Глава 18

Программная реализация управляющих автоматов в базисе лестничных схем

Одним из наиболее распространенных языков программирования ПЛК является язык лестничных схем — «Ladder Diagram». Модификации этого языка применяются в контроллерах таких фирм, как «Siemens» (Германия), «Telemecanique» (Франция), «Modicon» (США), «Allen—Bradley» (США), «Omron» (Япония), «Mitsubishi Electric» (Япония) и других.

Базовая часть этих языков содержит команды, позволяющие изображать на дисплее релейно-контактные схемы, и поэтому эта часть языка называется РКС («релейно-контактные схемы»). Однако наличие большого числа других команд, изображаемых на месте обмоток релейных схем, делает использование названия «лестничные схемы» (ЛС) более оправданным. В качестве таких команд могут применяться различные функциональные блоки,

В литературе распространено мнение, что ЛС наиболее просто строить за счет непосредственной замены элементов уже существующих аппаратных РКС (АРКС) программно реализуемыми элементами библиотеки лестничных схем. Однако даже в этом случае существенные различия между этими классами схем делает такой подход нецелесообразным. Перечислим основные особенности этих классов схем.

Для аппаратных релейно-контактных схем характерны:

— принципиальная асинхронность

— произвольная дисциплина изменений входных воздействий, риск комбинационных схем и состязания элементов памяти автоматов, что делает поведение последних весьма трудно предсказуемым априори, так как схема, построенная по графу переходов, может иметь поведение, отличное от заданного;

— возможность построения как параллельно-последовательных (*П*-схемы), так и мостиковых схем (*Н*-схемы);

— невозможность принадлежности одного и того же контакта различным реле;

— возможность применения переключательных контактов, диодов и поляризованных реле;

— ограниченная номенклатура элементов базиса;

— принципиальная потенциальность.

Лестничные схемы характеризуются тем, что:

— реализация схемы осуществляется циклически и синхронно;

— входные перемеиные не могут быть введены повторно в течение одного программного цикла, а выходные и внутренние переменные считываются в его конце;

— состязания элементов отсутствуют, и поэтому схема работает однозначно: либо правильно, либо неправильно;

— не могут быть мостиковыми, но при многих выходах могут быть непараллельно-последовательными;

— различные «обмотки» с одним и тем же наименованием могут управлять одним и тем же «контактом», состояние которого для «обмоток» без памяти определяется состоянием последней из опрошенных «обмоток»;

— переключательные контакты не используются, а применяются только замыкающиеся и размыкающиеся контакты;

— имеют широкий набор «элементов», не включающий, правда, таких элементов, как например диоды или переключающие контакты;

— обеспечивают простоту формирования импульсных переменных и содержат команды, выполняемые только по переднему фронту входных переменных;

— являются планарными и изображаются только в одном направлении — слева направо и сверху вниз.

В свою очередь принципиальное отличие аппаратных РКС и лестничных схем от функциональных схем состоит в том, что в схемах первых двух типов формулы И и ИЛИ делаются на «проводах», а элементы типа «неравнозначность» отсутствуют.

Отличие аппаратных РКС и лестничных схем от ГСА состоит в том, что первые являются принципиально двудольными (в каждой цепи сначала изображаются «контакты», а затем «обмотки»), в то время как в граф-схемах алгоритмов при подходе к их построению, отличном от предлагаемого в гл. 13, условные и операторные вершины могут быть перемешаны. Другое принципиальное отличие указанных схем от ГСА состоит в том, что параллельное соединение «обмоток» соответствует последовательному соединению операторных вершин, а последовательному соединению блоков в граф-схемах алгоритмов соответствует параллельное соединение цепей в схемах.

Из изложенного следует, что использование методов реализации аппаратных РКС [18] для построения лестничных схем требует корректировки. При этом необходимо отметить, что фирменные руководства по программированию содержат лишь примеры построения лестничных схем, но не содержат методов реализации автоматов. В ряде случаев поведение эвристически построенных схем иллюстрируется временными диаграммами, которые для сложных схем не позволяют отразить все изменения значений каждой выходной и каждой внутренней переменной при всех допустимых изменениях значений входных переменных, что весьма просто отображается с помощью графов переходов.

Вызывает удивление также и тот факт, что в документации одних и тех же фирм (например, «Siemens», «Omron») построение лестничных схем и диаграмм «Графсет» рассматривается с различных позиций. При

этом описывается возможность их совместного применения [260], в то время как, например, методы построения лестничных схем по графам переходов или диаграммам «Графсет» не излагаются.

В некоторых документах (например, фирмы «Mitsubishi Electric») «простые» автоматы с памятью предлагается строить с помощью лестничных схем, эвристически, а для «сложных» автоматов — использовать диаграммы «Графсет». Однако из рассмотрения этих документов остается не ясным, где проходит грань между простыми и сложными автоматами.

В настоящем разделе с единых позиций излагаются методы реализации лестничных схем для автоматов различной сложности.

18.1. Построение комбинационных лестничных схем

На примере реализации СБФ

$$z_1 = x_1 \,\&\, x_2 \vee x_3; \quad z_2 = x_1 \,\&\, x_2 \,\&\, x_3$$

покажем, какие схемы могут строиться как для аппаратных РКС, так и для лестничных схем, а какие — только для аппаратных РКС или только для лестничных схем.

На рис. 18.1, 18.2 приведены *П*-схемы, применяемые в обоих базисах.

Следующие две схемы специфичны для аппаратных РКС (рис. 18.3, 18.4). Построение первой из них основано на ортогонализации первой формулы.

