pi_B = \varnothing$. Пусть n_i – число обобщенных интервалов $V\left(B_i^j\right)$, представляющих класс $B_i \in \Pi_C$. Здесь j меняется от единицы до $n_i\left(i=\overline{1,I}\right)$. Множества Π_A и Π_B формируются следующим образом:

$$\begin{aligned}(n_i = 1) &\rightarrow B_i \in \Pi_A;\\ (n_i > 1) &\rightarrow B_i \in \Pi_B.\end{aligned} \qquad (5)$$

Отметим, что каждый класс $B_i \in \Pi_A$ однозначно определяется содержимым CT. Таким образом, классы $B_i \in \Pi_A$ соответствуют ситуации, показанной на рис. 2,*в*.

Закодируем классы $B_i \in \Pi_B$ двоичными кодами $C(B_i)$, имеющими

$$R_1 = \left\lceil \log_2\left(\left|\Pi_B\right|+1\right)\right\rceil \qquad (6)$$

разрядов. Единица в (6) добавляется для учета ситуации $B_i \in \Pi_A$. Для формирования кодов $C(B_i)$ необходим блок преобразователя адреса (ПА) выхода ОЛЦ $\alpha_g \in B_i$. Отметим, что достаточно преобразовывать только обобщенные интервалы $V\left(B_i^j\right)$, где $B_i \in \Pi_B$. Пусть для кодирования классов $B_i \in \Pi_B$ используются переменные $z_r \in Z$, где $|Z| = R_1$.

Пусть для реализации УП используются ППЗУ, имеющие S входов и t выходов. Примем для упрощения, что $R = S$. Тогда, для реализации УП требуется m_1 микросхем ППЗУ:

$$m_1 = \left\lceil \frac{N+2}{t} \right\rceil. \qquad (7)$$

Числитель формулы (7) определяется числом микроопераций (N) и двумя дополнительными переменными (y_0, y_E). Это следует непосредственно из рис. 1.

Очевидно, часть выходов ППЗУ блока УП может быть избыточной. Число избыточных выходов R_2 определяется формулой:

$$R_2 = m_1 t - \left(N+2\right). \qquad (8)$$

Эти выходы могут быть использованы для формирования R_2 переменных $z_r \in Z$. Таким образом, уменьшается число выходов схемы ПА (до $R_1 - R_2$). Это равносильно разбиению множества Z на классы Z^1 и Z^2, где $|Z^1| = R_1 - R_2$, $|Z^2| = R_2$.

С учетом приведенных рассуждений для реализации УУ предлагается модель U_2 (рис. 3).

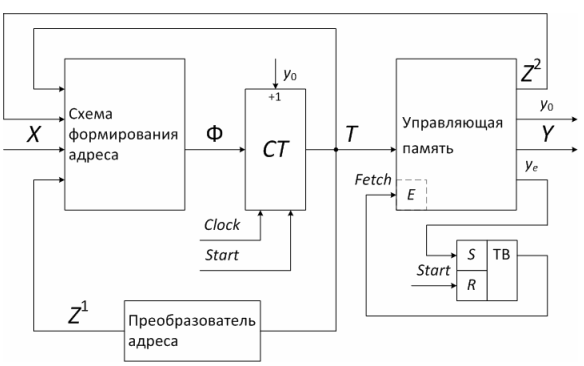


Рис. 3. Структурная схема КМУУ U_2

В КМУУ U_2 имеется три источника кодов классов $B_i \in \Pi_C$: счетчик CT, блок ПА и блок УП. Блок СФА реализует систему функций

$$\Phi = \Phi(T, Z, X). \quad (9)$$

Блок ПА реализует систему функций

$$Z^1 = Z^1(T), \quad (10)$$

а блок УП формирует переменные $y_n \in Y$, y_0, y_E и $z_r \in Z^2$, как функции от переменных $T_r \in T$.

Очевидно, КМУУ U_2 представляет собой обобщенную модель КМУУ с общей памятью и множественными источниками кодов классов ПОЛЦ. В КМУУ U_2 число переходов гарантировано равно числу переходов автомата Мили, синтезируемого по блочной ГСА вида, приведенного на рис. 2,*в*. В таблице показаны возможные модели КМУУ, которые могут быть получены на основе модели U_2.

В таблице показаны источники кодов классов $B_i \in \Pi_C$, а также условия применения этих моделей. Отметим, что модели U_1, U_3 и U_4 достаточно подробно рассмотрены в литературе [4]. Остальные модели носят оригинальный характер.

