УДК 519.711
https://doi.org/10.37661/1816-0301-2023-20-1-91-101

Оригинальная статья
Original Paper

Аппаратная реализация булевых функций на основе автоматной модели

А. А. Бутов

Белорусский государственный университет информатики и радиоэлектроники,
ул. П. Бровки, 6, Минск, 220013, Беларусь
E-mail: tmkrb9@gmail.com

Аннотация

Цели. В настоящее время электронные устройства управления все шире внедряются в различные изделия бытового и производственного назначения. В качестве таких устройств широко применяются микроконтроллеры самой разной конфигурации. Можно предложить другой подход, где устройство управления со стандартной структурой синтезируется из типовых интегральных схем и реализует булеву функцию, описывающую требуемые управляющие воздействия. Целью работы является исследование возможности реализации булевых функций с помощью устройств со стандартной структурой, проектирование которых основано на использовании модели дискретного автомата.

Методы. Исходная булева функция, подлежащая реализации, задается в виде дизъюнктивной нормальной формы. Для нее строится бинарная диаграмма решений (англ. Binary Decision Diagram, BDD), оптимизированная по числу вершин, на основе которой формируется граф переходов синхронного автомата Мура с абстрактным состоянием. Далее после выполнения этапа кодирования состояний автомата на основе его таблицы переходов формируется входная информация для прошивки (программирования) матричной памяти постоянного запоминающего устройства (ПЗУ).

Результаты. Устройство, реализующее булеву функцию на основе автоматной модели, синтезируется из типовых микросхем. Основным компонентом служит ПЗУ, которое в соответствии со стандартной структурой устройства дополняется сдвиговым регистром, регистром состояний, триггером и тремя селекторами начального и двух финальных состояний.

Заключение. Процесс проектирования устройства со стандартной структурой, реализующего булеву функцию, в итоге сводится к программированию матричной памяти ПЗУ на основе автоматной таблицы переходов. Использование многократно программируемого ПЗУ позволяет изменять функциональность устройства при сохранении прежней схемной реализации. Недостатком такого устройства, так же как и устройств, реализованных на основе микроконтроллеров, является низкое быстродействие, достоинством – возможность использования в различных изделиях и приборах, прежде всего бытового назначения, которые не требуют высокоскоростной реакции на изменение входного сигнала.

Ключевые слова: синтез комбинационных схем, булева функция, дискретный автомат, бинарная диаграмма решений, постоянное запоминающее устройство

Для цитирования. Бутов, А. А. Аппаратная реализация булевых функций на основе автоматной модели / А. А. Бутов // Информатика. – 2023. − Т. 20, № 1. – С. 91–101.
https://doi.org/10.37661/1816-0301-2023-20-1-91-101

Конфликт интересов. Автор заявляет об отсутствии конфликта интересов.

Поступила в редакцию | Received 19.12.2022
Подписана в печать | Accepted 20.01.2023
Опубликована | Published 29.03.2023

Hardware implementation of Boolean functions based on the automaton model

Alexey A. Butov

Belarusian State University of Informatics and Radioelectronics,
st. P. Brovka, 6, Minsk, 220013, Belarus
E-mail: tmkrb9@gmail.com

Abstract
Objectives. Currently, electronic control devices are increasingly being introduced into various household and production products. Microcontrollers of a wide variety of configurations are widely used as such devices. Another approach can be proposed where a control device with a standard structure is synthesized from typical integrated circuits and implements a Boolean function describing the required control actions. The purpose of the work is to investigate the possibility of implementing Boolean functions using devices with a standard structure, the design of which is based on the use of a discrete automaton model.
Methods. The original Boolean function to be implemented is given as a disjunctive normal form. A binary decision diagram (BDD) is built for such function, optimized by the number of vertices, on the basis of which a graph of transitions of a synchronous Moore automaton with an abstract state is formed. Further, after performing the state encoding step of the machine, input information for flashing (programming) of the matrix memory of the read-only memory (ROM) is generated based on its transition table.
Results. A device that implements a Boolean function based on an automaton model is synthesized from typical microcircuits. The main component is ROM, which, according to the standard structure of the device, is supplemented by a shift register, a state register, a trigger and three selectors of the initial and two final states.
Conclusion. The process of designing a device with standard structure that implements the Boolean function, as a result, comes down to programming the ROM matrix memory based on an automaton transition table. The use of a reprogrammable ROM allows to change the functionality of the device while maintaining the previous circuit implementation. The disadvantage of such a device, as well as devices implemented on the basis of microcontrollers, is the low speed, the advantage is the possibility of use it in various products and devices, primarily for household purposes, which do not require a high-speed response to the change of input signal.

Keywords: synthesis of combination circuits, Boolean function, discrete automaton, binary decision diagram, read-only memory

For citation. Butov A. A. *Hardware implementation of Boolean functions based on the automaton model*. Informatika [*Informatics*], 2023, vol. 20, no. 1, pp. 91−101 (In Russ.).
https://doi.org/10.37661/1816-0301-2023-20-1-91-101

Conflict of interest. The author declares of no conflict of interest.

