Глава 17

Программная реализация управляющих автоматов в базисе функциональных блоков

Одним из языков программирования ПЛК является язык функциональных блоков (схем) — «Function Blok Diagram». Этот язык применяется для программируемых логических контроллеров, выпускаемых различными фирмами, например «Siemens» (Германия), и для ряда управляющих систем, например для системы «Selma-2» фирмы «ABB Stromberg» (Финляндия), на примере которой излагается материал настоящего раздела.

Технология программирования в этом случае состоит: в построении функциональной схемы в базисе функциональных блоков, входящих в библиотеку (в общем случае расширяемую), в изображении схемы на дисплее и трансляции в программу, исполняемую системой.

При этом, однако, руководящие материалы по программированию [60] содержат лишь описания функциональных блоков и в лучшем случае примеры построения некоторых схем [229], но не содержат методов алгоритмизации — методов построения функциональных схем для автоматов с памятью. Это является существенным недостатком такого подхода к технологии программирования, так как эвристически построенную схему весьма трудно верифицировать, отлаживать, корректировать и читать. Для нее отсутствуют тесты для определения всех функциональных возможностей, и ее проверка выполняется обычно по прямому назначению, что принципиально не позволяет доказать для автомата с памятью, что схема, его реализующая, не делает ничего лишнего.

Методы, излагаемые в настоящей главе, совместно с методами, изложенными в разд. 4.2, позволяют устранить указанный пробел в указанном подходе.

Специфика программной реализации функциональных схем по сравнению с их асинхронной аппаратной реализацией включает в себя следующие особенности:

— фиксированную дисциплину опроса входных и считывания выходных и внутренних переменных;

— отсутствие состязаний и фиксированную дисциплину срабатывания элементов «схемы», что достигается за счет указания десятичных номеров рядом с функциональными блоками, из которых построена «схема»;

— однократную реализацию каждого блока с запоминанием результата в течение программного цикла;

— циклический характер обработки блоков в соответствии с их номерами: блоки с большими номерами не реализуются до тех пор, пока не реализованы блоки с меньшими номерами;

— недостаточность классических структурных моделей (автоматы Мура, Мили и т. д.);

— наличие блоков, работающих с многозначными переменными.

17.1. Анализ (чтение) функциональных схем

Пусть задана функциональная схема (рис. 17.1) и требуется прочесть ее — определить все ее функциональные возможности. Анализ этой схемы с двумя обратными связями в отличие от схемы, реализуемой аппаратно, может производиться не поэлементно:

$$v = y \vee z; \quad y = x \,\&\, v; \quad z = x \,\&\, \bar{y},$$

а выполнив частичную (16.1) либо полную подстановку (16.2). Граф переходов, построенный по СБФ (16.2), описывает поведение заданной

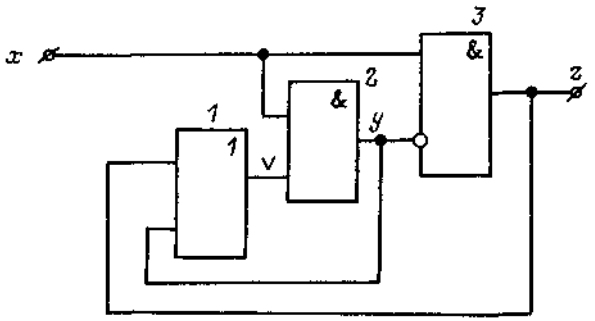

Рис. 17.1

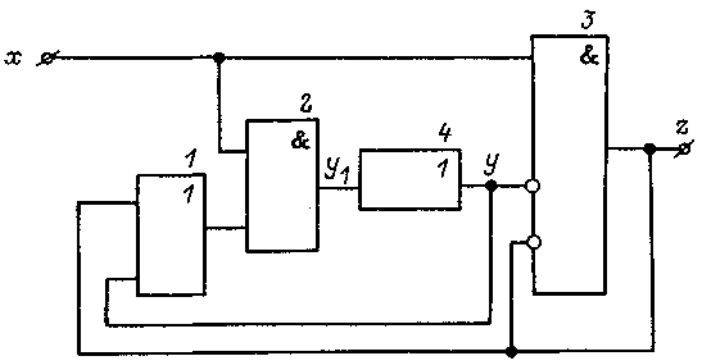

Рис. 17.2

схемы. Программная реализация заданного алгоритма может выполняться, например, по СБФ (16.4), которая может быть еще более упрощена — СБФ (16.6).

