Практика математического синтеза микропрограммных управляющих автоматов на основе ПЗУ и ПЛМ

Виктор САФРОНОВ,
к. т. н.
vik.saf@yandex.ru

В статье приведены практические приемы математического синтеза схем микропрограммных управляющих автоматов на основе ПЗУ и ПЛМ с помощью линейных алгоритмов Ляпунова и карт Карно-Вейча.

Введение

В журнале «Компоненты и технологии» была опубликована статья [1], в которой авторы описали процесс синтеза микропрограммных управляющих автоматов на базе ПЗУ и синхронных D-регистров. Затронутая тема актуальна, потому что именно этот синтез в свое время привел к поступательному развитию сначала программируемых микросхем ПЛМ, PLA, PLD, а затем GAL и FPGA (ПЛИС).

Теперь разработчики столкнулись с другой крайностью: стали справедливыми их нарекания на то, что в некоторых относительно простых практических случаях применять FPGA неоправданно дорого для формирования временных диаграмм управляющих сигналов. Частичным решением проблемы является использование популярных микропроцессоров семейств AVR и PIC, но при условии, что разработчик умеет программировать микропроцессоры и у него для этого есть соответствующее, к сожалению, тоже не дешевое оборудование. Собирать управляющую схему на «жесткой логике», то есть на логических микросхемах малой степени интеграции (МИС), тем более трудоемкое занятие, особенно если в ходе работ приходится многократно перепаивать схему, экспериментируя с алгоритмом управления.

Применение микросхем ПЗУ или ПЛМ может стать компромиссом между трудностью сборки схемы на МИС и гибкими возможностями управляющего автомата на микропроцессорах или FPGA. Поэтому необходимо, хотя бы кратко, привести очень важные приемы математического синтеза цифровых управляющих автоматов Уилкса на основе:

- мультиплексоров;
- ПЗУ;
- ПЛМ.

Об элементной базе и терминологии

ПЗУ (ROM) — микросхема с параллельным адресуемым доступом к массиву логических данных, представляемых, как правило, в виде полубайта или байта на выходах микросхемы.

ПЛМ (PAL, PLD) — для нашего случая несравнимо более «интеллектуальная» микросхема, чем ПЗУ, так как она позволяет программными средствами по логическим уравнениям реализовать соответствующие аппаратные цепочки. Как говорят, с ее помощью можно собирать электрические схемы без паяльника.

GAL — следующий шаг совершенствования микросхемы ПЛМ. Добавление в структуру упомянутых выше синхронных D-регистров делает эти микросхемы самодостаточными, что позволяет строить схемы синхронных последовательностных автоматов не только без паяльника, но и без использования дополнительных внешних регистров.

Общая структура микропрограммного автомата на ПЗУ изображена на рис. 1. Текущие и следующие коды микрокоманд, синхронизируемые фронтом сигнала C, циркулируют в кольце (показано красным цветом) прямых и обратных связей. Расшифруем обозначения на рис. 1:

- RG — синхронный D-регистр с разрядностью K1;
- L — логический блок с количеством входов K2 (его может не быть);
- ROM — ПЗУ с количеством входов K1 и количеством выходов K1, K4, K5;
- RS — сборка асинхронных RS-триггеров с количеством входов K5 (их может не быть);
- C — вход регистра и синхроимпульсы от тактового генератора (например, с активным положительным фронтом, в зависимости от типа регистра);
- R — вход регистра для установки его в «нулевое» состояние;
- M — текущая микрокоманда (двоичный код с разрядностью K1);
- M^+ — следующая микрокоманда (двоичный код с разрядностью K1).

Общеприняты следующие термины и понятия (рис. 1):

- Микротактом называют период времени между соседними синхронизирующими импульсами тактового генератора.