Введение. В настоящее время электронные устройства управления применяются не только в сложных технических устройствах и системах, но и в различных широко распространенных предметах повседневного пользования: бытовой технике, медицинском оборудовании, измерительной технике, электронных детских игрушках, торговых автоматах, роботах и др. Это в значительной мере обусловлено применением в них микроконтроллеров [1], выполняющих функции управления и контроля. Микросхема микроконтроллера предварительно программируется на выполнение определенных функций по управлению устройством, в которое она интегрирована, путем анализа значений входных переменных и выдачи команд, направленных на изменение состояния этого устройства. Микроконтроллеры применяются там, где не предъявляются жесткие требования к обеспечению быстродействия устройств, что характерно для огромного числа реальных объектов, допускающих автоматизацию их работы.

Электронной промышленностью выпускается широкий спектр различных микроконтроллеров, для которых имеются соответствующие среды разработки, автоматизирующие процесс прошивки памяти микроконтроллера. Для использования микроконтроллера в каком-либо проектируемом изделии необходимо выбрать не только микроконтроллер с нужной функциональностью, но и подходящую интегрированную среду разработки (Integrated Development

Environment, IDE), а также освоить работу в этой среде, включая изучение используемого в ней языка программирования (возможно, специализированного), программных библиотек, отладчика (симулятора) и др.

В настоящей статье предлагается другой подход к разработке подобных средств автоматизации, развиваемый в русле таких работ, как, например, [2–4]. Согласно этому подходу сначала строится автоматная модель, по которой с помощью типовых интегральных схем серий К155, К176, К555 или аналогичных им синтезируется устройство управления со стандартной структурой. Это устройство вычисляет значение булевой функции за некоторое число шагов, выполняемых в соответствии с автоматной таблицей переходов.

Описание подхода. Комбинационную схему (комбинационное устройство) с n входами и одним выходом на уровне функциональной модели можно задать булевой функцией $y = f(\mathbf{x})$, где компонентами вектора $\mathbf{x} = (x_1, x_2, ..., x_n)$ являются булевы переменные, образующие в совокупности множество X.

Можно считать, что комбинационные схемы отличаются от последовательностных схем тем, что имеют лишь одно внутреннее состояние, в то время как последние, описываемые автоматной моделью, могут изменять свое внутреннее состояние и, соответственно, поведение.

Суть предлагаемого подхода к проектированию устройства, реализующего булеву функцию f, заключается в том, что оно проектируется как последовательностная схема, основанная на синхронной автоматной модели Мура с абстрактным состоянием. В данной модели в каждом состоянии автомата анализируется значение лишь одной из переменных множества X. В следующий дискретный момент времени в зависимости от значения этой переменной выполняется переход в одно из двух новых состояний, в каждом из которых будет анализироваться значение какой-то другой переменной из множества X и т. д.

Процесс последовательного анализа значений переменных множества X, сопровождающийся изменением состояний автомата, заканчивается тогда, когда автомат окажется в одном из двух финальных состояний, соответствующих значениям нуля и единицы функции f. При этом будут последовательно проанализированы значения всех переменных, входящих в множество X или в некоторое его подмножество. В последнем случае процесс достижения финального состояния заканчивается раньше, так как значение функции f в какой-то момент времени становится известным и дальнейший анализ уже не требуется. После достижения одного из финальных состояний будет определено значение функции f. В следующий дискретный момент времени выполняется безусловный переход в начальное состояние автомата и тем самым запускается новый аналогичный процесс последовательного анализа входных переменных с соответствующей сменой внутренних состояний для нахождения обновленного значения функции f. Значения входных переменных $x_1, x_2, ..., x_n$ каждый раз обновляются в момент перехода автомата в начальное состояние, а значение функции f может корректироваться только в момент очередного достижения автоматом одного из финальных состояний. С использованием данной модели значение сигнала на выходе устройства будет обновляться с некоторой задержкой, поскольку каждый переход автомата из одного состояния в другое требует одного такта работы тактового генератора.

Обобщенная структура устройства. Компоненты устройства, работающего на основе автоматной модели, показаны на рис. 1. Для простоты предполагается, что изменение состояния каждого из ее компонентов, обладающих памятью, происходит (с некоторой задержкой, вызванной инерцией срабатывания электронных схем) только в момент изменения управляющего сигнала от уровня логического нуля к уровню логической единицы, т. е. по положительному перепаду сигнала или, как говорят, по его переднему фронту. Отметим, что разные серии микросхем и входящие в них компоненты имеют свои особенности и используют в целях управления своими состояниями не только положительные, но и отрицательные перепады. Во втором случае соответствующие входы называются инверсными и обозначаются на рисунках небольшим кружком.

