УДК 681. 324

А.А. Баркалов, Л.А. Титаренко, А.Ф. Буковец

РЕАЛИЗАЦИЯ НА FPGA МИКРОПРОГРАММНЫХ АВТОМАТОВ МИЛИ, ИНТЕРПРЕТИРУЮЩИХ ВЕРТИКАЛИЗОВАННЫЕ АЛГОРИТМЫ УПРАВЛЕНИЯ

Предлагаются структуры логической схемы микропрограммного автомата Мили, ориентированные на уменьшение числа LUT-элементов при реализации автомата в базисе FPGA. Методы основываются на вертикализации исходной граф-схемы алгоритма, что позволяет добиться полной совместимости микроопераций и использовать только один дешифратор для реализации системы микроопераций. Предложенный подход является альтернативой метода кодирования полей совместимых микроопераций. Дается сравнительный анализ предложенных структур и определяются области их эффективного применения.

Введение

Согласно принципу микропрограммного управления любая цифровая система может быть представлена как композиция управляющего (УА) и операционного (ОА) автоматов [1, 2]. В настоящее время имеется возможность реализации цифровых систем, включающих сотни миллионов эквивалентных вентилей, с помощью микросхемы типа «система-на-кристалле» [3–6]. Как правило, в таких микросхемах произвольная логика реализуется на программируемых вентильных матрицах (FPGA, field-programmable logic array), содержащих табличные элементы (LUT, look-up table) с ограниченным числом входов [6]. Логическая схема УА отличается нерегулярностью структуры и не может быть реализована в виде стандартного IP-ядра (intellectual property). В этой связи возникает задача реализации схемы УА на FPGA, при этом часто необходимо получить схему с минимальным числом LUT-элементов. Как правило, чем больше аргументов содержит функция, тем больше LUT-элементов требуется для ее реализации [6, 7]. В настоящей работе рассматриваются структуры логических схем микропрограммных автоматов (МПА) Мили, позволяющие уменьшить число функций с большим числом аргументов. Предлагаемые методы являются альтернативой методов, основанных на кодировании полей совместимых микроопераций.

1. Анализ методов реализации микропрограммных автоматов Мили с кодированием полей совместимых микроопераций

Пусть алгоритм управления цифровой системой задан в виде граф-схемы алгоритма (ГСА) Γ [1], операторные вершины которой образуют множество $O(\Gamma) = \{O_1, \ldots, O_K\}$. В операторной вершине $O_k \in O(\Gamma)$ записывается набор одновременно выполняемых микроопераций (микрокоманда) $Y(O_k) \subseteq Y$, где $Y = \{y_1, \ldots, y_N\}$ – множество микроопераций. В условных вершинах ГСА Γ записываются элементы множества логических условий $X = \{x_1, \ldots, x_L\}$. Пусть ГСА Γ отмечена состояниями автомата Мили [1], образующими множество $A = \{a_1, \ldots, a_M\}$. Поставим в соответствие каждому состоянию $a_m \in A$ двоичный код $K(a_m)$ разрядности $R =]\log_2 M[$, используя для кодирования элементы множества внутренних переменных $T = \{T_1, \ldots, T_R\}$. Поведение МПА может быть задано прямой структурной таблицей (ПСТ) со столбцами [1]: a_m, $K(a_m)$, a_s, $K(a_s)$, X_h, Y_h, Φ_h, h. Здесь a_m – исходное состояние МПА; a_s – состояние перехода автомата; X_h – входной сигнал, определяющий переход $\langle a_m, a_s \rangle$ и равный конъюнкции некоторых элементов множества X; $Y_h \subseteq Y$ – выходной сигнал на переходе $\langle a_m, a_s \rangle$; $\Phi_h \subseteq \Phi$ – набор функций возбуждения триггеров памяти из

множества $\Phi = \{\varphi_1, \ \ldots, \ \varphi_R\}$, принимающих единичное значение для переключения триггеров памяти МПА из $K(a_m)$ в $K(a_s)$; $h = 1, \ \ldots, \ H$ – номер перехода.

