Глава 16

Программная реализация управляющих автоматов, заданных системами булевых формул

16.1. Построение графов переходов для анализа поведения автоматов с памятью, заданных СБФ

Пусть задана СБФ, определяющая работу автомата, которая должна быть программно реализована. Особенности программной реализации системы булевых формул состоят в следующем:

— входные переменные изменяются в начале или в конце программного цикла;

— выходные и внутренние переменные считываются в конце программного цикла;

— вычисленные значения выходных и внутренних переменных подставляются в правую часть тех формул, в которых эти переменные встречаются;

— все промежуточные значения СБФ фильтруются, т. е. считываются только те значения выходных и внутренних переменных, которые сформировались после вычисления последней формулы системы.

Из вышеизложенного следует, что в отличие от асинхронных схем граф переходов можно строить не непосредственно по заданной СБФ, изоморфной асинхронной схеме, получая кроме конечных еще и все промежуточные значения выходных и внутренних переменных системы [336], а менее трудоемко — по новой СБФ, получаемой из заданной системы после всех возможных подстановок и переобозначений переменных, записываемых в левой части формул новой системы.

Пример 16.1. Пусть задана СБФ

$$y = x \,\&\, (y \vee z); \quad z = x \,\&\, \bar{y} \tag{16.1}$$

и требуется построить граф переходов программно реализованного автомата, заданного этой системой.

Предположим, что $y = 0$, $z = 1$. Тогда при $x = 0$ после вычисления первой формулы система сохраняет предыдущие значения: $y = 0$, $z = 1$. После вычисления второй формулы с учетом найденного значения первой формулы получим: $y = 0$, $z = 0$. При тех же исходных значениях y и z при $x = 1$ после вычисления первой формулы имеем: $y = 1$, $z = 1$, а после вычисления второй формулы — $y = 1$, $z = 1$.

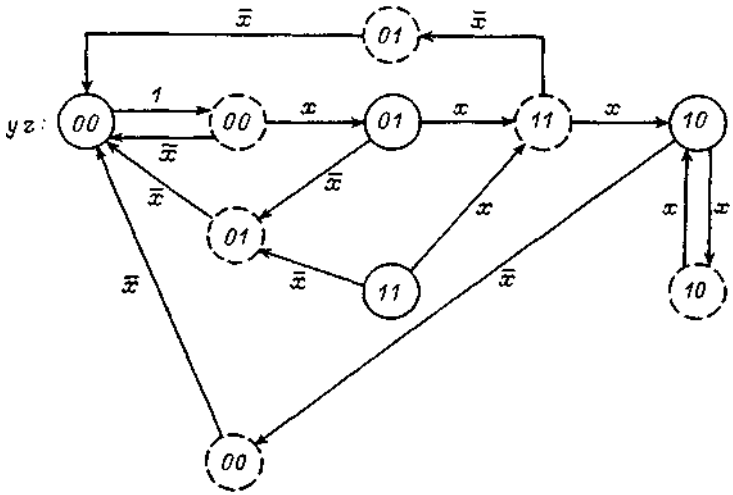

Рис. 16.1

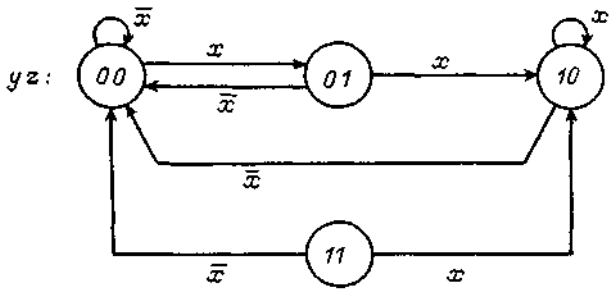

Рис. 16.2

Так как считывание результатов происходит только после вычисления второй формулы, можно считать, что результат вычисления первой формулы фильтруется. Обозначим «фильтруемые» вершины графа пунктиром.

Рассматривая в качестве исходных вершины: $y = 0$, $z = 0$; $y = 1$, $z = 0$; $y = 1$, $z = 1$ — и выполняя вычисления, аналогичные приведенным выше для вершины $y = 0$, $z = 1$, построим граф переходов (рис. 16.1). Если фильтруемые вершины исключить, то полученный ГП преобразуется в новый (рис. 16.2). Таким образом, можно утверждать, что СБФ (16.1) обеспечивает внешнее поведение, описываемое графом переходов на рис. 16.2.

Необходимо отметить, что при программной реализации в ходе переходного процесса смена фильтруемых значений происходит, в отличие от асинхронных схем, в строго фиксированном порядке: сначала устанавливается значение первой компоненты кода, задаваемой первой формулой системы, затем — второй, задаваемой второй формулой системы, и т. д.

Например, переход «11—00» при правильно организованной программе происходит за один программный цикл только следующим образом: «11—01—00».

