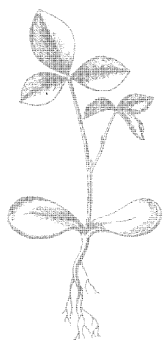
С. АНГЕР

АСИНХРОННЫЕ ПОСЛЕДОВАТЕЛЬНОСТНЫЕ СХЕМЫ

С. АНГЕР • АСИНХРОННЫЕ ПОСЛЕДОВАТЕЛЬНОСТНЫЕ СХЕМЫ

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ТЕХНИЧЕСКОЙ КИБЕРНЕТИКИ

ИЗДАТЕЛЬСТВО «НАУКА»
ГЛАВНАЯ РЕДАКЦИЯ
ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ
МОСКВА 1977



С. АНГЕР

АСИНХРОННЫЕ ПОСЛЕДОВАТЕЛЬНОСТНЫЕ СХЕМЫ

ПЕРЕВОД С АНГЛИЙСКОГО
А. С. БЕРНШТЕЙНА и Е. К. КОРНОУШЕНКО

ПОД РЕДАКЦИЕЙ
П. П. ПАРХОМЕНКО

ИЗДАТЕЛЬСТВО «НАУКА»
ГЛАВНАЯ РЕДАКЦИЯ
ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ
МОСКВА 1977

6Ф6.5
А 65
УДК 62-50

ASYNCHRONOUS SEQUENTIAL SWITCHING CIRCUITS

STEPHEN H. UNGER

Department of Electrical Engineering
Columbia University

WILEY — INTERSCIENCE A DIVISION
OF JOHN WILEY & SONS
NEW YORK LONDON SYDNEY TORONTO

[1969]

Асинхронные последовательностные схемы. Ангер С. Перев. с англ., серия «Теоретические основы технической кибернетики», Главная редакция физико-математической литературы издательства «Наука», М., 1977, 400 стр.

Излагаются основные понятия и свойства асинхронных дискретных устройств. Рассматриваются вопросы минимизации числа состояний, различные способы кодирования состояний, виды и влияние задержек, различные виды состязаний внутренних сигналов, способы предотвращения состязаний при разработке схем, вопросы минимизации числа контуров с обратной связью, числа усилителей в них, некоторые специальные виды схем. Свойства асинхронных схем и методы их проектирования иллюстрируются на примерах разнообразных счетчиков и регистров, используемых в качестве узлов ЦВМ.

Табл. 118, илл. 92, библ. 129

Стефен Ангер

АСИНХРОННЫЕ ПОСЛЕДОВАТЕЛЬНОСТНЫЕ СХЕМЫ

(Серия: «Теоретические основы технической кибернетики»)

М., 1977 г., 400 стр. с илл.

Редактор *Д. С. Фурманов*
Техн. редактор *Н. В. Кошелева.*
Корректоры *О. А. Бугусова, И. В. Хорошаева*

Сдано в набор 18/III 1977 г. Подписано к печати 1/IX 1977 г. Бумага 84×108¹/₃₂. Физ. печ. л. 12,5. Условн. печ. л. 21. Уч.-изд. л. 21,4. Тираж 4750 экз. Цена книги 2 р. 20 к. Заказ № 481
Издательство «Наука», Главная редакция физико-математической литературы, 117071, Москва, В-71, Ленинский проспект, 15

4-я типография издательства «Наука». 630077, Новосибирск, 77, Станиславского, 25.

А 30501—133 / 053(02)-77 467-77

ОГЛАВЛЕНИЕ

ОТ РЕДАКТОРА ПЕРЕВОДА

Вопросы создания последовательностных дискретных устройств, работающих устойчиво независимо от технологических и эксплуатационных разбросов временных задержек составляющих их элементов и компонент, относятся к числу наиболее сложных в теории и практике проектирования переключательных схем.

Предлагаемая читателю монография Стефена Р. Ангера «Асинхронные последовательностные схемы» является полным, глубоким и систематическим изложением задач синтеза асинхронных дискретных устройств с памятью, в том числе задач анализа и исключения состязаний сигналов в таких устройствах.

Несмотря на то, что со времени издания оригинала монографии прошло 7 лет, ее полезность для широкого круга специалистов в области дискретной техники не вызывает сомнений.

П. П. Пархоменко

ПРЕДИСЛОВИЕ АВТОРА

Теория переключательных схем охватывает вопросы анализа и синтеза таких систем, существенные переменные которых принимают конечное число значений. К подобным системам относятся цифровые вычислительные машины, телефонные переключательные системы и многие типы систем управления.

Задачи, рассматриваемые в теории переключательных схем, отличны от задач теории электрических цепей в том плане, что в первом случае, как правило, основные составные части системы уже заданы и являются довольно сложными схемами. Специалист по теории переключений, другими словами, инженер-логик, имеет дело не с напряжениями, токами, сопротивлениями, каким-либо видом энергии, а с идеализированными описаниями составных частей системы, внешними сигналами которых являются обычно единицы и нули. С другой стороны, теория переключательных схем отличается от теории построения цифровых систем в том плане, что она обладает весьма обширным и точным множеством понятий, применимых к решению лишь определенных четко очерченных проблем определенного уровня сложности. Разработка большой системы является скорее интуитивным процессом, связанным с разработкой и объединением многих подсистем, тогда как каждая из таких подсистем может быть четко описана инженером-логиком. Излишне говорить, что в реальной жизни границы не являются столь резкими и различия между инженером-логиком, инженером-электриком и разработчиком систем часто расплываются и пропадают. Одна из целей теории переключательных схем — осветить подробнее смежные вопросы.

Существенной частью теории переключательных схем является теория комбинационных схем, т. е. изучение переключательных схем без памяти. При введении свойства запоминания информации мы получаем последова-

тельностные схемы. Как подкласс может быть выделена теория синхронных переключательных схем — изучение последовательностных схем, в которых время, так же как и амплитуды сигналов, предполагается квантованным. Предметом настоящей книги является теория асинхронных схем, в которой не требуется, чтобы события происходили лишь на дискретных интервалах.

Естественно, эти определения приведены лишь для того, чтобы дать некоторое интуитивное представление о рассматриваемых ниже понятиях. Предполагается, что читатель знаком с теорией комбинационных переключательных схем и использованием карт Карно. Качественное представление о работе электрических схем, конечно, способствует пониманию ряда обсуждаемых моментов, но этого недостаточно. Для читателя без указанной подготовки останутся неясными довольно много вопросов из этой области. От читателя не требуется особой математической подготовки, хотя желательная подготовка и отличается от той, которая необходима при изучении лишь комбинационных схем, однако предполагается наличие у него определенного уровня «математической зрелости», поскольку излагаемый материал — математический по характеру и целый ряд цепочек рассуждений должен быть досконально прослежен читателем.

При изложении материала была сделана попытка минимизировать формализм и рассматривать читателя как человека, а не как автомат для проверки доказательств. Мне кажется, что высокий уровень формализма не всегда вызывает озноб у читателя; тем не менее, многие этапы проводимых здесь доказательств представлены в неформальной, хотя, я надеюсь, в ясной и точной форме. В некоторых случаях я руководствовался принципом, что для выражения сути какой-либо важной идеи не следует прибегать к полной и сложной аргументации, а достаточно дать лишь четкую схему доказательства. Несомненна также польза приводимых примеров для уяснения введенных понятий и процедур.

Цель настоящей книги состоит в том, чтобы дать ясное, единое и достаточно полное представление о предмете, которое позволит читателю применить на практике содержащиеся здесь идеи, либо продолжить исследования и дополнить обрисованную здесь картину. В первом слу-

чае не следует ожидать, что инженер найдет здесь удачное решение всех его проблем, но он по крайней мере получит представление о предмете, которое поможет наметить границы его возможностей и применить общие методы к его конкретным случаям.

Настоящая книга «Асинхронные последовательностные схемы» может быть использована как учебник или как пособие для самообразования. Каждая глава заканчивается перечислением ряда задач; решения примерно трети из них, помеченных «*» и «+», приведены в конце книги. Задачи, помеченные звездочкой, затрагивают важные идеи, о которых не говорится в тексте, и требуют специального рассмотрения.

Весь материал книги мог бы составить односеместровый аспирантский курс, причем достаточно трех часов занятий в неделю даже в случае, если бы преподаватель захотел подробно проработать некоторые темы. Я читал подобный курс на электротехническом отделении Колумбийского университета, но моему курсу предшествовал курс по теории переключательных схем.

Читателю рекомендуется изучать книгу последовательно, опуская материал, не вызывающий у него интереса, и возвращаясь к нему, когда он найдет это необходимым для понимания последующего раздела книги. Поскольку вводная глава является обзором тематики остальных глав, желательно перечитывать соответствующую часть Введения перед началом каждой новой главы.

Чтобы сноски не вносили беспорядка в текст, все материалы о первоисточниках идей каждой главы я поместил в заключительный раздел, озаглавленный «Замечания по библиографии». Цитирование в этих разделах так же, как и в других местах книги, производится согласно отдельному алфавитному библиографическому списку в конце книги. Этому списку предшествует краткое замечание об источниках, не упомянутых в тексте книги.

Еще в 1953 году все опубликованные работы по теории переключательных схем, будучи собранными вместе, составили бы довольно скромный том. Совершенно иное положение сейчас, даже в той из областей теории, которая здесь рассматривается. Пришлось произвести отбор материала для включения в книгу; этот отбор вызовет, конечно, противоречивые мнения, особенно по второсте-

пенным вопросам. Мне кажется, тем не менее, что узловые моменты изложены здесь полно.

В главе 1 сделана попытка обрисовать читателю круг рассматриваемых проблем. Глава 2 содержит современное изложение проблемы минимизации состояний для синхронных и асинхронных функций. В главе 3 рассматривается проблема кодирования строк таблиц переходов для асинхронных схем в плане минимизации числа состояний, длительности переходов и сложности схем. Предметом главы 4 являются задержки, состязания и временны́е соотношения в схемах при различных предположениях относительно паразитных задержек. Глава 5 посвящена обратным связям, показана их роль в асинхронных схемах; здесь рассматривается также вопрос о синтезе схем с минимальным числом обратных связей. В главе 6 исследуются различные типы операций, включая процедуру синтеза схем, генерирующих сигналы завершения и называемых иногда *схемами, не зависящими от скорости*, проблемы синхронизации и нечувствительности в импульсных или синхронных схемах, а также синтез двухранговых последовательностных схем. И, наконец, в главе 7 рассматривается целый ряд различного рода двоичных счетчиков в свете тех или иных концепций, высказанных в предыдущих главах.

Я благодарен Фонду Саймона Гуггенхейма за выделенную мне стипендию, которая сделала возможной написание настоящей книги, а также факультету электротехники и вычислительной техники Калифорнийского университета в Беркли, где я работал в качестве стипендиата, за дружескую атмосферу в процессе работы над книгой. Выражаю особенную благодарность докторам Армстронгу, Фридману и Менону из Белл Телефон Лаборатори за ознакомление меня с их еще не опубликованными результатами, что позволило включить материал, не появляющийся обычно в книгах на такой ранней стадии. На ряд ошибок и туманных мест в книге мне было указано моими студентами. И, наконец, следует отметить мастерскую и терпеливую работу, проделанную мисс Бетти Джинти из Колумбийского университета и миссис Билли Вртяк из Беркли, которые перепечатали рукопись.

Нью-Йорк, сентябрь 1969 *Стефен Г. Ангер*

ГЛАВА 1
ВВЕДЕНИЕ

Чтобы ясно представить существо задач, рассматриваемых в настоящей книге, и ввести необходимую терминологию, удобно начать с конкретного довольно правдоподобного примера. Используемые при этом понятия развиваются и варьируются в последующих главах.

1.1. Формальное представление задачи — таблица переходов

Рассмотрим довольно загруженную автостраду, пересекаемую сельской дорогой, по которой иногда проезжает случайный автомобиль. Если на переезде отсутствует светофор, автомобиль, движущийся по дороге, вряд ли благополучно пересечет автостраду в силу наличия на последней интенсивного высокоскоростного движения. Светофор, останавливающий движение по автостраде через каждую минуту или две безотносительно к тому, что в девяти из десяти случаев у переезда нет ожидающих автомобилей, мог бы быть установлен лишь в том случае, если бы отдельные фермеры объединились в своем требовании и их голосов оказалось достаточно, чтобы навязать такое неудобство гораздо большему количеству пассажиров.

Предположим, что этого не произошло, и неизвестна примерная стоимость остановки движения по автостраде. Тогда естественный и широко распространенный компромисс состоит в том, что устанавливается светофор, останавливающий движение по автостраде лишь тогда, когда у переезда появится ожидающий автомобиль, так что движение автомобилей по автостраде осуществляется при зеленом свете такого светофора и прекращается при включении красного сигнала, обусловленном наличием ожидающего автомобиля. Подобное включение может быть осуществлено схемой управления, получающей сигналы от

реле времени и от параллельно соединенных датчиков давления, вмонтированных в полотно сельской дороги в определенных местах.

Пусть схема управления должна функционировать следующим образом.

Временной сигнал x_1 отсутствует ($x_1 = 0$) в течение 60 сек и появляется ($x_1 = 1$) на 30 сек. Красный сигнал светофора ($z = 1$) на автостраде может включаться лишь на интервале, на котором $x_1 = 1$. Включившись в начале этого интервала, он должен оставаться включенным на протяжении всего интервала. Срабатывание датчиков давления от автомобилей обозначается в виде условия $x_2 = 1$ (отсутствию автомобилей соответствует условие $x_2 = 0$); z становится равным единице на очередном интервале включения x_1 *).

Требование, чтобы «красный» интервал был в фазе с фиксированным временны́м интервалом, делает возможной синхронизацию работы светофора на автостраде, желательную с точки зрения управления скоростью движения по автостраде. Задача упростилась бы, если бы этого не требовалось.

Итак, нам необходимо построить схему с двумя двоичными (т. е. двузначными) входами x_1 и x_2 и одним двоичным выходом z. Существенным при этом является то, что входные сигналы взаимно независимы. Нельзя, например, исключить возможность одновременного изменения x_1 и x_2.

Прежде всего, ясно, что никакая комбинационная схема не обладает требуемым поведением. Выход комбинационной схемы является функцией лишь текущих значений входных переменных, тогда как в нашем случае должны учитываться и прошлые значения входов.

Например, при $x_1 = 1$ и $x_2 = 0$ имеем $z = 1$, если $x_2 = 135$ секунд назад, и $z = 0$, если $x_2 = 0$ последние 120 секунд. Наша схема является, таким образом, последовательной схемой в том смысле, что она должна запоминать информацию о прошлых состояниях входов.

---

*) Заметим, что хотя ничего не говорилось о предупреждающем желтом свете, интервал включения желтого света может быть включен как начальная часть в 30-секундный «красный» интервал.

Подобная информация сохраняется в виде *внутренних состояний* схемы. В каждый момент времени схема находится в каком-либо состоянии из конечного их числа, определяемом совокупностью значений *внутренних переменных*. Внутренние состояния служат для учета предысторий поведения схемы. Комбинация внутреннего состояния и состояния входа называется *полным состоянием*.

Как выход последовательностной схемы, так и ее следующее внутреннее состояние являются функциями текущего полного состояния. Когда следующее внутреннее состояние то же, что и текущее внутреннее состояние, то полное состояние считается устойчивым в том смысле, что невозможно появление никаких изменений до тех пор, пока не изменится состояние входа.

Приведенные определения даны в схемной интерпретации. Эту же терминологию удобно принять и для поведенческого описания схемы, или, используя более подходящий термин, для *последовательностных функций*. Последовательностная функция связывает выходные последовательности с входными последовательностями в предположении известного начального внутреннего состояния. Приведенное выше описание функционирования светофора является примером неформального словесного описания последовательностной функции. Чтобы сделать такое описание более точным и более приемлемым для формальных преобразований, мы опишем функцию путем задания полных состояний, выходных функций и функций переходов некоторой схемы, реализующей желаемую последовательностную функцию. Из такого описания отнюдь не следует, что соответствующая ему схема полностью определена (она должна лишь реализовать исходную последовательностную функцию); просто подобное описание является удобной формой исходного формального задания. В последующем такое описание может быть преобразовано в более желательную форму.

Распространенный способ описания, который используется и здесь для указанных выше целей, связан с построением так называемой *таблицы переходов*, состоящей из строк и столбцов. Столбцы соответствуют состояниям входов, строки — текущим внутренним состояниям, элементами таблицы переходов являются упорядоченные пары, представляющие следующее внутреннее состояние и

текущий выходной сигнал. Элементы таблицы, соответствующие устойчивым состояниям, выделены жирным курсивом.

Исходное задание последовательностных функций представляется в форме таблиц переходов, имеющих точно одно устойчивое состояние в каждой строке. Такие таблицы называются *первичными таблицами переходов*. Покажем, как строится такая таблица, вернувшись к нашему примеру со светофором.

Рассмотрим вначале «статическую» ситуацию, соответствующую устойчивому состоянию. Пусть временной сигнал отсутствует ($x_1 = 0$), автомобилей на дороге нет, то есть датчики давления находятся в исходном состоянии ($x_2 = 0$), и в течение нескольких минут автомобилей по-прежнему нет. Тогда $z = 0$, и внутреннее состояние таково, что если значение $x_2 = 1$ не появится до появления значения $x_1 = 1$, то z должно оставаться равным нулю в течение очередного «красного» интервала. Обозначив внутреннее состояние через 1, можно записать полное состояние как 1—00 (первая цифра соответствует внутреннему состоянию, а две другие — состояниям входов x_1, x_2). Поскольку это состояние устойчиво, его нужно выделить (см. табл. 1.1а).

Таблица 1.1а

Частично заполненная таблица переходов, описывающая ситуацию на сельской дороге при отсутствии движения

	x_1x_2 00	01	11	10
1	***1***,0			2,0
2	1,0			***2***,0

Пусть теперь система находится в этом устойчивом состоянии и x_1 стал равным 1 (полное состояние теперь имеет вид 1—10), z по-прежнему остается равным нулю, следующее внутреннее состояние должно быть отличным от 1, поскольку в каждой строке первичной таблицы переходов должно быть только одно устойчивое состояние. Обозначим новое состояние цифрой 2. Таким образом,

полному состоянию 1—10 соответствует элемент (2,0) таблицы переходов. Рассмотрим теперь полное состояние 2 — 10. Внутреннее состояние 2 соответствует ситуации, в которой сигнал $x_2 = 1$ не появлялся продолжительное время, и, следовательно, z должен оставаться равным 0 по крайней мере во время текущего «красного» интервала. Таким образом, состояние 2—10 должно быть устойчивым с соответствующим элементом (2,0) в таблице.

Предположим теперь, что x_1 возвращается в 0, когда система в состоянии 2—10. Ситуация становится той же самой, что и в начале рассмотрения, следовательно, элементом таблицы переходов, соответствующим состоянию 2—00, будет (1,0). Фрагмент таблицы переходов, приведенный в табл. 1.1а, точно описывает смену ситуаций в устойчивом состоянии, когда временной сигнал x_1 последовательно принимает каждое из своих значений и движение по сельской дороге отсутствует. Полное состояние системы меняется циклически, проходя через состояния 1—00, 1—10, 2—10, 2—00, 1—00, ..., а z все время неизменно и равно 0.

Продолжим построение таблицы переходов, начав заполнять пустые места в строках 1 и 2 таблицы 1.1а; для

Таблица 1.1б

Результирующая таблица переходов, в которой не рассматриваются условия, связанные с одновременным изменением двух входных переменных

	00	01	11	10
1	***1***,0	3,0		2,0
2	1,0	3,0	7,0	***2***,0
3	4,0	***3***,0	6,1	
4	***4***,0	3,0		5,1
5	1,0	3,0	6,1	***5***,1
6		3,0	***6***,1	5,1
7		3,0	***7***,0	8,0
8	4,0	3,0	7,0	***8***,0

того чтобы этот процесс завершился (к счастью) построением полностью заполненной таблицы, оказывается, необходимо добавить новые строки и новые состояния.

Следующим шагом (см. табл. 1.1б) может быть рассмотрение полных состояний 1—01 и 3—01 и определе-

ние соответствующих им элементов (3,0) таблицы. Заметим, что после каждого изменения состояния входа система переходит из устойчивого состояния в неустойчивое в той же строке, а затем уже — в устойчивое состояние в том столбце, который соответствует новому состоянию входа. Такой вид поведения является характерным для первичных таблиц переходов, описывающих функции, в которых состояние выхода меняется не более одного раза при изменении состояния входа.

Когда система находится в полном состоянии 3—01 и x_2 меняется на 0, следующее устойчивое состояние должно находиться, конечно, в 00-столбце. Если бы этим состоянием было устойчивое состояние 1—00, то, очевидно, система «забыла» бы, что x_2 был включен. Поэтому введем новое устойчивое состояние 4—00, оно располагается в 00-столбце. Особенностью этого состояния является то, что z должен включиться в течение очередного «красного» интервала, на котором $x_1 = 1$; это отображено в элементах, соответствующих полным состояниям 4—10 и 5—10. Заметим здесь, что независимо от текущего состояния, последствия появления входа 01 (автомобиль вызвал срабатывание датчика давления, хотя $x_1 = 0$) должны быть одними и теми же. Следовательно, все элементы в 01-столбце должны быть равны (3,0).

Далее, поскольку z включен, он должен оставаться включенным, пока включен x_1, при этом в столбце 11 должно быть устойчивое состояние, в котором $z = 1$; значение $z = 1$ должно относиться как к этому состоянию, в котором $x_2 = 1$, так и к состоянию 5—10, в котором $x_2 = 0$ (такая попеременная смена значений x_2 может быть вызвана потоком машин на дороге, связанным, например, с предстоящей ярмаркой). Эти требования отражены в элементах, соответствующих состояниям 5—11, 6—11, 6—10. Состояние 6—11 является состоянием, в которое переходит система из 3—01 при включении x_1. Этому переходу соответствует элемент (6,1) в состоянии 3—11.

Возвращаясь к строке 2, отметим, что если x_2 включается при $x_1 = 1$, то сигнал z должен оставаться нулевым, но должен быть включен на очередном «красном» интервале. Это объясняет выбор элементов в 2—11 и 7—11. Если x_2 становится равным 0, когда система находится в

7—11, следующее состояние должно запомнить тот факт, что x_2 был равным 1. Так появляется переход в новое состояние 8—10. Элементы строки 8 заполняются аналогичным образом.

Когда система находится в 5—10, z включен и автомобилей на дороге нет ($x_2 = 0$). Следовательно, когда x_1 отключается, система может вернуться в устойчивое состояние 1—00 через элемент, соответствующий 5—00. Таким образом, построенная нами таблица переходов совпадает с таблицей 1.1б, где оставшиеся незаполненными места соответствуют одновременным изменениям значений x_1 и x_2.

Если одновременные изменения значений x_1 и x_2 происходят тогда, когда система была в одном из состояний 3—01, 4—00 или 7—11, то соответствующие элементы,

Таблица 1.1в

Полностью заполненная перзичная таблица переходов

	x_1x_2 00	01	11	10
1	***1***,0	3,0	6,1/7,0	2,0
2	1,0	3,0	7,0	***2***,0
3	4,0	***3***,0	6,1	5,1
4	***4***,0	3,0	6,1	5,1
5	1,0	3,0	6,1	***5***,1
6	1,0	3,0	***6***,1	5,1
7	4,0	3,0	***7***,0	8,0
8	4,0	3,0	7,0	***8***,0

относящиеся к состояниям 3—10, 4—11 или 7—00, должны быть такими, как показано в таблице 1.1в. Когда система находится в состоянии 6—11 и оба входа отключаются одновременно, целесообразнее всего предусмотреть переход системы в состояние 1—00, что соответствует элементу (1, 0) в состоянии 6—00. Если возникают некоторые опасения относительно того, что последний автомобиль проехал над датчиком давления, но еще не пересек автостраду, то заключительным состоянием следует взять 4—00 вместо 1—00.

Рассмотрим теперь случай, когда система находится в состоянии 1—00, а автомобиль, приближающийся к светофору, включает датчик x_2 в тот момент, когда x_1 меняет значение с 0 на 1. Одним из вариантов является немедленное включение z. Это сразу дало бы автомобилю зеленый свет, а в таблице переходов отобразилось бы помещением элемента (6,1) в клетке 1—11. При другом варианте автомобилю необходимо ждать начала следующего включения x_1, а система переходит в состояние 7. Выбор того или иного варианта несуществен, поскольку оба варианта приемлемы, а рассмотренная ситуация будет возникать редко. В этих случаях выбор варианта производится таким образом, чтобы получить более простую схему. Поскольку на первоначальном (и простейшем) этапе формулировки задачи еще не ясны преимущества того или иного варианта, наличие последних будет указываться в таблице переходов заданием всех приемлемых элементов так, как показано для состояния 1—11 таблицы 1.1в. Теперь, когда все клетки таблицы переходов заполнены, таблица 1.1в представляет полное формальное описание того, что мы хотим от схемы, управляющей поведением светофора. Слово «полное» должно быть, очевидно, дополнено замечанием, что в таблице переходов ничего не говорится о таких существенных характеристиках, как выбор временной шкалы и энергетических параметров схемы — таблица переходов описывает только желаемое логическое поведение схемы.

Перед переходом к следующему этапу построения решения для нашего примера следует сделать несколько общих замечаний о таблицах переходов, подобных рассмотренной выше. Прежде всего заметим, что при преобразовании естественного языкового описания или мысленного образа в точную форму невозможно применение какой-либо алгоритмической процедуры. Можно разработать лишь эвристические приемы и постоянно проверять, согласуется ли результирующее формальное описание с исходной неформальной идеей. Часто процесс формального описания событий порождает вопросы, позволяющие глубже заглянуть в суть описываемых явлений. Так, процесс построения таблицы переходов может сделать понятнее исходную задачу, если даже такая таблица и не будет далее применяться.

1.2. Минимизация таблицы переходов

Один из подходов к построению таблицы переходов, описываемый здесь, связан с определением тех устойчивых состояний, которые необходимы для каждого входного состояния — столбца таблицы. Начиная с этих устойчивых состояний, как некоторого базиса первичной таблицы переходов, находим переходы между ними и заполняем соответствующие клетки таблицы переходов. При этом может случиться, что выбранных устойчивых состояний недостаточно; тогда добавляем необходимые устойчивые состояния. Как станет понятным дальше, не будет особого ущерба, если будет использовано слишком много состояний. Дело в том, что следующим этапом процедур синтеза является нахождение таблицы переходов с минимальным числом состояний, которая удовлетворяет требованиям, заложенным в исходную первичную таблицу переходов.

Таблица 1.2

Минимизированная таблица переходов

	x_1x_2 00	01	11	10
1	***1***,0	2,0	4,0	***1***,0
2	***2***,0	***2***,0	3,1	3,1
3	1,0	2,0	***3***,1	***3***,1
4	2,0	2,0	***4***,0	***4***,0

Необходимость минимизации числа состояний и методы проведения такой минимизации составляют содержание главы 2. Пока достаточно лишь отметить, что в нашем примере при выборе элемента (7,0) в клетке 1—11 получается четыре пары одинаковых строк: 1 и 2, 3 и 4, 5 и 6, 7 и 8. Отсюда следует, что таблица 1.1в может быть минимизирована в таблицу 1.2 и не существует таблицы с меньшим числом строк, удовлетворяющей нашим требованиям.

1.3. Кодирование внутренних состояний

Построив таблицу переходов, которая удовлетворительно описывает желаемую последовательностную функцию, мы должны (следующий этап синтеза) выбрать конечное множество двоичных внутренних переменных и приписать каждой строке таблицы переходов одно или несколько состояний этих переменных. После этого

непосредственной задачей является построение таблиц истинности, определяющих значения выходов схемы и следующие значения внутренних переменных как функции текущих значений входов и внутренних переменных.

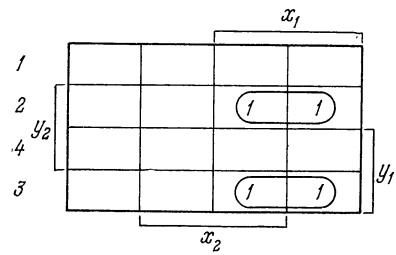
Для того чтобы однозначно закодировать каждую строку таблицы переходов, число внутренних переменных n должно удовлетворять условию $2^n \geqslant r$, где r — число строк таблицы. Удобно записать это условие в виде $n \geqslant \mathrm{E}[\log_2 r]$, где $\mathrm{E}[x]$ — наименьшее число, не меньшее x. (Величина $\mathrm{E}[\log_2 r]$ обозначается обычно как s_0.) Для таблицы переходов, представленной таблицей 1.2, в которой $r = 4$, необходимы по крайней мере две внутренние переменные. Предположим, что такое минимальное число переменных выбрано, переменные обозначены как y_1 и y_2 (договоримся i-ю внутреннюю переменную обозначать через y_i), тогда Y_i будет соответствовать следующему значению переменной y_i, и состояния переменных y_1y_2 (y-состояния) приписаны так, как показано в таблице 1.3, *а*. Таблицу переходов, в которой каждой строке приписано y-состояние, будем называть *матрицей переходов*.

Теперь можно построить карты Карно (K-карты) для z, Y_1 и Y_2 как функций от x_1, x_2, y_1 и y_2 (табл. 1.3, *б*, *в* и *г*). Столбцы карт Карно упорядочены таким же образом, как и столбцы матрицы переходов. Порядок строк другой, указанный слева в таблице 1.3, *б*. K-карта для z (z-карта) получается заполнением единицами клеток, соответствующих тем элементам матрицы переходов, в которых $z = 1$ (правые половины строк 2 и 3). Для получения Y_1-карты отметим по матрице переходов, что $y = 1$ имеют y-состояния, приписанные строкам 3 и 4. Следовательно, каждому полному состоянию, для которого следующим внутренним состоянием является 3 и 4, должна соответствовать единица в Y_1-карте. Каждое следующее внутреннее состояние в 11-столбце матрицы переходов есть 3 или 4, следовательно, все элементы в 11-столбце Y_1-карты (см. табл. 1.3, *в*) суть единицы. В 10-столбце только в строке 1 следующее внутреннее состояние отлично от состояний 3 и 4, следовательно, в Y_1-карте в столбце 10 будет незаполненное место только в строке 1. Заметив, аналогично, что Y_2 должно быть единицей всякий раз, когда следующим внутренним состоянием является 2 или 4, мы получим таблицу 1.3, *г*.

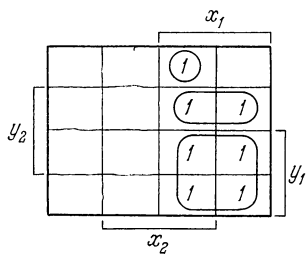
Таблица 1.3

	x_1x_2 00	01	11	10	y_1	y_2
1	***1***,0	2,0	4,0	***1***,0	0	0
2	***2***,0	***2***,0	3,1	3,1	0	1
3	1,0	2,0	***3***,1	***3***,1	1	0
4	2,0	2,0	***4***,0	***4***,0	1	1

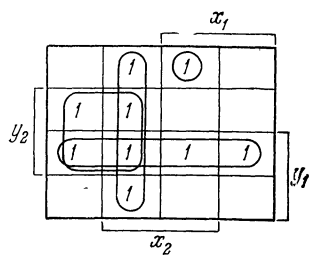
а) Матрица переходов



б) z-матрица



в) Y_1-матрица



г) Y_2-матрица

Применяя обычные методы описания комбинационных схем, по картам Карно получим следующие выражения:

$$z = x_1\bar{y}_1y_2 + x_1y_1\bar{y}_2, \tag{1.1}$$

$$Y_1 = x_1x_2\bar{y}_1\bar{y}_2 + x_1\bar{y}_1y_2 + x_1y_1, \tag{1.2}$$

$$Y_2 = \bar{x}_1x_2 + y_1y_2 + \bar{x}_1y_2 + x_1x_2\bar{y}_1\bar{y}_2. \tag{1.3}$$

Заметим, что выражения для Y_1 и z, а также для Y_1 и Y_2 содержат одинаковые произведения (члены).

Полученные выражения, в свою очередь, приводят к схеме на рис. 1.1, где функции z, Y_1 и Y_2 реализованы

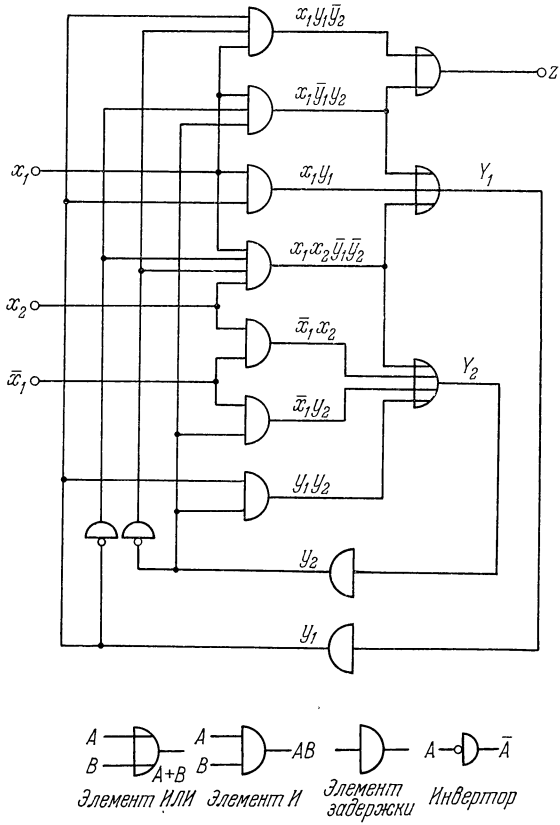


Рис. 1.1. Последовательностная схема, реализующая уравнения (1.1)—(1.3).

двухуровневыми логическими схемами. Предполагается, что в нашем распоряжении имеются как входные сигналы, так и их отрицания (парные входы), и для каждого i значение y_i получается из Y_i через элемент задержки.

Именно в этом состоит смысл понятия, что Y_i является следующим значением переменной y_i.

Можно многое сказать относительно элементов задержки, об их свойствах и необходимости их введения в схемы; пока нам достаточно лишь предположить, что с каждым элементом задержки связывается определенная внутренняя переменная и через них не проходят частые изменения состояний их входов *). Как станет понятным дальше, это очень консервативные предположения.

Для того чтобы глубже понять, как работает последовательностная схема, полезно рассмотреть в деталях некоторые из переходов. Прежде всего рассмотрим переход, когда система находится в состоянии 1—00 и x_2 меняется с 0 на 1.

В начальном полном состоянии $x_1 = x_2 = y_1 = y_2 = 0$ анализ схемы (или K-карты) показывает, что $Y_1 = Y_2 = 0$ для этого полного состояния; следовательно, поскольку значение входа каждого элемента задержки совпадает со значением его выхода, система действительно находится в устойчивом состоянии, что отображено в матрице. Теперь, если x_2 становится равным 1, что вызывает переход в состояние 1—01, Y_1 остается равным 0, но член $\bar{x}_1 x_2$ в выражении для Y_2 становится равным 1, и это приводит к изменению значения Y_2. Через время задержки благодаря этому изменению y_2 меняется на 1, так что следующим полным состоянием будет 2—01. При $x_1 = 0$, $x_2 = 1$, $y_1 = 0$, $y_2 = 1$ из структуры схемы видно, что $Y_1 = 0$ и $Y_2 = 1$, и, следовательно, система снова будет в устойчивом состоянии. Продолжая рассмотрение, убеждаемся, что наша схема работает в точном соответствии с таблицей переходов.

Пусть теперь система находится в состоянии 2—01 и x_1 принимает значение 1. Член $\bar{x}_1 y_1 y_2$ при этом равен 1,

*) Более точно: если такой элемент, называемый *элементом инерциальной задержки,* имеет задержку величины D, то после изменения входа элемента изменение его выхода происходит через время D. Таким образом, если, например, вход такого элемента длительное время был 1(0), потом стал равным 0(1) на интервале, меньшем D, после чего снова стал равным 1(0), то элемент «игнорирует» такие быстрые изменения входа и его выход все это время будет равен 1(0).

следовательно, Y_1 становится равным 1. С другой стороны, оба члена x_1x_2 и x_1y_2, которые возбуждали Y_2, теперь равны 0, благодаря чему Y_2 становится равным 0. Заметим, что поскольку $Y_1 \neq y_1$ и $Y_2 \neq y_2$, оба элемента задержки находятся в неустойчивых состояниях, причем сигнал 1 проходит через элемент задержки y_1, а сигнал 0 — через элемент задержки y_2. Предположим, что времена прохождения этих сигналов через соответствующие элементы задержки одинаковы. Тогда новое полное состояние будет 3—11; анализ схемы показывает, что это состояние устойчиво и поведение схемы должно совпадать с поведением, определяемым таблицей переходов. Но что произойдет, если y_1 изменится первым? Это случилось бы, если бы задержка y_1 была меньше задержки y_2. При таком изменении стало бы $x_1 = x_2 = y_1 = y_2$, член x_1y_1 возбудил бы Y_1, член y_1y_2 стал бы равным 1 и возвратил бы Y_2 снова в 1. Но тогда, в силу инерциальности, задержка y_2 вернулась бы в устойчивое состояние, «забыв» короткий период, в течение которого Y_2 было равно 0, и система стала бы устойчивой в состоянии 4—11! Такая возможность, естественно, не отражается в таблице переходов.