Из последнего соотношения следует, что заданная схема эквивалентна схеме на рис. 17.2, являющейся композицией автомата без выходного преобразователя первого рода и комбинационной схемы с самоблокировкой, охваченных внешней обратной связью. Нумерация элементов в последней схеме обеспечивает реализацию СБФ (16.4), в которой исключена подстановка значений формул друг в друга в течение программного цикла, как это имеет место для СБФ, соответствующей заданной схеме.

17.2. Синтез функциональных схем

В рамках предлагаемой технологии алгоритмизация проводится с помощью графов переходов. Функциональные схемы могут строиться либо по СБФ, полученной по графу переходов, либо непосредственно по этому графу.

При этом если схема не изоморфна графу, то при формальном переходе от графа к схеме вместо чтения ФС следует читать ГП, который при этом может использоваться также в качестве теста для демонстрации правильности работы схемы, которая либо реализует заданный граф переходов, либо отличается от него наличием вершин, имеющих исходящие дуги, но не содержащих входящих дуг.

При эвристическом построении ФС по ГП кроме указанных вариантов реализации возможен и третий вариант: функциональная схема реализует другой граф переходов. При этом если применять заданный граф переходов в качестве теста, то в одних случаях он позволяет обнаруживать ошибки в схеме, а в других позволяет обнаруживать ошибки лишь случайно. Продемонстрируем это на примере.

Предположим, что по графу переходов на рис. 17.3 эвристически построена функциональная схема, которая на самом деле реализует

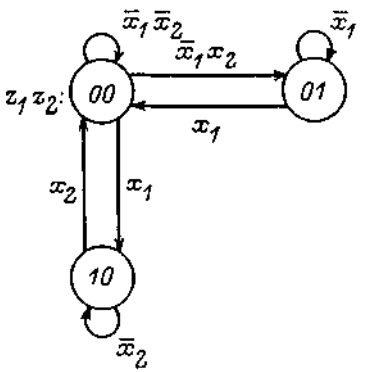

Рис. 17.3

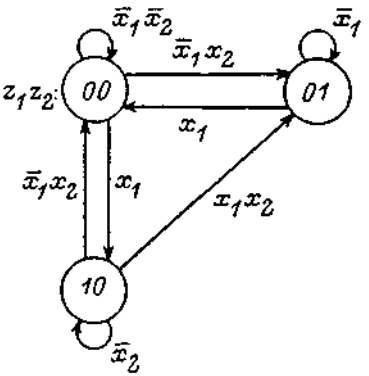

Рис. 17.4

другой граф (рис. 17.4). Используем для новой схемы в качестве теста исходный ГП. Так как в нем переход «10—00» не зависит от переменной x_1, то при $x_1 = 0$ тест в ошибочной схеме «проходит», а переход «10—01», имеющийся в ней, не проверяется вовсе из-за отсутствия его в исходном графе переходов.

Таким образом, при эвристическом построении функциональной схемы ее полная проверка с помощью тестов практически невозможна, а должна выполняться верификация, т. е. построение ГП по ФС и его сравнение с заданным. Проблема верификации снимается при формальном переходе от графа переходов к функциональной схеме.

17.2.1. Построение функциональных схем с обратными связями

Автомат Мили первого рода, реализующий последовательностный сумматор, построенный по СБФ (16.15), приведен на рис. 17.5.

Автомат Мили без применения элемента переобозначения, построенный по СБФ (16.16), изображен на рис. 17.6.

Композиция комбинационной схемы и автомата с самоблокировкой, построенная по СБФ (16.18), приведена на рис. 17.7.