Логическая схема автомата задается системой функций

$$\Phi = \Phi(T, X); \tag{1}$$

$$Y = Y(T, X), \tag{2}$$

формируемой по ПСТ. Термы дизъюнктивных нормальных форм (ДНФ) функций (1), (2) представляют собой конъюнкции

$$F_h = A_m^h X_h \quad (h = 1, \ \ldots, \ H), \tag{3}$$

где A_m^h – конъюнкция внутренних переменных, соответствующая коду $K(a_m)$ исходного состояния $a_m \in A$ из h-й строки ПСТ $(h = 1, \ \ldots, \ H)$. Система (1), (2) является основой для построения одноуровневой схемы МПА Мили, называемой Р-автоматом. Такой автомат включает регистр RY, необходимый для обеспечения устойчивости работы системы <УА, ОА> [1], и Р-подсхему, формирующую функции (1) и (2). Память МПА представлена регистром RG. Регистр RY хранит микрооперации Y_R, поступающие в ОА. Регистры RG и RY реализуются с использованием D-триггеров, связанных с выходами LUT-элементов FPGA [3, 6].

Пусть $X(F_h) \subseteq X$ – множество логических условий, образующих терм F_h $(h = 1, \ \ldots, \ H)$. Создадим множество $F = \Phi \cup Y$ и обозначим через $X(f_i)$ множество логических условий, входящих в ДНФ функции $f_i \in F$ $(i = 1, \ \ldots, \ N + R)$. В Р-автомате Р-подсхема имеет $t_1 = R + N$ выходов и каждая из функций $f_i \in F$ имеет $S_i = L_i + R$ аргументов, где $L_i = |X(f_i)|$, $i = 1, \ \ldots, \ N + R$. Для автоматов средней сложности [6, 7] $R \approx 8$, $N \approx 50$, $L \approx 30$, $|X(F_h)| \le 10$. Таким образом, даже для этого случая Р-подсхема реализует до 58 функций, каждая из которых имеет до 38 аргументов. Это приводит к значительным аппаратурным затратам в Р-подсхеме, а необходимость функциональной декомпозиции [7] систем функций (1) и (2) в силу ограниченности числа входов LUT-элементов приводит к увеличению числа уровней в Р-подсхеме и к уменьшению быстродействия цифровой системы.

Для уменьшения аппаратурных затрат в схемах МПА Мили на FPGA могут быть использованы методы структурной редукции [1, 6], в частности метод кодирования полей совместимых микроопераций. Напомним, что микрооперации $y_n, y_m \in Y$ называются совместимыми, если выполняется условие

$$y_n \in Y(O_k) \to y_m \notin Y(O_k) \quad (k = 1, \ \ldots, \ K). \tag{4}$$

Метод основан на нахождении разбиения $\pi_Y = \{Y^1, \ \ldots, \ Y^I\}$ множества Y на классы совместимых микроопераций с целью минимизации параметра

$$R_2 = \sum_{i=1}^{I} r_i, \tag{5}$$

где $r_i =]\log_2(|Y^i| + 1)[$ – минимальное число двоичных переменных, необходимых для кодирования микроопераций $y_m \in Y^i$ $(i = 1, \ \ldots, \ I)$. Каждой микрооперации ставится в соответствие код $K(y_n)$ разрядности r_i $(i = 1, \ \ldots, \ I)$. Кодирующие переменные для всех классов $Y^i \in \pi_Y$ образуют множество Z, включающее R_2 элементов.

Теперь МПА Мили может быть реализован как PD-автомат [8], в котором Р-подсхема реализует систему (1) и функции

$$Z = Z(T, X). \tag{6}$$

В PD-автомате система микроопераций Y реализуется в виде

$$Y = Y(Z), \tag{7}$$

для чего используется D-подсхема, содержащая I дешифраторов, i-й дешифратор соответствует классу $Y^i \in \pi_Y$ и имеет r_i входов.

Такой подход позволяет уменьшить число функций, зависящих от термов (3), до $t_2 = R + R_2$. В настоящей работе предлагается метод дальнейшего уменьшения числа выходов P-подсхемы, основанный на вертикализации исходной ГСА Γ [1]. Уменьшение числа функций, реализуемых P-подсхемой, дает потенциальную возможность уменьшения числа LUT-элементов, образующих P-подсхему.