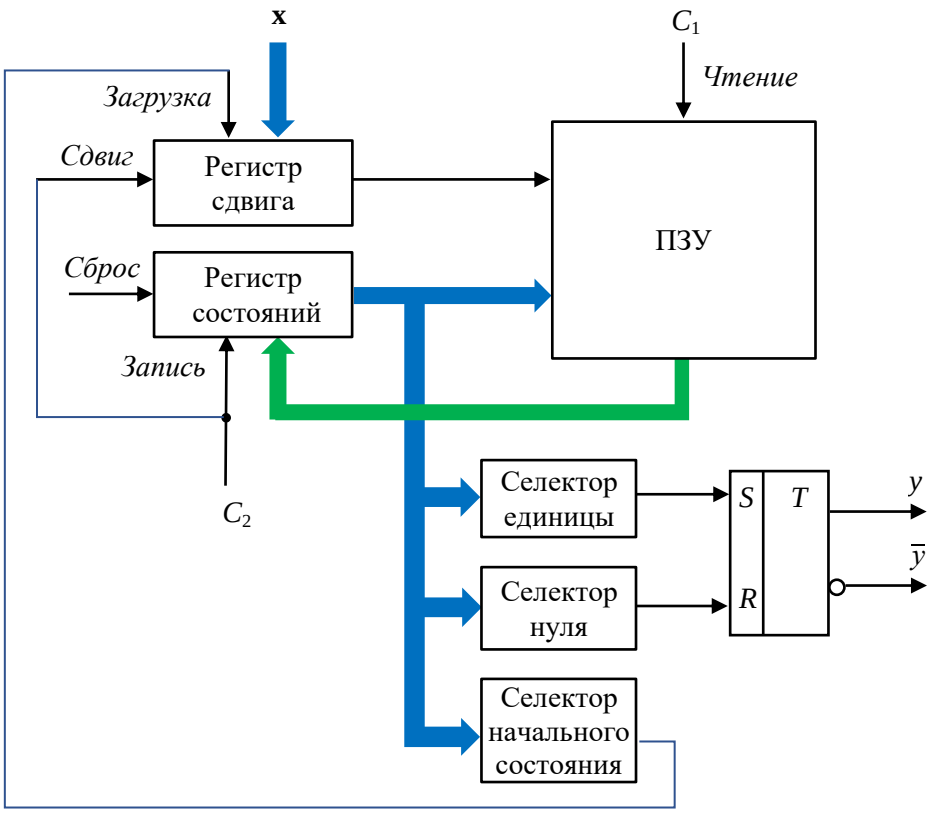


Рис. 1. Обобщенная структура устройства, реализующего булеву функцию

Fig. 1. Generalized structure of the device implementing the Boolean function

Основным компонентом устройства является ПЗУ, которое имеет v входов и w выходов. ПЗУ обладает двухуровневой структурой, которая включает в себя:

– полный дешифратор, преобразующий v-разрядный двоичный код, который поступает на его входы, в 2^v-разрядный унитарный код, в котором только один разряд может иметь значение единицы, а все остальные – нуля;

– программируемую матричную память, организованную в виде 2^v w-разрядных двоичных слов.

При подаче на входы ПЗУ v-разрядного двоичного кода в матричной памяти посредством дешифратора выбирается соответствующее w-разрядное слово, которое передается на выходы ПЗУ. Все входы ПЗУ разбиты на две неравные части: одна часть включает в себя лишь один вход, другая – остальные v–1 входов, объединенных для удобства графического изображения в параллельную шину. Также в параллельную шину объединены и w выходов, где $w = v$–1. Информация на выходах ПЗУ обновляется (в соответствии с адресуемым словом) передним фронтом тактового импульса C_1 в момент поступления его на управляющий вход «Чтение».

В n-разрядный регистр сдвига, входы которого для удобства объединены в шину, записываются значения входных переменных $x_1, x_2, ..., x_n$ передним фронтом сигнала, поступающего на управляющий вход «Загрузка» (в самом младшем разряде будет храниться значение переменной x_1, затем переменной x_2 и т. д.). Эта запись происходит в момент старта устройства и в момент перехода его в начальное состояние. С выхода регистра сдвига снимается значение самого младшего разряда, хранимого в регистре слова. Передним фронтом тактового импульса C_2, поступающего на его управляющий вход «Сдвиг», хранимая в регистре информация сдвигается вправо с потерей значения, содержащегося до этого момента в самом младшем разряде. Таким образом регистр сдвига позволяет поочередно выводить значения переменных $x_1, x_2, ..., x_n$.

Регистр состояний предназначен для хранения двоичного кода текущего состояния устройства и имеет v–1 входов и столько же выходов, которые для удобства объединены в соответствующие шины. Запись информации в регистр выполняется передним фронтом тактового импульса C_2 при его поступлении на управляющий вход «Запись». Для очистки регистра (установки на его выходах уровней логического нуля), необходимой для установки устройства в начальное состояние, используется управляющий вход «Сброс».

Компоненты под названием «Селектор единицы», «Селектор нуля» и «Селектор начального состояния» реализуют элементарные конъюнкции ранга v–1 и устанавливают на своих выходах уровень логической единицы тогда, когда устройство оказывается соответственно в одном или другом финальном состоянии либо в начальном состоянии.

RS-триггер управляется сигналами, формируемыми компонентами «Селектор единицы» и «Селектор нуля», и предназначен для хранения найденного значения функции f в течение интервала времени, необходимого для подсчета обновленного значения.

Для работы устройства необходим также генератор тактовых импульсов C_1, которые в свою очередь порождают инвертированные тактовые импульсы C_2.

Необходимо отметить, что поскольку используется модель инициального автомата, то началу работы устройства должна предшествовать установка его в начальное состояние путем очистки регистра состояний и сброса *RS*-триггера в нулевое состояние, а также загрузки в регистр сдвига значений переменных $x_1, x_2, ..., x_n$.