Специфика лестничных схем начинает проявляться уже при построении *П*-схем. Например, формулы $z = x_1 \,\&\, (x_2 \vee x_3)$ и $z = (x_2 \vee x_3) \,\&\, x_1$,

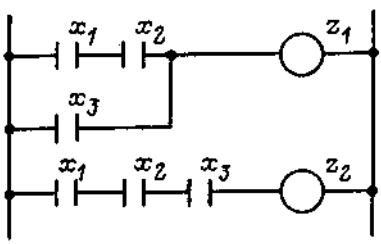

Рис. 18.1

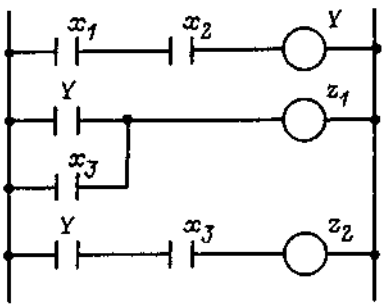

Рис. 18.2

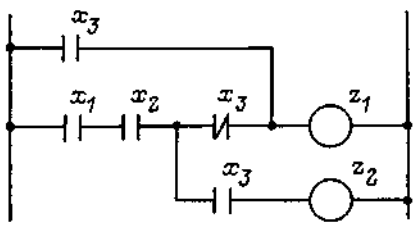

Рис. 18.3

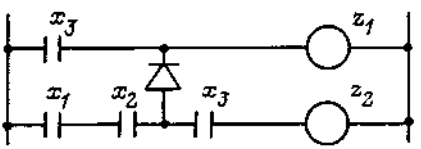

Рис. 18.4

соответствующие одной и той же булевой функции и реализуемые в классе аппаратных РКС с одинаковой сложностью, в классе лестничных схем, имея одинаковое число контактов, порождают программы разной сложности. Первая из этих формул реализуется сложнее, так как при трансляции переменная x_1, введенная в аккумулятор, должна быть переписана в промежуточную память, например стек, а при реализации второй формулы промежуточная память не требуется.

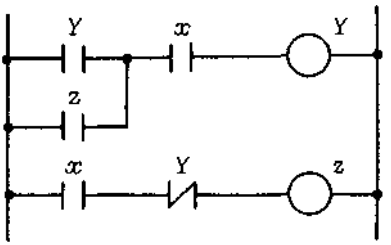

Рис. 18.5

При этом отметим, что минимизация программы по указанной причине в общем случае обеспечивается для бесповторных пороговых формул или бесповторных пороговых фрагментов произвольных формул при их записи в порядке возрастания весов переменных.

18.2. Анализ лестничных схем

Анализ комбинационных лестничных схем сводится к построению СБФ по схеме и к восстановлению по этой системе формул таблицы истинности.

Анализ лестничных схем, реализующих автоматы с памятью, состоит: в записи СБФ по заданной схеме; в построении новой СБФ, получаемой в результате всех возможных подстановок, проводимых в исходной системе; в построении графа переходов по новой СБФ.

Пусть задана лестничная схема (рис. 18.5) и требуется выполнить ее анализ. Эта схема описывается СБФ (16.1). Дальнейший анализ этой схемы выполнен в разд. 16.1.

18.3. Методы реализации автоматов

18.3.1. Построение лестничных схем по ГСА

Лестничные схемы могут строиться непосредственно по ГСА. На рис. 18.6 приведена лестничная схема, реализующая счетный триггер и построенная по ГСА (рис. 13.23). В этой схеме применяются переобозначения переменных z и y, кодирующих состояния автомата, для того чтобы формально исключить возможность использования в ней новых значений указанных переменных в течение одного программного цикла.

Выполняя анализ построенной схемы и учитывая тот факт, что значения входной переменной x не могут изменяться в течение программного цикла, можно показать, что в данном случае лестничная схема (рис. 18.7) будет функционировать корректно и без применения переобозначения переменных.

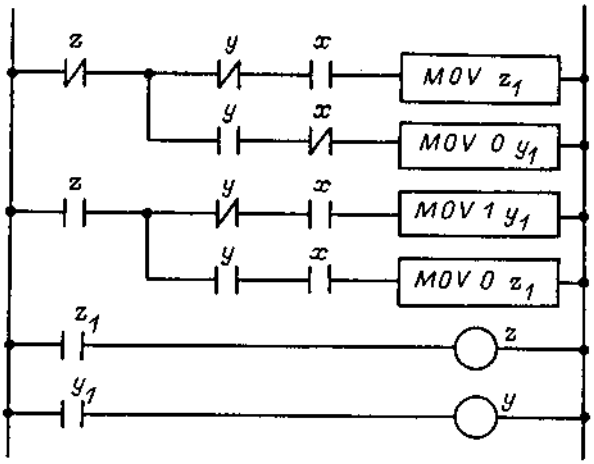

Рис. 18.6

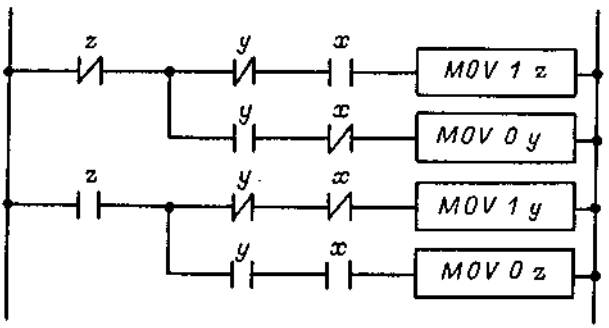

Рис. 18.7

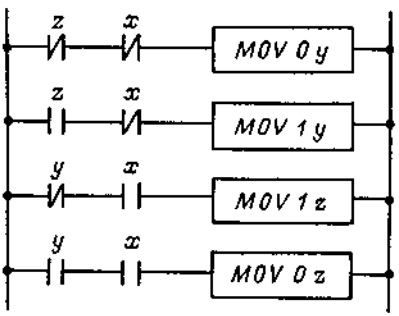

Рис. 18.8

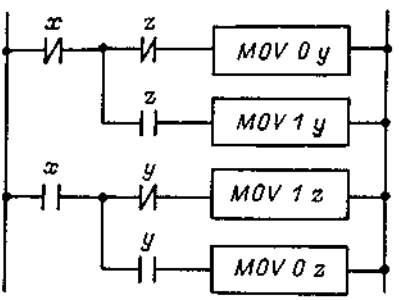

Рис. 18.9

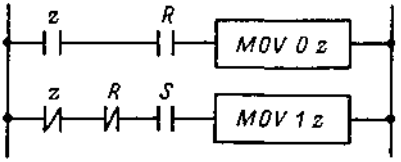

Рис. 18.10

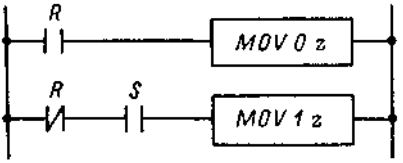

Рис. 18.11

Полученная схема может быть упрощена, если лестничную схему (рис. 18.8) построить по ГСА на рис. 13.24. В этой схеме также не требуется использовать переобозначения переменных, так как все цепи в ней в динамике ортогональны.