2. Основная идея метода

Назовем ГСА Γ вертикальной ГСА, если

$$|Y(O_K)| \le 1 \quad (k = 1, \ \ldots, \ K), \tag{8}$$

т. е. если в каждой ее операторной вершине записано не более одной микрооперации. Характерным свойством вертикальных ГСА является полная совместимость микроопераций, при которой $\pi_Y = Y$. Это позволяет закодировать микрооперации $y_n \in Y$ двоичными кодами $K(y_n)$ разрядности $R_3 =]\log_2 N[$ (максимальное кодирование микроопераций) и использовать только один дешифратор для реализации D-подсхемы. Отметим, что в общем случае произвольная ГСА Γ не является вертикальной. В настоящей работе предлагается использовать процедуру вертикализации произвольной ГСА Γ [1] для достижения полной совместимости микроопераций.

Рассмотрим фрагмент ГСА Γ_1 (рис. 1, *а*), которая не удовлетворяет условию (8).

Процесс вертикализации заключается в «расщеплении» каждой вершины $O_k \in O(\Gamma)$ на $N_k = |Y(O_k)|$ операторных вершин $O_k^1, \ \ldots, \ O_k^{N_k}$, при котором каждая вершина O_k^i содержит по одной уникальной микрооперации из вершины O_k $(i = 1, \ \ldots, \ N_k; k = 1, \ \ldots, \ K)$.

В настоящей работе предлагаются два варианта отметки полученной ГСА, которую назовем вертикализованной граф-схемой алгоритма (ВГСА) и обозначим символом $V(\Gamma)$:

– с использованием стандартной процедуры построения отмеченной ГСА автомата Мили [1];

– с сохранением отметок, присвоенных исходной ГСА Γ.

Применение этих процедур к фрагменту ГСА Γ_1 приводит к фрагментам ВГСА $V(\Gamma_1)$, показанным на рис. 1, *б* и *в* соответственно. Назначение сигнала y_0 будет рассмотрено ниже.

Очевидно, что автомат Мили, интерпретирующий ВГСА $V(\Gamma)$, требует больше времени для выполнения алгоритма управления по сравнению с МПА Мили, интерпретирующим исходную ГСА Γ. Кроме того, последовательное выполнение микроопераций $y_n \in Y(O_k)$ не всегда возможно из-за наличия зависимости по данным [1]. Для устранения этих недостатков в работе предлагается преобразование последовательности β_k микроопераций, соответствующих микрокоманде $Y(O_k)$ исходной ГСА, в параллельный код и запуск ОА только после формирования в регистре RY всех микроопераций $y_n \in Y(O_k)$ $(k = 1, \ \ldots, \ K)$. Для такого преобразования триггеры регистра RY должны синхронизироваться локальными синхросигналами, что возможно во всех современных FPGA [3, 4]. Сигнал y_0 будет использоваться для запуска операционного автомата.

В настоящей работе предлагаются три структуры логической схемы МПА Мили, интерпретирующего ВГСА $V(\Gamma)$ (табл. 1).