Ситуация может быть еще худшей, если обратить внимание на то (а читатель может это проверить), что, если в состоянии 2—01 y_2 меняется раньше, чем y_1, то система вначале придет в состояние 1—11 и лишь затем в 4—11. Получается, что система действительно будет работать должным образом лишь тогда, когда обе задержки в точности равны.

Детальное рассмотрение работы схемы и даже проведение процедуры синтеза после построения таблицы переходов не являются обязательными для определения условий неоднозначности переходов. Простая проверка таблицы 1.3, *а* показывает, что в течение перехода 2—11 → → 3—11 значения двух внутренних переменных y_1 и y_2 должны измениться с 01 на 10. Если предполагается, что y_1 изменяется первым, то y-состояние становится равным 11, а комбинация 11 приписана внутреннему состоянию 4, которое устойчиво в 11-столбце. Следовательно, существует возможность перехода системы в состояние 4—11. Если же y_2 изменяется первым, то система переходит в 1—11, где следующим внутренним состоянием

является состояние 4, так что существует вторая возможность неправильного функционирования схемы.

Ситуация, в которой в процессе перехода возможно изменение более чем одной внутренней переменной, называется *условием состязания*. Если правильное поведение схемы зависит от результата состязания, последнее называется *критическим состязанием*. Если же поведение системы совпадает с поведением, задаваемым таблицей переходов независимо от результата состязания, то такое состязание называется *некритическим*. Например, переход 3—01→2—01 в таблице 1.3, *а* содержит условие состязания. Однако независимо от того, какая переменная y_1 или y_2 изменится первой, происходит переход в состояние 2 с выработкой указанного в таблице значения выхода (значения z будут правильными во всех возможных промежуточных состояниях).

Пример критического состязания показан в таблице 1.4, *а*, которую следует рассматривать, как часть некоторой матрицы переходов. Переход из состояния 4 в состояние 7 содержит состязание между y_1 и y_2, которое

Таблица 1.4

	y_1y_2	y_3
4	7,0	0
5	7,0	1
6	**6**,1	1
7	**7**,0	1

а) Критическое состязание

	y_1y_2	y_3
4	5,0	1
5	7,0	1
6	**6**,1	1
7	**7**,0	1

б) Устранение состязания

является критическим, поскольку, если y_1 меняется первым, то система перейдет в состояние 6, а если y_2 изменится первым, то система совершит правильный переход в состояние 7. Это происходит потому, что промежуточное состояние, которому приписано y-состояние 011, также имеет состояние 7 в качестве следующего внутреннего состояния. Следовательно, можно «зафиксировать» это состязание, задав задержку элемента, связанного с y_1, значительно большей, нежели задержку, связанную с y_2. Такое решение имеет два недостатка:

1. Могут быть другие аналогичные состязания в матрице переходов, которые требуют, чтобы y_1 менялся быстрее, чем y_2.

2. Если взглянуть глубже, весьма нежелательно задавать более одного значения для y-задержек, поскольку введение второго значения для величины задержки потребует введения более «тонких» ограничений, рассмотрение которых потребует дополнительных затрат, не говоря уже о том, что подобное усложнение требований само по себе неудобно с точки зрения проектирования и анализа схем.

Имеется, однако, второй метод предотвращения рассматриваемых критических состязаний. Поскольку правильное поведение схемы имеет место, если переход $4 \rightarrow 7$ происходит через промежуточное состояние 5 (другими словами, имеет место переход $4 \rightarrow 5 \rightarrow 7$), мы можем так изменить матрицу переходов, что указанный переход будет всегда происходить именно таким образом. Это сделано в таблице 1.4, *б*, где следующее внутреннее состояние в строке 4 заменено на 5. Теперь переход $4 \rightarrow 7$ происходит в два этапа, каждый из которых не содержит критических состязаний (в этом случае вообще нет состязаний). Недостатком является лишь то, что переход стал длиннее, поскольку переменные y_1 и y_2 меняют теперь свои значения последовательно.

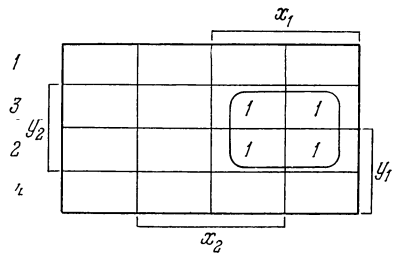
При наличии возможности такого предотвращения критические состязания типа, рассмотренного по таблице 1.4, *а*, будем называть *устранимыми критическими состязаниями* в противоположность таким, которые были рассмотрены по табл. 1.3, *а* и являются неустранимыми.

Из приведенных примеров должно быть понятно, что задача кодирования состояний является далеко не простой. Мало приписать строкам свои кодовые комбинации — последние должны быть так взаимосвязаны, чтобы никакой переход не содержал неустранимых состязаний. Кроме этого, желательно еще минимизировать время переходов (число этапов в каждом из переходов), число внутренних переменных и сложность результирующей логической схемы. В различных ситуациях основное значение приобретает тот или другой из указанных факторов, возможно также появление и других факторов, которые становятся основными. Различные подходы к этой сложной

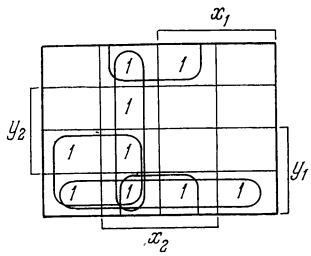
Таблица 1.5

	x_1x_2 00	01	11	10	y_1	y_2
1	***1***,0	2,0	4,0	1,0	0	0
2	***2***,0	***2***,0	3,1	3,1	1	1
3	1,0	2,0	***3***,1	***3***,1	0	1
4	2,0	2,0	***4***,0	***4***,0	1	0

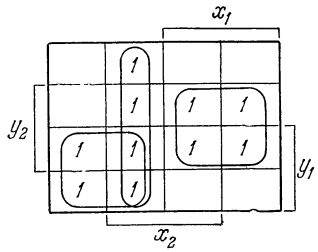
а) Пригодное кодирование строк



б) z-карта



в) Y_1-карта



г) Y_2-карта

проблеме рассматриваются в главе 3. Пока мы рассмотрим лишь решение для описанного выше примера.

В таблице 1.5, *а* показано другое (отличное от приведенного ранее в таблице 1,3, *а*) кодирование строк таблицы 1.2, при котором каждой из строк приписано однозначно свое y-состояние и которое свободно от критиче-

ских состязаний (некритические состязания имеются в 01-столбце).

Интересно отметить, что если в таблице 1.2 изменить содержимое клетки для полного состояния 2—10 на (1,0), то не существует способа кодирования, содержащего две внутренние переменные, такого, что результирующая таблица будет свободной от критических состязаний. Используя результаты главы 3, можно найти приемлемый способ кодирования, использующий три внутренних переменных.

1.4. Влияние паразитных задержек — комбинационные состязания

После того как мы закодировали по-новому строки таблицы переходов, построим K-карты (табл. 1,5, *б*, *в* и *г*) и алгебраические выражения для z, Y_1 и Y_2:

$$z = x_1 y_2,$$
$$Y_1 = \bar{x}_1 y_1 + y_1 \bar{y}_2 + x_2 \bar{y}_2 + \bar{x}_1 x_2,$$
$$Y_2 = x_1 y_2 + \bar{x}_1 y_1 + \bar{x}_1 x_2.$$

Схема, реализующая эти выражения, показана на рис. 1.2 (чтобы упростить рисунок, опущены линии, связывающие генерируемые схемой сигналы y_1, y_2 и $\bar{y}_2$ со входами элементов И). На первый взгляд кажется, что поведение этой схемы должно соответствовать нашей таблице переходов, и оно действительно соответствовало бы, если бы схема была реализована идеальным образом. В наших рассуждениях, однако, повсюду используется предположение, которое в принципе не может быть фактически реализовано, а именно, что в логических элементах и связях между элементами отсутствуют задержки. Если отойти от такого предположения, появляются различные типы возможных нарушений правильной работы схемы.

Рассмотрим, например, переход, происходящий тогда, когда система находится в состоянии 2—00 (см. табл. 1.5, *а*) и x_1 принимает значение 1. Этот переход приводит к появлению полного состояния 2—10, и выход $\bar{x}_1 y_1$-элемента рис. 1.2, возбуждающий γ_1 и γ_2, становит-

ся равным 0. В то же время выход x_1y_2-элемента возбуждается, удерживая Y_2 в 1. Таким образом, если бы такая ситуация действительно имела место в схеме, результатом явилось бы нулевое значение Y_1 и единичное значение Y_2, и схема перешла бы в состояние 3—10 в

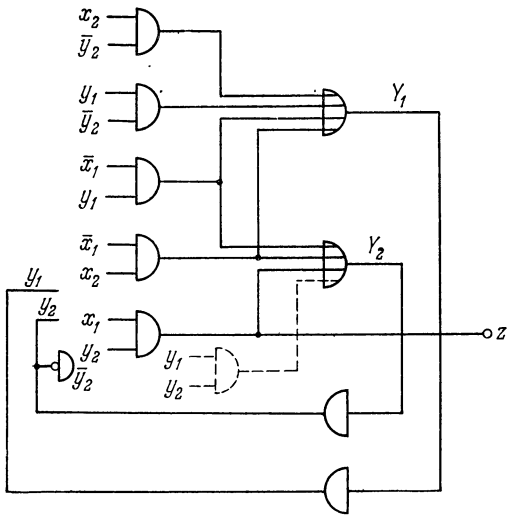


Рис. 1.2. Схема, построенная по табл. 1.5.

соответствии с таблицей переходов. Теперь допустим, что x_1y_2-элемент имеет большую задержку. Тогда Y_2 успеет стать равным 0, и если интервал, на котором выходы обоих $\bar{x}_1y_1$- и x_1y_2-элементов нулевые, достаточно длинный, то y_2 также станет равным 0, благодаря чему система перейдет в состояние 1—10. При этом x_1y_2-элемент не получит возбуждающего сигнала, и система останется в устойчивом состоянии.

Анализ этой ситуации, проведенный в общем виде в главе 4, показывает, с какими затруднениями приходится сталкиваться при описании поведения комбинационной схемы, генерирующей Y_2. Хотя установившееся значение выхода этой схемы есть действительно 1, как и задано, при условии, что полные состояния $x_1x_2y_1y_2$ есть 0001 и 1001, оно может измениться на 0, если между

этими состояниями происходит изменение входного сигнала. Принято считать, что переходные ошибки в работе комбинационной схемы, обусловленные наличием паразитных (незапланированных) задержек, являются результатом *комбинационных состязаний.*

Комбинационные состязания, связанные с изменением только одного входного сигнала, могут быть устранены путем модификации логической схемы, например, в нашем случае добавлением элемента И, генерирующего значение y_1y_2, в схему, реализующую функцию Y_2, что указано пунктирными линиями на рис. 1.2. Такое решение не всегда возможно, когда одновременно меняются несколько входов.

В некоторых случаях ошибочные сигналы, возникающие при переходах, могут быть отфильтрованы с помощью элементов задержки с инерциальными свойствами. Так, в нашем примере, если задержка между Y_2 и y_2 будет инерциальной и достаточной величины, выход x_1y_2-элемента станет равным 1 до того, как y_2 примет значение 0, препятствуя появлению ошибочного значения $Y_2 = 0$. Это одна из существенных причин введения элементов задержки в y-ветви схем.

Из предыдущего рассмотрения становится понятным, как ошибка, возникающая в комбинационной схеме в момент перехода, может привести к ошибочному установившемуся состоянию последовательностной схемы. Покажем теперь, как паразитные задержки могут доставить неприятности еще более хитрым образом.

Вернемся к таблице 1.5, *а* и схеме рис. 1.2 с добавленным y_1y_2-элементом, обозначенным пунктиром, и рассмотрим тот же переход, который рассматривался выше. Когда x_1 становится равным 1 (в исходном полном состоянии 2—00), выход $\bar{x}_1y_1$-элемента становится равным 0, что влечет $Y_1 = 0$, но в это время значение $Y_2 = 1$ еще поддерживается y_1y_2-элементом. Допустим еще раз, что x_1y_2-элемент имеет большую задержку. Тогда лишь y_1y_2-элемент поддерживает возбужденное состояние Y_2 в этот момент. Поскольку при этом $Y_1 = 0$, то после некоторой задержки (предположим, меньшей, чем задержка x_1y_2-элемента) y_1 становится равным 0. Это в свою очередь делает равным нулю выход y_1y_2-элемента, отключает Y_2 и тем самым снимает сигнал $y_2 = 1$ на вхо-

де x_1y_2-элемента с задержкой. Следовательно, Y_2 возвращается в 0 и в итоге система оказывается в состоянии 1—10.

Возможность появления подобной последовательности событий могла бы быть обнаружена и по матрице переходов без рассмотрения работы схемы. Именно, изменение входного сигнала x_1 вызвало (правильное) изменение внутреннего состояния с 2 на 3 (Y_1 стал равным 0, Y_2 остался равным единице), но по отношению к выходу Y_2 изменение внутреннего состояния произошло прежде, чем изменение входа из-за наличия задержек в логике дошло до этого выхода. Следовательно, система оказалась в состоянии 3—00, где Y_2 неустойчиво. При этом Y_2 изменится на 0 и, когда y_2 также станет равным нулю, система остановится в устойчивом состоянии 1—10.

Отсюда следует, что изменение логики схемы может быть недостаточным для решения задачи. Как показано в главе 4, недостаточным может оказаться даже изменение кодирования состояний. Единственным выходом является выбор такого соотношения между задержками, при котором невозможно появление подобного рода ошибок. Это всегда можно сделать включением элементов с одинаковыми задержками в y-ветви, хотя даже грубая оценка паразитных задержек показывает, что на практике подобное включение элементов не является необходимым. Состязания рассмотренного вида называются *существенными*, они появляются тогда, когда в таблице переходов имеются полное состояние T и вход x такие, что если система находится в состоянии T и x меняется один раз, то результирующее полное состояние отличается от того, которое было бы, если бы x изменился два раза или более.

Обусловленные паразитными задержками последствия (особенно зависимость их от того, где проявляются паразитные задержки), ограничения на изменения входов, типы элементов задержки и т. п. рассматриваются в главе 4.

1.5. Обратная связь

Анализ схемы на рис. 1.2 показывает, что в этой последовательностной схеме имеется несколько петель обратных связей. Это также следует из алгебраических выражений для Y_1 и Y_2. Эти выражения содержат члены,

которые являются положительными функциями от y_1 и y_2 соответственно. Значение y_i всегда определяется значением Y_i (возможно, через задержку), что говорит о наличии обратной связи.

Можно спросить, стоит ли говорить об этом по отношению к такой простой схеме. Дело в том, что, как будет показано дальше, только относительно небольшой класс таблиц переходов может быть реализован схемами без обратных связей. Для большого и представляющего интерес класса последовательностных функций схемные реализации таковы, что каждый Y_i зависит, в частности, от y_i.

Важной особенностью рассматриваемого здесь типа обратных связей является необходимость усиления сигнала в контуре обратной связи. В тех случаях, когда логика реализована на пассивных физических элементах (например, на диодных вентилях), в контуры необходимо включать усилители.

Наличие обратных связей затрудняет также определение мест неисправностей. Часто идут даже на включение в схему специальных цепей, разрывающих обратные связи, с тем, чтобы упростить ремонтные и диагностические процедуры.

Из сказанного следует важность использования так называемого *индекса обратной связи* как меры количества обратных связей в схеме. Этот индекс определяется как минимальное число проводов, которые необходимо разорвать с тем, чтобы устранить все обратные связи в схеме.

Число необходимых усилителей и число необходимых точек разрыва в точности равны индексу обратной связи.

Анализ процедуры синтеза показывает, что число внутренних переменных является верхней оценкой индекса обратной связи, поскольку, если разорван путь от Y_i к y_i для каждого i, то замкнутые петли отсутствуют. В примере на рис. 1.2 это число равно 2 и совпадает с индексом обратной связи.

Следующий очевидный вопрос: «Может ли индекс обратной связи быть меньше, чем число внутренних переменных?» Ответом является «да», и в главе 5, где подробно рассматриваются вопросы обратных связей

в последовательностных схемах, описывается метод синтеза схем с минимальным числом обратных связей, пригодный для широкого класса последовательностных функций.

1.6. Способы работы

Рассматриваемый столь подробно пример является типичным для большого и важного класса последовательностных схем. Имеется, однако, большое разнообразие подклассов таких схем, а также и существенные отличия в способах их работы. В этом разделе мы укажем свойства, по которым отличаются различные классы схем. Прежде всего, рассмотрим пример схемы такого типа, который существенно отличается от описанного ранее типа схем.

Пусть дана схема, показанная на рис. 1.3, с входами x_1, x_2, с *синхронизирующим входом* (СВ) и выходом z. Сигнал СВ состоит из последовательности импульсов, генерируемых независимым источником. Каждый из импульсов, проходя через элементы И, может вызывать переключение τ-триггеров *). Хотя слово «синхронизирующий» предусматривает регулярную, периодическую операцию, это несущественно — лишь бы соседние импульсы были разделены некоторым минимальным временны́м интервалом (паузой). Тем не менее, обычно импульсы синхронизации являются регулярными.

Входы x_1 и x_2 являются двоичными сигналами рассмотренного ранее типа с тем лишь ограничением, что они не могут меняться во время импульса СВ. На каждом импульсе СВ схема воспринимает текущее состояние входов. Во время пауз между синхронизирующими им-

*) τ-триггер (или *триггер со счетным входом.— Прим. ред.*), применяемый как элемент памяти, работает следующим образом: он имеет два состояния, причем в одном состоянии $y = 1$ и $\bar{y} = 0$, в другом $y = 0$ и $\bar{y} = 1$. При подаче на его вход импульса с определенными формой, шириной и амплитудой, состояние τ-триггера меняется, и он переходит из одного состояния в другое. При отсутствии импульсов τ-триггер будет находиться в текущем состоянии сколь угодно долго. Такие устройства довольно легко реализовать, используя лишь пару транзисторов в качестве активных элементов.

пульсами на входы не накладывается ограничений. В этом состоит основное отличие в работе подобных схем от схемы, изображенной на рис. 1.2.

Поведение системы может быть описано следующим образом.

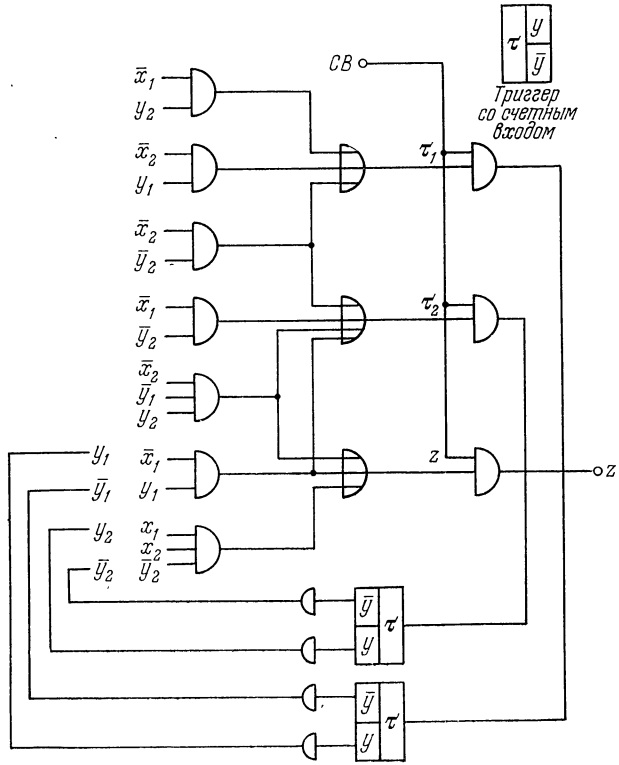


Рис. 1.3. Синхронная последовательностная схема.

Пусть на входы схемы (см. рис. 1.3) поданы некоторые значения сигналов x_1x_2, и τ-триггеры установлены в некоторое начальное состояние y_1y_2. Включается генератор синхронизирующих сигналов. На первом же импульсе синхронизации сигналы τ_1, τ_2 и z, выработанные соответствующими комбинационными схемами, поступают через

выходные элементы И на управление τ-триггерами и на выход системы. Триггеры переключаются и система переходит в новое внутреннее состояние. Синхронизирующий импульс заканчивается до того, как новые значения сигналов y_1 и y_2 могут прийти обратно к выходным элементам И. После окончания импульса синхронизации входы могут меняться произвольным образом. Потом снова приходит синхронизирующий импульс, и описанная картина повторяется.

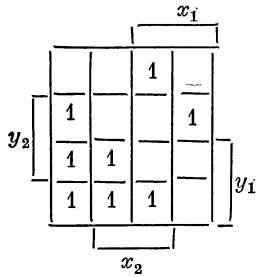
Допустим, нам задано в этом примере, что входные двоичные переменные x_1x_2 кодируют десятичные целые числа от 1 до 3 так, что $x_1x_2 = 01$ соответствует 1, $x_1x_2 = 10$ соответствует 2 и $x_1x_2 = 11$ соответствует 3; комбинация $x_1x_2 = 00$ никогда не появляется. Кроме того, начальным состоянием является $y_1y_2 = 00$. Тогда анализ схемы можно провести следующим образом. Составляем выражения (без учета синхронизирующего импульса) для z, τ_1 и τ_2:

$$z = \bar{x}_2\bar{y}_1y_2 + \bar{x}_1y_1 + x_1x_2\bar{y}_2,$$
$$\tau_1 = \bar{x}_1y_2 + \bar{x}_2y_1 + \bar{x}_2\bar{y}_2,$$
$$\tau_2 = \bar{x}_2\bar{y}_2 + \bar{x}_1\bar{y}_2 + \bar{x}_2\bar{y}_1y_2 + \bar{x}_1y_1.$$

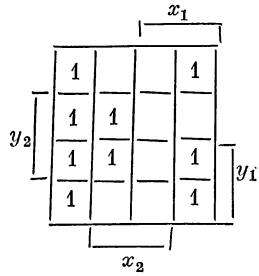
По этим выражениям, в свою очередь, строим карты, показанные в таблице 1,6, *а*, *б* и *в*. Затем очевидным образом строится матрица переходов, приведенная в табл. 1.6, *г*. Заметим, например, что в полном состоянии 2—01 $\tau_1 = 1$, $\tau_2 = 0$. Следовательно, y_1 будет меняться, тогда как y_2 остается неизменным. Поскольку при этом y_1y_2 есть 01, то следующим состоянием y_1y_2 будет 11. Это y-состояние соответствует строке 3, так что следующим внутренним состоянием для 2—01 будет 3. Значения выхода получаются непосредственно по таблице 1.6, *а*.

Заметим, что поскольку значения $x_1 = 0$, $x_2 = 0$ не могут появиться и начальным является состояние 1, то состояние 4 является недостижимым. Таким образом, таблица переходов, описывающая правильное поведение схемы, может не включать состояние 4, что показано в таблице 1.6, *д*: вычеркнут столбец 00, а остальные столбцы переобозначены и упорядочены в соответствии с рассматриваемой интерпретацией состояний входов. Внешнее

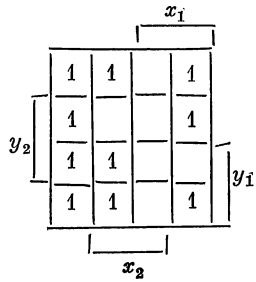
Таблица 1.6



а) z-карта



б) τ_1-карта



в) τ_2-карта

	x_1x_2 00	01	11	10	y_1	y_2
1	3,0	2,0	1,1	3,0	0	0
2	4,1	3,0	2,0	1,1	0	1
3	1,1	1,1	3,0	2,0	1	1
4	2,1	3,1	4,1	2,0	1	0

г) Матрица переходов

	n_1	n_2	n_3
1	2,0	3,0	1,1
2	3,0	1,1	2,0
3	1,1	2,0	3,0

д) Таблица переходов

поведение схемы точно и полно описывается полученной таблицей переходов. Заметим, что устойчивые состояния здесь не выделяются каким-либо образом, поскольку они несущественны для такого типа схем. При начальном состоянии 1 мы можем легко определить выходную последовательность для любой входной последовательности. Пусть, например, входная последовательность есть $n_3n_2n_1$. На первый входной символ на выходе вырабатывается

единица, соответствующая полному состоянию $1 - n_3$, и следующее состояние есть 1. Вход n_2 приводит к получению 0 на выходе и переводит систему в состояние 3, в котором вход n_1 генерирует 1 на выходе и меняет внутреннее состояние снова на 1. Найти ясное, полное словесное описание, эквивалентное таблице переходов, в лучшем случае крайне тяжело, а обычно и просто невозможно. При внимательном анализе таблицы переходов можно заметить, что выход z становится равным 1 всякий раз, когда сумма индексов поданных входов делится на 3.

Заметим, что некоторые из рассматриваемых переходов должны были бы содержать критические состязания, в которых требуется одновременное изменение y_1 и y_2 (см., например, полное состояние 1—10). Критические состязания не представляют проблемы в *синхронных системах* таких, как рассматриваемая, поскольку синхронизирующий импульс оканчивается до того, как изменение состояния какого-либо триггера успеет пройти через элемент задержки и вызвать изменение на входе комбинационной схемы. Таким образом, единственное требование на выбираемый способ кодирования строк состоит в том, чтобы никаким двум различным строкам не была приписана одна и та же кодовая комбинация. Подобное упрощение процедуры кодирования и тот факт, что проблема состязаний фактически исчезает, обусловил широкое использование таких систем. Однако за все это приходится расплачиваться или быстродействием, или надежностью системы, поскольку ограничения на синхронизирующие импульсы должны выполняться для худших сочетаний паразитных задержек и технологических отклонений. Возможны многие разновидности рассматриваемой здесь модели, в частности, большие преимущества достигаются при использовании двух синхронизирующих генераторов, работающих в различных фазах. Некоторые из подобных проблем, связанных с синхронными системами, рассматриваются в главе 6.

Другой способ представления информации, содержащейся в таблице 1.6, *д*, а именно *диаграмма состояний* *), показан на рис. 1.4. Состояния представляются в

*) Часто диаграмму состояний называют *графом переходов*. (*Прим. ред.*)

виде пронумерованных вершин, а переходы — в виде дуг, помеченных символами входов и выходов. Хотя некоторые исследователи утверждают, что такие графы содержат информацию в более наглядной форме, по мнению автора это преимущество быстро исчезает, когда число вершин в графе становится больше пяти или шести. Диаграммы состояний имеют некоторое преимущество перед таблицами переходов при описании систем со многими состояниями входов, причем для каждого внутреннего состояния допустимы лишь несколько состояний входов. Соответствующая таблица переходов в таких случаях содержит много столбцов, но большинство клеток в них будут не заполнены.

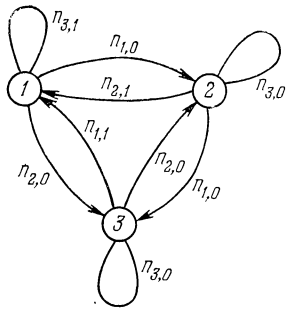


Рис. 1.4. Диаграмма состояний, эквивалентная табл. 1.6, *д*.

Таким образом, мы рассмотрели два основных вида ограничений на входные сигналы. В первом из них, иллюстрированном примером со светофором, требовалось лишь, чтобы между последовательными изменениями был некоторый временной интервал. Схемы, работающие подобным образом, будем называть *схемами Хаффмана*, поскольку он первым разработал большинство из основных принципов, лежащих в основе их анализа и синтеза.

Более подробный анализ физических условий указывает на то, что с такой схемой необходимо связать два критических интервала δ_1 и δ_2, где $\delta_1 > \delta_2$. Если два сигнала меняются на временном интервале, меньшем чем δ_2, они рассматриваются как меняющиеся одновременно. Если изменения разделены временным интервалом, бо́льшим δ_1, схема раздельно реагирует на каждое из этих изменений. Если же изменения происходят на интервале, лежащем между δ_1 и δ_2, то нельзя предсказать, как поведет себя схема. В нашем примере, где нет ограничений на относительные временные изменения x_1 и x_2, не важно, рассматриваем ли мы изменения входов (состоящие в изменении сначала одной переменной, а затем другой) как происходящие одновременно или после-

довательно. Однако во многих ситуациях накладывается ограничение, состоящее в том, что каждое изменение входа соответствует изменению значения только одной переменной.

Общее требование, чтобы изменения входов были минимально разнесены во времени, очень близко к определению так называемого *основного способа работы* схемы, согласно которому «на входы накладывается такое ограничение, что они могут меняться лишь тогда, когда все элементы памяти находятся в устойчивых состояниях ($Y_i = y_i$ для всех i)». Однако это определение более «жесткое», оно не охватывает, например, схемы, в которых устойчивое состояние таблицы переходов соответствует циклической смене (генерации) некоторых y-состояний.

Вторая разновидность ограничений на входы, иллюстрированная схемой на рис. 1.3, состоит в разрешении изменения входов только во время пауз между синхронизирующими импульсами. При этом учитываются лишь те «чистые» изменения входов, которые происходят во время пауз. Весьма близким к этой разновидности было бы ограничение, состоящее в том, чтобы входные сигналы подавались на схему в начале периодов, определяемых синхронизирующими импульсами, и снимались в моменты окончания синхроимпульсов. Если входы удовлетворяют первому ограничению, то последнее ограничение легко реализовать, подав каждый вход и его отрицание на элемент И, управляемый сихронизирующим сигналом. Сигналы, удовлетворяющие последнему ограничению, удовлетворяют также первому ограничению. Существенной особенностью *синхронных систем,* т. е. систем, реализующих *синхронный способ работы,* является введение независимо генерируемого сигнала, определяющего периоды, на которых разрешены изменения входов.

Третий класс ограничений на входах охватывает случай, когда последовательностная схема вырабатывает сигнал «готовности», и изменения входов допускаются только тогда, когда выработан сигнал готовности, который, таким образом, можно рассматривать как синхроимпульс, генерируемый изнутри. Преимущество здесь состоит в том, что, если схема может быть выполнена

таким образом, что сигнал готовности генерируется, как только схема «классифицирует» предыдущее состояние входа, и если можно, чтобы состояния входов подавались на схему «по требованию», то средняя частота изменений входов, при которой эти изменения могут быть обработаны схемой, будет существенно выше той частоты, которая возможна в рассмотренных выше схемах. Поскольку основы теории схем, функционирующих при таких ограничениях, были разработаны Д. Мюллером, эти схемы будут называться здесь *схемами Мюллера* (будем использовать также термин «*схемы, не зависящие от скорости*»).

Четвертая категория схем, где учитываются ограничения на входы, включает требование, чтобы для каждого состояния входа был свой входной полюс, при этом каждое состояние входа отмечается наличием импульса на соответствующем полюсе (естественно, в каждый момент времени только на одном входном полюсе будет такой импульс). Ширина входных импульсов ограничена, и между соседними импульсами должен быть некоторый минимальный временной интервал (пауза). Хотя здесь и нет синхронизирующих сигналов, схемы подобного рода, которые мы будем называть *импульсными схемами*, или схемами, реализующими *импульсный способ работы*, строятся точно так же, как и синхронные схемы. Используется та же форма таблицы переходов; аналогичной является и проблема кодирования строк.

Схемы Мюллера и Хаффмана можно классифицировать как *асинхронные схемы*, в отличие от синхронных или импульсных схем, которые требуют синхронизации входов каким-либо внешним сигналом или импульсом.

Дальнейшее подразделение схем может быть сделано по характеру выходных сигналов. Выходами синхронных или импульсных схем могут быть импульсы, синхронизированные с тактовым сигналом, как на рис. 1.3, или со входным импульсом. Можно также сделать выходы таких схем зависящими только от внутренних состояний, без учета синхронизирующих сигналов.

Выходы асинхронных схем могут быть классифицированы по числу возможных изменений состояния выхода в ответ на однократное изменение состояния входа. Пожалуй, наиболее важный класс схем составляют те схемы, в которых однократное изменение состояния входа

вызывает не более одного изменения состояния выхода. Таблицы переходов для последовательностных функций с этим свойством всегда могут быть построены таким образом, что каждое неустойчивое состояние переходит непосредственно в устойчивое с тем же самым состоянием выхода. В таблице 1.2 представлена функция такого типа, которую мы назовем *функцией с однократным изменением выхода* (ОИВ-функция). Если добавляется условие, что в каждый момент времени может меняться лишь одна входная переменная и схема реализует основной способ работы, то используется термин «*нормальный основной способ работы*».

Таблица 1.7

	x_1x_2 00	01	11	10
1	***1***,00	2,10	***1***,00	4,10
2	***2***,10	3,00	***2***,01	3,11
3	1,01	4,11	***3***,01	***3***,10
4	***4***,10	***4***,01	3,00	***4***,01

а) Функция с многократным изменением выхода

	A	*B*	*C*
1	***1***,0	2,0	4,0
2	1,0	1,1	4,0
3	1,1	***3***,0	***3***,1
4	***4***,0	3,0	***4***,1

б) Функция с неограниченным изменением выхода, устойчивые состояния предсказуемы

	A	*B*	*C*
1	***1***,00	2,01	***1***,00
2	***2***,01	3,11	3,01
3	***3***,11	1,10	***3***,11

в) Функция с неограниченным изменением выхода, устойчивые состояния не предсказуемы

Более широкий класс составляют функции, в которых в ответ на однократное изменение состояния входа состояние выхода меняется несколько раз, но не больше некоторого фиксированного числа. Пример такой *функции с многократным изменением выхода* (МИВ-функции) приводится в таблице 1.7, *а*. Если в системе, находящей-

ся в состоянии 1—00, с состоянием выхода $z_1z_2 = 00$, вход изменился на 01, последовательность полных состояний будет 1—00, 1—01, 2—01, 3—01, 4—01, а последовательность состояний выходов — 00, 10, 00, 11, 01. Значение выхода z_1 меняется четыре раза, выхода z_2 — дважды. Заметим, что в этой таблице никакое изменение входов не может вызвать более чем четырехкратное изменение выходного сигнала.

Таблица 1.7, *б* описывает функцию, в которой в ответ на однократное изменение состояния входа выход может меняться неограниченное число раз. Мы назовем такие функции *функциями с неограниченным изменением выхода* (НИВ-функция). Если система находится в состоянии $1 — A$ и состояние входа меняется на B, то начинается циклическая смена состояний $1 — B$ и $2 — B$, которая будет продолжаться до тех пор, пока вход снова не изменится. При этом значение выхода z колеблется между значениями 0 и 1. Число колебаний является функцией длины интервала, на котором сохраняется значение входа, равное B, и временных параметров схемы. Теперь, если колебания прекращены путем изменения входа, следующим устойчивым состоянием будет $1—A$ или $4—C$ в зависимости от того, какой вход подан на схему, A или C соответственно, безотносительно к тому, какое внутреннее состояние, 1 или 2, было в момент изменения входа. Таким образом, последовательность устойчивых состояний в таблице переходов всегда может быть предсказана при задании начального состояния и входной последовательности.

Сказанное не справедливо для таблицы 1.7, *в*, в которой изменение состояния входа с B на A может привести к переходу в любое из трех устойчивых состояний в зависимости от того, в каком состоянии цикла, 2—01, 3—11 или 1—10, система находилась в момент изменения входа.

Полезно также рассмотреть класс функций, в которых в ответ на однократное изменение состояния входа каждый выходной сигнал не может менять свое значение более одного раза. Этот класс, очевидно, содержит всякую ОИВ-функцию. В то же время, если система имеет, например, три выходных полюса, однократное изменение входа может повлечь следующие изменения состояния выхода $000 \to 100 \to 110 \to 111$, так что функция не яв-

ляется ОИВ-функцией, но принадлежит указанному здесь более широкому классу.

Большинство материала, рассматриваемого в последующих главах, относится к асинхронным схемам. Затрагиваются лишь те аспекты работы синхронных схем во времени, которые требуют для своего решения тех же методов, что и в асинхронных схемах. В классе асинхронных схем основное внимание уделено схемам Хаффмана, поскольку по ним накоплен обширный материал, особенно в плане проблемы синтеза. Гораздо меньше данных по формальным методам синтеза схем Мюллера. Однако в главы 4 и 6 включен материал, который связывает эти два вида схем: предлагается некоторые методы синтеза схем Хаффмана использовать для синтеза схем Мюллера. Следует заметить, что при этом никакая попытка сравнения с результатами, полученными школой Мюллера, не предпринималась. В необходимых местах читатель отсылается к первоисточникам.