Так как для рассматриваемой задачи не может быть построен автомат Мили второго рода, то на рис. 17.8 приведена схема автомата, одновременно использующая свойства автоматов Мили первого и второго рода. Эта схема построена по СБФ (16.19).

Рассмотренные схемы содержат лишь одну обратную связь. Функциональная схема (рис. 17.9), реализующая ГП на рис. 16.4, построенная по СБФ (16.9), содержит перекрестные обратные связи. Эти обратные связи делают схему практически нечитаемой. Однако в данном случае, как, впрочем, и в предыдущих, читать схему нет необходимости, так как имеется граф переходов, по которому она построена формально. Если блоки КС1 и КС2 реализовать древовидными схемами из элементов И, ИЛИ, НЕ, то они будут обладать нерегулярной структурой.

Регулярность структуры может быть повышена, если применить метод реализации произвольных булевых формул в базисе И, ИЛИ, НЕ из h букв схемой с линейной структурой из h мультиплексоров «2 в 1» [87, 277]. Метод состоит в построении по заданной БФ, например $z_{11} = (x_1 \,\&\, \bar{z}_1 \vee \bar{x}_2 \,\&\, z_1) \,\&\, \bar{z}_2$, линейного бинарного графа (рис. 17.10) и в

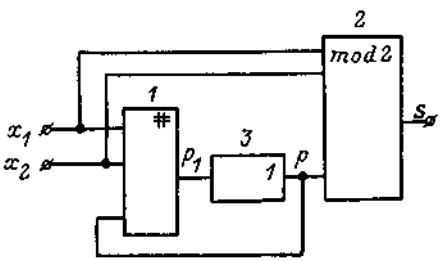

Рис. 17.5

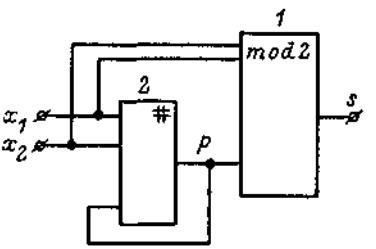

Рис. 17.6

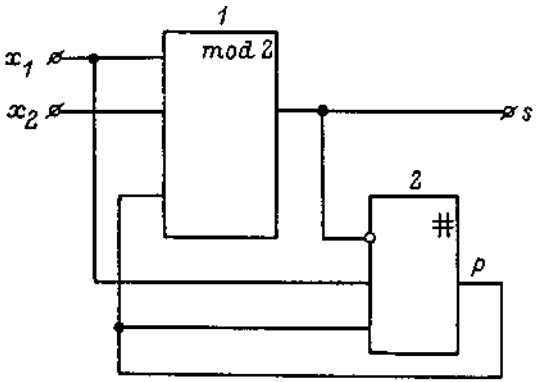

Рис. 17.7

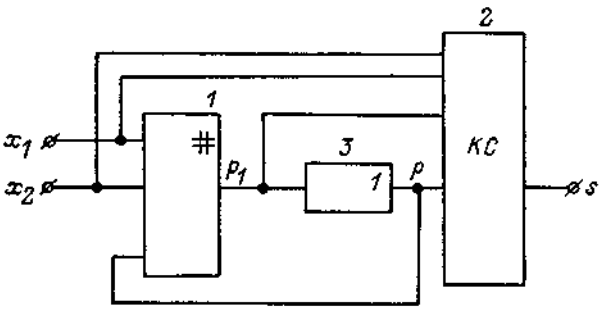

Рис. 17.8

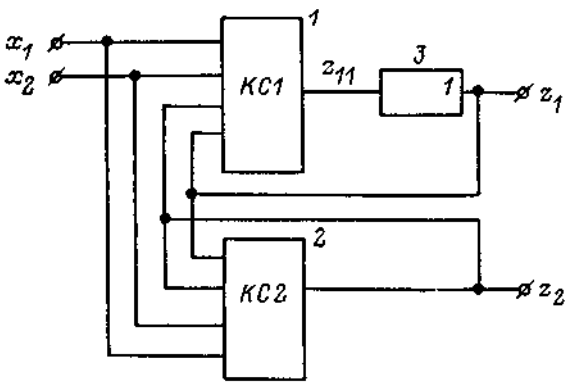

Рис. 17.9

замене каждой условной вершины в ЛБГ мультиплексором (рис. 17.11). В полученной схеме последний элемент всегда может быть исключен и заменен переменной или ее инверсией, подаваемой на управляющий вход предпоследнего мультиплексора, что обеспечивает реализацию рассматриваемого класса формул линейной схемой из h - 1 мультиплексоров.