Этот же граф переходов можно получить непосредственно (без построения графа с фильтруемыми вершинами), если выполнить подстановку первой формулы СБФ во вторую:

$$y = x \,\&\, (y \vee z); \quad z = x \,\&\, \overline{(x \,\&\, (y \vee z))} = x \,\&\, \bar{y} \,\&\, \bar{z}. \qquad (16.2)$$

Если по этой СБФ построить граф переходов, то он совпадает с ГП на рис. 16.2. Из анализа этого графа следует, что заданная система реализует по переднему фронту входной переменной x потенциальный сигнал z длительностью в один программный цикл.

Для программной реализации СБФ (16.2) должна быть преобразована:

$$y_1 = x \,\&\, (y \vee z); \quad z_1 = x \,\&\, \bar{y} \,\&\, \bar{z}; \quad y = y_1; \quad z = z_1. \qquad (16.3)$$

Эта система может быть упрощена одним из следующих способов:

$$y_1 = x \,\&\, (y \vee z); \quad z = x \,\&\, \bar{y} \,\&\, \bar{z}; \quad y = y_1; \qquad (16.4)$$

$$z_1 = x \,\&\, \bar{y} \,\&\, \bar{z}; \quad y = x \,\&\, (y \vee z); \quad z = z_1. \qquad (16.5)$$

При использовании соотношений (16.3)—(16.5) подстановки в течение программного цикла невозможны. Интересным является тот факт, что если вместо соотношений (16.4) записать соотношения

$$y = x \,\&\, (y \vee z); \quad z = x \,\&\, \bar{y} \,\&\, \bar{z} \qquad (16.6)$$

и выполнить подстановку:

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

то, так как вторая формула не изменяется, в этом случае можно применять выражения (16.6), а не (16.4).

Соотношения (16.5) упростить не удается, так как СБФ

$$z = x \,\&\, \bar{y} \,\&\, \bar{z}; \quad y = x \,\&\, (y \vee z) \qquad (16.7)$$

порождает граф переходов, отличный от представленного на рис. 16.2.

Из изложенного следует, что организация вычислений с переобозначениями (соотношения (16.3)—(16.5)) обеспечивает за один программный цикл реализацию не более одного перехода в графе переходов для устойчивых вершин (вершины с петлями) и один переход для неустойчивых вершин (вершины без петель).

Из приведенного примера следует, что в отличие от асинхронных схем при программной реализации состязания между переменными, кодирующими вершины графа переходов, отсутствуют, так как порядок вычисления фиксирован и заранее известен. Поэтому можно утверждать, что СБФ и функциональная или лестничная схемы, построенные по этой СБФ и реализованные программно (при фиксированной дисциплине опроса входных переменных в начале программного цикла), не могут вести себя недетерминированно, а в зависимости от их организации правильно или неправильно реализуют заданный алгоритм.

16.2. Методы построения СБФ по графам переходов для программной реализации автоматов с памятью

16.2.1. Использование принудительного кодирования

Пример 16.2. Пусть требуется построить автомат, управляющий трехпозиционным исполнительным механизмом с памятью от двух кнопок без памяти (x_1 — закрытие и x_2 — открытие) (рис. 16.3), поведение которого описывается графом переходов на рис. 16.4.

В данном случае применяется принудительное кодирование состояний автомата, так как коды вершин графа переходов различны. При этом нет необходимости использовать для различения вершин графа (состояний автомата) внутренние переменные.

Запись условий установки и сохранения единичных значений переменных

Аналитический метод использования полных кодов. Запишем СБФ по заданному графу переходов, применяя полные коды. Каждая формула представляет собой дизъюнкцию условий, при которых переменная z_i принимает и сохраняет значение, равное единице:

$$z_1 = x_1 \,\&\, \bar{z}_1 \,\&\, z_2 \vee \bar{x}_2 \,\&\, z_1 \,\&\, \bar{z}_2 = (x_1 \,\&\, z_1 \vee \bar{x}_2 \,\&\, z_1) \,\&\, \bar{z}_2;$$
$$z_2 = \bar{x}_1 \,\&\, x_2 \,\&\, \bar{z}_1 \,\&\, \bar{z}_2 \vee \bar{x}_1 \,\&\, \bar{z}_1 \,\&\, z_2 = (x_2 \vee z_2) \,\&\, \bar{x}_1 \,\&\, \bar{z}_1. \qquad (16.8)$$

Этой системе соответствует ГП на рис. 16.5.

Интересно отметить, что, несмотря на то что в исходном графе переходов (рис. 16.4) используются только соседние коды, применять СБФ (16.8) для программной реализации нельзя, так как эта система в результате подстановки при $x_1 = 0$ и $x_2 = 1$ порождает переход «10—01», отсутствующий в исходном графе, а переход «10—00» происходит не при $x_2 = 1$, как в исходном графе, а только при $x_1 = x_2 = 1$.

