Глава 4

Алгоритмические модели автоматов

После выбора структурной модели автомата, вида кодирования его состояний, построения соответствующего графа переходов и анализа его поведения, например с помощью графа достижимых маркировок, возникает вопрос о выборе и построении алгоритмической модели, реализующей этот граф.

В настоящей работе рассматриваются следующие алгоритмические модели: СБФ, ФС, ГСА, ГП и матрицы переходов.

При этом необходимо отметить, что принятый вариант кодирования ограничивает разновидности алгоритмических моделей, которые при этом можно применять. Так, например, при многозначном кодировании булевы формулы не могут быть использованы.

4.1. Системы булевых формул (СБФ)

4.1.1. Построение СБФ при логарифмическом кодировании

Рассмотрим метод построения формул переходов и выходов при использовании логарифмического кодирования.

В этом случае число формул переходов равно $]\log_2 s[$, а число формул выходов равно числу выходов M. Получающиеся в этом случае формулы в дизъюнктивной нормальной форме весьма сложны, так как каждая конъюнкция содержит все переменные, кодирующие состояния, а каждая из этих переменных входит в формулу одновременно как в прямой, так и инверсной форме. Это приводит к повторности булевых формул по этим переменным, в общем случае мало уменьшающейся даже после вынесения повторяющихся переменных за скобки.

Формулы переходов для каждой переменной, кодирующей состояния, строятся отдельно. Для построения такой формулы для переменной y_i необходимо выделить все вершины, в которых эта переменная закодирована единицей, и построить дизъюнкцию (по всем этим вершинам) конъюнкций, каждая из которых состоит из формулы, помечающей дугу (петлю), входящую в вершину с $y_i = 1$, и конъюкции, соответствующей коду вершины, из которой исходит указанная дуга (петля).

Формула для каждого из выходов строится отдельно. Для автоматов Мура для переменной z_j формула выходов является дизъюнкцией (по всем вершинам, в которых $z_j = 1$), состоящей из конъюкций, соответствующих кодам состояний указанных вершин.

Для автоматов Мили формула выходов является дизъюнкцией (по всем дугам (петлям), на которых $z_j = 1$), включающей в себя конъюнкции, каждая из которых состоит из формулы, помечающей рассматриваемую дугу (петлю), и конъюнкции, соответствующей коду вершины, из которой исходит указанная дуга (петля).

Пример 4.1. Построить булеву формулу для ГП (рис. 4.1) при использовании логарифмического кодирования.

Закодируем состояние «1» кодом 0, а состояние «2» — кодом 1 (рис. 4.2). При этом $y' = x_1\bar{y} \vee \bar{x}_2 y$. Сложность формулы, определяемая числом букв в ее правой части, равна 4. Для проверки построим ГП по этой формуле.

При $y = 0$ $y' = x_1$. Тогда при $x_1 = 0$ $y' = 0$, а при $x_1 = 1$ $y' = 1$. При $y = 1$ $y' = x_2$. Тогда при $x_2 = 0$, а при $x_2 = 1$ $y' = 0$. Таким образом, при $x_1 = 1$ имеет место переход 0 —> 0; при x_1— переход 0 —> 1; при $x_2 = 0$ — переход 1 —> 1, а при $x_2 = 1$ — переход 1 —> 0.

В рассмотренном примере число вершин s в исходном ГП равно 2^i где (= 1. Поэтому СБФ, построенная по этому графу, реализует новый ГП с тем же числом вершин. Если в исходном графе число вершин $s \neq 2^i$, то СБФ, построенная по этому графу, реализует новый ГП с числом вершин $s_1 = 2^{\lceil \log_2 s \rceil}$. Таким образом, в этом случае $s_1 > s$

Изложенный метод построения СБФ по исходному графу переходов обеспечивает построение нового графа, в котором все вновь вводимые вершины, количество которых равно $2^{\lceil \log_2 s \rceil} - s$, безусловно связаны с вершиной 00... 0, в то время как остальная часть нового графа (ядро) совпадает с исходным ГП. В силу полноты по переходам каждой вершины исходного графа вершины ядра не имеют переходов во вновь введенные вершины.

Пример 4.2. Построить формулы переходов для ГП (рис. 4.3).

Закодируем состояния автомата следующим образом: «1» — 00; «2» — 01; «3» — 10 (рис. 4.4).

При этом

$$y'_1 = \bar{x}_2 x_3 \bar{y}_1 y_2 \vee x_2 y_1 \bar{y}_2;$$

$$y'_2 = x_1 \bar{y}_1 \bar{y}_2 \vee (\bar{x}_2 \bar{x}_3 \vee x_2 x_3) \bar{y}_1 y_2 = (x_1 \bar{y}_2 \vee (\bar{x}_2 \bar{x}_3 \vee x_2 x_3) y_2) \bar{y}_1.$$

Сложность реализации, определяемая суммарным числом букв в правых частях этих формул, равна 15.

Построим по найденной СБФ граф переходов. В этом графе ядро совпадает с исходным ГП, в то время как при $y_1 = y_2 = 1$ $y_1' = y_2' = 0$. Дугу, соответствующую безусловному переходу из состояния «4» (код — 11) в состояние «1» (код — 00), будем помечать символом 1 (рис. 4.5).

Сложность СБФ при свободном кодировании зависит от назначения кодов. Выполним их назначение следующим образом: «1» — 00; «2» — 01; «3» — 11. Используя рассматриваемый метод, получим:

$$y_1' = (\bar{x}_2 x_3 \bar{y}_1 \vee x_2 y_1) y_2;$$

$$y_2' = (x_1 \bar{y}_2 \vee (\bar{x}_2 \bar{x}_3 \vee x_2 x_3) y_2) \bar{y}_1 \vee y_1'.$$

Сложность этой системы, так же как и в предыдущем случае, равна 15. Метод назначения кодов, минимизирующий сложность СБФ, не известен, и поэтому обычно используются эвристики.

Минимизация формул переходов. Рассмотрим метод, позволяющий при выбранном назначении кодов исходного ГП минимизировать сложность СБФ.

Суть метода состоит в том, что по закодированному исходному ГП строится кодированная таблица переходов в форме не полностью определенной таблицы истинности. Используя, например, карты Карно, получим минимизированную систему.

Применяя изложенный подход к исходному ГП, рассмотренному в предыдущем примере, при первом варианте назначения кодов получим:

$$y_1' = \bar{x}_2 x_3 y_2 \vee x_2 y_1;$$
$$y_2' = x_1 \bar{y}_1 \bar{y}_2 \vee (\bar{x}_2 \bar{x}_3 \vee x_2 x_3) y_2.$$

В этом случае сложность системы равна 13.

Особенность изложенного подхода состоит в том, что для вновь вводимых вершин разрешены переходы (в отличие от предыдущего метода) не только в вершину с кодом 00 ... 0, но и в любую другую. ГП, построенный по найденной системе формул, приведен на рис. 4.6.

Обратим внимание на тот факт, что приведенные на рис. 4.5 и 4.6 графы переходов эквивалентны в том смысле, что при любом начальном состоянии при определенных условиях каждый из них может перейти в заданное «ядро» (рис. 4.4). Если заданный граф доопределить так, что в вершине «11» он будет иметь петлю, помеченную единицей, то при начальном состоянии в этой вершине он никогда не сможет перейти в заданное «ядро». Отметим, что и в этом случае может быть построена СБФ, содержащая 15 букв.

Таким образом, в зависимости от принятого доопределения исходного графа переходов имеет место различное влияние начального состояния на дальнейшее поведение автомата.

Более глубоко вопрос о влиянии начального состояния на динамическое поведение системы рассмотрен в [301].

Неоднозначность кодирования состояний. Первый метод нахождения СБФ, изложенный выше, весьма прост, но строит достаточно сложные системы. Второй метод трудоемок, но позволяет оптимизировать сложность СБФ. Рассмотрим третий метод, занимающий промежуточное положение по сравнению с указанными методами. Отличие этого метода от предыдущих состоит в назначении для некоторых состояний автомата не одного кода, а сразу нескольких.

Для рассматриваемого примера выполним следующее назначение кодов: «1» — 00; «2» — 01 и 11; «3» — 10. При этом первому состоянию соответствует конъюнкция $\bar{y}_1 \bar{y}_2$, второму — $\bar{y}_1 y_2 \vee y_1 y_2 = y_2$, а третьему — $y_1 \bar{y}_2$. Записывая формулы не по единичным значениям переменных y_1 и

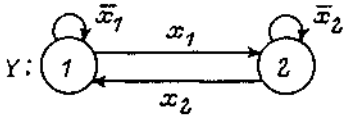

Рис. 4.1

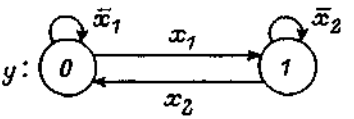

Рис. 4.2

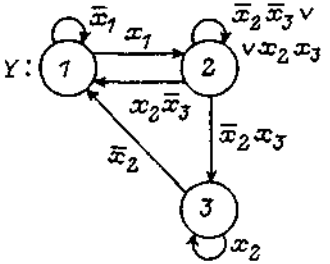

Рис. 4.3

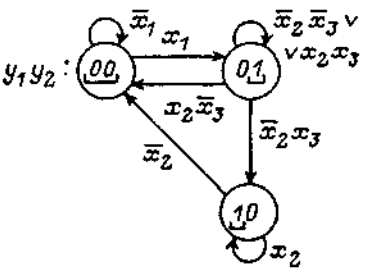

Рис. 4.4

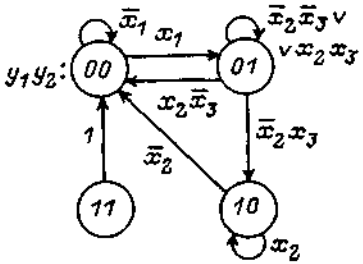

Рис. 4.5

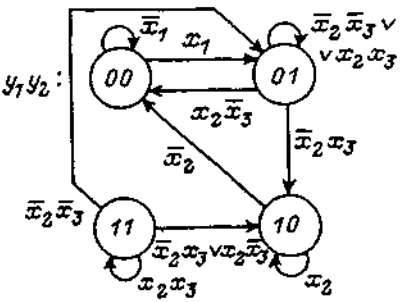

Рис. 4.6

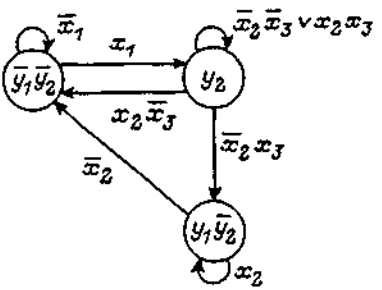

Рис. 4.7

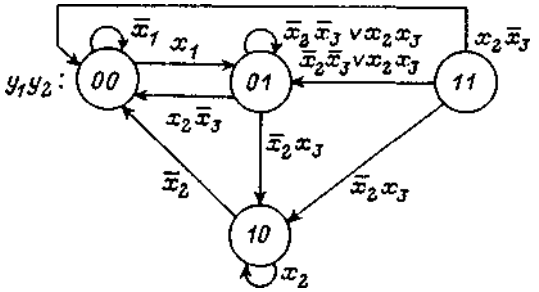

Рис. 4.8

y_2, как в первом методе, а по их безынверсным символам (рис. 4.7), получим:

$$y_1' = \bar{x}_2 x_3 y_2 \vee x_2 y_1 \bar{y}_2;$$

$$y_2' = x_1 \bar{y}_1 \bar{y}_2 \vee (\bar{x}_2 \bar{x}_3 \vee x_2 x_3) y_2 .$$

Сложность этой системы равна 14. ГП, построенный по найденной СБФ, приведен на рис. 4.8.

Отсутствие упоминания переменной у, во втором состоянии на рис. 4.7 свидетельствует о том, что значение этой переменной в этом состоянии безразлично.

Более простое решение может быть найдено, если код «11» назначить дважды: «1» — 00; «2» — 01 и 11; «3» — 10 и 11. При этом первому состоянию соответствует конъюнкция $\bar{y}_1 \bar{y}_2$, второму — y_2, а третьему — у,. Соответствующие этим выражениям значения переменных отмечены на рис. 4.4. В этом случае получается та же СБФ, что и найденная выше с помощью карты Карно.

Рассмотренный подход может быть назван кодированием неполными кодами — фрагментами кодов, различающих вершины.

4.1.2. Построение СБФ при унитарном кодировании

Выше было показано, что при использовании логарифмического кодирования получающиеся формулы в общем случае являются повторными по переменным состояний.

Переход к унитарному кодированию в его классической трактовке (каждой из s вершин исходного ГП присваивается .v-местный код, содержащий только одну единицу), увеличивая число формул перехода до л, не сокращает повторности формул. Более того, их повторность увеличивается.

Если в этом случае построить СБФ, используя первый метод, изложенный в предыдущем разделе, то ей соответствует ГП, содержащий 2^s вершин, из которых только s — рабочие. Все нерабочие вершины не взаимодействуют с ядром ГП, а «переходят» при определенных условиях также в нерабочую вершину, код которой состоит только из одних нулей.

Единственное достоинство этого вида кодирования — простота обнаружения ошибок.

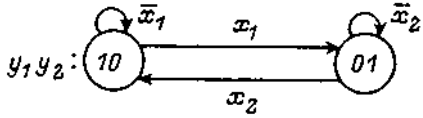

Рис. 4.9

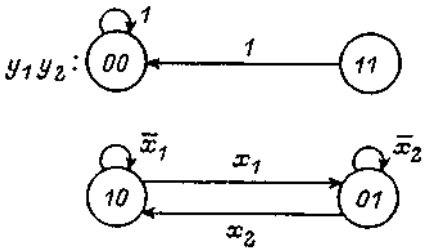

Рис. 4.10

Пример 4.3. Построить систему булевых формул для ГП (рис. 4.1) при использовании унитарного кодирования.

Закодируем состояние «1» кодом 10, а состояние «2» — кодом 01 (рис. 4.9). Тогда

$$y_1' = \bar{x}_1 y_1 \bar{y}_2 \vee x_2 \bar{y}_1 y_2; \qquad y_2' = x_1 y_1 \bar{y}_2 \vee \bar{x}_2 \bar{y}_1 y_2.$$

Сложность СБФ равна 12. Двухкомпонентный ГП, построенный по этой системе, приведен на рис. 4.10.

4.1.3. Построение СБФ при двоичном кодировании

В предыдущем разделе каждой вершине ГП присваивался один код. Присвоим l-й вершине наряду с рабочим кодом с одной единицей на i-й позиции также и нерабочие коды с большим числом единиц так, чтобы в результате минимизации логического выражения этой вершине можно было приписать только одну переменную y_i. При этом различным вершинам присваиваются в том числе и одинаковые нерабочие коды.

Это обеспечивает возможность записи s формул переходов и $\boldsymbol{m}$ формул выходов в бесповторной форме относительно переменных состояний. Если по полученной СБФ построить ГП, то все нерабочие вершины не взаимодействуют с «нулевой» вершиной, а «переходят» при определенных условиях в вершины, образующие ядро. Вершины ядра не имеют переходов в нерабочие вершины.

Таким образом, увеличение числа формул переходов с $]\log_2 s[$ до s по сравнению с методами, использующими логарифмическое и унитарное кодирование, позволяет при двоичном кодировании исключить в них повторность по переменным состояний. Повторность по входным переменным обычно незначительна либо отсутствует вовсе.

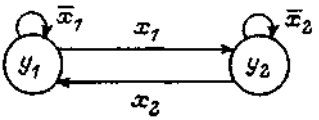

Рис. 4.11

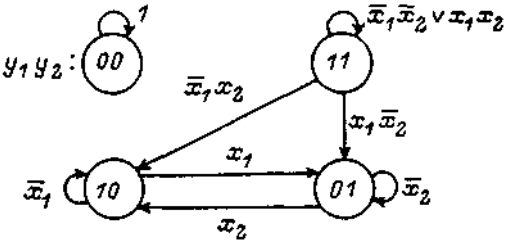

Рис. 4.12

Пример 4.4. Построить СБФ для ГП (рис. 4.1) при двоичном кодировании.

Присвоим состоянию «1» коды 10 и 11, а состоянию «2» — коды 01 и 11. При этом вершину «1» можно пометить символом y_1, а вершину «2» — символом y_2 (рис. 4.11), что позволяет записать следующие выражения:

$$y_1' = \bar{x}_1 y_1 \vee x_2 y_2; \qquad y_2' = x_1 y_1 \vee \bar{x}_2 y_2.$$

Двухкомпонентный ГП, построенный по этим формулам, приведен на рис. 4.12. Сложность СБФ равна 8.

Пример 4.5. Построить СБФ для ГП (рис. 4.3) при двоичном кодировании.

Обозначим вершины следующим образом: «1» — y_1; «2» — y_2 «3» — y_3 (рис. 4.13). При этом

$$y_1' = \bar{x}_1 y_1 \vee x_2 \bar{x}_3 y_2 \vee \bar{x}_2 y_3; \quad y_2' = x_1 y_1 \vee (\bar{x}_2 \bar{x}_3 \vee x_2 x_3) y_2;$$
$$y_3' = \bar{x}_2 x_3 y_2 \vee x_2 y_3.$$

Сложность этой СБФ равна 20. Все формулы бесповторны по переменным состояний.

Из последнего примера следует, что процесс построения СБФ с помощью этого метода является наиболее простым из рассмотренных в разд. 4.1, а структура каждой из получаемых формул также является наиболее простой. Это позволяет резко снизить вероятность ошибок при записи системы, несмотря на то что ее общая сложность при этом может и возрасти.

Алгоритмические модели второго порядка. После выбора системы булевых формул в качестве алгоритмической модели автомата возникает вопрос о выборе алгоритмической модели второго порядка — модели, соответствующей методу программной реализации этой системы. К таким моделям

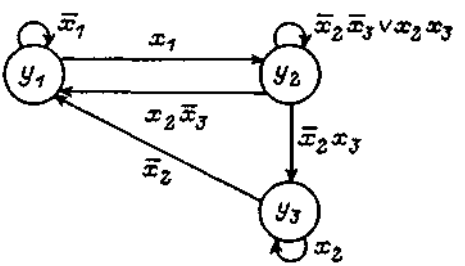

Рис. 4.13

относятся: таблицы истинности, булевы формулы, линейные, линеаризованные или плоскостные ГСА, пороговые реализации, арифметические полиномы, спектральные представления, интерполяционные полиномы, логические шкалы и т. д. [90].

Каждая из этих алгоритмических моделей после выбора соответствующей программной модели порождает программы, существенно отличающиеся по памяти и быстродействию (Приложения 1 и 2).

Примеры алгоритмических моделей второго порядка рассмотрены также в разд. 4.3.

4.2. Функциональные схемы

4.2.1. Триггерные схемы и их использование

Простейшие триггерные схемы являются автоматами без выходного преобразователя и имеют два состояния (0 и 1) и два входа: R (от «Reset» — сброс) и 5 (от «Set» — установка).

Построим кодированные таблицы переходов (КТП) для следующих типов триггеров: *RS, R,* 5, *E* и *JK.* При этом буква *J* соответствует слову Jerk (толкать), а буква K — слову Kill (убивать).

Эти триггеры имеют одинаковое поведение при всех значениях входов, за исключением случая $R = S = 1$. При Л = 5 = 0 предыдущее состояние сохраняется; при $R = 0$, $S = 1$ осуществляется установка выходного сигнала $(y' = 1)$ вне зависимости от предыдущего состояния; при $R=1$, $S = 0$ осуществляется сброс выходного сигнала $(y' = 0)$ вне зависимости от предыдущего состояния.

Для *RS*-триггера подача набора $R = S = 1$ запрещена. При этом выход не определен: $y' = —$. В качестве примера триггера этого типа можно привести реле типа РПС (реле поляризованное слаботочное).

Для R-триггера при $R = S = 1$ отдается приоритет сигналу сброса $(/ = 0)$; для S-триггера — сигналу установки $(y' = 1)$; для E-триггера предыдущее состояние сохраняется $(y' = y)$; для *JK*-триггера при $(J = S) = (K = R) = 1$ предыдущее состояние инвертируется $(y' = \bar{y})$.

Кодированные таблицы переходов для этих типов триггеров приведе-

Таблица 4.1

$R(K)$	$S(J)$	y	y'_{RS}	y'_{R}	y'_{S}	y'_{E}	y'_{JK}
0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1
0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0
1	1	0	—	0	1	0	1
1	1	1	—	0	1	1	0

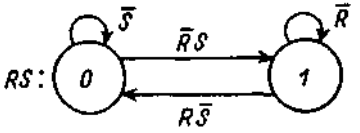

Рис. 4.14

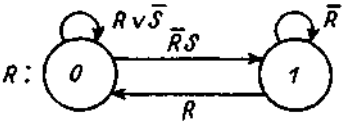

Рис. 4.15

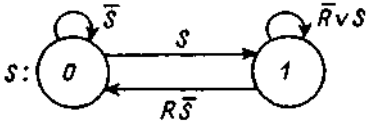

Рис. 4.16

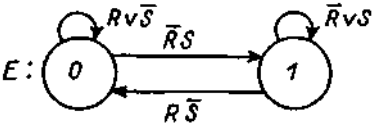

Рис. 4.17

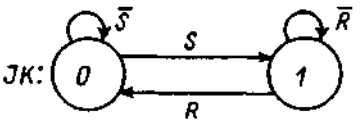

Рис. 4.18

Построим по этой таблице формулы и ГП для всех типов триггеров.

Для *RS*-триггера формула, описывающая его функционирование, не может быть построена, а ГП строится (рис. 4.14). Полученный граф не полон по переходам.

R-триггер описывается формулой $y' = \bar{R}(S \vee y)$ и ГП (рис. 4.15).

S-триггер описывается формулой $y' = S \vee \bar{R}y$ и ГП (рис. 4.16).

E-триггер описывается формулой $y' = (S \vee y)\bar{R} \vee Sy$ и ГП (рис. 4.17).

JK-триггер описывается формулой $y' = S\bar{y} \vee \bar{R}y = J\bar{y} \vee \bar{K}y$ и ГП (рис. 4.18).

Метод синтеза схем на триггерах изложен в [44, 86]. Однако этот метод является табличным и связан с доопределением значений функций (формул) возбуждения триггеров, что приводит к большой трудоемкости его использования. При этом отметим, что возможность доопределения формул возбуждения, используемая в этом методе, обеспечивает их оптимизацию.

Другой метод, основанный на решении логических уравнений, предложен в [336]. Ниже автором излагается простой аналитический метод синтеза схем на триггерах, базирующийся на других принципах. При этом оптимизация формул возбуждения исключается, однако появляется возможность проводить синтез схем даже большой размерности вручную.

В литературе [44] *JK*-триггер называется универсальным, однако трактовка этого названия с точки зрения теории переключательных схем [47] обычно (за исключением [1]) не приводится.

Это название связано с тем, что в правой части формулы, описывающей функционирование *JK*-триггера, переменная *y* встречается как в прямой, так и инверсной форме, что позволяет определять формулы

возбуждения его входов с помощью универсального подхода, а именно путем использования разложения Шеннона [46] для каждой формулы переходов автомата. Это разложение имеет следующий вид:

$$f(x_1, ..., x_i, ..., x_n) = \bar{x}_i f(x_1, ..., x_i = 0, ..., x_n) \vee x_i f(x_1, ..., x_i = 1, ..., x_n).$$

Если формулу перехода y_i' разложить по переменной y_i, то подформула, связанная знаком конъюнкции с переменной $\bar{y}_i$, является формулой возбуждения входа J i-го триггера, а подформула, связанная таким же образом с переменной y_i, является инверсией формулы возбуждения входа K того же триггера [1].

Пример 4.6. Построить формулы возбуждения входов JK-триггеров для автомата, рассмотренного в примере *4.4* (разд. 4.1).

Так как JK-триггер описывается формулой $y' = J\bar{y} \vee \bar{K}y$, а формулы переходов имеют вид:

$$y_1' = \bar{x}_1 y_1 \vee x_2 y_2 = x_2 y_2 \bar{y}_1 \vee (\bar{x}_1 \vee x_2 y_2) y_1;$$
$$y_2' = x_1 y_1 \vee \bar{x}_2 y_2 = x_1 y_1 \bar{y}_2 \vee (x_1 y_1 \vee \bar{x}_2) y_2,$$

то

$$J_1 = x_2 y_2; \quad K_1 = x_1 (\bar{x}_2 \vee \bar{y}_2); \quad J_2 = x_1 y_1; \quad K_2 = (\bar{x}_1 \vee \bar{y}_1) x_2.$$

Изложенный подход может быть использован при любом типе бинарного кодирования состояний автомата. Он наиболее целесообразен при логарифмическом кодировании, при котором функции переходов обычно немонотонны и обладают существенной повторностью по переменным, кодирующим состояния.

При двоичном кодировании состояний формулы переходов положительно монотонны (безынверсны) и бесповторны по переменным состояний и поэтому в этом случае универсальным становится S-триггер, содержащий в формуле, описывающей его функционирование, переменную y, входящую только в одну из двух конъюнкций, а остальные переменные (R и S) в этих конъюнкциях между собой не связаны.

R- и E-триггеры на роль универсального (в указанном смысле) триггера не подходят, так как представление формул переходов в виде, соответствующем формулам, описывающим их функционирование, в аналитической форме весьма сложно.

Пример 4.7. Построить формулы возбуждения входов S-триггеров для автомата, рассмотренного в примере *4.4* (разд. 4.1).

Так как S-триггер описывается формулой $y' = S \vee \bar{R}\, y$, а формулы переходов имеют вид:

$$y_1' = \bar{x}_1 y_1 \vee x_2 y_2;$$
$$y_2' = x_1 y_1 \vee \bar{x}_2 y_2,$$

то

$$R_1 = x_1; \quad S_1 = x_2 y_2; \quad R_2 = x_2; \quad S_2 = x_1 y_1.$$

В ряде случаев возникает проблема синтеза универсальных JK- и S-триггеров на базе триггеров других типов. Так, например, в библиотеке

функциональных элементов системы «Selma-2» имеется только один тип двухвходового триггера — *R*-триггер [60].

Построим универсальные триггеры в базисе *R*-триггера. В силу того что в данном случае предложенный аналитический метод не может быть использован, применим табличный метод [86].

Табл. 4.2 является таблицей возбуждения *R*-триггера.