1.7. Применения

Асинхронные последовательностные схемы того или иного вида встречаются в разнообразнейших ситуациях, от автоматических подъемников до межпланетных кораблей. Некоторые из них весьма просты и их функции определяются непосредственно функциями системы, к которой они принадлежат, как, например, в случае со светофором; построение других — неважно, простых или сложных,— рассматривается как составная часть процедуры синтеза всей системы, как, например, синтез схем управления при разработке вычислительных машин. Даже если вся система разрабатывается как синхронная, возможны ситуации, где необходимо применять асинхронные схемы, например, в схемах, управляющих взаимодействием между быстродействующей ЦВМ и более медленными периферийными устройствами.

Часто схемы Хаффмана можно спроектировать таким образом, что они будут более быстродействующими и более надежными, чем синхронные или импульсные схемы, выполняющие те же функции, но обычно платой за это является бо́льшее число элементов. Сказанное иллюстрируется схемами в главе 7. Требование сохранения

стоимости схемы в разумных пределах определяет некоторую номинальную частоту работы схемы, причем эта частота, похоже, непрерывно возрастает, а с появлением дешевых быстродействующих интегральных схем такая тенденция существенно усилится.

Быстрая эволюция новых элементов и технологии производства схем, необходимость новых применений поставили перед разработчиками ряд новых проблем, на которые пока нет ответа. Такое положение обусловлено как относительной новизной применяемых средств, так и сложностью критериев, которым должны удовлетворять разрабатываемые устройства. Несомненно, разрыв между теорией и практикой будет сокращаться, поскольку новые элементы и задачи с теоретической точки зрения могут быть аналогичны изученным ранее. Например, теория релейно-контактных схем, которые кажутся архаизмом в сравнении с современными быстродействующими системами, может быть тем не менее пригодна при разработке логических схем, использующих полевые транзисторы и, возможно, криотроны.

Природа технических наук такова, что теория намечает лишь исходные точки для инженера, указывает пределы возможностей различных направлений, выделяет критические факторы, оценивает влияние различных параметров и дает отдельные решения частных задач. Не следует ожидать, что будет найден теоретический подход, который даст законченное решение сложных проблем, возникающих при синтезе систем. Так, например, хотя теория переключательных схем и дает методы реализации заданной таблицы переходов схемой с минимальным числом усилителей, отсюда никак не следует, что разработчик должен слепо применять эти методы, даже если такие усилители дороги при используемой технологии. Каждый раз необходимо искать такой вариант, в котором были бы «сбалансированы» различные факторы. Процедура «чистого» синтеза, при котором оптимизируется какой-либо один фактор, полезна в том смысле, что она устанавливает предел, до которого можно дойти, двигаясь в данном направлении; иногда полезно задаться некоторым исходным вариантом и вводить в него различные изменения, связанные с рассмотрением других возможных направлений.

Большинство алгоритмов из теории переключательных схем запрограммировано на ЦВМ и, несомненно, такая практика станет общепринятой. Далее, следует ожидать появления совместимых комплектов программ, которые разработчик может использовать в сочетании друг с другом. Опыты с такого рода программами показывают, что использование ЦВМ отнюдь не исключает необходимости всеобъемлющего понимания рассматриваемых процессов, а также необходимости разработки эффективных алгоритмов. Комбинаторный характер проблем, имеющих место при разработке переключательных схем, обусловливает резкое увеличение времени вычислений при увеличении размеров задачи. В этих условиях применение упомянутых алгоритмов ненамного эффективнее вычислений вручную. Поэтому, хотя в последующих главах машинные методы и не рассматриваются подробно, необходимо иметь в виду, что они являются «фоном» и не влияют заметно на значимость представленного материала.

ЗАМЕЧАНИЯ ПО БИБЛИОГРАФИИ

Подробные ссылки на источники, относящиеся к понятиям настоящей главы, сделаны в последующих главах, где эти понятия рассматриваются более подробно. Одной из ранних работ по асинхронным последовательностным схемам, а также по теории комбинационных переключательных схем является работа Кейстера, Ричи и Уошберна [55]. Основы современного подхода к асинхронным схемам были заложены Хаффманом [49]. Первоначальные сведения можно найти в хороших книгах Колдуэлла [12], Маккласки [74] и Миллера [79, 80], в последней из которых содержится введение в теорию схем Мюллера. Другими основными работами по теории переключательных схем являются работы Мили [77] и Мура [81]. Понятия «основной способ работы» и «нормальный основной способ работы» были введены Маккласки [71]. Здесь же следует отметить работы Смита, Трейси, Шёффеля и Маки [102] и Тана [106] по автоматизации проектирования. Интересное применение рассмотренных выше понятий для параллельных вычислительных процессов (в плане аппаратуры и матобеспечения) дано в работе Бредта и Маккласки [8].

ЗАДАЧИ*)

1.1^{+}. Предполагая, что в каждый момент времени меняется только одна входная переменная и начальным состоянием является 1—00, дать соответствующее словесное описание для функции, задаваемой первичной таблицей переходов (табл. 1.8).

Таблица 1.8

	x_1x_2 00	01	11	10	z
1	***1***	5	—	7	0
2	***2***	6	—	7	0
3	***3***	6	—	7	1
4	2	***4***	10	—	0
5	3	***5***	10	—	0
6	3	***6***	10	—	1
7	1	—	11	***7***	0
8	1	—	12	***8***	0
9	1	—	12	***9***	1
10	—	4	***10***	8	0
11	—	4	***11***	9	0
12	—	4	***12***	9	1

1.2. Составить минимальную первичную таблицу переходов для последовательностной функции, имеющей два входа x_1 и x_2, один выход z и удовлетворяющей следующим условиям:

а) всякий раз, когда x_1 меняется из 0 в 1, z меняется три раза;

б) всякий раз, когда x_2 меняется из 0 в 1, z меняется дважды;

в) во всех других случаях z остается постоянным;

г) считаем, что x_1 и x_2 никогда не изменяются одновременно и входные изменения происходят только при установившихся значениях;

д) в состоянии 1—00 $z = 0$.

1.3. По заданному участку дороги с однорядным движением изредка проезжают автомобили и грузовики, дви-

*) Для задач, отмеченных знаками «×» и «+», в конце книги приведены решения.

жущиеся лишь в одном направлении. Необходимо разработать систему, отделяющую грузовики от легковых автомобилей. Будем использовать при этом пару фотоэлементов x_1 и x_2, разместив их в определенных точках дороги, как

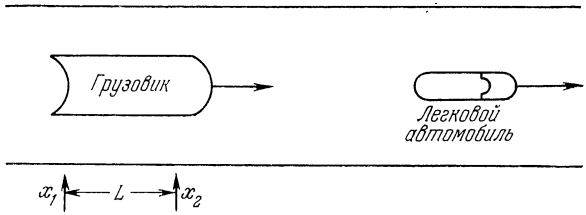


Рис. 1.5.

указано на рис. 1.5. Фотоэлементы разделены некоторым расстоянием L, и каждый из них выдает постоянный сигнал (напряжение), пока какая-нибудь часть машины проходит перед ним. Длина грузовика считается большей L, но меньшей $2L$, а все легковые автомобили короче чем L. Предполагаем, что при прохождении перед фотоэлементами, расстояние между соседними машинами не может быть меньше чем $2L$.

Построить первичную таблицу переходов для асинхронного устройства со входами x_1 и x_2, согласно которой выход z возбуждается всякий раз при прохождении по дороге грузовика слева направо.

1.4$^+$. Построить первичную таблицу переходов для последовательностной схемы, которая будет генерировать тактовые импульсы C_1 и C_2 следующим образом. Входом схемы является последовательность импульсов минимальной ширины w с минимальной паузой между импульсами. После импульса C_1 должен следовать импульс C_2 через интервал, не меньший чем s, и такой же интервал должен отделять последующий импульс C_1, появляющийся вслед за импульсом C_2. Ширина импульсов C_1 и C_2 приблизительно равна w. Нам надо, по существу, так разделить входные импульсы, чтобы они поступали попеременно на соответствующие выходные полюсы C_1 и C_2.

1.5. В импульсной синхронной системе необходимо точно синхронизировать положение сигнала x. Сигнал x

должен появляться на входе системы в некоторый момент времени t. Однако вследствие паразитных задержек нам неизвестно точно, насколько раньше t может появиться передний фронт этого сигнала. Для устранения этой неопределенности предлагается использовать новый импульсный сигнал z, передний фронт которого появляется в момент t, а ширина равна w, если в момент t имеется сигнал x. Управление импульсом z производится с помощью сигнала повторения k, который включается в момент t, а выключается — в момент $t + w$. Построить первичную таблицу переходов для схемы, вырабатывающей сигнал z как требуемую функцию от x и k.

1.6. Рассмотрите синхронную последовательностную функцию одной переменной x, которая вырабатывает на выходе значение 1, если к текущему моменту времени число поступивших единичных значений входов превышает число поступивших нулевых значений входов; в противном случае выход равен 0. Если возможно, составьте таблицу переходов для такой функции. При невозможности составления такой таблицы укажите, в чем причина затруднений.

ГЛАВА 2
МИНИМИЗАЦИЯ ТАБЛИЦ ПЕРЕХОДОВ

Можно указать много причин, по которым желательно уменьшение числа строк (минимизация) таблицы переходов. Одна из них состоит в том, что меньшая таблица более обозрима. Лишние состояния затрудняют понимание работы описываемой системы. Вторая причина в том, что уменьшение числа состояний может привести к уменьшению числа внутренних переменных, необходимых для кодирования таблицы. Это, в свою очередь, приводит к экономии тех компонент (например, элементов задержки, усилителей, триггеров), которые соответствуют внутренним переменным. Возможна также экономия и в логике, поскольку, во-первых, необходимо реализовать меньше функций возбуждения, и, во-вторых, каждая из оставшихся функций будет зависеть от меньшего числа аргументов. Если даже число состояний и не уменьшается, экономия в логике может быть достигнута за счет введения в матрицу переходов дополнительных «безразличных» строк. Указанные выше виды экономии сопровождаются повышением надежности, уменьшением технологических и эксплуатационных расходов (меньше компонент могут выходить из строя, проще процедуры проверок и замены компонент).

Конечно, каждое из таких преимуществ не вытекает непосредственно из процесса уменьшения числа состояний. Можно указать случай, когда таблица с меньшим числом состояний менее обозрима, чем эквивалентная таблица с бóльшим числом состояний; существуют также таблицы, которые для своей реализации требуют более сложной логики, чем эквивалентные таблицы с бóльшим числом кодируемых строк и бóльшим числом внутренних переменных. В среднем, однако, при отсутствии очевидных контрпримеров с уменьшением числа состояний можно ожидать уменьшения сложности реализации. Вряд ли

возможно, чтобы исходная неминимальная таблица сразу же имела форму, приводящую к оптимальной реализации. Проектировщик должен уметь устранить нежелательную избыточность в построенной таблице и затем каким-либо стандартным методом (требующим, может быть, даже некоторого расширения таблицы) привести ее к определенному желаемому виду.

При решении задач минимизации таблиц переходов применяются различные покрытия и отношения эквивалентности, которые используются при обработке таблиц переходов не только с целью ликвидации избыточных состояний.

Подход к проблеме минимизации таблиц переходов, принятый здесь, является довольно общим в том плане, что он применим к таблицам переходов, представляющим как синхронные, так и асинхронные функции, а также и к неполностью определенным таблицам. Тем не менее, основная теория, разрабатываемая в первых пяти разделах, оперирует с синхронными функциями, а аспекты проблемы, касающиеся асинхронных функций, отложены до раздела 2.6.

2.1. Общие понятия

Рассмотрим функцию, описываемую таблицей 2.1, *а*. Предполагается, что исходная система всегда находится в некотором внутреннем состоянии (строке) и при подаче

Таблица 2.1

	A	*B*	*C*	*D*
1	2,0	—,1	3,—	2,0
2	3,0	5,1	2,0	—
3	3,0	4,1	—	5,0
4	—	1,1	2,—	—
5	—	—	1,1	—

а) Таблица переходов для неполностью определенной последовательностной функции

t	*A*	*B*	*C*	*D*
	t,—	*t*,—	*t*,—	*t*,—

б) Поглощающая строка

входного сигнала следующее внутреннее состояние и выход системы определяются соответствующей клеткой таб-

лицы переходов. Так, если система находится вначале в состоянии, соответствующем строке 1, и подается входная последовательность *AAABB*, система пройдет через состояния 2, 3, 3, 4 и возвратится в состояние 1, выдав выходную последовательность 00011.

Черточками (прочерками) в таблице отмечены неопределенные состояния и значения выходов. Если система вначале находится в состоянии 3 и приложена входная последовательность *BCC*, то результирующая выходная последовательность будет 1—0. Черточка в выходной последовательности указывает на то, что нам безразлично, какой выходной сигнал будет выдан в этот момент времени. Считаем, что схема удовлетворяет требуемым условиям, независимо от того, генерирует она в этот момент 0 или 1, так же как не обязательно, чтобы данной черточке всегда соответствовал один и тот же вырабатываемый сигнал. Последнее замечание крайне важно, так как из него следует, что нельзя минимизировать неполностью определенную таблицу путем произвольного заполнения прочерков всеми возможными способами с последующей минимизацией результирующих полностью определенных таблиц (см. задачу 2.2).

Если в состоянии 4 прикладывается входная последовательность *BB*, то результирующая последовательность внутренних состояний будет 1 —, а выходная последовательность 11. Видим, что внутреннее состояние становится и остается неопределенным, так что независимо от того, какой входной сигнал подается после *BB*, все выходные сигналы будут неопределенными. Результат попадания системы в неопределенное состояние можно описать, дополнив таблицу «поглощающей» строкой t (табл. 2.1, *б*), в которой все следующие внутренние состояния равны t, а выходы не определены. Однако в методах, рассматриваемых ниже, это делать не обязательно.

Говорят, что состояние s_A таблицы переходов A *покрывает* состояние s_B таблицы переходов B, если для всякой конечной входной последовательности соответствующая выходная последовательность, получаемая по A с начальным состоянием s_A, совпадает с выходной последовательностью, получаемой по B с начальным состоянием s_B, всякий раз, когда последняя определена. Если, например, s_A покрывает s_B и имеется некоторая входная последова-

тельность, генерирующая 0—1— —0 из s_B, эта же входная последовательность может генерировать 001—00 из s_A.

Говорят, что таблица переходов A *покрывает* таблицу переходов B, если каждое состояние в B покрывается хотя бы одним состоянием из A. Если таблица A покрывает таблицу B, то таблица A является подходящей заменой таблицы B в том смысле, что любая схема, удовлетворяющая требованиям таблицы A, будет удовлетворять требованиям и таблицы B. Задача минимизации таблицы переходов состоит в нахождении для данной таблицы таблицы с минимальным числом строк, покрывающей исходную таблицу. Заметим, что отношение покрываемости (и для состояний, и для таблиц) является рефлексивным и транзитивным, но не симметричным.

Говорят, что таблицы переходов A и B *эквивалентны*, если A покрывает B и B покрывает A. В случае полностью определенных таблиц легко показать (см. раздел 2.4), что если A покрывает B, то B покрывает A, так что полностью определенные таблицы должны быть эквивалентны, если каждая из них покрывает другую. Определение эквивалентности является простым обобщением этого определения, данного в терминах полностью определенных таблиц. Две таблицы являются *тривиально эквивалентными*, если они совпадают с точностью до переобозначения строк.

В случае, когда множеством начальных состояний является некоторое заданное подмножество множества всех состояний, состояния, недостижимые из состояний этого подмножества, можно сразу вычеркнуть из таблицы переходов. Ниже предполагается, что перед применением описываемых процедур подобное удаление «лишних» состояний уже проведено. Таблица переходов называется *сильно связной*, если для любой пары состояний i и j найдется входная последовательность, переводящая i в j.

2.2. Совместимые состояния

Если таблица B может быть покрыта таблицей A с меньшим числом строк, то в B должны быть совокупности строк, покрываемых отдельными строками таблицы A. Отсюда вытекает понятие *совместимых состояний* как таких, которые покрываются одной и той же строкой неко-

торой таблицы A. Множество состояний таблицы переходов, покрываемых некоторым состоянием какой-либо другой таблицы переходов, называется *совместимым множеством.* Определение множеств совместимых состояний является ключевым моментом в процессе минимизации. Перед описанием этого процесса мы введем несколько понятий, полезных здесь и далее.

Для заданных входа I и начального состояния s, образующих полное состояние $s - I$, определим $N(s, I)$ и $Z(s, I)$ соответственно как следующее состояние и вырабатываемый выход, если последние определены. Далее, если $\mathcal{T}$ — последовательность состояний входа, определим $Z(s, \mathcal{T})$ как последний элемент (если он определен) выходной последовательности, вырабатываемой системой, находящейся в s, в ответ на входную последовательность $\mathcal{T}$. Аналогично определяется $N(s, \mathcal{T})$ как состояние (если оно определено), в котором окажется система, находящаяся в s, после приложения последовательности $\mathcal{T}$.

Возвращаясь к таблице 2.1, *а*, посмотрим, не могут ли состояния 2 и 5 быть покрыты некоторым состоянием s_{25} какой-либо другой таблицы? Если могут, то каким должно быть значение выхода для s_{25}? Согласно столбцу A, для состояния 2 таким значением должен быть 0, это же справедливо и для состояния 5, поскольку здесь элемент $Z(5, A)$ не определен. Аналогично, при входе B таким значением должна быть 1. При входе D, поскольку значения выходов не определены ни для 2, ни для 5, значение выхода для s_{25} также может быть неопределенным. Чтобы покрыть состояние 2, s_{25} при входе C должно вырабатывать выходной сигнал $Z(s_{25}, C) = 0$. Но тогда s_{25} не покрывает 5, так как для этого входа в состоянии 5 вырабатывается выходной сигнал 1. Следовательно, никакая строка не может покрыть состояния 2 и 5, поскольку их определенные выходы различны в столбце C. Таким образом, 2 и 5 несовместимы.

Необходимое условие совместимости двух строк состоит, очевидно, в том, чтобы в тех столбцах, где оба выходных сигнала определены, они были одинаковыми. Это условие так называемой *совместимости по выходам.*

Пара строк 1 и 3 удовлетворяет этим условиям. При определении строки s_{13}, покрывающей строки 1 и 3, не возникает, таким образом, проблемы задания значения

выходного сигнала. В каждом столбце I $Z(s_{13}, I)$ является множеством, равным $Z(1, I)$ или $Z(3, I)$ (тому, которое определено); $Z(s_{13}, I)$ является неопределенным, если $Z(1, I)$ и $Z(3, I)$ не определены. (Такими значениями для s_{13} будут $0, 1, —, 0$ при рассмотрении столбцов слева направо.) Такое определение значений выхода гарантирует, что условие покрываемости будет выполняться для всех входных последовательностей единичной длины. Рассмотрим, однако, задачу определения следующих состояний для s_{13}, в частности для входа D. Чтобы s_{13} покрывало состояние 1, $N(s_{13}, D)$ должно соответствовать строке, покрывающей состояние 2, определяемое как $N(1, D)$. Это условие является, очевидно, необходимым, поскольку если $\mathcal{J}$ — произвольная входная последовательность, то $Z(s_{13}, D\mathcal{J})$ (где $D\mathcal{J}$ — последовательность, первым символом которой является D, а далее идет $\mathcal{J}$) должно быть тем же самым, что и $Z(2, \mathcal{J})$, если последнее определено. Аналогично, $N(s_{13}, D)$ должно соответствовать строке, покрывающей состояние 5, определяемое как $N(3, D)$. Но это означает, что должно существовать некоторое состояние, покрывающее состояния 2 и 5, которые, как было показано выше, являются несовместимыми. Отсюда следует, что не существует удовлетворяющего нас значения для $N(s_{13}, D)$; поэтому состояние s_{13} не может существовать, поскольку строки 1 и 3 также несовместимы. Будем писать $a \sim b$, если a и b совместимы, в противном случае $a \nsim b$.

Предыдущие рассуждения можно подытожить, сказав, что $1 \sim 3$ следует из $2 \sim 5$, и поскольку $2 \nsim 5$, то $1 \nsim 3$.

В общем случае будем говорить, что множество Q *порождает* множество R, если R есть множество последователей (следующих состояний) для состояний из Q по некоторому входу. Предыдущий пример можно легко обобщить, показав, что, если p и q совместимы, то совместимой будет любая пара строк, порождаемая парой pq. С учетом такого обобщения справедлива следующая лемма.

Лемма 2.1. *Q является совместимым множеством тогда и только тогда, когда Q совместимо по выходам и по любой из прикладываемых входных последовательностей Q не порождает множеств, несовместимых по выходам.*

Доказательство. (а) Рассмотрим вначале *необходимость*. Если Q несовместимо по выходам, то Q, оче-

видно, не является совместимым множеством, так как при этом не существует непротиворечивого способа определения значений выхода для какого-либо покрывающего состояния. Допустим далее, что Q порождает несовместимое по выходам множество Q_m через совокупность последовательно порождаемых множеств Q, Q_1, ..., Q_m, в которой каждое последующее множество порождается предыдущим. Тогда Q_{m-1} должно быть несовместимым множеством, поскольку для любого состояния, покрывающего Q_{m-1}, невозможно приемлемое задание следующего состояния по тому входу, по которому Q_{m-1} порождает Q_m. Просматривая аналогичным образом все остальные множества в обратном порядке, устанавливаем несовместимость множества Q.

(б) Для доказательства *достаточности* предположим, что Q удовлетворяет указанным условиям. Тогда может быть построена таблица B, имеющая состояние s_Q такое, что, если $q_1 \in Q$ и вход I такой, что $Z(q_1, I)$ определено, то $Z(s_Q, I) = Z(q_1, I)$. Пусть теперь R — множество, порождаемое множеством Q по некоторому входу J. Ясно, что R также удовлетворяет исходным условиям. Следовательно, если $N(s_Q, J)$ определено как состояние s_R в таблице B, то значения выходов для s_R могут быть выбраны с учетом значений выходов для состояний множества R, тем же самым образом, как выбирались значения выходов для s_Q с учетом значений выходов для состояний из Q. Аналогично, последователи для s_R определяются точно так же, как они определялись для s_Q. Продолжая этот процесс, получим состояния таблицы B для каждого из множеств, порождаемых непосредственно множеством Q или множествами его последователей. Этот процесс является конечным, поскольку конечно число порождаемых множеств.

Но теперь легко видеть, что для каждой входной последовательности $\mathcal{T}$ и каждого состояния $q \in Q$ значение $Z(q, \mathcal{T})$, когда оно определено, будет равным $Z(s_Q, \mathcal{T})$, следовательно, s_Q покрывает Q. Таким образом, Q — совместимое множество.

Определим простую рекурсивную процедуру нахождения несовместимых пар состояний в таблице переходов и, следовательно, косвенного выделения всех совместимых пар.

Процедура 2.1. *Нахождение несовместимых пар состояний.*

1. Выписать все пары, несовместимые по выходам.

2. Приписать к ним все пары, порождающие уже выписанные пары. Повторять этот процесс до тех пор, пока не перестанут появляться новые пары.

В нашем примере (табл. 2.1) на шаге 1 появляется пара 25. Повторяя многократно операции шага 2, добавляем к списку несовместимых пар сначала пару 13, порождающую пару 25, а затем аналогично находим 15 и 24. Все другие пары состояний, 12, 14, 23, 34, 35 и 45, совместимы. Следует отметить, что отношение совместимости хотя и является рефлексивным (x, очевидно, совместим с x) и симметричным ($a \sim b$ эквивалентно $b \sim a$), но не транзитивно. Так, в нашем примере из $1 \sim 2$ и $2 \sim 3$ никак не следует $1 \sim 3$.

Для большей эффективности в процедуре 2.1 используются так называемые *карты пар*: примером карты пар является треугольная таблица 2.2, *а*. Столбцы и строки

Таблица 2.2

	1	2	3	4	5
23					
23 25	45				
23	15	14			
13	×		12		

а) Карта исходных пар для таблицы 2.1, *а*

	1	2	3	4	5
23					
23× 25×	45				
23	15× ×	14			
13× ×	××		12		

б) Карта финальных пар для таблицы 2.1, *а*

карты пар отмечены символами состояний. Каждой паре состояний однозначно соответствует клетка карты пар. Если данная пара состояний несовместима по выходам, в соответствующую клетку карты пар заносится × (см. пару 25 в таблице 2.2, *а*). В противном случае в ij-клетку заносятся все пары, порождаемые парой ij. Так, напри-

мер, поскольку пара 13 порождает пары 23 и 25 (см. табл. 2.1), они обе заносятся в 13-клетку. Случай, когда пара *ij* порождает пару *ij*, не несет информации и исключается из рассмотрения. Два состояния, которые совместимы по выходам и не порождают других пар, очевидно, совместимы; это обстоятельство отмечается тем, что соответствующая клетка в карте пар остается незаполненной (например, 35-клетка).

Процедура начинается с ×-клеток и продолжается в обратном порядке, в результате чего к списку исходных несовместимых пар добавляются новые пары. Выберем ×-клетку (в нашем примере единственной начальной клеткой является 25-клетка) и запишем знаки × во все клетки, содержащие координаты исходной клетки. В нашем примере (см. табл. 2.2, *б*) знак × необходимо записать в 13-клетку, содержащую пару 25. Затем запишем второй знак × в исходную ×-клетку, чтобы указать на то, что эту клетку не нужно более рассматривать. Повторяем этот процесс до тех пор, пока не останется клеток, содержащих один знак ×. В нашем примере мы должны выбрать клетку 13, записать × в 15, добавить второй знак × в 13 и т. д. Для окончательной проверки необходимо убедиться в том, что каждая пара из клетки, не содержащей ×, порождает пары, указанные в клетках, не содержащих ×. Результатом описанных операций является таблица 2.2, *б*, которая называется *картой финальных пар*. Клетки, содержащие ×, соответствуют несовместимым парам состояний, клетки без × соответствуют совместимым парам. Это весьма эффективный процесс, который можно успешно применять для больших таблиц.

После того как найдены все пары совместимых состояний, желательно найти более крупные множества состояний, которые могут быть покрыты отдельными состояниями некоторой таблицы. В пределе нам хотелось бы найти все множества совместимых состояний. Заметим, что если S является совместимым множеством, то таковым является и любое подмножество из S, поскольку состояние, покрывающее S, покрывает также любое подмножество из S. Определим *максимальное совместимое множество* (сокращенно МС-множество) для данной таблицы переходов как *такое совместимое множество, которое не является подмножеством никакого другого совместимого*

множества. Если задана совокупность МС-множеств, по ней легко находить все другие совместимые множества, являющиеся, очевидно, подмножествами МС-множеств. Таким образом, нашей следующей задачей является построение процедуры нахождения всех МС-множеств по полной совокупности совместных пар состояний. При этом нам потребуется следующая лемма.

Лемма 2.2. *Множество состояний S совместимо тогда и только тогда, когда каждая пара состояний в S совместима.*

Доказательство. (а) Из совместимости каждого подмножества совместимого множества следует совместимость каждой пары состояний этого множества, поскольку всякое состояние, покрывающее совместимое множество, покрывает и всякое подмножество этого множества.

(б) Допустим, что каждая пара состояний в S совместима. Легко видеть, что при этом S должно быть совместимым по выходам. Если R — множество, порождаемое S, то каждая пара состояний из R порождается некоторой парой состояний из S. Следствием леммы 2.1 является то, что каждая пара, порождаемая совместимой парой состояний, будет также совместимой. Следовательно, каждая пара состояний из R будет совместимой, так что R удовлетворяет исходным условиям. Далее, по индукции получаем, что никакое множество, порождаемое последовательно из S, не может быть несовместимым по выходам множеством. Поэтому, согласно лемме 2.1, множество S должно быть совместимым.

Задача нахождения совокупности МС-множеств по заданному множеству совместимых пар возникает и в ряде случаев, отличных от рассматриваемого здесь. Задача 2.15 иллюстрирует подобную ситуацию, другие случаи рассмотрены в главе 3. Ниже рассматриваются два алгоритма нахождения МС-множеств, и вкратце обсуждается третий алгоритм. Первый из них работает непосредственно с парами совместимых состояний, представленными в карте финальных пар, искомые множества строятся последовательно, шаг за шагом.

Процедура 2.2. *Нахождение МС-множеств по совокупности совместимых пар.*

1. Включаем в список совместимых пар (c-список) пары, расположенные в таком первом справа столбце таб-

лицы финальных пар, в котором имеется по крайней мере одна клетка, не содержащая знака $\times$.

2. Движемся влево, столбец за столбцом. Пусть S_i — множество, состоящее из тех состояний, которым соответствуют клетки рассматриваемого столбца i, не имеющие знаков $\times$. Пересекаем S_i с каждым членом текущего c-списка. Если пересечение с некоторым членом содержит более одного состояния, добавляем к текущему списку множество, являющееся объединением состояния i и состояний из пересечения. По окончании операции пересечения для столбца i удаляем из c-списка повторяющиеся члены и те, которые содержатся в других членах. Кроме того, добавляем в c-список пары, состоящие из состояния i и каждого состояния из S_i и не содержащиеся ни в каком из найденных пересечений.

3. Окончательный c-список и одноэлементные множества, включающие состояния, не вошедшие в c-список, образуют искомую совокупность МС-множеств.

В качестве примера рассмотрим карту пар, представленную в таблице 2.3, где незаполненные клетки соответствуют совместимым парам. Проводя рассмотрение справа налево и начав со столбца 8, мы найдем, что множества S_i и c-списки в процессе последовательной обработки столбцов будут иметь следующий вид:

Первый шаг: $c = \{89\}$,

$S_7 = 9$: $c = \{89, 79\}$,

$S_6 = 79$: $c = \{89, 679\}$,

$S_5 = 78$: $c = \{89, 679, 57, 58\}$,

$S_4 = 5679$: $c = \{89, 4679, 457, 58\}$,

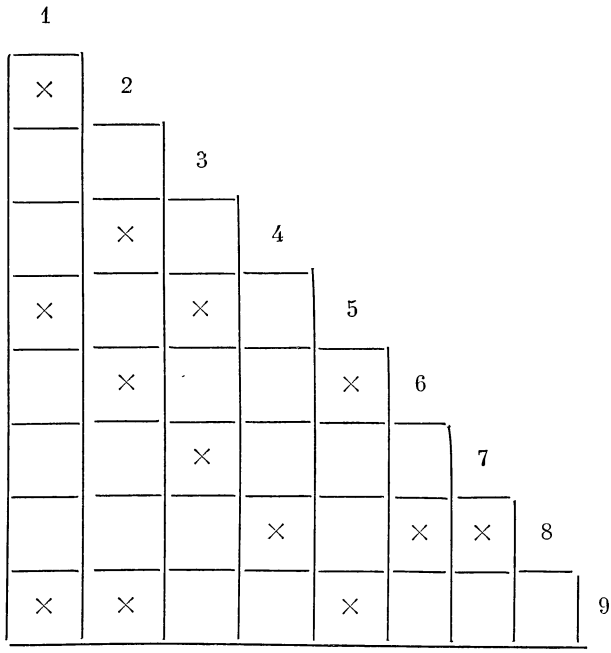
$S_3 = 4689$: $c = \{389, 4679, 3469, 457, 58\}$,

$S_2 = 3578$: $c = \{389, 238, 4679, 3469, 457, 257, 258\}$,

$S_1 = 34678$: $c = \{389, 138, 238, 4679, 1467, 3469, 1346, 457, 257, 358\}$.

Нет необходимости доказывать справедливость этого метода, ибо она очевидна. Следующий метод, примерно такой же по эффективности, использует более тонкий подход. В нем оперируют с несовместимыми парами.

Таблица 2.3



Пусть $\mathcal{K}$ — совокупность совместимых множеств, соответствующая карте финальных пар, представленной в таблице 2.2, *б*, где несовместимыми парами являются 13, 15, 24 и 25. Никакое множество из $\mathcal{K}$ не может одновременно содержать 1 и 3, или 1 и 5, или 2 и 4, или 2 и 5. Множество состояний, отсутствующих в каждом из совместимых множеств из $\mathcal{K}$, которое мы назовем *дополнительным множеством* для $\mathcal{K}$, должно включать 1 или 3, и 1 или 5, и 2 или 4, и 2 или 5. Выражая это условие алгебраически, видим, что дополнительное множество для каждого множества из $\mathcal{K}$ должно удовлетворять следующему булеву выражению:

$$A = (1+3)(1+5)(2+4)(2+5).$$

Таким образом, каждому элементарному произведению *p*

из A соответствует совместимое множество, которое может быть найдено как совокупность элементов, не принадлежащих p, и обратно, каждому совместимому множеству c соответствует элементарное произведение в A, которое может быть получено как совокупность элементов, не принадлежащих c.

Рассмотрим теперь максимальное совместимое множество. Ему соответствует дополнительное множество минимального размера, а поскольку минимальными элементарными произведениями функции являются ее простые импликанты, то отсюда следует, что элементарным произведением в A, соответствующим дополнительному множеству МС-множества, является простой импликант. Обратно, если p — простой импликант из A, то соответствующее совместимое множество максимально, поскольку ничего нельзя добавить в его дополнительное множество. Таким образом, МС-множества являются просто дополнительными множествами простых импликантов из A. Отсюда получаем вторую процедуру для нахождения МС-множеств.

Процедура 2.3. *Нахождение МС-множеств по совокупности несовместимых пар.*

1. Построить булево выражение A в виде произведения сумм, где каждая сумма образована элементами несовместимых пар, при этом каждая несовместимая пара входит в A только один раз.

2. Найти простые импликанты выражения A.

3. Построить МС-множество по каждому импликанту как совокупность состояний, не представленных в этом импликанте.

Применим этот метод для таблицы 2.2, *б*; начнем с выражения A:

$$A = (1+3)(1+5)(2+4)(2+5).$$

После применения дистрибутивного закона получим:

$$A = (1+35)(2+45),$$

а после перемножения:

$$A = 12 + 145 + 235 + 345.$$

Поскольку A — монотонно возрастающая функция (в нее не входят отрицания переменных), то единственной операцией по упрощению полученной суммы произведений

является операция исключения членов (произведений), поглощаемых другими членами (закон поглощения). Так как в нашем выражении для A нет избыточных членов, то отсюда следует, что каждый из членов этого выражения является простым импликантом. (Заметим, что для монотонно возрастающей функции — как и для любой однородной функции — так сокращенная сумма состоит из всех простых импликантов этой функции.) После выполнения шага 3 получим совокупность МС-множеств: 345, 23, 14 и 12.

Для больших таблиц нахождение простых импликантов выражения A может оказаться бессистемным, так что весьма желательно разработать эффективные методы, учитывающие специфику рассматриваемой задачи. Первым очевидным шагом при этом является объединение произведений, полученных после рассмотрения каждого столбца, с помощью дистрибутивного закона. Для таблицы 2.3 получаем

$$A = (1 + 259)\,(2 + 469)\,(3 + 57)\,(4 +)\times \\ \times (5 + 69)\,(6 + 8)\,(7 + 8).$$

При дальнейшем решении этой задачи можно убедиться в высокой вычислительной эффективности процедуры 2.3.

При обработке таблиц вручную автор отдает предпочтение процедуре 2.2, хотя было бы нелегко обосновать такой выбор. Любая из приведенных процедур может быть запрограммирована, и такие программы, действительно, были составлены, так же как и для других, не рассмотренных здесь процедур.

В самом начале рассмотрения задачи минимизации таблицы переходов становится очевидным, что нахождение максимальных совместимых множеств не является наиболее трудной частью задачи.

2.3. Замкнутые совокупности совместимых множеств

После нахождения МС-множеств для данной таблицы переходов может показаться, что единственное, что остается сделать, это выбрать из них подмножества, покрывающие все множество состояний и по совокупности таких под-

множеств построить минимизированную таблицу переходов. Так, для таблицы 2.1, *а* подобный подход может привести к выбору подмножеств 12 и 345. Допустим, мы хотим построить двухстрочную таблицу, покрывающую таблицу 2.1, *а*, такую, в которой строки 1 и 2 покрывают соответственно подмножества 12 и 345 исходной таблицы 2.1, *а*, как показано в таблице 2.4. (Справа от каждой строки новой таблицы выписаны номера строк исходной таблицы, покрываемых этой строкой. Подобное выписывание используется и далее в настоящей главе.)

Таблица 2.4

Попытка использования МС-множеств 12 и 345

	A	*B*	*C*	*D*	
1	?,0	2,1	?,0	1,0	12
2	2,0	?,1	1,1	2,0	345

При определении значений выхода для новых строк не возникает никаких затруднений, поскольку объединенные состояния, очевидно, совместимы по выходам. В столбце *D* следующими состояниями для строк 1 и 2 исходной таблицы будут 2 и «—» соответственно, так что в новой таблице клетка столбца *D* для строки, покрывающей указанные строки 1 и 2 (для строки 1), должна соответствовать строке, покрывающей 2. Таким образом, записываем сюда состояние 1.