Необходимо отметить также, что если исходный граф содержит число вершин s меньше чем 2^m, где m — число выходных переменных, то в графе переходов, построенном по полученной СБФ, появляется $(2^m - s)$ вершин, из которых исходят дуги в нулевую вершину. Так как в дополнительные вершины дуги не входят, можно утверждать, что СБФ реализует заданный граф переходов (хотя из изложенного следует, что эта система делает несколько больше, чем задано).

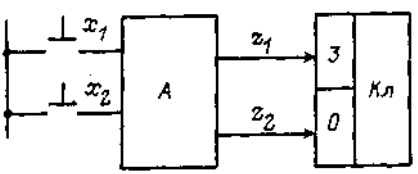

Рис. 16.3

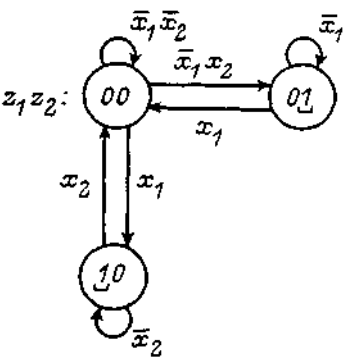

Рис. 16.4

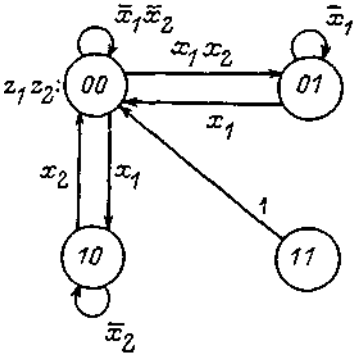

Рис. 16.5

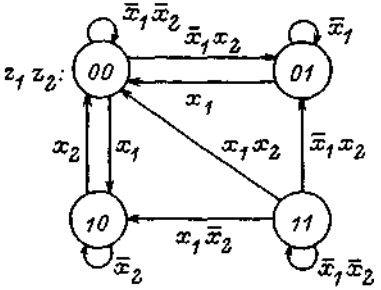

Рис. 16.6

Программная реализация на основе соотношений (16.8) требует использования переобозначения:

$$z_{11} = (x_1 \,\&\, \bar{z}_1 \vee \bar{x}_2 \,\&\, z_1) \,\&\, \bar{z}_2; \quad z_2 = (x_2 \vee z_2) \,\&\, \bar{x}_1 \,\&\, \bar{z}_1; \quad z_1 = z_{11}. \quad (16.9)$$

Число букв в правых частях этих формул равно 10.

Аналитический метод использования неполных кодов. Так как вершина «01» в графе переходов (рис. 16.4) отличается от остальных вершин наличием $z_2 = 1$, а вершина «10» — наличием $z_1 = 1$ (эти значения отмечены в вершинах графа), то этот граф может быть реализован следующим образом:

$$z_{11} = x_1 \,\&\, \bar{z}_1 \,\&\, \bar{z}_2 \vee \bar{x}_2 \,\&\, z_1;$$

$$z_2 = \bar{x}_1 \,\&\, x_2 \,\&\, \bar{z}_1 \,\&\, \bar{z}_2 \vee \bar{x}_1 \,\&\, z_2 = \bar{x}_1 \,\&\, (x_2 \,\&\, \bar{z}_1 \vee z_2); \quad (16.10)$$

$$z_1 = z_{11}.$$

Интересно отметить, что, несмотря на то что в этом случае применяются более компактные коды (сокращенное кодирование), число букв по сравнению с предыдущим случаем не изменяется. СБФ (16.10) соответствует графу переходов на рис. 16.6.

Табличный метод построения систем булевых формул. Заполним кодированную таблицу переходов (табл. 16.1) на основе заданного графа переходов (рис. 16.4), размещая прочерки в столбцах значений на позициях, соответствующих отсутствующей в графе вершине «11».

Используя карты Карно, доопределим эту таблицу, проставляя вместо прочерков следующие значения переменных z_1 и z_2 — 1 и 1; 0 и 1; 1 и 0; 0 и 0 соответственно. СБФ, построенная на основе этой таблицы, совпадает с соотношениями (16.10).

Таким образом, использование трудоемкого табличного метода, применение которого целесообразно лишь для задач небольшой размерности,

Таблица 16.1

z_1	z_2	x_1	x_2	z_{11}	z_2	z_1	z_2	x_1	x_2	z_{11}	z_2
0	0	0	0	0	0	1	0	0	0	1	0
0	0	0	1	0	1	1	0	0	1	0	0
0	0	1	0	1	0	1	0	1	0	1	0
0	0	1	1	1	0	1	0	1	1	0	0
0	1	0	0	0	1	1	1	0	0	−1	−1
0	1	0	1	0	1	1	1	0	1	−0	−1
0	1	1	0	0	0	1	1	1	0	−1	−0
0	1	1	1	0	0	1	1	1	1	−0	−0

не упрощает в данном случае систем булевых формул, полученных другими методами.