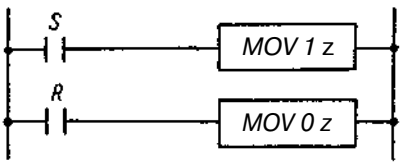

Рис. 18.12

Дальнейшее упрощение лестничной схемы (рис. 18.9) достигается при построении ее по ГСА на рис. 13.25. Эта схема анализируется на ортогональность цепей проще предыдущих, так как в ней контактная часть начинается с опроса значений входной переменной *x,* не изменяющейся в течение программного цикла.

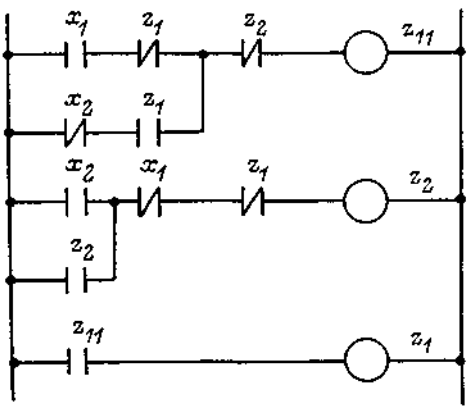

Рис. 18.13

Выполним аналогичные построения лестничных схем для *R*-триггера. На рис. 18.10 приведена лестничная схема, построенная по ГСА (рис. 4.83) с учетом замены переменной *y* на переменную *z*. В этой схеме переобозначение переменной г не применяется, так как цепи ортогональны по входной переменной *R*.

Упрощение лестничной схемы, реализующей *R*-триггер, правда, при одновременном ухудшении ее читаемости, достигается при построении последней (рис. 18.11) по ГСА на рис. 4.76. Простейшая схема *R*-триггера (рис. 18.12) строится по ГСА на рис. 4.79.

18.3.2. Синтез лестничных схем по СБФ, построенным по графам переходов

Пусть, например, по графу переходов на рис. 16.4 построена СБФ (16.9), которая реализуется лестничной схемой (рис. 18.13). Рассмотрение этой схемы развенчивает миф о том, что лестничные схемы легко читать и поэтому они могут применяться в качестве единственного языка при

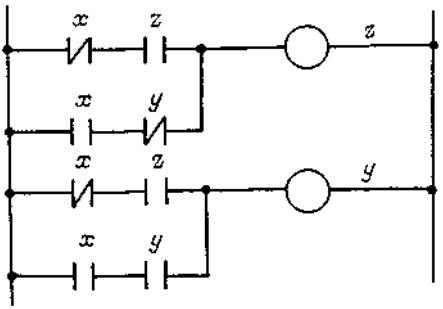

Рис. 18.14

алгоритмизации, программировании и эксплуатации. При использовании предлагаемого подхода читать лестничную схему нет необходимости, так как она формально построена по графу переходов, который и следует читать.

В заключение раздела отметим, что изложенный метод синтеза является универсальным и позволяет для произвольного графа переходов строить работоспособную лестничную схему, реализующую его, используя не более $m + l - 1$ переобозначений, где m и l — число выходных и внутренних переменных в графе переходов соответственно. При этом отметим, что применять переобозначения переменных требуется не всегда. Если счетный триггер (рис. 13.22) реапизовать не по СБФ (16.12), а по СБФ (16.13), то в лестничной схеме (рис. 18.14) не требуется переобозначать переменные.

18.3.3. Построение лестничных схем непосредственно по графам переходов

Использование *R*-триггеров. Идея построения схемы в этом случае состоит в отражении в *R*-триггерах изменений значений выходных и внутренних переменных, возникающих при переходах в графе переходов. При таком подходе, ввиду того что лестничная схема должна быть планарна и изображаться сверху вниз, а триггеры на плоскости изображаются в виде столбца, в лестничной схеме каждой дуге графа переходов в общем случае соответствует отдельная цепь. Число переобозначений в этом случае не превышает величины $m + l - 1$.

На рис. 18.15 приведена лестничная схема, построенная для графа переходов (рис. 16.4) для случая использования неполных кодов. Достоинство этой схемы состоит в том, что по ней в отличие от схемы на рис. 18.13 можно без вычислений однозначно восстановить граф переходов, но, правда, с некоторыми трудностями. Эти трудности уменьшаются, если при построении схемы (рис. 18.16) использовать полные коды, применяемые в вершинах ГП (рис. 16.4).

Использование команд записи нуля и единицы в ячейки памяти. В этом случае может быть обеспечен полный изоморфизм между графом переходов и лестничной схемой. Схема начинается с дешифратора полных или неполных кодов, применяемых в вершинах графа переходов, затем реализуются формулы переходов, а потом размещаются команды записи нуля и единицы в ячейки промежуточной и (или) выходной памяти, например команды `RST Z (MOV 0 Z)` и `SET Z (MOV 1 Z)`. В конце схемы в случае необходимости располагается блок, осуществляющий переобозначения переменных.

На рис. 18.17 приведена лестничная схема, реализующая ГП на рис. 16.4 при использовании неполных кодов состояний. Упрощение контактной части схемы и улучшение ее читаемости по сравнению со схемами, приведенными выше, компенсируются усложнением ее операционной части.

Применение полных кодов обеспечивает наилучшую читаемость лестничной схемы (рис. 18.18), так как по ней без дополнительных вычислений легко может быть восстановлен граф переходов без умолчаний значений всех переменных, размещаемых в вершинах. Для предлагаемого подхода число переобозначений может увеличиться до величины $m + l$.