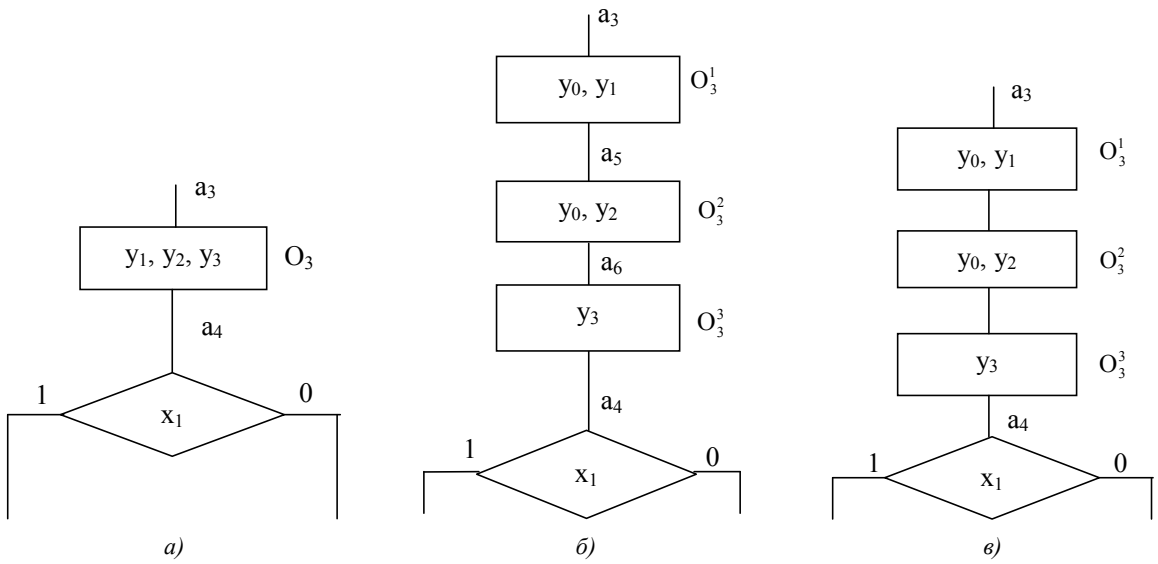
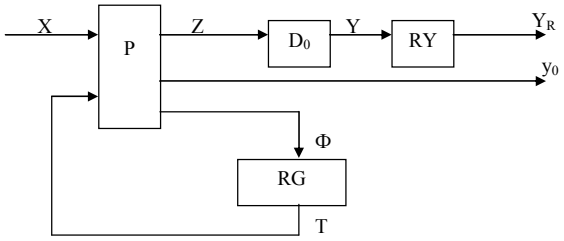
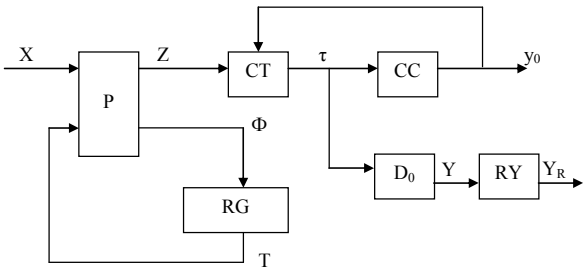
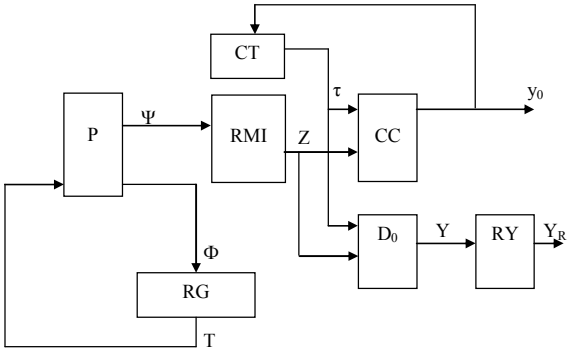


Рис. 1. Фрагмент отмеченной ГСА Γ_1 : *а)* до вертикализации; *б)*, *в)* после

Таблица

Структурные схемы автоматов Мили, интерпретирующих ВГСА

Обозначение	Структурная схема
PD^1_V	
PD^2_V	
PD^3_V	

3. Структуры логической схемы МПА Мили

Структуры логической схемы МПА Мили обозначим $PD_V^1 - PD_V^3$ (таблица). Здесь PD_V^1 – автомат Мили, который интерпретирует отмеченную ВГСА $V(\Gamma)$, соответствующую фрагменту на рис. 1, *б*, а PD_V^2 - и PD_V^3 -автоматы – фрагменту на рис. 1, *в*. Во всех предлагаемых структурах микрооперации $y_n \in Y$ формируются D_0 -подсхемой, состоящей из одного дешифратора.

Сигнал запуска ОА y_0 формируется либо непосредственно Р-подсхемой (PD_V^1 -автомат), либо специальным преобразователем кодов СС (PD_V^2 - и PD_V^3 -автоматы).

В PD_V^1 -автомате Р-подсхема формирует функции (1), (6) и $y_0 = y_0(T, X)$, для чего требуется

$$t_3 = R_4 + R_3 + 1 \tag{9}$$

выходов. Здесь $R_4 =]\log_2 M_1[$, где M_1 – мощность множества состояний PD_V^1 -автомата, определяемая как

$$M_1 = M + \sum_{k=1}^{K} (N_k - 1). \tag{10}$$

Для синтеза схемы PD_V^1 -автомата необходимо построить его ПСТ, закодировать микрооперации $y_n \in Y$ R_3-разрядными кодами $K(y_n)$ и преобразовать исходную ПСТ, заменив микрооперации $y_n \in Y$ переменными $z_r \in Z$, принимающими единичные значения в коде $K(y_n)$. Недостатком такого подхода является возможность увеличения разрядности кодов состояний по сравнению с эквивалентным PD-автоматом. Достоинство метода – минимально возможная разрядность кодов микроопераций.