Таблица 4.2

y	y'	R	S
0	0	$a_i \vee b_i$	a_i
0	1	0	1
1	0	1	—
1	1	0	—

Здесь a_i, $b_i \in \{0, 1\}$. С помощью этой таблицы по кодированной таблице переходов (табл. 4.3) построим формулы возбуждения входов триггеров:

$$R_S = R\bar{S}; \quad S_S = S; \quad R_{JK} = yK; \quad S_{JK} = J.$$

Таблица 4.3

y	$R(K)$	$S(J)$	y'_S	R_S	S_S	y'_{JK}	R_{JK}	S_{JK}
0	0	0	0	$a_0 \vee b_0$	a_0	0	$a_0 \vee b_0$	a_0
0	0	1	1	0	1	1	0	1
0	1	0	0	$a_1 \vee b_1$	a_1	0	$a_1 \vee b_1$	a_1
0	1	1	1	0	1	1	0	1
1	0	0	1	0	—	1	0	—
1	0	1	1	0	—	1	0	—
1	1	0	0	1	—	0	1	—
1	1	1	1	0	—	0	1	—

Для доказательства правильности построения этих формул осуществим их подстановку в формулу *R*-триггера:

$$y' = \bar{R}_S (S_S \vee y) = \overline{R\bar{S}}(S \vee y) = (\bar{R} \vee S)(S \vee y) =$$
$$= \bar{R}S \vee \bar{R}\, y \vee S \vee Sy = S \vee \bar{R}\, y;$$
$$y' = \bar{R}_{JK} (S_{JK} \vee y) = \overline{yK}(J \vee y) = (\bar{y} \vee \bar{K})(J \vee y) =$$
$$= \bar{y}J \vee \bar{K}\, J \vee y\, \bar{K} = \bar{y}\, J \vee y\, \bar{K}.$$

Схемы, построенные по полученным формулам возбуждения, приведены на рис. 4.19 (для *S*-триггера) и на рис. 4.20 (для *JK*-триггера).

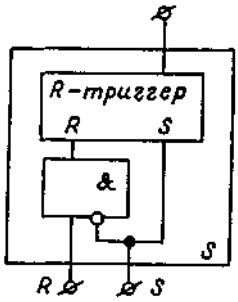

Рис. 4.19

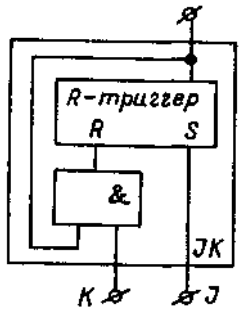

Рис. 4.20

Использование этих схем позволяет применять изложенный аналитический метод синтеза и для построения схем в базисе *R*-триггеров.

Кроме рассмотренных при программной реализации функциональных схем могут применяться еще два универсальных метода их синтеза с использованием одновходовых триггеров.

1. При использовании *D*-триггеров (элементов задержки на такт) каждый из них в данном случае реализуется одновходовым логическим элементом ИЛИ, осуществляющим переобозначение соответствующей внутренней переменной $(y_i = y_i')$, которое выполняется после реализации всех формул переходов, что на схеме отображается нумерацией элементов, указывающей порядок их реализации. При этом новое значение внутренней переменной передается в схему в следующем программном цикле, что и обеспечивает задержку на «такт».

При этом для примера *4.4* (разд. 4.1) получим следующую систему булевых формул:

$$y_1' = \bar{x}_1 y_1 \vee x_2 y_2; \quad y_2' = x_1 y_1 \vee \bar{x}_2 y_2; \quad y_1 = y_1'; \quad y_2 = y_2',$$

которая может быть упрощена следующим образом

$$y_1' = \bar{x}_1 y_1 \vee x_2 y_2; \quad y_2 = x_1 y_1 \vee \bar{x}_2 y_2; \quad y_1 = y_1'.$$

2. При использовании *T*-триггеров (счетных триггеров), структура каждого из которых в данном случае может быть описана формулой $y_i' = T_i \oplus y_i$, возможны два подхода к нахождению их функций возбуждения.

Первый из них основан на решении логических уравнений. При этом из уравнения $y_i' = T_i \oplus y_i$ следует, что $T_i = y_i' \oplus y_i$

Исходя из изложенного, для примера *4.4* (разд. 4.1) получим следующую систему булевых формул:

$$T_1 = (\bar{x}_1 y_1 \vee x_2 y_2) \oplus y_1; \quad T_2 = (x_1 y_1 \vee \bar{x}_2 y_2) \oplus y_2;$$

$$y_1 = T_1 \oplus y_1; \quad y_2 = T_2 \oplus y_2.$$

Второй подход основан на применении к каждой из формул переходов преобразования Артюхова—Шалыто [274]

$$y_i' = (\bar{y}_i \,\&\, y_i' (y_i = 0) \vee y_i \,\&\, \overline{y_i' (y_i = 1)}) \oplus y_i.$$

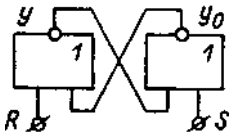

Рис. 4.21

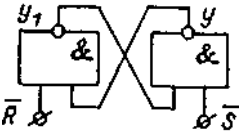

Рис. 4.22

Из сопоставления этого соотношения с формулой, описывающей счетный триггер, приведенной выше, следует, что, так же как и в [1],

$$T_i = \bar{y}_i \,\&\, y_i' \,(y_i = 0) \vee y_i \,\&\, \overline{y_i'(y_i = 1)}.$$

m, = y, & y.! (y,- = 0) v y, *& y;{y, =* 1).

При этом для примера *4.4* (разд. 4.1) получим следующую систему булевых формул:

$$T_1 = \bar{y}_1 x_2 y_2 \vee y_1 x_1 (\bar{x}_2 \vee \bar{y}_2); \qquad T_2 = \bar{y}_2 x_1 y_1 \vee y_2 x_2 (\bar{x}_1 \vee \bar{y}_1);$$

$$y_1 = T_1 \oplus y_1; \quad y_2 = T_2 \oplus y_2.$$

В заключение раздела отметим, что из преобразований Артюхова—Шалыто, приведенных в [274], следует, что

$$f \oplus x_i = \bar{x}_i f(x_i = 0) \vee x_i \overline{f(x_i = 1)};$$

$$f \oplus \bar{x}_i = \bar{x}_i \overline{f(x_i = 0)} \vee x_i f(x_i = 1).$$

Таким образом, для инвертирования половины столбца значений произвольной булевой функции / по переменной x_i, эту функцию необходимо сложить по модулю два с указанной переменной или ее инверсией. (Известно [1], что инвертирование всего столбца значений функций / выполняется при ее сложении по модулю два с единицей).

В заключение раздела рассмотрим вопрос о построении ГП для *RS*-триггеров, реализованных на элементах ИЛИ—НЕ (рис. 4.21) и И—НЕ (рис. 4.22) соответственно. При этом отметим, что если при программной реализации можно использовать одновыходные триггеры, то в аппаратуре для индикации переходных процессов [14] требуется применять двухвыходные триггеры.

Для триггера на элементах ИЛИ—НЕ справедливы соотношения:

$$y = \overline{R \vee y_0} = \bar{R}\,\bar{y}_0; \quad y_0 = \overline{S \vee y} = \bar{S}\,\bar{y}. \tag{4.1}$$

Отметим, что для одного выхода y эта схема является R-триггером:

$$y = \bar{R}\,\overline{\bar{S}\,\bar{y}} = \bar{R}\,(S \vee y).$$

По соотношениям (4.1) для y и y_0 построим КТП (табл. 4.4).

Построим ГП (рис. 4.23) по этой таблице. Из рассмотрения этого ГП следует, что если в качестве индикатора завершения переходных процессов использовать элемент ИЛИ, то на его выходе при переходе из одного рабочего состояния в другое (через транзитное состояние «00») формируется последовательность 1 —> 0 —> 1.

Таблица 4.4

y	y_0	R	S	y	y_0	y	y_0	R	S	y	y_0
0	0	0	0	1	1	1	0	0	0	1	0
0	0	0	1	1	0	1	0	0	1	1	0
0	0	1	0	0	1	1	0	1	0	0	0
0	0	1	1	0	0	1	0	1	1	0	0
0	1	0	0	0	1	1	1	0	0	0	0
0	1	0	1	0	0	1	1	0	1	0	0
0	1	1	0	0	1	1	1	1	0	0	0
0	1	1	1	0	0	1	1	1	1	0	0

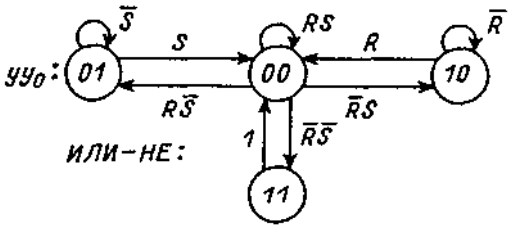

Рис. 4.23

Для триггера на элементах И—НЕ справедливы соотношения:

$$y = \overline{\overline{S}\,y_1} = S \vee \overline{y}_1; \quad y_1 = \overline{\overline{R}\,y} = R \vee \overline{y}. \tag{4.2}$$

Отметим, что для одного выхода y эта схема является S-триггером:

$$y = S \vee \overline{R \vee \overline{y}} = S \vee \overline{R}\,y.$$

По соотношениям для y и y_1 (4.2) построим кодированную таблицу переходов (табл. 4.5).

Таблица 4.5

y	y_1	R	S	y	y_1	y	y_1	R	S	y	y_1
0	0	0	0	1	1	1	0	0	0	1	0
0	0	0	1	1	1	1	0	0	1	1	0
0	0	1	0	1	1	1	0	1	0	1	1
0	0	1	1	1	1	1	0	1	1	1	1
0	1	0	0	0	1	1	1	0	0	0	0
0	1	0	1	1	1	1	1	0	1	1	0
0	1	1	0	0	1	1	1	1	0	0	1
0	1	1	1	1	1	1	1	1	1	1	1

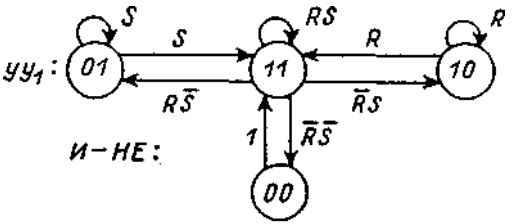

Рис. 4.24

Построим ГП (рис. 4.24) по этой таблице. Из рассмотрения этого ГП следует, что если в качестве индикатора использовать элемент И, то на его выходе при переходе из одного рабочего состояния в другое (через транзитное состояние «11») формируется последовательность О -» 1 -» 0.

Использование индикаторов завершения переходных процессов позволяет при аппаратной реализации строить схемы, работающие по реальным задержкам [13].

Из рассмотрения построенных графов становится ясно, почему для *RS*-триггеров входной набор $R = S = 1$ является запрещенным: вместо прямого и инверсного сигналов на выходах формируются одинаковые сигналы.

4.2.2. Анализ функциональных схем

Выше отмечалось, что для правильного построения функциональных схем либо их синтез, либо анализ должны выполняться с помощью формализованных методов.

В предыдущих разделах были изложены методы синтеза функциональных схем по ГП. Поэтому при программной реализации алгоритмов анализ функционирования схем можно не проводить, так как при правильном использовании указанных методов достаточно выполнить лишь семантический и синтаксический анализ ГП.

Однако на практике обычно функциональные схемы строятся эвристически, которые также эвристически и проверяются, что не позволяет быть полностью уверенным в правильности их работы.

Этот недостаток устраняется, если при эвристическом синтезе выполнить формализованный анализ — построение по функциональной схеме графа переходов. Рассмотрим пример.

Пример 4.8. Предположим, что по словесному заданию (пример *5.10)* эвристически построена функциональная схема на *R*-триггерах (рис. 4.25). Выполним анализ этой схемы. Для этого построим СБФ, описывающую схему:

$$y_1' = \bar{x}_3 (x_2 \vee y_1); \qquad y_2' = \bar{x}_3 (x_1 \vee y_2).$$

Заполним КТП по этой СБФ (табл. 4.6).

Таблица 4.6

y_1	y_2	x_1	x_2	x_3	y_1'	y_2'	y_1	y_2	x_1	x_2	x_3	y_1'	y_2'
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	1	0	0	1	0	0	0	1	0	0
0	0	0	1	0	1	0	1	0	0	1	0	1	0
0	0	0	1	1	0	0	1	0	0	1	1	0	0
0	0	1	0	0	0	1	1	0	1	0	0	1	1
0	0	1	0	1	0	0	1	0	1	0	1	0	0
0	0	1	1	0	1	1	1	0	1	1	0	1	1
0	0	1	1	1	0	0	1	0	1	1	1	0	0
0	1	0	0	0	0	1	1	1	0	0	0	1	1
0	1	0	0	1	0	0	1	1	0	0	1	0	0
0	1	0	1	0	1	1	1	1	0	1	0	1	1
0	1	0	1	1	0	0	1	1	0	1	1	0	0
0	1	1	0	0	0	1	1	1	1	0	0	1	1
0	1	1	0	1	0	0	1	1	1	0	1	0	0
0	1	1	1	0	1	1	1	1	1	1	0	1	1
0	1	1	1	1	0	0	1	1	1	1	1	0	0

Построим по табл. 4.6 матрицу переходов (табл. 4.7).

Таблица 4.7

	00	01	10	11
00	$\bar{x}_1\bar{x}_2 \vee x_3$	$x_1\bar{x}_2\bar{x}_3$	$\bar{x}_1 x_2\bar{x}_3$	$x_1 x_2\bar{x}_3$
01	x_3	$\bar{x}_2\bar{x}_3$	—	$x_2\bar{x}_3$
10	x_3	—	$\bar{x}_1\bar{x}_3$	$x_1\bar{x}_3$
11	x_3	—	—	$\bar{x}_3$

Этой матрице соответствует ГП (рис. 4.26). Из его рассмотрения следует, что, несмотря на простоту формул в СБФ, он весьма сложен и близок к полному графу. Это определяется малой связностью триггеров в схеме (несвязанным триггерам соответствует полный ГП).

Из рассмотрения этого ГП следует, что если заданную схему тестировать в соответствии с техническим заданием, то при положительных результатах обычно считают, что оно реализовано правильно.

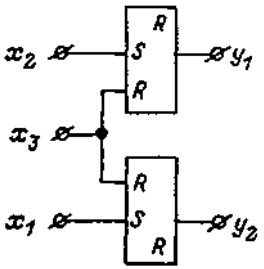

Рис. 4.25

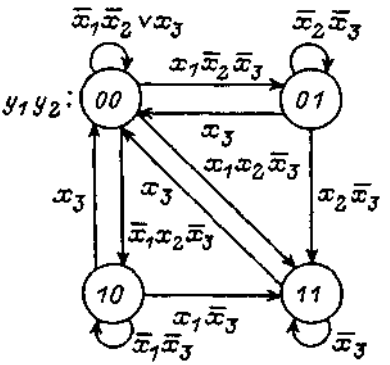

Рис. 4.26

Однако из построенного ГП следует, что эта схема обеспечивает возможность перехода в состояние «11», что в техническом задании не было оговорено.

Если одновременное появление выходных сигналов необходимо исключить, то схема должна быть откорректирована, что наиболее удобно осуществить изменением графа переходов.

Рассмотрим три подхода к корректировке ГП.

Первый подход состоит в переносе стрелок, ведущих в вершину «11», в другие вершины. Рассмотрим три варианта переноса стрелок, в каждом из которых объединяются дуги ($\bar{x}_2\bar{x}_3$ и $x_2\bar{x}_3$; $\bar{x}_1\bar{x}_3$ и $x_1\bar{x}_3$). Варианты отличаются переносом дуги с пометкой $x_1x_2\bar{x}_3$: в вершину «00» (рис. 4.27); в вершину «01» (рис. 4.28, приоритет открытия); в вершину «10» (рис. 4.29, приоритет закрытия).

Выбор того или иного варианта ГП требует обоснования. Выберем первый граф переходов, так как наиболее естественным является случай, когда при появлении одного и отсутствии всех остальных сигналов

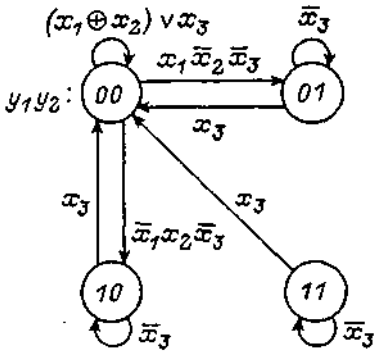

Рис. 4.27

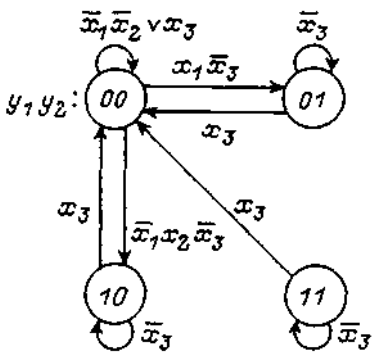

Рис. 4.28

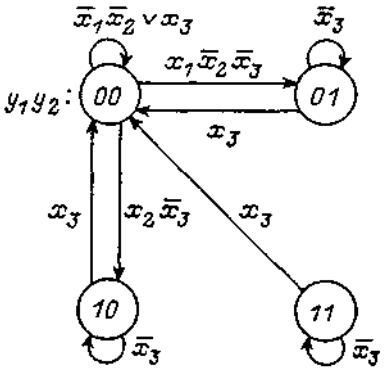

Рис. 4.29

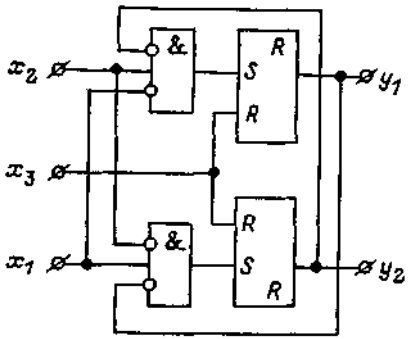

Рис. 4.30

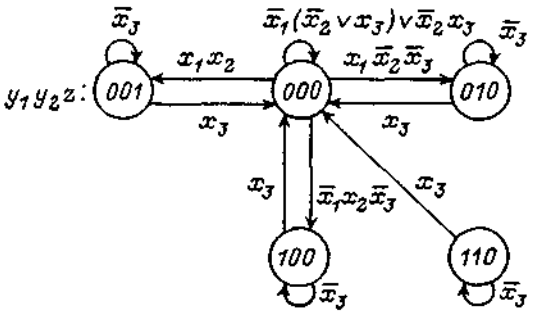

Рис. 4.31

осуществляется требуемый переход, а при всех других входных наборах состояние автомата сохраняется.

Воспользовавшись методом, изложенным в разд. 4.1.1, определим СБФ по ГП (рис. 4.27):

$$y_1' = (\bar{x}_1x_2\bar{y}_2 \vee y_1)\bar{x}_3; \qquad y_2' = (x_1\bar{x}_2\bar{y}_1 \vee y_2)\bar{x}_3.$$

Построим по этой СБФ схему (рис. 4.30) на *R*-триггерах, структура которых изоморфна реализуемым формулам.

Второй подход состоит не только в переносе стрелок, ведущих в «11», но и во введении дополнительного состояния и выходной переменной z, сигнализирующей о том, что две кнопки были нажаты одновременно (рис. 4.31).

Третий подход состоит в переходе от модели автомата без выходного преобразователя к модели автомата Мура (рис. 4.32). При этом формулы выходов имеют следующий вид: $z_1 = y_1\bar{y}_2$ и $z_2 = \bar{y}_1y_2$. Схема, реализующая этот автомат, приведена на рис. 4.33.

В заключение раздела отметим, что на практике часто возникает ситуация, в которой по техническому заданию эвристически строится

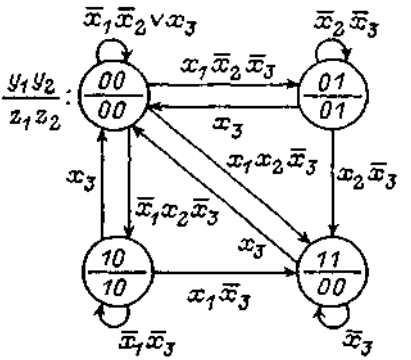

Рис. 4.32

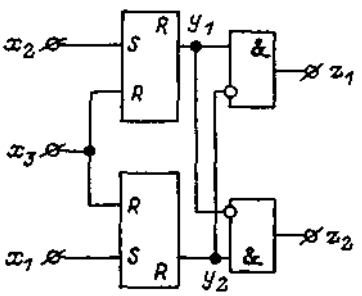

Рис. 4.33

функциональная схема, а затем по нему также эвристически строится ГП и утверждается, что этот ГП описывает работу схемы, так как она и ГП строились по одному заданию одним и тем же человеком.

В подавляющем большинстве случаев при формальной проверке эквивалентность этих моделей отсутствует. Это может быть связано с рядом причин, и в том числе с тем, что функциональной схеме соответствует полный по состояниям ГП, а эвристически построенный ГП содержит только рабочие состояния. При этом в зависимости от доопределения переходов из неиспользованных состояний могут быть построены различные ГП. Поэтому может возникнуть ситуация, в которой функциональной схеме соответствует простая СБФ, а эвристически построенному ГП с небольшим числом рабочих состояний — сложная СБФ.

Эвристически построенный ГП обычно не отражает всех функциональных возможностей схемы, а может использоваться лишь в качестве графа проверки некоторой части этих возможностей. Граф переходов, построенный по ФС формализованным методом, позволяет проанализировать все ее функциональные возможности.

Отметим также, что структурно простой функциональной схеме может соответствовать сложный ГП, и наоборот. Это свидетельствует о том, что при упоминании терминов «простой» и «сложный» необходимо уточнять, к какому аспекту — структурному или поведенческому — используемый термин относится.

4.2.3. Реализация графов переходов схемами из мультиплексоров

В разд. 4.2.1 был рассмотрен метод построения функциональных схем для автоматов с памятью при их программной реализации. При этом схемы строятся в базисе логических элементов и триггеров.

Необходимость построения функциональных схем при такой реализации возникает в случае, если они выступают в качестве языка программирования, как это имеет место для многих ПЛК и управляющих систем, и в частности для системы «Selma-2» [60].

Даже при использовании традиционных языков программирования, например СИ, построение функциональных схем на триггерах может быть весьма целесообразным, так как реализация булевых формул, описывающих такие схемы, более проста по сравнению с реализацией формул переходов, ввиду того что с помощью триггеров осуществляется декомпозиция этих формул (Приложение 5).

Однако функциональная схема на триггерах, построенная по ГП с помощью предложенного метода, обладает тем недостатком, что она не изоморфна этому графу.

Указанный недостаток при программной реализации может быть устранен с помощью подхода, предложенного автором совместно с В. Н. Кондратьевым [124].

Если в библиотеку программно реализованных элементов для построения ФС наряду с указанными выше логическими элементами входят также логические и цифровые мультиплексоры [260] (как это имеет место для системы «Selma-2»), то появляется возможность вложения реализуемого ГП, использующего многозначное кодирование, в одну из предлагаемых ниже стандартных схем. При этом отметим, что если элементы указанных типов в библиотеке отсутствуют, то их целесообразно ввести в нее.

Рассмотрим указанные схемы.

Логический мультиплексор (ЛМС) имеет m логических управляющих, 2^m цифровых информационных входов и один цифровой выход, а цифровой мультиплексор (ЦМС) — один цифровой управляющий и s цифровых информационных входов (s — число вершин в ГП), а также один цифровой выход.

На логические входы подаются константы 0 и 1, а на цифровые — константы $0, 1, 2, \ldots, s$. На цифровых выходах также формируются константы $0, 1, 2, \ldots, s$

Обратим внимание на отличие этих элементов от элементов, традиционно называемых мультиплексорами, все входы и выходы которых являются логическими (двоичными) [57].

При использовании стандартных схем предлагается осуществлять вложение в них ГП автоматов Мура.

Рассматриваются три варианта стандартных схем, первый из которых предполагает наличие ЛМС при $m_i = 1$, второй — п $m_i = n_i$, а третий — при $m_i = l_i$, где и, — число переменных в булевых формулах, помечающих дуги, исходящие из i-й вершины ГП; l_i — число дуг, исходящих из i-и вершины.

В первом случае стандартную схему можно описать следующим образом: комбинационная схема строится из логических элементов и реализует формулы, указанные на всех дугах ГП, за исключением петель; выходы этой схемы подключены к логическим входам ЛМС; выходы ЛМС подключены к информационным входам первого ЦМС, выход которого обратной связью соединен со своим управляющим входом и первыми информационными входами ЛМС (для реализации петель ГП), а прямой связью — с управляющим входом второго ЦМС, на информационные входы которого подаются десятичные эквиваленты значений выходных переменных в соответствующих вершинах ГП; выход второго ЦМС подключен ко входам преобразователя «десятичный код—двоичный код», осуществляющего преобразование

десятичного кода в двоичные выходные переменные; на вторые информационные входы ЛМС подаются десятичные эквиваленты номеров вершин, в которые осуществляются соответствующие переходы в ГП.

Оценим число элементов в такой схеме. Если число различных входных переменных в пометках дуг равно N, то число инверторов в схеме не превысит N. Если общее число букв во всех булевых формулах на дугах (за исключением петель) равно H, то число логических элементов, реализующих эти формулы, не превышает $H - L$, где L— число дуг (без учета петель) в ГП. Это объясняется тем, что для реализации i-й ($i = 1, ..., L$) формулы из h_i букв требуется не более двухвходовых $h_i - 1$ элементов без учета инверсий, которые реализуются в первом слое. Так как каждый переход (дуга) в графе реализуется одним ЛМС, то их общее число равно L. В силу того что в схему входят два ЦМС и один преобразователь, общее число элементов

$$Э_1 \leqslant N + (H - L) + L + 3 = N + H + 3.$$

Использование изложенного подхода позволяет реализовать ГП (рис. 4.34) схемой (рис. 4.35) из 11 элементов ($Э_1 < 3 + 6 + 3 = 12$).