Автоматная модель устройства. Способ построения автоматной модели основан на использовании разложения Шеннона булевой функции f по переменным, входящим в множество X [5]. Представление вида

$$f(x_1, x_2,\ldots, x_n) = x_i f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) \vee \overline{x}_i f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$$

называется разложением функции f по переменной x_i. Функции $f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$ и $f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$ называются коэффициентами разложения функции f по переменной x_i. Они не зависят от переменной x_i и получаются путем подстановки в исходную функцию констант 1 и 0 соответственно вместо переменной x_i. В свою очередь, каждый из полученных коэффициентов разложения можно далее разложить по другой переменной, например x_j, получая в результате четыре коэффициента разложения функции f по переменным x_i и x_j. Продолжая процесс разложения по оставшимся входным переменным и учитывая, что некоторые из получаемых при этом коэффициентов разложения вырождаются в константы 1 и 0 или оказываются дубликатами, можно построить ориентированный ациклический граф, называемый в литературе бинарной диаграммой решений. В зарубежной литературе используется название binary decision diagram, сокращенно BDD [6–9].

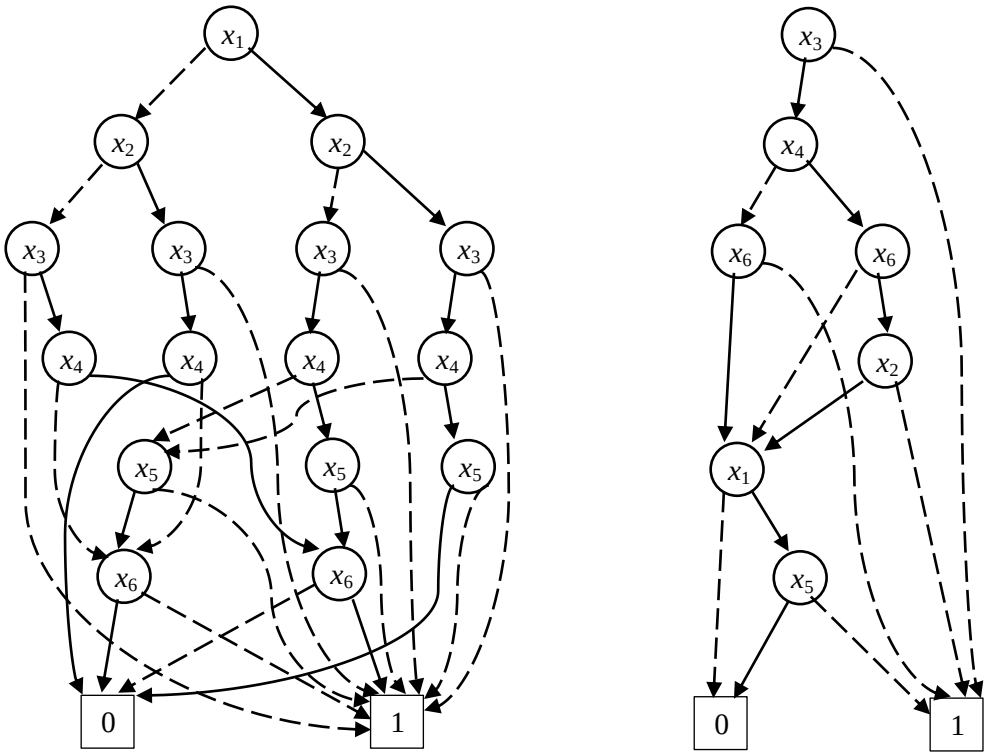
BDD содержит начальную, внутренние и две концевые вершины, которые распределены по уровням. Если переменные для разложения выбирать в порядке увеличения их индекса, то начальная вершина, расположенная на первом уровне, помечается переменной x_1. Внутренние вершины, расположенные на уровнях 2, 3, …, n, помечаются переменными $x_2, x_3, ..., x_n$ соответственно. Концевые вершины, расположенные на уровне n+1, будут помечены константами 0 и 1, соответствующими двум возможным значениям булевой функции f. Каждая из вершин, кроме концевых, связана с двумя вершинами-потомками. Переход к одной из них выполняется в зависимости от значения переменной, которой помечена вершина. Значениям 0 и 1 этой переменной соответствуют переходы, графически отображаемые пунктирной и непрерывной линиями соответственно. Тогда значение, которое принимает функция f на конкретном наборе значений своих переменных $x_1, x_2, ..., x_n$, в BDD однозначно определяется путем из начальной вершины в одну из концевых вершин.

Рассмотрим пример. На рис. 2, *а* изображена BDD для функции

$$f(x_1, x_2, \ldots, x_6) = x_1\overline{x}_5 \vee \overline{x}_2x_4x_6 \vee \overline{x}_3 \vee x_3\overline{x}_4\overline{x}_6,$$

заданной в дизъюнктивной нормальной форме.