Следующими состояниями в столбце *C* для строк 3, 4 и 5 исходной таблицы являются соответственно «—», 2 и 1, следовательно, состояние, стоящее в таблице 2.4 на пересечении строки 2 и столбца *C*, должно соответствовать строке, покрывающей строки 1 и 2 исходной таблицы. Ясно, что таким состоянием является 1.

Аналогичным образом могут быть заполнены клетки $2 — A$, $1 — B$ и $2 — D$ в таблице 2.4. Но как быть с клеткой $1 — A$? Покрываемыми состояниями являются 1 и 2, а следующими состояниями для этих строк в столбце *A* являются состояния 2 и 3 соответственно. Но никакое состояние таблицы 2.4 не покрывает пару состояний 2 и 3, так что невозможно никакое приемлемое заполнение клетки $1 — A$. Аналогичная ситуация возникает и в других местах таблицы 2.4, отмеченных вопросительными знаками; так, в клетке $2 — B$ необходимо покрывать пару состояний 1 и 4.

Отсюда видно, что несмотря на то, что 12 и 345 — совместимые множества, покрывающие все строки исходной таблицы, по ним не может быть построена таблица переходов, покрывающая исходную.

Введем в рассмотрение строки новой таблицы, которые покрывают совместимые (не обязательно максимальные) множества строк исходной таблицы, порождаемые исходными совместимыми множествами; так, 12 порождает 23, а 345 порождает 14. Если в новую таблицу добавлены такие строки, покрывающие порожденные совместимые множества, и если последние не порождают других множеств строк исходной таблицы, не покрываемых полученной расширенной совокупностью строк новой таблицы, то покрывающая таблица может быть заполнена без затруднений. Поскольку множества 12, 23, 14 и 345 образуют полную совокупность МС-множеств, каждая клетка покрывающей таблицы определяется очевидным образом.

Для исходной таблицы 2.1, *а* указанным способом составлена покрывающая таблица 2.5. Мало того, что для

Таблица 2.5

Таблица, покрывающая таблицу 2.1

	A	*B*	*C*	*D*	
1	3,0	2,1	3,0	1/3,0	12
2	2/3,0	4,1	1,1	2,0	345
3	2/3,0	2,1	1/3,0	2,0	23
4	1/3,0	1/4,1	3,—	1/3,0	14

каждой клетки таблицы всегда может быть найдено следующее состояние, удовлетворяющее необходимым условиям, таких состояний может быть несколько. Например, состояние в клетке 2 — *A* должно в новой таблице соответствовать строке, покрывающей строку 3 исходной таблицы. Но ее покрывают также вторая и третья строка новой таблицы, так что оба состояния 2 и 3 годятся для заполнения указанной клетки. Чтобы сохранить возможность подобного выбора, что имеет большое значение в излагаемой ниже процедуре синтеза, элемент, заносимый в клетку 2 — *A*, будем обозначать как 2/3. Аналогичным образом заполнены и другие клетки таблицы 2.5.

В общем случае покрывающая таблица может быть построена лишь по такой совокупности совместимых множеств, в которой:

1. Каждая строка исходной таблицы покрывается хотя бы одним из этих множеств.

2. Совокупность замкнута в том смысле, что любое множество строк, порождаемое всяким множеством из этой совокупности, является подмножеством хотя бы одного множества из этой совокупности.

Совокупность всех МС-множеств всегда удовлетворяет этим условиям, так что число МС-множеств является верхней границей для числа строк минимальной покрывающей таблицы. Однако, поскольку число МС-множеств часто превышает число строк исходной таблицы, такая оценка нередко оказывается бесполезной.

Нет причины ограничиваться при выборе совместимых множеств только максимальными совместимыми множествами. Более, того, можно показать, что в ряде случаев подобное ограничение делает невозможным нахождение решения с минимальным числом строк. Наша задача, таким образом, сводится к нахождению замкнутой совокупности совместимых множеств, покрывающей все состояния и имеющей минимальное число множеств. Такая совокупность будет далее называться *минимальным замкнутым покрытием*. К сожалению, не существует простого и эффективного алгоритма нахождения такого покрытия в общем случае. В следующем разделе рассматриваются решения этой задачи для некоторых важных специальных случаев; здесь же мы рассмотрим некоторые общие принципы и полезные эвристические приемы поиска решений.

При построении минимального замкнутого покрытия производится увеличение числа совместимых множеств с тем, чтобы покрыть не покрытые до этого состояния таблицы переходов или покрыть совместимые множества, порождаемые ранее выбранными множествами. Подобное добавление совместимых множеств может вызвать появление совершенно новых порождаемых множеств, что в свою очередь потребует дальнейшего увеличения числа множеств. Появление новых совместимых множеств, кроме возможного увеличения числа рассматриваемых порождаемых множеств, вызывает также увеличение числа состояний в результирующей таблице. Объединение двух со-

вместимых множеств в одно более крупное множество также может вызвать появление новых порождаемых множеств, но оно уменьшает число состояний результирующей таблицы. Расщепление совместимых множеств дает, естественно, обратный эффект.

Если строку исходной таблицы покрывает только одно некоторое МС-множество и оно не порождает совместимых множеств, не являющихся подмножествами этого МС-множества, то такое МС-множество должно входить в некоторое минимальное замкнутое покрытие. Совместимое множество c_1 всегда может быть заменено на совместимое множество c_2 без опасности потерять минимальное решение, если c_1 включает c_2 и если всякое множество, порождаемое c_1, включается в множество, порождаемое c_2.

Структура порождений одних двухэлементных совместимых множеств другими настолько наглядна, что она может служить превосходной основой при построении минимального решения. Рассмотрим эффективный метод использования этой структуры, и в качестве примера найдем минимальное замкнутое покрытие для табл. 2.1, *а*.

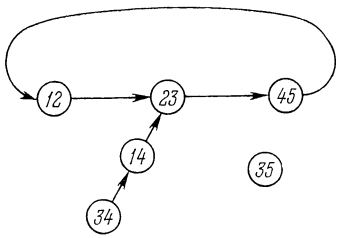


Рис. 2.1. Граф порождений.

После того как найдены все МС-множества (12, 14, 23 и 345) по карте финальных пар состояний для таблицы (см. табл. 2.2, *б*), построим *граф порождений* следующим образом: каждая совместимая пара состояний представляется вершиной, и если пара p_1 порождает пару p_2, то от вершины, представляющей пару p_1, проводится дуга к вершине, представляющей пару p_2 (см. рис. 2.1).

Рассмотрим процедуру отбора пар, входящих в искомое замкнутое покрытие. Поскольку пара 35 не порождает никакой другой пары, она может быть выбрана сразу без всяких опасений. Если прекратить минимизацию на этом шаге и включить состояния 1, 2 и 4 в замкнутое покрытие, то получим решение, состоящее из четырех строк. Чтобы получить «лучшее» покрытие, необходимо выбрать еще какую-либо пару состояний. Но из рисунка видно, что каждая другая пара состояний порождает (в общем

случае, косвенно) три пары 12, 23 и 45. Попробуем поэтому начать наш выбор с этих трех пар. Поскольку эти пары не порождают других пар и покрывают все состояния, они образуют замкнутое покрытие, состоящее из трех множеств. Никакие два из этих множеств не могут быть объединены, так как два из них являются МС-множествами, так что не существует лучшего решения. Соответствующее решение с минимальным числом строк для нашего примера показано в таблице 2.6. Заметим, что множество 45 не является МС-множеством, а множества 12 и 23 перекрываются.

Таблица 2.6

Минимальное по числу строк решение

	A	*B*	*C*	*D*	
1	2,0	3,1	2,0	1/2,0	12
2	2,0	3,1	1/2,0	3,0	23
3	—	1,1	1,1	—	45

В общем случае граф порождений используется для выбора пар с минимальным числом новых порождаемых ими пар и в процессе объединения выбираемых пар (максимальные совместимые множества указывают допустимые пределы подобных объединений), при этом требуется, чтобы очередная выбираемая пара не порождала большого числа не выбранных до этого пар. Идея состоит в том, чтобы начинать с конечных вершин графа (вершин, либо вообще не порождающих других вершин, либо порождающих малое число других вершин) и двигаться по графу против стрелок, когда необходимо. Заметим, что граф порождений часто можно упростить, удалив дуги, параллельные другим путям. Можно построить также графы порождений, вершины которых будут представлять совместимые множества с числом элементов больше двух, но такие графы могут быть весьма громоздкими.

Хотя и МС-множества, и парные порождения являются ценным подспорьем в поиске решения, в ряде случаев они могут не содержать необходимой для получения верного решения информации, а именно,— порождений из числа многоэлементных совместимых множеств. Если,

например, *abc* порождает *def*, то отсюда следует, что пары *ab*, *bc* и *ac* порождают пары *de*, *df* и *ef* (не обязательно в этом порядке), но обратное утверждение не всегда справедливо. Следовательно, после нахождения решения по графу порождений необходимо проверить, чтобы все множества, порождаемые выбранными многоэлементными совместимыми множествами, принадлежали каким-либо из выбранных множеств. Эти условия, очевидно, выполняются, если при построении таблицы переходов используется какое-либо замкнутое покрытие из совместимых множеств. Ясно, что удачное построение таблицы переходов по некоторой совокупности множеств состояний всегда является проверкой того, что данная совокупность действительно является замкнутой совокупностью совместимых множеств. В конце этой главы, в задаче 2.8 подчеркивается роль множеств, порождаемых многоэлементными совместимыми множествами.

Иногда полезной нижней оценкой размера наименьшей замкнутой совокупности совместимых множеств является число состояний в наибольшем максимальном несовместимом множестве. Такие несовместимые множества могут быть найдены по карте пар с применением тех же методов, что и для максимальных совместимых множеств, но для несовместимых пар.

Рассмотрим теперь довольно трудную задачу минимизации числа состояний для таблицы 2.7а, карта финальных пар и МС-множества для которой приведены в таблице 2.7б, а граф порождений показан на рис. 2.2. (После предварительной грубой обработки из него были удалены некоторые дуги, а другие перерисованы таким образом, чтобы представить его структуру в более наглядной форме.) В последующее рассмотрение не включаются все детали процесса поиска решения, анализируются лишь некоторые «тупиковые» пути.

Прежде всего отметим, что состояниями, входящими в совместимые множества, не порождающие никаких множеств, являются 2, 3, 5, 6, 7, 8 и 9. Их лучше всего можно покрыть с помощью четырех совместимых множеств таких, как, например, 9, 23, 56 и 78. Никакие два из этих множеств не могут быть объединены (без появления новых порождаемых множеств), так что лучшее, что можно сделать с использованием совместимых множеств, не

Таблица 2.7а

Таблица переходов, иллюстрирующая трудный случай минимизации

	A	*B*	*C*	*D*
1	2,—	4,—	—	3,—
2	6,—	9,—	—	—
3	—	—	7,—	8,—
4	2,—	1,—	6,—	5,—
5	—	—	—	6,—
6	1,0	—	2,—	—,1
7	5,1	2,—	—	—
8	5,—	—	—	1,0
9	5,—	3,—	—	—

Таблица 2.7б

Карта пар и МС-множества

	1	2	3	4	5	6	7	8	9	МС-множества
1	26 49									45789 4569
2	38									23789
3	35× ×	19× 26×	58× 67×							2369
4	36		68× ×	56						25789
5	12	16	27	12 26						2569
6	24× 25×	29 56		12 25		× ×				1238
7	13 25	56	18	15 25	16	× ×				1236
8	25× 34×	39 56		13 25		15	23			1258 1256
9										

порождающих новых множеств, это — получить решение из шести членов, взяв отсутствующие состояния 1 и 4. Попробуем теперь улучшить это решение путем рассмотрения последовательностей порождений в графе. Начав

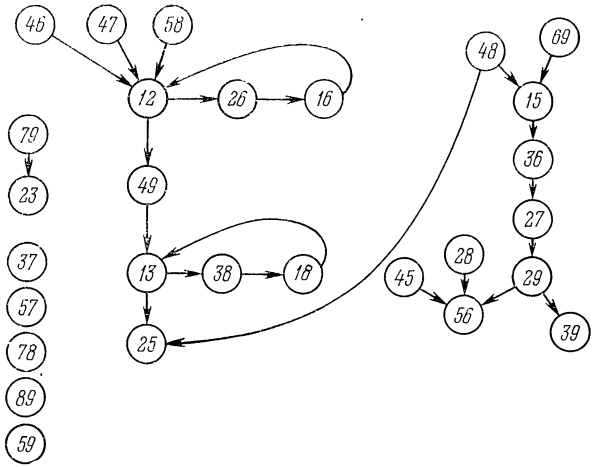


Рис. 2.2. Обработанный граф порождений.

с пары 25 как исходной для одной из таких последовательностей, видим, что на следующем шаге добавляется петля, состоящая из пар 13, 18 и 38; эти три множества не порождают новых пар и могут быть объединены в одно множество 138. Таким образом, другим решением из шести членов является решение {25, 138, 4, 6, 7, 9}. Но сделав еще один шаг в построении рассматриваемой последовательности, получим, что можно объединить 4 и 9, это дает {25, 138, 49, 6, 7} — решение из пяти членов.

Результатом еще одного шага является возможность объединения множеств 12, 26 и 16, при этом элемент 6 в полученном решении может быть заменен множеством 126. В результате такой замены число членов в нашей совокупности {25, 138, 49, 126, 7} не уменьшилось, но эта замена позволяет нам рассматривать пары, которые порождают множества 12, 26 и 16.

Исследуем теперь возможность объединения множеств в последней совокупности. Ряд вариантов, таких, как, на-

пример, объединение 25 и 138, должен быть сразу отброшен из-за отсутствия соответствующего МС-множества. Хотя 1256 есть МС-множество, мы не воспользуемся возможностью объединения 25 и 126, поскольку при этом нужно будет рассмотреть пару 15, а даже из беглого анализа графа видно, что выбор пары 15 потребует последующего выбора многих новых пар. По той же причине (в несколько ослабленной форме) мы откажемся от объединения 25 и 7.

Следующая возможность состоит в образовании 479 из 49 и 7. При этом необходимо рассмотреть пары, порождаемые парами 47 и 79. Пара 47 не порождает новых пар, пара 79 требует добавления одной новой пары, именно, пары 23. Тогда получим {25, 138, 126, 479, 23}. Объединение 23 и 138 требует добавления пары 56, порождаемой парой 28; в результате имеем {25, 126, 479, 1238, 56}. Но 25 и 56 могут быть объединены без всяких осложнений, при этом {126, 479, 1238, 256} — решение из четырех членов. Заметим, что рассмотренные пошаговые преобразования выполнялись таким образом, чтобы на каждом шаге добавлялось минимальное число новых порождаемых пар.

В этом месте процедуры, возможно, после нескольких неудачных попыток уменьшить результирующую совокупность, мы приходим к выводу, что полученная совокупность из четырех членов, похоже, минимальна. И хотя все девять строк исходной таблицы могут быть покрыты единственной совокупностью из двух МС-множеств 1236 и 45789, эти множества не образуют замкнутой совокупности (36, например, порождает пару 27, которой нет в этой совокупности). Таким образом, решения из двух членов не существует, и остается только решить вопрос о существовании решения с тремя членами.

Прежде всего попытаемся найти такое решение путем расщепления одного множества из {1236, 45789} с тем, чтобы избавиться от порождения множеств, отсутствующих в этой совокупности. Однако необходимо расщеплять оба множества, поскольку пара 45 порождает пару 56, которой нет в совокупности, и всякое решение из трех членов должно было бы содержать множества, являющиеся подмножествами трех различных МС-множеств.

Имеется двадцать одна трехчленная совокупность МС-множеств, покрывающих девять состояний, и тем не

менее ни одна из них не позволяет построить замкнутую совокупность, содержащую по одному подмножеству каждого МС-множества. (Расщепление МС-множеств увеличило бы, естественно, число членов в решении до четырех и более.) Не будем утомлять читателя перечислением всех вариантов, а выберем случайно какой-либо из них, например, {4569, 23789, 1238}. Состояния 4 и 6 присутствуют только в первом множестве этой совокупности, так что они оба не могут быть удалены из этого множества. Но по рис. 2.2 видно, что пара 46 косвенным образом порождает пару 16, которой нет ни в одном из множеств нашей совокупности, так что по ней не может быть построено решение с тремя членами.

Таким образом, таблица 2.7а не может быть сжата до таблицы, содержащей менее четырех строк. Одно из возможных минимальных решений, построенных по рассмотренному ранее замкнутому покрытию, приведено в таблице 2.8. Конечно же, могут существовать и другие минимальные решения с четырьмя строками, отличные от таблицы 2.8.

Таблица 2.8

Минимальная таблица, покрывающая таблицу 2.7а

	A	*B*	*C*	*D*	
1	1,0	2,—	1/3/4,—	3,1	126
2	4,1	3,—	1/4,—	4,—	479
3	4,—	2,—	2,—	3,—	1238
4	1,0	2,—	1/3/4,—	1/4,1	256

Как оказалось, рассмотренные методы, несмотря на их эвристическую природу, довольно эффективны при обработке таблиц, содержащих до пятнадцати строк. Было бы интересно проанализировать возможность их реализации на ЦВМ.

В следующих двух разделах рассматривается несколько важных случаев, когда существуют эффективные алгоритмы минимизации. Обсуждаются также условия частичной минимизации.

2.4. Эквивалентность, полностью определенные таблицы, частичные сжатия

Рассмотрим состояние p полностью определенной таблицы переходов. Если q — некоторое состояние той же или какой-либо другой таблицы переходов и q покрывает p, то, очевидно, p также должно покрывать q. Это следует из того, что для каждой входной последовательности $\mathcal{T}$ значение $Z(p, \mathcal{T})$ определено, причем $Z(p, \mathcal{T})$ должно быть равно $Z(q, \mathcal{T})$. Таким образом, в полностью определенной таблице отношение покрываемости приобретает свойство симметричности и становится, следовательно, отношением эквивалентности (в разделе 2.1 было отмечено, что отношение покрываемости всегда рефлексивно и транзитивно). Отсюда также следует, что если полностью определенная таблица T_1 покрывается таблицей T_2, то T_2 должна быть также полностью определенной (в предположении, что в ней нет таких состояний, которые недостижимы из состояний, покрывающих T_1), причем T_1 покрывает T_2. Таким образом, T_1 и T_2 эквивалентны согласно определению, данному в разделе 2.1.

Предположим теперь, что p и r — совместимые состояния полностью определенной таблицы переходов. Тогда некоторое состояние q покрывает и p, и r. Следовательно, p и r эквивалентны по отношению к q, так что они эквивалентны и по отношению друг к другу. Таким образом, в полностью определенной таблице отношение совместимости состояний становится отношением эквивалентности. Это означает, что все множество состояний такой таблицы можно разбить на взаимно непересекающиеся подмножества, все состояния каждого из которых эквивалентны. Эти множества не могут быть ничем другим, кроме как МС-множествами, образующими, очевидно, единственное минимальное замкнутое покрытие. Отсюда непосредственно приходим к решению с минимальным числом строк, которое единственно с точностью до тривиального переобозначения строк.

Далее, само нахождение МС-множеств по исходным совместимым парам при этом чрезвычайно просто и этот процесс мы представим в виде следующей процедуры.

Процедура 2.4. *Минимизация полностью определенных таблиц переходов.*

1. Найти двухэлементные совместимые множества, как и в процедуре 2.1.

2. Найти МС-множества следующим образом:

2.1. В качестве первого МС-множества выписать множество, содержащее строку 1 и все совместимые со строкой 1 состояния, какие указаны в клетках столбца 1 карты финальных пар.

2.2. После нахождения первых j — 1 МС-множеств, в качестве j-го МС-множества выбирается множество, содержащее состояние с наименьшим номером k, еще не включенное в список выбранных состояний, а также все совместимые с состоянием k состояния, указанные в клетках столбца k карты финальных пар. Повторяем эту процедуру, пока все состояния не будут входить в какое-либо из МС-множеств. Может случиться, что в качестве МС-множеств будут выбраны одноэлементные множества, содержащие состояния, несовместимые ни с какими другими.

Таблица 2.9

	x = 0	x = 1
1	3,0	2,1
2	1,1	6,0
3	5,0	2,1
4	5,1	6,0
5	5,0	4,1
6	2,0	1,1

а) Полностью определенная таблица

1	2	3	4	5	6
× ×					
35	× ×				
× ×	15	× ×			
35 24	× ×	24	× ×		
12× 23×	× ×	12× 25×	× ×	14× 25×	

б) Карта финальных пар

	x = 0	x = 1	
1	1,0	2,1	135
2	1,1	3,0	24
3	2,0	1,1	6

в) Минимизированная таблица

3. По замкнутому покрытию, содержащему полученные МС-множества, построить минимизированную таблицу переходов, как описано в разделе 2.3.

В качестве примера применения этой процедуры рассмотрим таблицу 2.9, *а* — полностью определенную таблицу, карта финальных пар которой приведена в таблице 2.9, *б*. МС-множества (выписанные в порядке нахождения) суть 135, 24, 6, а соответствующей ей таблицей с минимальным числом строк является таблица 2.9, *в*.

Во многих случаях при минимизации неполностью определенной таблицы переходов возможно наличие таких пар строк, которые совместимы и не порождают никаких других пар, или порождают такие пары, совместимость которых очевидна. В таких случаях можно попытаться осуществить некоторое предварительное объединение строк в надежде уменьшить общий объем необходимых вычислений, а потом уже применять общую процедуру

Таблица 2.10

	A	*B*	*C*
1	1,0	2,—	5,0
2	2,0	3,—	4,0
3	2,0	1,—	4,0
4	—	2,—	7,1
5	—	1,—	7,1
6	1,—	2,—	5,0
7	1,1	2,1	8,0
8	6,0	3,0	6,1

а) Минимизируемая таблица

	A	*B*	*C*	
1	1,0	1,—	2,0	1236
2	—	1,—	3,1	45
3	1,1	1,1	4,0	7
4	1,0	1,0	1,1	8

б) Неудачный вариант предварительного объединения строк

	A	*B*	*C*	
1	1,0	1,—	2,0	123
2	—	1,—	4,1	45
3	1,—	1,—	2,0	6
4	1,1	1,1	5,0	7
5	3,0	1,0	3,1	8

в) Удачный вариант предварительного объединения строк

	A	*B*	*C*	
1	1,0	1,—	2,0	1
2	3,0	1,0	3,1	25
3	1,1	1,1	2,0	34

г) Минимальная таблица, покрывающая таблицы 2.10, *а* и *в*

минимизации числа строк. Например, анализ таблицы 2.10, *а* показывает, что можно сразу же объединить множества строк 1236 и 45, поскольку каждое из этих

множеств порождает лишь их подмножества. Это приводит к таблице 2.10, *б*, которая не может быть минимизирована далее, поскольку каждая пара ее строк — несовместима. Менее решительным предварительным шагом является объединение строк 123 и 45 таблицы 2.10, *а*, приводящее к таблице 2.10, *в*. По описанной ранее в настоящей главе процедуре преобразуем ее в таблицу 2,10, *г*, содержащую меньше строк, чем таблица 2.10, *б*. Видим, что неудачный выбор варианта предварительного объединения строк приводит к «тупиковой» таблице, которая не может быть минимизирована далее. (Заметим, что, если объединить строки 1 и 3 таблицы 2.10, *в* — а это довольно «очевидное» объединение,— то придем к той же «тупиковой» таблице.)

Дело здесь в том, что, поскольку покрываемость не является, в общем случае, симметричным отношением, можно найти такую таблицу B, которая покрывает таблицу A, и таблицу D, которая покрывает таблицу A, но не покрывает таблицу B. Если B — результат предварительного объединения строк, а D — минимальная таблица, покрывающая A, то может случиться так, что ни D, ни какая-либо другая таблица с минимальным числом строк, покрывающая A, не покрывают B, так что предварительное сжатие таблицы делает невозможным нахождение желаемого решения. Если, однако, предварительное сжатие таково, что результирующая таблица эквивалентна исходной таблице, эта опасность исчезает. Таким образом, если предварительное объединение строк приводит к таблице C, которая эквивалентна A, то всякая таблица, покрывающая A, покрывает и C, так что всякая минимальная таблица, покрывающая C, будет также минимальной таблицей, покрывающей A. Покажем, как провести предварительное объединение строк, чтобы получить такую таблицу C.

Определим два состояния как *I-эквивалентные,* если (1) они совместимы по выходам и (2) нет ни одного столбца, в котором следующее состояние или выходной сигнал одного из них не определены, тогда как для другого определены, и (3) они не порождают ни одной пары не I-эквивалентных состояний, т. е. состояний, для которых не выполняется (1), (2).

Для полностью определенных таблиц это определение сводится к обычному определению эквивалентности, рас-

смотренному выше. Если применить это определение к неполностью определенным таблицам, потребуется рассматривать прочерки в таблице переходов как некоторые специфические выходные сигналы или следующие состояния; при этом нетрудно показать, что два состояния I-эквивалентны тогда и только тогда, когда они покрывают друг друга. Если таблица C получается из таблицы A путем объединения I-эквивалентных состояний (как, например, при построении таблицы 2.10, *в* по таблице 2.10, *а*), то C и A будут эквивалентны. Поскольку нахождение минимального по числу строк эквивалента данной таблицы намного проще, чем нахождение ее минимального покрытия (если вместо совместимых рассматривать I-эквивалентные состояния, то вполне применима процедура 2.4), то может оказаться выгодным с точки зрения вычислительной сложности применение двухэтапной процедуры, изложенной при построении таблицы 2.10. Обратимый характер процесса сжатия заданной таблицы до ей эквивалентной, в противоположность в общем случае необратимому характеру процесса сжатия до покрывающих таблиц, в некоторых случаях может иметь большое значение.

2.5. Ограничения на входные последовательности

Причиной появления незаполненных клеток в таблицах переходов являются ограничения на вид допустимых входных последовательностей. В частности, когда такие клетки появляются в результате того, что известные определенные пары входных сигналов не могут быть смежными в последовательностях их подачи, задача минимизации существенно упрощается в том, что все совокупности МС-множеств, которые покрывают каждую строку, являются замкнутыми.

Чтобы сделать изложение более строгим, необходимо ввести ряд определений.

Упорядоченная пара входных символов $\mathcal{F}$ является *запрещенной парой* для некоторой таблицы переходов, если для всех состояний s значения $N(s, \mathcal{F})$ и $Z(s, \mathcal{F})$ не определены. Таблица переходов называется *таблицей с ограничениями на пары*, если для каждого состояния s значения $N(s, I_1I_2)$ и $Z(s, I_1I_2)$ определены лишь при

невыполнении следующих условий: (а) I_1I_2 — запрещенная пара, или (б) имеется вход I_0 такой, что I_0I_1 — запрещенная пара, а для состояния t при этом $N(t, I_0) = s$. Если выполняется или (а) или (б), то и $Z(s, I_1I_2)$ и $N(s, I_1I_2)$ не определены.

Грубо говоря, таблица с ограничениями на пары такова, что все ее незаполненные клетки объясняются наличием одной или нескольких неразрешенных входных последовательностей длины два. В качестве примера таблицы с ограничениями на пары рассмотрим таблицу 2.11, *а*.

Таблица 2.11

	A	*B*	*C*	*D*
1	6,0	1,0	2,0	4,0
2	—	6,0	2,0	4,0
3	—	1,0	3,0	4,0
4	5,1	—	2,0	4,0
5	5,1	5,0	3,0	4,0
6	1,0	6,0	3,0	4,0

а) Таблица переходов с ограничениями на пары

МС-множества	Состояния 1	2	3	4	5	6
45				1	1	
234		1	1	1		
1236	1	1	1			1

б) Таблица покрытий

Независимо от начального состояния подача последовательностей CA или DB приводит к тому, что результирующие выходной сигнал и состояние не определены. Более того, никакое полное состояние, для которого не определены ни следующее состояние, ни выходной сигнал, не может быть достигнуто подачей входной последовательности длины два, отличной от CA или DB. Это легко проверить.

Покажем, что при определенных условиях совместимость в таблице с ограничениями на пары приобретает свойство транзитивности.

Лемма 2.3. *Если таблица переходов является таблицей с ограничениями на пары, причем существуют вход* I_0 *и состояния* i, j, k, p *и* q *такие, что* $i \sim j$, $j \sim k$, $N(p, I_0) = i$, $N(q, I_0) = j$, *то* $i \sim k$.

Доказательство (от противного). Допустим $i \not\sim k$. Тогда для некоторой входной последовательности $\mathscr{L}$ $Z(i, \mathscr{L})$ и $Z(k, \mathscr{L})$ определены и не равны. Пусть $I_0\mathscr{L}$ — последовательность, начинающаяся с символа I_0,

после которого следует $\mathscr{L}$. Тогда, поскольку $N(p, I_0) = i$, получаем, что $Z(p, I_0\mathscr{L})$ определено, так что $I_0\mathscr{L}$ не содержит запрещенной пары. Покажем теперь, что выражение $Z(q, I_0\mathscr{L}) = Z(j, \mathscr{L})$ определено. Из ограничения на пары следует, что поскольку $I_0\mathscr{L}$ не содержит запрещенной пары, единственным условием, когда $Z(q, I_0\mathscr{L})$ могло бы быть неопределенным, является наличие состояния r и входа I таких, что пара $I I_0$ является запрещенной и $N(r, I) = q$. Но тогда $N(q, I_0)$ было бы не определено, чего не может быть по условию. Таким образом, значение $Z(q, I_0\mathscr{L}) = Z(j, \mathscr{L})$ должно быть определено. Так как $j \sim k$, то $Z(j, \mathscr{L}) = Z(k, \mathscr{L})$, и, поскольку последовательность выбрана такой, что $Z(i, \mathscr{L}) \neq Z(k, \mathscr{L})$, отсюда следует $Z(i, \mathscr{L}) \neq Z(j, \mathscr{L})$, что противоречит условию леммы; это противоречие завершает доказательство.

Теперь можно доказать следующую теорему.

Теорема 2.1. *Для любой таблицы переходов с ограничениями на пары каждая совокупность МС-множеств, покрывающих все состояния таблицы, является замкнутой.*

Доказательство. Допустим, что C — совместимое множество, порождаемое некоторым множеством из этой совокупности. Пусть c_1 — некоторое состояние в C, а M — МС-множество из этой совокупности такое, что c_1 принадлежит M. Покажем, что каждое состояние из C должно принадлежать M. Пусть c_i — произвольное состояние в C, а m — произвольное состояние в M. Тогда, очевидно, $c_i \sim c_1$ и $c_1 \sim m$. Поскольку C порождается некоторым множеством состояний исходной таблицы, то c_1c_i должно порождаться некоторым подмножеством этого множества, скажем, pq; значит, для некоторого входа I_0 $N(p, I_0) = c_1$ и $N(q, I_0) = c_i$. Но тогда выполняются условия леммы 2.3, откуда следует $c_i \sim m$. Поскольку m выбрано произвольным, получаем, что c_i совместимо с каждым элементом из M; следовательно, по определению МС-множества c_i должно быть состоянием из множества M. Далее, поскольку c_i — произвольное состояние из C, то каждое состояние из C должно принадлежать M. Следовательно, C является подмножеством множества M, а так как это справедливо для всякого совместимого множества, порождаемого каким-либо множеством исходной совокупности МС-множеств, то такая совокупность является замкнутой, что и требовалось доказать.

Эту теорему можно использовать следующим образом: дана таблица переходов с ограничениями на пары — найти по ней все МС-множества (используя процедуры раздела 2.2) и выбрать из них наименьшую совокупность МС-множеств, включающую все состояния. Это та же задача покрытия, которая возникает при минимизации булевых функций и которая может быть представлена в форме таблицы покрытий, называемой обычно *таблицей простых импликантов*. В литературе можно найти ряд методов решения этой задачи (см., например, [74] и [29]).

Теорема 2.1 гарантирует, что результирующая совокупность замкнута и может быть, следовательно, выбрана для построения таблицы, покрывающей исходную таблицу переходов. При этом могут быть, конечно, другие покрытия с тем же числом МС-множеств, но не существует покрытия с меньшим числом последних. Это легко показать. Пусть $\{c_1, c_2, \ldots, c_n\}$ — произвольное замкнутое покрытие нашей таблицы. Каждое c_i содержится по крайней мере в одном МС-множестве, которое обозначим через M_i. Тогда замкнутая совокупность $\{M_1, M_2, \ldots, M_n\}$ содержит не больше МС-множеств, чем исходная совокупность (а возможно, и меньше, если некоторое МС-множество содержит два различных члена c_i и c_j), и может быть построена по нашей процедуре.

Читатель может убедиться, что МС-множествами таблицы 2.11, *а* являются {45, 234, 1236}. Построенная по ним таблица покрытий приведена в таблице 2.11, *б*; здесь единицы поставлены в тех местах, в которых состояние, соответствующее столбцу, входит в МС-множество, соответствующее строке. Задача состоит в выборе минимальной совокупности строк, такой, что для каждого столбца имеется по крайней мере одна строка этой совокупности, содержащая единицу в этом столбце. Такой совокупностью является, очевидно, {1236, 45}, причем согласно теореме она будет замкнутой, по ней строится минимизированная таблица 2.12. (Интересно отметить, что к таблице 2.11, *а* может быть при-

Таблица 2.12

Минимизированная форма таблицы 2.11

	A	*B*	*C*	*D*	
1	1,0	1,0	1,0	2,0	1236
2	2,1	2,0	1,0	2,0	45

менен метод предварительного объединения строк, рассмотренный в разделе 2.4.)

Ограничения на входные последовательности могут быть выражены не только с помощью неполностью определенных таблиц переходов. Можно рассматривать полностью определенную таблицу переходов и требовать при этом, чтобы некоторые подпоследовательности не входили ни в какую из допустимых входных последовательностей.

Обсудим кратко эту ситуацию при помощи примера.

Таблица 2.13

	A	*B*	*C*
1	3,0	4,0	2,0
2	4,0	4,0	2,0
3	3,1	1,0	1,0
4	3,1	4,0	1,0

а) Исходная таблица

	A	*B*	*C*	
1	2/3,0	3,0	1,0	1278
2	2/3,1	1,0	1,0	356
3	2/3,1	3,0	1,0	45678

в) Минимизированная таблица

	A	*B*	*C*	
1	5,0	4,0	2,0	1
2	6,0	4,0	2,0	2
3	5,1	1,0	1,0	3
4	5,1	4,0	1,0	4
5	5,1	7,0	1,0	3(*A*)
6	5,1	8,0	1,0	4(*A*)
7	—	4,0	2,0	1(*AB*)
8	—	4,0	1,0	4(*AB*)

б) Расширенная таблица с запрещенной последовательностью *ABA*

Пусть дана таблица 2.13, *а* с тем ограничением, что *ABA* не является частью никакой входной последовательности, и требуется найти минимальную таблицу для исходной таблицы и всех допустимых входов. Прежде всего отметим, что заданная таблица в представленном виде является несжимаемой. Имеется несколько подходов к решению поставленной задачи, но мы рассмотрим лишь один, а именно — расширение заданной таблицы до такой, в которой незаполненные клетки связаны с ограничениями на входы, с последующей минимизацией полученной таблицы с помощью стандартных процедур.

В нашем примере расширенная таблица представлена в виде таблицы 2.13, *б*, которая была получена следующим образом. Первые четыре строки соответствуют строкам исходной таблицы (соответствующие номера строк последней выписаны справа) с теми же значениями выходов и следующих состояний во всех столбцах (B и C), кроме тех (столбец A), которые соответствуют начальным символам запрещенных входных последовательностей. В таких столбцах следующие состояния соответствуют новым строкам, для которых справа выписаны соответствующие номера строк исходной таблицы и начальные символы запрещенной входной последовательности, подача которой приводит к этой строке. Так, если A прикладывается к системе в состоянии 1, следующим состоянием расширенной таблицы будет состояние 5, которое соответствует состоянию 3 исходной таблицы и помечено справа как $3(A)$, чтобы показать, что оно может быть достигнуто только после подачи входа A — начального символа запрещенной последовательности ABA. Аналогичным образом новая строка 6 связана со строкой 4 исходной таблицы. Значения выходов в добавленных строках те же, что и в соответствующих строках исходной таблицы.

Следующие состояния в столбцах, требующих специального рассмотрения, определяются аналогично тому, как было описано выше. Дополнительно надо иметь в виду, что в клетке строки, помеченной справа начальным отрезком x запрещенной входной последовательности, и столбца I, где xI — также начальный отрезок этой последовательности и I не является ее последним символом, следующим состоянием является номер строки, имеющей справа пометку xI. Таким образом, следующим состоянием для полного состояния $5—B$ является состояние 7, которое помечено $1(AB)$, чтобы показать, что оно соответствует строке 1 исходной таблицы и может быть достигнуто только по входной последовательности AB. Если I является последним символом запрещенной входной последовательности xI, то в клетке строки, имеющей справа пометку x, и столбца I ставится прочерк. Так, полное состояние $7—A$, достижимое только по запрещенной последовательности ABA, имеет незаполненную клетку.