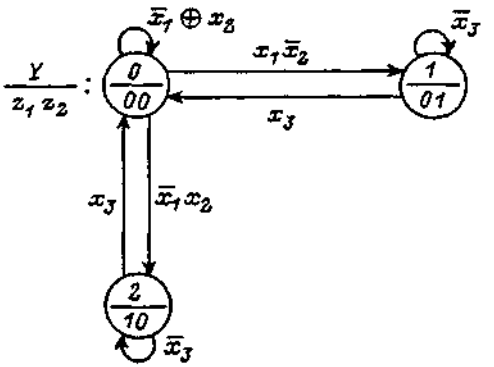

Рис. 4.34

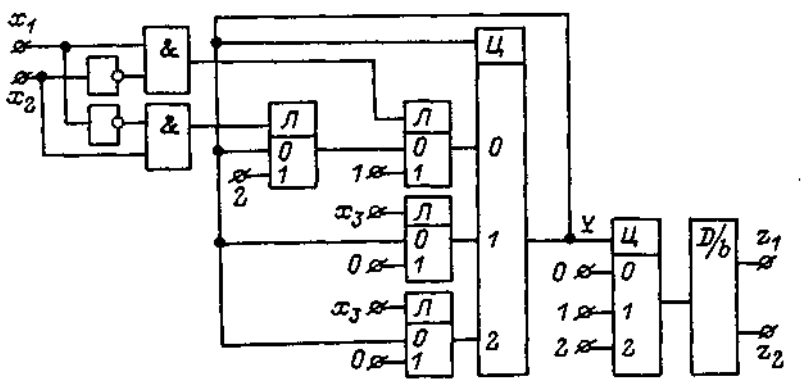

Рис. 4.35

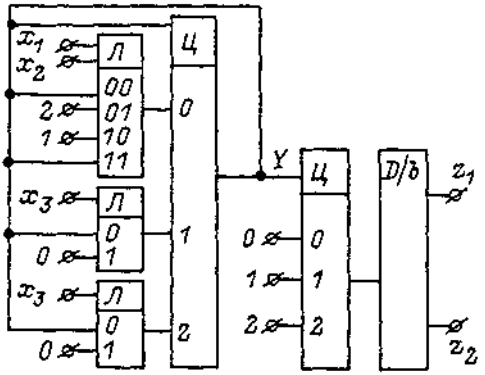

Рис. 4.36

Комбинационная схема в стандартной схеме рассматриваемого типа может быть упрощена, если противоречивость в ГП устраняется не ортогонализацией, а расстановкой приоритетов. При этом чем выше приоритет дуги, исходящей из j-й вершины ГП, тем ближе ЛМС, соответствующий этой дуге, располагается к j-му информационному входу ЦМС.

Вторая разновидность стандартных схем отличается от первой тем, что комбинационная схема из логических элементов не строится, а j-й ($j = 1, ..., s$) ЛМС реализует все булевы формулы, помечающие дуги, исходящие из j-й вершины ГП. Это удается осуществить ввиду того, что все указанные формулы взаимно ортогональны. Таким образом, число ЛМС в схеме равно s, а общее число элементов в этом случае равно $Э_2 = s + 3$

На рис. 4.36 приведена схема из шести элементов, реализующая ГП с $s = 3$ (рис. 4.34).

Третий вариант стандартных схем является промежуточным между рассмотренными. Он содержит, как и вторая схема, s логических мультиплексоров, но в силу того что в этом случае число информационных входов (i-го ЛМС ограничено и равно $M = 2^{\lceil \log_2 I_i \rceil}$, то приходится строить комбинационную схему так же, как и в первом случае. Свободные информационные входы подключены к выходу ЦМС, что обеспечивает сохранение состояний автомата. Число элементов в этом случае оценивается следующим образом:

$$Э_3 \leqslant N + (H - L) + s + 3.$$

ГП (рис. 4.37) реализуется схемой (рис. 4.38) из 10 элементов ($Э_3 \leqslant 4 + (7 - 4) + 3 + 3 = 13$).

Сравнивая рассмотренные схемы, можно утверждать, что второй тип схем является наиболее однородным, а первый — наиболее просто моделируется программой на языке СИ при использовании алгоритмической конструкции, соответствующей автомату Мура второго рода и построенной с помощью двух конструкций `switch` (разд. 5.1.2). При этом каждой конструкции `switch` в программе соответствует ЦМС в схеме,

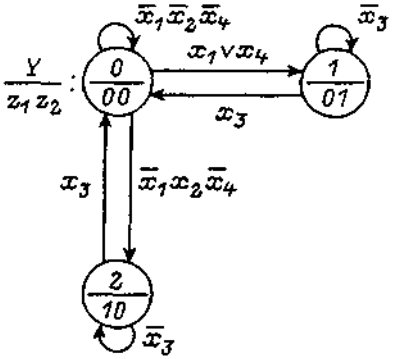

Рис. 4.37

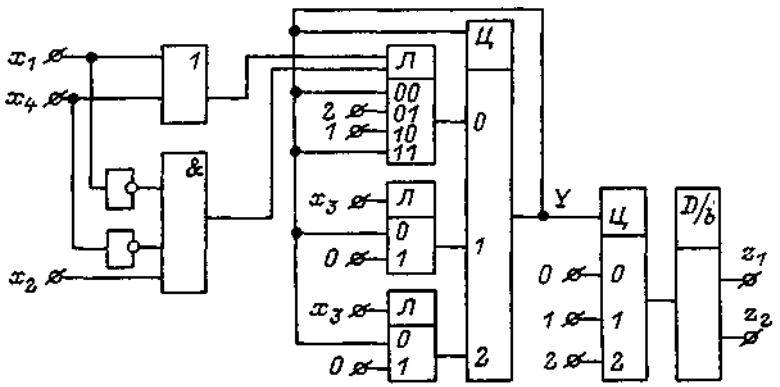

Рис. 4.38

а каждой конструкции `if` — соответствующий ЛМС. Это позволяет моделировать функциональную схему изоморфной программой.

Рассмотрим вопрос о реализации алгоритмов, заданных несколькими ГП, например двумя. В этом случае для упрощения схемы обмен между ГП осуществляется не с помощью многозначных переменных состояний, а за счет введения дополнительных двоичных выходных (внутренних) переменных.

Число таких переменных равно числу состояний, используемых во взаимных блокировках. Каждая из этих переменных принимает значение «1» в одном состоянии и «0» во всех остальных состояниях, т. е. используется двоичное кодирование дополнительно вводимых разрядов выходного кода в отличие от двоичного принудительного кодирования для остальных разрядов.

На рис. 4.39 приведены ГП с дополнительно введенными выходными сигналами z_5 (для первого ГП) и z_6 (для второго ГП). Эти сигналы обеспечивают связь между графами переходов. На рис. 4.40 приведена функциональная схема, реализующая указанные ГП.

В заключение раздела отметим, что рассмотренный подход обеспечивает построение более компактных схем по сравнению со схемами на

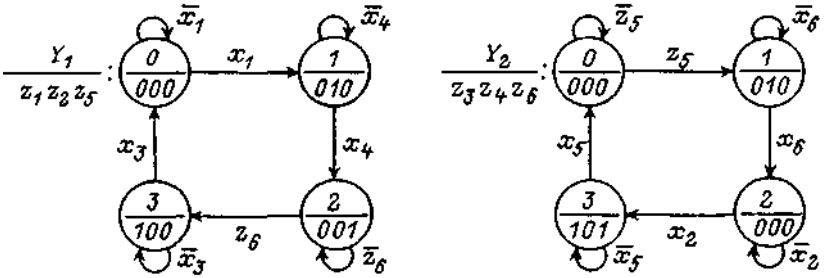

Рис. 4.39

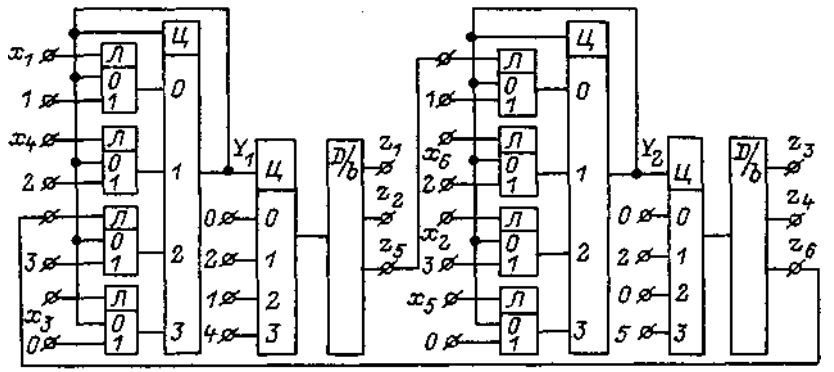

Рис. 4.40

двоичных мультиплексорах [57], при использовании которых многозначное кодирование не может быть применено.

Предложенный подход следует применять во всех случаях, когда параметры транслируемых по этим схемам программ не превосходят ограничений (если они имеются) на эти параметры. В противном случае должны применяться функциональные схемы с обратными связями или построенные на триггерах.

Еще раз подчеркнем, что в структуре первого типа непротиворечивость может быть обеспечена за счет порядка подключения входных переменных или комбинационных схем к логическим входам ЛМС, соединенных последовательно. При подключении к последнему логическому мультиплексору цепочки обеспечивается наибольший приоритет, а при подключении к первому — наименьший. Это позволяет отказаться от обеспечения ортогональности в ГП и упростить комбинационные схемы, используемые в структуре. Для достижения хорошей читаемости приоритеты в ГП должны быть отражены с помощью определенной символики, например штрихов, помечающих дуги графа, число которых совпадает с номером приоритета.

4.3. Граф-схемы алгоритмов (ГСА)

При построении ГСА будем считать (за исключением случаев, оговариваемых особо), что если оператор «Ввод» и оператор «Вывод» в них в явном виде не используются, то считается, что по умолчанию они размещены соответственно в начале и в конце тела каждой граф-схемы.

4.3.1. Реализация булевых формул ГСА

Рассмотрим первоначально методы построения простейших ГСА (ПГСА) для одной булевой функции.

Будем называть ГСА простейшей, если она содержит лишь условные вершины, каждая из которых проверяет одну переменную, и операторные вершины, в которых осуществляется присвоение констант 0 и 1 выходной переменной.

Если булева функция задана формулой в базисе И, ИЛИ, НЕ из h букв, то формульный метод [68, 271] позволяет реализовать ее линейной и планарной ПГСА [88], содержащей $h+2$ вершины, из которых h вершин — условные. На рис. 4.41 представлена ПГСА, реализующая формулу $y = (x_1x_2 \vee x_3)x_4$

Оптимизация линейной ПГСА по числу путей (среднему быстродействию) может быть выполнена с помощью метода [88, 273]. Суть этого метода состоит в изменении, в соответствии с соотношениями, приведенными в [88, 273], порядка записи реализуемой формулы. Например, если формулу, рассмотренную выше, записать как $y = x_4(x_3 \vee x_2x_1)$ и построить по ней линейную ПГСА (рис. 4.42), то число путей в ней сокращается. При этом уменьшается также и число точек объединения дуг графа, располагающихся на его остове.

Рассмотренные методы строят оптимальные по числу вершин ПГСА для бесповторных в указанном базисе формул. Для повторных формул более эффективными являются метод каскадов Поварова [209] и канонический метод Блоха [19], которые, естественно, могут применяться и для

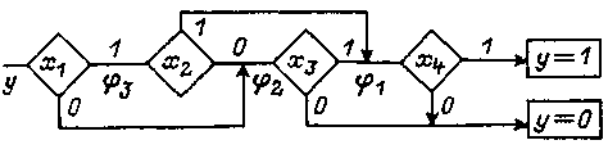

Рис. 4.41

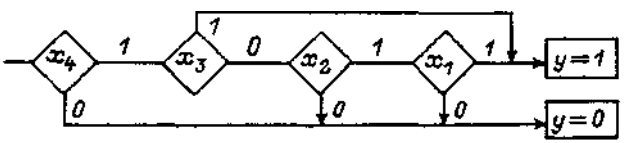

Рис. 4.42

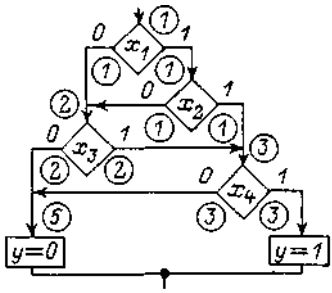

Рис. 4.43

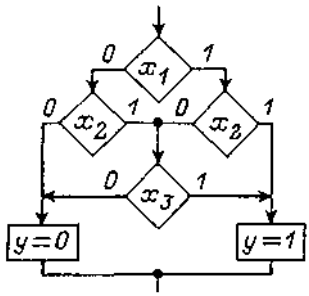

Рис. 4.44

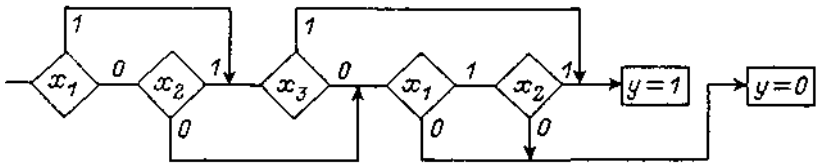

Рис. 4.45

бесповторных формул. Эти методы базируются на разложении Шеннона, приведенном в разд. 4.2.1.

Первый метод является аналитическим, а второй — табличным. Оба метода при фиксированном порядке переменных строят одинаковые ПГСА, однако второй метод более удобен для оптимизации числа вершин. На рис. 4.43 приведена ПГСА, реализующая формулу $y = (x_1x_2 \vee x_3)x_4$ которая построена каноническим методом.

Эти методы для бесповторных формул строят плоскостные ПГСА, изоморфные линейным граф-схемам, синтезируемым формульным методом, а для повторных формул обеспечивают оптимизацию числа условных вершин. Поэтому применять эти методы целесообразно лишь для реализации повторных формул.

Применение формульного метода для таких формул может приводить к неоднозначным результатам в тех случаях, когда за время вычисления формулы значения повторяющихся переменных изменяются.

Изменение значений переменных за время прохода одного пути, содержащего одинаковые проверки, может быть названо риском по аналогии с риском в комбинационных схемах [13].

Для устранения риска необходимо либо реализовать процедуру общения с «внешним миром» с использованием буфера, либо реализовывать формулу таким образом, чтобы при прохождении одного пути переменные повторно не опрашивались. Это всегда может быть обеспечено с помощью метода каскадов или канонического метода. Отметим также, что число путей в ГСА при использовании этих методов не более 2^n, в то время как при применении формульного метода для повторных формул их число может быть больше этой величины.

На рис. 4.44 приведена ПГСА, реализующая формулу $y = (x_1 \vee \vee x_2)x_3 \vee x_1x_2$ с помощью метода каскадов, а на рис. 4.45 — формульным

методом. В первом случае при четырех условных вершинах число путей равно шести, а во втором — при пяти условных вершинах — одиннадцати.

Для формулы $y = (x_1 \vee x_2 \vee x_3)x_4 \vee x_1x_2x_3$ в первом случае эти показатели равны шести и восьми, а во втором — семи и четырнадцати.

4.3.2. Реализация булевых формул структурированными ГСА

Перейдем к рассмотрению методов построения структурированных ПГСА (СПГСА).

Если простейшую ГСА, построенную одним из рассмотренных методов, преобразовать в дерево, то она будет структурированной. Этот метод называется методом дублирования [83]. Построенная СПГСА содержит только вложенные блоки «полный выбор». Оптимизация числа вершин в этом случае обеспечивается при построении ее на базе ПГСА с минимальным числом путей. На рис. 4.46 приведена СПГСА, построенная за счет преобразования граф-схемы указанного типа (рис. 4.42) в дерево.

Дальнейшая оптимизация СПГСА этого класса по числу вершин достигается за счет многократного «выноса вверх» некоторых операторных вершин по графическим правилам (рис. 4.47). На рис. 4.48, 4.49 приведены СПГСА, полученные за счет указанных преобразований граф-схемы (рис. 4.46). Таким образом, изложенный подход позволяет использовать в структуре как вложенные блоки «неполный выбор», так и последовательное соединение операторных вершин с блоками этого типа.

Так как число вершин в СПГСА при использовании рассматриваемого метода определяется числом путей, то верхняя оценка числа вершин в этом случае является показательной функцией от числа букв в реализуемой формуле [88, 272].

Дальнейшее расширение вариантов допустимого соединения блоков обеспечивается при использовании метода независимых фрагментов [69, 91], позволяющего соединять вложенные конструкции последовательно. Основное достоинство метода состоит в том, что он позволяет строить СПГСА за счет замены фрагментов специально проектируемого аналитического выражения (объединенной формулы), описывающего ее структуру, библиотечными блоками (рис. 4.50).

Такой подход близок к построению параллельно-последовательных контактных схем по булевым формулам в базисе И, ИЛИ, НЕ, но в отличие от последних указанные аналитические выражения не описывают функционирование СПГСА.

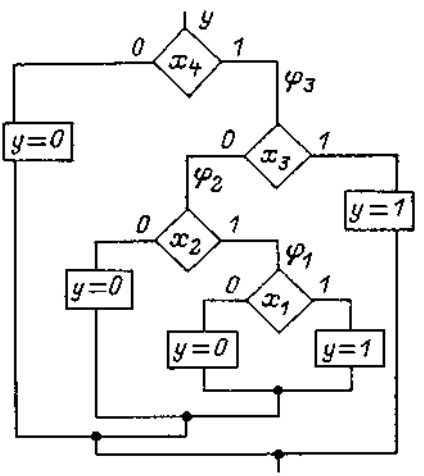

Рис. 4.46

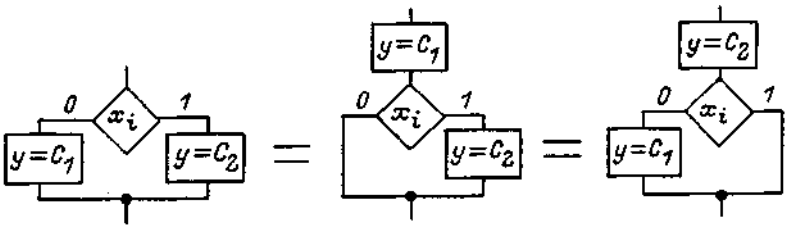

Рис. 4.47

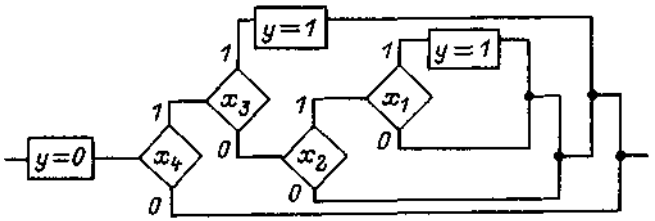

Рис. 4.48

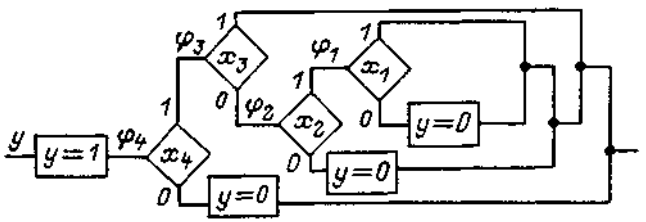

Рис. 4.49

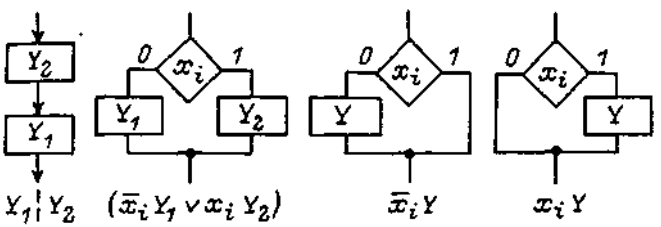

Рис. 4.50

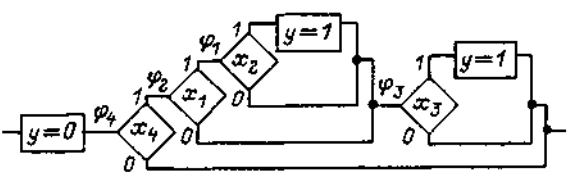

Рис. 4.51

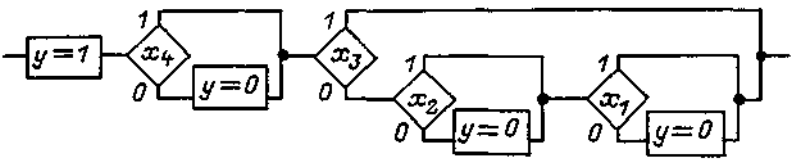

Рис. 4.52

Неструктурированные ПГСА аналогичны мостиковым контактным схемам, так как формулы, по которым они строятся, описывают функционирование, но не описывают структуру.

Для формулы $y = (x_1x_2 \vee x_3)x_4$ указанное аналитическое выражение строится следующим образом. Заданная формула записывается в бесскобочной форме: $y = x_1x_2x_4 \vee x_3x_4$. Строится объединенная формула, в которой каждая конъюнкция помечается символом y: $Y = x_1x_2x_4y \mid x_3x_4y$ Осуществляется вынос за скобки влево повторяющихся переменных x_i: $Y = x_4(x_1x_2y \mid x_3y)$. Фрагменты этого выражения заменяются библиотечными элементами (рис. 4.50), а в начало СПГСА вводится операторная вершина $y = 0$ (рис. 4.51). При этом $Y = \bar{y} \mid x_4(x_1x_2y \mid x_3y)$. Символу «|» в построенном выражении соответствует последовательное соединение блоков в СПГСА. Для этой цели может быть использован также символ «II» (Приложение 2, примеры *10* и *11).*

Модифицируем изложенный метод для построения формулы y с помощью дополнительной структуры. При этом синтез ведется по инверсии формулы. Конъюнкции в бесскобочном выражении помечаются символом $\bar{y}$, а в начале СПГСА устанавливается вершина y=1. Для рассматриваемого примера $Y = y \mid \bar{x}_4\bar{y} \mid \bar{x}_3(\bar{x}_2\bar{y} \mid \bar{x}_1y)$. Реализация заданной формулы в этом случае приведена на рис. 4.52.

Отметим, что при использовании метода независимых фрагментов последовательно соединенные блоки (кроме блока начальной установки) допускается менять местами. Этот метод может быть использован также для построения СПГСА по системе булевых формул. Пусть $y_1 = x_1x_2x_4$; $y_2 = x_3x_4$. Тогда $Y = \bar{y}_1 \mid \bar{y}_2 \mid x_4(x_1x_2y_1 \mid x_3y_2)$

Рассмотренный метод для одной формулы обеспечивает минимальную нижнюю оценку числа вершин $B_н = h + 2$ и нелинейную верхнюю оценку.

Еще один метод структурирования носит название «метод булевых признаков» [103]. При его использовании для каждой точки объединения дуг остова линейного ПГСА (за исключением соединенной с операторной вершиной, если такая точка имеется) вводится булев признак (промежуточная переменная). После этого ПГСА разбивается на фрагменты, которые структурируются методом дублирования с введением дополнительных операторных вершин, в которых осуществляется присвоение булевым признакам значения, равного единице. Фрагменты объединяются в СПГСА с помощью условных вершин, в которых проверяются значения булевых признаков. На рис. 4.53 приведена СПГСА, построенная этим методом по ПГСА (рис. 4.41).

Исключить проверку промежуточных переменных, соответствующих точкам остова, заменив ее проверкой значений выходной переменной,

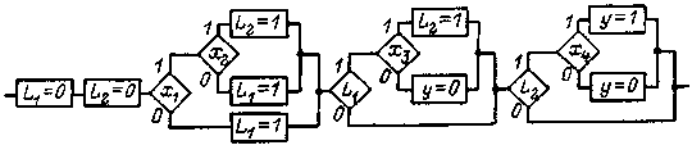

Рис. 4.53

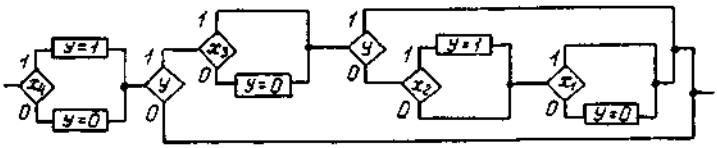

Рис. 4.54

удается в тех случаях, когда при каждом разложении Шеннона по крайней мере одна из остаточных формул равна константе.

Метод построения СПГСА для каждой формулы этого класса состоит в построении линейной ПГСА и в преобразовании ее в структурированную. При этом ПГСА с максимальным числом путей читается в обратном порядке и по ней формируются h фрагментов, первый из которых является блоком «полный выбор», а остальные — блоками «неполный выбор». В каждом блоке используется одна из условных вершин линейной ПГСА. Первый блок с помощью правила «выноса вверх» (рис. 4.47) всегда может быть преобразован в последовательное соединение двух блоков, один из которых является операторной вершиной, а второй — блоком «неполный выбор».

Блоки формируются следующим образом. Если в линейной ПГСА дуга, исходящая из условной вершины и не входящая в остов, соединена с «нулевой» («единичной») операторной вершиной или точкой остова, то соответствующая дуга условной вершины, расположенной в блоке, соединяется с операторной вершиной $y = 0$ $(y = 1)$.

Блоки в СПГСА соединяются между собой последовательно до тех пор, пока в остове линейной ПГСА не встретится точка, которая заменяется условной вершиной, проверяющей значение выходной переменной.

В СПГСА рассматриваемой структуры предполагается, что операция «Вывод» выполняется в конце вычислений.

На рис. 4.54 представлена СПГСА, построенная предлагаемым методом по формуле $y = (x_1x_2 \vee x_3)x_4$. При его использовании по заданной формуле строится линейная ПГСА (рис. 4.41), которая затем структурируется. Эта формула принадлежит рассматриваемому классу, так как при разложении Шеннона соответствующие остаточные формулы равны константам:

$$y = \bar{x}_4 0 \vee x_4(x_1x_2 \vee x_3) = \bar{x}_4 0 \vee x_4\varphi; \qquad \varphi = \bar{x}_3(x_1x_2) \vee x_3 1$$

и т. д.

Оценим число вершин в ПГСА в этом случае. Число условных вершин с независимыми переменными равно Л, число операторных вершин (ОВ) — $(h + 1)$, число условных вершин с выходной переменной — T, где T— число точек остова, не связанных с ОВ. Таким образом,

$$B = 2h + T + 1.$$

Так как $0 \leqslant T \leqslant h-2$, то $2h+1 \leqslant B \leqslant 3h-1$

Число вершин в СПГСА (рис. 4.54) соответствует верхней оценке и равно 11.

Если по реализуемой формуле строить линейную ПГСА с минимальным числом путей, то $0 \leqslant T \leqslant \left[\frac{h-2}{2}\right]$. При этом $2h+1 \leqslant B \leqslant [2.5h]$

Из изложенного следует, что если формула может быть реализована линейной ПГСА с $T=0$, то СПГСА образуется только последовательным соединением одного блока «полный выбор» и (Л - 1)-го блока «неполный выбор». Этим свойством обладают бесповторные пороговые формулы (БПФ) [39]. Простейшим признаком принадлежности формулы этому классу является возможность реализации однотипной с ней положительной монотонной формулы линейной схемой из двухвходовых элементов И и ИЛИ [274].

БПФ реализуется последовательной СПГСА, если она записана в порядке невозрастания весов ее переменных.