Размер BDD, определяемый числом ее вершин, зависит не только от исходной булевой функции, но и от выбора порядка входных переменных, используемых при построении BDD. В общем случае задача выбора порядка входных переменных, приводящая к построению BDD наименьшего размера, сопряжена с громоздкими вычислениями, резко возрастающими с ростом числа аргументов функции f. Имеются работы (например, [10–12]), посвященные решению этой задачи. В настоящей статье ограничимся рассмотрением удобного на практике приближенного варианта решения данной задачи, заключающегося в следующем. На очередном шаге построения BDD выбирается переменная, использование которой приводит к появлению наименьшего числа различающихся коэффициентов разложения, отличных от нуля и единицы. Если таких вариантов несколько либо их вообще нет, то выбирается вариант, у которого больше коэффициентов разложения, вырожденных в одиночную переменную (возможно, с инверсией). Если и здесь вариантов несколько либо они отсутствуют, то выбирается вариант с наименьшим суммарным числом литералов, входящих в элементарные конъюнкции дизъюнктивной нормальной формы всех коэффициентов разложения. Применение такого способа выбора переменных при построении BDD часто позволяет значительно уменьшить ее размер (рис. 2, *b*).



a) *b)*

Рис. 2. BDD: *a)* порядок разложения $x_1 \to x_2 \to \ldots \to x_6$; *b)* порядок разложения $x_3 \to x_4 \to x_6 \to x_2 \to x_1 \to x_5$

Fig. 2. BDD: a) decomposition order: $x_1 \to x_2 \to \ldots \to x_6$; *b) decomposition order* $x_3 \to x_4 \to x_6 \to x_2 \to x_1 \to x_5$

После того как для булевой функции f построена BDD, ее можно достаточно просто отобразить в граф поведения инициального автомата Мура с абстрактным состоянием, который и будет служить функциональной моделью устройства рассматриваемого здесь типа. Автомат может находиться в различных внутренних состояниях, образующих множество $Q = \{q_1, q_2, \ldots, q_m\}$, где m – общее число вершин в BDD. Начальным состоянием служит состояние q_1. Финальные состояния, соответствующие концевым вершинам BDD, дополняются дугами для безусловного перехода в начальное состояние, поскольку в отличие от BDD значение функции f будет вычисляться многократно. Например, BDD, изображенной на рис. 2, *b*, соответствует граф поведения автомата Мура с абстрактным состоянием, который показан на рис. 3. Финальные состояния изображены в форме эллипсов.

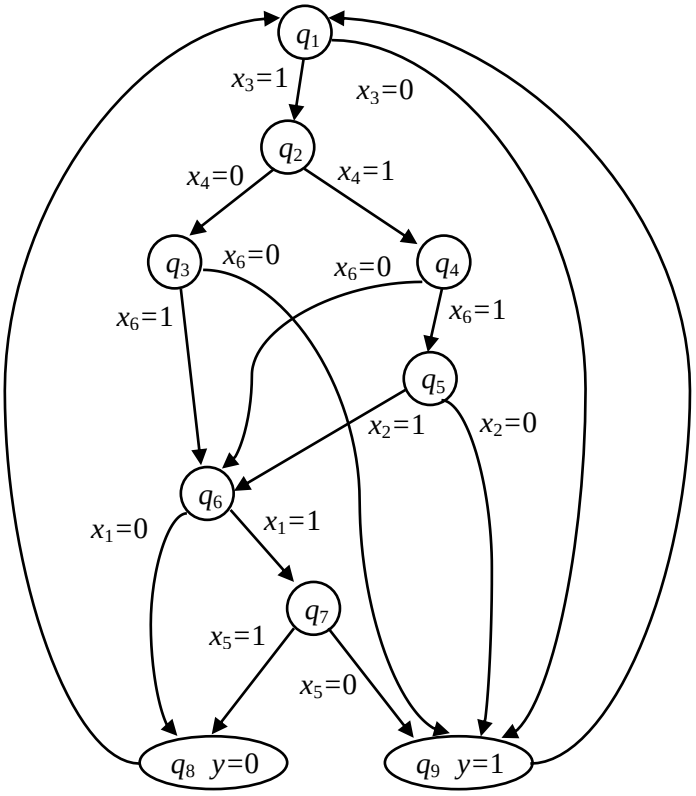


Рис. 3. Граф поведения автомата Мура с абстрактным состоянием
Fig. 3. A graph of the behavior of a Moore automaton with an abstract state

Автомат Мура с абстрактным состоянием можно задать также с помощью таблицы переходов (табл. 1), каждая строка которой определяет состояние q^+, в которое переходит автомат из состояния q, если на его вход подано значение соответствующей входной переменной. Для автомата Мура таблица переходов дополняется столбцом, задающим значения функции f. Особенностью этой таблицы является то, что символ «-», используемый обычно при задании интервалов булева пространства входных переменных, означает еще и то, что значение соответствующей переменной не принимается во внимание, поскольку не оказывает никакого влияния на функционирование автомата. В частности, интервал - - - - - - указывает, что из состояний q_8 и q_9 переход выполняется всегда в состояние q_1 независимо от значений входных переменных. Символ «*», используемый при задании значения функции f, означает, что переход в данное состояние не меняет значение функции, т. е. она сохранит то же значение, которое было и в предыдущем состоянии.