Как только расширенная таблица построена, она может быть приведена к минимальной форме с помощью

описанных выше процедур. (В нашем примере такой формой является таблица 2.13, *а*.) Чтобы учесть при этом свойства исходной таблицы, необходимо всегда начинать с какого-либо состояния, покрывающего одно из состояний исходной таблицы (в нашем примере с состояния, покрывающего одно из первых четырех состояний таблицы 2.13, *б*); при этом значения получаемых выходных сигналов будут совпадать со значениями соответствующих выходных сигналов исходной таблицы для всех допустимых входных последовательностей. Если бы мы, например, начали с состояния 5, для допустимой входной последовательности *BA* поведение системы было бы не определено.

Расширенная таблица, построенная для случая, когда все запрещенные последовательности имеют длину 2, является всегда таблицей с ограничениями на пары, так что в этом случае применимы методы предыдущего раздела. Можно было бы определение ограничений на пары распространить на больший класс расширенных таблиц, однако неизвестно, приведет ли это к упрощению задачи минимизации таких таблиц.

2.6. Асинхронные таблицы переходов

Предметом настоящего раздела является применение рассмотренных идей к минимизации таблиц переходов асинхронных функций. Будет показано, что для функций с однократным изменением выхода (ОИВ-функций *)) потребуются лишь небольшие изменения рассмотренных методов, тогда как для функций с многократным изменением выхода (МИВ-функций) ситуация значительно усложняется.

Имеются две основные причины, по которым асинхронные функции нельзя рассматривать так же, как импульсные или синхронные функции.

1. Предположение о том, что состояние входа может меняться лишь тогда, когда для системы выполняются условия, соответствующие устойчивым состояниям таблицы переходов, или, в случае НИВ-функций, циклической смене состояний;

*) Определения ОИВ-, МИВ- и НИВ-функций см. на стр. 43—44.

2. Тот факт, что при анализе выходной последовательности мы обычно учитываем лишь последовательность изменений состояний выходов безотносительно в какой-либо шкале времени, так что нет смысла говорить о том, что то же самое состояние выхода появляется повторно в соседние моменты времени (исключения из этого правила будут рассмотрены далее).

Мы начнем со случая ОИВ-функций как довольно простого, более важного и хорошо понимаемого. Функцию этого типа описывает, например, таблица 2.14, *а*, и если к ней применить методы раздела 2.2, нетрудно убедиться, что никакая пара строк не является совместимой, так что таблица уже представлена в своей минимальной форме. Однако несовместимость некоторых пар обусловлена тем фактом, что выход в неустойчивом состоянии 1—*B* принимается отличным от выхода в последующем устойчивом состоянии 2—*B*. Но в последовательности изменений выхода ничего бы не изменилось, если $Z(1, B)$ сделать равным $Z(2, B)$, и если это действительно сделать, то состояния 1 и 2 станут эквивалентными и таблица может быть сжата до таблицы 2.14, *б*. В общем случае ясно, что для

Таблица 2.14

	A	*B*	*C*
1	**1**,0	2,0	3,0
2	**2**,0	***2***,1	3,0
3	1,0	***3***,0	***3***,0

а) Таблица с задерживаемыми изменениями выхода

	A	*B*	*C*	
1	***1***,0	***1***,1	2,0	12
2	1,0	**2**,0	**2**,0	3

б) Минимизированная форма таблицы

функций рассматриваемого класса последовательность изменений выходов никогда не меняется, если положить значения выходов в неустойчивых состояниях равными их значениям в соответствующих следующих состояниях. Более того, ясно также, что хотя при этом несовместимые строки и могут стать совместимыми, обратное никогда не имеет места.

Следующий важный момент иллюстрируется таблицей 2.15, *а*, которая также не имеет совместимых пар. В частности, строки 1 и 2 несовместимы потому, что по входу *B* пара состояний 12 порождает пару 23. Это в свою оче-

редь обусловлено тем, что $N(1, B) = 3$, и роль состояния 3 состоит в обеспечении двухшагового перехода от $1—B$ к $2 — B$ (через $3 — B$) без изменений выхода. Если же $N(1, B)$ заменить на состояние 2, устранив таким образом избыточный шаг, функция фактически не изменится, но строки 1 и 2 станут совместимыми, допуская сжатие рассматриваемой таблицы до таблицы 2.15, *б*.

Таблица 2.15

	A	*B*	*C*
1	***1***,0	3,0	***1***,0
2	***2***,0	***2***,0	1,0
3	***3***,1	2,0	1,0

а) Таблица с избыточным переходом

	A	*B*	*C*	
1	***1***,0	***1***,0	***1***,0	12
2	***2***,1	1,0	1,0	3

б) Минимизированная форма таблицы а)

Путем изменения следующих состояний и значений выходов таблицы переходов для неустойчивых состояний там, где это необходимо, таблица переходов для асинхронной ОИВ-функции может быть приведена к *стандартной форме* по следующей процедуре.

Процедура 2.5. *Преобразование таблиц переходов с однократным изменением выхода (ОИВ-таблиц) в стандартную форму.*

Если u — состояние, которое неустойчиво в столбце I, а s — устойчивое состояние, достижимое из u при входе, неизменном и равном I, то: (1) положить $N(u, I) = s$ и (2) положить $Z(u, I) = Z(s, I)$.

Говорят, что таблица переходов T для асинхронной ОИВ-функции (ОИВ-таблица) *A-покрывается* таблицей V, если для каждого устойчивого состояния p_t в T найдется такое устойчивое состояние p_v в V, что всякий раз, когда одна и та же входная последовательность прикладывается к обеим таблицам с начальными состояниями p_t и p_v соответственно, выходная последовательность, получаемая по V, эквивалентна по существу выходной последовательности, получаемой по T, везде, где последняя последовательность определена. (Выходные последовательности для функций этого типа эквивалентны по существу, если они представляют одну и ту же последовательность изменений

выходов. Так, например, все последовательности 0101, 01101, 01001, 01—001 и 0—10—1 эквивалентны по существу.) Нетрудно видеть, что стандартная форма таблицы переходов A-покрывает исходную таблицу, обратное также справедливо.

Для ОИВ-таблиц задача минимизации числа состояний сводится к нахождению минимальной таблицы, A-покрывающей исходную. Покажем, что решение при этом можно найти, применив описанные ранее процедуры к стандартной форме исходной таблицы.

Теорема 2.2. *Пусть T — произвольная ОИВ-таблица, T^* — стандартная форма для T, а T_m — минимальное покрытие для T^*. Тогда T_m — минимальное A-покрытие для T.*

Доказательство. Поскольку покрываемость влечет A-покрываемость, T_m A-покрывает T^*. Но T^* A-покрывает T, и тогда в силу транзитивности T_m A-покрывает T. Теперь, чтобы доказать минимальность T_m, допустим, что T_A — некоторая таблица, A-покрывающая T, причем T_A^* — стандартная форма таблицы T_A. Тогда T_A^* A-покрывает T, которая, в свою очередь, A-покрывает T^* (любая таблица A-покрывает свою стандартную форму). Следовательно, T_A^* A-покрывает T^*. Но если стандартная форма какой-либо таблицы A-покрывает стандартную форму другой таблицы, то первая из них покрывает (в обычном смысле) другую. Следовательно, T_A^* покрывает T^*. Тогда в T_A^* во всяком случае столько же строк, сколько и в T_m (минимальном покрытии для T^*), а поскольку в T_A не меньше строк, чем в T_A^*, то в T_A во всяком случае столько же строк, сколько и в T_m; это завершает доказательство того, что T_m — минимальное A-покрытие для T.

Связанный с этим другой результат состоит в том, что, если в произвольной ОИВ-таблице переходов для каждого неустойчивого состояния s — I $N(s, I)$ соответствует устойчивому состоянию в столбце I, причем $Z(s, I)$ не определен, а все остальные клетки определены, то минимальная таблица, покрывающая такую таблицу, может быть получена минимизацией таблицы, в которой выходы $Z(s, I)$ доопределены таким образом, чтобы удовлетворить определению стандартной формы таблицы. Доказательство этого утверждения использует тот факт, что совместимые пары

и порождаемые множества состояний в получаемой полностью определенной таблице те же, что и в исходной таблице.

Переходя к таблицам переходов для функций с многократным изменением выхода (МИВ-таблицы), будем считать, что система начинает функционировать из полного состояния, которое является устойчивым. Таким образом, если в таблице переходов имеется строка, не содержащая устойчивых состояний, ее следующие состояния и выходы могут быть лишь промежуточными состояниями и выходами в многошаговых переходах и достижимы только в результате переходов между различными столбцами. Следствием такого допущения является то, что такие неустойчивые строки не нужно покрывать строками покрывающих таблиц переходов.

Здесь рассматриваются две основные интерпретации МИВ-таблиц переходов. В первой из них, которая предусматривает независимость от времени (и гораздо ближе по духу к предыдущему рассмотрению), существенной представляется лишь последовательность, в которой появляются различные состояния выхода. В другой интерпретации, учитывающей зависимость от времени, считается, что продолжительность переходных состояний выхода определяется числом имеющихся в таблице переходов по внутренним

Таблица 2.16

	x	
	0	1
1	*1*,0	2,0
2	1,1	3,1
3	2,1	4,1
4	3,0	*4*,0

а) МИВ-таблица переходов

	x		
	0	1	
1	*1*,0	2,0	1
2	1,1	3,1	23
3	2,0	*3*,0	4

б) Таблица а), минимизированная без учета временных зависимостей

состояниям, в течение которых сохраняется неизменным то или иное состояние выхода. Так, в случае таблицы 2.16, *а*, в этой интерпретации считается, что после того, как $x = 1$ подан на систему, находящуюся в исходном состоянии $1 - 0$, z остается равным нулю в течение

единичного интервала времени (единичный интервал соответствует времени, в течение которого происходит смена внутреннего состояния), затем становится равным 1 в течение двух единичных интервалов времени (пока система находится в состояниях 2 и 3), а затем снова принимает значение 0. В интерпретации, не связанной с рассмотрением времени, единственное, что учитывается при таком переходе, это то, что значение z изменилось в 1, а потом снова вернулось в 0.

Если использовать общее определение совместимости, то таблица 2.16, *а* не имеет совместимых пар, и, следовательно, несжимаема. Однако в интерпретации, не учитывающей временных соотношений, таблица 2.16, *б*, очевидно, эквивалентна таблице 2.16, *а*. Но это несправедливо при учете временных соотношений, поскольку единичные значения выхода в таблице 2.16, *б* появляются лишь в течение одного единичного интервала времени. Две таблицы считались бы эквивалентными, если бы они имели одинаковые продолжительности соответствующих состояний выхода. Если при учете временных зависимостей одна таблица покрывает другую, то она покрывает ее и без учета таких зависимостей, обратное же не всегда верно.

Рассмотрим сначала случай игнорирования временных зависимостей. В таблице 2.17, *а* строки 1 и 4 несовместимы, поскольку они порождают 25 по входу B, а эта пара несовместима по выходам при входе A. Основанием для такого утверждения является то, что, приложив в состояниях 1 и 4 входную последовательность BA, получим две различные выходные последовательности, именно 01 и 00. Но это так лишь для синхронного случая и совсем не так, если рассматривать эту таблицу как задание некоторой асинхронной функции. Вход B должен быть включен довольно долго, чтобы система могла достигнуть следующего устойчивого состояния, которым в нашем примере является 3 для обоих исходных состояний. Выходная последовательность, генерируемая при этом в обоих случаях, равна 010. Для того чтобы учесть это обстоятельство, необходимо так модифицировать таблицу переходов, чтобы показать, что промежуточное состояние, следующее за $1 - B$, является таким, в котором следующим входом может быть только B (аналогично для состояния, следующего за $4 - B$). Такая модификация (табл. 2.17, *б*) может быть сде-

Таблица 2.17

	A	*B*	*C*
1	***1***,0	2,0	2,0
2	***2***,1	3,1	***2***,0
3	1,0	***3***,0	5,0
4	***4***,0	5,0	2,0
5	4,0	3,1	***5***,0

а) Исходная таблица

	A	*B*	*C*
1	***1***,0	6,0	2,0
2	***2***,1	3,1	***2***,0
3	1,0	***3***,0	5,0
4	***4***,0	7,0	2,0
5	4,0	3,1	***5***,0
6	—	3,1	—
7	—	3,1	—

б) Расширение исходной таблицы

	A	*B*	*C*	
1	***1***,0	2,0	2,0	14
2	***2***,1	3,1	***2***,0	267
3	1,0	***3***,0	4,0	3
4	1,0	3,1	***4***,0	5

в) Минимизированная таблица

лана путем расширения таблицы — введением в нее новых состояний 6 и 7, для которых определены следующие состояния только в столбце B. Модифицированная таблица имеет привычную для нас структуру, где появление незаполненных клеток обусловлено ограничениями, связанными со способом работы системы, и может быть, очевидно, минимизирована с помощью основной процедуры, в результате которой получается таблица 2.17, *в*. В общем случае, если в исходной таблице $N(s, I) = t$, где $t - I$ неустойчиво, то в расширенной таблице вводится новое состояние t', с незаполненными элементами во всех столбцах, кроме столбца I, причем $Z(t', I) = Z(t, I)$, $N(t', I) = N(t, I)$. Заменяем также $N(s, I)$ на t'. Если $N(t, I)$ неустойчиво в столбце I, необходимо добавить другую новую строку и т. д., пока всякое неустойчивое состояние не будет переводиться непосредственно в устойчивое состояние или в такое полное состояние, которое является единственным состоянием с определенными следующим состоянием и выходом в содержащей его строке. При сделанных допущениях (о том, в частности, что система начинает

функционировать из устойчивого состояния) результирующая таблица покрывает исходную.

Учет временных соотношений оправдан в тех случаях, когда желательно найти грубые оценки абсолютной или относительной продолжительности выходных импульсов, генерируемых асинхронной системой. При реализации такой системы кодирование строк таблицы переходов необходимо проводить таким образом, чтобы сохранить эти величины (см. подраздел 3.3.5). Ясно, что разброс в продолжительностях переходов между внутренними состояниями зависит от задержек в петлях обратных связей.

Таблица 2.18

	x = 0	x = 1
1	*1*,0	2,0
2	1,1	3,1
3	2,0	4,1
4	3,0	*4*,0

а) Исходная таблица

	x = 0	x = 1
1	*1*,0	2,0
2	—	3,1
3	—	4,1
4	5,0	*4*,0
5	6,0	—
6	1,1	—

б) Расширение исходной таблицы

	x = 0	x = 1	
1	*1*,0	2,0	1
2	1,1	3,1	236
3	2,0	*3*,0	45

в) Минимизированная таблица

	x = 0	x = 1
1	*1*,0	2,1
2	1,1	*2*,0

г) Минимальная таблица

В качестве примера того, как можно минимизировать таблицу переходов без учета временных соотношений, рассмотрим таблицу 2.18, *а*. Первый очевидный шаг состоит в точно таком же расширении таблицы, как и рассмотренное выше; при этом получаем таблицу 2.18, *б*. (По-прежнему считаем, что система начинает функционировать из

устойчивого состояния. Заметим, например, что никакая строка новой таблицы не покрывает строку 3 — неустойчивую строку исходной таблицы.) Теперь, для минимизации новой таблицы необходимо ввести модифицированное определение совместимости. Будем называть состояния p и q *B-совместимыми*, если для каждого входа I: (1) подходящим доопределением пустых клеток можно добиться того, что выходная последовательность, генерируемая системой в состоянии p в ответ на I, эквивалентна по существу (в рассмотренном выше смысле) выходной последовательности, когда начальным состоянием является q, и (2) устойчивые состояния, достигаемые в конце концов из начальных состояний p и q при подаче I, B-совместимы.

В нашем примере, согласно этому определению, имеются следующие максимальные B-совместимые множества: {1, 235, 236, 45}. При этом порождения следует рассматривать только в терминах достигаемых в конце концов устойчивых состояний, а здесь их просто нет. Таким образом, мы приходим к выбору совокупности {1, 236, 45} в качестве минимального замкнутого покрытия, по которому строится табл. 2.18, *в*.

Нетрудно, однако, заметить, что двухстрочная таблица 2,18, *г* является, в действительности, минимальным покрытием исходной таблицы (а также таблиц 2.18, *б* и *в*). Правда, эта таблица не может быть получена по нашей процедуре, которая, следовательно, не приводит к нахождению полного решения задачи минимизации в случае, когда временные соотношения не учитываются. Фактором, не учтенным нами при рассмотрении примера, является возможность исключения первого состояния выхода из последовательности, генерируемой после приложения нового входа, если этот выход совпадает с выходом в начальном устойчивом состоянии.

Возможный подход, который, похоже, применим к рассматриваемым здесь ситуациям, состоит в преобразовании расширенной таблицы в первичную форму с соответствующими пустыми клетками для выходов, являющихся первыми символами в переходных последовательностях. Так, в нашем примере таблица 2.18, *б* может быть преобразована в таблицу 2.19, *а* с максимальными B-совместимыми множествами {123, 235, 45}, не имеющими порождаемых множеств. Минимальным замкнутым покрытием будет,

очевидно, {123, 45}, которое определяет минимальное решение, представленное в таблице 2.19, *б*. Обобщение этого подхода и доказательство того, что он приводит к минимальному решению, пока отсутствуют.

Таблица 2.19

	x	
	0	1
1	*1*,0	2,—
2	—	3,1
3	—	4,1
4	5,—	*4*,0
5	1,1	—

а) Расширение исходной таблицы

	x		
	0	1	
1	*1*,0	2,1	123
2	1,1	*2*,0	45

б) Минимальная таблица

Минимизация числа состояний для НИВ-таблиц не рассматривалась детально, хотя проблемы при этом, похоже, те же, что и в изложенном здесь случае конечного числа изменений выхода.

ЗАМЕЧАНИЯ ПО БИБЛИОГРАФИИ

Решения задачи минимизации полностью определенных таблиц переходов были впервые представлены Хаффманом [49], Муром [81] и Мили [77]. Гинзбург [30—32] первым привел примеры того, что методы минимизации полностью определенных таблиц не всегда приводят к минимальным решениям для неполностью определенных таблиц, он предложил некоторые новые методы и первым дал нижнюю оценку для минимального решения, связанную с размером наибольшего несовместимого множества. Рассмотренный в настоящей главе общий подход к проблеме минимизации был сформулирован Полом и Ангером [90, 113], а применение в нем методов теории графов вытекает из идей, высказанных Ангером [114].

Идея введения «поглощающей строки», рассмотренная в разделе 2.1, принадлежит Нарасимхану [89], процедура 2.3 нахождения максимальных совместимых множеств была разработана Маркусом [66]. Рассмотрение частич-

ных объединений строк основано на работе Ангера [113]. Маккласки доказал теорему 2.1 [72] для таблиц с ограничениями на пары и в случае, когда такие таблицы соответствуют функциям с однократным изменением выхода и представлены в первичной форме. Эта теорема была затем обобщена Ангером [113] до рассмотренного здесь вида.

Материал последней части раздела 2.5 относительно таблиц переходов с заданным списком запрещенных входных последовательностей взят из работы Пола [92]. Пол и Вальдбаум [91] формально ввели процедуру 2.5 для приведения ОИВ-таблиц к стандартной форме перед их последующей минимизацией. Формальное доказательство того, что в этом классе таблиц выходы в неустойчивых состояниях можно положить равными выходам в последующих устойчивых состояниях без потери при этом желаемых решений, было представлено Ридом [95], который ссылается на Хаффмана [49], как на впервые высказавшего это утверждение. Материал по минимизации МИВ-таблиц переходов с учетом временны́х соотношений взят из работы Фридза [18]. Пол и Вальдбаум рассмотрели задачу минимизации МИВ-таблиц переходов без учета временны́х соотношений [92] (но без нашего предположения об устойчивости начальных состояний) и ввели понятие, названное здесь B-совместимостью.

Интересный подход к нахождению минимальных замкнутых покрытий путем представления задачи в форме модифицированной таблицы простых импликантов описан в работе Грасселли и Луччио [34, 35]. Этот не рассмотренный здесь метод кажется неудобным для ручных вычислений, но вполне приемлем при использовании ЦВМ. Авторы рассматривают также возможность применения к этой задаче целочисленного программирования. Грасселли и Луччио [35] рассмотрели наряду с задачей минимизации строк также задачу минимизации числа столбцов в таблице переходов. Другой не рассмотренный здесь подход к минимизации таблиц переходов связан с работами Битти и Миллера [4, 5, 79]. Интересное применение задачи о назначениях к понятиям совместимого и максимального совместимого множеств и рассмотрение задач, связанных с составлением алгоритмов для нахождения максимальных совместимых множеств, можно найти в работе Холла и Эктона [39].

ЗАДАЧИ

2.1. Построить таблицу переходов не более чем с четырьмя столбцами входов, для которой таблица 2.20 является таблицей и картой финальных пар.

Таблица 2.20

1	2	3	4
23×	×		
×	12	24	

Таблица 2.21

	x = 0	x = 1
1	1,—	2,0
2	3,0	1,0
3	2,1	1,0

2.2*. Построить по таблице 2.21 две различные таблицы путем доопределения значения выхода сначала нулем, а потом единицей. Минимизировать каждую из этих таблиц. Можно ли найти таблицу, покрывающую таблицу 2.21 и имеющую меньше строк, чем в каждой из полученных минимизированных таблиц?

2.3. Найти максимальные совместимые множества для таблицы 2.22.

Таблица 2.22

1	2	3	4	5	6	7
×						
×	×	×				
		×	×			
	×			×		

Таблица 2.23

	x_1x_2 = 00	01	11	10
1	2,0	3,0	5,0	7,0
2	2,0	1,0	4,0	5,0
3	3,0	2,0	3,0	6,0
4	3,0	2,0	1,0	5,1
5	7,0	3,0	6,0	2,0
6	7,0	3,0	1,0	2,0
7	2,0	5,0	4,0	6,0

Таблица 2.24

	x = 0	x = 1
1	3,00	4,——
2	2,——	5,——
3	1,01	5,10
4	1,11	—,——
5	2,——	3,11

Таблица 2.25

	x_1x_2 = 00	01	11	10
1	3,0	1,—	—	—
2	6,—	2,0	1,—	—
3	—,1	—	4,0	—
4	1,0	—	—	5,1
5	—	5,—	2,1	1,1
6	—	2,1	6,—	4,1

Таблица 2.26

	x_1x_2 = 00	01	11	10		00	01	11	10
1	6,—	—	—	7,—	6	—	6,0	—	4,1
2	3,—	2,—	—	7,0	7	7,0	6,1	4,—	5,—
3	2,—	7,—	—	—	8	8,—	—,1	—,1	10,—
4	8,—	10,—	4,0	8,—	9	10,1	1,—	4,—	—
5	—	—	9,—	—	10	—,0	—	—	3,—

Таблица 2.27

	A	*B*	*C*	*D*
1	2,0	4,0	—	6,0
2	3,0	5,0	5,0	—
3	—,0	—,0	6,0	4,0
4	1,—	—,1	—,1	—
5	3,—	1,1	—	7,0
6	—,1	2,1	—	—
7	—,0	—,1	8,—	—
8	—,1	—	9,0	—
9	—,1	—	—,1	1,1

а)

	A	*B*	*C*	*D*
1	2,0	4,0	—,0	7,—
2	3,0	5,0	—,0	—
3	1,0	6,0	—,0	—
4	—	2,1	—,1	1,—
5	—	—	—,1	3,0
6	2,1	—	—,1	—
7	—,0	—,1	8,—	—
8	—,1	—,0	9,—	—
9	—,1	—,1	—	1,1

б)

2.4. Найти таблицу с минимальным числом строк, эквивалентную таблице 2.23.

2.5. Привести таблицу 2.24 к минимальной покрывающей таблице.

2.6 $^{+}$. Привести таблицу 2.25 к минимальной покрывающей таблице.

Таблица 2.28

1	2,0	—,1	3,—	2,0
2	8,0	5,1	2,0	—
3	3,0	6,1	—	5,0
4	—	7,1	2,—	—
5	—	1,1	1,1	—
6	—	1,1	2,—	—
7	2,0	—,1	8,—	2,0
8	3,0	4,1	—	5,0

Таблица 2.29

	x_1x_2 00	01	11	10
1	1,0	2,1	3,0	3,0
2	1,1	4,0	3,0	2,0
3	4,0	3,1	1,0	1,0
4	2,1	1,0	4,0	3,0

а)

	x_1x_2 00	01	11	10
1	2,0	1,1	3,0	3,0
2	4,1	3,0	2,0	1,0
3	3,0	4,1	1,0	1,0
4	3,1	2,0	1,0	4,0

б)

Таблица 2.30

	A	*B*	*C*
1	3,—	2,—	1,0
2	3,—	1,0	1,0
3	1,1	2,1	3,0

а)

	A	*B*	*C*
1	2,—	2,0	1,0
2	3,1	3,1	2,0
3	2,—	3,—	1,0

б)

2.7. Привести таблицу 2.26 к минимальной покрывающей таблице (это сложная задача).

2.8 *. Найти таблицы с минимальным числом строк, покрывающие соответственно таблицы 2.27, *а* и *б*.

2.9. Найти таблицу с минимальным числом строк, эквивалентную таблице 2.28.

2.10. Эквивалентны ли таблицы 2.29, *а* и *б*?

2.11. а)$^+$ Покрывает ли таблица 2,30, *а* таблицу 2.30, *б*?

б) Покрывает ли таблица 2,30, *б* таблицу 2,30, *а*?

2.12. Найти таблицу с минимальным числом строк, покрывающую обе таблицы 2.31, *а* и *б*.

Таблица 2.31

	A	*B*	*C*
1	2,—	—	1,0
2	1,1	3,1	2,0
3	2,—	1,0	1,0

а)

	A	*B*	*C*
1	2,1	2,1	1,0
2	1,—	2,—	3,0
3	1,—	1,0	3,0

б)

2.13. Найти таблицу с минимальным числом строк, покрывающую таблицу 2.32 при условии, что никакая входная последовательность не содержит *ABB* или *BC* в качестве подпоследовательностей.

2.14. При учете временны́х соотношений найти наименьшую таблицу, покрывающую таблицу 2.33.

Таблица 2.32

	A	*B*	*C*
1	2,0	3,0	2,0
2	2,0	4,0	4,1
3	3,0	2,0	3,0
4	3,0	1,0	4,0

Таблица 2.33

	x_1x_2 00	01	11	10
1	***1***,0	2,1	4,0	***1***,0
2	***2***,1	5,0	—	***2***,0
3	***3***,1	5,0	***3***,1	***3***,1
4	1,0	***4***,1	***4***,0	2,0
5	3,1	***5***,1	***5***,0	6,1
6	1,0	3,1	4,0	***6***,0

2.15*. Зачеты назначены на понедельник, вторник, среду и четверг в 3 часа дня. Каждый из десяти классов *A*, *B*, *C*, *D*, *E*, *F*, *G*, *H*, *J*, *K* должен сдать свой зачет. Хотя некоторые студенты могут учиться не только в одном классе, желательно, чтобы никому из них не пришлось одновременно сдавать более одного зачета. Показать, как это может быть сделано, при условии, что единственными парами классов, не имеющими общих студентов, являются следующие пары: *AG*, *CH*, *EK*, *AD*, *CJ*, *GH*, *AF*, *DE*, *BE*, *DF*, *BK*.

ГЛАВА 3
ПРОБЛЕМА КОДИРОВАНИЯ СОСТОЯНИЙ

Как отмечалось в разделе 1.3, проблема кодирования состояний (называемая также проблемой кодирования строк) заключается в приписывании строкам таблицы переходов комбинаций значений двоичных внутренних переменных таким образом, чтобы все внутренние переходы успешно завершались независимо от относительных значений задержек в цепях внутренних переменных. В рамках этого основного требования имеются также и другие условия, которые могут включать требования минимальности числа используемых внутренних переменных, минимальности числа внутренних переменных, значения которых должны изменяться при одном переходе, минимальности числа необходимых логических элементов, простоты процесса проектирования и минимальности среднего потребления энергии.

Фундаментальный характер проблемы кодирования состояний обусловил развитие ряда методов нахождения решений, удовлетворяющих различным критериям. Некоторые из рассматриваемых здесь методов связаны с универсальным кодированием, пригодным для произвольных таблиц переходов с заданным числом строк, тогда как другие пригодны для кодирования лишь таблиц переходов специальных видов. Для некоторых способов кодирования приводятся верхние оценки числа необходимых внутренних переменных как функции числа строк.

3.1. Связанные строчные множества

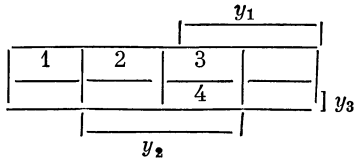
Попробуем закодировать таблицу 3.1, *а* с четырьмя строками минимально возможным числом внутренних переменных, а именно, двумя переменными. Это означает, что каждая из четырех комбинаций значений (*y*-состояний) двух внутренних переменных должна быть припи-

сана какой-либо одной строке таблицы. Допустим, что мы начали с приписывания 00 строке 1. Это не влечет за собой никакой потери общности, поскольку если некоторый вариант кодирования строк пригоден, то пригодны и все

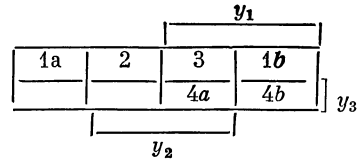
Таблица 3.1

	A	*B*	*C*	*D*
1	***1***,0	2,0	***1***,0	***1***,0
2	3,0	***2***,0	1,0	***2***,0
3	***3***,0	4,1	1,0	1,0
4	1,0	***4***,1	1,0	***4***,0

а) Исходная таблица



б) Фрагмент кодирования строк



в) Завершенный вариант кодирования

	A	*B*	*C*	*D*	y_1	y_2	y_3
1*a*	***1a***,0	2,0	1*b*,0	***1a***,0	0	0	0
1*b*	***1b***,0	1*a*,0	***1b***,0	***1b***,0	1	0	0
2	3,0	***2***,0	1*b*,0	***2***,0	0	1	0
3	***3***,0	4*a*,0	1*b*,0	1*b*,0	1	1	0
4*a*	4*b*,0	***4a***,0	1*b*,0	***4a***,0	1	1	1
4*b*	1*b*,0	***4b***,1	1*b*,0	***4b***,0	1	0	1

г) Матрица переходов

другие варианты, получающиеся из исходного путем отрицаний значений некоторой совокупности внутренних переменных.

Поскольку имеется переход от строки 1 к строке 2 (в столбце *B*) и этот переход является обязательным в том смысле, что он не может быть произведен через какое-либо другое состояние (переход 3 → 1 в столбце *C*, например,

можно произвести через строку 2, заменив $N(3, C)$ на 2), необходимо, чтобы состояние, приписанное строке 2, было соседним (отличающимся от последнего значением только одной внутренней переменной) состоянию, приписанному строке 1. Используем комбинацию 01 для строки 2, опять-таки без потери общности, поскольку перестановка переменных в пригодном кодировании дает снова пригодное кодирование. Поскольку y-состояние, приписанное строке 2, должно быть соседним по отношению к y-состоянию для строки 3 из-за состояния $N(2, A)$, получаем, что строке 3 нужно приписать комбинацию 11. Но тогда появляется критическое состязание для перехода между строками 3 и 1, начинающегося в состоянии $3 - D$. Таким образом, для этой таблицы не существует приемлемого варианта кодирования с помощью двух переменных.

Допустим, что мы можем использовать большее число внутренних переменных, но ограничиваем себя условием, чтобы каждой строке было приписано единственное y-состояние и каждый переход между строками был непосредственным и состоял в изменении значения единственной внутренней переменной (содержал единственное y-изменение). Тогда мы должны найти три y-состояния s_1, s_2, s_3, которые должны быть попарно соседними в силу наличия переходов $1 \longrightarrow 2$, $2 \longrightarrow 3$ и $3 \longrightarrow 1$. Это также невозможно, поскольку, если поочередно происходит каждый из указанных переходов, система выйдет и вернется в состояние 1 и при этом произойдут три y-изменения. Таким образом, значение по крайней мере одной из y-переменных после последнего перехода $3 \longrightarrow 1$ будет отличаться от исходного, что противоречит условию единственности y-состояния, приписанного строке 1.

Для разрешения этого затруднения ослабим условия, состоящие в том, что каждой строке приписывается единственное y-состояние и каждый переход совершается непосредственно (за один шаг). Кроме того, разрешим ввести третью y-переменную. Определим *строчное множество* R_i как множество y-состояний, приписанных строке i в таблице переходов. Никакое y-состояние не входит более чем в одно строчное множество. Когда система находится в y-состоянии из R_i и входом является I_j, выходом будет $Z(i, I_j)$, и если $N(i, I_j) = k$, то система приходит в состоя-

ние из R_k либо непосредственно, либо через ряд переходов между y-состояниями из R_i.

Построим строчные множества таким образом, что: (1) если t и u являются y-состояниями одного и того же R_i, то существует последовательность переходов (путь) из t в u и, наоборот, из u в t через ряд попарно соседних состояний в R_i (такое множество R_i называется *связанным*); (2) если в таблице переходов имеется обязательный переход от строки i к строке j, то множества R_i и R_j — *соседние* в том смысле, что некоторое состояние из R_i является соседним некоторому состоянию из R_j.

В нашем примере парами строк, которые должны иметь соседние y-состояния, являются (1, 2), (2, 3), (3, 4), (1, 4) и (1, 3). Попытаемся построить связанные строчные множества с учетом необходимых переходов, хотя и следует заметить, что пока для этого нет алгоритмов, не использующих перебор. Цель предлагаемого здесь примера состоит в том, чтобы дать читателю некоторое интуитивное описание проблемы, проиллюстрировать на практике, как она может быть решена без особых затруднений, и ввести способ универсального кодирования строк связанных строчных множеств.

Выбранный вариант кодирования удобно наносить на карту Карно (K-карту), координатами которой являются y-переменные. Начнем с приписывания 000 строке 1, 010 — строке 2, 110 — строке 3 и 111 — строке 4. Такой выбор удовлетворяет требованиям, связанным с первыми тремя соседними парами, приведенными выше, он зафиксирован в таблице 3.1, *б*. Теперь необходимо добавить y-состояния так, чтобы сделать множество R_1 соседним множествам R_3 и R_4. Выберем один из возможных вариантов такого добавления, именно, расширим R_1, чтобы сделать его соседним R_3, а затем расширим R_4, чтобы сделать его соседним R_1. Этот вариант показан в таблице 3.1, *в*, где добавленными состояниями являются 1*в* и 4*в*.

В таблице 3.1, *г* показана матрица переходов, соответствующая выбранному варианту кодирования, и в ней все переходы, которые должны быть организованы между y-состояниями. Например, если система находится в $1b - B$, она не может непосредственно перейти в $2 - B$ (см. табл. 3.1, *а*), поскольку при этом возникает критическое состязание (меняют свои значения две переменные

y_1 и y_2). Поэтому сначала производится переход внутри множества R_1 в состояние $1a$, соседнее R_2. Возможны различные варианты определения следующего состояния в матрице переходов, особенно с учетом устойчивых состояний. Можно, например, определить $N(1a, A)$ или $N(1b, A)$ как $1a$ или $1b$, что дает нам четыре варианта, в том числе вариант $N(1a, A) = 1b$ и $N(1b, A) = 1a$, который приведет к непрерывной смене (генерации) значений переменной y_1 в устойчивом состоянии, соответствующем состоянию $1—A$ таблицы переходов 3.1, *а*. Выбор подобных вариантов возможен с целью, например, упрощения логических схем. Заметим, что два неиспользованных y-состояния 001 и 011 приведут к появлению двух не показанных в таблице 3.1, *г* строк с незаполненными клетками в матрице переходов.

Вообще говоря, метод нахождения варианта кодирования, показанного в таблице 3.1, *г*, является методом проб и ошибок. Удовлетворив одни условия, можно обнаружить, что другие условия при этом перестают выполняться, это приводит к необходимости изменения решений, принятых на предыдущих шагах. С приобретением опыта можно, конечно, достичь некоторых успехов, но в общем случае для данной таблицы нелегко определить даже число необходимых внутренних переменных. Единственное, что всегда можно сделать это применить универсальное кодирование, пригодное для любой таблицы переходов. Не следует, конечно, ожидать, что такое «безопасное» кодирование будет наилучшим для конкретной таблицы переходов.