Для формулы $y = (x_1x_2 \vee x_3)x_4$ последовательная СПГСА (рис. 4.55) строится по линейной ПГСА (рис. 4.42). При этом число вершин в СПГСА $B = 2h + 1 = 9$

Отметим, что при использовании предлагаемого метода в отличие от метода независимых фрагментов изменение порядка расположения последовательно соединенных блоков недопустимо и поэтому он может быть назван методом зависимых фрагментов.

СПГСА для БПФ, записанных в порядке неубывания весов ее переменных, может быть построена (без предварительного построения ПГСА) по таблице истинности с помощью предлагаемой автором модификации канонического метода [19]. Смысл последней состоит в том, что из столбца значений на остов канонической таблицы из пар, образующих ее кусты, выносятся отдельные значения (как единичные, так и нулевые), которые заменяются прочерками, а в дальнейшем эти значения переносятся по ребрам канонической таблицы к ее корню с целью получения максимального числа одинаковых кустов, симметрично расположенных относительно корня.

Предлагаемые подходы позволяют, в частности, реализовать положительно монотонные формулы И и ИЛИ только из однотипных блоков (не считая блока начальной установки), соединенных последовательно (рис. 4.56, 4.57).

Это обеспечивает возможность циклической реализации этих формул при простейшем теле цикла. Следует различать многократное выполнение ГСА от циклической реализации формулы, которая в свою очередь может выполняться многократно.

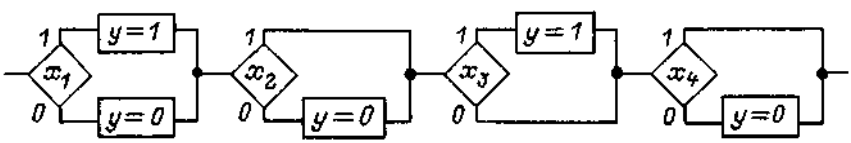

Рис. 4.55

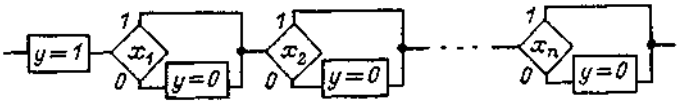

Рис. 4.56

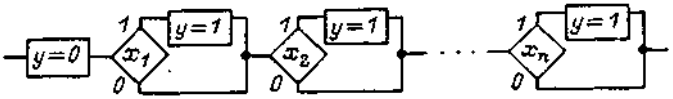

Рис. 4.57

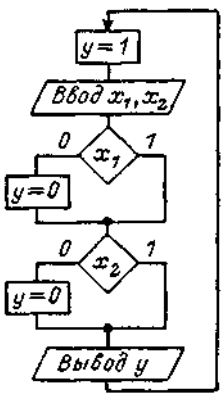

Рис. 4.58

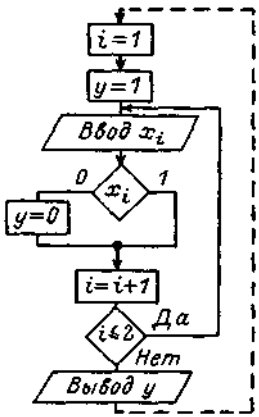

Рис. 4.59

Многократная реализация ГСА достигается простым замыканием ее выхода на вход (возвратом назад), а циклическая реализация требует использования специальной конструкции.

На рис. 4.58 приведена схема многократной реализации формулы $y = x_1x_2$, а на рис. 4.59 — ее однократная (без учета пунктира) и многократная (с учетом пунктира) циклические реализации, соответствующие оператору `do–while` языка СИ.

Возвращаясь к методу зависимых фрагментов, отметим, что если структурированная ГСА (СГСА) не обязательно должна принадлежать классу простейших, то булева формула с пороговым образом может быть реализована только последовательным соединением блоков. Для этого заданная формула переобозначением подформул сводится к БПФ меньшего числа переменных. Бесповторная пороговая формула записывается в порядке невозрастания весов переменных и реализуется линейной ГСА, которая в свою очередь преобразуется в СГСА.

Пусть требуется реализовать формулу

$$y = (x_1x_2 \vee x_3x_4)(x_5 \vee x_6)(x_7 \vee x_8).$$

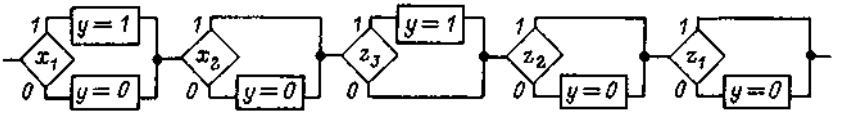

Рис. 4.60

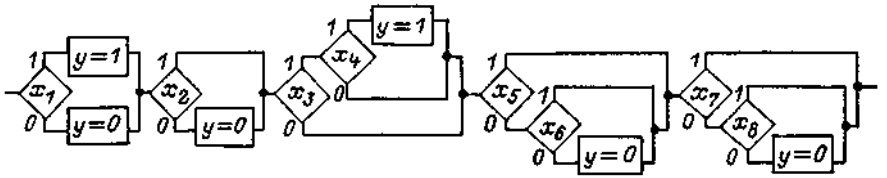

Рис. 4.61

Преобразуем ее к виду:

$$y = z_1 z_2 (z_3 \vee x_2 x_1).$$

где

$$z_1 = x_7 \vee x_8; \qquad z_2 = x_5 \vee x_6; \qquad z_3 = x_3 x_4.$$

Реализуем эту формулу методом зависимых фрагментов (рис. 4.60). После обратной подстановки формул вместо переменных z_1, z_2 и z_3 получим искомую СГСА.

Перейдем к изложению смешанного метода, суть которого состоит в том, что методом зависимых фрагментов строится последовательная СГСА с подформулами И и ИЛИ в некоторых условных вершинах, реализуемых методом независимых фрагментов без использования вершин начальной установки. При этом если в блоке СГСА используется оператор $y = 1$, то методом независимых фрагментов реализуется подформула, а в случае оператора $y = 0$ с помощью этого метода строится инверсная формула для получения дополнительной реализации.

При применении предложенного метода для последнего примера третий блок в структуре, построенной методом зависимых фрагментов, должен реализовываться методом независимых фрагментов по формуле $z_3 = x_3 x_4$, а четвертый и пятый блоки — по инверсным формулам: $\bar{z}_2 = \bar{x}_5 \bar{x}_6$ и $\bar{z}_1 = \bar{x}_7 \bar{x}_8$ (рис. 4.61). Число вершин в построенной СПГСА — 14. При использовании метода зависимых фрагментов в этом случае число вершин равно 20, а для метода независимых фрагментов — 25 (для прямой реализации) и 17 (для дополнительной). За счет более эффективной реализации первых двух блоков в СПГСА (рис. 4.61) число вершин может быть сокращено до 13. При этом указанные блоки заменяются структурой, описываемой выражением $\bar{y}_1 x_1 x_2 y$.

Рассмотрим еще два метода построения СГСА для системы булевых функций. При этом предполагается, что значения этой системы заданы их десятичными эквивалентами, т. е. система заменена одной функцией, в которой аргументы двоичны, а значения — многозначны.

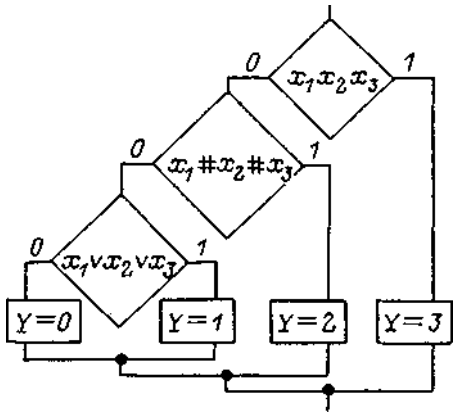

Рис. 4.62

Первый метод строит древовидную ГСА простейшей структуры, а второй — ГСА, в которой все блоки «неполный выбор» соединены последовательно. В этих ГСА в условных вершинах могут проверяться не только отдельные переменные, но и булевы формулы.

Первый метод состоит в том, что в столбце значений таблицы истинности, заменяющей функции, выбирается одно из значений и определяется функция его выбора, принимающая значение, равное единице на входных наборах, на которых расположено выбираемое значение, и ноль — на всех остальных наборах. Выбранное значение исключается из таблицы истинности, а заменяющая функция становится не полностью определенной, что позволяет при выборе следующего ее значения доопределять функцию выбора до простейшей. Эта процедура повторяется (R - 1) раз и строит ГСА, состоящую из (2R - 1) вершин, где R — число различных значений в СБФУ.

Реализуем в качестве примера СБФУ с транспонированным столбцом значений $| Y_1 |^{\mathrm{T}} = | 0, 1, 1, 2, 1, 2, 2, 3 |$ при порядке переменных x_1, x_2, x_3. Значение $Y_1 = 3$ выбирает функцию $y_1 = x_1x_2x_3$. При этом $| Y_{11} |^{\mathrm{T}} = | 0, 1, 1, 2, 1, 2, 2, — |$. Выбор значения $Y_1 = 2$ осуществим функцией голосования $y_2 = x_1 \# x_2 \# x_3$. При этом $| Y_{12} |^{\mathrm{T}} = | 0, 1, 1, —, 1, —, —, — |$. Выбор значений $Y_1 = 0$ и $Y_1 = 1$ осуществляет функция $y_3 = x_1 \vee x_2 \vee x_3$. Построенная ГСА приведена на рис. 4.62. Эта схема интересна тем, что реализация двух булевых формул («голосование» и «сумма по модулю два») осуществлена с использованием трех формул, одна из которых является функцией голосования.

Второй метод состоит в том, что сначала для заменяющей функции строится остов канонической таблицы. На остов, во второй ярус, выносятся из столбца значений таблицы истинности целые числа, большие нуля, максимально и одинаково уменьшая значения в каждой паре столбца, образующей куст канонической таблицы. Следующие этапы выноса осуществляются аналогично и проводятся для каждого куста. После завершения выноса всех чисел осуществляется многозначное кодирование вершин канонической таблицы, состоящее в том, что одина-

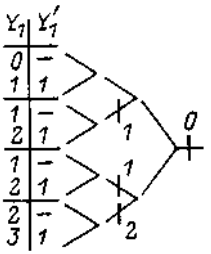

Рис. 4.63

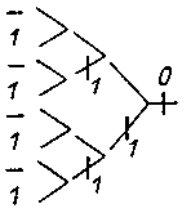

Рис. 4.64

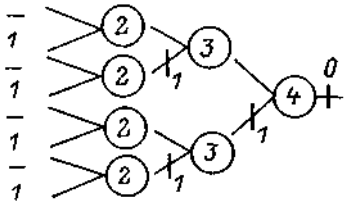

Рис. 4.65

ковые кусты обозначаются одинаковыми цифрами, а их нумерация начинается от столбца значений [84]. Осуществляется объединение одноименных кустов и исключаются нулевые пометки в преобразованном столбце значений таблицы истинности. По полученной канонической таблице строится ГСА, заменяя при этом цифру «C_i» на ребре канонической таблицы операторной вершиной с пометкой «$Y=Y+C_i$». Начальная установка осуществляется с помощью операторной вершины «$Y=C_j$», где C_j — максимальное значение, на которое могут быть уменьшены все значения столбца значений таблицы истинности.

Пример 4.9. Пусть требуется реализовать СБФУ, заданную столбцом значений Y_j, (рис. 4.63). На этом рисунке приведен также упрощенный столбец значений ТИ, получающийся из исходного после выноса целых чисел в канонической таблице. На рис. 4.64 выполнен вынос единицы в третий ярус. Многозначное кодирование вершин канонической таблицы проведено на рис. 4.65. Объединение одинаковых кустов в упрощенной канонической таблице показано на рис. 4.66. Структурированная ГСА, построенная по этой таблице, приведена на рис. 4.67.

Сформулируем утверждение. Если ГСА состоит только из последовательного соединения операторной вершины «$Y=C_0$» и блоков «неполный выбор» с операторными вершинами: «$Y=Y+C_1$», «$Y=Y+C_2$»,..., «$Y=Y+C_n$», то СБФУ, соответствующая этой граф-схеме, может быть реализована линейным арифметическим полиномом $Y = C_0 + C_1x_1 + C_2x_2 + \ldots + C_nx_n$.

Из рассмотрения ГСА (рис. 4.67) следует, что $Y_1 = 1x_1 + 1x_2 + 1x_3$, в то время как СБФУ $|\ Y_2\ |^T = |\ 0, 2, 4, 6, 3, 5, 7, 9\ |$ реализуется полиномом $Y_2 = 3x_1 + 4x_2 + 2x_3$ (рис. 4.68).

Таким образом, можно утверждать, что предложены новая трактовка условия линейности арифметических полиномов [82] и графический метод их построения в случае, если это условие выполняется.

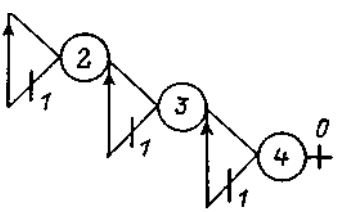

Рис. 4.66

Если построенная ГСА имеет структуру, отличную от описанной, то СБФУ реализуется нелинейным арифметическим полиномом [82]. На рис. 4.69, *б* приведена ГСА, реализующая СБФУ $|\ Y_3\ |^T = |\ 2, 2, 2, 2, 1, 2, 2, 3\ |$

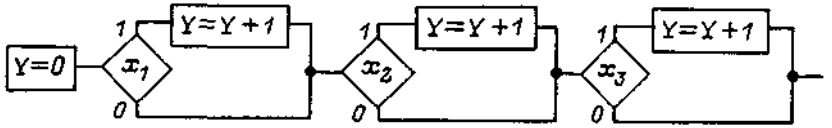

Рис. 4.67

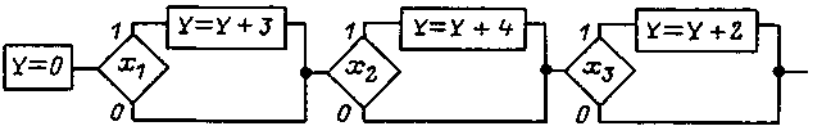

Рис. 4.68

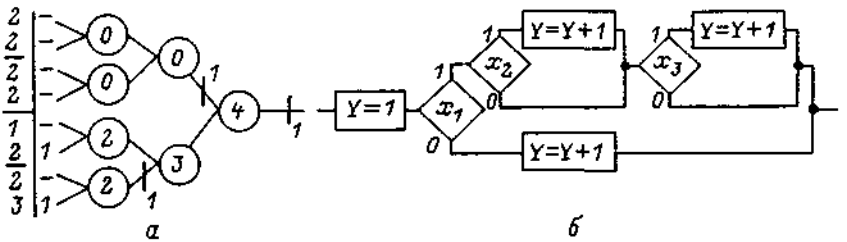

Рис. 4.69

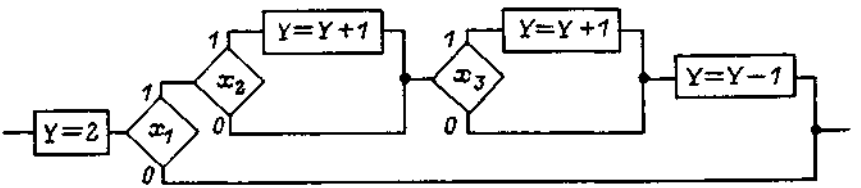

Рис. 4.70

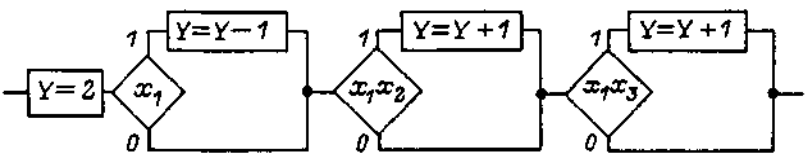

Рис. 4.71

предложенным методом (рис. 4.69, *а*). Запишем минимальное по числу символов арифметическое выражение по этой ГСА: $Y_3 = 1 + 1(1 - x_1) + x_1(1x_2 + 1x_3) = 2 + x_1(1x_2 + 1x_3 - 1)$. Построим по этому выражению ГСА (рис. 4.70). Преобразуем полученное выражение в полиномиальную форму $Y_3 = 2 - x_1 + x_1x_2 + x_1x_3$ и построим по этому выражению ГСА (рис. 4.71).

Из рассмотренных примеров следует, что произвольная СБФУ всегда может быть реализована структурированной линейной ГСА со сложными условиями и операторными вершинами.

Изложенный метод является новым методом независимых фрагментов, так как последовательно соединенные фрагменты в построенной на его основе ГСА могут переставляться.

В заключение отметим, что изложенный подход строит арифметические полиномы без использования аналитических [80] и матричных преобразований [82].

4.3.3. Реализация автоматов с памятью ГСА

Рассмотрение этого вопроса проведем на примере *R*-триггера.

Если автомат задан в виде ГП (рис. 4.15), то он может быть изоморфно изображен в виде ГСА (рис. 4.72). Однако ввиду возвратов назад эта ГСА весьма неудобна для программирования, так как требует многократного использования операторов ввода и вывода.

Рассмотрим вопрос о построении ГСА с одним возвратом назад, которая содержит лишь по одному указанному оператору.

В силу того что *R*-триггер описывается формулой $y = \bar{R}(S \vee y)$, он может быть реализован операторной ГСА, представленной на рис. 4.73.

Для построения СПГСА разложим формулу триггера и ее остаточные по Шеннону (по переменным *R, S, y* соответственно) или реализуем трехстолбцовую таблицу переходов или четырехстолбцовую кодированную таблицу переходов триггера при указанном порядке переменных каноническим методом (рис. 4.74). Укрупним полученную ГСА, заменив последний куст операторной вершиной «$y = y$» и введя начальную вершину «$y = C$» (рис. 4.75). Ввиду наличия возврата назад вершину «$y = y$» в ГСА можно исключить (рис. 4.76).

Тело построенной СПГСА состоит из вложенных блоков. Используем метод зависимых фрагментов для построения СПГСА, тело которой состоит из последовательно соединенных блоков. Построим линейную ПГСА по формуле *R*-триггера (рис. 4.77). Реализуем с помощью метода, предложенного в предыдущем разделе, по этой граф-схеме СПГСА, состоящую из последовательности блоков (рис. 4.78). В этой ГСА первый блок может быть исключен. Вводя блок начальной установки, получим искомую схему (рис. 4.79).

Поступая аналогично, построим простейшие схемы для *S*-триггера (рис. 4.80, 4.81).

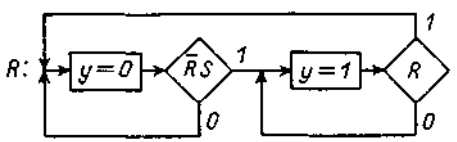

Рис. 4.72

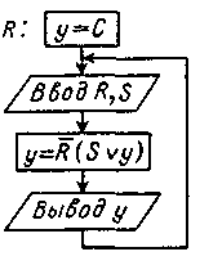

Рис. 4.73

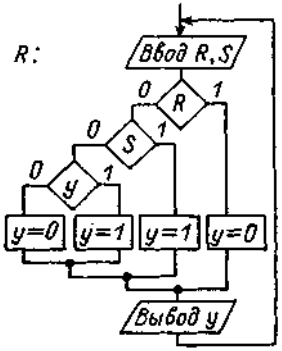

Рис. 4.74

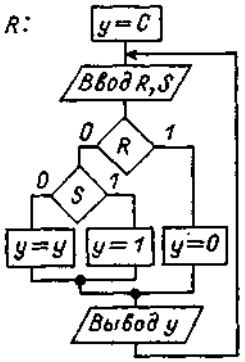

Рис. 4.75

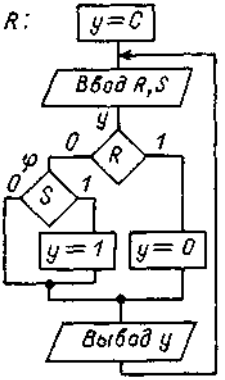

Рис. 4.76

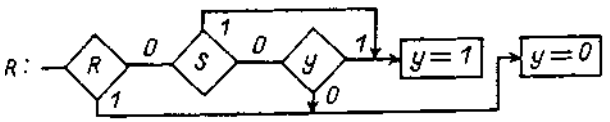

Рис. 4.77

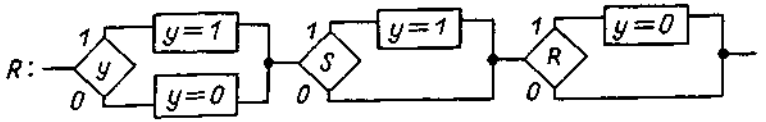

Рис. 4.78

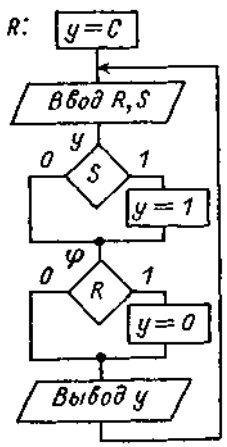

Рис. 4.79

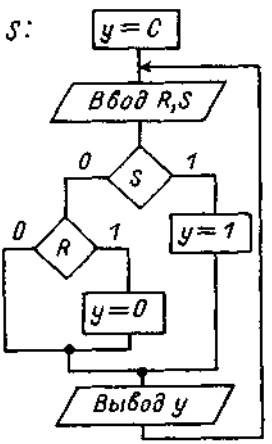

Рис. 4.80

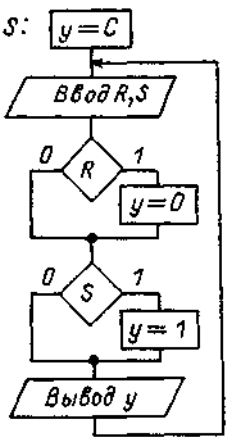

Рис. 4.81

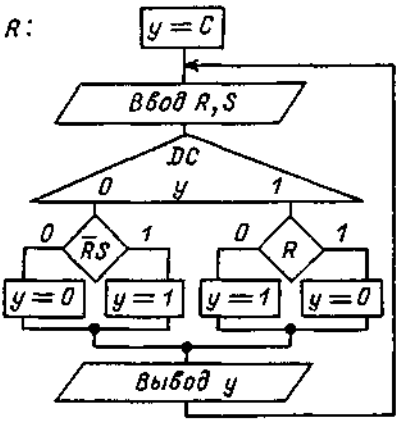

Рис. 4.82

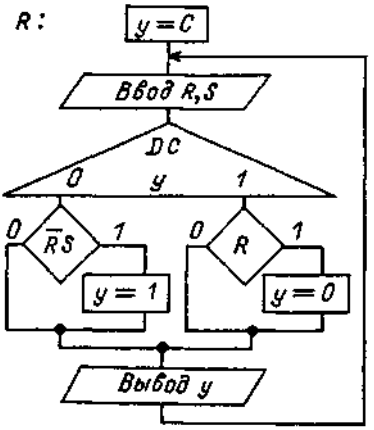

Рис. 4.83

Рассмотрим еще один подход к построению СПГСА. Суть его состоит в том, что предлагается структура с одним возвратом назад, позволяющая изоморфно отразить исходный ГП. Тело СГСА состоит из блока начальной установки, цифрового дешифратора состояний с одним входом и s выходами (где s — число состояний), i-й выход которого подключен к условной вершине (возможно, многовыходной), выходы которой связаны с операторными вершинами, соответствующими состояниям, в которые переходит автомат из состояния i. На рис. 4.82 представлена схема, построенная по ГП (рис. 4.15). В этой структуре ввиду наличия возврата назад операторные вершины, обеспечивающие сохранение предыдущего состояния, могут быть исключены (рис. 4.83).

Рассмотренный подход изложен применительно к автомату без выходного преобразователя, но легко модифицируется на другие классы автоматов (гл. 13).

Предлагаемая структура обеспечивает новую (по сравнению с рассмотренными выше) разновидность композиции блоков — параллельное соединение блоков, число которых равно s. При этом объединение блоков осуществляется дешифратором состояний, а не входных переменных [79], что резко упрощает «чтение» ГСА.

Структура, близкая к рассмотренной, использована в методе Ашкрофта и Манны [102] для структурирования неструктурированных ГСА, что принципиально отличает такой подход от предлагаемого, при использовании которого «промежуточная» ГСА не строится.

Сравнение полученной схемы, например с ГСА (рис. 4.76), показывает, что изложенный подход приводит к большим затратам памяти, однако он существенно проще, а получающиеся схемы более наглядны. Предложенная структура изоморфна ГП и языковой конструкции `switch`, что обосновывает применение последней при программной реализации ГСА в случаях, когда жесткие ограничения на объем памяти и быстродействие отсутствуют.

4.3.4. Верификация ГСА

Верификация в зависимости от структуры ПГСА осуществляется от выхода к входу, от входа к выходу и совместно, т. е. часть фрагментов верифицируется от входа к выходу, а остальные — от выхода к входу.

Если в ПГСА отсутствуют последовательно соединенные блоки выбора, то она может быть верифицирована от выхода к входу, используя разложение Шеннона для каждой условной вершины.

Определим булеву формулу для линейной ПГСА (рис. 4.41), читая ее справа налево:

$$\varphi_1 = \bar{x}_4 0 \vee x_4 1 = x_4; \qquad \varphi_2 = \bar{x}_3 0 \vee x_3 \varphi_1 = x_3 x_4;$$

$$\varphi_3 = \bar{x}_2 \varphi_2 \vee x_2 \varphi_1 = (x_2 \vee x_3) x_4; \quad y = \bar{x}_1 \varphi_2 \vee x_1 \varphi_3 = (x_1 x_2 \vee x_3) x_4.$$

Линейные ПГСА обладают замечательным свойством [87]: на выходе каждой условной вершины реализуется фрагмент заданной формулы, а не подформула, что не обеспечивается другими методами.

Определим булеву формулу для плоскостной ПГСА (рис. 4.46), читая ее снизу вверх:

$$\varphi_1 = \bar{x}_1 0 \vee x_1 1 = x_1; \quad \varphi_2 = \bar{x}_2 0 \vee x_2 \varphi_1 = x_1 x_2;$$

$$\varphi_3 = \bar{x}_3 \varphi_2 \vee x_3 1 = x_1 x_2 \vee x_3; \quad y = \bar{x}_4 0 \vee x_4 \varphi_3 = (x_1 x_2 \vee x_3) x_4.$$

Пусть имеется ГСА для автомата без памяти, реализующая СБФ и содержащая последовательно соединенные блоки с одним входом и одним выходом, каждый из которых реализует одну формулу. Можно утверждать, что эти блоки не влияют друг на друга и поэтому их можно анализировать независимо, что резко уменьшает размерность решаемой задачи.