Чтобы перейти к технической реализации устройства рассматриваемого типа, необходимо сначала выполнить кодирование состояний автомата путем замены многозначной внутренней переменной q на булев вектор $\mathbf{z} = (z_1, z_2, \ldots, z_w)$. В этом случае автомат с абстрактным состоянием будет преобразован в булев автомат [5, 13]. Синхронная реализация автомата на основе ПЗУ не требует решения задачи противогоночного кодирования и выполнения операций, связанных с упрощением булевых выражений, так как какая-либо иная управляющая логика отсутствует. Поэтому задача кодирования состояний сводится к присваиванию состояниям автомата уникальных w-разрядных двоичных кодов. При этом начальное состояние необходимо всегда кодировать вектором, состоящим из одних нулей, поскольку автомат является инициальным. Требуемое число разрядов $w = \lceil \log_2 m \rceil$, где $\lceil a \rceil$ означает минимальное целое число, не меньшее a. Для простоты будем использовать следующее правило: состояние q_i, $i \in \{1, 2, \ldots, m\}$,

кодируется двоичным кодом, имеющим десятичное значение i–1. Таким образом, состояния q_1, q_2, …, q_m будут закодированы двоичными числами с десятичными эквивалентами 0, 1, …, m–1 соответственно. В рассматриваемом примере w = 4. С учетом этого получаем таблицу переходов булева автомата Мура (табл. 2).

Таблица 1
Таблица переходов автомата Мура с абстрактным состоянием

Table 1
The transition table of the Moore automaton with an abstract state

x	q	q^+	y	**x**	q	q^+	y
- - 0 - - -	q_1	q_9	*	- 0 - - - -	q_5	q_9	*
- - 1 - - -	q_1	q_2	*	- 1 - - - -	q_5	q_6	*
- - - 0 - -	q_2	q_3	*	0 - - - - -	q_6	q_8	*
- - - 1 - -	q_2	q_4	*	1 - - - - -	q_6	q_7	*
- - - - - 0	q_3	q_9	*	- - - - 0 -	q_7	q_9	*
- - - - - 1	q_3	q_6	*	- - - - 1 -	q_7	q_8	*
- - - - - 0	q_4	q_6	*	- - - - - -	q_8	q_1	0
- - - - - 1	q_4	q_5	*	- - - - - -	q_9	q_1	1

Таблица 2
Таблица переходов булева автомата Мура

Table 2
Transition table of the Boolean Moore automaton

x	**z**	$\mathbf{z}^+$	y	**x**	**z**	$\mathbf{z}^+$	y
- - 0 - - -	0 0 0 0	1 0 0 0	*	- 0 - - - -	0 1 0 0	1 0 0 0	*
- - 1 - - -	0 0 0 0	0 0 0 1	*	- 1 - - - -	0 1 0 0	0 1 0 1	*
- - - 0 - -	0 0 0 1	0 0 1 0	*	0 - - - - -	0 1 0 1	0 1 1 1	*
- - - 1 - -	0 0 0 1	0 0 1 1	*	1 - - - - -	0 1 0 1	0 1 1 0	*
- - - - - 0	0 0 1 0	1 0 0 0	*	- - - - 0 -	0 1 1 0	1 0 0 0	*
- - - - - 1	0 0 1 0	0 1 0 1	*	- - - - 1 -	0 1 1 0	0 1 1 1	*
- - - - - 0	0 0 1 1	0 1 0 1	*	- - - - - -	0 1 1 1	0 0 0 0	0
- - - - - 1	0 0 1 1	0 1 0 0	*	- - - - - -	1 0 0 0	0 0 0 0	1

Прошивка памяти ПЗУ. Обозначим входы и выходы ПЗУ через $A_0, A_1, \ldots, A_{v\text{-}1}$ и $Q_1, Q_2, \ldots, Q_w$ соответственно. При подаче на входы ПЗУ v-разрядного кода в матричной памяти посредством дешифратора выбирается соответствующее w-разрядное слово, которое передается на выходы ПЗУ.

Если для адресации используются входы $A_1, A_2, \ldots, A_{v-1}$, то в матричной памяти будет выбрано одно из двух слов, адреса которых различаются лишь значением самого младшего разряда, связанного с адресным входом A_0. Значение сигнала, поступающего на этот вход, определит, какое из двух слов будет выбрано. Адресный вход A_0 связан с выходом сдвигового регистра, в который предварительно записываются значения входных переменных из множества X в порядке, обеспечивающем наименьший размер BDD. Адресные входы $A_1, A_2, \ldots, A_{v-1}$ связаны с соответствующими выходами регистра состояний, хранящего код текущего внутреннего состояния устройства. С учетом этого можно сделать вывод, что прошивка памяти ПЗУ должна выполняться согласно таблице переходов булева автомата. Тогда каждое из двух слов, соответствующих конкретному внутреннему состоянию, будет хранить код следующего состояния, т. е. состояния, в которое перейдет устройство в зависимости от текущего значения сигнала на входе A_0. На рис. 4 показана прошивка памяти ПЗУ, обеспечивающая техническую реализацию булева автомата, заданного табл. 2.