Построение лестничных схем по системе взаимосвязанных графов переходов. При реализации СВГП для каждого графа переходов должна быть построена лестничная схема, после чего полученные схемы объединяются в одну компоненту. На рис. 18.19 приведена схема, реализующая СВГП (рис. 16.10).

В ряде случаев учет специфики графа переходов и его программной реализации позволяет некоторые переменные не переобозначать, исключая при этом соответствующие цепи в лестничной схеме. Так, например, цепи 3 и 5 на рис. 18.19 ортогональны, и поэтому переменную Y_1 можно не переобозначать.

Построение лестничных схем при многозначном кодировании состояний автоматов Мура. На рис. 18.20 приведена лестничная схема, реализующая

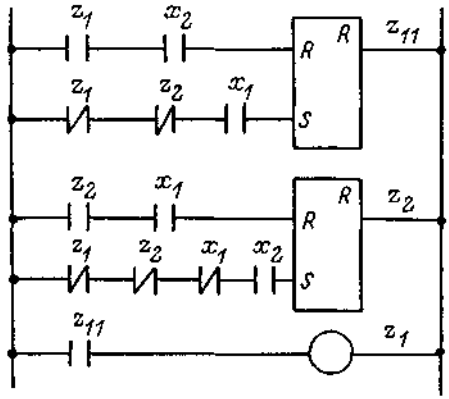

Рис. 18.15

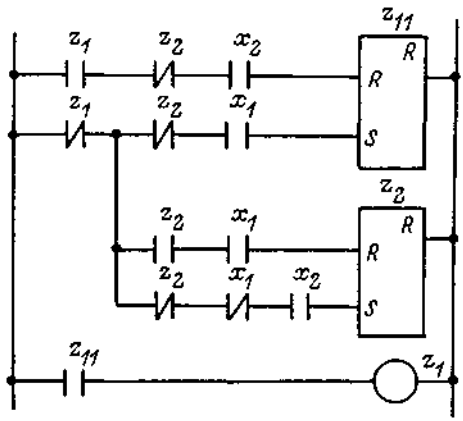

Рис. 18.16

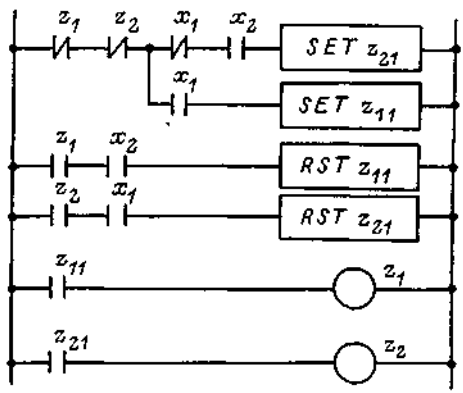

Рис. 18.17

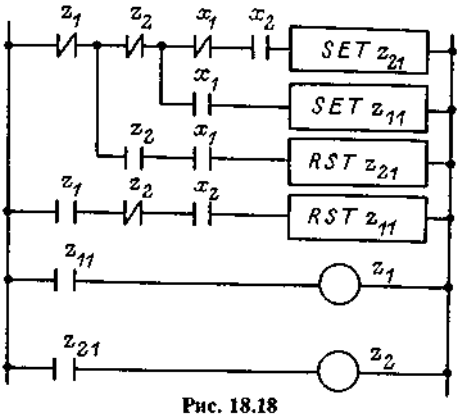

Рис. 18.18

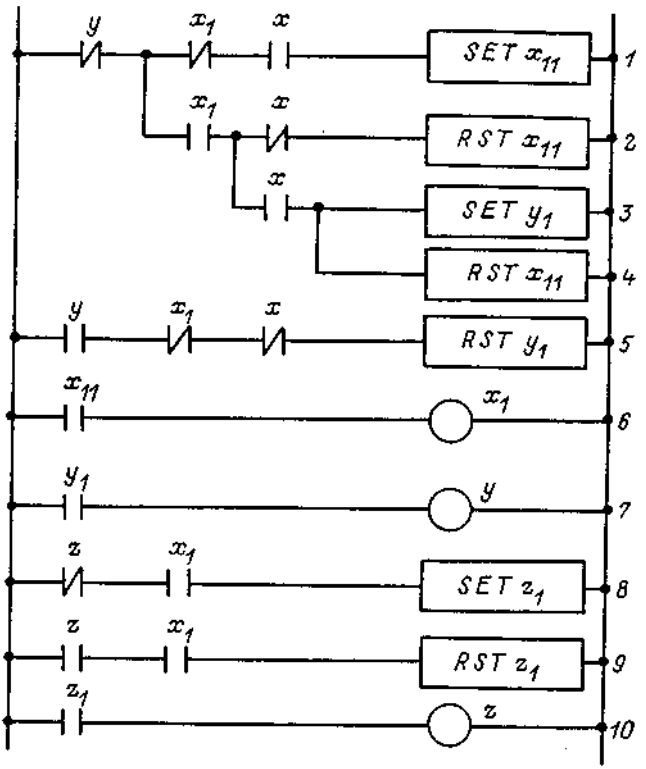

Рис. 18.19

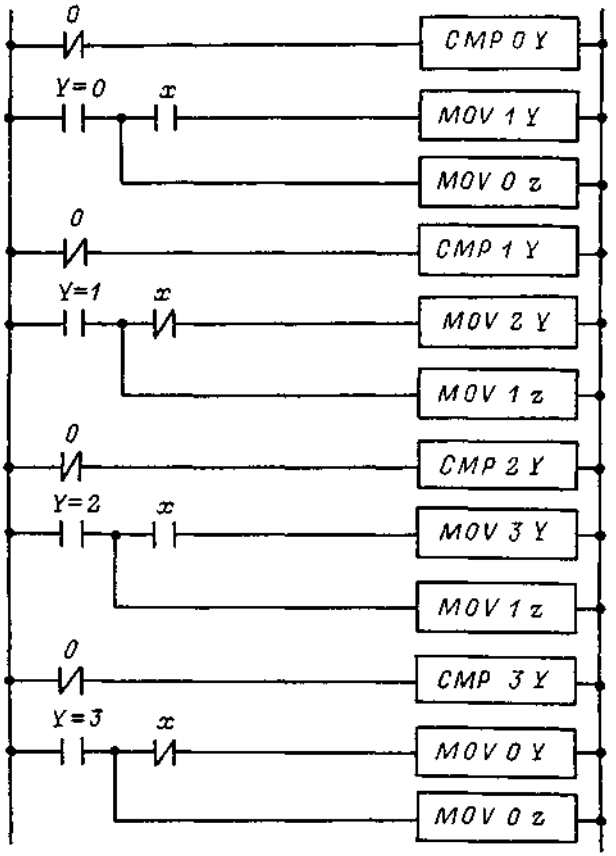

Рис. 18.20

ГП автомата Мура (рис. 4.124) счетного триггера при многозначном кодировании его состояний. В этой схеме не применяется переобозначение переменной *Y,* так как в графе переходов для всех вершин входные и выходные дуги имеют ортогональные пометки.

Реализация графов переходов автоматов Мили лестничными схемами. На рис. 18.21 приведена лестничная схема, реализующая ГП автомата Мили для последовательностного сумматора (рис. 4.113). Обратный переход от этой схемы к графу переходов может быть выполнен без дополнительных вычислений.