Таблица

Модель	СТ	ПА	УП	Условия применения модели
U_1	1	0	0	$\Pi_A = \Pi_C;\ \Pi_B = \varnothing$.
U_2	1	1	1	$\Pi_A \neq \varnothing;\ \Pi_B \neq \varnothing;\ 0 < R_2 < R_1$.
U_3	0	0	1	$\Pi_A = \varnothing;\ \Pi_B = \Pi_C;\ R_2 \geq R_1$;
U_4	0	1	0	$\Pi_A = \varnothing;\ \Pi_B = \Pi_C;\ R_2 = 0$;
U_5	0	1	1	$\Pi_A = \varnothing;\ \Pi_B = \Pi_C;\ 0 < R_2 < R_1$.
U_6	1	0	1	$\Pi_A \neq \varnothing;\ \Pi_B \neq \varnothing;\ R_2 \geq R_1$;
U_7	1	1	0	$\Pi_A \neq \varnothing;\ \Pi_B \neq \varnothing;\ R_2 = 0$;

В процессе исследований предложенных моделей авторами разработана САПР, совместимая с системой *WebPack*. Рассмотрим подробнее методику и результаты исследований.

Результаты исследований

Для проведения исследований, с целью определения области применения той или иной структуры, разработан программный комплекс (рис. 4).

Данный программный комплекс состоит из следующих элементов:

• *fsmEditor* – программа для создания и редактирования граф-схем алгоритмов с помощью графических примитивов. С помощью встроенного модуля *fsmGen* позволяет пакетно генерировать ГСА с заданными параметрами.

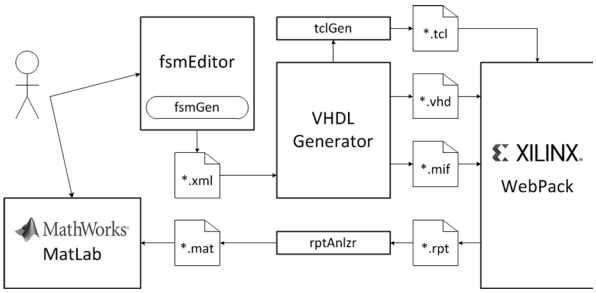


Рис. 4. Программный комплекс для проведения исследований

• *VHDL Generator* – позволяет, на основе *XML*-описания алгоритма и необходимых параметров, создать *VHDL*-модель управляющего автомата, функционирующего согласно ГСА.

• *tclGen* – утилита, необходимая для создания скрипта управления работой системы *XST*, входящей в *Xilinx WebPack*, на основе данных, полученных от *VHDL Generator*. Возможна пакетная генерация на основе набора созданных ранее **.vhd* файлов.

• *Xilinx WebPack* – программный пакет, необходимый для проектирования устройств на микросхемах ПЛИС фирмы *Xilinx*. Входящий в его состав модуль *XST*, на основе входных файлов **.vhd* и **.mif*, согласно скрипту управления, переданному от *tclGen*, генерирует файл прошивки микросхемы, а также множество файлов отчета, среди которых находятся данные об аппаратурных затратах, занимаемых при реализации схемы автомата на данной микросхеме.

• *rptAnlzr* – утилита, пакетно обрабатывающая наборы **.rpt* файлов, полученных на предыдущем шаге и формирующая на их основе таблицы данных в формате **.mat*

• *MathWorks MatLab* – пакет прикладных программ для решения задач технических вычислений. В данном комплексе выступает как средство построения графиков на основе полученных на предыдущих шагах данных.

Для эксперимента выбраны ГСА со следующими параметрами:

– количество вершин от 10 до 400 с шагом 10;

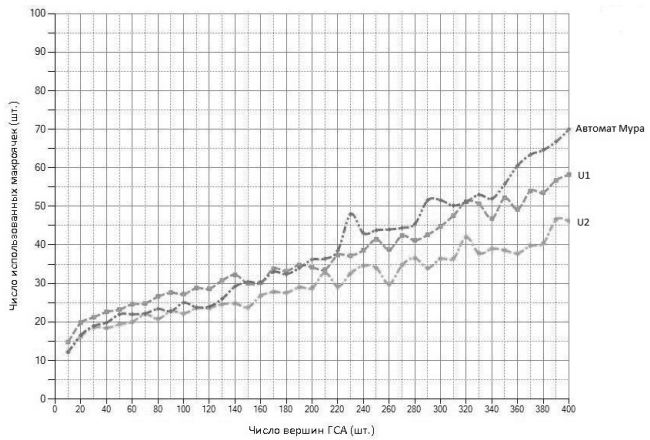
– доля операторных вершин от 50% до 90% с шагом 10%;

– количество микроопераций $N = 15$;

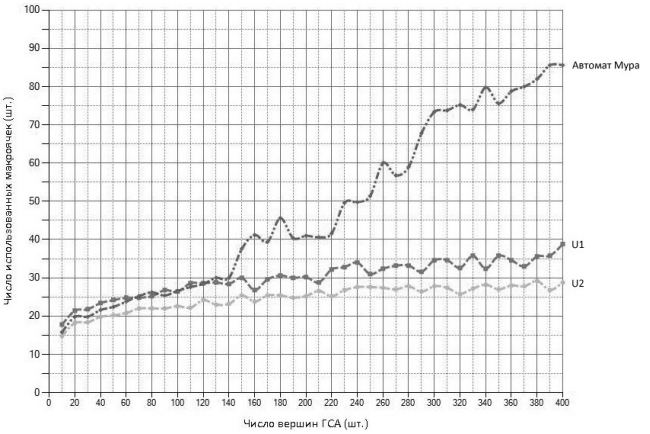
– количество логических условий $L = 5$.