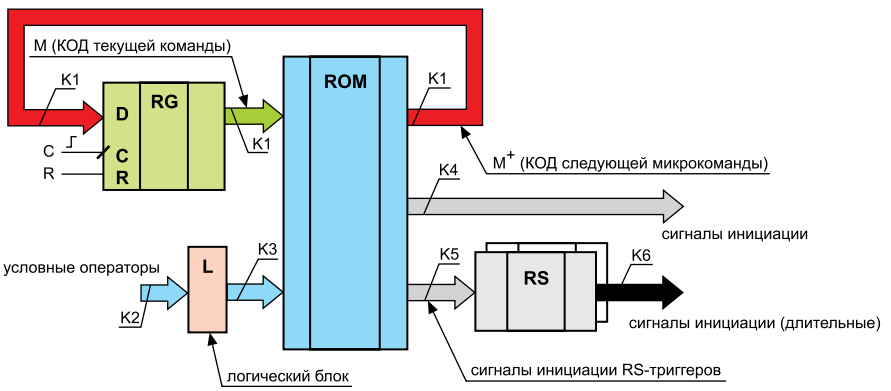

Рис. 1. Общая структура микропрограммного синхронного автомата

- Микропрограммным автоматом называют электронную схему, исполняющую сколь угодно длинные последовательности микрокоманд (микрокоманда за микрокомандой).
- Микрокомандой называют группу операторов, формирующих на выходах автомата сигналы инициации внешних устройств с учетом значений условных операторов на его входах. Каждая микрокоманда имеет поля: K2 разрядов условных операторов, K1-разрядного кода текущей микрокоманды, K1-разрядного кода следующей микрокоманды и (K4+K6) выходов сигналов инициации внешних устройств. Все операторы микрокоманды исполняются в течение только одного микротакта. (Это очень важно помнить!)
- Условными операторами называют логические сигналы на входе логического блока l или непосредственно на входах ПЗУ. Они изменяют последовательный порядок исполнения микрокоманд.
- Сигналами инициации (или операторами) называют выходные сигналы, воздействующие на внешние устройства. Их продолжительность составляет только один микротакт: от микрокоманды до следующей микрокоманды. (Это очень важно помнить!)
- Микропрограммой называют последовательность микрокоманд, выполняющих законченное действие над внешним устройством.
- Командой (уровень ассемблера в микропроцессорах) называют последовательность микропрограмм, выполняющих замысел разработчика. На выполнение одной команды требуется столько микротактов, сколько в ней содержится микрокоманд. Из команд слагают программу.

Пример синтеза без мультиплексирования входов ПЗУ

Рассмотрим синтез микропрограммного автомата на примере схемы управления силовым реверсивным Н-мостом, условно изображенным на рис. 2, и с электродвигателем постоянного тока, включенным в его диагональ. На рис. 2 приняты следующие обозначения:

- Z = 1 — условно положительное направление вращения электродвигателя;
- Z = 0 — условно отрицательное направление вращения электродвигателя;
- τ — условный оператор для формирования ШИМ;
- *stop* — условный оператор выключения двигателя;
- *current* — логический сигнал, равный «1» при превышении допустимого тока от датчика тока;
- *direction* (*dir*) — условный оператор направления вращения;
- ω — безусловный оператор перехода;
- • — точка, безусловный оператор «конец алгоритма».

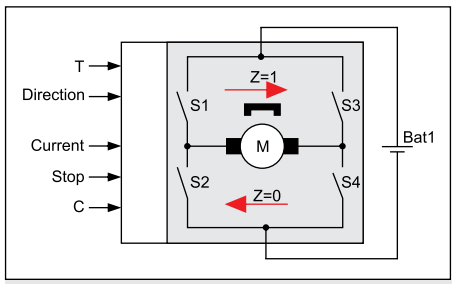

Рис. 2. Управление Н-мостом

Правила корректного управления электрической схемой полного Н-моста известны всем разработчикам следящих приводов и выражаются в четырех действиях:

1. Замыкая только нижние S2 и S4 ключи (или только верхние S1 и S3, безразлично), заставляем работать электродвигатель в режиме динамического торможения его вала. Это значительно уменьшает длительность переходного процесса. Режим динамического торможения можно использовать также для рекуперации кинетической энергии вращения ротора двигателя в источник питания.
2. В каждом периоде ШИМ уровнем сигнала, равным «лог. 1», включаем ключи в нужной диагонали моста. Уровнем «лог. 0» не размыкаем все ключи моста (!), а переводим мост в конфигурацию по п. 1.
3. Для предотвращения пробоя транзисторов реверсируем мост в три этапа:
   – Выключаем ключи работающей диагонали.
   – Переводим мост в конфигурацию по п. 1, причем длительность удержания в этой конфигурации должна быть не меньше удвоенного времени переключения, указанного в паспорте любого из транзисторов моста. Эта же длительность ограничивает выбор минимального периода колебаний тактового генератора.
   – Включаем ключи другой диагонали.
4. Нельзя, даже кратковременно, включать оба ключа одного плеча: это может вызвать короткое замыкание в источнике питания и вывести из строя транзисторы.