16.2.2. Запись условий изменений значений переменных и их сохранения

Запишем по графу переходов на рис. 16.4 булевы формулы, соответствующие условиям установки и сброса каждой выходной переменной, используя метод синтеза автоматов с распределенной памятью [126]:

$$S_1 = x_1; \quad R_1 = x_2; \quad S_2 = \bar{x}_1 \,\&\, x_2; \quad R_2 = x_1.$$

Так как одна и та же переменная в графе переходов не может одновременно устанавливаться и сбрасываться, то для ее сохранения может применяться любой из типов двухвходовых триггеров, например R-триггеры:

$$z_1 = \overline{R_1 \,\&\, (S_1 \vee z_1)}; \quad z_2 = \overline{R_2 \,\&\, (S_2 \vee z_2)}. \tag{16.11}$$

Выполняя подстановку, получим:

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

Эта СБФ обладает важным свойством: подстановка одной формулы в другую не может быть выполнена, и поэтому переобозначать переменные не требуется.

Однако, к сожалению, эта система реализует не граф переходов на рис. 16.4, а другой граф (рис. 16.7), частично совпадающий с заданным. Порочность использованного подхода состоит в том, что при записи условий изменений значений не учитывались значения переменных,

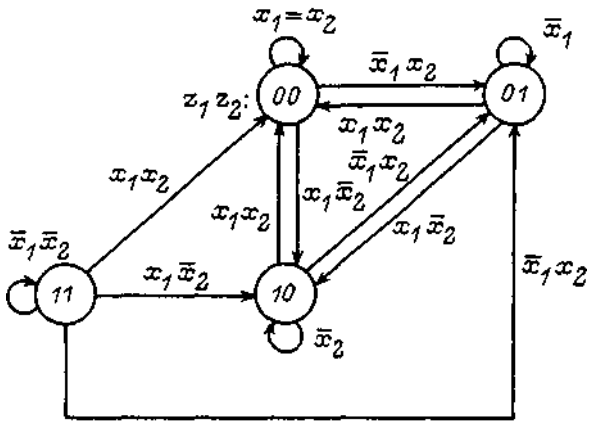

Рис. 16.7

определяющих предыдущие состояния. Устраняя этот недостаток, получим:

$$S_1 = x_1 \,\&\, \bar{z}_1 \,\&\, \bar{z}_2; \quad R_1 = x_2 \,\&\, z_1 \,\&\, \bar{z}_2; \quad S_2 = \bar{x}_1 \,\&\, x_2 \,\&\, \bar{z}_1 \,\&\, \bar{z}_2;$$

$$R_2 = x_1 \,\&\, \bar{z}_1 \,\&\, z_2; \quad z_1 = \overline{R}_1 \,\&\, (S_1 \vee z_1); \quad z_2 = \overline{R}_2 \,\&\, (S_2 \vee z_2).$$

При программной реализации этой СБФ переобозначения переменных выполнять не требуется. Осуществим в последней системе все возможные подстановки и минимизацию формул, а также переобозначение переменной z_1, так как в противном случае в новой СБФ будет иметь место подстановка z_1 в z_2:

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

Этой СБФ соответствует граф переходов на рис. 16.8, удовлетворяющий заданным требованиям. Эта система является наиболее сложной из рассмотренных.

При кодировании состояний неполными кодами комбинационная часть приобретает вид:

$$S_1 = x_1 \,\&\, \bar{z}_1 \,\&\, \bar{z}_2; \quad R_1 = x_2 \,\&\, z_1; \quad S_2 = \bar{x}_1 \,\&\, x_2 \,\&\, \bar{z}_1 \,\&\, \bar{z}_2; \quad R_2 = x_1 \,\&\, z_2.$$

Если выполнить подстановку этих соотношений в (16.11), а также переобозначение переменной z_1, то получим:

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

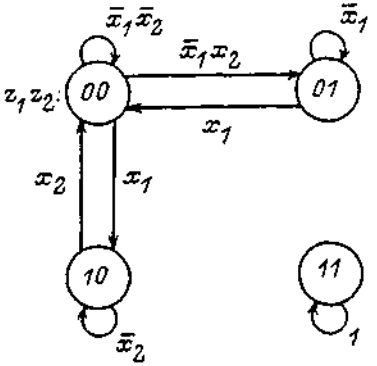

Рис. 16.8

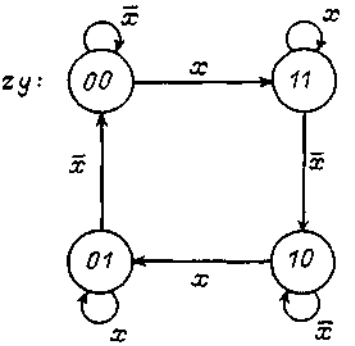

Рис. 16.9

Эта СБФ преобразуется к виду (16.10) и реализует граф переходов (рис. 16.6).