Для каждой ГСА синтезирована структура КМУУ с общей памятью – U_1, а также исследуемая структура U_2. Для сравнения с классом автоматов с жесткой логикой также были синтезированы автоматы Мура. Каждое измерение, представленное на графиках, есть средним значением результатов синтеза пяти различных ГСА с одинаковыми параметрами.

Исследование аппаратурных затрат показало эффективность применения предлагаемых методик для всех исследуемых ГСА (рис. 5).



а



б

Рис. 5. Аппаратурные затраты при реализации структур КМУУ и автомата Мура. ГСА содержит: *а* – 60% операторных вершин; *б* – 90% операторных вершин

Как видно из рис. 5, модель КМУУ U_2 требует меньше аппаратурных затрат в сравнении с базовой моделью КМУУ U_1 и с автоматом Мура. Заметим, что с ростом процента операторных вершин в ГСА аппаратурные затраты для реализации автомата Мура увеличиваются, а для реализации устройства класса КМУУ уменьшаются.

Максимальный абсолютный выигрыш при использовании структуры U_2 в сравнении с применением структуры автомата Мура достигал около 40 макроячеек для ГСА с 60% операторных вершин и 50 макроячеек для ГСА с 90% операторных вершин.

Заключение. Проведенные исследования для КМУУ U_2 подтвердили гипотезу об уменьшении аппаратурных затрат при реализации композиционных микропрограммных устройств управления с несколькими источниками кодов в базисе *CPLD*-микросхем в сравнении с известными методами реализации КМУУ. Это уменьшение возможно благодаря таким главным факторам, как большой коэффициент объединения по входу современных макроячеек ПМЛ, а также естественная избыточность микросхем ППЗУ, обусловленная тем, что количество их выходов ограничено определенным множеством чисел. Как показали проведенные исследования, метод наиболее эффективен для интерпретации линейных ГСА. Средняя эффективность от применения метода составила 30% в сравнении с известными методами реализации КМУУ.

Научная новизна предложенного метода заключается в использовании особенностей КМУУ (наличие классов псевдоэквивалентных ОЛЦ) и элементного базиса для уменьшения аппаратурных затрат в логической УУ. Практическая значимость метода заключается в уменьшении площади кристалла, занимаемой комбинационной схемой КМУУ, что позволяет получить схемы, обладающие меньшей стоимостью, чем известные из литературы аналоги.

Дальнейшие исследования связаны с разработкой методов синтеза других структур КМУУ, описанных в таблице, сравнением параметров синтезированных схем, составлением аналитических зависимостей между параметрами исходных ГСА и характеристиками получаемых схем, позволяющих определять устройство, наилучшим образом удовлетворяющее выдвигаемым требованиям.

Окончание на стр. 78.

Кроме того, планируется исследование применения предложенных методов к микросхемам ПЛИС типа *FPGA*.

1. *Глушков В.М.* Синтез цифровых автоматов. – М.: Физматгиз, 1962. – 476 с.
2. *De Micheli G.* Synthesis and Optimization of Digital Circuits. – N.Y.: McGraw Hill, 1994. – 541 p.
3. *Barkalov A.*, *Titarenko L.* Logic Synthesis for FSM-based Control Units. – Berlin: Springer, 2009. – 233 p.
4. *Barkalov A.*, *Titarenko L.* Logic Synthesis for Compositional Microprogram Control Units. – Berlin: Springer, 2008. – 272 p.
5. *Грушвицкий Р.И.*, *Мурсаев А.Х.*, *Угрюмов Е.П.* Проектирование систем на микросхемах программируемой логики. – Петербург: БХВ-Петербург, 2002. – 636 с.
6. *Соловьев В.В.*, *Климович А.С.* Логическое проектирование цифровых систем на основе программируемых логических интегральных схем. – М.: Горячая линия – Телеком, 2008. – 376 с.
7. *Altera* devices overview. – http://www.altera.com/products/devices/common/dev-family_overview.html
8. *Xilinx* CPLDs. – http://www.xilinx.com/products/silicon_solutions/cplds/index.htm
9. *Баркалов А.А.*, *Цололо С.А.* Оптимизация числа макроячеек *PAL* в схеме автомата Мура // УСиМ. – 2008. – № 2. – С. 54–59.
10. *Баркалов А.А.*, *Цололо С.А.* Оптимизация автомата Мура, реализуемого в базисе *CPLD* // Там же. – № 4. – С. 43–48.
11. *Baranov S.* Logic and System Design of Digital Systems. – Tallinn: TUT Press, 2008. – 266 p.

Поступила 13.11.2011
Тел. для справок: +380 62 301-0735 (Донецк)
E-mail: *A.Barkalov@iie.uz.zgora.pl*