Инициирующие выходные сигналы должны формироваться RS-триггерами. В отличие от операторов, существующих в течение только одного микротакта, они должны продолжать действие и после окончания микротакта, пока не будут модифицированы следующей микрокомандой.

Далее при написании алгоритмов приняты следующие обозначения для R/S-сигналов инициализации RS-триггеров:

- ВыкS1/ВкS1 — выключить/включить верхний левый ключ моста;
- ВыкS2/ВкS2 — выключить/включить нижний левый ключ моста;
- ВыкS3/ВкS3 — выключить/включить верхний правый ключ моста;
- ВыкS4/ВкS4 — выключить/включить нижний правый ключ моста.

Построение микрокоманд алгоритма функционирования

Запишем линейный алгоритм Ляпунова. Напоминаем: если в алгоритме условный оператор ложен, то он заставляет немедленно перейти к оператору по ссылке, то есть по направлению от восходящей нумерованной стрелки к одноименно нумерованной нисходящей стрелке. В противном случае выполняется оператор, следующий далее по порядку. Безусловный оператор ω всегда вызывает переход по ссылке, указанной над его стрелкой.

Тогда получим следующую запись (эти пять подстрок следует считать одной слитной строкой):

$\downarrow^{1}$ВыкS1–ВыкS3–ВкS2–ВкS4–*stop*$\uparrow^{6}\omega\uparrow^{1}$ — набор операторов, реализующих динамическое торможение;
$\downarrow^{6}\tau\uparrow^{1}$*dir*$\uparrow^{4}\omega\uparrow^{5}$ — набор операторов, реализующих анализ условных операторов;
$\downarrow^{5}$ВыкS2–ВыкS3–ВкS1–ВкS4–*current*$\uparrow^{6}\omega\uparrow^{7}$ — набор операторов, реализующих управление вращением «по часовой стрелке»;
$\downarrow^{4}$ВыкS1–ВыкS4–ВкS2–ВкS3–*current*$\uparrow^{6}\omega\uparrow^{7}$ — набор операторов, реализующих управление вращением «против часовой стрелки»;
$\downarrow^{7}$ВыкS1–ВыкS3–ВкS2–ВкS4• — набор операторов, реализующих аварийное отключение и остановку алгоритма.

Затем разбиваем алгоритм на части, отделяя противоречивые [2] операторы друг от друга, и группируем непротиворечивые операторы в микрокоманды (они обозначены *Mi*), не забывая, что все операторы в каждой микрокоманде должны выполняться одновременно, то есть за один микротакт. Получим 11 микрокоманд, содержащих группы операторов:

*M*1: ВыкS1–ВыкS3→*M*2,
*M*2: ВкS2–ВкS4→*M*3,
*M*3: *stop*$\uparrow^{M4}$→*M*1,
*M*4: τ$\uparrow^{M1}$→*M*5,
*M*5: *dir*$\uparrow^{M10}$→*M*6,
*M*6: ВыкS2–ВыкS3→*M*7,
*M*7: ВкS1–ВкS4–*current*$\uparrow^{M4}$→*M*8,
*M*8: ВыкS1–ВыкS3→*M*9,
*M*9: ВкS2–ВкS4→*M*10,
*M*10: ВыкS1–ВыкS4→*M*11,
*M*11: ВкS3–ВкS2–*current*$\uparrow^{M4}$→*M*8.

Количество входов у микросхем ПЗУ, как правило, небольшое. Их может не хватить для построения проекта. Во втором примере будет показано, как можно их «экономить». Только по этой причине операторы *stop*, τ и *dir* введены в отдельные микрокоманды *M*3, *M*4 и *M*5 для реализации идеи мультиплексирования (совмещения функций) входов ПЗУ. Тогда в схему нужно вводить логический блок (блок L на рис. 1), пока же на это можно не обращать внимание.