18.4. Реализация управляющих автоматов лестничными схемами

На рис. 18.22 приведена лестничная схема, построенная по СБФ, описывающей граф переходов (рис. 16.13) автомата, входящего в состав управляющего автомата. На этой схеме символом TIM обозначен таймер, выполняющий функцию элемента задержки на срабатывание.

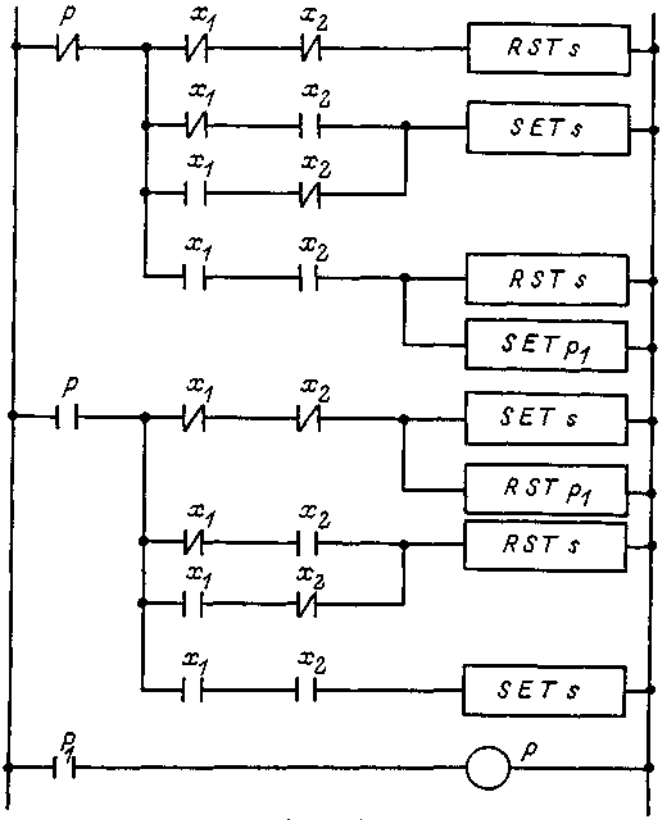

Рис. 18.21

Непосредственно по этому графу переходов при использовании полных кодов по переменным z_1 и z_2 может быть также построена лестничная схема (рис. 18.23). В этой схеме не применяются переобозначения переменных, так как, до тех пор пока не сработает таймер, каждая пара цепей в ней ортогональна. Приведенные в настоящем разделе схемы не изоморфны графам переходов.

Для того чтобы лестничная схема была изоморфна ГП, специфика ее построения требует, чтобы граф переходов имел форму автомата Мура и однократно запускал таймеры. На рис. 18.24 приведен ГП автомата Мура, эквивалентный графу переходов на рис. 16.13. Лестничная схема, построенная по этому графу, приведена на рис. 18.25.

18.5. Устранение генерации в лестничных схемах

На рис. 18.26 приведена лестничная схема, реализующая граф переходов на рис. 14.13. В этой схеме используется команда «Differentiation—Up» (DIFU), обеспечивающая по переднему фронту входной