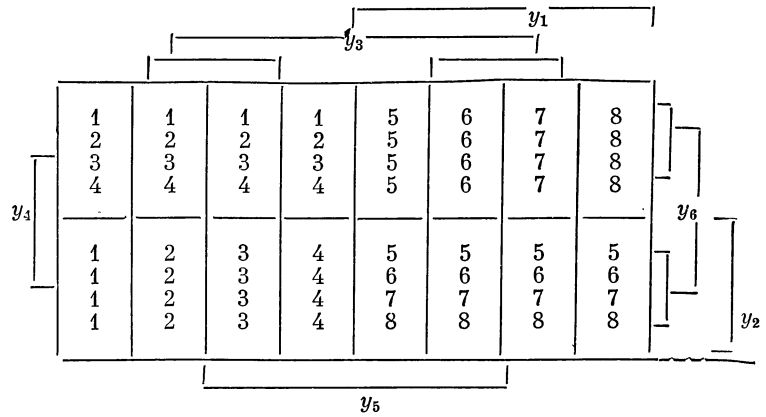
Основная идея состоит в том, чтобы построить связанные строчные множества, попарно *сцепленные* друг с другом в том смысле, что при данных i и j R_i является соседним R_j. Отсюда следует, что, независимо от вида переходов в исходной таблице, они могут быть осуществлены в виде последовательности одношаговых (без критических состязаний) переходов внутри строчного множества, соответствующего начальному состоянию, и конечного одношагового перехода через «границу» в строчное множество, соответствующее конечному состоянию рассматриваемого перехода.

Таблица 3.2 является K-картой, на которой показан вариант кодирования восьмистрочной таблицы переходов,

использующий 6 переменных, причем все строчные множества являются связанными и попарно сцепленными. При рассмотрении такой карты удобно использовать понятие квадрантов K-карты, выделяемых значениями переменных y_1 и y_2 (10 есть верхний правый квадрант и т. д.). Переменные с нечетными индексами обозначают

Таблица 3.2

$2S_0$-кодирование для таблиц с восемью строками



столбцы на этой карте, переменные с четными индексами — строки. Рассмотрим кодовые комбинации, приписанные состоянию 2, некоторому «типичному» состоянию таблицы переходов. В 00-квадранте все y-состояния строки, для которой $y_4y_6=01$ (впредь будем ее называть 01-строкой), отнесем к R_2. Ясно, что эти состояния образуют связанное подмножество, включающее все состояния, для которых $y_1y_2y_4y_6=0001$ (y_3 и y_5 принимают все четыре возможные комбинации значений). Вообще, внутри каждого квадранта строки и столбцы образуют связанные подмножества. Столбец, для которого $y_3y_5=10$ (10-столбец) в 01-квадранте, также отнесем к R_2. Поскольку 01-строка в 00-квадранте, естественно, содержит клетку 10 столбца (y-состояние 001001), то отсюда следует, что два указанных подмножества состояний из R_2 (состояния 01-строки и 10-столбца) связаны между собой, так что R_2 — связанное строчное множество. Аналогичное утверждение справедливо для любого другого строчного множества.

Доказательство того, что две компоненты в R_2 являются соседними, можно обобщить, показав, что каждая из строк 00-квадранта связана с каждым из столбцов в двух соседних квадрантах (01 и 10). Так, например, 01-строка 00-квадранта является соседней каждому из столбцов в 01-квадранте, которые, просматривая слева направо, следует отнести к множествам R_1, R_2, R_3 и R_4 соответственно. Эта же строка является соседней каждому из столбцов квадранта 10, которые принадлежат множествам R_5, R_6, R_7 и R_8. Естественно, в силу симметрии, столбцы аналогичным образом связаны с каждой строкой каждого соседнего квадранта. Отсюда следует, что строчные множества сцеплены друг с другом, так что мы действительно имеем универсальное кодирование для всех таблиц с восемью состояниями.

Показанная в таблице 3.2 схема построения связанных строчных множеств легко может быть обобщена на таблицы переходов с 2^n строками, где n — любое положительное целое число. Рассмотрим случай $n=4$, когда строчных множеств требуется в два раза больше, чем в нашем примере. Общая структура таблицы сохраняется неизменной, но число строк и столбцов в каждом квадранте удваивается. Это изменение требует дополнительного введения y-переменных с нечетными индексами для столбцов и с четными индексами для строк. Соображения по доказательству связности строк и столбцов в соседних квадрантах остаются теми же. Пусть $S_0=\mathrm{E}[\log_2 r]$ (см. раздел 1.3), тогда можно видеть, что число y-переменных, используемых в этом типе кодирования для r-строчной таблицы, равно $2S_0$; такое кодирование можно назвать, таким образом, $2S_0$-*кодированием*.

Назовем *тактом* максимальный отрезок времени, необходимый для изменения значения любой y-переменной. Тогда в случае использования $2S_0$-кодирования всякий внутренний переход в таблице может быть произведен не более чем за два такта. Например, в таблице 3.2 переход из y-состояния 001001, которое принадлежит R_2, в y-состояние, соответствующее строке 5, может быть осуществлен в два этапа. Прежде всего заметим, что могли бы быть некритические состязания в пределах 01-строки 00-квадранта при переходе из состояния 001001 в состояние 000011, соседнее R_5. Однако переходы в пределах

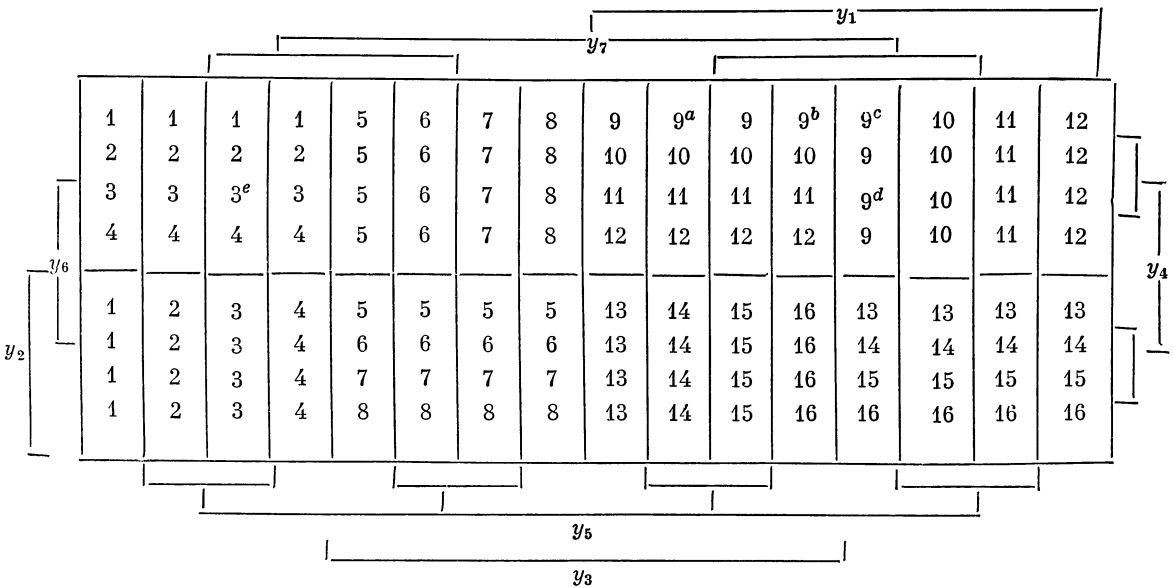
строчного (или столбцового) подмножества строчного множества всегда можно организовать без некритических состязаний, поскольку при этом изменяются только столбцовые (или строчные) переменные, так что все промежуточные состояния находятся в том же самом строчном множестве. Время, необходимое для прохождения этого первого этапа, равно одному такту, так как всем тем y-переменным, которые должны измениться, новые Y-значения задаются одновременно. Второй этап требует изменения переменной квадранта (y_1 в нашем примере), что переводит систему в новое строчное множество. В ряде случаев оказывается необходимым лишь один этап, например, при переходе от какого-либо элемента из R_2 к какому-либо элементу из R_3 или тогда, когда начальное состояние перехода оказывается на границе строчного множества, содержащего конечное состояние перехода.

Итак, в случае $2S_0$-кодирования любая таблица переходов может быть закодирована с помощью $2S_0$ y-переменных без возникновения критических состязаний, причем любой переход завершится за два такта. Покажем теперь, как сократить число y-переменных до $2S_0 - 1$, но ценой удвоения максимального времени переходов между строками.

В $(2S_0-1)$-кодировании квадранты $2S_0$-кодирования заменяются на октанты, причем иногда могут потребоваться переходы между строчными и столбцовыми подмножествами, приписанными разным состояниям таблицы переходов. В таблице 3.3 показан вариант кодирования с помощью семи переменных, в котором получаются шестнадцать сцепленных связанных строчных множеств. Значения переменных y_1, y_2 и y_3 определяют октанты, y_5 и y_7 указывают столбец октанта, а y_4 и y_6 — строку октанта. Каждое строчное множество содержит строку и столбец в смежных октантах (так что они оказываются связанными), и если какое-либо отдельное строчное (столбцовое) подмножество не является соседним состоянию некоторого строчного множества, содержащего конечное состояние некоторого перехода, то входящее в это множество столбцовое (строчное) подмножество, напротив, будет соседним. Например, если система сначала находится в y-состоянии из R_9, помеченном верхним индексом a в таблице 3.3, и необходимо перейти в состояние из R_3, то

Таблица 3.3

$(2S_0-1)$-кодирование для таблиц с шестнадцатью строками



1	1	1	1	5	6	7	8	9	9^a	9	9^b	9^c	10	11	12
2	2	2	2	5	6	7	8	10	10	10	10	9	10	11	12
3	3	3^e	3	5	6	7	8	11	11	11	11	9^d	10	11	12
4	4	4	4	5	6	7	8	12	12	12	12	9	10	11	12
1	2	3	4	5	5	5	5	13	14	15	16	13	13	13	13
1	2	3	4	6	6	6	6	13	14	15	16	14	14	14	14
1	2	3	4	7	7	7	7	13	14	15	16	15	15	15	15
1	2	3	4	8	8	8	8	13	14	15	16	16	16	16	16

такой переход не может быть совершен непосредственно из строки 00, содержащей исходное состояние. Сначала необходим переход в 01-столбец в 100-октанте (состояние 9 с индексом b), сопровождаемый изменением октантной переменной y_3, что позволяет затем системе перейти в состояние 9, помеченное индексом c. Изменение индекса c на d в новом октанте сопровождается, в итоге, изменением октантной переменной y_1, вследствие чего система переходит в состояние 3, помеченное индексом e. Это случай наихудшего перехода, требующего четырех тактов.

Подчеркнем, что удвоение числа строчных множеств достигается удвоением общего числа строк и столбцов K-карты, что требует двух дополнительных y-переменных. С другой стороны, для таблицы переходов с четырьмя строками было найдено, что каждый октант, соответствующий состоянию таблицы, содержит только одну строку и один столбец. Читателю предлагается найти универсальное кодирование рассматриваемого вида тремя переменными для четырехстрочных таблиц.

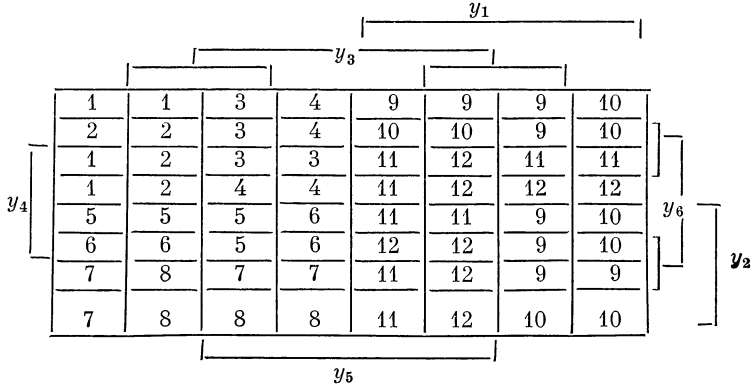
Между прочим, имеется класс $(2S_0+1)$-кодирований, подобных рассмотренным выше, в которых K-карта разбивается лишь на два сектора (см. задачу 3.3). Может возникнуть мысль, что при увеличении числа секторов число необходимых y-переменных должно уменьшаться. Однако это не так, и было показано, правда, без доказательства, что в универсальном кодировании, основанном на сцепленных строчных множествах, требуется $2S_0-1$ переменных.

Если число r строк исходной таблицы переходов не является степенью числа 2 и удовлетворяет соотношению $2^{S_0-1}<r\leqslant(3/4)2^{S_0}$, то возможно кодирование, основанное на использовании взаимно-сцепленных связанных строчных множеств, требующее лишь $2S_0-2$ переменных. Для таблиц с двенадцатью строками такое кодирование показано в таблице 3.4. Левая половина этой таблицы в точности является картой $(2S_0-1)$-кодирования для восьми состояний, а правая половина кодирует оставшуюся часть состояний таким образом, что принадлежащие ей строчные множества являются соседними друг другу и каждому из строчных множеств, представленных в левой половине.

Рассмотренные варианты кодирования интересны в том плане, что они довольно четко устанавливают определен-

Таблица 3.4

($2S_0-2$)-кодирование для таблиц с двенадцатью строками



1	1	3	4	9	9	9	10
2	2	3	4	10	10	9	10
1	2	3	3	11	12	11	11
1	2	4	4	11	12	12	12
5	5	5	6	11	11	9	10
6	6	5	6	12	12	9	10
7	8	7	7	11	12	9	9
7	8	8	8	11	12	10	10

ные границы и допускают разновидности, полезные в отдельных частных случаях. Тем не менее, вряд ли возможно доказать, что какое-либо из универсальных кодирований является эффективным для конкретной таблицы переходов.

3.2. Кодирование с совместным использованием строк

Если в предыдущем разделе предполагалось, что каждое y-состояние, используемое при кодировании на основе строчных множеств, приписано какой-либо определенной строке таблицы переходов, то сейчас мы примем более «гибкий» подход. С каждой строкой будем связывать единственное y-состояние, а остальные y-состояния будут использоваться при необходимости организации промежуточных переходов между строками, y-состояния которых не являются соседними. Каждое такое «промежуточное» состояние, которое выступает в виде дополнительной строки в матрице переходов, может использоваться в разных столбцах для построения промежуточных переходов между разными парами строк. Отсюда происходит термин *совместное использование строк*.

Будем использовать понятие *множества предшественников*. Каждому устойчивому состоянию $q-I$ соответству-

ет множество предшественников D_{Iq}, содержащее все те строки, которые в столбце I имеют q в качестве следующего состояния; таким образом, $D_{Iq} = \{p \mid N(p, I) = q\}$.

Для иллюстрации предлагаемого подхода, который по своей эвристической сущности подобен подходу, рассмотренному ранее в разделе 3.1, используем таблицу 3.5. Первый шаг состоит в выписывании множеств предшественников для каждого столбца. В столбце A нашего примера существует возможность выбора, поскольку для состояния 6—A множество предшественников (3, 4, 6) является трехэлементным. Может оказаться более удобным изменить таблицу переходов, положив $N(3, A)$ равным 4 или $N(4, A)$ равным 3, сделав тем самым один из переходов в устойчивое состояние 6—A непрямым без изменения функции. (Эта операция, по сути, обратна первому шагу процедуры 2.5 получения таблиц переходов в стандартной форме.) Указанное изменение дает нам возможность заменить пару переходов $3 \to 6$, $4 \to 6$ другой парой переходов, если такая замена помогает найти приемлемый вариант кодирования. Для нашего примера множества предшественников следующие (одноэлементные множества опущены):

Таблица 3.5

Исходная таблица переходов

	A	*B*	*C*	*D*
1	***1***,0	***1***,0	4,0	6,0
2	1,0	***2***,0	5,0	***2***,0
3	6,0	4,0	***3***,0	***3***,0
4	6,0	***4***,0	***4***,0	5,0
5	***5***,0	1,0	***5***,0	***5***,0
6	***6***,0	7,0	***6***,0	***6***,0
7	***7***,1	***7***,0	3,0	2,0

$$A: (2,1)\,(3,\ 4,\ 6),$$
$$B: (3,4)\,(5,1)\,(6,7),$$
$$C: (1,4)\,(2,5)\,(7,3),$$
$$D: (1,6)\,(4,5)\,(7,2).$$

Сначала мы неформально покажем, что не существует варианта кодирования с помощью трех переменных, использующего понятие строчного множества. Поскольку в таблице переходов семь строк, при трех переменных только одно из строчных множеств может иметь два состояния. Рассмотрим переходы между состояниями пар (1,2), (1,4), (1,5), (2,5) и (4,5). Из рассмотрения следует, что строчные множества для этих четырех состояний

должны быть соседними, характер этого соседства показан на рис. 3.1, *а* (строчные множества, соединенные ребром, должны быть соседними). Допустим, что каждое из этих строчных множеств содержит единственное y-состояние, тогда, поскольку ребро графа соответствует изменению только одной y-переменной, согласно этому графу можно перейти от R_1 к R_2 и через R_5 вернуться снова в R_1, осуществив в точности три изменения y-переменной. Ясно,

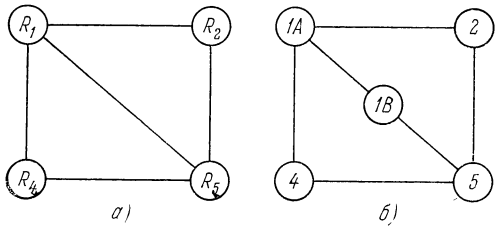


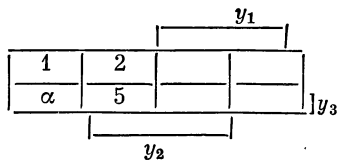
Рис. 3.1. *а*) Графическое представление требований соседства. *б*) Один из вариантов, удовлетворяющих графу *а*).

что это невозможно, так же как невозможно осуществление аналогичной петли из R_1, R_4 и R_5. Чтобы разорвать обе эти «нечетные» петли, необходимо добавить второе состояние или в R_1, или в R_5. Добавим его в R_1 (рассуждения остаются теми же и для второго случая), получим граф, в котором y-состояния приписаны строкам 1, 2, 4 и 5 так, как показано на рис. 3.1, *б*. Однако нельзя приписать y-состояния каждой вершине этого графа таким образом, чтобы соседним вершинам этого графа соответствовали соседние кодовые комбинации. Причина здесь в том, что для двух y-состояний, отличающихся значениями двух переменных, имеются лишь два других y-состояния, каждое из которых является соседним обоим y-состояниям. Но согласно рис. 3.1, *б* каждое из трех состояний 2, 1*B* и 4 является соседним обоим состояниям 1*A* и 5. Таким образом, мы пришли к противоречию.

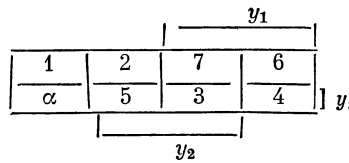
Посмотрим, нельзя ли найти решение с использованием трех переменных при кодировании с совместным использованием строк. Каждое из множеств предшественников должно быть связанным, для этого, если нужно, добавляются дополнительные состояния. Поскольку сов-

местно используемая строка может быть закодирована лишь одним y-состоянием и поскольку строка 1 принадлежит четырем множествам предшественников, важно, чтобы состояние, приписанное совместно используемой строке (обозначим ее через α), было соседним состоянию, приписанному строке 1. Состояния 2 и 5 должны быть связаны, а состояние 1 должно быть связано с каждым из них. Можно сделать состояния 2 и 5 соседними друг другу, сделав одно из них соседним α, а другое — соседним 1. Это дает нам частичное кодирование, показанное в таблице 3.6, *а*. Теперь необходимо разместить состояние 4

Таблица 3.6



а) Частичное кодирование



б) Окончательное кодирование

	A	*B*	*C*	*D*	y_1	y_2	y_3
1	***1***,0	***1***,0	α,0	6,0	0	0	0
2	1,0	***2***,0	5,0	***2***,0	0	1	0
3	4,0	4,0	***3***,0	***3***,0	1	1	1
4	6,0	***4***,0	***4***,0	α,0	1	0	1
5	***5***,0	α,0	***5***,0	***5***,0	0	1	1
6	***6***,0	7,0	***6***,0	***6***,0	1	0	0
7	***7***,1	***7***,0	3,0	2,0	1	1	0
α	—	1,0	4,0	5,0	0	0	1

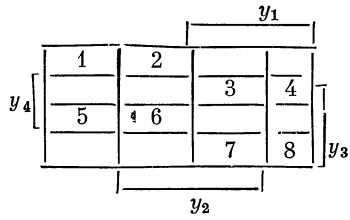
в) Матрица переходов для таблицы 3.5

таким образом, чтобы оно было связано с 1 и 5. Это можно сделать, только приписав ему y-состояние из оставшихся y-состояний, соседних α. Далее необходимо приписать состоянию 6 оставшееся y-состояние, связанное с 1. Поскольку 7 должно быть связано с 2 и 6, ему следует приписать лишь то из оставшихся y-состояний, которое является соседним обоим этим состояниям. Таким образом, у нас осталось лишь одно y-состояние, которое приписываем оставшейся строке 3.

Проверяя полученное кодирование, показанное в табл. 3.6, *б*, по перечню переходов, видим, что в столбце A множество (3, 4, 6) является связанным в силу того, что 3 — соседнее 4, а 4 — соседнее 6. В столбце B (1,5) является связанным через α. Аналогично совместно используемая строка связывает множества (1,4) в столбце C и (4,5) в D. Все другие переходы удовлетворяют условию прямого соседства, так что полученное кодирование пригодно. Заметим, что в каждом из столбцов совместно используемая строка может служить для более чем одного перехода лишь тогда, когда все эти переходы имеют одно и то же множество предшественников. Матрица переходов для полученного кодирования показана в табл. 3.6, *в*, где изменены некоторые из следующих состояний, чтобы ввести промежуточные переходы, о чем уже говорилось выше.

Таблица 3.7а

Универсальное кодирование для таблиц с восемью строками



Предыдущий пример лишний раз показывает, что по числу необходимых y-переменных способ кодирования с совместным использованием строк в общем случае более эффективен, чем способ, основанный на строчных множествах.

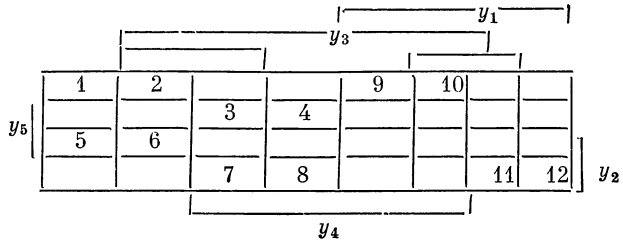
Более интересным и содержательным примером является пример кодирования с помощью четырех переменных, показанный в таблице 3.7а; такое кодирование пригодно для всех ОИВ-функций с восемью состояниями. Для каждого столбца такой таблицы можно использовать непротиворечивым образом восемь неприписанных y-состояний в качестве совместно используемых строк и построить с их помощью переходы между всеми несоседними состояниями. При кодировании с использованием строчных множеств для этого класса таблиц потребуется пять внутренних переменных. Такой «удачный» вариант кодирования состояний четырьмя переменными был обнаружен с помощью машинного алгоритма поиска. Его пригодность была затем установлена более обычным образом, путем группировки y-состояний в соседние пары, рассмотрения всех возможных в столбце комбинаций переходов между

парами (причем для уменьшения полного числа таких комбинаций использовалось свойство симметрии) и доказательства того, что промежуточные состояния удовлетворяют необходимым требованиям в каждом случае.

Аналогичным образом было найдено кодирование, показанное в таблице 3.7б, пригодное для всех ОИВ-таблиц

Таблица 3.7б

Универсальное кодирование для таблиц с двенадцатью строками

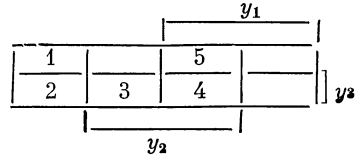


с двенадцатью строками и использующее пять внутренних переменных (тогда как в $(2S_0-2)$-кодировании с использованием строчных множеств необходимы шесть переменных), однако для него нет пока отдельного доказательства. Конечно же, не следует пренебрегать и универсальным кодированием, описанным в разделе 3.1 и пригодным для МИВ- и НИВ-таблиц, так же как и для ОИВ-таблиц.

Пока универсальное кодирование с совместным использованием строк не распространялось на таблицы, число строк в которых отлично от восьми или двенадцати (хотя, естественно, кодирование для восьми строк пригодно для таблиц с семью строками). Было установлено, однако, что пригодное кодирование с совместным использованием строк вида, показанного в таблице 3.7в, можно найти для всякой ОИВ-таблицы, имеющей пять строк и менее пятнадцати столбцов. В частности, приведенное в таблице 3.7в кодирование пригодно для любой ОИВ-таблицы,

Таблица 3.7в

Вариант кодирования, пригодный для всех таблиц с пятью строками и менее чем с пятнадцатью столбцами



не содержащей столбца с множествами предшественников (1,4) и (2,5). Для такой таблицы можно просто менять местами y-состояния, приписанные состояниям 4 и 5. Но это нельзя делать, если множествами предшественников в каком-либо столбце являются (1,5) и (2,4), в этом случае требуется другая перестановка. Существует пятнадцать таких различных вариантов столбцов, и если только таблица переходов содержит все эти пятнадцать столбцов, то для кодирования, представленного в таблице 3.7в, невозможно найти подходящую перестановку.

Ясно, что существует ряд интересных нерешенных вопросов, касающихся проблемы реализации таблиц переходов с помощью минимального числа внутренних переменных. Однако в последующих разделах мы отдаем предпочтение некоторым другим критериям.

3.3. Кодирование с однотактными переходами

При построении переключательных схем часто большое внимание уделяется скорости выполнения операций в схеме, и было бы предосудительным расплачиваться за увеличение скорости существенным увеличением числа используемых компонент. Ключевым фактором, влияющим на скорость операций асинхронных последовательностных схем, является максимальное число тактов, необходимых для совершения переходов между состояниями. Как показано в предыдущем разделе, при использовании $2S_0$-кодирования состояний это число равно 2. Ниже показано, что имеется ряд способов уменьшения этого числа вдвое. Кодирование, при котором для совершения любого перехода достаточно одного такта, называется *кодированием с однотактными переходами* (ОТП-кодированием).

Первым из ОТП-кодирований следует рассмотреть *кодирование соседними кодовыми комбинациями,* обладающее дополнительным свойством, что при каждом внутреннем переходе между строками имеет место изменение только одной внутренней переменной, при этом решение представляется в виде класса универсальных кодирований. Свойство соседства кодовых комбинаций весьма существенно, когда необходимые изменения значений внутренних переменных связаны с энергетическими затратами,

тепловыми потерями, размещением компонент или возможностью появления неисправностей. Такое кодирование можно рассматривать и как полезный инструмент в разрешении ряда теоретических вопросов.

Менее ограниченный класс ОТП-кодирований, называемых *кодированием с некритическими состязаниями* (НКС-кодированиями), может включать одновременные изменения нескольких внутренних переменных в определенных переходах, причем конечные состояния переходов не зависят от порядка, в каком в действительности изменяются y-переменные. Такие кодирования специфичны для определенных таблиц переходов и являются, как можно ожидать, обычно более эффективными, чем кодирования соседними кодовыми комбинациями. Как будет показано в следующих двух главах, определенные свойства таких кодирований делают их весьма ценными также для других целей. В настоящем разделе выводятся некоторые оценки числа переменных, необходимых для реализации ОТП-кодирований.

3.3.1. Кодирование соседними кодовыми комбинациями. При применении универсального кодирования этого типа в случае r-строчной таблицы переходов должны использоваться по крайней мере $r-1$ внутренних переменных, поскольку из y-состояния, представляющего строку i, путем изменения какой-либо одной переменной должны быть достижимы $r-1$ других состояний. Понятие совместно используемых строк уже неприменимо, так как здесь нет промежуточных состояний; все переходы должны быть прямыми от начального к конечному состоянию перехода. Это с неизбежностью приводит нас к подходу с использованием сцепленных строчных множеств, хотя прямой характер переходов и запрещает переходы между состояниями одного и того же множества, так что незачем делать эти множества связанными. Эти соображения иллюстрируются таблицей 3.8, на которой представлено универсальное кодирование соседними кодовыми комбинациями с помощью трех переменных, пригод-

Таблица 3.8

Универсальное кодирование соседними кодовыми комбинациями для таблиц с четырьмя строками

y_1				
0	2	3	1	y_3
3	1	0	2	
y_2				

ное для всех четырехстрочных таблиц (по причине, которая вскоре станет ясной, строки таблицы переходов в настоящем подразделе нумеруются от 0 до $r-1$). Каждое строчное множество имеет два состояния, кодовое расстояние между которыми выбирается по возможности наибольшим, и каждое состояние в R_i является соседним состоянию в R_j для каждого $j \neq i$. Здесь число внутренних переменных равно $r-1$, это минимальное число, необходимое для кодирования всех четырехстрочных таблиц даже путем таких кодирований, которые не обладают ОТП-свойствами. Рассматриваемое кодирование является, таким образом, интересным и полезным независимо от его иллюстративной наглядности.

Теперь рассмотрим обобщение этого кодирования на большие таблицы с $r=2^n-1$. (Для тех, кто знаком с кодами Хэмминга, исправляющими одиночные ошибки [41], можно сказать, что y-состояния, приписанные R_0, соответствуют кодовым комбинациям кода Хэмминга, не содержащим ошибок, а для $i>0$ R_i состоит из таких кодовых комбинаций, которые содержат ошибку в i-м разряде.) Необходимо сделать следующее формальное замечание. Пусть $b_j(I)$ обозначает j-й символ справа в двоичном представлении неотрицательного целого числа I ($I \leqslant 2^n-1$), так что

$$I = \sum_{j=1}^{n} b_j(I)\, 2^{j-1}.$$

Например, $b_1(6)=0$, $b_2(6)=1$ и $b_3(6)=1$. Пусть p_j — множество неотрицательных целых чисел, в двоичных представлениях которых имеется 1 в j-м разряде. Таким образом, $p_j=\{I \mid b_j(I)=1\}$; например, $p_1=\{1, 3, 5, 7...\}$, $p_2=\{2, 3, 6, 7...\}$, $p_4=\{8, 9, 10, 11, 12, 13, 14, 15, 24, 25...\}$. Для y, обозначающего двоичный n-мерный вектор $(y_n, y_{n-1}, \ldots, y_1)$, и для операции $\oplus \sum_{i \in p_k} y_j$, обозначающей суммирование по модулю 2, определим $C_k(\mathrm{y}) = \oplus \sum_{j \in p_k} y_j$. Например, $C_1(1101)=0$, $C_2(1101)=1$. (Это соответствует уравнениям проверки четности.)

Определим теперь строчное множество R_I так: $R_I = \{\mathrm{y} \mid C(\mathrm{y}) = b_k(I)$ для $k=1, 2, \ldots, \mathrm{E}[\log_2 n]\}$. Например,

в случае таблицы из восьми строк R_0 содержит 0000000, 0110011 и 1010010, тогда как R_5 содержит 0010000, 0100011 и 1101000.

Для доказательства сцепленности строчных множеств введем еще одно определение. Пусть $I(+)J = \sum_{k=1}^{n} [b_k(I) \oplus \oplus b_k(J)]\, 2^{k-1}$. Эта сумма определяет $I(+)J$ как число, двоичное представление которого имеет 1 в тех разрядах, где двоичные представления чисел I и J отличаются, и эквивалентна другой форме записи:

$$b_k[I(+)J] = b_k(I) \oplus b_k(J) \text{ для } k = 1, 2, \ldots, n.$$

Например, $3(+)5=6$, $12(+)7=11$. Следующая теорема указывает на то, какие переменные должны измениться, чтобы был совершен переход от одного строчного множества к другому в данной паре строчных множеств.

Теорема 3.1. *Если* $\mathbf{y}^*$ *отличается от* $\mathbf{y}$ *лишь в* $(I(+)J)$*-м разряде и если* $\mathbf{y} \in R_1$ *то* $\mathbf{y}^* \in R_J$, (Если, например, $\mathbf{y}$ принадлежит R_3 и если мы изменим значение y_6, результирующий вектор $\mathbf{y}^*$ будет в R_5.)

Доказательство. 1. $b_k(I) \neq b_k(J)$ тогда и только тогда, когда $b_k[I(+)J] = 1$, в соответствии со второй формой определения операции $(+)$.

2. $b_k(I) \neq b_k(J)$ тогда и только тогда, когда $I(+)J \in P_k$. Это следует из п. 1 и определения множества P_k.

3. Поскольку, по предположению, $\mathbf{y} \in R_I$, то $C_k(\mathbf{y}) = b_k(I)$ для $k = 1, 2, \ldots, n$, тогда из п. 2 следует, что $C_k(\mathbf{y}) \neq b_k(J)$ тогда и только тогда, когда $I(+)J \in P_k$.

4. Если изменить y_m, то меняются лишь те C_k, для которых $m \in P_k$, что может быть видно из определения множества C_k. Таким образом, если меняется $y_{I(+)J}$, то при этом будут меняться лишь те C_k, для которых $I(+)J \in P_k$.

5. Но тогда из пп. 4 и 3 следует, что изменение $y_{I(+)J}$ приводит к изменениям лишь тех C_k, для которых $C_k(\mathbf{y}) \neq b_k(J)$. Таким образом, если $\mathbf{y}$ меняется на $\mathbf{y}^*$, имеем $C_k(y) = b_k(J)$ для всех k и, следовательно, $\mathbf{y}^* \in R_J$, что и требовалось доказать.

Кодирование соседними кодовыми комбинациями с использованием кодов Хэмминга, описанное выше, требует $2^{S_0} - 1$ внутренних переменных, так что число необходи-

мых переменных возрастает линейно в зависимости от числа строк, в отличие от кодирований раздела 3.1, в которых требуемое число переменных возрастало лишь как логарифм от числа строк.

Другим специальным типом кодирования соседними кодовыми комбинациями, который в ряде случаев оказывается полезным, является *кодирование отраженным кодом Грея*. Оно непосредственно применимо к таблицам переходов с 2^n состояниями, в которых переходы происходят между состояниями i и $i+1$ $(1 \leqslant i < 2^n)$ или между 2^n и 1. (Как станет дальше очевидным, допустимы также и другие переходы.) Каждой строке здесь приписывается одно y-состояние и всего используется n y-переменных. Кодовые комбинации, приписанные состояниям i и $i+1$ $(1 \leqslant i < 2^n)$, отличаются значением только одной y-переменной, как и кодовые комбинации, приписанные состояниям 1 и 2^n. Определим это кодирование рекурсивным образом.

1. Припишем строке 1 кодовую комбинацию 000...0, а строке 2 — комбинацию 100...0.

2. Допустим, что для некоторого $k \geqslant 1$ строкам с первой до 2^k-й приписаны кодовые комбинации такие, что $y_i = 0$ для всех $i > k$. Припишем строкам с (2^k+1)-й до 2^{k+1}-й кодовые комбинации следующим образом: для всех этих строк положим $y_{k+1} = 1$, положим также $y_i = 0$ для всех $i > k+1$. Для каждой строки 2^k+j $(1 \leqslant j \leqslant 2^k)$ положим $y_i (i \leqslant k)$ равным значению y_i в строке 2^k+1-j.

Термин «отраженный» обусловлен тем обстоятельством, что значения переменных $y_1, y_2, \ldots, y_k$ симметричны относительно линии, разделяющей строки 2^k и 2^k+1. Ниже показан такой код для восьми состояний.

Строка	y_1	y_2	y_3
1	0	0	0
2	1	0	0
3	1	1	0
4	0	1	0
5	0	1	1
6	1	1	1
7	1	0	1
8	0	0	1

3.3.2. Однозначное ОТП-кодирование в случае ОИВ-функций. В этом подразделе мы рассмотрим класс ОТП-кодирований, в которых могут происходить переходы без критических состязаний при изменении значений всех переменных, по которым различаются начальные и конечные состояния переходов. Будем рассматривать только ОИВ-функции и ограничимся лишь такими кодированиями, в которых каждой строке приписывается единственное y-состояние и которые поэтому будем называть *однозначными кодированиями*. Для обозначения однозначных ОТП-кодирований будем пользоваться сокращением «ООТП-кодирование». (Обобщения таких кодирований рассматриваются в подразделах 3.3.4 и 3.3.5.)

С помощью описываемого здесь метода строятся кодирования, специфические для определенного рода таблиц, и поскольку y-состояния, не приписанные строкам, могут появиться в качестве промежуточных состояний в тех или иных переходах в различных столбцах, эти кодирования можно классифицировать как кодирования с совместным использованием строк. Предполагается, что исходные таблицы переходов представлены в стандартной форме, являющейся результатом применения процедуры 2.5.

Считая $\mathscr{L}$ множеством y-состояний, определим $T(\mathscr{L})$ как множество состояний, *натянутое* на множество $\mathscr{L}$ и состоящее из всех тех y-состояний, которые содержатся в наименьшем подкубе, содержащем каждое y-состояние из $\mathscr{L}$. Другими словами, это множество состоит лишь из тех y-состояний, которые содержат нули в каждом таком разряде, в каком каждое состояние из $\mathscr{L}$ имеет 0, и единицы в каждом таком разряде, в каком каждое состояние из $\mathscr{L}$ имеет 1; например, T(00110, 10010, 00111) состоит из тех y-состояний, для которых $y_2=0$ и $y_4=1$, а именно: {00010, 00011, 00110, 00111, 10010, 10011, 10110, 10111}. Удобный способ определения такого множества состоит в том, чтобы выписать отдельное y-состояние с требуемыми зафиксированными значениями y-переменных и прочерками в местах, соответствующих остальным переменным. В нашем примере это дает —0—1—. Заполняя теперь пустые места единицами и нулями в каждой из 2^3 возможных комбинаций, получим требуемое множество.