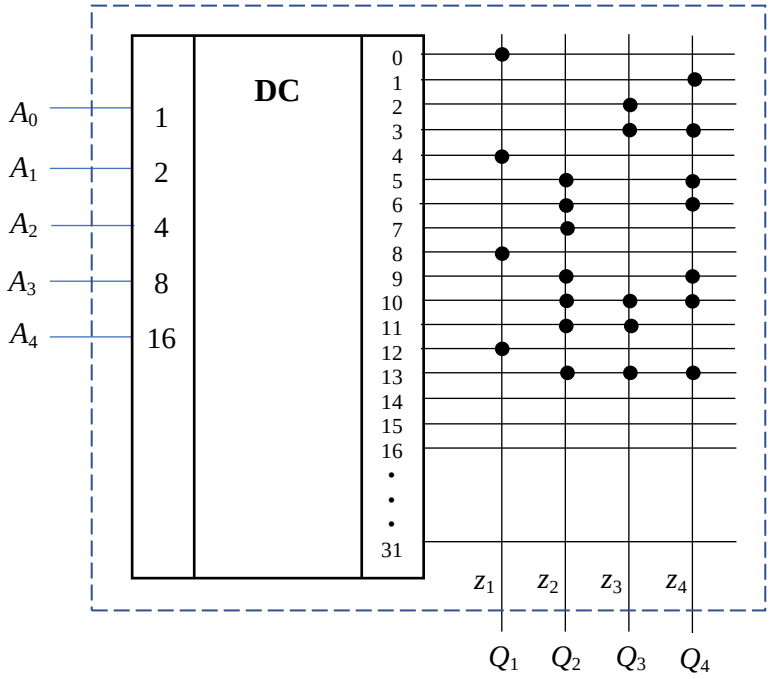


Рис. 4. Прошивка памяти ПЗУ

Fig. 4. ROM Memory Firmware

Заключение. Устройства, реализованные на основе автоматной модели (как и проектируемые на базе микроконтроллеров), могут найти применение в самых разнообразных изделиях, прежде всего предметах бытового назначения, не требующих обеспечения высоких показателей по быстродействию и времени реакции. В отношении этих характеристик данные устройства значительно уступают комбинационным схемам, синтезированным из традиционных логических элементов И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ и др., например из программируемых логических матриц [5, 13] или LUT-компонентов [14], представляющих собой разновидность программируемых логических интегральных схем комбинационного типа. С другой стороны, преимуществом первых является возможность изменения их функциональности при сохранении той же самой схемной реализации. Для этого в устройстве необходимо использовать не однократно, а многократно программируемое ПЗУ, которое путем перепрошивки своей матричной памяти позволит реализовать любую другую булеву функцию при следующих условиях:

1) число ее аргументов не будет превосходить разрядность регистра сдвига;

2) размер BDD, оптимизированной по числу вершин, не будет превышать 2^w;

3) финальные состояния булева автомата сохранят свои прежние кодировки (чтобы избежать внесения изменений в компоненты «Селектор единицы» и «Селектор нуля»).

Первое условие можно не учитывать, если изначально использовать сдвиговый регистр большей разрядности, резервируя при этом старшие разряды для возможных изменений функциональности устройства в будущем.

Рассматриваемое устройство в некоторых случаях можно упростить, удалив регистр состояний и соединив выход ПЗУ с его входом, а также с компонентами «Селектор единицы», «Селектор нуля» и «Селектор начального состояния». В этом случае дискретный автомат, моделирующий работу устройства, перестает быть инициальным. В свою очередь это может привести к тому, что самый первый переход устройства в одно из финальных состояний может быть ошибочным, так как устройство начнет работу из какого-то состояния, которое невозможно заранее предсказать. Дальнейшее же функционирование устройства будет правильным. Поэтому, если такой «начальный сбой» допустим, указанное упрощение устройства можно выполнять.

Устройства, реализованные на базе микроконтроллеров, по сравнению с описываемым в статье устройством оказываются менее быстродействующими. Особенно это касается тех из них, которые программируются не на ассемблере, а на языке высокого уровня, так как после компиляции образуются довольно длинные последовательности машинных команд процессора. С другой стороны, использование микроконтроллеров позволяет реализовать значительно более сложные алгоритмы управления, чем те, которые описываются булевыми функциями.

Следует отметить, что предложенный способ реализации булевых функций опирается на классическую автоматную модель, хотя тот же конечный результат можно получить, основываясь на моделях микропрограммного или секвенциального автоматов [5, 13].

Список использованных источников

1. Белов, А. В. Программирование микроконтроллеров для начинающих и не только… / А. В. Белов. – СПб. : Наука и техника, 2016. – 352 с.

2. Кузьмин, В. А. Реализация функций алгебры логики автоматами, нормальными алгоритмами и машинами Тьюринга / В. А. Кузьмин // Проблемы кибернетики. – 1965. – Вып. 13. – С. 75–96.

3. Хопкрофт, Д. Введение в теорию автоматов, языков и вычислений / Д. Хопкрофт, Р. Мотвани, Дж. Ульман. – 2-е изд., пер. с англ. – М. : Изд. дом «Вильямс», 2008. – 528 с.

4. Шалыто, А. А. Логическое управление. Методы аппаратной и программной реализации алгоритмов / А. А. Шалыто. – СПб. : Наука, 2000. – 780 с.

5. Закревский, А. Д. Логические основы проектирования дискретных устройств / А. Д. Закревский, Ю. В. Поттосин, Л. Д. Черемисинова. – М. : Физматлит, 2007. – 592 с.