Этап кодирования микрокоманд

Следующий этап синтеза заключается в кодировании микрокоманд с помощью карты Карно-Вейча. Этот этап до сих пор в теории синтеза автоматов не формализован. Не опасаясь критических гонок в синхронном автомате, после нескольких проб и ошибок получаем карту размещения микрокоманд (рис. 3). Микрокоманды можно разместить на карте и иначе, но такое размещение позволяет закодировать все микрокоманды «соседними» кодами, за исключением пар *M*2–*M*3 и *M*3–*M*4, отмеченных красным цветом (рис. 3). При прочих равных условиях показанное четырехразрядное кодирование позволяет исключить некритичные комбинационные гонки и, значит, дополнительно увеличить быстродействие автомата.

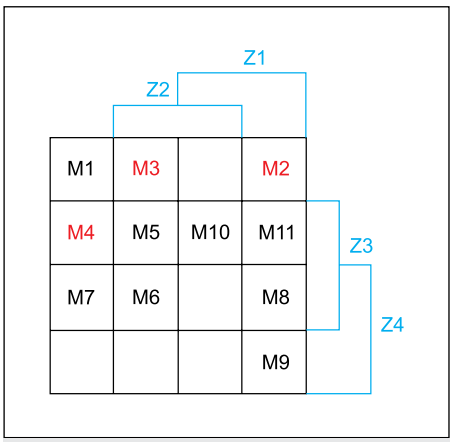

Рис. 3. Карта Карно-Вейча кодирования микрокоманд алгоритма. (Стрелками показаны переходы между полученными 11 микрокомандами)

	ПЕРЕХОДЫ		УСЛОВНЫЕ ОПЕРАТОРЫ				КОД ТЕКУЩЕЙ МК				КОД СЛЕДУЮЩЕЙ МК				ВЫХОД ИНИЦИАЦИИ							
	M	M+	T	Dir	Stop	Car	Z1	Z2	Z3	Z4	Z1+	Z2+	Z3+	Z4+	S1	R1	S2	R2	S3	R3	S4	R4
1	M1	M2	X	X	X	X	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0
2	M2	M3	X	X	X	X	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0
3	M3	M4	X	X	0	X	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
4	M3	M1	X	X	1	X	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
5	M4	M1	0	X	X	X	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
6	M4	M5	1	X	X	X	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0
7	M5	M10	X	0	X	X	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0
8	M5	M6	X	1	X	X	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0
9	M6	M7	X	X	X	X	0	1	1	1	0	0	1	1	0	0	0	1	0	1	0	0
10	M7	M4	X	X	X	0	0	0	1	1	0	0	1	0	1	0	0	0	0	0	1	0
11	M7	M8	X	X	X	1	0	0	1	1	1	0	0	1	1	0	0	0	0	0	1	0
12	M8	M9	X	X	X	X	1	0	1	1	1	0	0	1	0	1	0	0	0	1	0	0
13	M9	M9	X	X	X	X	1	0	0	1	1	0	0	1	0	0	1	0	0	0	1	0
14	M10	M11	X	X	X	X	1	1	1	0	1	0	1	0	0	1	0	0	0	0	0	1
15	M11	M8	X	X	X	1	1	0	1	0	1	0	1	1	0	0	1	0	1	0	0	0
16	M11	M4	X	X	X	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0

Рис. 4. Таблица полей микрокоманд (для составления массива «прошивки» ПЗУ)

Для микрокоманды *M*1 выбран код 0000, поэтому сброс в «лог. 0» D-регистра при включении питания сразу же приводит алгоритм к началу его работы.

Осталось нарисовать таблицу полей для микрокоманд, их 16 видов (рис. 4). Переписывая многократно строки таблицы с заменой в них символов «X» на «1» и «0» всеми возможными комбинациями, получим в нашем примере не более $16 \times 2^4 = 256$ строк «таблицы прошивки». Теперь, принимая восьмиразрядные двоичные числа в полях «условные операторы» и «код текущей МК» в качестве адресов ячеек ПЗУ, «прошиваем» данные (правая часть таблицы серого цвета).

Недостаток схемы на рис. 5 в том, что приходится использовать две микросхемы ПЗУ из-за большого суммарного количества выходов сигналов управления и разрядов кода микрокоманды. Зато оказавшиеся «лишними» три адресных входа ПЗУ (A8, A9, A10) позволяют записать еще $2^3 = 8$ таких же по размеру зон, например для прошивки других вариантов программ. Переключая логические уровни на входах A8, A9, A10, можно реализовать даже мультипрограммный режим работы автомата или использовать эти зоны для многократного резервирования программы.