В PD_V^2 -автомате Р-подсхема формирует только функции (1) и (6), а сигнал y_0 формируется схемой СС. Для синтеза схемы необходимо выполнить естественную адресацию микроопераций в последовательностях β_k $(k = 1, \ \ldots, \ K)$:

$$(y_n^k = p_i^r \beta_k \ \& \ y_m^k = p_{i+1}^r \beta_k = 1) \to K(y_m^k) = K(y_n^k) + 1. \tag{11}$$

Здесь индекс k в обозначениях микроопераций подчеркивает тот факт, что микрооперация $y_n \in Y$ должна иметь разные коды для разных микрокоманд $Y(O_k) \subseteq Y$. Для выполнения условия (11) необходимо кодировать микрооперации $y_n^k \in Y_0$, где

$$| Y_0 | = N_0 = \sum_{k=1}^{K} N_k \ . \tag{12}$$

При этом код $K(y_n^k)$ будет иметь $R_5 =]\log_2 N_0[$ разрядов и число выходов Р-подсхемы определяется как

$$t_4 = R + R_5. \tag{13}$$

Код $K(y_n^k)$ хранится в счетчике СТ, содержимое которого увеличивается на единицу при $y_0 = 1$. Если $y_0 = 1$, то состояние PD_V^2 -автомата не меняется. Недостатком такого подхода являются увеличение числа выходов дешифратора D_0 по сравнению с N и необходимость формирования микроопераций $y_n \in Y$ в виде

$$y_n = \bigvee_{k=1}^{K} C_{nk} y_n^k \quad (n = 1, \ \ldots, \ N), \tag{14}$$

где C_{nk} – булева переменная, равная единице, если и только если $y_n \in Y(O_k)$. Представление (14) вызывает необходимость объединения соответствующих выходов дешифратора D_0. Преимущество такого подхода – сохранение разрядности кодов состояний, что упрощает функции (1) и (6) по сравнению с PD_V^1-автоматом.

В PD_V^3-автомате также формируются микрооперации $y_n^k \in Y$, но код $K(y_n^k)$ представляется в виде конкатенации

$$K(y_n^k) = K(Y_k) * C(y_n^k), \tag{15}$$

где * – знак конкатенации; $K(Y_k)$ – код микрокоманды $Y(O_k)$ разрядности $R_6 =]\log_2 Q[$; Q – число микрокоманд в ГСА Γ; $C(y_n^k)$ – код микрооперации $y_n^k \in Y(O_k)$ разрядности $R_7 =]\log_2 M_0[$, $M_0 = \max(N_1, \ \ldots, \ N_K)$. Код $K(Y_k)$ формируется в регистре микрокоманды RMI при помощи функций

$$\Psi = \Psi(T, X), \tag{16}$$

создаваемых P-подсхемой.

Код $C(y_n^k)$ формируется в счетчике CT, содержимое которого увеличивается на единицу при $y_0 = 1$ и обнуляется при $y_0 = 0$. При этом режим адресации (11) реализуется путем выполнения операции

$$C(y_m^k) = C(y_n^k) + 1, \tag{17}$$

где $y_n^k = pr_i\beta_k$, $y_m^k = pr_{i+1}\beta_k$ $(i = 1, \ \ldots, \ N_k - 1; \ k = 1, \ \ldots, \ K)$. Сигнал y_0 формируется схемой CC в виде

$$y_0 = y_0(\tau, Z). \tag{18}$$

В этом случае число выходов P-подсхемы является минимально возможным и равным

$$t_5 = R + R_6, \tag{19}$$

причем число входов P-подсхемы также минимально. Недостаток данного подхода заключается в увеличении разрядности кода микрооперации $K(y_n^k)$ до $R_6 + R_7 \geq R_5$. Однако такая избыточность позволяет минимизировать функции $y_n^k \in Y_0$ за счет наличия несущественных наборов. Очевидно, выходные функции $y_n \in Y$ также формируются по закону (14).