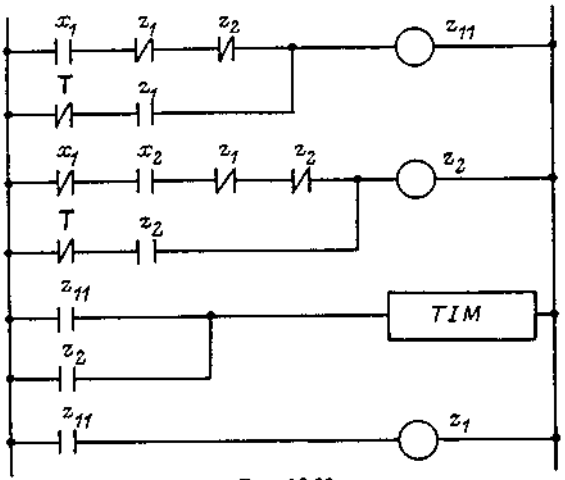

Рис. 18.22

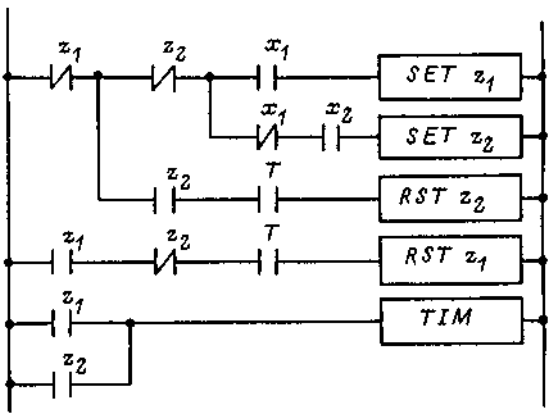

Рис. 18.23

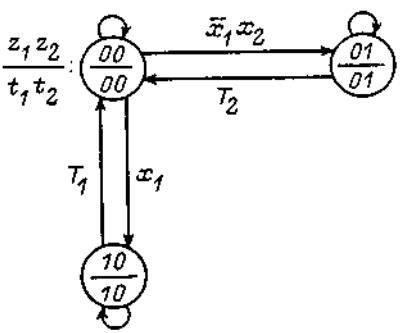

Рис. 18.24

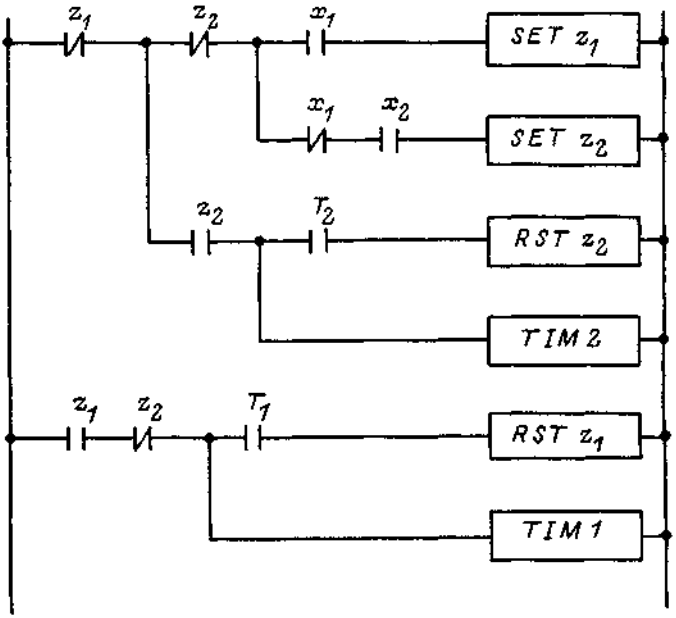

Рис. 18.25

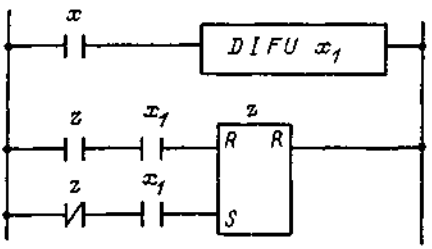

Рис. 18.26

переменной сохранение единичного значения на один программный цикл. В ПЛК существуют и другие возможности формирования импульсных переменных.

18.6. Метод программной реализации автоматов по объединенной формуле — метод независимых фрагментов

Формула называется объединенной, если она построена по СБФ следующим образом:

— переменная z_i или y_i, находящаяся в левой части i-й формулы СБФ ($i = 1,..., m$), обозначается символом z_{i1} или y_{i1} соответственно;

— каждая формула СБФ записывается в дизъюнктивно нормальной форме;

— каждая конъюнкция формулы z_{i1} (y_{i1}) умножается на переменную z_{i1} (y_{i1});

— все конъюнкции всех формул преобразованной СБФ объединяются символами дизъюнкций] , отличными от общепринятых.

В объединенной формуле могут быть выполнены любые вынесения переменных за скобки. По объединенной формуле, используя правила алгебры логики, может быть построена лестничная схема, использующая команды записи единицы в ячейки оперативной памяти z_{i1} (y_{i1}) — команды MOV 1 z_{i1} (MOV 1 y_{i1}) или SET z_{i1} (SET y_{i1}).

Для корректной работы эта схема должна быть дополнена еще двумя блоками, первый из которых состоит из m цепей, соединенных параллельно, каждая из которых образована контактом z_{i1} (y_{i1}), соединенным последовательно с обмоткой z_{i1} (y_{i1}) или с командой OUT z_i (OUT y_i), а второй блок представляет собой безусловно замкнутый контакт, подключенный последовательно к параллельному соединению из m команд записи нуля в ячейки оперативной памяти z_{i1} (y_{i1}) — команд MOV 0 z_{i1} (MOV 1 y_{i1}) или команд RST z_{i1} (RST y_{i1}).

Пример 1. Реализовать СБФ

$$z_1 = x_1 \,\&\, x_2 \vee x_3; \qquad z_2 = x_1 \,\&\, x_2 \,\&\, x_3$$

лестничной схемой с помощью предлагаемого метода.

Построим объединенную формулу и выполним в ней вынесение влево за скобки переменных x_1 и x_2:

$$\Phi = x_1 \,\&\, x_2 \,\&\, z_{11} \,\dot{\vee}\, x_3 \,\&\, z_{12} \,\dot{\vee}\, x_1 \,\&\, x_2 \,\&\, x_3 \,\&\, z_{21} =$$

$$= x_1 \,\&\, x_2 \,\&\, (z_{11} \,\dot{\vee}\, x_3 \,\&\, z_{21}) \,\dot{\vee}\, x_3 \,\&\, z_{11}.$$

Лестничная схема, построенная предлагаемым методом, приведена на рис. 18.27.

Пример 18.2. Реализовать СБФ

$$z_1 = x_1 \,\&\, (x_2 \vee x_3); \qquad z_2 = x_1 \,\&\, \bar{x}_2 \,\&\, \bar{x}_3$$

лестничной схемой с помощью предлагаемого метода.

Построим объединенную формулу и выполним в ней вынесение влево за скобки переменной x_1, а вправо за скобки — переменной z_{11}:

$$\Phi = x_1 \,\&\, x_2 \,\&\, z_{11} \,\dot{\vee}\, x_1 \,\&\, x_3 \,\&\, z_{11} \,\dot{\vee}\, x_1 \,\&\, \bar{x}_2 \,\&\, \bar{x}_3 \,\&\, z_{21} =$$

$$= x_1 \,\&\, ((x_2 \,\dot{\vee}\, x_3) \,\&\, z_{11} \,\dot{\vee}\, \bar{x}_2 \,\&\, \bar{x}_3 \,\&\, z_{21}).$$

Лестничная схема, построенная предлагаемым методом, изображена на рис. 18.28.