6. Andersen, H. R. An Introduction to Binary Decision Diagrams. Lecture Notes / H. R. Andersen. – Copenhagen : IT University of Copenhagen, 1999. – 35 p.

7. Akers, S. B. Binary decision diagrams / S. B. Akers // IEEE Transactions on Computers. – 1978. – Vol. C-27, no. 6. – P. 509–516.

8. Bryant, R. E. Ordered binary decision diagrams / R. E. Bryant, C. Meinel // Logic Synthesis and Verification / eds.: S. Hassoun, T. Sasao, R. K. Brayton. – Kluwer Academic Publishers, 2002. – P. 285–307.

9. Yang, S. BDS: a BDD-based logic optimization system / S. Yang, M. Ciesielski // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. – 2002. – Vol. 21, no. 7. – P. 866–876.

10. Бибило, П. Н. Применение диаграмм двоичного выбора при синтезе логических схем / П. Н. Бибило. – Минск : Беларус. навука, 2014. – 231 с.

11. Meinel, C. Algorithms and Data Structures in VLSI Design: OBDD – Foundations and Applications / C. Meinel, T. Theobald. – Berlin, Heidelberg : Springer-Verlag, 1998. – 267 p.

12. Ebendt, R. Advanced BDD Optimization / R. Ebendt, G. Fey, R. Drechsler. – Springer, 2005. – 222 p.

13. Закревский, А. Д. Логический синтез каскадных схем / А. Д. Закревский. – М. : Наука, 1981. – 416 с.

14. Vemuri, N. BDD-based logic synthesis for LUT-6-based FPGAs / N. Vemuri, P. Kalla, R. Tessier // ACM Transactions on Design Automation of Electronic Systems. – 2002. – Vol. 7, no. 4. – P. 501–525.

References

1. Belov A. V. Programmirovanie mikrokontrollerov dlya nachinayushchih i ne tol'ko… *Microcontroller Programming for Beginners and Beyond*… Saint-Petersburg, Nauka i tekhnika, 2016, 352 p. (In Russ.).

2. Kuz'min V. A. *Realization of functions of the algebra of logic by automata, normal algorithms and Turing machines*. Problemy kibernetiki [*Cybernetics Issues*], 1965, vol. 13, pp. 75–96 (In Russ.).

3. Hopcroft J. E., Motwani R., Ullman J. D. *Introduction to Automata, Theory, Languages and Computation*, 2nd edition. Addison-Wesley, 2000, 521 p.

4. Shalyto A. A. Logicheskoe upravlenie. Metody apparatnoj i programmnoj realizacii algoritmov. *Logical Control. Methods of Hardware and Software Implementation of Algorithms.* Saint-Petersburg, Nauka, 2000, 780 p. (In Russ.).

5. Zakrevskij A. D., Pottosin Yu. V., Cheremisinova L. D. Logicheskie osnovy proektirovanija diskretnyh ustrojstv. *Logical Fundamentals of Discrete Devices Design*. Moscow, Fizmatlit, 2007, 592 p. (In Russ.).

6. Andersen H. R. *An Introduction to Binary Decision Diagrams. Lecture Notes*. Copenhagen, IT University of Copenhagen, 1999, 35 p.

7. Akers S. B. Binary decision diagrams. *IEEE Transactions on Computers*, 1978, vol. C-27, no. 6, pp. 509–516.

8. Bryant R. E., Meinel C. Ordered binary decision diagrams. *Logic Synthesis and Verification*. In S. Hassoun, T. Sasao, R. K. Brayton (eds.). Kluwer Academic Publishers, 2002, pp. 285–307.

9. Yang S., Ciesielski M. BDS: a BDD-based logic optimization system. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2002, vol. 21, no. 7, pp. 866–876.

10. Bibilo P. N. Primenenie diagram dvoichnogo vybora pri sinteze logicheskih shem. *Application of Binary Selection Diagrams in the Synthesis of Logic Circuits*. Minsk, Belaruskaja navuka, 2014, 231 p. (In Russ.).

11. Meinel C., Theobald T. *Algorithms and Data Structures in VLSI Design: OBDD – Foundations and Applications.* Berlin, Heidelberg, Springer-Verlag, 1998, 267 p.

12. Ebendt R., Fey G., Drechsler R. *Advanced BDD Optimization.* Springer, 2005, 222 p.

13. Zakrevskij A. D. Logicheskij sintez kaskadnyh skhem. *Logical Synthesis of Cascading Circuit*. Moscow, Nauka, 1981, 416 p. (In Russ.).

14. Vemuri N., Kalla P., Tessier R. BDD-based logic synthesis for LUT-6-based FPGAs. *ACM Transactions on Design Automation of Electronic Systems*, 2002, vol. 7, no. 4, pp. 501–525.

Информация об авторе

Бутов Алексей Александрович, кандидат технических наук, доцент, Белорусский государственный университет информатики и радиоэлектроники.
E-mail: tmkrb9@gmail.com

Information about the author

Alexey A. Butov, Ph. D. (Eng.), Assoc. Prof., Belarusian state university of informatics and radioelectronics.
E-mail: tmkrb9@gmail.com