Предположим, что каждой строке i исходной таблицы переходов приписано единственное y-состояние $\mathbf{y}^i$ и что

в некотором столбце имеется переход $i \to j$. Если $\mathbf{y}^i$ и $\mathbf{y}^j$ отличаются более чем одной переменной, то, в зависимости от распределения значений задержек в схеме, y-переменные, по которым различаются $\mathbf{y}^i$ и $\mathbf{y}^j$, могут при переходе измениться в любом порядке. Те же y-переменные, которые имеют одинаковые значения в $\mathbf{y}^i$ и $\mathbf{y}^j$, не меняются вовсе. Таким образом, в течение перехода система может пройти через любое состояние из $T(\mathbf{y}^i, \mathbf{y}^j)$. (Там, где не возникнет недоразумения, вместо $T(\mathbf{y}^i, \mathbf{y}^j)$ будем писать $T(i, j)$ и называть $T(i, j)$ *переходным множеством*.) Если $\mathbf{y}^i$ и $\mathbf{y}^j$ отличаются только одной y-переменной, то промежуточные состояния отсутствуют, что вытекает из соотношения $T(i, j) = \{\mathbf{y}^i, \mathbf{y}^j\}$. Если i устойчиво в столбце, содержащем i, так что $i=j$ (переход в этом случае тривиален: $i \to i$), то $T(i, j) = \{\mathbf{y}^i\}$.

Если в данной таблице переходов $N(i, I) = j$, то для того, чтобы схема оказалась свободной от критических состязаний и обладала свойством однотактности переходов, в соответствующей матрице переходов для каждого s, y-состояние которого принадлежит $T(i, j)$, $N(s, I)$ должно быть равным j. Отсюда следует, что если $i \to j$ и $k \to m$ есть переходы в одном и том же столбце таблицы переходов и $m \neq j$, то переходные множества $T(i, j)$ и $T(k, m)$ должны быть непересекающимися. В противном случае такие два перехода накладывали бы взаимно исключающие требования на следующие состояния матрицы переходов для общих y-состояний. Но если подобные противоречия отсутствуют в каждом столбце, то, очевидно, можно приписать значения следующих состояний для всех промежуточных состояний в каждом столбце таким образом, что будет гарантировано получение нужного конечного состояния на каждом переходе. Это устанавливает

Лемма 3.1. *Кодирование строк в ОИВ-таблице переходов не приводит к критическим состязаниям тогда и только тогда, когда для каждой пары переходов $i \to j$ и $k \to m$, имеющих место в одном и том же столбце и таких, что $j \neq m$ и $i \neq j$ или $k \neq m$, множества $T(i, j)$ и $T(k, m)$ непересекающиеся.*

Нашим следующим шагом является нахождение необходимых и достаточных для выполнения леммы 3.1 условий, которым должны удовлетворять внутренние переменные.

Считая U и V непересекающимися подмножествами строк таблицы переходов, определим *частичную дихотомию по состояниям* (которую дальше будем называть просто *дихотомией*) как неупорядоченную пару (U, V). Для данной пары переходов $i \to j$ и $k \to m$, где каждое i и j отлично от k и m, *связанной дихотомией* назовем пару (U, V), где $U = \{i, j\}$, $V = \{k, m\}$. Эта дихотомия обычно записывается как $\{ij, km\}$. Если один из переходов является вырожденным (скажем, $i=j$), получаем дихотомию (i, km), если же оба перехода вырождены, то связанная дихотомия превращается в (i, k). Будем говорить, что внутренняя переменная y_i в некотором кодировании *покрывает* дихотомию (U, V), если $y_i=0$ для каждого состояния из U и $y_i=1$ для каждого состояния из V (или наоборот). Докажем следующую теорему.

Теорема 3.2. *Кодирование ОИВ-таблицы является пригодным ООТП-кодированием тогда и только тогда, когда для каждой пары переходов $i \to j$ и $k \to m$, имеющих место в одном и том же столбце и таких, что $j \neq m$, связанная дихотомия (ij, km) покрывается по крайней мере одной y-переменной в этом кодировании.*

Доказательство. Если (ij, km) покрывается y-переменной y_c, то $y_c=0$ для каждого элемента из $T(i, j)$ и $y_c=1$ для каждого элемента из $T(k, m)$ (или наоборот), так что эти множества не пересекаются. Если это выполняется для каждой дихотомии, построенной в предположениях леммы 3.1, то в силу этой леммы критические состязания отсутствуют. Пусть две строки i и j таковы, что для некоторого входа I_k $N(i, I_k) \neq N(j, I_k)$, тогда имеется дихотомия, разделяющая i и j. Покрывающая y-переменная будет различать два y-состояния, приписанные этим строкам. Если не существует указанного входа I_k, то i и j совместимы и не имеют порождений. (Можно, конечно, рассматривать в этом случае дихотомию (i, j), если по каким-либо причинам желательно использовать разные коды для i и j.)

Чтобы доказать достаточность, нам следует рассмотреть условия, при которых два переходных множества могут быть непересекающимися. Как показано выше, каждое такое множество можно определить, зафиксировав значения тех y-переменных, которые имеют одни и те же значения в настоящем и следующем состояниях. Некоторое

y-состояние принадлежит данному переходному множеству тогда и только тогда, когда фиксированные переменные имеют те же значения в этом состоянии, что и в данном множестве. Два переходных множества не могут, очевидно, содержать одни и те же состояния лишь тогда, когда некоторая y-переменная является фиксированной в обоих множествах, но имеет противоположные значения.

Наоборот, допустим теперь, что каждая y-переменная либо фиксируется одним и тем же значением в обоих множествах, либо не фиксируется в каком-либо одном из этих множеств. (Такой парой множеств состояний являются, например, 010—0— и 0—010—.) Тогда независимо от того, какие переменные фиксированы в обоих множествах, имеется непустое пересечение этих множеств. (В нашем примере таким пересечением будет 01010—.) Отсюда следует, что если два переходных множества не пересекаются, должна быть хотя бы одна переменная, фиксированная нулем в одном множестве и единицей в другом. Такая переменная покрывает связанную дихотомию.

Таким образом, если для каждой пары переходов $i \to j$ и $k \to m$, удовлетворяющих условиям теоремы, соответствующие переходные множества $T(i, j)$ и $T(k, m)$ не пересекаются, то дихотомия (ij, km) должна быть покрыта некоторой y-переменной; это завершает доказательство теоремы.

Для всякой ОИВ-таблицы переходов можно получить множество дихотомий; после этого задача состоит в таком приписывании значений y-переменных, чтобы покрыть каждый элемент этого множества. Поскольку, очевидно, это можно сделать различными способами, обычно идут по пути, требующем минимального числа переменных. Прежде всего укажем на ряд обстоятельств, упрощающих задачу. Если U^* содержит U, а V^* содержит V, будем говорить, что дихотомия (U^*, V^*) *покрывает* дихотомию (U, V), и поскольку, очевидно, всякая y-переменная, покрывающая (U^*, V^*), должна покрывать и (U, V), то совершенно не обязательно включать (U, V) в список дихотомий. Кроме всего прочего, это означает, что если $i \to j$ есть переход в некотором столбце, нам никогда не нужно рассматривать переход $j \to j$ в этом столбце, поскольку всякая дихотомия, связанная с последним переходом, покрывается дихотомией, включающей в себя переход $i \to j$.

Далее, из списка может быть удалена дихотомия (i, km), порождаемая в некотором столбце, если в каком-либо другом столбце порождается дихотомия (ij, km).

Для иллюстрации введенных понятий, а также последующих шагов процедуры синтеза рассмотрим таблицу 3.9. Переходами в столбце A, порождающими непокрываемые дихотомии, являются переходы $1 \to \to 1$, $2 \to 3$ и $4 \to 5$; непокрываемыми являются дихотомии (1,23), (1,45) и (23,45) (полученные в результате рассмотрения всех пар переходов). Столбец B порождает десять дихотомий вида (i, j), каждая из которых покрывается «бóльшей» дихотомией из других столбцов. Столбец C имеет переходы $1 \to 4$, $3 \to 4$ и $5 \to 2$, порождающие две непокрываемые дихотомии (14,25) и (25,34), столбец D добавляет дихотомии (15,34) и (25,34); последнюю дихотомию следует удалить, так как она появилась второй раз. Ни (1,23), ни (1,45) не покрываются никакой из «бóльших» дихотомий, так что больше из списка ничего нельзя вычеркнуть.

Таблица 3.9

ОИВ-таблица переходов

	A	B	C	D
1	***1***,0	***1***,0	4,0	5,0
2	3,0	***2***,0	***2***,0	5,0
3	***3***,0	***3***,0	4,0	***3***,0
4	5,1	***4***,0	***4***,0	3,0
5	***5***,1	***5***,0	2,0	***5***,0

Теперь можно было бы приписать различные y-переменные так, чтобы покрыть шесть дихотомий и получить кодирование шестью переменными. Однако такое кодирование было бы слишком «расточительным», поскольку нетрудно найти пары дихотомий в этом списке, которые могут быть покрыты одной и той же y-переменной. (Например, (1,45) и (23,45) или (14,25) и (25,34).) Если две дихотомии можно покрыть одной и той же y-переменной, то всегда можно построить «бóльшую» дихотомию (такую, например, в которой увеличено по крайней мере одно из множеств-компонент), покрывающую обе первоначальных. Так, в нашем примере (134,25) покрывает и (14,25), и (25,34). Тогда задачу можно переформулировать как задачу нахождения минимальной совокупности дихотомий такой, что каждая дихотомия из исходного списка покрывается по крайней мере одной дихотомией из этой совокупности. Далее уже весьма просто приписать y-переменную каждому из членов минимальной покрывающей совокупности.

Таким образом, в итоге получим задачу, аналогичную задаче, рассматривавшейся в разделе 2.2, а именно, задачу нахождения максимальных совместимых множеств. Первое желание состоит в том, чтобы назвать две дихотомии совместимыми, если они обе могут быть покрыты некоторой дихотомией, и уже рассмотренными методами искать максимальные совместимые множества. Однако такая процедура совершенно неработоспособна, что вытекает, например, из рассмотрения трех дихотомий (1,23), (1,45) и (23,45). Все они попарно совместимы, и в соответствии с леммой 2.2 можно было бы ожидать, что найдется некоторая дихотомия, покрывающая все эти три дихотомии. Но такой дихотомии не существует, поскольку для того, чтобы покрыть (1,23) и (1,45), y-переменная должна иметь одно и то же значение для каждого состояния множества {2, 3, 4, 5}, но в этом случае она не будет покрывать дихотомию (23,45).

Положение можно облегчить, заменив каждую дихотомию (U, V) парой *упорядоченных дихотомий* $[U, V]$ и ее *дополнением* $\overline{[U, V]} = [V, U]$. Будем называть U и V соответственно *левым* и *правым множествами* дихотомии $[U, V]$. Две упорядоченные дихотомии считаются *совместимыми*, если никакое состояние не содержится в левом множестве одной дихотомии и в правом множестве другой.

Нетрудно показать, что совместимые упорядоченные дихотомии можно покрыть некоторой y-переменной, и лемма 2.2 применима в том смысле, что множество упорядоченных дихотомий можно покрыть какой-либо упорядоченной дихотомией тогда и только тогда, когда элементы этого множества попарно совместимы. Теперь можно найти максимальные совместимые множества (МС-множества) каким-либо из методов раздела 2.2 и выбрать из них минимальную совокупность таким образом, чтобы представитель от каждой дихотомии (какая-либо одна из упорядоченной пары дихотомий) покрывался по крайней мере одним элементом из этой совокупности. Дихотомии, очевидно, не имеют порождений, так что оставшаяся часть задачи может быть решена в терминах задачи покрытия с использованием понятия простых импликантов. В итоге выбирается покрывающая y-переменная для каждого из отобранных МС-множеств и строится матрица переходов

с соответствующими значениями в ее клетках для состояний каждого переходного множества.

Прежде чем рассматривать пример, введем одно полезное упрощение. Многие из исходных дихотомий обычно можно заменить одной упорядоченной дихотомией, что существенно сокращает объем вычислений, необходимых для нахождения МС-множеств. Этот прием основывается на следующей теореме.

Теорема 3.3. *Пусть D — множество упорядоченных дихотомий, полученное из некоторого множества неупорядоченных дихотомий. Для некоторого состояния s пометим через p_1, p_2 и т. д. дихотомии из D, имеющие s в своих левых множествах, и через q_1, q_2 и т. д.— дихотомии из D, не содержащие s ни в каком из своих множеств. Тогда минимальную совокупность МС-множеств, покрывающих каждую дихотомию из D, можно найти, рассматривая лишь упорядоченные дихотомии, помеченные символами p или q.*

(Будем считать, что, если некоторая дихотомия из D отмечена символом p_i, то ее дополнению сопоставляется символ $\bar{p}_i$.)

В примере с таблицей 3.9 (шесть дихотомий которой были выписаны ранее) можно выбрать $s=5$. (Это удачный выбор, поскольку 5 входит в наибольшее число дихотомий, так что наибольшим будет и число нерассматриваемых упорядоченных дихотомий.) Теорема утверждает, что вместо двенадцати упорядоченных дихотомий необходимо рассмотреть только семь, именно, дихотомии [1, 23] и [23, 1], помеченные символом q, и дихотомии [45, 1], [45, 23], [25, 14], [25, 34], [15, 34], помеченные символом p.

Доказательство. Если две упорядоченные дихотомии a и b совместимы, то совместимы также их дополнения $\bar{a}$ и $\bar{b}$. Следовательно, если каждую дихотомию в МС-множестве заменить ее дополнением, в результате получим снова МС-множество, покрывающее те же самые неупорядоченные дихотомии. Заметим также, что для любых заданных i и j никакое МС-множество из D не может содержать одновременно обе дихотомии, отмеченные символами p_i и $\bar{p}_j$ (поскольку, по определению, все дихотомии, помеченные символом p, содержат s в своих левых множествах, а все дихотомии, помеченные символы $\bar{p}$, содер-

жат s в своих правых множествах). Допустим теперь, что нам известно минимальное покрытие совокупности D. Если некоторое МС-множество в этом покрытии содержит дихотомии, отмеченные символом $\bar{p}$ ($\bar{p}$-дихотомии), мы можем заменить его МС-множеством без $\bar{p}$-дихотомий, полученным путем перехода к дополнениям соответствующих его дихотомий (к p-дихотомиям), и снова будем иметь минимальное покрытие. Но такая замена дает нам минимальное покрытие в терминах лишь p-дихотомий и q-дихотомий, которое могло бы быть получено непосредственно путем игнорирования дихотомий, помеченных символами $\bar{p}$.

Чтобы закончить наш пример, по семи отобранным упорядоченным дихотомиям построим карту пар, показанную в таблице 3.10, *а* (для удобства описания дихотомии помечены буквами). Несовместимые пары находятся легко, столбец за столбцом, следующим образом. Просматриваем столбец сверху вниз, ставя знак X в тех строках, в которых левое множество соответствующей дихотомии содержит элемент правого множества дихотомии, соответствующей рассматриваемому столбцу. (В столбце *а* знак X занесем, таким образом, в строки $\bar{a}$, d и e.) Затем повторяем процесс, рассматривая вместо левых правые и вместо правых левые множества. (Это добавляет знак X в строку b.)

Проведя подобное заполнение для всей таблицы (пустые клетки соответствуют совместимым парам), с помощью метода, такого же как в процедуре 2.2, находим МС-множества: ef, de, bc, $\bar{a}b$, $\bar{a}d$, ac, af. Хотя в этом случае минимальное покрытие может быть получено без всяких специальных процедур, весьма полезно использование таблицы покрытий, показанной в таблице 3.10, *б*. Столбцы в ней помечены буквами, соответствующими неупорядоченным дихотомиям, а строки соответствуют МС-множествам. В каждой строке значение 1 появляется лишь в тех столбцах, дихотомии которых покрываются МС-множеством, соответствующим данной строке. Минимальной совокупностью строк такой, что каждый столбец содержит 1 по крайней мере в одной из них, является совокупность $\{ef, \bar{a}d, bc\}$ (другой такой совокупностью является $\{af, de, bc\}$). Заметим, что нет необходимости включать

Таблица 3.10

a (1, 23)	$\bar{a}$ (23,1)	b (45, 1)	c (45, 23)	d (25,14)	e (25,34)	f (15, 34)
X						
X						
	X					
X		X	X			
X	X	X	X			
	X	X	X	X		

а) Карта пар для нахождения МС-множеств

Неупорядоченные дихотомии

МС-множества	a	b	c	d	e	f
ef					1	1
de				1	1	
bc		1	1			
$\bar{a}b$	1	1				
$\bar{a}d$	1			1		
ac	1		1			
af	1					1

б) Таблица покрытий

в нашу совокупность одновременно МС-множества, покрывающие a, и МС-множества, покрывающие $\bar{a}$.

В соответствии с первым решением имеем покрывающие дихотомии [125, 34], [235, 14] и [45, 123], по которым построена матрица переходов, представленная в таблице 3.11, где переходы в состояния α_1, α_2 и α_3 могут иметь место только из-за состязаний. В столбце A, например, α_1 и α_2 являются состояниями множества $T(4,5)$; сле-

довательно, поскольку они могут появиться при переходе $4 \to 5$, в клетках $\alpha_1 - A$ и $\alpha_2 - A$ матрицы переходов следует поместить следующее состояние 5 и выход 1.

Описанный метод можно формализовать в виде следующей процедуры.

Таблица 3.11

Матрица переходов для таблицы 3.9

	A	B	C	D	y_1	y_2	y_3
1	***1***,0	***1***,0	4,0	5,0	0	1	1
2	3,0	***2***,0	***2***,0	5,0	0	0	1
3	***3***,0	***3***,0	4,0	***3***,0	1	0	1
4	5,1	***4***,0	***4***,0	3,0	1	1	0
5	***5***,1	***5***,0	2,0	***5***,0	0	0	0
α_1	5,1	—	4,0	5,0	0	1	0
α_2	5,1	—	4,0	3,0	1	0	0
α_3	—	—	4,0	3,0	1	1	1

Процедура 3.1. *Нахождение ООТП-кодирования с минимальным числом y-переменных для ОИВ-таблиц переходов* (будем называть эту процедуру — *метод Трэйси*).

1. Отметить переходы вида $i \to i$ в каждом таком столбце, в котором i устойчиво и не является следующим состоянием ни в каком другом переходе. (Не надо как-то специально выделять состояния, для которых не определены следующие состояния, поскольку, если кодирование удовлетворяет теореме 3.2, такие состояния никогда не появятся в переходных множествах состояний при переходах с различными конечными состояниями).

2. Для каждого столбца найти дихотомии (ij, km) для каждой пары переходов $i \to j$ и $k \to m$, где $j \neq m$.

3. Выбрать состояние s, встречающееся наибольшее число раз в совокупности дихотомий, найденных в соответствии с п. 2. Каждой дихотомии из этой совокупности сопоставить пару упорядоченных дихотомий. Образовать список упорядоченных дихотомий следующим образом. Для тех дихотомий совокупности, которые не содержат s, в список включить соответствующие им пары упорядоченных дихотомий. Для остальных дихотомий, содержащих s,

в список включить по одному элементу пары, содержащему s в левом множестве.

4. Используя таблицу пар и какой-либо метод раздела 2.2, найти МС-множества, соответствующие упорядоченным дихотомиям.

5. Применяя, если необходимо, таблицу покрытий, найти минимальную совокупность МС-множеств, которые покрывают каждую дихотомию.

6. Беря поочередно объединения всех левых и всех правых множеств для каждого МС-множества п. 5, построить совокупность покрывающих дихотомий.

7. Построить матрицу переходов, приписав покрывающую y-переменную каждой из покрывающих дихотомий. В каждом столбце I для каждого перехода $i \to j$ найти множество $T(i, j)$ и, если оно не пусто, ввести в матрицу дополнительные строки, заполнив их клетки в столбце I так, как заполнена клетка i—I.

На шаге 7 часто случается, что некоторые из покрывающих дихотомий не содержат всех состояний, например, покрывающей дихотомией для таблицы переходов с шестью строками может быть дихотомия (12,456). Здесь значение покрывающей переменной в строке 3 можно сделать равным как 0, так и 1. Такая неоднозначность обусловлена тем, что (12,456) покрывается и (123,456), и (12,3456), так что может быть использована каждая из этих дихотомий. Записывая прочерк в строке 3 в столбце y-значений для покрывающей переменной, мы оставляем возможность последующего выбора значения этой переменной, что может быть использовано позднее при рассмотрении каких-либо других условий. Можно, конечно, прочерк трактовать и как 0, и как 1 одновременно, так что строке 3 будут приписаны несколько y-состояний. При этом переход в состояние 3 можно рассматривать как переход в то или иное из этих состояний, что обеспечивает возможность выбора при задании переходов. (Ясно, что кодирование в этом случае перестает быть однозначным кодированием.)

Метод Трэйси приводит к ООТП-кодированию с минимальным числом внутренних переменных (хотя часто требует большого объема вычислений), однако полезно рассмотреть другой подход к той же задаче, который мы назовем — *метод Лю*. Кодирования по этому методу не всегда минимальны по числу используемых y-переменных, но

зато они обладают такими свойствами, которые важны в ряде вопросов, рассматриваемых в последующих главах.

Напомним, что множества предшественников в каждом столбце разбивают совокупность строк на непересекающиеся множества в соответствии с устойчивыми состояниями каждого столбца (см. раздел 3.2). Например, множествами предшественников в столбце I_2 таблицы 3.12 являются множества {12}, {356} и {47}. Идея, лежащая в основе метода Лю, состоит в том, чтобы каждому входному

Таблица 3.12

ОИВ-таблица переходов

	I_1	I_2	I_3	Основной вариант кодирования по Лю y_1	y_2	y_3	y_4	y_5	y_6	y_7
1	***1***,0	2,0	***1***,0	0	0	0	0	0	0	0
2	3,0	***2***,0	5,1	1	0	0	0	0	0	1
3	***3***,0	6,0	***3***,0	1	0	0	0	1	1	0
4	***4***,0	7,0	3,0	1	1	1	1	0	1	0
5	***5***,0	6,0	***5***,1	1	1	0	0	1	0	1
6	4,0	***6***,0	5,1	1	1	1	0	1	0	1
7	***7***,1	***7***,0	1,0	0	1	0	1	0	0	0

столбцу I_i поставить в соответствие некоторое подмножество V_i y-переменных, а каждому множеству предшественников для столбца I_i — единственное состояние (комбинацию значений) y-переменных из V_i. Такое кодирование показано в таблице 3.12, где V_1, V_2 и V_3 суть $y_1y_2y_3$, y_4y_5 и y_6y_7 соответственно.

Если в столбце I_n имеются переходы $i \to j$ и $k \to m$, $j \neq m$, то i и j принадлежат D_{nj}, а k и m принадлежат D_{nm}. Поскольку состояния y-переменных из V_n неодинаковы для D_{nm} и D_{nj}, то в V_n должна быть некоторая y-переменная, покрывающая дихотомию (ij, km). Таким образом, условия теоремы 3.2 выполняются.

Известно несколько разновидностей метода Лю, в которых возможно уменьшение числа необходимых внутренних переменных. Прежде всего, необязательно, чтобы множества V_j и V_k не пересекались при $j \neq k$, так что если y-переменные в каждом из двух V-множеств имеют

только одинаковые или только противоположные значения для каждой строки, следует рассматривать лишь какую-либо одну из таких переменных.

Допустим, далее, что число множеств предшественников для I_i равно k, где k не является степенью двойки. Тогда в V_i необходимы $E[\log_2 k]=n$ внутренних переменных, и число (2^n) всех состояний y-переменных из V_i превышает число множеств предшественников. При этом возможно частичное определение значений некоторых переменных из V_i. Например, в таблице 3.12 имеется три множества предшественников в столбце I_3, так что нулевые значения, приписанные переменной y_7 в строках 3 и 4 (элементы множества D_{33}), можно заменить на прочерки, поскольку всякий раз, когда y_6 (другая переменная из V_3) равна 1, множеством предшественников является D_{33}. Если сделать указанную замену, то значения y_1 удовлетворяют требуемым значениям для y_7 и, следовательно, y_1 может заменить y_7 в V_3, оставаясь, конечно, в V_1.

Другой способ, используемый для уменьшения числа внутренних переменных в кодировании по Лю, назван здесь *объединение предшественников.* (Существует по крайней мере одно применение кодирования по Лю, описанное в разделе 5.4, где этот способ непригоден.) Способ объединения предшественников основывается на том, что в теореме 3.2 не требуется, чтобы y-переменные из V_i различали все одноэлементные множества предшественников для столбца I_i. Это делает возможным перед началом приписывания y-переменных из V_i сжатие (объединение) всех или части одноэлементных множеств предшественников любого столбца в некоторое меньшее число более крупных множеств. Например, в таблице 3.12 можно рассматривать множества D_{11}, D_{15} и D_{17} как единое трехэлементное множество. Это будет означать уменьшение множества V_1 до двух элементов.

Подобные соображения реализованы в таблице 3.13, где (в y-столбцах, озаглавленных «начальное кодирование») множества V_1, V_2 и V_3 состоят соответственно из y_1y_2, y_3y_4 и y_5y_6. Три одноэлементных множества предшественников для I_1 объединяются в одно множество, которое становится наибольшим по величине среди всех множеств предшественников, и ему приписывается состояние (или, точнее, состояния) 0 — из V_1. (Приписывание

неполностью определенных состояний наибольшим множествам предшественников является довольно выгодной стратегией, так как при этом максимизируется число прочерков в y-столбцах и, следовательно, число вариантов

Таблица 3.13

Упрощенное кодирование по Лю

	I_1	I_2	I_3	Начальное кодирование y_1	y_2	y_3	y_4	y_5	y_6	Сжатое кодирование y_1	y_2	y_3	y_4
1	***1***,0	2,0	***1***,0	0	—	0	0	0	0	0	0	0	0
2	3,0	***2***,0	5,1	1	0	0	0	1	—	1	0	0	1
3	***3***,0	6,0	***3***,0	1	0	1	—	0	1	1	0	1	0
4	***4***,0	7,0	3,0	1	1	0	1	0	1	1	1	0	0
5	***5***,0	6,0	***5***,1	0	—	1	—	1	—	0	—	1	1
6	4,0	***6***,0	5,1	1	1	1	—	1	—	1	1	1	1
7	***7***,1	***7***,0	1,0	0	—	0	1	0	0	0	1	0	0

объединения y-столбцов.) Для I_2 V_2-состояниями, приписанными множествам D_{22}, D_{26} и D_{27}, являются соответственно 00, 1— и 01.

После завершения начального кодирования, требующего шести переменных, делается попытка найти такие y-столбцы, которые могли бы заменить два или более исходных столбцов. Например, y_1 отличается от y_6 только прочерками в некоторых местах, где y_6 имеет определенные значения, так что y_1 можно использовать для замены элемента y_6 в V_3. Аналогично, y_2 и y_4 совпадают с точностью до незаполненных мест, так что существует y-столбец, покрывающий и y_2, и y_4. Указанные две пары y-столбцов заменяются соответственно на столбцы y_1 и y_2, помещенные под заглавием «сжатое кодирование», где столбцы y_3 и y_4 являются взятыми без изменения столбцами y_3 и y_5 начального кодирования.

Задача нахождения минимальной совокупности y-столбцов, покрывающих все y-столбцы в исходном кодировании, полностью аналогична рассмотренной ранее задаче нахождения минимальной совокупности покрывающих дихотомий. Мы должны рассмотреть каждый столбец

и его дополнение (в этом случае применима теорема 3.3, что облегчает нашу задачу), назвав два столбца *совместимыми*, если они не отличаются друг от друга ни в одной строке с определенными значениями. Каждое МС-множество минимального покрытия заменяется одним покрывающим столбцом, в котором значение в строке i берется равным 0 (1), если хотя бы один из совмещаемых столбцов имеет в этой строке значение нуль (единица), и не определено, если оба совмещаемых столбца не определены в строке i.

В силу наличия вариантов выбора при приписывании V-состояний каждому множеству D_{ij}, приведенная выше процедура не всегда приводит к кодированию по Лю минимальным числом переменных. Определим *обобщенное кодирование по Лю* как такое, в котором для каждого входа I_i и каждой пары состояний p и q, которые устойчивы в I_i, дихотомия (D_{ip}, D_{iq}) покрывается некоторой y-переменной. Все кодирования по Лю, получаемые с помощью описанного выше метода, удовлетворяют этому условию, которое, конечно, предполагает выполнение условий теоремы 3.2.

Обобщенные кодирования по Лю минимальным числом переменных можно найти с помощью процедуры 3.1, в которой вместо дихотомий, которые строятся на этапах 1 и 2 процедуры, берутся дихотомии вида (D_{ip}, D_{iq}). Хотя в общем случае этот подход приводит к использованию меньшего числа y-переменных, чем в рассмотренном ранее методе Лю, число переменных, связанных с каждым входным столбцом, уже не может быть минимальным, что, конечно, является недостатком в некоторых случаях (см. раздел 5.4).

Все кодирования по Лю характеризуются тем, что для каждого входа I_i и каждого состояния p, устойчивого в I_i, имеется единственный подкуб в y-пространстве, который содержит $T(D_{ip})$ и не пересекается с $T(D_{iq})$ для всех других состояний q, устойчивых в I_i. Такое свойство, важное в ряде приложений (см. подраздел 6.1.2), не присуще другим типам кодирований, удовлетворяющим теореме 3.2, что может быть проиллюстрировано тем фактом, что в таблице 3.11 переходы в $4 - C$ имеют место внутри двух различных, хотя и перекрывающихся подкубов y-пространства.

Объем вычислений, необходимых для нахождения минимальных кодирований по Лю, в общем случае меньше, чем при применении метода Трэйси, поскольку число дихотомий, которые следует рассмотреть, здесь не может быть бо́льшим, а часто намного меньше, чем во втором случае. Однако за это иногда приходится расплачиваться увеличением числа необходимых y-переменных, что обусловлено потерей свободы при покрытии некоторых дихотомий. Так, например, в минимальном ООТП-кодировании, найденном для таблицы 3.11, дихотомия (134,25) не покрывается, а (25,34) и (25,14) покрываются двумя различными y-переменными, тогда как в кодировании по Лю дихотомия (134,25) должна быть обязательно покрыта.

В следующем подразделе рассматривается вопрос о числе внутренних переменных, необходимых для реализации «наихудшей» ОИВ-таблицы переходов с n строками в случае ООТП-кодирования.

3.3.3. Оценки числа внутренних переменных, необходимых в ООТП-кодировании. Кодирование соседними кодовыми комбинациями, описанное в подразделе 3.3.1, определяет оценку для числа внутренних переменных, необходимых в универсальном ОТП-кодировании, равную 2^{S_0} — —1. Ниже показывается, что эту оценку можно существенно понизить, если отбросить свойство соседства кодовых комбинаций. Вначале неконструктивным образом устанавливается некоторая верхняя оценка и далее описывается конструктивная процедура построения универсального ООТП-кодирования, число необходимых внутренних переменных в котором лежит между двумя найденными границами.

Теорема 3.4. *Существует универсальное ООТП-кодирование n переменными для ОИВ-таблиц переходов такое, что* $n=21S_0+9$.

Доказательство. Определим вначале максимальное число вариантов кодирований n переменными, непригодных для любой таблицы с r строками. Очевидное следствие теоремы 3.2 состоит в том, что кодирование пригодно для любой таблицы с r строками тогда и только тогда, когда оно покрывает каждую дихотомию вида (ij, km), где i, j, k и m суть различные строки.

1. Прежде всего рассмотрим, какими из y-столбцов нельзя покрыть некоторую дихотомию (ab, cd) r-строчной

таблицы. Это те столбцы, которые не содержат значения 0, 0, 1, 1 или 1, 1, 0, 0 в строках a, b, c и d соответственно. Число таких столбцов $(2^4-2)2^{r-4}=14\cdot 2^{r-4}=(7/8)2^r$.

2. Полное число вариантов кодирований n переменными, которые можно построить, выбирая n таких y-столбцов (с повторениями), равно $[(7/8)2^r]^n=(7/8)^n 2^{rn}$.

3. Число различных дихотомий вида (ij, km) в таблице с r строками равно $3\binom{r}{4}$.

4. Число различных вариантов кодирований n переменными, в которых не покрывается по крайней мере одна дихотомия вида (ij, km), должно быть меньше (благодаря перекрытиям), чем $3\binom{r}{4}(7/8)^n 2^{rn}$.

5. Полное число различных вариантов кодирования n переменными для таблиц с r строками равно $(2^r)^n=2^{rn}$.

6. Таким образом, число различных вариантов кодирований n переменными, которые покрывают все дихотомии вида (ij, km) и в силу этого пригодны для всех ОИВ-таблиц переходов с r строками, больше, чем $V = 2^{rn} - 3\times$

$\times\binom{r}{4}(7/8)^n 2^{rn}$.

7. Если при заданных r и n V неотрицательно, то существует по крайней мере один вариант кодирования рассматриваемого типа, который пригоден для любой ОИВ-таблицы с r строками. Чтобы получить желаемую оценку числа таких вариантов, нужно найти наименьшее n как функцию от r, для которого V не меньше нуля.

8. Считая $V\geqslant 0$, получим:

$$2^{rn} - 3\binom{r}{4}(7/8)^n 2^{rn}\geqslant 0,$$

после сокращения на 2^{rn} имеем

$$\left(\frac{7}{8}\right)^n \leqslant \left[3\binom{r}{4}\right]^{-1}.$$

9. Взяв логарифм (по основанию 2) и разрешив неравенство относительно n, получим:

$$n\leqslant \frac{\log_2[3r(r-1)(r-2)(r-3)]}{\log_2(8/7)},$$

10. Заменяя $r(r-1)(r-2)(r-3)$ на r^4 и считая, что $\log_2(8/7) \simeq 0{,}193$, а $\log_2 3 \simeq 1{,}59$, имеем окончательно:

$$n \leqslant \frac{4\log_2 r + 1{,}59}{0{,}193} \leqslant 21\log_2 r + 9 = 21 S_0 + 9,$$

что и требовалось доказать.

Эта оценка не является слишком жесткой прежде всего в силу п. 4, она превышает даже оценку $2^{S_0} - 1$, пока S_0 не более 8. Тем не менее она говорит о том, что оценки для ООТП-кодирований возрастают линейно с ростом S_0, как и оценки для общего вида строчных кодирований, а не экспоненциально, как оценка для кодирования соседними кодовыми комбинациями. Заметим, что пока нет процедур для построения $(21S_0+9)$-кодирований.

Опишем конструктивное доказательство того, что $(S_0^3 + 5S_0)/6$ является также оценкой в рассматриваемой задаче. Это доказательство интересно само по себе, кроме того, для $S_0 \leqslant 11$ оно дает лучшую оценку, чем теорема 3.4.

Начнем с определения *разделяющей системы* $S_{i,j}(S_0)$ как множества y-переменных, приписанных множеству состояний с 2^{S_0} элементами, такого, что всякая пара P и Q непересекающихся и содержащих элементы i и j соответственно подмножеств этих состояний покрывается y-переменной из этого множества. Необходимое число y-переменных в $S_{i,j}(S_0)$ обозначается через $N_{i,j}(S_0)$. Заметим, что ООТП-кодирование, пригодное для любой ОИВ-таблицы с 2^{S_0} состояниями, образует разделяющую систему $S_{2,2}(S_0)$. Покажем, что $N_{2,2}(S_0) \leqslant (S_0^3 + 5S_0)/6$.

Ясно, что $N_{1,1}(S_0) = S_0$, поскольку достаточно S_0 y-состояний для разделения всех 2^{S_0} состояний системы (другими словами, для покрытия всех пар (i, j), где i и j — различные состояния системы).

На основе блочной матрицы R, представленной в таблице 3.14, покажем, что $N_{1,2}(S_0) \leqslant S_0(S_0+1)/2$. Столбцы матрицы соответствуют y-переменным, а строки — состояниям. Подматрица A представляет разделяющую систему $S_{1,1}(S_0-1)$, $\bar{A}$ представляет A, в которой все элементы заменены их отрицаниями, а B представляет разделяющую систему $S_{1,2}(S_0-1)$. Теперь необходимо показать, что для любых трех состояний i, j и k некоторый столбец

из R покрывает (ij, k). Пусть $m=2^{S_0-1}$, а i^*, j^* и k^* равны соответственно i, j, k по модулю m. Тогда, если i^*, j^* и k^* различны, то в B есть столбец, который покрывает (i^*j^*, k^*), и соответствующий столбец в R покрывает (ij, k). Далее остаются случаи, когда $k=i$ (или j) по модулю m, т. е. такие, когда i и k (или j) находятся в соответствующих строках верхней и нижней половин R. Без потери общности предположим, что k находится в нижней половине. Тогда получаем следующие случаи (в которых i и j меньше, чем m):

1. $(ij,\ i+m)$: покрывается крайним правым столбцом в R.