Если каждый из этих блоков не содержит в свою очередь последовательно соединенных блоков, то их анализ может проводиться не только от выхода к входу, как было рассмотрено выше, но и от входа к выходу.

При этом первоначально с помощью метода Акерса [88] подсчитывается число единичных и нулевых путей в блоке.

Если число единичных путей в блоке меньше, чем число нулевых путей, то для каждого единичного пути строится конъюнкция входных переменных, ему соответствующая. Эти конъюнкции объединяются в дизъюнктивную нормальную форму искомой формулы.

Если число нулевых путей меньше, чем число единичных путей, то действия, указанные выше, выполняются для инверсии формулы, по которой в дальнейшем строится сама формула.

Полученное выражение может быть преобразовано с целью построения по нему нового блока ГСА с лучшими по сравнению с исходным блоком характеристиками (число вершин, число путей, линейность, планарность и т. д.).

Для ГСА на рис. 4.43 с помощью метода Акерса подсчитано число единичных (три) и нулевых (пять) путей. Перечислим единичные пути и объединим их в дизъюнктивную нормальную форму:

$$y = x_1 x_2 x_4 \vee x_1 \bar{x}_2 x_3 x_4 \vee \bar{x}_1 x_3 x_4.$$

Проминимизируем эту формулу:

$$y = (x_1x_2 \vee (x_1\bar{x}_2 \vee \bar{x}_1)x_3)x_4 = (x_1x_2 \vee (\bar{x}_1 \vee \bar{x}_2)x_3)x_4 =$$
$$= (x_1x_2 \vee \overline{x_1x_2}x_3)x_4 = (x_1x_2 \vee x_3)x_4.$$

После построения формулы, эквивалентной ГСА (рис. 4.43), имеется возможность провести этап программирования по этой формуле, а не по граф-схеме, либо построить линейную ГСА (рис. 4.41), которая читается лучше, чем исходная.

Если СБФ реализована в виде единого блока, то анализ с помощью изложенных подходов должен выполняться для каждой формулы в отдельности.

Построим булеву формулу для плоскостной ПГСА (рис. 4.49) с «пустыми» дугами:

$$\varphi_1 = \bar{x}_1 0 \vee x_1 y = x_1 y; \qquad \varphi_2 = \bar{x}_2 0 \vee x_2\varphi_1 = x_1x_2y;$$
$$\varphi_3 = \bar{x}_3\varphi_2 \vee x_3 y = (x_1x_2 \vee x_3)y; \qquad \varphi_4 = \bar{x}_4 0 \vee x_4\varphi_3 = (x_1x_2 \vee x_3)x_4y;$$
$$y = (x_1x_2 \vee x_3)x_4.$$

Определим булеву формулу для автомата с памятью, реализуемого ПГСА (рис. 4.76):

$$\varphi = \bar{S}y \vee S1 = S \vee y; \qquad y = \bar{R}\varphi \vee R0 = \bar{R}(S \vee y).$$

Если ПГСА состоит только из последовательно соединенных блоков «выбор» (рис. 4.79), то она верифицируется от входа к выходу:

$$\varphi = \bar{S}y \vee S1 = S \vee y; \qquad y = \bar{R}\varphi \vee R0 = \bar{R}(S \vee y).$$

Если ПГСА образована как вложенными конструкциями, так и последовательным соединением блоков, то первые верифицируются от выхода к входу, а вторые — от входа к выходу.

Определим булеву формулу для ПГСА (рис. 4.51):

$$\varphi_1 = \bar{x}_2 y \vee x_2 1 = x_2 \vee y; \qquad \varphi_2 = \bar{x}_1 y \vee x_1\varphi_1 = x_1x_2 \vee y;$$
$$\varphi_3 = \bar{x}_3\varphi_2 \vee x_3 1 = x_1x_2 \vee x_3 \vee y; \qquad \varphi_4 = \bar{x}_4 y \vee x_4\varphi_3 = (x_1x_2 \vee x_3)x_4 \vee y;$$
$$y = (x_1x_2 \vee x_3)x_4.$$

Определим СБФ для автомата с памятью, реализуемого СПГСА (рис. 4.84):

$$\begin{cases} y_{11} = \bar{x}_1 y_1 \vee x_1 y_1 = y_1; \\ y_{21} = \bar{x}_1 y_2 \vee x_1 1 = x_1 \vee y_2; \end{cases} \qquad \begin{cases} y_{12} = \bar{x}_2 y_{11} \vee x_2 1 = x_2 \vee y_1; \\ y_{22} = \bar{x}_1 y_{21} \vee x_2 y_{21} = x_1 \vee y_2; \end{cases}$$

$$\begin{cases} y_1' = \bar{x}_3 y_{12} \vee x_3 0 = \bar{x}_3(x_2 \vee y_1); \\ y_2' = \bar{x}_3 y_{22} \vee x_3 0 = \bar{x}_3(x_1 \vee y_2). \end{cases}$$

Таким образом, рассмотренная СПГСА реализует два R-триггера — с раздельной установкой и общим сбросом (рис. 4.25).

Построим с помощью метода независимых фрагментов по условиям переходов R-триггеров (рис. 4.15) еще одну ГСА для этого примера:

$$Y = x_2\bar{x}_3y_1 \mid x_1\bar{x}_3y_2 \mid x_3\bar{y}_1 \mid x_3\bar{y}_2 = (\bar{x}_3(x_1y_2 \mid x_2y_1) \vee x_3(\bar{y}_1 \mid \bar{y}_2)).$$

Эта ГСА (рис. 4.85) верифицируется аналогично предыдущей. В последнем выражении символ «|» обозначает последовательное соединение блоков в граф-схеме.

Рассмотрим еще несколько примеров, в которых ряд шагов осуществляется от выхода к входу (на основе разложения Шеннона), а остальные шаги — от входа к выходу (на основе подстановки).

На рис. 4.86 приведена ГСА, в которой каждое число, расположенное в квадрате, соответствует порядковому номеру разложения Шеннона.

Эта ГСА, реализующая автомат, описывается следующим образом:

$$\begin{cases} y_{11} = \bar{x}_3\, y_1 \vee x_3\, y_1 = y_1, \\ y_{12} = \bar{x}_3\, y_2 \vee x_3\, 1 = x_3 \vee y_2; \end{cases}$$

$$\begin{cases} y_{21} = \bar{x}_3\, y_1 \vee x_3\, 0 = \bar{x}_3\, y_1, \\ y_{22} = \bar{x}_3\, 1 \vee x_3\, y_2 = \bar{x}_3 \vee y_2; \end{cases}$$

$$\begin{cases} y_{31} = \bar{x}_2\, y_{11} \vee x_2\, y_{21} = (\bar{x}_2 \vee \bar{x}_3)\, y_1, \\ y_{32} = \bar{x}_2\, y_{12} \vee x_2\, y_{22} = \bar{x}_2\, x_3 \vee x_2\bar{x}_3 \vee y_2; \end{cases}$$

$$\begin{cases} y_{41} = 1, \\ y_{42} = y_{12} = x_3 \vee y_2; \end{cases}$$

$$\begin{cases} y_{51} = \bar{x}_1\, y_{41} \vee x_1\, y_{31} = \bar{x}_1 \vee (\bar{x}_2 \vee \bar{x}_3)\, y_1, \\ y_{52} = \bar{x}_1\, y_{42} \vee x_1\, y_{32} = x_1\, x_2 \oplus x_3 \vee y_2; \end{cases}$$

$$\begin{cases} y_{61} = \bar{x}_4\, 1 \vee x_4\, y_1 = \bar{x}_4 \vee y_1, \\ y_{62} = y_2; \end{cases}$$

$$\begin{cases} y_{71} = y_{51} \leftarrow y_{61} = \bar{x}_1 \vee (\bar{x}_2 \vee \bar{x}_3)\,(\bar{x}_4 \vee y_1), \\ y_{72} = y_{52} \leftarrow y_{62} = x_1\, x_2 \oplus x_3 \vee y_2; \end{cases}$$

$$\begin{cases} y_{81} = y_{71} = \bar{x}_1 \vee (\bar{x}_2 \vee \bar{x}_3)\,(\bar{x}_4 \vee y_1) = y_1', \\ y_{82} = y_{72} \leftarrow (y_2 = 0) = x_1\, x_2 \oplus x_3 = y_2'. \end{cases}$$

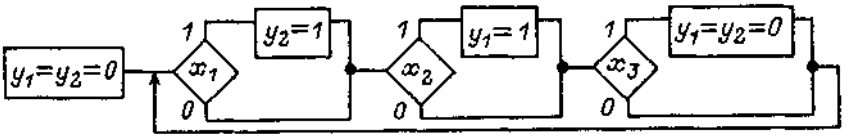

Рис. 4.84

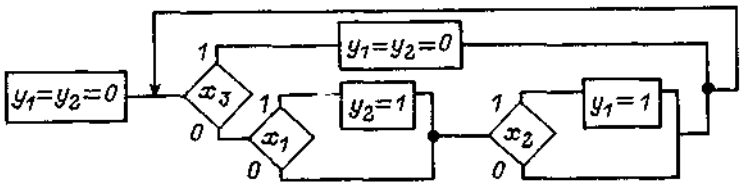

Рис. 4.85

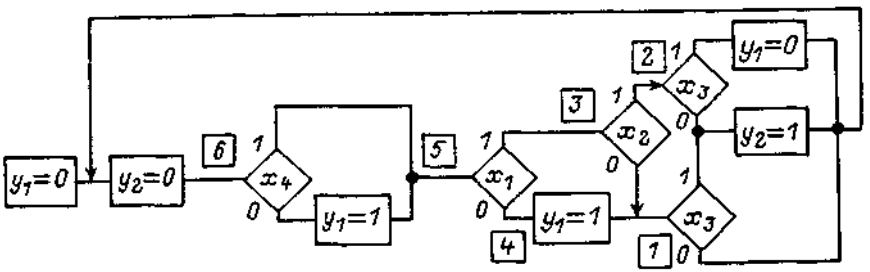

Рис. 4.86

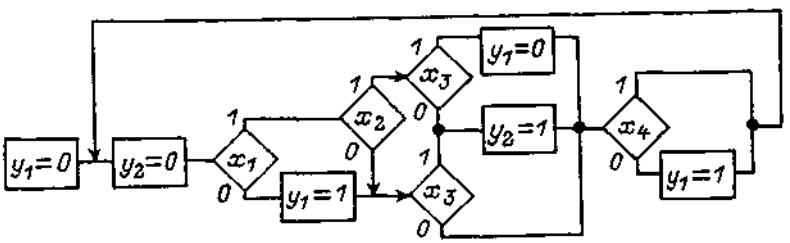

Рис. 4.87

Изменение порядка расположения последовательных фрагментов в граф-схеме (рис. 4.87) приводит к СБФ вида

$$y_1' = \bar{x}_1 \vee (\bar{x}_2 \vee \bar{x}_3)\, y_1 \vee x_4; \qquad y_2' = x_1 x_2 \oplus x_3.$$

Дальнейшее изменение в граф-схеме, связанное с перестановкой операторных вершин условной вершины, помеченной переменной x_3 (рис. 4.88), существенно изменяет реализуемую СБФ:

$$y_1' = \bar{x}_1 \bar{x}_3 \vee (\bar{x}_2 \bar{x}_3 \vee x_1 x_2 x_3)\, y_1 \vee \bar{x}_4; \qquad y_2' = x_1 x_2 x_3.$$

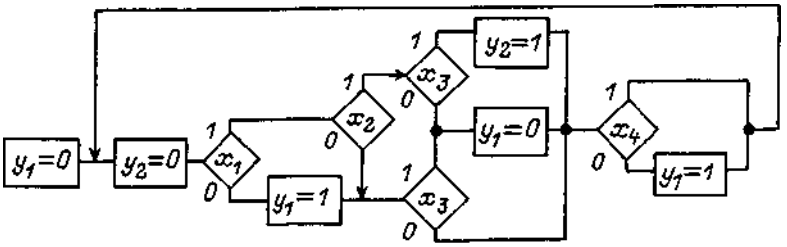

Рис. 4.88

Отметим, что каждая из рассмотренных в последних трех примерах ГСА реализует либо совокупность, состоящую из двух автоматов, связанных только по входным переменным (первый из которых является автоматом без выходного преобразователя с двумя состояниями, а второй — комбинационной схемой), либо один автомат без выходного преобразователя с четырьмя состояниями, закодированными принудительно.

Эти примеры подтверждают тот факт, что неформальные гарантированные изменения в граф-схемы алгоритмов вносить практически невозможно, так как сравнительно небольшие изменения в структуре приводят к принципиальным изменениям в их поведении. Это следует также и из анализа ГП автоматов, эквивалентных граф-схемам. Поэтому после проведения изменений в ГСА для доказательства ее правильности должна выполняться верификация — построение по ГСА системы булевых формул, а по ней в случае необходимости ГП. Отметим, что для ГП автоматов Мура без флагов и без сохранения значений выходных переменных (без использования умолчаний) в отличие от ГСА локальные изменения структуры не могут привести к глобальному изменению их поведения.

Верификация должна производиться для доказательства правильности построения ГСА не только в тех случаях, когда вносятся изменения, но и при «нормальном» их построении. Если программа построена без использования ГСА, то последняя должна быть восстановлена по тексту программы для проведения последующей верификации.

4.3.5. Внесение изменений в ГСА, вычисляющие булевы формулы

Одной из важнейших характеристик качества программ, используемых в системах логического управления, является их модификационная способность [183], так как в сложных системах корректировка алгоритма может потребоваться даже после многих лет успешной эксплуатации.

Трудоемкость корректировки зависит не только от приспособленности программы для внесения изменений, но и вида используемой памяти и наличия программных и технических средств, обеспечивающих внесение изменений, в том числе и в условиях эксплуатации. При этом, однако, в любом случае требуется, чтобы объем корректировки, с одной стороны, был минимален, а с другой — по возможности не затрагивал имеющуюся

программу, так как изменения в одной ее части могут привести к нежелательным последствиям в другой.

В предыдущем разделе отмечалось, что для проверки корректности внесения изменений в ГСА необходимо проводить ее верификацию.

Верификацию можно не проводить, если изменения вносятся стандартным путем, гарантирующим правильность их выполнения.

Известный подход, упрощающий внесение изменений, состоит в построении интерпретирующих программ [41], универсальных в определенном классе функций, при применении которых изменения в случае необходимости вносятся не в качественно проверенную программу, а в массив, настраивающий эту программу на реализацию заданных функций (Приложение 3).

Для компилируемых программ, в основном используемых в настоящей работе, стандартный подход к внесению изменений отсутствует.

Ниже, применительно к программам, вычисляющим СБФ, такой подход предлагается..

При этом ставится следующая задача: организовать процесс вычислений СБФ таким образом, чтобы при любых изменениях реализуемых формул исходная программа корректировалась минимально, а изменения вносились лишь в свободную область памяти, использованный объем которой также должен быть минимальным.

Указанная задача решается за счет введения корректирующих формул *X,* для определения которых используется следующее соотношение алгебры логики: если $f \oplus X = f'$, то $X = f \oplus f'$

Пусть требуется реализовать СБФ вида

$$f_1 = \bar{x}_1\bar{x}_2 \vee x_3; \quad f_2 = x_1x_3 \vee x_2,$$

которая вычисляется ГСА (рис. 4.89). Предположим, что первая формула системы должна быть изменена на $f_1' = \bar{x}_1\bar{x}_2x_3 \vee x_2\bar{x}_3$. Тривиальное решение этой задачи (рис. 4.90) связано со значительным объемом корректировки уже построенной программы.

Объем изменений существенно снижается, если в первую формулу ввести корректирующую формулу, определяемую следующим образом (рис. 4.91):

$$(\bar{x}_1\bar{x}_2 \vee x_3) \oplus X = \bar{x}_1\bar{x}_2x_3 \vee x_2\bar{x}_3;$$

$$X = (\bar{x}_1\bar{x}_2 \vee x_3) \oplus (\bar{x}_1\bar{x}_2x_3 \vee x_2\bar{x}_3) = x_1 \vee x_2.$$

Отметим, что суммирование по модулю два более целесообразно выполнять не в аналитической, а в табличной форме. Предложенная ГСА не решает поставленную задачу, так как корректировка весьма существенно изменяет построенную программу.

Решение задачи, практически исключающее внесение изменений в исходную программу, связано с вычислением новой формулы f_1, после завершения вычислений исходных формул f_1 и f_2. Однако при таком подходе занимаемый объем свободной памяти не минимален. Уменьшение этого показателя достигается при использовании вычисленного значения исходной формулы f_1 и его корректировки (рис. 4.92).

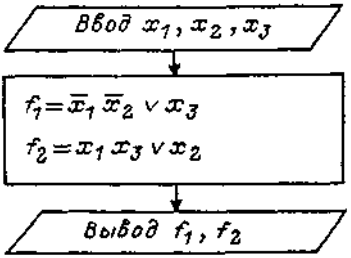

Рис. 4.89

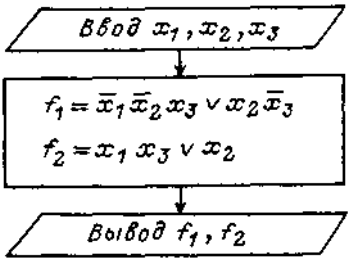

Рис. 4.90

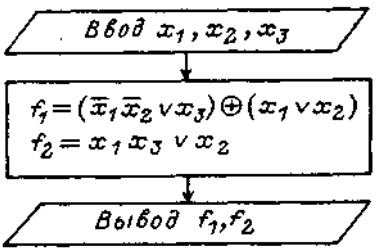

Рис. 4.91

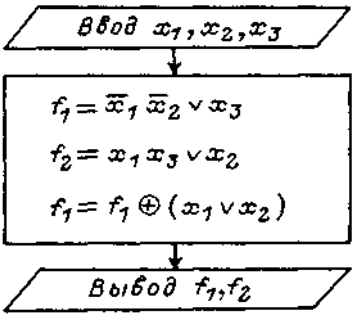

Рис. 4.92

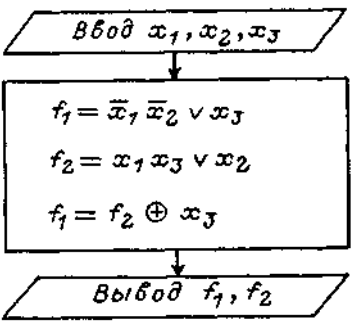

Рис. 4.93

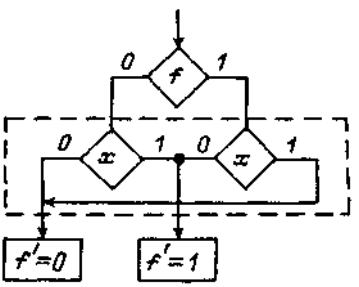

Рис. 4.94

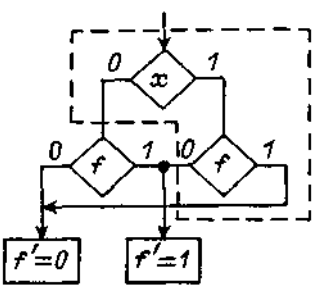

Рис. 4.95

Полученный результат в общем случае может быть улучшен, если для реализации желаемой формулы использовать результат вычислений не только корректируемой формулы, но и любой другой формулы заданной СБФ (рис. 4.93).

При применении схем ветвления используемое соотношение с операцией «сумма по модулю два» реализуется либо ГСА на рис. 4.94, либо ГСА на рис. 4.95. При этом вторая схема требует меньших изменений уже построенной программы.

Рассмотренный подход является стандартным и весьма простым, но обладает двумя недостатками: всегда применяется операция «сумма по модулю два», которая достаточно сложно реализуется; корректирующая формула строится относительно лишь одной формулы исходной системы.

Использование «гарвардского метода» [1] позволяет строить новую функцию, зависящую в общем случае от всех входных и выходных переменных системы. При этом, так как эта функция становится существенно недоопределенной, появляется возможность с помощью карты Карно определить ее простейшую реализацию. Для рассматриваемого примера новая функция определяется данными табл. 4.8, которые заданы лишь на восьми наборах из тридцати двух.

Таблица 4.8

x_1	x_2	x_3	f_1	f_2	f_1
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	0	1	1
0	1	1	1	1	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	1	1
1	1	1	1	1	0

Простейшая формула для этого примера имеет вид:

$$f_1 = f_2 \oplus x_3.$$

Таким образом, в данном случае применение гарвардского метода не улучшает полученного выше результата.

4.4. Графы переходов (ГП)

4.4.1. **Реализация булевых функций автоматами**

Если булева функция задана следующим образом:

$$x_1 = x_2 = 0, \qquad y = 0;$$

$$x_1 = 0, \quad x_2 = 1, \quad y = 0;$$
$$x_1 = 1, \quad x_2 = 0, \quad y = 0;$$
$$x_1 = 1, \quad x_2 = 1, \quad y = 1,$$

то она может быть реализована автоматом Мили, граф переходов которого представлен на рис. 4.96, или существенно более простой булевой формулой $y = x_1 x_2$.

Для реализации булевой функции автоматом без выходного преобразователя построим таблицу истинности, введя столбец «у» в левую часть таблицы таким образом, чтобы значения функции от этой переменной существенно не зависели (табл. 4.9).

Таблица 4.9

y	x_1	x_2	y'
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Построим ГП (рис. 4.97) по этой таблице. Этот граф внешне напоминает ГП двухвходовых триггеров, но в отличие от них в этом графе все дуги (включая петли), входящие в вершины, имеют одинаковые пометки.

Реализация систем булевых функций с помощью ГП обычно также нецелесообразна, так как такое представление весьма громоздко.

На рис. 4.98 представлен ГП автомата Мили, непосредственно реализующий табл. 4.10.

Таблица 4.10

x_1	x_2	y_1	y_2
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Реализация m булевых функций автоматом без выходного преобразователя приводит к построению ГП, являющегося полным графом, содержащим 2^m вершин.

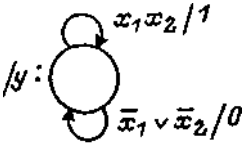

Рис. 4.96

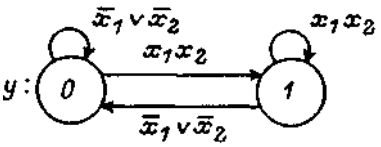

Рис. 4.97

Табл. 4.11 является расширением табл. 4.10 для построения ГП автомата без выходного преобразователя (рис. 4.99).

Таблица 4.11

y_1	y_2	x_1	x_2	y_1'	y_2'	y_1	y_2	x_1	x_2	y_1'	y_2'
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	1	0	1	1	0	0	1	0	1
0	0	1	0	0	1	1	0	1	0	0	1
0	0	1	1	1	0	1	0	1	1	1	0
0	1	0	0	0	0	1	1	0	0	0	0
0	1	0	1	0	1	1	1	0	1	0	1
0	1	1	0	0	1	1	1	1	0	0	1
0	1	1	1	1	0	1	1	1	1	1	0

Интересным является тот факт, что при исключении вершины «11» и соответствующих ей дуг оставшийся ГП реализует автомат с памятью.

В рассмотренных ГП в качестве пометок дуг используются булевы формулы произвольной сложности. Это позволяет сокращать число состояний автоматов. Покажем, что ограничение класса пометок дуг ГП только отдельными переменными либо их инверсиями приводит к увеличению числа состояний автоматов.

При учете этого ограничения вопрос о реализации булевых функций автоматами приобретает смысл. Такая постановка задачи соответствует, например, последовательному вычислению булевых формул с помощью условных переходов и при-

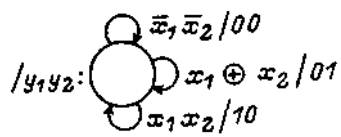

Рис. 4.98

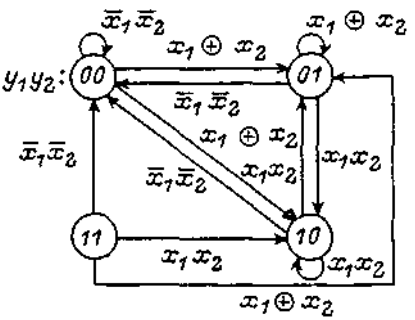

Рис. 4.99

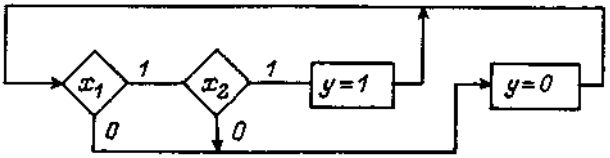

Рис. 4.100

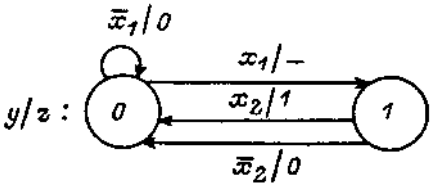

Рис. 4.101

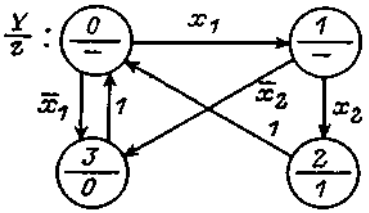

Рис. 4.102

сваиваний выходным переменным констант, как это имеет место при их реализации с использованием операций && и || языка СИ [225].

Для построения ГП в этом случае автором предлагается с помощью метода, изложенного в [88, 271], сначала построить ГСА для вычисления булевой формулы, а затем преобразовать ее в ГП.

На рис. 4.100 приведена ГСА для последовательного вычисления формулы $z = x_1 x_2$. По этой ГСА построены графы переходов автоматов Мили (4.101) и Мура (рис. 4.102). Для этих автоматов характерно сохранение значений выходов (обозначаются символом «прочерк») на некоторых переходах (для автоматов Мили) или в некоторых вершинах (для автоматов Мура).

Для СБФ, заданной ТИ, граф-схема алгоритма может быть построена с помощью модификации канонического метода Блоха [144].

Для системы булевых функций, заданной табл. 4.10, ГСА, ее реализующая, приведена на рис. 4.103, а соответствующие ГП автоматов Мили и Мура — на рис. 4.104 и 4.105.

Асимптотические оценки сложности числа состояний автоматов Мили, реализующих булевы функции и их системы, приведены в [96]. Оценки являются экспоненциальными.

В заключение раздела остановимся на вопросе, на каком языке проводить описание поведения автоматов без памяти в рамках SWITCH-технологии, при использовании которой автоматы с памятью описываются графами переходов?