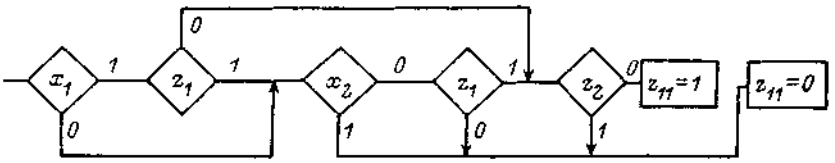

Рис. 17.10

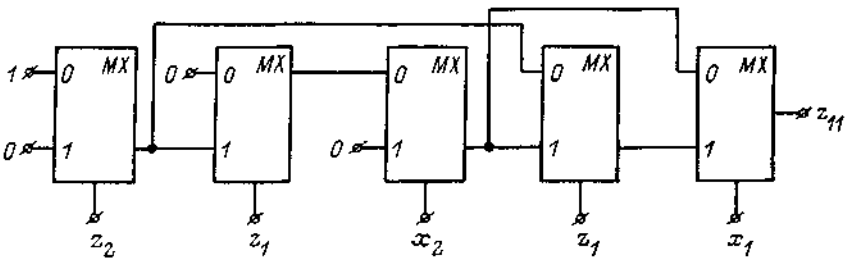

Рис. 17.11

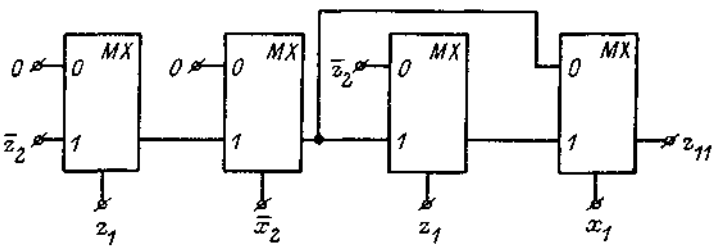

Рис. 17.12

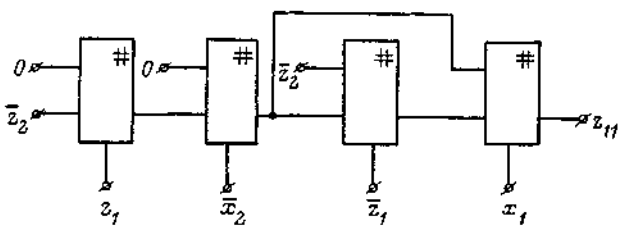

Рис. 17.13

Если на входы схемы подавать не только входные переменные, но и их инверсии, то однородность схемы может быть повышена (рис. 17.12). Можно показать [87, 277], что если в последней схеме каждый мультиплексор заменить на мажоритарный элемент, а входные переменные использовать в форме, определяемой реализуемой формулой, то это обеспечит реализацию рассматриваемого класса формул схемой с линейной структурой из h - 1 мажоритарных элементов (рис. 17.13).

17.2.2. Построение функциональных схем, использующих триггеры

Схема, приведенная на рис. 17.1, эквивалентна схеме, в которой применяется *R*-триггер (рис. 17.14). Для построения другой схемы на *R*-триггерах преобразуем соотношение (16.4) к следующему виду:

$$y_1 = x \,\&\, (y_1 \vee z); \quad z = x \,\&\, \bar{y} \,\&\, \bar{z}; \quad y = y_1.$$

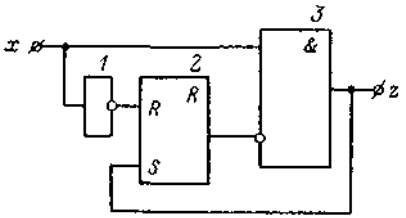

Рис. 17.14

Этой СБФ соответствует схема на рис. 17.2. Использование одной и той же переменной в левой и правой частях первой формулы позволяет реализовать ее с помощью *R*-триггера (рис. 17.15).