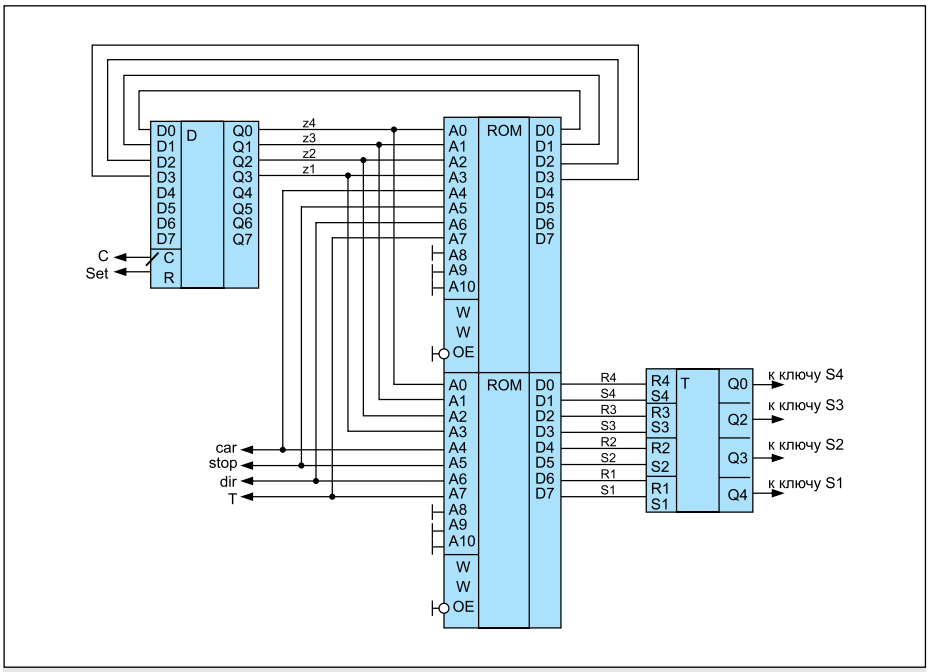

Рис. 5. Синтезированная схема микропрограммного автомата на основе ПЗУ

Использование мультиплексора на входах ПЗУ

Внимательный читатель заметил, какое большое количество строк (256) в ПЗУ понадобилось для решения не очень сложной задачи. Это отрицательное свойство схем, синтезированных на базе ПЗУ, но с ним можно бороться, например с помощью логического блока L мультиплексора (рис. 1). Именно для этого в алгоритме каждый из трех условных операторов (*stop*, τ, *dir*) выделен в отдельную микрокоманду. Очевидно, что использование мультиплексора высвободит (4–1 = 3) три адресных входа ПЗУ, уменьшит число строк до $2^5 = 32$ и позволит выбрать для проекта гораздо более простые и дешевые микросхемы ПЗУ.

Воспользовавшись этой идеей, перерисуем схему (рис. 6), принимая коды в полях условий таблицы (рис. 4) в качестве адресов для мультиплексора.

Пример синтеза с использованием ПЛМ

В схеме на рис. 6, к сожалению, опять пришлось использовать две микросхемы ПЗУ. Устраним и этот недостаток автомата, применив ПЛМ вместо ПЗУ. Опять используем полученные результаты синтеза, чтобы написать таблицу переходов и формулы «прошивок» для ПЛМ. Этой таблице (рис. 7) соответствует следующая система логических уравнений для кодирующих переменных:

$$Z1^{+} = \overline{Z1}\times\overline{Z2}\times\overline{Z3}+Z1\times\overline{Z2}\times \times Z4+Z1\times Z2\times\overline{Z4}+ +current\times Z1\times\overline{Z3}\times\overline{Z4}+ +current\times\overline{Z2}\times Z4+\overline{dir}\times Z2\times Z3\times\overline{Z4}$$

$$Z2^{+} = Z1\times\overline{Z3}\times\overline{Z4}+\overline{Z1}\times Z2\times Z3+ +\overline{Z1}\times Z2\times Z3\times\overline{Z4}\times\tau$$

$$Z3^{+} = Z1\times\overline{Z3}\times\overline{Z4}+\overline{stop}\times Z2+ +Z2\times Z3+\tau\times Z3\times\overline{Z4}+Z4\times\overline{Z1}$$

$$Z4^{+} = Z1\times Z4+\overline{Z1}\times Z3\times dir+ +current\times Z1\times\overline{Z2}\times Z3+current\times Z4$$