16.2.3. Использование принудительно-свободного кодирования

Условия установки и сохранения единичных значений переменных. Пусть задан граф переходов (рис. 13.21), описывающий работу счетного триггера. Так как этот граф содержит вершины, помеченные одинаково, введем дополнительную переменную «y», значения которой позволят различать вершины. При этом в отличие от асинхронных схем при программной реализации эти значения могут быть выбраны произвольно, т. е. коды необязательно должны быть соседними. В данном случае существует 16 различных вариантов назначения кодов, два из которых приведены на рис. 13.22 и 16.9. В силу того что выбор назначения, обеспечивающего простейшую реализацию, связан с большим перебором, выберем один из вариантов соседнего кодирования (рис. 13.22). Этот граф реализуется следующей СБФ:

$$z_1 = \bar{x}\ \&\ z \vee x\ \&\ \bar{y};\quad y = \bar{x}\ \&\ z \vee x\ \&\ y;\quad z = z_1. \qquad (16.12)$$

Из рассмотрения этой системы следует, что в данном случае требуется два типа внутренних переменных: y — для кодирования вершин графа, z_1 — для корректной реализации этого графа.

Если формулы этой СБФ реализуются на более низком уровне — бинарной программой, то другие внутренние переменные, кроме перечисленных, могут не применяться. Если реализация на этом уровне выполняется операторной программой, в том числе и использующей стек, то в общем случае проявляются внутренние переменные третьего типа, предназначенные для запоминания промежуточных значений формул. Для рассматриваемого примера для вычисления каждой формулы требуется только одна переменная третьего типа, а следовательно, она может не применяться, так как для этих целей в первой формуле уже имеется внутренняя переменная z_1, а во второй — переменная y.

В отдельных случаях переобозначения переменных можно не производить. Проверка возможности использования СБФ без переобозначений может быть выполнена способом, состоящим в том, что осуществляются все возможные подстановки в этой системе без выполнения переобозначений. Если при этом получающаяся система функций (не формул) отличается от исходной, то переобозначения должны производиться. Если отличия отсутствуют, то переобозначения можно не проводить. В случае, когда необходимо определить, чем отличается поведение, порождаемое новой системой, от поведения, порождаемого исходной системой, требуется построить граф переходов по новой СБФ и сравнить его с исходным.

Исходя из изложенного, можно утверждать, что соотношения (16.12) могут быть упрощены:

$$z = \bar{x}\,\&\,z \vee x\,\&\,\bar{y}; \quad y = \bar{x}\,\&\,z \vee x\,\&\,y, \tag{16.13}$$

так как

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

Запись условий изменений значений переменных и их сохранение. Покажем, что в этом случае может быть предложена стандартная реализация, состоящая из следующих пяти блоков, соединенных последовательно:

— формирователь условий переходов, при которых в графе переходов существуют изменения;

— дешифратор состояний, осуществляющий преобразование «полных» двоичных кодов в «единичные». При этом число конъюнкций, образующих дешифратор, равно числу вершин в графе переходов;

— дешифратор изменений, объединяющий конъюнкциями соответствующие выходы дешифратора состояний и формирователя условий переходов. При этом число конъюнкций равно числу дуг (без петель) в графе переходов;

— формирователь изменений, образованный дизъюнкциями переменных, получаемых от дешифратора изменений;

— блок R-триггеров.

Запишем СБФ, соответствующую стандартной структуре, реализующей граф переходов на рис. 16.9:

1. $F_0 = x; \quad F_1 = \bar{x}; \quad F_2 = x; \quad F_3 = \bar{x};$
2. $Y_0 = \bar{z}\,\&\,\bar{y}; \quad Y_1 = \bar{z}\,\&\,y; \quad Y_2 = z\,\&\,\bar{y}; \quad Y_3 = z\,\&\,y;$
3. $D_0 = F_0\,\&\,Y_0; \quad D_1 = F_1\,\&\,Y_1; \quad D_2 = F_2\,\&\,Y_2; \quad D_3 = F_3\,\&\,Y_3;$
4. $R_1 = D_2; \quad S_1 = D_0; \quad R_2 = D_1 \vee D_3; \quad S_2 = D_0 \vee D_2;$
5. $z = \bar{R}_1\,\&\,(S_1 \vee z); \quad y = \bar{R}_2\,\&\,(S_2 \vee y).$