Применение триггеров при использовании рассматриваемой библиотеки функциональных блоков [60] ограничено, так как в ее состав входят только *R*-триггеры и не входят *S*- и *JK*-триггеры. Это не позво-

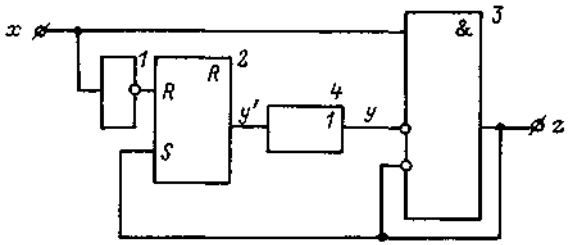

Рис. 17.15

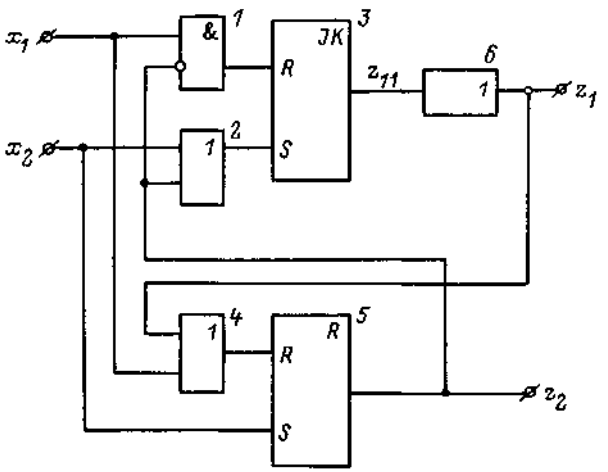

Рис. 17.16

ляет непосредственно применить малотрудоемкий аналитический метод синтеза схем на триггерах, предложенный в разд. 4.2.1. Однако его использование возможно после расширения библиотеки, например, как показано на рис. 4.19 и 4.20.

Если в состав библиотеки добавить *JK*-триггеры, которые могут быть описаны формулой $z = S\ \&\ !\, z \vee !\, R\ \&\ z$, то, преобразуя, например, (16.9) к виду:

$$z_{11} = x_1\ \&\ \bar{z}_2\ \&\ \bar{z}_{11} \vee \bar{x}_2\ \&\ \bar{z}_2\ \&\ z_{11};\quad z_2 = (x_2 \vee z_2)\ \&\ \bar{x}_1\ \&\ \bar{z}_1;\quad z_1 = z_{11},$$

эту СБФ можно реализовать схемой (рис. 17.16), построенной на двух триггерах разных типов.

При применении двоичного кодирования любой граф переходов может быть весьма просто реализован либо только на *R*-, либо только на *S*-триггерах (разд. 16.2.4).

17.2.3. Реализация управляющих автоматов

На рис. 17.17 приведена функциональная схема, построенная по СБФ (16.20). В этой схеме элемент *T*— таймер, а переменная *D* — величина задержки на срабатывание.

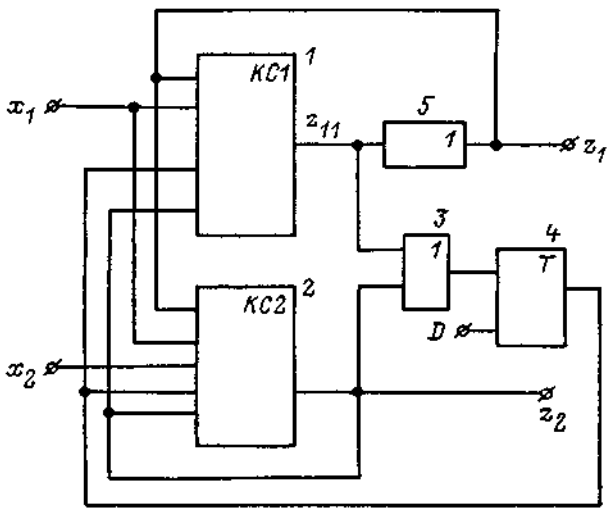

Рис. 17.17

17.2.4. Изоморфная реализация графов переходов функциональными схемами

Все рассмотренные схемы чрезвычайно сложно читаются, т. е. непосредственно по схеме без дополнительных вычислений не построить граф переходов.