Выше в этом разделе было показано, что описание автоматов без памяти графами переходов весьма громоздко. Поэтому хотелось бы, с одной стороны, чтобы язык спецификаций позволил уменьшить сложность описания этого класса автоматов, а с другой — для всего класса автоматов (как с памятью, так и без памяти) использовать один и тот же язык.

Этот компромисс может быть найден, если обратить внимание на тот факт, что «алфавитом» языка графа переходов можно считать следу-

ющие «буквы»: вершины; дуги, включая петли; цифры, помечающие вершины и (или) дуги; булевы формулы, помечающие дуги.

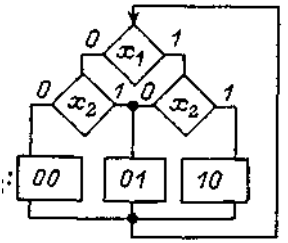

Рис. 4.103

При таком понимании «алфавита» языка графов переходов использование булевых формул для описания автоматов без памяти можно рассматривать как частный случай применения графов переходов.

При этом, описывая в составе одного алгоритма компоненты, соответствующие автоматам с памятью, графами переходов, а компоненты, соответствующие автоматам без памяти, булевыми формулами, можно считать, что алгоритм описан на одном языке спецификаций — языке графов переходов.

Такой подход методически целесообразен по двум причинам.

Во-первых, весьма часто бывают случаи, когда, реализуя алгоритм управления и сигнализации в виде одной компоненты, он является автоматом с памятью со сравнительно большим числом состояний и описывается поэтому весьма громоздким графом переходов. При декомпозиции этого алгоритма на два отдельных алгоритма — управления и сигнализации — первый из них обычно является автоматом с памятью, а второй — автоматом без памяти. В этой ситуации вряд ли можно считать, что декомпозиция приводит к смене языка спецификаций, и, описывая алгоритм сигнализации булевыми формулами, их целесообразно рассматривать как частный случай графов переходов.

Вторая из указанных выше причин состоит в том, что если булевы формулы входят в состав графов переходов, то «контакты» и «функциональные блоки» в его состав не входят, и поэтому, несмотря на эквивалентность с булевыми формулами, их можно считать в отличие от

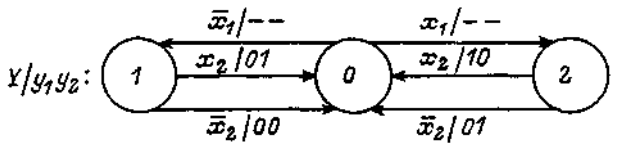

Рис. 4.104

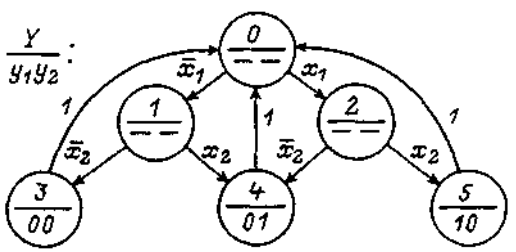

Рис. 4.105

последних представителями других языков спецификаций. Допустимость использования таких языков в рамках языка «Графсет» нарушает его «чистоту».

4.4.2. О взаимосвязи числа вершин и сложности формул в ГП

Рассмотрим две задачи, первая из которых состоит в минимизации числа вершин ГП за счет усложнения пометок его дуг, а вторая — в упрощении пометок за счет увеличения числа вершин.

Пусть требуется построить автомат, управляющий лифтом для трехэтажного дома (рис. 4.106). Управление осуществляется кнопками без памяти x_1, x_2, x_3. Информация о достижении лифтом i-го этажа передается в автомат от сигнализатора положения лифта Q_i $(i = 1, 2, 3)$. Автомат формирует следующие выходные сигналы: z_1 — подъем, z_2 — спуск, z_3 — стоп.

Для упрощения изображения в ГП, используемых в этом примере, обеспечение их полноты и непротиворечивости не производится.

Автомат с простейшими формулами на дугах ГП имеет число состояний $s = 11$ (рис. 4.107). Можно показать, что в этом случае $s > n^2$, где n — число этажей.

Уменьшение числа состояний $(s = 7)$ достигается за счет усложнения формул на дугах (рис. 4.108). При этом $s = 3n - 2$.

Однотипность структуры полученного ГП позволяет сократить число состояний автомата до $s = 3$ за счет дальнейшего усложнения формул на дугах (рис. 4.109). С ростом n число состояний не изменяется, а сложность формул растет:

$$F_{\text{п}} = \bigvee_{i=1}^{n-1} Q_i \left(\bigvee_{j=i+1}^{n} X_j \right); \qquad F_{\text{п}1} = Q_n \vee \left(\bigvee_{i=2}^{n-1} Q_i X_i \right);$$

$$F_{\text{с}} = \bigvee_{i=2}^{n} Q_i \left(\bigvee_{j=1}^{i-1} X_j \right); \qquad F_{\text{с}1} = Q_1 \vee \left(\bigvee_{i=2}^{n-1} Q_i X_i \right).$$

При этом необходимо отметить, что сокращение числа вершин при переходе от первого ГП к третьему осуществляется не только за счет усложнения пометок на дугах, но и путем изменения типа входных переменных. Если первый ГП правильно функционирует как при потен-

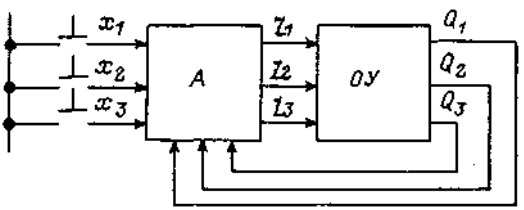

Рис. 4.106

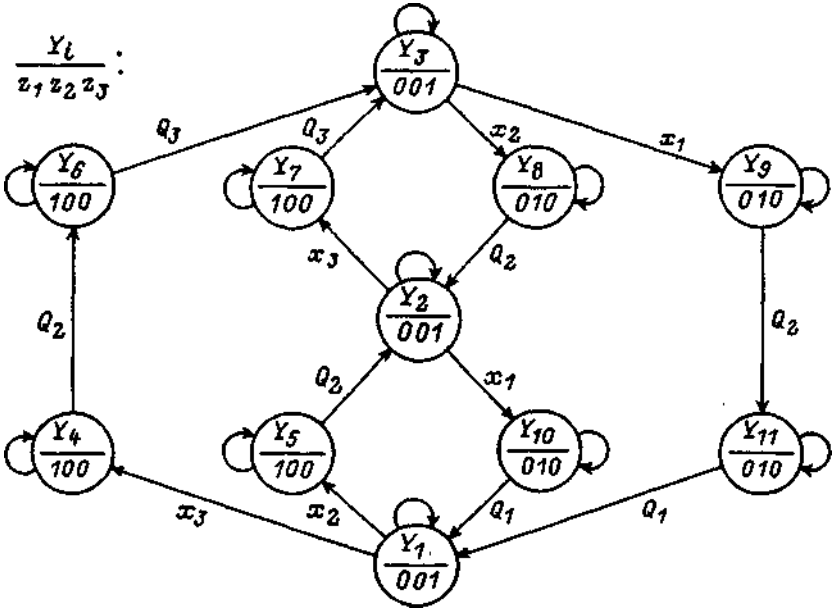

Рис. 4.107

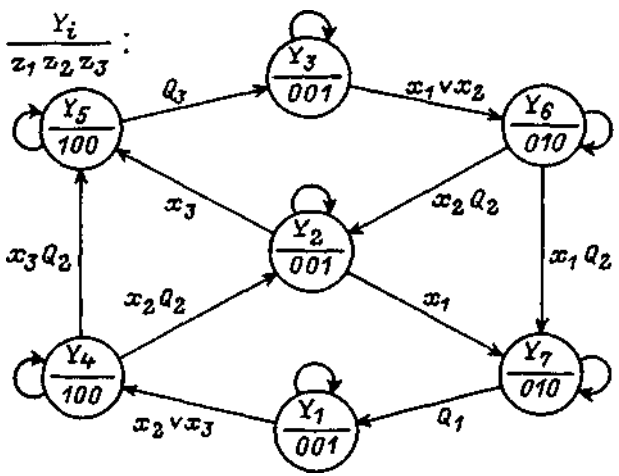

Рис. 4.108

циальных, так и импульсных сигналах, то второй и третий графы — только при потенциальных сигналах, длительность которых должна быть не менее времени соответствующих переходных процессов.

В задаче, рассмотренной в гл. 9, число состояний в автомате сокращается без изменения типа входных переменных за счет расширения

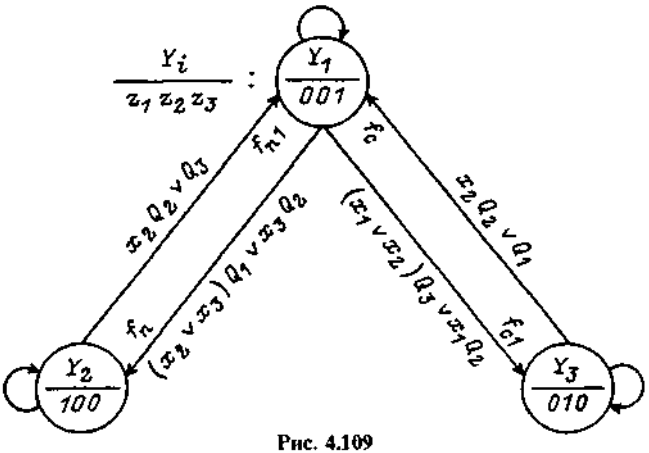

Рис. 4.109

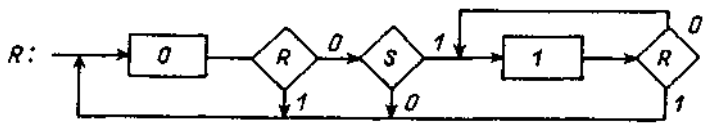

Рис. 4.110

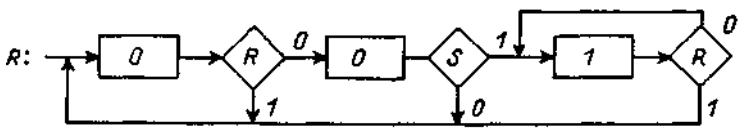

Рис. 4.111

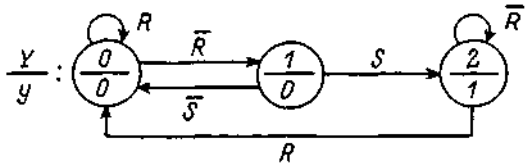

Рис. 4.112

модели комплекса «УА—ОУ» введением дополнительных элементов памяти, внешних относительно автомата. Это приводит к необходимости в общем случае формировать в вершинах ГП автомата кроме выходных сигналов на ИМ, СПИ и ФЭЗ также и сигналы, воздействующие на эти элементы памяти, а на дугах графа — проверять (кроме значений входных и временных переменных, а также выходных и внутренних переменных других графов) значения переменных (флагов) на выходах введенных элементов памяти.

При этом число дополнительных элементов памяти (в общем случае многозначных) зависит от сложности реализуемого алгоритма.

Перейдем к рассмотрению второй (обратной) задачи — преобразование ГП со сложными пометками в ГП, в котором в качестве пометок допустимо использование только отдельных входных переменных либо их инверсий.

Продемонстрируем решение этой задачи на примере *R*-триггера (рис. 4.15), осуществляя указанное преобразование за счет увеличения числа вершин ГП. Для этого по исходному ГП построим вспомогательную ГСА с возвратами назад, в которой булевы формулы реализованы бинарными граф-схемами (рис. 4.110). Преобразуем эту ГСА, вводя между каждой парой смежных условных вершин операторную вершину, сохраняющую значение выхода (рис. 4.111). Выполняя обратное преобразование и переходя к автомату Мура, получим искомый ГП (рис. 4.112), предназначенный для реализации с помощью конструкции `switch`.

4.4.3. Преобразование автоматов Мили в автоматы Мура

В системах логического управления в большинстве случаев наиболее удобным является описание алгоритма в виде ГП автомата Мура. Однако существуют автоматы, для которых более естественным является исходное задание в виде кодированной таблицы переходов и выходов.

При этом если функции выходов существенно зависят от входных переменных, то наиболее компактным является представление автомата в виде ГП автомата Мили.

К этому классу относится, например, автомат, реализующий последовательный сумматор. Это устройство имеет два входа — x_1 и x_2, на которые последовательно, начиная с младших разрядов, подаются двоичные разряды складываемых чисел (на вход x_1— разряды первого числа, на вход x_2 — второго). Одноименные разряды на входы x_1 и x_2 подаются параллельно. Разряды суммы (начиная с младшего) последовательно формируются на выходе *S*. Перенос формируется на выходе *P* и используется автоматом в качестве промежуточной переменной. После обработки всех разрядов складываемых чисел значение на выходе *P* равно старшему разряду суммы.

Функционирование последовательного сумматора описывается следующей кодированной таблицей переходов и выходов (КТПВ) (табл. 4.12).

Таблица 4.12

P	x_1	x_2	P'	S	P	x_1	x_2	P'	S
0	0	0	0	0	1	0	0	0	1
0	0	1	0	1	1	0	1	1	0
0	1	0	0	1	1	1	0	1	0
0	1	1	1	0	1	1	1	1	1

Построим -полный и непротиворечивый ГП по этой таблице. Из состояния «О» в состояние «О» автомат переходит при $x_1 = x_2 = 0$, формируя выходной сигнал $S = 0$. Тот же переход осуществляется п $x_1 = 0$, $x_2 = 1$, однако в этом случае $S = 1$. Продолжая аналогичное рассмотрение таблицы, построим ГП (рис. 4.113), описывающий автомат Мили.

Выберем структурную модель автомата. Если использовать автомат Мили первого рода (рис. 3.8), то он адекватно описывается кодированной таблицей переходов и выходов, в которой новое состояние и выход зависят от предыдущего состояния и входов.

Единая таблица переходов и выходов является наиболее естественной и традиционной для теории автоматов, и поэтому для автоматов Мили обычно применяется эта модель.

Если программирование в данном случае выполнять по СБФ, то она может быть получена по КТПВ (табл. 4.12) традиционным для теории автоматов путем:

$$\begin{cases} P' = (x_1 \vee x_2) P \vee x_1 x_2; \\ S = x_1 \oplus x_2 \oplus P. \end{cases}$$

Эта же СБФ может быть получена непосредственно по ГП (рис. 4.113), записывая условия, определяемые предыдущим состоянием и входным воздействием, при которых P и S соответственно равны единице:

$$\begin{cases} P' = x_1 x_2 \overline{P} \vee x_1 x_2 P \vee (x_1 \oplus x_2) P = (x_1 \vee x_2) P \vee x_1 x_2; \\ S = (x_1 \oplus x_2) \overline{P} \vee \overline{x}_1 \overline{x}_2 P \vee x_1 x_2 P = x_1 \oplus x_2 \oplus P. \end{cases}$$

Рассмотрим методы перехода от автоматов Мили к автоматам других типов.

Осуществим переход к автомату без выходного преобразователя за счет расширения КТПВ (табл. 4.12). Выполним это расширение за счет введения переменной S в левую часть таблицы (табл. 4.12) таким образом, чтобы функции P' и S' в правой части новой таблицы существенно не зависели от вновь введенной переменной (табл. 4.13).

Таблица 4.13

P	S	x_1	x_2	P'	S'	P	S	x_1	x_2	P'	S'
0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	1	1	0	0	1	1	0
0	0	1	0	0	1	1	0	1	0	1	0
0	0	1	1	1	0	1	0	1	1	1	1
0	1	0	0	0	0	1	1	0	0	0	1
0	1	0	1	0	1	1	1	0	1	1	0
0	1	1	0	0	1	1	1	1	0	1	0
0	1	1	1	1	0	1	1	1	1	1	1

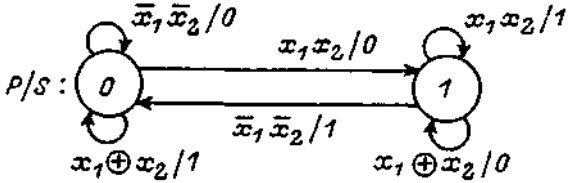

Рис. 4.113

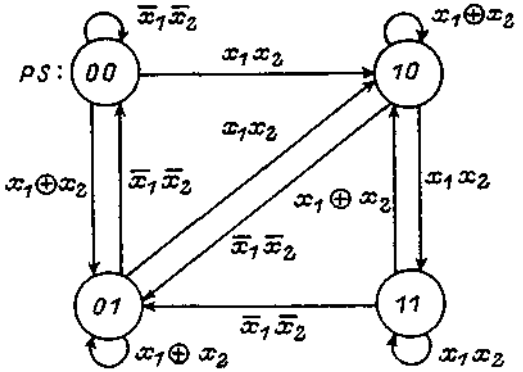

Рис. 4.114

Эта таблица описывает работу автомата без выходного преобразователя, эквивалентного исходному автомату Мили. Построим по этой таблице ГП (рис. 4.114) для нового автомата и преобразуем его в ГП автомата Мура с многозначным кодированием состояний (рис. 4.115).

Обратим внимание на тот факт, что, несмотря на то что в этом ГП все вершины имеют петли, первая из них неустойчива при $x_1 = x_2 = 1$, а третья — при $x_1 = x_2 = 0$.

Из изложенного следует, что изменение модели связано в данном случае с увеличением числа состояний.

Рассмотренный табличный метод ограничивает размерность решаемых задач. Поэтому изложим графический метод, использующий ГСА в качестве промежуточного языка.

Необходимым условием того, чтобы автомат Мура имел большее число состояний по сравнению с эквивалентным автоматом Мили, является наличие кратных дуг в ГП автомата Мили. Если кратные дуги в ГП автомата Мили отсутствуют, то эквивалентный автомат Мура может иметь то же или большее число состояний.

При этом всегда может быть построен автомат Мура, число состояний s в котором равно числу дуг l в эквивалентном автомате Мили. Это число

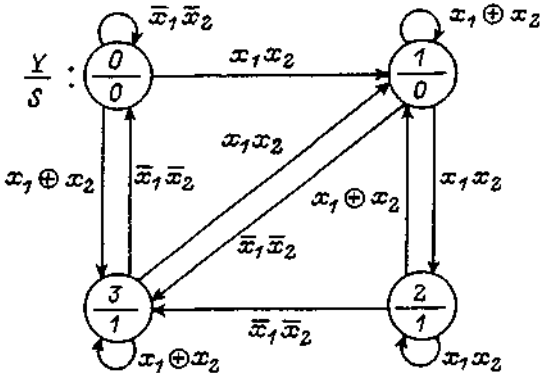

Рис. 4.115

состояний является верхней оценкой числа состояний для данного автомата Мура, так как в ряде случаев оно может быть уменьшено за счет объединения эквивалентных состояний.

Переход от автомата Мили к автомату Мура и сокращение числа состояний в последнем будем производить используя в качестве промежуточного язык ГСА.

При этом каждой дуге ГП автомата Мили будет соответствовать одна операторная вершина ГСА, которой в свою очередь соответствует одно состояние автомата Мура или одна вершина в его ГП.

Пусть автомат Мили задан ГП (рис. 4.116). Так как в этом случае $l = 8$, то для него может быть построен ГП автомата Мура с $s = 8$. Для этого по ГП автомата Мили построим изоморфную ГСА (рис. 4.117), которая должна начинаться с некоторого начального состояния, которое (как и все остальные состояния) обозначим символом «крест» и цифрой, соответствующей номеру рассматриваемого состояния.

Такой способ построения ГСА базируется на известном в аппаратной теории автоматов [16, 36] факте, что при построении ГП автомата Мили по ГСА его состояниям соответствуют те точки в ГСА, которые следуют за операторными вершинами.

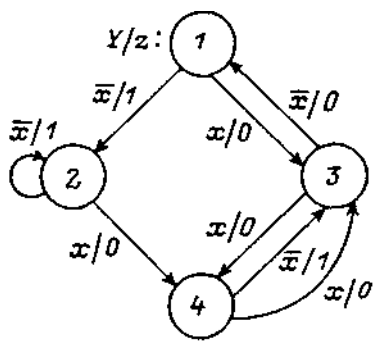

Рис. 4.116

Дальнейшее преобразование основывается на том факте [16, 36], что каждому состоянию автомата Мура соответствует одна операторная вершина в ГСА.

Из изложенного следует, что, уменьшая число операторных вершин в ГСА, мы тем самым сокращаем число состояний в соответст-

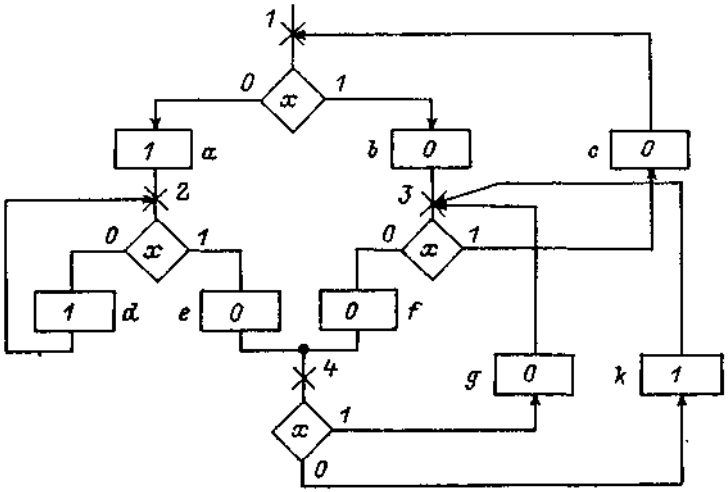

Рис. 4.117

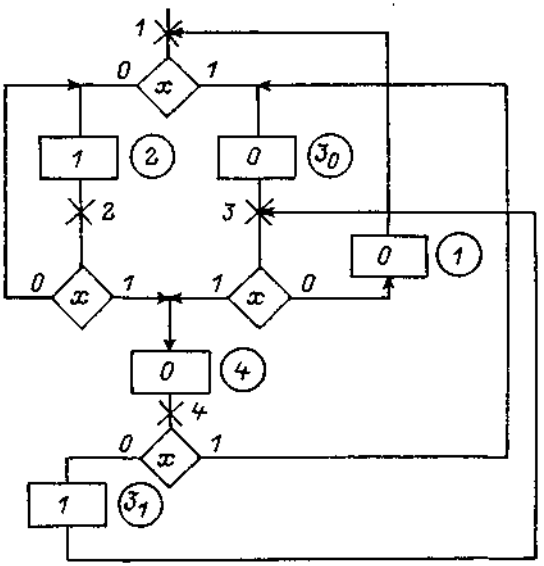

Рис. 4.118

вующем автомате Мура. Поэтому пометим операторные вершины в ГСА (рис. 4.117) буквами и выполним их минимизацию за счет объединения эквивалентных вершин: *a* и *d, b* и *g, e* и *l*. Так как в полученной ГСА (рис. 4.118) число операторных вершин равно пяти, то по ней может быть построен изоморфный ГП автомата Мура с пятью вершинами.

ГСА автомата Мура должна начинаться с операторной вершины, соответствующей наименьшей неиндексированной цифре в кружке. По-

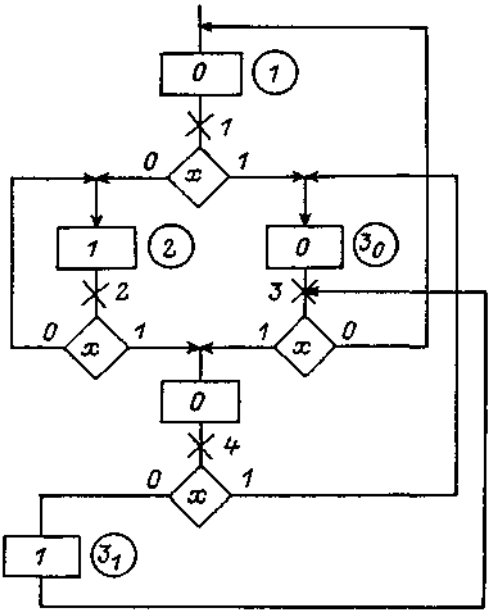

Рис. 4.119

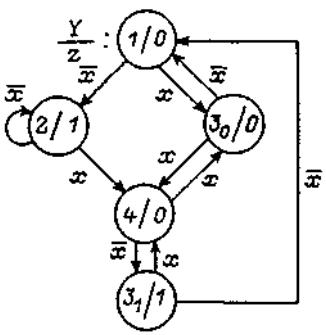

Рис. 4.120

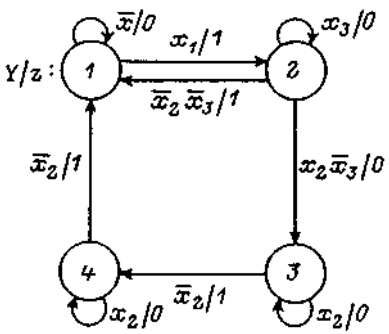

Рис. 4.121

этому изобразим ГСА (рис. 4.118) иначе (рис. 4.119). Этой ГСА соответствует ГП на рис. 4.120.

Переход от автомата Мура к автомату Мили осуществляется в обратном порядке. На рис. 4.118 крестиками помечены точки, соответствующие состояниям автомата Мили.

Приведем пример ГП автомата Мили без кратных дуг (рис. 4.121), которому соответствует автомат Мура с большим числом состояний (рис. 4.122).

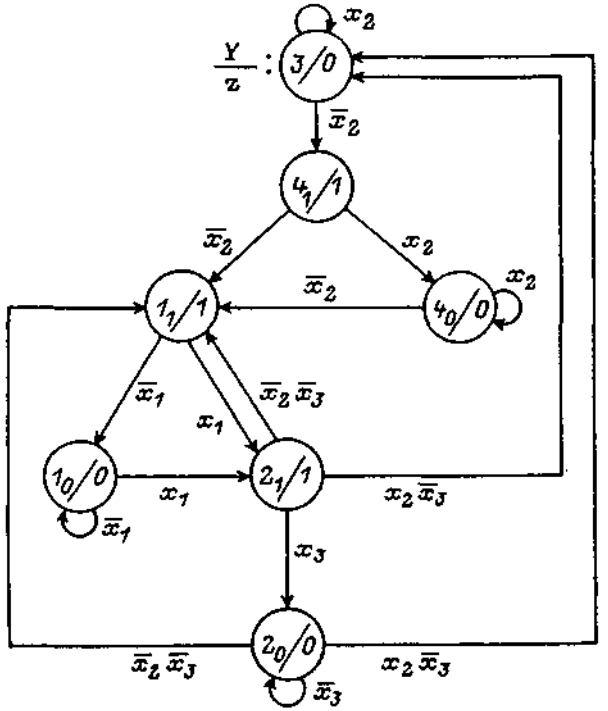

Рис. 4.122

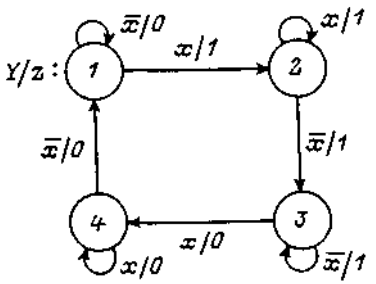

Рис. 4.123

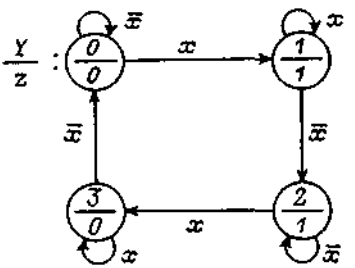

Рис. 4.124

Приведем также пример ГП автомата Мили без кратных дуг (рис. 4.123), которому соответствует автомат Мура с тем же числом состояний (рис. 4.124).