При такой организации СБФ переобозначений переменных не требуется. Выполняя различные виды подстановок в этой системе, можно получить различные системы формул, каждой из которых соответствует своя реализация. Одна из таких СБФ имеет следующий вид:

$$R_1 = x\,\&\,z\,\&\,\bar{y}; \quad S_1 = x\,\&\,\bar{z}\,\&\,\bar{y}; \quad R_2 = \bar{x}\,\&\,y; \quad S_2 = x\,\&\,\bar{y};$$
$$z = \bar{R}_1\,\&\,(S_1 \vee z); \quad y = \bar{R}_2\,\&\,(S_2 \vee y).$$

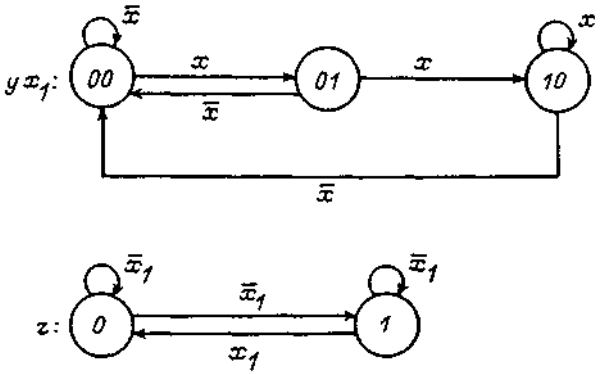

Рис. 16.10

Изложенный подход может применяться и для графов переходов с принудительным кодированием, в том числе и неполными кодами.

Использование композиции автоматов. Если входная переменная x является импульсной (сохраняется в течение только одного программного цикла), то счетный триггер может быть реализован по графу переходов (рис. 14.13). Для этого графа

$$z = x \,\&\, \bar{z} \vee \bar{x} \,\&\, z = x \oplus z. \qquad (16.14)$$

Из изложенного следует, что композиция на рис. 16.10, образованная на базе ГП на рис. 16.2 и 14.13, реализует счетный триггер для потенциальной переменной x. Из соотношений (16.1) и (16.14) следует, что простейшая реализация потенциального счетного триггера имеет вид:

$$y = x \,\&\, (y \vee x_1); \quad x_1 = x \,\&\, \bar{y}; \quad z = x_1 \oplus z.$$

16.2.4. Использование двоичного кодирования в автоматах Мура

Вместо принудительно-свободного кодирования может применяться двоичное кодирование, позволяющее за счет увеличения числа внутренних переменных (числа формул) упростить сами формулы. При этом для графа переходов с s вершинами для кодирования вершин требуется использовать s внутренних переменных Y_j, принимающих значение «единица», когда граф «находится» в вершине j, и «ноль», когда граф в этой вершине «не находится». На рис. 13.30 приведен ГП автомата Мура счетного триггера при двоичном кодировании вершин.

Кодирование каждой вершины графа переходов одной двоичной переменной. Граф переходов (рис. 13.30) в этом случае реализуется следующей СБФ:

$$\begin{aligned} & Y_0 = 1; \\ \text{M:}\ & Y_{01} = Y_0 \,\&\, \bar{x} \vee Y_3 \,\&\, \bar{x} = (Y_0 \vee Y_3) \,\&\, \bar{x}; \\ & Y_{11} = Y_0 \,\&\, x \vee Y_1 \,\&\, x = (Y_0 \vee Y_1) \,\&\, x; \end{aligned}$$

$$Y_{21} = Y_1 \,\&\, \bar{x} \vee Y_2 \,\&\, \bar{x} = (Y_1 \vee Y_2) \,\&\, \bar{x};$$
$$Y_3 = Y_2 \,\&\, x \vee Y_3 \,\&\, x = (Y_2 \vee Y_3) \,\&\, x;$$
$$z = Y_{11} \vee Y_{21};$$
$$Y_0 = Y_{01};\ Y_1 = Y_{11};\ Y_2 = Y_{21};$$
goto M.

Кодирование каждой вершины графа переходов одним *R*-триггером. В этом случае граф переходов (рис. 13.30) реализуется следующей **СБФ:**

$$Y_0 = 1;$$
$$\text{M: } Y_{01} = (\bar{x} \,\&\, Y_3 \vee Y_0) \,\&\, \bar{Y}_1;$$
$$Y_{11} = (x \,\&\, Y_0 \vee Y_1) \,\&\, \bar{Y}_2;$$
$$Y_{21} = (\bar{x} \,\&\, Y_1 \vee Y_2) \,\&\, \bar{Y}_3;$$
$$Y_3 = (x \,\&\, Y_2 \vee Y_3) \,\&\, \bar{Y}_0;$$
$$z = Y_{11} \vee Y_{21};$$
$$Y_0 = Y_{01};\ Y_1 = Y_{11};\ Y_2 = Y_{21};$$
goto M.

16.2.5. Реализация автоматов Мили

На рис. 4.113 приведен ГП автомата Мили, реализующий последовательностный сумматор, построенный по кодированной таблице переходов и выходов (табл. 4.12).