Каноническое уравнение RS-триггера, как известно, имеет вид:

$$Q^{+} = Q\times\overline{R}+S.$$

Тогда:

- каноническое уравнение RS-триггера для управления ключом S1:

$$Q^{+} = Q\times\overline{M1+M8+M10}+M7;$$

- каноническое уравнение RS-триггера для управления ключом S2:

$$Q^{+} = Q\times\overline{M6}+(M7+M9+M11);$$

- каноническое уравнение RS-триггера для управления ключом S3:

$$Q^{+} = Q\times\overline{M1+M6+M8}+M11;$$

- каноническое уравнение RS-триггера для управления ключом S4:

$$Q^{+} = Q\times\overline{M10}+M2+M7+M9,$$

где

$M1 = \overline{Z1}\times\overline{Z2}\times\overline{Z3}\times\overline{Z4}$; $M2 = Z1\times\overline{Z2}\times\overline{Z3}\times\overline{Z4}$; $M6 = \overline{Z1}\times Z2\times Z3\times Z4$; $M7 = \overline{Z1}\times\overline{Z2}\times Z3\times Z4$; $M8 = Z1\times\overline{Z2}\times Z3\times\overline{Z4}$; $M9 = Z1\times\overline{Z2}\times\overline{Z3}\times Z4$; $M10 = Z1\times Z2\times Z3\times\overline{Z4}$; $M11 = Z1\times\overline{Z2}\times Z3\times\overline{Z4}$.

Получившаяся очень простая схема, соответствующая этим уравнениям, показана на рис. 8.

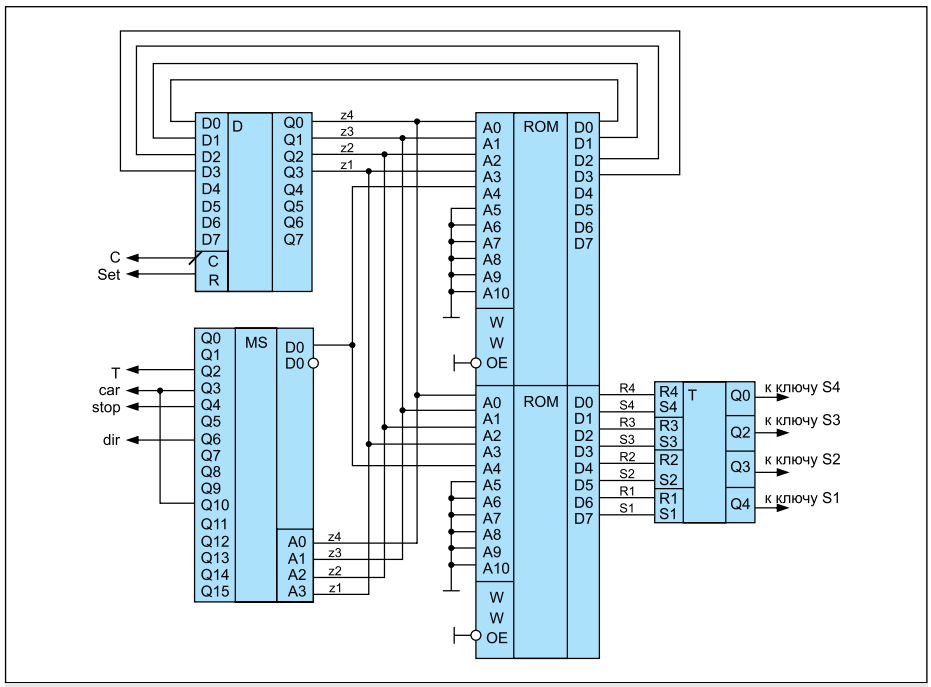

Рис. 6. Схема микропрограммного автомата, приведенная на рис. 5, но с мультиплексором