Сравнение табличного и графического методов перехода от автомата Мили к автомату Мура показывает, что в первом случае всегда строится полный по числу вершин ГП автомата Мура, содержащий число вершин, равное два в степени, который описывается той же СБФ, что и автомат Мили,

а во втором — неполный по числу вершин ГП, что приводит к возможности получения различных СБФ, в том числе и отличных от исходной.

4.4.4. Об эквивалентности автоматов Мили первого и второго рода

В предыдущем разделе показано, что рассмотренный ГП и кодированная таблица переходов и выходов соответствуют автомату Мили первого рода. Однако при задании автомата СБФ использование автомата Мили второго рода может оказаться целесообразнее. При этом возникает вопрос о взаимном преобразовании этих моделей [3].

Переход от модели второго рода к модели первого рода всегда возможен (разд. 3.2) только за счет изменения комбинационной схемы выходного преобразователя и места съема сигнала состояния, что осуществляется весьма просто. Так, например, если автомат Мили второго рода (рис. 4.125) задан СБФ:

$$y' = \bar{x}_2(x_3 \vee y); \qquad z = x_1\bar{y}_1',$$

то переход к этой модели первого рода (рис. 4.126) осуществляется за счет подстановки:

$$y' = \bar{x}_2(x_3 \vee y); \qquad z = x_1\,\overline{\bar{x}_2(x_3 \vee y)} = x_1(x_2 \vee \bar{x}_3\,\bar{y}). \tag{4.3}$$

Таким образом, переход от автомата Мили второго рода к автомату Мили первого рода связан только с изменением комбинационной схемы выходного преобразователя.

По СБФ автомата Мили первого рода может быть построена единая кодированная таблица переходов и выходов, по которой легко строится ГП. Автомат Мили второго рода в табличной форме задается весьма неудобно — двумя таблицами (переходов и выходов).

Для рассмотренного примера единая таблица (табл. 4.14) имеет следующий вид:

Таблица 4.14

y	x_1	x_2	x_3	y'	z	y	x_1	x_2	x_3	y'	z
0	0	0	0	0	0	1	0	0	0	1	0
0	0	0	1	1	0	1	0	0	1	1	0
0	0	1	0	0	0	1	0	1	0	0	0
0	0	1	1	0	0	1	0	1	1	0	0
0	1	0	0	0	1	1	1	0	0	1	0
0	1	0	1	1	0	1	1	0	1	1	0
0	1	1	0	0	1	1	1	1	0	0	1
0	1	1	1	0	1	1	1	1	1	0	1

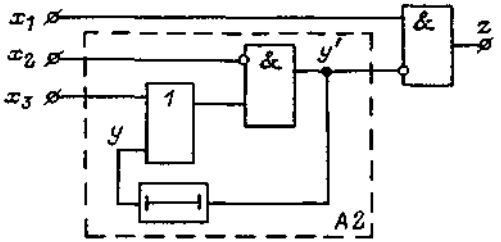

Рис. 4.125

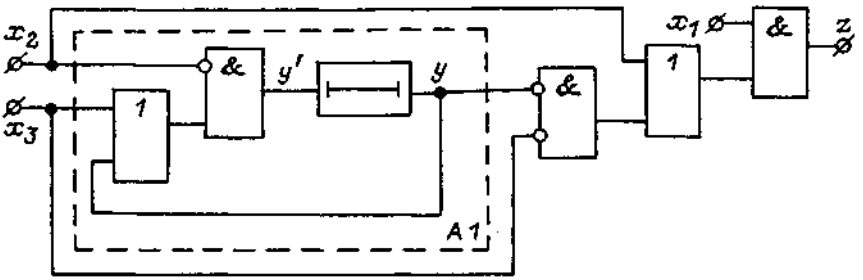

Рис. 4.126

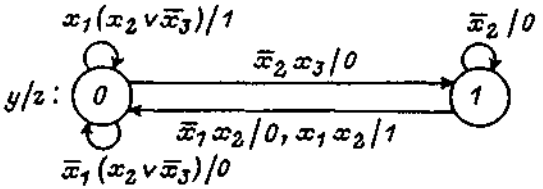

Рис. 4.127

ГП автомата Мили, построенный по этой таблице, приведен на рис. 4.127, а эквивалентный ему ГП автомата без выходного преобразователя, полученный за счет введения переменной z в левую часть табл. 4.14, изображен на рис. 4.128.

Переход от автомата Мили первого рода к автомату Мили второго рода не всегда возможен без введения дополнительных элементов задержки.

Универсальный метод перехода от автомата без выходного преобразователя первого рода к аналогичной модели второго рода базируется на преобразовании, представленном на рис. 4.129. Таким образом, сигнал состояния во второй модели опережает сигнал состояния в первой модели на величину задержки, равной одному «такту».

Следовательно, если требуется преобразовать автомат (рис. 4.126), то универсальный метод обеспечивает построение схемы, представленной на рис. 4.130. Схема автомата A2 приведена на рис. 4.125.

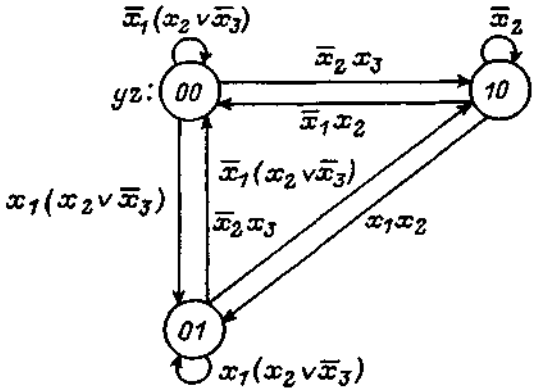

Рис. 4.128

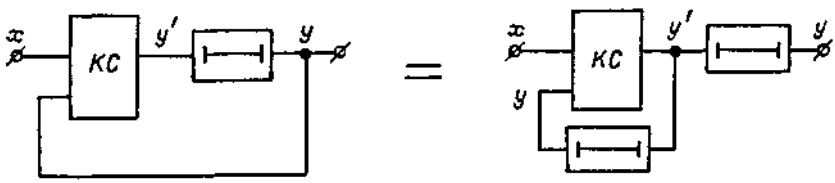

Рис. 4.129

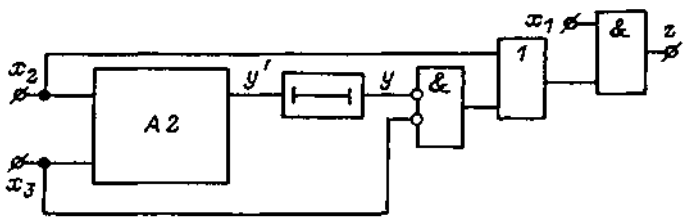

Рис. 4.130

Однако универсальность указанного метода приводит в данном случае явно к неоптимальному результату, так как существует более простая схема (рис. 4.125).

Предложим аппарат для решения рассматриваемой задачи для автоматов с двумя состояниями, отличный от традиционно применяемого при решении логических уравнений [29].

Действительно, пусть требуется решить уравнение $a = x \vee b$, где $a, x, b \in \{0, 1\}$. В табличной форме это уравнение решается весьма просто. Для этого достаточно преобразовать таблицу истинности (табл. 4.15), соответствующую этому уравнению, в таблицу, определяющую решение (табл. 4.16).

x	b	a
0	0	0
0	1	1
1	0	1
1	1	1

Однако получение аналитического решения требует перехода от логической формы к арифметической [80, 82]. Так как $a = x \vee b = x + b - xb$, то $x = (a - b)/(1 - b)$. Из этого соотношения следует, что при $a = 0$, $b = 1$ результатом является бесконечность (нет решений), а при $a = b = 1$ — неопределенность (два решения).

Для уравнения $a = x \,\&\, b$ решением является соотношение $x = a/b$ или табл. 4.17.

a	b	x
0	0	0,1
0	1	0
1	0	—
1	1	1

Логически можно решить уравнение $a = x \oplus b$. Действительно, $a \oplus b = x \oplus b \oplus b = x$. Это объясняется тем, что таблица решения в этом случае не имеет ни бесконечности, ни неопределенности.

Возвращаясь к поставленной задаче, в качестве первого примера определим по СБФ (4.3) соотношение вида $z = f(x, y')$, не пользуясь операцией подстановки, невозможной в общем случае.

Для этого сначала ортогонализуем первую формулу в (4.3):

$$y' = \bar{x}_2(x_3 \vee y) = \bar{x}_2(x_3 \vee \bar{x}_3 y)$$

В ортогональной форме можно заменить символы «V» на символы «+» [82], а $\bar{x}_i$ на $1 - x_i$. Тогда

$$y' = (1 - x_2)(x_3 + (1 - x_3)y).$$

При этом

$$y = \frac{y' - x_3 + x_2 x_3}{(1 - x_2)(1 - x_3)}. \qquad (4.4)$$

Запишем вторую формулу в (4.3) в арифметической форме:

$$z = x_1(x_2 \vee \bar{x}_3\bar{y}) = x_1(x_2 \vee \bar{x}_2\bar{x}_3\bar{y}) = x_1(x_2 + (1 - x_2)(1 - x_3)(1 - y)).$$

Подставляя в это выражение соотношение (4.4), после упрощения получим:

$$z = x_1\bar{y}'.$$

Рассмотрим второй пример обратного перехода. Пусть автомат Мили первого рода (рис. 4.131) задан СБФ:

$$y' = \bar{x}_2(x_3 \vee y); \qquad z = x_1(x_3 \vee \bar{x}_2 y). \tag{4.5}$$

Попытаемся решить эту задачу без использования универсального метода с целью нахождения более простого решения.

В этом случае

$$z = x_1(x_3 + (1 - x_3)(1 - x_2)y).$$

Подставляя в это выражение соотношение (4.4) и учитывая из первого соотношения в (4.5), что x_2x_3 и y' ортогональны, получим:

$$z = x_1(x_2x_3 \vee y').$$

Покажем, что при этом выполняется весьма странное равенство:

$$x_1(x_2x_3 \vee y') = x_1(x_3 \vee y').$$

Подставляя в левую и правую часть этого равенства выражение для y' из системы (4.5), получим:

$$x_1(x_2x_3 \vee \bar{x}_2x_3 \vee \bar{x}_2y) = x_1(x_3 \vee \bar{x}_2x_3 \vee \bar{x}_2y); \qquad x_1(x_3 \vee \bar{x}_2y) = x_1(x_3 \vee \bar{x}_2y).$$

Таким образом, вместо системы (4.5) можно записать более простую СБФ:

$$y' = \bar{x}_2(x_3 \vee y); \qquad z = x_1(x_3 \vee y'),$$

соответствующую схеме (рис. 4.132).

Рассмотрим третий пример обратного перехода. Пусть автомат первого рода (рис. 4.133) задан СБФ вида;

$$y' = \bar{x}_2(x_3 \vee y); \qquad z = x_1\bar{y}.$$

Подставляя в выражение $z = x_1(1 - y)$ соотношение (4.4), получим:

$$z = x_1 \frac{(1 - x_2 - y')}{(1 - x_2)(1 - x_3)}.$$

Это выражение не удается преобразовать к логическому виду, что свидетельствует о том, что в классе комбинационных схем указанная задача не решается. Схема, содержащая элемент задержки на выходе автомата А2, строится универсальным методом (рис. 4.134).

Из изложенного следует, что в одних случаях более простым является автомат Мили второго рода (первый и второй примеры), а в других — автомат Мили первого рода (третий пример).

Однако для построения ГП (одного и того же для обеих моделей структурной реализации) должен использоваться автомат Мили первого рода, так как он описывается единой кодированной таблицей переходов

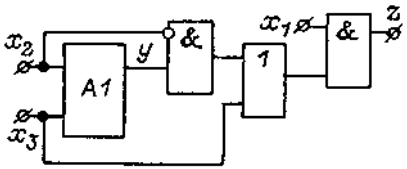

Рис. 4.131

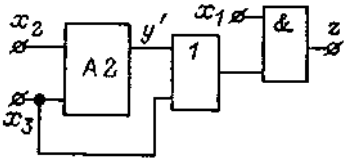

Рис. 4.132

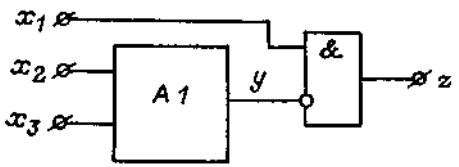

Рис. 4.133

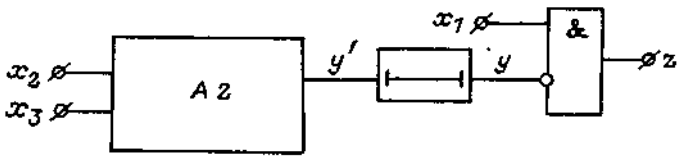

Рис. 4.134

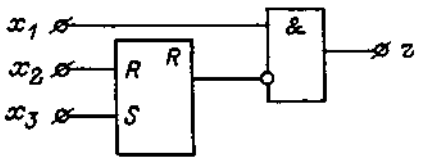

Рис. 4.135

и выходов, однозначно соответствующей символике, применяемой при построении ГП автомата Мили.

К рассмотренному чрезвычайно близок вопрос о чтении функциональных схем, содержащих триггеры. Пусть задана схема (рис. 4.135), построенная на базе *R*-триггера. Требуется выполнить анализ работы этой схемы. При этом сначала необходимо ответить на вопрос: является ли эта схема автоматом Мили первого или второго рода?

В силу того что *R*-триггер описывается формулой $y' = \bar{R}(S \vee y)$,, то рассматриваемую схему естественно описать СБФ:

$$y' = \bar{x}_2(x_3 \vee y); \quad z = x_1\bar{y}'.$$

Таким образом, эта схема является автоматом Мили второго рода. Анализ схемы, описываемой этой СБФ, выполнен в первом примере настоящего раздела.

4.4.5. Минимизация числа состояний автоматов Мили

Несмотря на то что при минимизации числа состояний автомата происходит изменение структуры ГП, построенного по исходному заданию Разработчиком и поэтому понятного ему, при ограничениях на объем памяти такую минимизацию проводить все-таки целесообразно.

Методы минимизации числа состояний основаны на склеивании вершин ГП (совместно с исходящими дугами), соответствующих эквивалентным состояниям автомата. Эквивалентными называют [3] такие состояния автомата, которые удовлетворяют трем следующим условиям:

— им соответствуют одни и те же значения входов;

— им соответствуют одни и те же значения выходов;

— любой последовательности значений входов соответствует одна и та же последовательность значений выходов независимо от того, какое из рассматриваемых состояний взять за исходное.

Рассмотрим в качестве примера метод минимизации числа состояний автомата Мили, предложенный Хаффменом [211]. Пусть автомат Мили с одним входом и одним выходом и семью состояниями задан [23] таблицей переходов и выходов (табл. 4.18).

Таблица 4.18

x_i	Y						
	1	**2**	**3**	**4**	**5**	**6**	**7**
0	**2, 1**	**5,1**	**7,0**	**1, 0**	**5, 1**	**4, 0**	**5,0**
1	**3, 0**	**6,0**	**4,1**	**3, 1**	**6, 0**	**7, 1**	**6,1**

Если двум состояниям автомата соответствуют различные столбцы значений выходов, то эти состояния не эквивалентны. Поэтому разобьем множество состояний заданного автомата на классы условно эквивалентных состояний, в каждый из которых входят состояния с одинаковыми столбцами значений выходов: $Y_1 = \{1,2, 5\}$; $Y_2 = (3, 4, 6, 7)$.

Преобразуем исходную таблицу переходов, используя обозначения введенных классов эквивалентности (табл. 4.19).

Таблица 4.19

x_i	Y						
	1	2	3	4	5	6	7
0	Y_1	Y_1	Y_2	Y_1	Y_1	Y_2	Y_1
1	Y_2	Y_2	Y_2	Y_2	Y_2	Y_2	Y_2

Из рассмотрения этой таблицы следует, что второй класс условно эквивалентных состояний не является классом эквивалентных состояний, так как состояниям этого класса соответствуют различные значения столбцов. Поэтому разобьем этот класс на новые классы условно эквивалентных состояний, каждый из которых образован состояниями с одинаковыми значениями столбцов: Y_1, = {1,2, 5); Y_{21} = (3, 6}; Y_{22}={4, 7).

Преобразуем исходную таблицу переходов, используя вновь введенные обозначения (табл. 4.20).

Таблица 4.20

x_i	Y						
	1	2	3	4	5	6	7
0	Y_1	Y_1	Y_{22}	Y_1	Y_1	Y_{22}	Y_1
1	Y_{21}	Y_{21}	Y_{22}	Y_{21}	Y_{21}	Y_{22}	Y_{21}

В силу того что табл. 4.19 и 4.20 изоморфны, классы K_1, K_{21}, K_{22} являются классами эквивалентности и поэтому все состояния одного класса ведут себя как одно и то же состояние, которым они и могут быть заменены. Исходя из изложенного, число состояний в заданном автомате Мили сокращается с семи до трех (табл. 4.21).

Таблица 4.21

x_i	Y		
	1	21	22
0	1, 1	22, 0	1, 0
1	21, 0	22, 1	21, 1

В заключение раздела обратим внимание на тот факт, что в рассмотренном автомате возможны автоколебания.

4.4.6. Настраиваемые графы переходов

Пусть заданы два ГП, приведенные на рис. 4.136, 4.137. Требуется построить один ГП, из которого путем настройки может быть получен каждый из этих графов.

Будем использовать в этом случае простейший способ настройки — присвоение настроечной переменной z констант 0 и 1. Присвоим первую константу первому графу, а вторую — второму.

Построим для каждого графа формулы, описывающие его функционирование. В силу того что первый ГП не полон по состояниям, доопре-

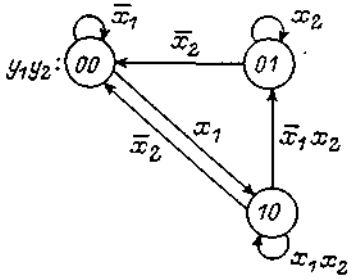

Рис. 4.136

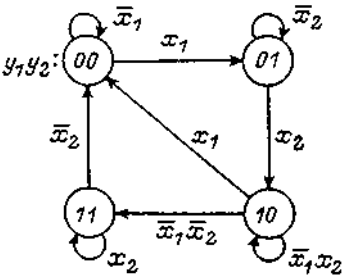

Рис. 4.137

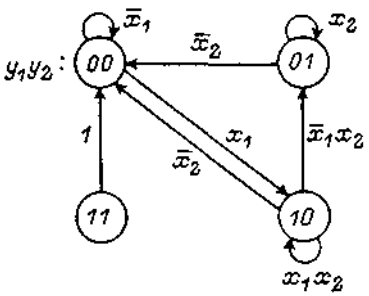

Рис. 4.138

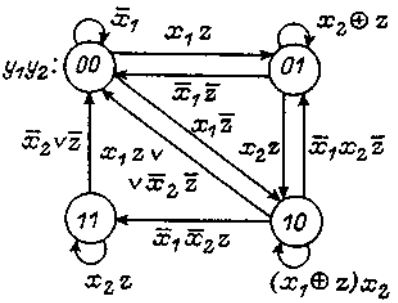

Рис. 4.139

делим этот граф таким образом, чтобы он был максимально изоморфен второму графу (рис. 4.138). При этом

$$y_1' = x_1\bar{y}_1\bar{y}_2 \vee 0\,\bar{y}_1 y_2 \vee x_1x_2y_1\bar{y}_2 \vee 0\,y_1y_2;$$
$$y_2' = 0\,\bar{y}_1\bar{y}_2 \vee x_2\bar{y}_1 y_2 \vee \bar{x}_1 x_2y_1\bar{y}_2 \vee 0\,y_1y_2.$$

В силу того что второй ГП полон по состояниям, СБФ в этом случае может быть построена непосредственно по этому графу:

$$y_1'' = x_2y_1y_2 \vee \bar{x}_1\bar{x}_2 y_1\bar{y}_2 \vee \bar{x}_1 x_2y_1\bar{y}_2 \vee x_2\bar{y}_1 y_2 =$$
$$= 0\,\bar{y}_1\bar{y}_2 \vee x_2\bar{y}_1 y_2 \vee \bar{x}_1 y_1\bar{y}_2 \vee x_2y_1 y_2;$$

$$y_2'' = x_2y_1y_2 \vee \bar{x}_1\bar{x}_2 y_1\bar{y}_2 \vee x_1\bar{y}_1\bar{y}_2 \vee \bar{x}_2\bar{y}_1 y_2 =$$
$$= x_1\bar{y}_1\bar{y}_2 \vee \bar{x}_2\bar{y}_1 y_2 \vee \bar{x}_1\bar{x}_2 y_1\bar{y}_2 \vee x_2y_1 y_2.$$

Объединим построенные формулы с помощью переменной z:

$$Y_1 = y_1'\bar{z} \vee y_1''z = x_1\bar{z}\,\bar{y}_1\bar{y}_2 \vee x_2 z\,\bar{y}_1 y_2 \vee$$
$$\vee (x_1x_2\bar{z} \vee \bar{x}_1 z)\,y_1\bar{y}_2 \vee x_2 z\,y_1y_2;$$

$$Y_2 = y_2'\bar{z} \vee y_2''z = x_1 z\,\bar{y}_1\bar{y}_2 \vee (x_2 \oplus z)\,\bar{y}_1 y_2 \vee$$
$$\vee \bar{x}_1 (x_2 \oplus z)\,y_1\bar{y}_2 \vee x_2 z\,y_1y_2.$$

Для построения настраиваемого ГП заполним по этой СБФ кодированную таблицу переходов (табл. 4.22).

Таблица 4.22

x_1	x_2	z	$y_1=0,\ y_2=0$		$y_1=0,\ y_2=1$		$y_1=1,\ y_2=0$		$y_1=1,\ y_2=1$	
			Y_1	Y_2	Y_1	Y_2	Y_1	Y_2	Y_1	Y_2
0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	1	1	0	0
0	1	0	0	0	0	1	0	1	0	0
0	1	1	0	0	1	0	1	0	1	1
1	0	0	1	0	0	0	0	0	0	0
1	0	1	0	1	0	1	0	0	0	0
1	1	0	1	0	0	1	1	0	0	0
1	1	1	0	1	1	0	0	0	1	1

Искомый ГП, построенный по этой таблице, приведен на рис. 4.139. Этот граф при $z=0$ преобразуется в доопределенный первый граф (рис. 4.138), а при $z=1$ — во второй (рис. 4.137).

В заключение раздела отметим, что если первый граф доопределить по критерию максимальной простоты СБФ, его описывающей, то объединенный граф будет обладать более сложной структурой.

Изложенный подход решает проблему построения многофункциональных модулей с простой настройкой [52] для автоматов с памятью.

Необходимо отметить также, что и каждый автомат в свою очередь может рассматриваться как многофункциональный модуль, реализующий в каждом состоянии соответствующую ортогональную СБФ, зависящую от входных воздействий.

Таким образом, внутренние переменные можно рассматривать в качестве настроечных. Вычисленное значение СБФ, соответствующей некоторому состоянию, определяет следующее значение настроечных и выходных переменных.

Еще три подхода к построению настраиваемых графов переходов изложены в разд. 8.1, Приложениях 7 и 8 соответственно.

Первый из них состоит в построении «покрывающего» вызываемого графа, настройка которого состоит в передаче ему номера состояния головного графа.

Второй подход состоит в построении по заданным графам «порождающей» функции с формальными параметрами, которой при вызове передаются фактические параметры, соответствующие реализуемому графу.

Третий подход близок к предыдущему. При этом различные графы (объекты) объединяются в класс, для которого строится порождающая функция класса. При вызове этой функции осуществляется ее настройка на реализацию соответствующего объекта.

4.4.7. Реализация алгоритмов управления совокупностью автоматов

В ряде случаев заданный алгоритм невозможно или нецелесообразно реализовывать одним автоматом. Предположим, что этот алгоритм реализуется в виде нескольких автоматов.

При этом основными являются следующие вопросы:

— как обеспечить взаимосвязь автоматов?

— как доказать, что совокупность автоматов реализует заданный алгоритм и не выполняет ничего лишнего?

Взаимосвязь автоматов может быть обеспечена с помощью входных, внутренних и выходных переменных или их совокупности, а также введением супервизора — автомата, осуществляющего координацию работы других автоматов.

Новизна подхода, предлагаемого в настоящей работе, состоит в том, что применение языковых средств при программировании позволяет осуществлять взаимосвязь по состояниям не с помощью двоичных переменных, а на основе многозначных переменных, что резко уменьшает число внутренних переменных, используемых в программах, и делает последние более компактными и обозримыми.

Решение вопроса о правильности функционирования совокупности автоматов сильно осложняется в связи с тем, что исследовать все их функциональные возможности весьма трудоемко, и поэтому проверка правильности программ заменяется обычно тестированием.

При тестировании возникает вопрос о том, что использовать в качестве тестов. При создании систем логического управления тесты обычно выбираются на основе опыта Разработчика и технического задания, по которому осуществлялось проектирование. Однако если при таком подходе удается показать, что программа реализует задание, то доказать, что она не делает ничего лишнего, практически невозможно. Поэтому можно утверждать, что программы, принятые указанным образом, не являются полностью проверенными.

Другая особенность, затрудняющая полную проверку, состоит в том, что традиционно для рассматриваемого класса систем методика проверки функционирования является документом, в котором в строчной форме отражаются требуемые реакции системы на заданные входные воздействия.