СБФ, программно реализующая этот автомат, имеет вид:

$$P_1 = x_1 \,\&\, x_2 \,\&\, \bar{P} \vee (x_1 \oplus x_2) \,\&\, P \vee x_1 \,\&\, x_2 \,\&\, P =$$
$$= (x_1 \vee x_2) \,\&\, P \vee x_1 \,\&\, x_2 = x_1 \,\#\, x_2 \,\#\, x_3; \qquad (16.15)$$
$$S = (x_1 \oplus x_2) \,\&\, \bar{P} \vee \bar{x}_1 \,\&\, \bar{x}_2 \,\&\, P \vee x_1 \,\&\, x_2 \,\&\, P = x_1 \oplus x_2 \oplus P;$$
$$P = P_1.$$

16.2.6. Использование неклассических моделей автоматов

Так как в формулу «переноса» переменная S не входит, то СБФ (16.15) можно упростить:

$$S = x_1 \oplus x_2 \oplus P; \qquad P = (x_1 \vee x_2) \,\&\, P \vee x_1 \,\&\, x_2. \qquad (16.16)$$

Определим более однородную реализацию. Для этого применим соотношение [247]

$$y = (\bar{x}_i \,\&\, y\,(x_i = 0) \vee x_i \,\&\, \bar{y}\,(x_i = 1)) \oplus x_i$$

к функции P, полагая, что $x_1 = x_2$. При этом

$$P = (\bar{x}_1 \,\&\, x_2 \,\&\, \bar{P} \vee x_1 \,\&\, \bar{x}_2 \,\&\, P) \oplus x_2 = Q \oplus x_2.$$

Определим, может ли формула Q быть представлена в виде

$$Q = (x_1 \oplus x_2) \,\&\, \Psi(x_2, P).$$

Решим это уравнение относительно Ψ табличным методом (табл. 16.2).

Таблица 16.2

x_1	x_2	P	Q	$x_1 \oplus x_2$	Ψ	
0	0	0	0	0	—	0
0	0	1	0	0	—	1
0	1	0	1	1	1	
0	1	1	0	1	0	
1	0	0	0	1	0	
1	0	1	1	1	1	
1	1	0	0	0	—	1
1	1	1	0	0	—	0

В табл. 16.2 символом «прочерк» обозначены значения Ψ, которые могут быть выбраны произвольно, а рядом с этим символом приведены выбранные значения функции. Из этой таблицы следует, что $\Psi = x_2 \oplus P$ и поэтому

$$S = x_1 \oplus x_2 \oplus P;\ P = (x_1 \oplus x_2) \,\&\, (x_2 \oplus P) \oplus x_2. \qquad (16.17)$$

Применяя гарвардский метод (разд. 4.3.5) к табл. 4.12, запишем P_1, как не полностью определенную функцию от переменных P, x_1, x_2, S.

Используя карту Карно, доопределим эту функцию так, чтобы она зависела только от переменных P, x_1, S:

$$P_1 = x_1 \,\#\, P \,\#\, \bar{S}.$$

Учитывая, что в СБФ эта формула вычисляется последней, новая система может быть записана в виде:

$$S = x_1 \oplus x_2 \oplus P; \quad P = x_1 \,\#\, P \,\#\, \bar{S}. \qquad (16.18)$$

Вновь применяя тот же метод к табл. 4.12, запишем S как не полностью определенную функцию от переменных P, x_1, x_2, P_1.

Применяя карту Карно, доопределим в этом случае функцию так, чтобы получающаяся формула содержала минимальное число букв:

$$S = (x_1 \vee x_2 \vee P) \,\&\, P_1 \vee x_1 \,\&\, x_2 \,\&\, P.$$

Так как теперь формула P_1 вычисляется первой, то новая система должна быть записана в виде:

$$P_1 = x_1 \,\#\, x_2 \,\#\, P; \quad S = (x_1 \vee x_2 \vee P) \,\&\, \bar{P}_1 \vee x_1 \,\&\, x_2 \,\&\, P;$$
$$P = P_1. \qquad (16.19)$$

При этом отметим, что не существует соотношения $s = f(p_1, x_1, x_2)$ и, следовательно, заданный алгоритм не может быть реализован автоматом Мили второго рода.

16.3. Реализация управляющих автоматов

Пусть требуется управлять трехпозиционным исполнительным механизмом с памятью от двух кнопок, причем сброс сигналов с него осуществляется через время T. На рис. 16.11 приведена схема связей «УА—ОУ». Декомпозируем управляющий автомат на автомат и ФЭЗ (рис. 16.12). Предположим, что в этом случае поведение автомата описывается графом переходов на рис. 16.13. Применяя в этом случае сокращенное кодирование, получим следующую программу:

$$\begin{aligned} z_{11} &= x_1 \,\&\, \bar{z}_1 \,\&\, \bar{z}_2 \vee \bar{T} \,\&\, z_1; \\ z_2 &= \bar{z}_1 \,\&\, x_2 \,\&\, \bar{z}_1 \,\&\, \bar{z}_2 \vee \bar{T} \,\&\, z_2; \\ t &= z_{11} \vee z_2; \\ T &= f(t); \\ z_1 &= z_{11}. \end{aligned} \tag{16.20}$$

Соотношение $T = f(t)$ требует специального программирования.