При этом отметим, что при построении структурированных ГСА методом независимых фрагментов вынесение в объединенной формуле переменных вправо за скобки недопустимо (разд. 4.3.2).

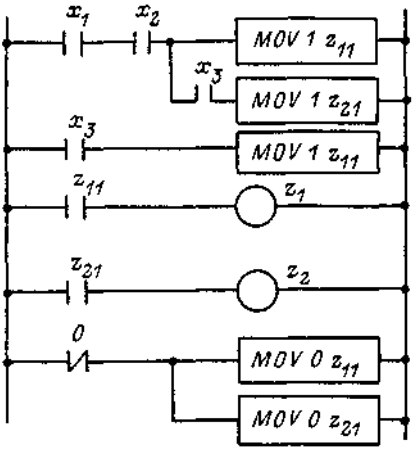

Рис. 18.27

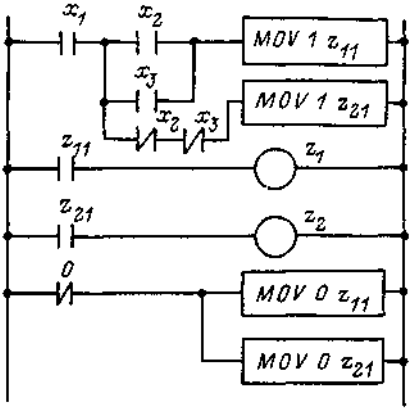

Рис. 18.28

Пример 18.3. Реализуем лестничной схемой счетный триггер на основе СБФ (16.13), полученной по графу переходов на рис. 13.22:

$$z = \bar{x}\,\&\,z \vee x\,\&\,\bar{y}; \qquad y = \bar{x}\,\&\,z \vee x\,\&\,y.$$

Построим объединенную формулу и выполним в ней вынесение переменных влево за скобки:

$$\Phi = \bar{x}\,\&\,z\,\&\,z_1 \dot{\vee} x\,\&\,\bar{y}\,\&z_1 \dot{\vee} \bar{x}\,\&\,z\,\&\,y_1 \dot{\vee} x\,\&\,y\,\&\,y_1 =$$

$$= \bar{x}_1\,\&\,z\,\&\,(z_1 \dot{\vee} y_1) \dot{\vee} x\,\&\,(\bar{y}\,\&\,z_1 \dot{\vee} y\,\&\,y_1).$$

Лестничная схема, построенная по этой формуле, приведена на рис. 18.29.

Изложенный метод отражает специфику лестничных схем, и применение его невозможно для аппаратных РКС, так как, например, в последней схеме (рис. 18.29) несколько команд управляет одним и тем же контактом. В предложенном методе попытка минимизации числа элементов в схеме во многом обеспечивается за счет увеличения числа обмоток, что в принципе не применимо для аппаратных РКС.

Предложенный метод позволяет строить по формуле непараллельно-последовательные (лестничные) схемы для СБФ, описывающих в том числе и автоматы с памятью, что принципиально отличает предлагаемый метод от известного метода [26], обеспечивающего построение параллельно-последовательных схем для каждой формулы, возможно, факторизованной системы.

Метод, несмотря на свою оригинальность, целесообразно применять лишь в тех случаях, когда за счет нетрадиционного объединения одинаковых частей разных формул обеспечивается экономия памяти, что весьма сложно, так как цена контакта 3—4 байта, а цена команд OUT и MOV

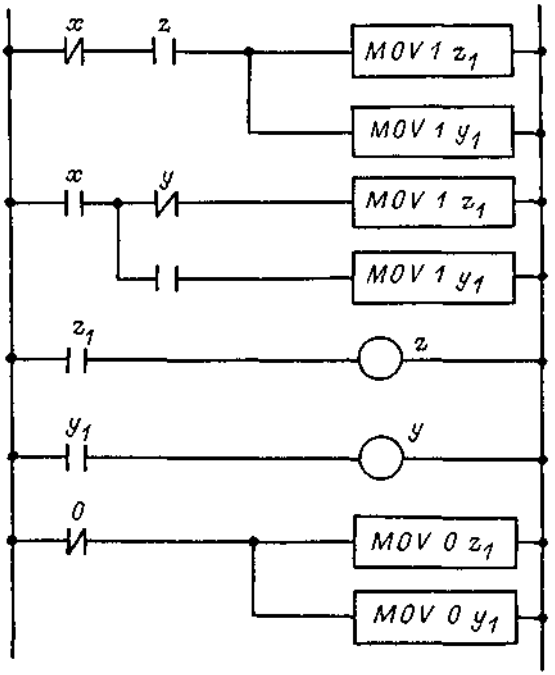

Рис. 18.29

соответственно 6 и 8—10 байт. Эффект от применения метода снижается и при появлении точек объединения контактов, транслируемых как команда OUT [236].

Область эффективности предлагаемого метода резко возрастает при его применении для построения граф-схем алгоритмов. Его можно рассматривать как развитие метода независимых фрагментов (разд. 4.3.2), предложенного для построения структурированных ГСА, реализующих автоматы без памяти по СБФ, на класс автоматов с памятью, также задаваемых системой булевых формул.

При этом необходимо отметить, что в отличие от лестничных схем для ГСА использование дизъюнкций в объединенной формуле не вполне естественно, так как они в этом случае отображают последовательное, а не параллельное, как в лестничных схемах, соединение блоков. На рис. 18.30 приведена ГСА, реализующая предлагаемым методом счетный триггер, которая эквивалентна лестничной схеме (рис. 18.29).

В заключение раздела отметим, что часть ГСА, реализующая собственно обобщенную формулу, всегда может быть структурированной, если при минимизации объединенной формулы отказаться от выноса вправо за скобки переменных z_{i1}.