Такая форма приемлема для комбинационных автоматов (хотя вопрос о полноте тестирования должен решаться и здесь), но не пригодна для автоматов с памятью, так как указанная выше проверка должна производиться в каждом состоянии, в то время как это понятие в практике проектирования СЛУ обычно даже не используется.

В силу того что совокупность автоматов в целом должна вести себя как некоторый единый автомат, автором предлагается строить единый ГП проверки в форме автомата Мура, в вершинах которого в числителе указываются желаемые состояния каждого из автоматов, а в знаменателе — значения их выходных переменных.

При этом тестирование совокупности автоматов состоит в воспроизведении на дисплее номеров их состояний и значений выходных переменных в качестве реакции на входные переменные, указанные на дугах ГП

проверки, с целью определения соответствия этих реакций желаемым результатам.

В случае если реакция совокупности автоматов не совпадает с желаемой, то в указанной совокупности должна быть выполнена корректировка и вновь проведено тестирование. Корректировка в большинстве случаев проводится за счет введения дополнительных блокировок между автоматами, так как к моменту тестирования совокупности обычно каждый автомат в отдельности уже проверен.

После того как выяснилось, что граф переходов проверки реализуется совокупностью автоматов, предлагается заменить в нем значения числителя номерами его состояний и получить ГП единого автомата (наконец-то получить формализованную спецификацию в виде одной компоненты).

Если построить по этому ГП программу и заменить ею программу, реализующую совокупность автоматов, то можно утверждать, что построенная программа реализует лишь только то, что проверено и ничего лишнего (на языке ГСА предложенная идея могла бы быть реализована следующим образом: по проверенным путям построить новую ГСА, которая содержит только эти пути и никаких больше).

Предложенный подход для его практического внедрения имеет один существенный недостаток — отказ от модульности программ. Вместо унификации программ на уровне отдельных сравнительно «мелких» автоматов, которые могут использоваться при создании различных программ, приходится переходить к более «крупным» объектам унификации, которые трудно применять многократно.

Таким образом, как ни странно, имеет место противоречие между модульностью программы и возможностью доказательства ее правильности.

Другой подход к доказательству правильности программ основан на использовании графов достижимых маркировок и рассмотрен в разд. 5.6.

4.4.8. Анализ поведения совокупности графов переходов

В предыдущем разделе был рассмотрен вопрос о тестировании совокупности графов переходов с помощью графа переходов проверки, который строится эвристически с целью определения желаемого поведения совокупности автоматов как единого целого.

Однако граф проверки обычно не отражает всех функциональных возможностей совокупности автоматов, и поэтому, если не осуществлять замену этой совокупности указанным графом с новым кодированием, вопрос о полноте проверки остается открытым.

Полнота проверки и отказ от тестирования (в пользу доказательства правильности — верификации) могут быть обеспечены в случае формализованного построения единого графа по заданной совокупности графов, что, правда, связано с весьма громоздкими логическими преобразованиями и большой размерностью (для реальных задач) получаемого графа. Другой недостаток этого подхода состоит в том, что ввиду отсутствия соответствующего математического аппарата приходится отказываться от

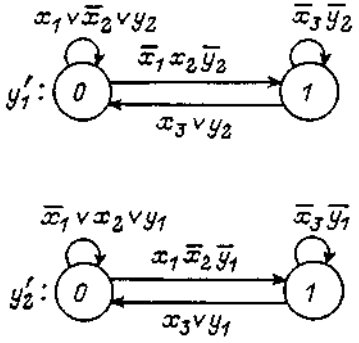

Рис. 4.140

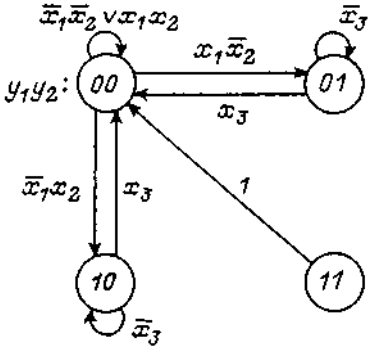

Рис. 4.141

использования многозначного кодирования, столь удобного при программировании с применением конструкции switch (гл. 5). При использовании такого кодирования для анализа поведения совокупности автоматов целесообразно строить граф достижимых маркировок.

Необходимо отметить, что при программировании автоматов по СБФ последняя одновременно описывает поведение как совокупности автоматов, так и единого автомата, им эквивалентного.

Пусть имеется СБФ вида:

$$y_1' = \bar{x}_1 x_2 \bar{y}_1 \bar{y}_2 \vee \bar{x}_3 y_1 \bar{y}_2; \qquad y_2' = x_1 \bar{x}_2 \bar{y}_1 \bar{y}_2 \vee \bar{x}_3 \bar{y}_1 y_2.$$

По этой системе могут быть построены либо два ГП (рис. 4.140), либо один (рис. 4.141). В данном случае единый ГП является более читаемым.

С другой стороны, одному и тому же графу переходов единого автомата могут соответствовать различные совокупности графов переходов.

Пусть, например, задана СБФ вида

$$y_1' = x_1 \bar{y}_1 \vee \bar{x}_2 y_1; \qquad y_2' = (x_1 \bar{y}_1 \vee \bar{x}_2 y_1) \bar{y}_2 \vee \bar{x}_3 y_2.$$

Этой системе соответствует единый ГП (рис. 4.142).

Этой же СБФ соответствует совокупность из двух ГП (рис. 4.143), реализуемых параллельно (двумя схемами или двумя процессорами) или псевдопараллельно (одним процессором).

Термин «псевдопараллелизм» понимается в том смысле, что, несмотря на то что структура СБФ предполагает параллельные вычисления всех формул системы при одних и тех же исходных данных, при соответствующей организации вычислительного процесса последовательное вычисление формул обеспечивает получение правильных результатов.

Отказ от псевдопараллелизма возможен, если СБФ представить в следующем виде:

$$y_1' = x_1 \bar{y}_1 \vee \bar{x}_2 y_1; \qquad y_2' = y_1' \bar{y}_2 \vee \bar{x}_3 y_2.$$

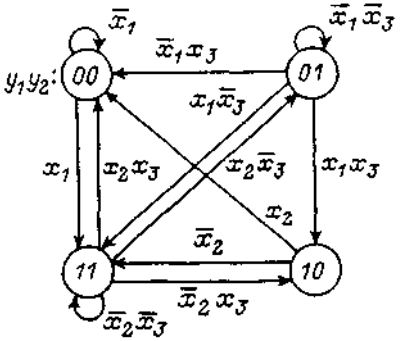

Рис. 4.142

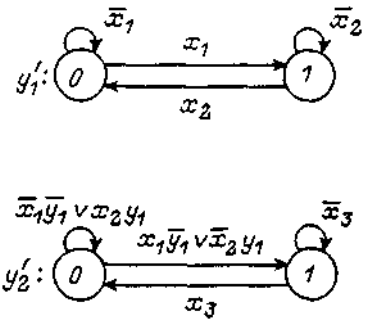

Рис. 4.143

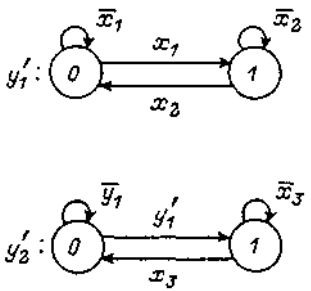

Рис. 4.144

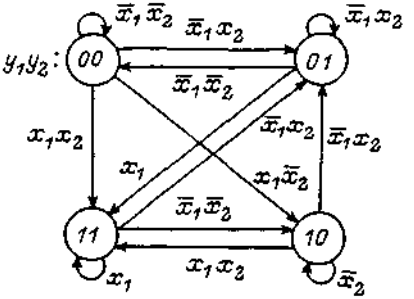

Рис. 4.145

Этой системе соответствует другая совокупность ГП (рис. 4.144). Структурная простота этих ГП создает иллюзию простоты их совместного поведения, которая исчезает при рассмотрении единого ГП (рис. 4.142).

Применение единого ГП (рис. 4.145) для анализа целесообразно даже и в тех случаях, когда графы переходов по переменным, кодирующим состояния, не связаны (рис. 4.146). В этом случае СБФ имеет вид:

$$y_1 = x_1 \vee \bar{x}_2y_1; \quad y_2 = x_2 \vee x_1y_2.$$

Если совокупность ГП не имеет общих переменных, то единый граф переходов может стать необозримым. В этом случае целесообразно задавать единый ГП в табличной форме.

Пусть, например, требуется реализовать два не связанных между собой триггера — *R*- и *S*-типов (рис. 4.147):

$$y_1 = \bar{x}_1(x_2 \vee y_1); \quad y_2 = x_3 \vee \bar{x}_4y_2.$$

Единый ГП, заданный матрицей переходов, приведен в табл. 4.23.

Таблица 4.23

y_1y_2	00	01	10	11
00	$(x_1 \vee \bar{x}_2)\,\bar{x}_3$	$(x_1 \vee \bar{x}_2)\,x_3$	$\bar{x}_1\,x_2\,\bar{x}_3$	$\bar{x}_1\,x_2\,x_3$
01	$(x_1 \vee \bar{x}_2)\,\bar{x}_3\,x_4$	$(x_1 \vee \bar{x}_2)$ & & $(x_3 \vee \bar{x}_4)$	$\bar{x}_1\,x_2\,\bar{x}_3\,x_4$	$\bar{x}_1\,x_2\,(x_3 \vee \bar{x}_4)$
10	$x_1\,\bar{x}_3$	$x_1\,x_3$	$\bar{x}_1\,\bar{x}_3$	$\bar{x}_1\,x_3$
11	$x_1\,\bar{x}_3\,x_4$	$x_1\,(x_3 \vee \bar{x}_4)$	$\bar{x}_1\,\bar{x}_3\,x_4$	$\bar{x}_1\,(x_3 \vee \bar{x}_4)$

Отметим, что для многих практически важных алгоритмов логического управления матрицы переходов содержат большое число нулевых элементов, что обеспечивает эффективную реализацию ГП программой-интерпретатором (Приложение 3), так как при этом в настроечном массиве индивидуально указываются лишь ненулевые элементы матрицы.

Из рассмотренных примеров следует, что уменьшение связности СБФ в общем случае приводит к увеличению связности единого ГП.

Усложнение единого ГП по сравнению с совокупностью из N не связанных между собой графов переходов, эквивалентной единому ГП, является платой за переход от возможной их параллельной реализации к последовательной. В совокупности ГП в каждый момент времени активизировано N вершин, в то время как в едином графе — только одна.

При этом отметим, что на первый взгляд кажется, что при переходе от N не связанных между собой графов переходов к одному графу переходов «надежность» алгоритма снижается, так как можно предположить, что, например, при непоявлении какой-либо переменной в первом случае в рамках предлагаемой технологии «не сработает» только один граф, а остальные графы могут нормально функционировать, а

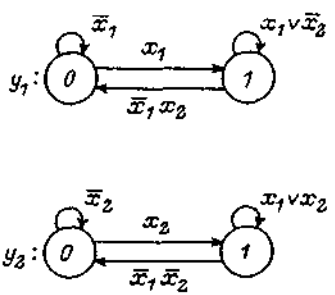

Рис. 4.146

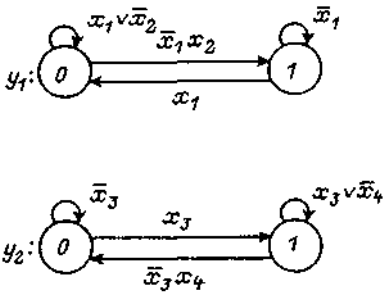

Рис. 4.147

во-втором — единый граф «застревает» в некотором состоянии. Однако последнее утверждение неверно при условии, что один граф функционально эквивалентен системе графов переходов.

Можно утверждать, что «надежность» эквивалентных алгоритмов не зависит от формы представления и языков описания.

4.4.9. Методика построения графа переходов управляющего автомата, реализуемого программно

1. Строится схема связей «источники информации—управляющий автомат—средства представления информации—исполнительные механизмы», являющаяся в общем случае схемой с обратными связями, как это принято в теории автоматического управления. Возможны варианты этой схемы, в которых средства представления информации или исполнительные механизмы объекта (объектов) управления не применяются.

Каждая информационная связь в схеме помечается переменной. При этом считается, что входные переменные управляющего автомата принадлежат множеству двоичных переменных *X,* а его выходные переменные — множеству двоичных переменных Z.

Схема может содержать также краткие комментарии, поясняющие смысл используемых переменных и свойства органов управления, исполнительных механизмов и собственно объекта управления.

2. Вводится понятие «состояние» объекта управления. Определяются и перечисляются (классифицируются) его состояния, что формирует пространство состояний объекта.

Состояния объекта управления могут быть как устойчивыми, так и неустойчивыми. Так как обычно объекты управления при автоматизации технологических процессов весьма инерционны по сравнению со временем программного цикла управляющего вычислительного устройства, то поэтому они не только в устойчивых, но и в неустойчивых состояниях могут находиться достаточно «долго». Например, для такого объекта, как клапан, устойчивыми являются такие его состояния, как «закрыт» и «открыт», а неустойчивыми — «открывается» и «закрывается».

Таким образом, в предположении об исправной работе клапана в графе переходов функционирования объекта (граф объекта) достаточно использовать только эти четыре состояния.

Отметим, что при переключении клапана его сигнализаторы не позволяют отличить последовательность устойчивых состояний «закрыт» —» «открыт» от последовательности «открыт» —> «закрыт». Неустойчивые состояния клапана «открывается» и «закрывается» позволяют различить эти последовательности.

Четыре состояния можно выделить также и в таком объекте управления, как лампа, два из которых устойчивы («не горит» и «горит»), а два — неустойчивы («загорается» и «гасится»).

При этом отметим, что если для клапана устойчивые состояния могут быть отделены от неустойчивых с помощью сигнализаторов, то для лампы в каждой паре состояний «горит»—«загорается» и «не горит»—«гасится»

устойчивое состояние от неустойчивого по внешнему проявлению неразличимы.

Неустойчивые состояния в этом случае введены для того, чтобы отличить последовательность устойчивых состояний «не горит» —> «горит» от последовательности «горит» —> «не горит».

Неустойчивым состояниям лампы в дальнейшем будут соответствовать устойчивые состояния управляющего ею автомата, которые сохраняются при единичном (неустойчивом) значении входной переменной, а устойчивым состояниям лампы — устойчивые состояния автомата, которые сохраняются при нулевом (устойчивом) значении той же входной переменной.

Если последовательность изменения устойчивых состояний лампы несущественна, то ее неустойчивые состояния могут не рассматриваться, а управляющий автомат становится комбинационным.

В случае, когда требуется учесть возможные неисправности объекта, пространство его состояний должно быть расширено.

При этом если выбор числа «нормальных» состояний является в некотором смысле объективным, то выбор числа дополнительных состояний субъективен и зависит от желания Разработчика алгоритма. С увеличением числа дополнительных состояний объекта «разрешающая» способность создаваемого алгоритма увеличивается. При этом чем больше нюансов Разработчик желает отразить в графе объекта, а в дальнейшем и в алгоритме управления, тем больше состояний должно быть выделено. Предполагая, например, что неисправность клапана может состоять либо в его неоткрытии, либо в незакрытии, в граф объекта может быть введено лишь одно дополнительное состояние «неисправность». Если же принято решение отличить неоткрытие от незакрытия, то вместо одного состояния должны использоваться два состояния, каждое из которых соответствует «своей» неисправности: «неоткрытие» и «незакрытие».

Таким образом, множество состояний объекта может состоять из двух подмножеств: «норма» и «неисправности».

3. Каждому состоянию объекта сопоставляется вершина в графе объекта. Вершины располагаются на плоскости и для идентификации нумеруются десятичными числами от 0 до s **- 1,** где s — число выделенных состояний объекта. При этом можно считать, что числа являются значениями одной многозначной переменной s.

4. Каждая вершина графа объекта через дробь с кодирующей ее цифрой помечается кортежем значений двоичных переменных, являющихся подмножеством множества X, которые формируются сигнализаторами объекта в этом состоянии. Пометка вершин из подмножества «норма» определяется свойствами объекта, а вершин из подмножества «неисправности» — решением Разработчика о том, какая информация характеризует соответствующее состояние. При этом различные вершины графа объекта могут быть помечены одинаковыми кортежами.

5. Определяются все допустимые переходы между состояниями объекта, что отражается введением соответствующих дуг в граф объекта. Дуги между вершинами из подмножества «норма» вводятся в соответствии со свойствами объекта, а дуги между вершинами из подмножеств «норма» и «неисправности» вводятся Разработчиком в зависимости от принятых

на втором и четвертом этапах методики решении об идентификации неисправностей и решении о том, что делать после их устранения. При этом для каждой вершины графа, соответствующей устойчивому состоянию объекта, вводится петля.

6. Каждая дуга и петля в графе объекта помечается конъюнкциями переменных или их инверсий из подмножества множества Z, которые соответствуют значениям переменных, подаваемых на входы исполнительных механизмов объекта и средств представления информации. В этом графе в неустойчивых вершинах исходящие дуги могут быть помечены одинаково.

На этом завершается построение графа объекта.

7. По графу объекта строится модель объекта, которую будем называть графом переходов функционирования модели (граф модели). В графе модели полностью сохраняется «скелет» графа объекта и пометки его вершин. В графе модели все вершины должны быть устойчивыми, и поэтому каждая его вершина должна иметь инцидентную ей петлю.

В каждую вершину к значениям переменных из множества X добавляются значения двоичных переменных из множества t, управляющих функциональными элементами задержки, которые моделируют времена пребывания объекта в неустойчивых состояниях. Количество этих переменных может быть сведено к одной, если можно сделать предположение о том, что все переходные процессы в объекте имеют одинаковую длительность.

Дуги графа модели, исходящие из вершин, соответствующих устойчивым вершинам графа объекта, помечаются прямыми или инверсными переменными из множества Z, входящими в состав конъюнкций, инициирующих соответствующие переходы в графе объекта.

Каждая дуга графа модели, исходящая из вершины, соответствующей неустойчивой вершине графа объекта, помечается конъюнкцией, состоящей из двоичной переменной из множества T, которая определяет факт срабатывания функционального элемента задержки, и из минимального числа двоичных переменных или их инверсий из подмножества множества X, характеризующих состояние объекта в смежной устойчивой вершине.

При необходимости осуществляется пометка петель вершин графа модели за счет обеспечения полноты переходов из каждой вершины.

8. Построение графа переходов автомата начинается с анализа технического задания с целью выявления необходимости использования функциональных элементов задержки в алгоритме управления.

Если такая необходимость отсутствует, то строится новая схема связей, отличающаяся от исходной заменой словосочетания «управляющий автомат» на слово «автомат».

В случае, когда в задании упоминаются временные задержки, строится новая схема связей, отличающаяся от исходной тем, что в ней управляющий автомат декомпозирован на автомат и ФЭЗ, число которых равно числу различных временных задержек, упоминаемых в задании. При этом для автомата вводятся два множества двоичных переменных t и T, описанных в предыдущем пункте, первое из которых для автомата является выходным, а второе — входным.

Из изложенного следует, что в рамках предлагаемой методики для автомата функциональные элементы задержки рассматриваются в качестве внешних устройств, аналогичных объекту управления, рассматриваемому совместно с его исполнительными механизмами и сигнализаторами.

Это позволяет при алгоритмизации процесса управления применять такую математическую модель (граф переходов), в которой время в явном виде не используется.

9. Каждому состоянию объекта сопоставляется состояние автомата, управляющего объектом. На плоскости изображаются s вершин графа переходов автомата.

В качестве комментария у каждой вершины может быть написано название соответствующего ему состояния объекта управления.

Так как в технологических процессах обычно используются весьма инерционные объекты управления, то как устойчивым, так и неустойчивым состояниям объекта в данном случае должны соответствовать только устойчивые состояния автомата управления. При этом каждому состоянию автомата должна соответствовать вершина графа переходов, которой инцидентна петля.

Так, например, автомат (счетный триггер), управляющий лампой, для которой выделены четыре состояния, также должен иметь четыре состояния, каждое из которых устойчиво.

10. Для каждой вершины графа переходов автомата (граф автомата) определяется кортеж значений всех его выходных переменных, образующих множество Z, который обеспечивает пребывание объекта в состоянии (устойчивом или неустойчивом), соответствующем рассматриваемому состоянию автомата.

Найденные кортежи значений выходных переменных используются для пометки вершин графа автомата.

Если рассматриваемый автомат входит в состав управляющего автомата, то в кортежи, помечающие вершины, вводятся также значения всех двоичных переменных из множества t, управляющих функциональными элементами задержки.

11. Вершины графа автомата соединяются дугами в соответствии с соединением вершин в графе объекта.

12. Каждая дуга графа автомата помечается булевой формулой, составленной из переменных множества X и (или) множества T и равенство единице которой инициирует соответствующий переход в автомате.

13. В граф автомата могут быть введены также и дополнительные дуги, отсутствующие в графе объекта. Например, в граф автомата может быть введена дополнительная дуга для устранения возможного несоответствия начальных состояний автомата и объекта.

14. Если дополнительные дуги вводятся в граф автомата, то каждая из них должна быть помечена булевой формулой, равенство единице которой инициирует переход между вершинами, соединенными этой дугой. Например, несоответствие между начальными состояниями автомата и объекта может быть устранено в результате перехода автомата по введенной дуге при равенстве единице булевой формулы, зависящей от минимально возможного числа переменных, характеризующих исходное состояние объекта.

15. Для каждой вершины графа автомата ортогонализацией или расстановкой приоритетов обеспечивается непротиворечивость переходов в другие вершины.

16. При необходимости осуществляется пометка петель вершин графа автомата за счет обеспечения полноты переходов из каждой вершины.

17. Построенный граф автомата анализируется на наличие генерирующих контуров, отличных от петель, которые устраняются при их обнаружении.

18. Если в построенном графе автомата имеются вершины, помеченные одинаково, то для их различения должно использоваться кодирование.

Если в построенном графе автомата все вершины помечены различными кортежами выходных переменных, то эти кортежи могут непосредственно применяться в качестве кодов состояний автомата. Однако и в этом случае бывает целесообразно закодировать вершины графа переходов значениями переменной Y, принимающей значения от 0 до $(s-1)$, т. е. сохранить верхнюю пометку вершин графа объекта.

На этом построение графа автомата может быть завершено.

19. Наличие одинаковых кортежей значений выходных и временных переменных в различных вершинах графа автомата является необходимым (но недостаточным) условием для минимизации числа состояний в автомате и соответственно числа вершин в его графе.

В теории автоматов разработаны математические методы минимизации числа состояний автоматов (для автоматов Мили такой метод изложен в разд. 4.4.5). Однако эти методы не позволяют непосредственно работать с графами автоматов и являются даже для задач сравнительно небольшой размерности весьма трудоемкими, что резко затрудняет их практическое использование.

Поэтому автором предлагается при необходимости выполнять эвристическую минимизацию числа состояний автомата в графической форме. При этом требуется совместить между собой по крайней мере некоторые из вершин графа автомата, помеченные одинаковыми кортежами значений выходных и временных переменных.

В новом графе устраняются противоречивость, неполнота и генерирующие контуры, если они появляются. Откорректированный граф автомата анализируется на возможность его использования в качестве управляющего для рассматриваемого объекта. Если этот граф семантически корректен, то на этом построение графа автомата завершается.

При этом необходимо отметить, что различные варианты устранения противоречивости и неполноты переходов в автомате могут приводить к различным графам переходов, соответствующим различным автоматам.

Таким образом, можно утверждать, что могут существовать такие автоматы, число состояний в которых меньше, чем число состояний в управляемом ими объекте, так как различные состояния объекта могут «поддерживаться» одинаковыми кортежами значений выходных переменных, формируемыми в одном и том же состоянии автомата.

Проведение минимизации числа состояний автоматов бывает необходимо только при жестких ограничениях на объем памяти и быстродействие и нецелесообразно при отсутствии таких ограничений.

Более того, при минимизации числа состояний структура графа автомата начинает «существенно» отличаться от структуры графа объекта, что, например, крайне неудобно, если требуется выполнять сигнализацию состояний объекта не от его сигнализаторов, а используя построенный автомат.

Отметим, что минимизация числа состояний автомата может проводиться не только совмещением вершин его графа переходов, но и, например, за счет построения на его основе головного и вызываемого графов, последний из которых реализует однотипные фрагменты заданного графа (гл. 8).

20. Выше рассматривались состояния автомата, которые прямо (для автоматов без минимизации числа состояний) **или обобщенно** (для автоматов с минимизированным числом состояний) **были связаны с состояниями исправного или неисправного объекта управления.**

Однако автоматы могут содержать также и такие состояния, которые с состояниями управляемого объекта не связаны, а их введение может определяться, например, необходимостью отражения в графе переходов неправильных действий Оператора. Учет таких состояний требует корректировки схемы связей за счет введения в общем случае дополнительных органов управления, средств представления информации и ФЭЗ.

Построенная схема связей и граф переходов автомата, входящего в эту схему, в исчерпывающей, однозначной и компактной форме определяют техническое задание на разработку программного обеспечения. При этом словесное описание может использоваться только в качестве комментария, так как полностью даже простой граф переходов словами описать весьма сложно и громоздко.

21. Изложенная методика отвечает на вопрос: откуда «берутся» состояния в управляющем автомате?

Из ее рассмотрения следует, что если понятие «состояние» не применять, то при алгоритмизации возникают весьма существенные трудности, как это имеет место, например, при построении граф-схем для алгоритмов с памятью, которые строятся обычно не в результате анализа состояний автомата, а за счет проверки значений его отдельных переменных. Метод построения граф-схем алгоритмов, использующих состояния, изложен в гл. 13.

22. В заключение раздела отметим, что предложенная методика позволяет строить графы переходов для автоматов без выходного преобразователя или для автоматов Мура, которые наиболее целесообразно использовать при логическом управлении такими технологическими объектами, как например клапаны, насосы, вентиляторы, а также их совокупности, рассматриваемые в качестве единого объекта управления.

Вопросы построения графов переходов для других классов автоматов (например, автоматов Мили, которые в одном и том же состоянии могут «поддерживать» различные состояния объекта управления за счет формирования различных значений выходных переменных) и систем графов переходов рассмотрены в других разделах книги.

Развернутый пример, иллюстрирующий все пункты предлагаемой методики, приведен в Приложении 12 [302].

Предложенная методика, на первый взгляд, кажется весьма традиционной для логического проектирования, однако многие введенные автором аспекты, например многозначное кодирование состояний, отличает ее от известных методик [304], что позволяет при ее использовании решать задачи программного логического управления достаточно большой размерности.