Заключение

Использование вертикализации исходной ГСА, интерпретируемой автоматом Мили, позволяет добиться полной совместимости всех микроопераций и уменьшить аппаратурные затраты по сравнению с кодированием полей совместимых микроопераций. Проведенные авторами исследования по реализации схем предложенных автоматов PD_V^1 – PD_V^3 в базисе FPGA показали, что применение предложенных методов всегда позволяет уменьшить число LUT-элементов в схеме в сравнении с PD-автоматом. Применение PD_V^1-автомата наиболее эффективно, если число переменных, кодирующих внутренние состояния, для PD- и PD_V^1-автоматов совпадает. В противном случае более эффективны PD_V^2- и PD_V^3-автоматы. Приме-

нение PD_V^3-автомата целесообразно, если разрядности кодов микроопераций в PD_V^2- и PD_V^3-автоматах совпадают, в противном случае PD_V^2-автомат требует меньше LUT-элементов. Однако конкретный выбор зависит от характеристик исходной ГСА.

Для исследования эффективности предложенных методов был разработан программный комплекс, в состав которого входил генератор граф-схем (C++) и модуль формирования уравнений, описывающих схемы рассмотренных в настоящей статье автоматов (C++). Полученные уравнения использовались системой ISE фирмы Xilinx для синтеза схем автоматов в базисе FPGA. Как показали исследования авторов, лучшая в конкретных условиях модель автомата Мили, интерпретирующего ВГСА, позволяет уменьшить на 24–32 % число LUT-элементов в сравнении с автоматами, использующими принцип кодирования полей совместимых микроопераций. Отметим, что предложенные МПА обладают меньшим быстродействием, чем PD-автоматы, поэтому их применение целесообразно, только если критерием эффективности схемы является минимум аппаратурных затрат.

Список литературы

1. Баркалов, А.А. Синтез операционных устройств / А.А. Баркалов. – Донецк: РВА ДонНТУ, 2003. – 306 с.

2. Gajski, D. Principles of Digital Systems / D. Gajski. – Prentice Hall, NJ, 1997. – 412 p.

3. Грушвицкий, Р.И. Проектирование систем на микросхемах программируемой логики / Р.И. Грушвицкий, А.Х. Мурсаев, Е.П. Угрюмов. – СПб.: БХВ – Петербург, 2002. – 608 с.

4. Соловьев, В.В. Проектирование цифровых систем на основе программируемых логических интегральных схем / В.В. Соловьев. – М.: Горячая линия – Телеком, 2001. – 636 с.

5. Bursky, D. Embedded Logic and Memory Fund Home in FPGA / D. Bursky // Electronic Design. – 1999. – № 14. – P. 43–56.

6. De Micheli, G. Synthesis and Optimization of Digital Circuits / G. De Micheli. – McGraw-Hill, NY, 1994. – 574 p.

7. Sasao, T. Switching theory for logic synthesis / T. Sasao. – Boston: Kluwer Academic Publishers, 1999. – 362 p.

8. Баранов, С.И. Цифровые устройства на программруемых БИС с матричной структурой / С.И. Баранов, В.А. Скляров. – М.: Радио и связь, 1986. – 272 с.

Поступила 16.01.06

Зеленогурский университет,
Польша, Зеленая Гура, Подгорная, 50
E-mail: a.barkalov@iie.uz.zgora.pl
l.titarenko@iie.uz.zgora.pl
a.bukowiec@iie.uz.zgora.pl

A.A. Barkalov, L.A. Titarenko, A.F. Bukowec

REALIZATION OF MEALY FINITE-STATE-MACHINE ON FPGA FOR INTERPRETATION OF VERTICAL CONTROL ALGORITHMS

Structures of logical circuit of Mealy finite-state-machine are proposed, which are oriented on decreasing of amount of LUT-elements under implementation of FSM on FPGAs. The proposed methods are based on verticalization of initial flow-chart of algorithm, that permits to reach the total compatibility of microoperations and to use only single decoder to implement the system of microoperations. The proposed approach is alternative to method of encoding fields of compatible microoperations. An comparative analysis of proposed structures was done and areas of their effective application are determined.