Изложим два метода построения хорошо читаемых схем.

Метод построения хорошо читаемых функциональных схем на *R*-триггерах. Этот метод базируется на фиксации в триггерах изменений выходных и внутренних переменных, происходящих при переходах в графе переходов, вершины которого закодированы полными кодами. Функциональная схема, изоморфно реализующая граф переходов (рис. 16.9), приведена на рис. 17.18. Эта схема хорошо читается, так как по ней видно, что для состояния $z = 1$, $y = 0$ при $x = 1$ формирование нового состояния связано с изменением каждой из переменных: первая должна сбрасываться, а вторая устанавливаться — $z = 0$, $y = 1$. При переходе из состояния «00» в состояние «11» при $x = 1$ должны изменяться две переменные; при переходах по $x = 0$ — по одной переменной. Структура схемы позволяет достаточно просто без дополнительных преобразований восстановить граф переходов, который она реализует.

Если в приведенной схеме каждый трехвходовой элемент И представить в виде суперпозиции двух двухвходовых элементов И, первый из которых дешифрирует номер состояния, а второй определяет значение входной переменной, при котором выполняется переход из этого состояния, то структура схемы будет напоминать структуру ГСА (дешифратор состояний—дешифратор входных переменных—формирователь следующего состояния), предложенную в разд. 13.4.2 для случая принудительно-свободного кодирования состояний автоматов.

Метод построения хорошо читаемых функциональных схем на цифровых мультиплексорах. Достоинство этого метода, изложенного в разд. 4.2.3, состоит в возможности использования для кодирования состояний произвольного автомата всего лишь одной многозначной перемен-

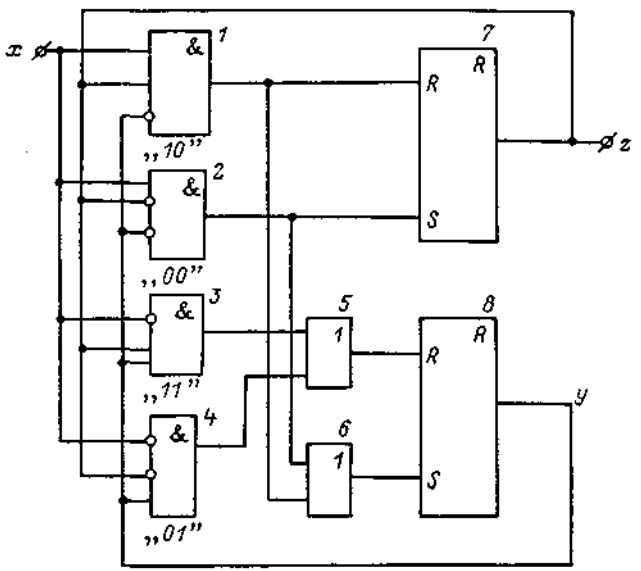

Рис. 17.18

ной, что упрощает отладку и чтение схем, построенных на основе предложенного метода. До построения схемы может быть выполнено ее моделирование с помощью изоморфной программы, написанной на языке СИ, в которой применяется конструкция `switch`.

Недостаток метода — большие затраты памяти на реализацию функциональных блоков, используемых в схеме.

В заключение раздела отметим, что рассматриваемая библиотека функциональных блоков содержит также и блок «STACON», который позволяет непосредственно реализовывать графы переходов достаточно большой размерности. Однако сложность настройки этого блока резко ограничивает целесообразность его применения.