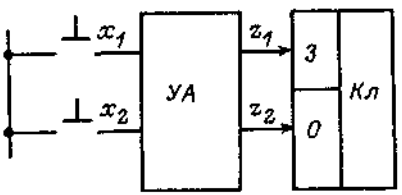

Рис. 16.11

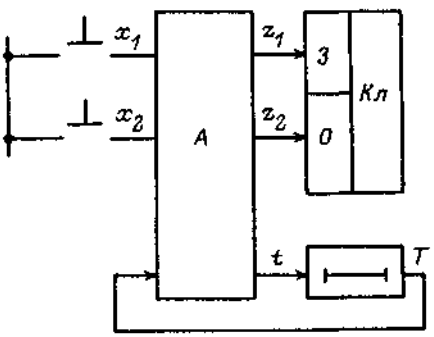

Рис. 16.12

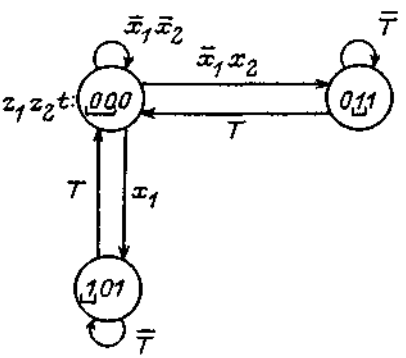

Рис. 16.13

16.4. Две трактовки автоматов с памятью, заданных СБФ

В первой из них считается, что автомату соответствует 2^s комбинационных схем, каждая из которых зависит только от входных переменных. Здесь s — число состояний автомата. При этом в каждом состоянии функционирует только одна КС, задающая состояния, в которые может перейти автомат из рассматриваемого состояния при смене значений входных переменных, номенклатура которых определяется этим состоянием. При такой трактовке автомат с памятью рассматривается как многофункциональный модуль, настраиваемый состояниями на реализацию различных комбинационных схем.

В этой трактовке автомату с памятью соответствует разложение Шеннона по переменным, кодирующим состояния. Это разложение для СБФ (16.2), порождающей четыре комбинационные схемы, имеет вид:

$$y_1 = 0 \,\&\, \bar{y} \,\&\, \bar{z} \vee x \,\&\, \bar{y} \,\&\, z \vee x \,\&\, y \,\&\, \bar{z} \vee x \,\&\, y \,\&\, z;$$
$$z_1 = x \,\&\, \bar{y} \,\&\, \bar{z} \vee 0 \,\&\, \bar{y} \,\&\, z \vee 0 \,\&\, y \,\&\, \bar{z} \vee 0 \,\&\, y \,\&\, z.$$

При этом для заданного автомата можно рассматривать только s таких схем, так как остальные соответствуют состояниям, которые не имеют переходов из заданных состояний.

При второй трактовке неавтономный автомат с памятью рассматривается как2"автономных автоматов. Здесь n — число входных переменных. При этом автомат с памятью можно рассматривать как многофункциональный модуль, настраиваемый значениями входных переменных на реализацию различных автономных автоматов. Так как за время программного цикла входной набор измениться не может, то можно утверждать, что в течение этого времени функционирует только один автономный автомат.

Граф переходов такого автомата, настраиваемого некоторым входным набором, задает все переходы неавтономного автомата, происходящие в графе переходов последнего при этом входном наборе. Этой трактовке автомату с памятью соответствует разложение Шенноиа по входным переменным. Это разложение для СБФ (16.2) имеет вид:

$$y_1 = 0 \,\&\, \bar{x} \vee (y \vee z) \,\&\, x;$$
$$z_1 = 0 \,\&\, \bar{x} \vee \bar{y} \,\&\, \bar{z} \,\&\, x.$$

Из этого соотношения следует, что в данном случае имеются два автономных автомата, первый из которых задает все переходы при $x = 0$, а второй — при $x = 1$.

При построении схем и ГСА первый вариант трактовки определяет использование дешифратора состояний, а второй — дешифратора значений входных переменных (событий).

Построение конструкций с дешифратором состояний может производиться по графу переходов непосредственно, в то время как переход к конструкциям с дешифратором значений входных переменных существенно более трудоемок, особенно учитывая тот факт, что обычно $s \ll 2^n$.

Любые промежуточные варианты между этими двумя вариантами трактовки могут уменьшить объем памяти программ при одновременном снижении их читаемости.