Так, например, если по скобочной формуле с вынесенной вправо за скобки переменной z_{11}, приведенной в примере *18.2,* построить ГСА, то она будет неструктурированной (рис. 18.31). Если в объединенной

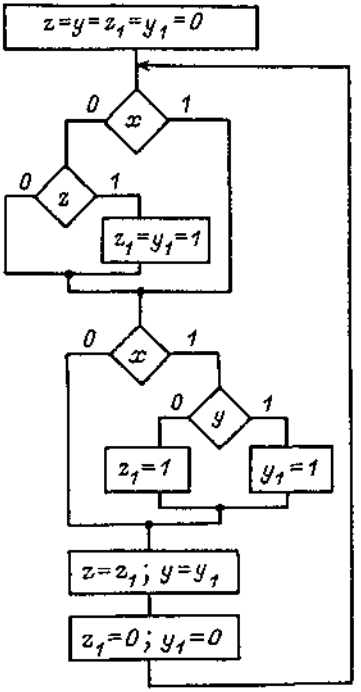

Рис. 18.30

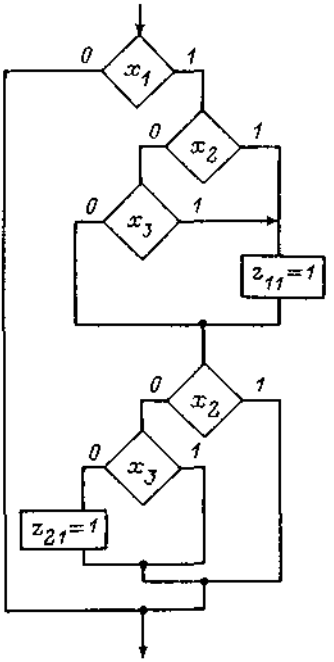

Рис. 18.31

формуле, приведенной в этом примере, вынести влево только переменную x_1:

$$\Phi = x_1 \,\&\, (x_2 \,\&\, z_{11} \mid x_3 \,\&\, z_{11} \mid \bar{x}_2 \,\&\, \bar{x}_3 \,\&\, z_{21}),$$

то построенный по этой формуле первый блок (рис. 18.32) в полной ГС А будет структурированным.

Если по рассматриваемой объединенной формуле построить другую скобочную формулу

$$\Phi = x_1 \,\&\, (x_3 \,\&\, z_{11} \mid (\bar{x}_2 \,\&\, \bar{x}_3 \,\&\, z_{21} \vee x_2 \,\&\, z_{11})),$$

то появляется возможность применения в структурированной ГСА конструкции «полный выбор» по переменной x_2 (рис. 18.33).

Отметим также, что фрагмент ГСА, реализующий объединенную формулу, является комбинационным и поэтому последовательно соединенные блоки в граф-схеме допускают перестановку. Так, например, для рассматриваемой формулы эквивалентная ГСА может строиться и по другой формуле:

$$\Phi = x_1 \,\&\, ((\bar{x}_2 \,\&\, \bar{x}_3 \,\&\, z_{21} \vee x_2 \,\&\, z_{11}) \mid x_3 \,\&\, z_{11}).$$

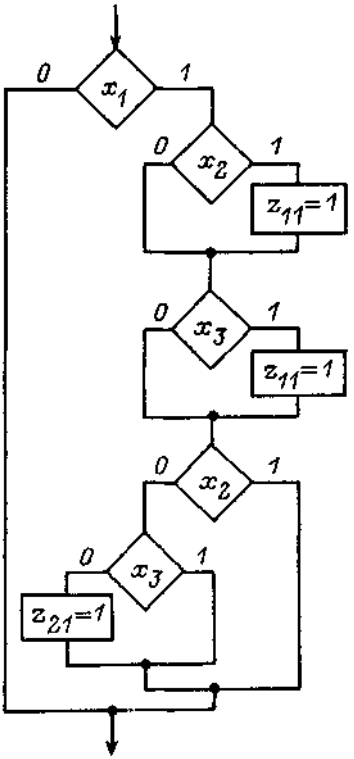

Рис. 18.32

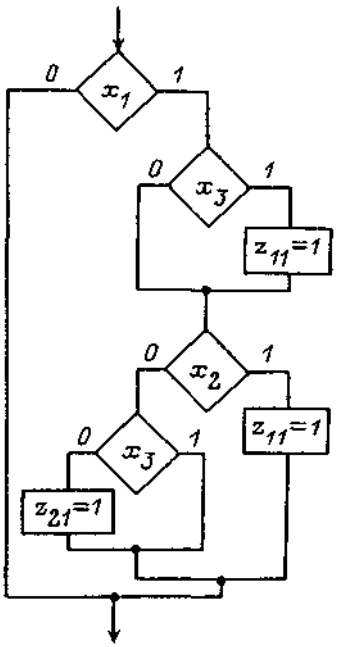

Рис. 18.33

Именно по этой причине в [69] предложенный метод построения по объединенным формулам структурированных ГСА для автоматов без памяти был назван методом независимых фрагментов. Из изложенного следует, что это название может быть использовано и для предлагаемого метода, предназначенного для программной реализации автоматов с памятью «схемами», обладающими трехблочной канонической структурой. В этих схемах первый блок вычисляет значения дополнительно введенных промежуточных переменных, определяющих следующее состояние автомата, второй блок осуществляет присвоение значений этих переменных промежуточным и выходным переменным, применяемым в реализуемой СБФ, а третий — выполняет установку в ноль значений всех дополнительно введенных промежуточных переменных.

Таким образом, объединенная формула однозначно описывает структуру первого блока лестничной схемы (структурированной ГСА) и, наоборот, по первому блоку такой схемы (структурированной ГСА) однозначно может быть построена указанная формула.

При этом отметим, что второй и третий блоки структуры описываются простейшими булевыми формулами.

Существенное отличие объединенных формул и скобочных форм, получаемых в результате их преобразования, от булевых формул для параллельно-последовательных контактных схем состоит в том, что предлагаемые формулы описывают структуру «вычислительного» блока «схемы», но не описывают его функционирование.

Однако этот недостаток не уменьшает важности введения понятия «объединенная формула», так как до сих пор отсутствовали формульное представление структуры и формальные методы ее минимизации для непараллельно-последовательных лестничных схем и структурированных ГСА, реализующих автоматы с памятью.