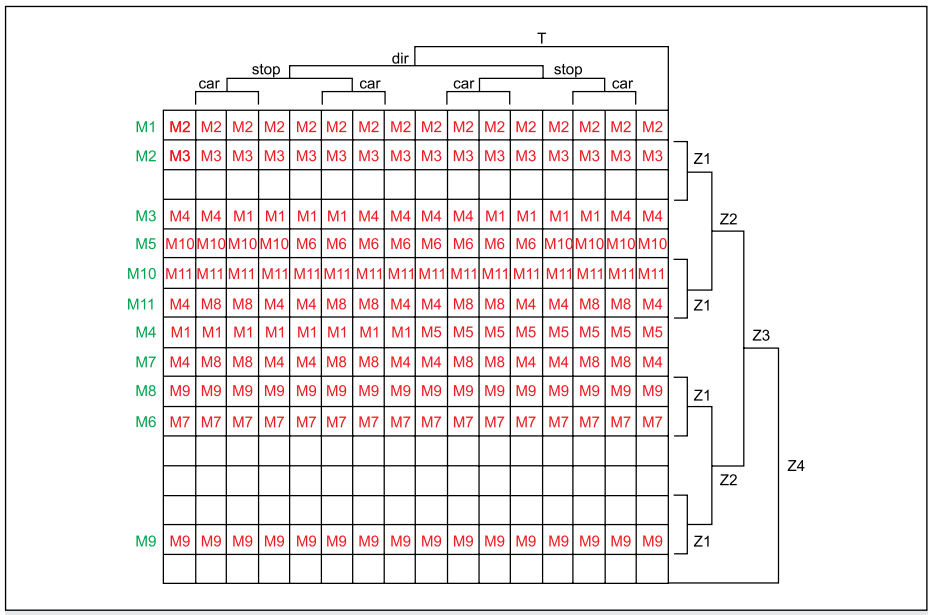

Рис. 7. Таблица переходов

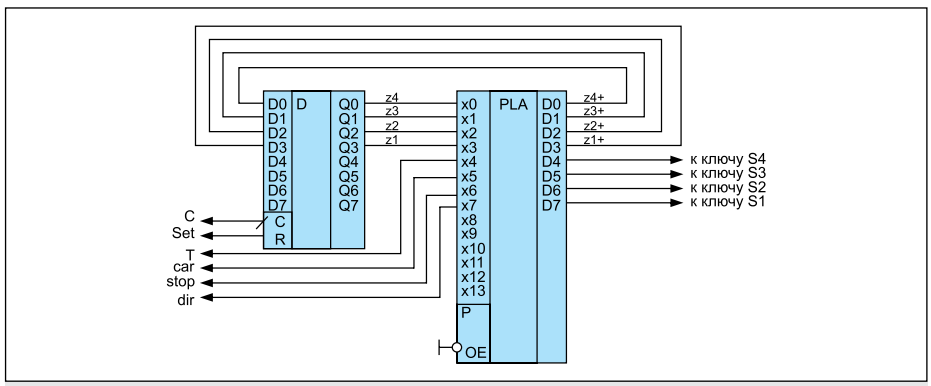

Рис. 8. Схема микропрограммного автомата на базе ПЛМ

Заключение

Применение микросхем ПЗУ или ПЛМ для построения управляющих автоматов в некоторых случаях позволяет получить компактное решение при малых затратах труда, но с такими же гибкими возможностями, как и при использовании микропроцессоров или FPGA.

Методы математического анализа и синтеза синхронных и асинхронных конечных автоматов Мура и Мили по таблицам переходов, картам финальных пар и картам Карно-Вейча исчерпывающе полно и классически корректно изложены в книге [2].

Анализ и синтез микропрограммных управляющих автоматов Уилкса удобно проводить с помощью линейных алгоритмов Ляпунова [3] с последующим применением карт Карно-Вейча для минимизации логических уравнений.

Литература

1. Гончаров С., Николаев Д., Никитин В., Писецкий В. Схемотехническая реализация автомата // Компоненты и технологии. 2013. № 2.
2. Ангер С. Асинхронные последовательностные схемы. М.: Наука, 1977.
3. Лазарев В. Г., Пийль Е. И. Синтез управляющих автоматов. М.: Энергия, 1978.