2. $(i(i+m),\ j)$: покрывается тем столбцом из R, который соответствует столбцу из B, покрывающему (i, j).

3. $(i(j+m),\ i+m)$: покрывается тем столбцом из R, который соответствует столбцу из A, покрывающему (i, j).

Таблица 3.14

$S_{1,2}(S_0)$-система

$R=$

A	B	$\begin{matrix}0\\0\\\cdot\\\cdot\\\cdot\\0\end{matrix}$
$\overline{A}$	B	$\begin{matrix}1\\1\\\cdot\\\cdot\\\cdot\\1\end{matrix}$

Таким образом, R действительно описывает разделяющую систему $S_{1,2}(S_0)$. Вычислим рекурсивным образом значения $N_{1,2}(S_0)$, соответствующие R:

1. $N_{1,2}(1)=1$ (для таблицы переходов с двумя строками достаточно одной y-переменной).

2. $N_{1,2}(S_0)\leqslant N_{1,2}(S_0-1)+N_{1,1}(S_0-1)+1$, $N_{1,2}(S_0)\leqslant N_{1,2}(S_0-1)+S_0$ (по построению R).

Это приводит к следующему:

$$N_{1,2}(S_0)\leqslant\sum_{i=1}^{S_0} i=\frac{S_0(S_0+1)}{2}.$$

Теперь, используя аналогичный подход, построим систему $S_{2,2}(S_0)$, также соответствующую матрице R (табл. 3.14), где уже A является системой $S_{1,2}(S_0-1)$, $\overline{A}$ — ее дополнением, а B — системой $S_{2,2}(S_0-1)$. Пригодность матрицы R устанавливается, как и ранее, путем рассмотрения различных случаев, среди которых выделим десять основных. Здесь уже дихотомии, которые необходимо покрывать, имеют вид (ij, pq).

1. Если i^*, j^*, p^*, q^* все различны, то необходимый покрывающий столбец есть в B.

В шести последующих случаях предполагается $i^*=q^*$. Мы будем указывать лишь покрывающий столбец, предоставляя читателю самому убедиться в этом (считаем, что все i, j и p меньше q).

2. $(pj,\ i(i+m))$: B-столбец, покрывающий $(pj,\ i)$;

3. $(p(j+m),\ i(i+m))$: B-столбец, покрывающий $(pj,\ i)$;

4. $(ij,\ p(i+m))$: A-столбец, покрывающий $(ij,\ p)$;

5. $(ij,\ (p+m)(i+m))$: крайний правый столбец;

6. $(i(j+m),\ p(i+m))$: A-столбец, покрывающий (i, jp);

7. $(i(j+m),\ (p+m)(i+m))$: A-столбец, покрывающий $(ip,\ j)$;

В последних трех случаях считается, что в $(i^*,\ j^*,\ p^*,\ q^*)$ имеется лишь два различных элемента.

8. $(ij,\ (i+m)(j+m))$: крайний правый столбец;

9. $(i(i+m),\ j(j+m))$: B-столбец, покрывающий $(i,\ j)$;

10. $(i(j+m),\ j(i+m))$: A-столбец, покрывающий $(i,\ j)$.

Ясно, что $N_{2,2}(1)=1$, так что по построению и с учетом нашего предыдущего результата получим рекурсивное соотношение:

$$N_{2,2}(S_0) \leqslant N_{2,2}(S_0-1)+N_{1,2}(S_0-1)+1,$$

$$N_{2,2}(S_0) \leqslant N_{2,2}(S_0-1)+\frac{S_0(S_0-1)}{2}+1.$$

В итоге получим:

$$N_{2,2}(S_0) \leqslant 1+\sum_{i=2}^{S_0}\left[1+\frac{i(i-1)}{2}\right]=\frac{S_0^3+5S_0}{6},$$

что и требовалось доказать. Таким образом, справедлива следующая теорема.

Теорема 3.5. *Существует конструктивная процедура построения универсального ООТП-кодирования для ОИВ-таблиц переходов, в которой требуется* $(S_0^3+5S_0)/6$ *внутренних переменных.*

Эта оценка никогда не превышает оценки вида $2^{S_0}-1$, становится меньше нее при $S_0=4$ и остается меньшей при дальнейшем увеличении S_0. Если $S_0 \leqslant 11$, то эта оценка

также меньше оценки $21S_0+9$. Недавно была разработана более сложная рекурсивная процедура, несколько подобная описанной выше, в которой для различных бесконечных последовательностей значений S_0 выполняется $N_{1,2}(S_0) \leqslant S_0^{1,59}$ и $N_{2,2}(S_0) \leqslant S_0^2$.

Можно также кратко отметить ряд дополнительных результатов, касающихся небольших таблиц переходов. Непосредственное применение описанных выше методов к совокупности из пятнадцати дихотомий вида (ij, km), возможной в таблицах с пятью строками, показало, что для построения универсального ООТП-кодирования ОИВ-таблиц этого класса необходимо и достаточно шести внутренних переменных. Более тонкие рассуждения приводят к выводу, что для построения аналогичных универсальных кодирований для таблиц переходов с шестью строками необходимо и достаточно семи внутренних переменных.

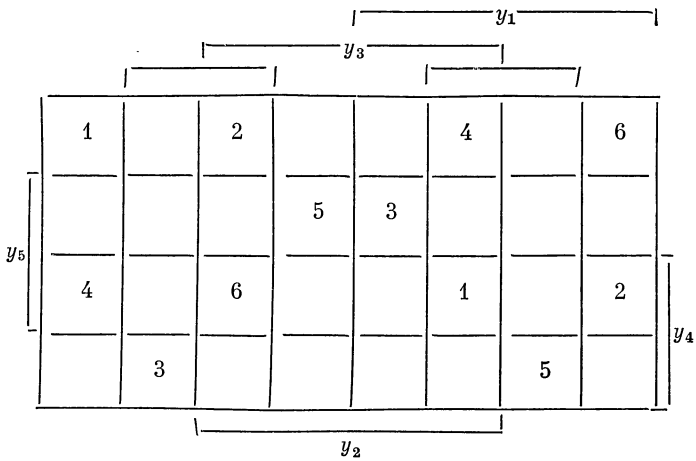
В следующем разделе будет показано, что путем введения неоднозначного кодирования можно найти как универсальные кодирования, так и кодирования для конкретных таблиц переходов такие, в которых требуется меньше внутренних переменных, чем в лучших вариантах однозначных кодирований для соответствующих случаев.

3.3.4. Многозначное ОТП-кодирование. Рассмотрим вначале вариант кодирования, показанный в таблице 3.15, в котором используется пять внутренних переменных и каждой из шести строк приписывается пара дополняющих друг друга y-состояний. Если i, j, k и m суть четыре различных состояния, читатель может убедиться в том, что для каждого i-состояния наименьший подкуб на карте, содержащий это состояние и одно из j-состояний, не пуст и не перекрывается с минимальным подкубом, содержащим какое-либо k-состояние и ближайшее m-состояние. Таким образом, безотносительно к тому, в каком i-состоянии находится система, имеется j-состояние такое, что соответствующее множество $T(i, j)$ не пересекается с другими множествами $T(k, m)$. Например, переход между 2-состоянием 01100 и ближайшим 5-состоянием 01001 можно сделать таким образом, чтобы не проходить через область, относящуюся к переходу $4 \longrightarrow 6$, начинающемуся или в 00011, или в 11100.

Такое кодирование является, следовательно, универсальным ОТП-кодированием, пригодным для ОИВ-таблиц

Таблица 3.15

Универсальное многозначное ОТП-кодирование пятью переменными для ОИВ-таблиц с шестью строками



с шестью строками, причем здесь требуется на две внутренних переменных меньше, чем в лучшем из соответствующих однозначных кодирований. Можно найти также универсальное ОТП-кодирование одиннадцатью переменными для таблиц переходов с четырнадцатью строками, тогда как в лучшем из соответствующих однозначных кодирований, скорее всего, потребуется бóльшее число переменных (это, однако, не доказано). Эти варианты были получены специально и служат, в основном, для иллюстрации возможностей, открывающихся при использовании более чем одного y-состояния на строку.

Обратимся теперь к таблице 3.16, *а*, которая описывает ОИВ-функцию. Применение методов подраздела 3.3.2 показывает, что для ООТП-кодирования в этом случае потребуется по крайней мере четыре переменных. Однако вариант кодирования с тремя переменными, показанный в таблице 3.16, *б*, в котором строке 1 приписаны два y-состояния, является пригодным ОТП-кодированием для таблицы 3.16, *а*. Более глубокий анализ этого примера обеспечит нам основу для построения общего алгоритма.

Таблица 3.16

	I_1	I_2	I_3	I_4	I_5
1	***1***,0	***1***,0	***1***,0	4,0	***1***,0
2	1,0	6,0	3,0	***2***,0	***2***,0
3	***3***,0	1,0	***3***,0	4,0	2,0
4	3,0	***4***,0	1,0	***4***,0	***4***,0
5	***5***,1	***5***,0	6,0	***5***,0	***5***,0
6	5,1	***6***,0	***6***,0	2,0	***6***,0

а) ОИВ-таблица переходов

	y_1			
$1a$	2	3	4	
5	6	$1b$		y_3
y_2				

б) Многозначное кодирование

	I_1	I_2	I_3	I_4	I_5
11	***11***,0	***11***,0	***11***,0	4,0	***11***,0
12	***12***,0	***12***,0	***12***,0	4,0	***12***,0
13	***13***,0	***13***,0	***13***,0	4,0	***13***,0
24	11,0	62,0	33,0	***24***,0	***24***,0
25	11,0	62,0	33,0	***25***,0	***25***,0
31	***31***,0	12,0	***31***,0	4,0	25,0
33	***33***,0	12,0	***33***,0	4,0	25,0
4	31,0	***4***,0	13,0	***4***,0	***4***,0
5	***5***,1	***5***,0	63,0	***5***,0	***5***,0
62	5,1	***62***,0	***62***,0	24,0	***62***,0
63	5,1	***63***,0	***63***,0	24,0	***63***,0

в) Расширенная таблица переходов

Ясно, что дихотомии (12, 34) и (13, 26), обусловленные переходами в столбцах I_1 и I_2 соответственно, несовместимы. Но если мы расщепим строку 1 на $1a$ и $1b$ и заменим переход $2 \longrightarrow 1$ в столбце I_1 на $2 \longrightarrow 1a$, а переход $3 \longrightarrow 1$ в столбце I_2 на $3 \longrightarrow 1b$, то указанные выше дихотомии примут вид: ($1a$2, 34) и ($1b$3, 26), а соответствующие им упорядоченные дихотомии [$1a$2, 34] и [26, $1b$3] будут совместимы. В приведенном многозначном кодировании фактически сделано такое расщепление строк, там y_1 покрывает [$1a$256, $1b$34]. Использованный здесь частный вид расщепления строк позволяет состоянию, устойчивому в двух столбцах, приписывать различные кодовые комбинации в этих столбцах.

Для того чтобы перейти от этой идеи к общему алгоритму, можно начать с расширения исходной таблицы в

эквивалентную ей таблицу такую, что если состояние i исходной таблицы устойчиво в столбцах I_1 и I_2 и в них есть переходы в состояние i, то в расширенной таблице состоянию i соответствуют состояния i_1 и i_2, каждое из которых устойчиво в обоих столбцах I_1 и I_2, а все переходы в состояние i в столбце I_1 (или I_2) заменяются переходами, ведущими в i_1 (или i_2). Расширенная таблица для нашего примера представлена таблицей 3.16, *в*, в которой образом состояния i в столбце I_k является ik. Затем к расширенной таблице применяется процедура 3.1 с тем отличием, что некоторые дихотомии не используются. В частности, состояния ij и ik рассматриваются как эквивалентные (каковыми они и являются); поэтому для столбца I_k не нужно строить дихотомии такие, для которых переходные множества $T(p, ik)$ не содержат состояние ij. Так, в нашем примере не нужно выписывать дихотомию ((12)(33), 11), что обычно следовало бы делать, исходя из значений элементов столбца I_2. Следовательно, паре строк таких, как ij и ik, полученных из строки i исходной таблицы, можно приписать одно и то же y-состояние, что в какой-то степени обратно процессу расширения таблицы переходов. В нашем примере подобный эффект «сжатия» проявляется в значительной степени.

Каждая дихотомия исходной таблицы переходов порождает множество взаимно совместимых дихотомий расширенной таблицы; следовательно, всякое покрытие исходной таблицы соответствует покрытию (с тем же числом y-переменных) расширенной таблицы. Обратное, конечно, неверно, что показано на примере. Следовательно, при использовании процедуры расширения результирующее кодирование имеет не больше, а, может быть, меньше внутренних переменных, чем наилучшее из однозначных кодирований для исходной таблицы. Однако этот метод не всегда приводит к наилучшему многозначному кодированию. В нем, например, не используется то предположение, что, может быть, было бы лучше, если бы некоторые переходы в устойчивое состояние в каком-либо столбце расширенной таблицы имели в качестве конечных различные эквивалентные состояния, а не одно состояние. Показано, что при этом предположении можно получить уменьшение числа y-переменных, невозможное при других способах. Однако учет этого предположения весьма

загромождает процедуру, которая и без того усложнилась необходимостью обработки расширенной таблицы, и нет уверенности, что таким путем может быть достигнут оптимальный результат. Следовательно, задача нахождения алгоритма, приводящего к ОТП-кодированию с наименьшим числом внутренних переменных, остается пока нерешенной.

3.3.5. Кодирование состояний для МИВ-таблиц переходов. Зависящие от времени МИВ-функции были введены в разделе 2.6, где они рассматривались в плане минимизации числа состояний. Поскольку каждому переходу в таблице приписывается некоторый номинальный временной интервал, для сохранения этого свойства кодирование строк должно быть таким, чтобы каждый переход требовал примерно одного и того же времени. Этому условию удовлетворяют ОТП-кодирования. Универсальное кодирование соседними кодовыми комбинациями, рассмотренное в подразделе 3.3.1, пригодно для любого типа таблиц переходов, так что его сразу можно применять к этой задаче. Однако теорема 3.2, устанавливающая условия, при которых ООТП-кодирование является пригодным, справедлива лишь для ОИВ-таблиц. В настоящем разделе мы расширим неформальным образом условия покрытия в этой теореме так, чтобы можно было рассматривать и МИВ-функции.

Если таблица переходов не является ОИВ-таблицей в силу наличия таких ситуаций, что для некоторых столбца I_k и строки i $Z(i, I_k) \neq Z(N(i, I_k), I_k)$ (примером такого многократного изменения выхода является переход 3—$A \longrightarrow$ 4—A в таблице 3.17, *а*), то процедура 3.1 еще, тем не менее, применима. Имеются две новые ситуации, порождающие дополнительные дихотомии, которые необходимо покрыть. Переходное множество, соответствующее переходу $i \longrightarrow j$ в том же самом столбце, в каком имеется переход $j \longrightarrow k$, не должно включать k, иначе j может быть вообще недостижимо и могут быть пропущены некоторые выходные значения. Таким образом, (ij, k) должно быть покрыто. Необходимо также покрыть (i, jk), чтобы помешать повторному достижению i в течение перехода $j \longrightarrow k$. Остается еще пересечение множеств состояний $T(i, j)$ и $T(j, k)$: состояниям этого пересечения приписываются те же выходы и те же последующие состояния,

которые приписаны состоянию j. Во всех других случаях не возникает недоразумений при заполнении клеток таблицы для промежуточных состояний переходных множеств: эти клетки должны быть такими же, что и клетки, соответствующие конечным состояниям переходов.

Таблица 3.17

	A	B	C
1	***1***,00	2,11	2,00
2	1,01	3,01	***2***,00
3	4,10	***3***,00	5,00
4	***4***,00	3,00	2,00
5	***5***,11	6,00	***5***,00
6	2,00	***6***,00	2,00

а) МИВ-таблица переходов

	A	B	C	y_1	y_2	y_3	y_4
1	***1***,00	2,11	2,00	0	0	0	0
2	1,01	3,01	***2***,00	1	0	0	1
3	4,10	***3***,00	5,00	1	0	1	0
4	***4***,00	3,00	***2***,00	1	0	1	1
5	***5***,11	6,00	***5***,00	1	1	1	0
6	2,00	***6***,00	2,00	1	1	0	1
α	1,00	2,11	2,00	0	0	0	1
β	1,00	3,01	2,00	1	0	0	0
γ	—	6,00	—	1	1	0	0
ε	—	6,00	—	1	1	1	1

б) Матрица переходов

Вторая ситуация возникает, когда в некотором столбце I имеются переходы $i \longrightarrow j$ и $m \longrightarrow j$, где $Z(i, I) \neq Z(m, I) \neq Z(j, I)$. В этом случае $T(i, j)$ не должно включать m, иначе в течение перехода $i \longrightarrow j$ может появиться неправильный выходной сигнал. Таким образом, (ij, m) должно быть покрыто.

Первая из отмеченных ситуаций имеет место в столбце таблицы 3.17, *а*, в котором существует переход $6 \longrightarrow 2 \longrightarrow 1$, требующий покрытия дихотомий (1, 26) и (12, 6). Столбец B содержит пример второй ситуации, поскольку в нем есть переходы $2 \longrightarrow 3$ и $4 \longrightarrow 3$ такие, что $Z(4, B) \neq Z(2, B) \neq Z(3, B)$. Следовательно, (2,34) должно быть покрыто. Дихотомиями, соответствующими каждому столбцу, в нашем примере являются:

A: (12,34) (12,5) (12,6) (1,26) (26,34) (26,5) (34,5),

B: (1,23) (12,34) (12,56) (23,56) (2,34) (34,56),

C: (12,35) (26,35) (24,35).

Удалим отсюда дихотомии (например, (12,6)), покрываемые другими дихотомиями, тогда применение шагов

2—7 процедуры 3.1 дает минимальную покрывающую совокупность дихотомий, содержащую (1,23), (1234,56), (126,345) и (135,246). Результирующая матрица переходов представлена в таблице 3.17, *б*, где (1,23) для наглядности заменено на (1,23456). Не показанным y-состояниям соответствуют клетки с прочерками.

3.4. Позиционное кодирование

Интересным специальным случаем однозначного кодирования является так называемое *позиционное кодирование*, характеризующееся тем, что для каждой строки таблицы переходов только одна из y-переменных имеет значение 1. Например, в таблице 3.18 переменная y_i и

Таблица 3.18

ОИВ-таблица и вариант позиционного кодирования

	x_1x_2 00	01	11	10	y_1	y_2	y_3	y_4	y_5
1	***1***	2	3	***1***	1	0	0	0	0
2	—	***2***	3	—	0	1	0	0	0
3	4	***3***	***3***	1	0	0	1	0	0
4	***4***	—	***4***	5	0	0	0	1	0
5	1	***5***	4	***5***	0	0	0	0	1

только она равна 1 для каждой строки i. Мы покажем, что если МИВ-таблица кодируется таким образом, то всегда возможно так синтезировать логическую схему, что каждый переход между состояниями будет совершаться в два такта, причем промежуточными состояниями будут такие, в которых точно две y-переменные равны 1. Следовательно, если используемые компоненты характеризуются тем, что желательно как можно дольше держать элементы памяти в нулевом состоянии, чтобы минимизировать, скажем, энергетический расход, или тепловые потери, или износ компонент, то такое кодирование представляется наилучшим. Другие его достоинства станут нагляднее при рассмотрении процедуры синтеза.

Пусть исходная ОИВ-таблица переходов задана в стандартной форме (каждое неустойчивое состояние пере-

ходит непосредственно в устойчивое с тем же значением выхода). Если в столбце I имеется переход из состояния p в состояние q, спроектируем логику таким образом, что сначала y_q становится равным 1 (y_p, очевидно, первоначально равен 1), а лишь потом y_p принимает значение 0. Общий вид функции возбуждения для y_i таков: $Y_i = F_{i1} + y_i\overline{F}_{i2}$. Функция F_{i1} есть просто сумма членов, представляющих неустойчивые полные состояния, ведущие непосредственно в строку i. Следовательно, F_{q1} содержит член Iy_p. Функция F_{i2} есть сумма членов, соответствующих следующим состояниям, достижимым из строки i. Член y_iF_{i2} служит для того, чтобы поддерживать значение y_i, равное 1, пока не станет равной 1 y-переменная, соответствующая следующему состоянию. Для перехода $p \to q$ в F_{p2} имеется член y_q.

Y-выражениями для таблицы 3.18 являются:

$$
\begin{aligned}
Y_1 &= \bar{x}_1\bar{x}_2y_5 + x_1\bar{x}_2y_3 + y_1\bar{y}_2\bar{y}_3,\\
Y_2 &= \bar{x}_1x_2y_1 + y_2\bar{y}_3,\\
Y_3 &= x_1x_2y_1 + x_1x_2y_2 + y_3\bar{y}_1\bar{y}_4,\\
Y_4 &= \bar{x}_1\bar{x}_2y_3 + x_1x_2y_5 + y_4\bar{y}_5,\\
Y_5 &= x_1\bar{x}_2y_4 + y_5\bar{y}_1\bar{y}_4.
\end{aligned}
$$

Каждое неустойчивое состояние порождает одно произведение в выражении F_{i1} и добавляет одну букву в произведение F_{i2}. Логические выражения можно составить путем анализа матрицы переходов, и легкость, с какой они находятся, является одним из достоинств позиционного кодирования. Если m — число входных переменных, r — число строк, а u — число неустойчивых состояний, то верхняя оценка числа входов логических элементов, необходимых для двухуровневой реализации функций возбуждения при позиционном кодировании ОИВ-таблицы переходов, есть $2r + u(m + 3)$, это выражение получено путем вычисления числа членов и букв в выражениях F_{i1} и F_{i2} в зависимости от числа неустойчивых состояний в том или ином столбце. (Верхняя оценка числа входов логических элементов, необходимых для реализации выходов, равна $k(m + 2)$, где k — пол-

ное число единичных значений выходов во всех клетках таблицы переходов.) Ясно, что сложность логики, связанной с внутренними переменными, возрастает линейно с ростом r. В частности, когда число неустойчивых состояний относительно невелико, сложность логики, необходимой в схемах, реализующих позиционное кодирование, обычно не превышает той сложности, которая необходима при использовании вариантов кодирования, требующих меньшего числа переменных. В любом случае полезно иметь легко вычисляемую верхнюю оценку сложности логики, необходимой для реализации данной таблицы переходов.

При синтезе МИВ-таблиц необходимо модифицировать процедуру логического синтеза; покажем это на примере рассмотрения перехода $1 \to 2 \to 3 \to 4$ в столбце A некоторой таблицы переходов. Последовательность значений переменных $y_1y_2y_3y_4$ (значения всех других y-переменных сохраняются нулевыми) должна быть следующей: $1000 \to 1100 \to 0100 \to 0110 \to 0010 \to 0011 \to \to 0001$. Этого можно достичь с помощью следующих выражений, в которых приведены лишь члены, относящиеся к рассматриваемым переменным:

$$Y_1 = \ldots + y_1\bar{y}_2 + \ldots,$$
$$Y_2 = \ldots + Ay_1 + y_2\bar{y}_3 + \ldots,$$
$$Y_3 = \ldots + A\bar{y}_1y_2 + y_3\bar{y}_4 + \ldots,$$
$$Y_4 = \ldots + A\bar{y}_2y_3 + y_4 + \ldots$$

Единственным необходимым изменением является добавление дополнений y-переменных в функции F_{i1}, чтобы предотвратить одновременное появление единичных значений более чем у двух y-переменных.

3.5. Кодирование с учетом простоты структуры

Основным вторичным критерием для оценки того или иного варианта кодирования является сложность получаемых схем. При выборе какого-либо варианта кодирования для импульсных или синхронных систем этому уделяется основное внимание. Хартманис и Стирнз [44]

разработали методы построения кодирований для синхронных схем таким образом, что многие функции возбуждения зависят только лишь от некоторых y-переменных. Такие *кодирования с ослабленной зависимостью* обычно приводят к построению схем, гораздо более простых, чем если бы кодирование производилось каким-либо бессистемным образом.

Один из подходов, тесно примыкающий к подходу, плодотворно исследованному Хартманисом и Стирнзом, состоит в декомпозиции синхронных последовательных схем на взаимосвязанные подсхемы таким образом, чтобы все обратные связи были лишь внутренними в этих подсхемах. Такие *декомпозиции без обратных связей* весьма желательны, так как они позволяют строить схемы, более удобные с точки зрения понимания их работы, диагностирования, эксплуатации и изготовления (особенно при реализации в виде интегральных схем). При этом может быть уменьшено также общее число компонент таких схем.

Далее в этом разделе мы покажем, как эти понятия распространить на ОИВ-асинхроннные схемы. Чтобы излагаемый материал сделать более понятным для читателя, в следующем подразделе дается краткое изложение результатов Хартманиса и Стирнза. Однако, как станет понятным, применение описываемых методов связано с использованием некоторых специальных процедур, которые здесь не приводятся.

Идеи, содержащиеся в этом разделе, возникли недавно и еще не привели к разработке основанных на них практических методов. Тем не менее, важность этого материала оправдывает его включение в настоящую книгу.

3.5.1. Основные понятия структурной теории машин. *Разбиением* на множестве состояний является группирование состояний в непересекающиеся подмножества, называемые *блоками*, таким образом, что каждое состояние принадлежит в точности одному блоку. Например, можно определить разбиение с четырьмя блоками на состояниях $\{1, 2, \ldots, 8\}$ следующим образом: $\alpha = (156, 2, 38, 47)$. Запись $a \underset{\beta}{=} b$ (β уравнивает a и b) означает, что состояния a и b находятся в одном и том

же блоке разбиения β. В предыдущем примере $1 \underset{\alpha}{=} 5$, $3 \underset{\alpha}{=} 8$, $5 \underset{\alpha}{\neq} 7$. Тривиальное разбиение с одним элементом в каждом блоке называется 0-разбиением, тогда как разбиение, в котором все состояния объединены в один блок, называется I-разбиением.

Если задано однозначное кодирование таблицы переходов, с каждой y-переменной y_i можно связать однозначно разбиение ρ_i с двумя блоками такое, что в одном блоке находятся все состояния, для которых $y_i = 0$, а в другом — те состояния, для которых $y_i = 1$. Пусть α и β — два разбиения (если не оговорено противное, будем всегда полагать, что рассматриваемые разбиения определены на одном и том же множестве состояний), будем говорить, что $\alpha \geqslant \beta$, если каждый блок разбиения β содержится в некотором блоке разбиения α. Это то же самое, что сказать, что из $a = b$ следует $a \underset{\alpha}{=} b$. Например, $(1356, 24, 78) \geqslant (15, 24, \underset{\beta}{36}, 7, 8)$. *Произведением* $\alpha\beta$ является разбиение, которое уравнивает a и b тогда и только тогда, когда оба разбиения α и β уравнивают a и b. Таким образом,

$$(1356, 24, 78)\,(1236, 45, 78) = (13, 2, 4, 5, 6, 78).$$

Заметим, что, если $\lambda = \alpha\beta$, $\alpha \geqslant \beta$ и $\beta \geqslant \delta$, то $\lambda \geqslant \delta$, так что $\alpha\beta$ называют *наибольшей нижней гранью* для α и β. Если $y_1, y_2, \ldots, y_m$ — y-переменные однозначного кодирования для заданной таблицы переходов, то $\prod_{i=1}^{m} \rho_i = 0$, это эквивалентно тому, что каждой строке приписывается единственное y-состояние.

Допустим, что α и β — разбиения на состояниях исходной таблицы переходов. Тогда (α, β) образуют *пару разбиений*, записываемую как $P(\alpha, \beta)$, если для каждого входа I и каждой пары состояний p и q таких, что $p \underset{\alpha}{=} q$, справедливо $N(p, I) \underset{\beta}{=} N(q, I)$. В случае таблицы 3.19 разбиения $\alpha = (14, 235)$ и $\beta = (15, 234)$ образуют пару разбиений $P(\alpha, \beta)$. Заметим, что состояния 1 и 4 принадлежат одному и тому же блоку разбиения α, а множествами последующих состояний для этой пары в столбцах A, B и C являются соответственно множества

24, 34 и 15, состояния каждого из которых содержатся в определенном, одном и том же блоке разбиения β.

Пусть разбиения λ и φ образуют пару разбиений $P(\lambda, \varphi)$ в некоторой таблице переходов, S_1 и S_2 — подмножества индексов y-переменных, а кодирование строк произведено таким образом, что $\lambda = \prod_{i \in S_1} \rho_i$ и $\varphi = \prod_{j \in S_2} \rho_j$. Тогда можно показать, что для каждого $j \in S_2$ Y_j является функцией входных переменных и y_i таких, что $i \in S_1$. (Мы предполагаем, что таблица реализована на синхронизированных элементах задержки.) Если S_1 соответствует *собственному подмножеству* всего множества y-переменных, то функции возбуждения для y-переменных, соответствующих множеству S_2, зависят не от всех, а лишь от некоторых y-переменных. Это как раз то, что называется ослабленной зависимостью. Читатель может убедиться, что, если выписанные в таблице 3.19 столбцы y_1 и y_2 являются фрагментом некоторого кодирования строк, то Y_2 есть функция от состояния входа и y_1. Нетрудно показать, что такую функцию можно представить в виде

Таблица 3.19

Таблица переходов для синхронной функции

	A	B	C	y_1	y_2
1	4	3	5	0	0
2	5	3	4	1	1
3	5	2	3	1	1
4	2	4	1	0	1
5	1	4	2	1	0

$$A\overline{y_1} + B + Cy_1,$$

не зная вида остальных y-столбцов в этом кодировании.

Случай, когда пара (φ, φ) является парой разбиений $P(\varphi, \varphi)$, чрезвычайно важный и заслуживает специального рассмотрения. Будем говорить, что такое φ обладает *свойством подстановки*, и записывать это свойство в виде $S(\varphi)$. Так, в таблице 3.19 разбиение $(12, 3, 45)$ есть разбиение со свойством подстановки.

Вместо оперирования с разбиениями можно рассматривать так называемые *системы множеств*, которые позволяют нам охватить случай многозначных кодирований. Система множеств на данном множестве состояний отличается от разбиения на этом множестве лишь тем, что ее блоки могут перекрываться. Кроме требования, чтобы каждое состояние входило по крайней мере в один блок,

единственным ограничением является требование, чтобы никакой блок не был подмножеством какого-либо другого блока. Определения пары разбиений, разбиения со свойством подстановки и соображения по ослабленной зависимости непосредственным образом распространяются и на системы множеств. Если блоки разбиения ρ_i перекрываются, это значит, что состояниям, входящим в пересечение, приписаны оба значения 0 и 1 переменной y_i. Например, в таблице 3.20 $\rho_1 = (123, 345)$ соответствует столбцу y_1. Заметим, что ρ_1 обладает свойством подстановки, так что мы можем найти Y_1 как функцию входа и y_1, независимо от того, как определены другие y-столбцы. Действительно, по таблице 3.20 получаем

$$Y_1 = A + By_1.$$

Как станет очевидным в следующих подразделах, свойство ослабленной зависимости в асинхронном случае несколько менее наглядно в силу всяких усложнений, вытекающих из возможности появления критических состязаний.

Таблица 3.20

Таблица переходов, содержащая систему множеств со свойством подстановки

	A	B	C	y_1
1	4	2	3	0
2	4	1	1	0
3	5	3	2	0 или 1
4	3	5	2	1
5	4	5	2	1

3.5.2. Свойство ослабленной зависимости для асинхронных схем. В синхронном случае простой факт, что $P(\rho_1\rho_2, \rho_3)$ есть пара разбиений, достаточен для нашей уверенности в том, что при любом кодировании строк можно определить Y_3 без знания внутренних переменных, отличных от y_1 и y_2. Иллюстрацией трудностей, возникающих при расширении этого подхода на асинхронные схемы, является таблица переходов, представленная таблицей 3.21.

Рассмотрим кодирующие переменные y_1 и y_2 (первые два y-столбца справа от таблицы). Найдем вначале $\rho_1\rho_2 = (125, 34)$ $(1245, 3) = (125, 3, 4)$ и убедимся, что $P(\rho_1\rho_2, \rho_3)$ есть пара разбиений, так что непосредственное применение теории, разработанной для синхронных схем, привело бы нас к выводу, что Y_1 можно выразить как функцию от x_1, x_2, y_1 и y_2. Действительно, мы можем

получить такое выражение в следующей форме:

$$Y_1 = \bar{x}_1\bar{x}_2y_1 + x_2y_1\bar{y}_2 + \bar{x}_1x_2.$$

Однако анализ перехода $3 \to 1$ в столбце 11 показывает, что при этом изменяются оба y_1 и y_2, и если y_2 меняется первым, система переходит в состояние 4—11, при этом

Таблица 3.21

ОИВ-таблица, для которой может быть построено кодирование с ослабленной зависимостью

	x_1x_2 00	01	11	10	y_1	y_2	y_3	Кодирование с ослабленной зависимостью y_1	y_2	y_3	y_4
1	***1***,0	4,0	***1***,0	5,0	0	0	0	0	0	0	0
2	1,0	4,0	***2***,1	5,0	0	0	0	0	0	0	1
3	***3***,1	4,0	1,0	5,0	1	1	0	0	1	0	0
4	3,1	***4***,0	***4***,0	5,0	1	0	1	1	1	0	0
5	***5***,0	4,0	2,1	***5***,0	0	0	0	0	0	1	1

Y_1 имеет неправильное значение. Другими словами, между y_1 и y_2 имеется состязание, которое является критическим по отношению к Y_1. Таким образом, значений y_1 и y_2 недостаточно для правильного задания значений Y_1 при каждом переходе.

Рассмотрим теперь переменную y_3, показанную в третьем y-столбце таблицы 3.21. Поскольку $\rho_1\rho_3 = \rho_1\rho_2$, пара $P(\rho_1\rho_3, \rho_1)$ также является парой разбиений, причем той же, что и ранее. Однако теперь справедливо равенство

$$Y_1 = \bar{x}_1\bar{x}_2y_1 + x_2y_3 + \bar{x}_1x_2.$$

Здесь нет уже критических состязаний, как в предыдущем случае. Существенным является то, что y_1 и y_3 покрывают все дихотомии, порождаемые парами переходов, в которых y_1 имеет различные значения в конечных состояниях этих переходов. Докажем теперь теорему, которая завершит проделанное рассмотрение.

Теорема 3.6. *Пусть задано пригодное ООТП-кодирование для ОИВ-таблицы переходов, где* σ *— подмножество внутренних переменных,* y_0 *— некоторая внутренняя пе-*

ременная, $\mathbf{x}$ *— входной вектор.* $Y_0 = f_0(\mathbf{x}, \sigma)$ *тогда и только тогда, когда:*

а) пара $P\left(\prod_{y_i \in \sigma} \rho_i, \rho_0\right)$ *является парой разбиений,*

б) для каждой пары переходов $a \to b$ *и* $c \to d$ *в одном и том же столбце таких, что* $b \underset{\rho_0}{=} d$, *дихотомия* (ab, cd) *покрывается некоторым* $y_i \in \sigma$.

Доказательство. Опишем лишь схему доказательства.

Необходимость условия а) доказывается так же, как и для синхронных схем. Если для некоторой пары переходов $a \to b$ и $c \to d$ не выполняется условие б), можно показать, как и при доказательстве теоремы 3.2 (определяющей условия пригодности ООТП-кодирования), что в булевом подпространстве, определенном элементами множества σ, $T(a, b)$ пересекается с $T(c, d)$. Но тогда существуют такие комбинации значений σ-переменных, для которых в рассматриваемом столбце требуется два различных значения для Y_0.

Далее предположим, что условие а) выполняется. Как и в синхронном случае, можно показать, что определенное значение функции Y_0 для каждого y-состояния, приписанного строке таблицы переходов, зависит лишь от входных переменных и от состояний подмножества σ. Если, кроме того, выполняется условие б), то тогда в соответствии со схемой доказательства теоремы 3.2 можно показать, что при переходах в состояния одного и того же столбца с различными значениями переменной y_0 нам не встретится никакое промежуточное y-состояние (заметим, что рассматриваются лишь σ-переменные). Это означает, что для всех полных состояний Y_0 может быть получено соответствующим образом по значениям только входных переменных. Недавно было показано, что условие а) в этом доказательстве можно опустить, поскольку из б) следует а). Доказательство этого, не очень сложное, предоставляется читателю.

Другая теорема, близкая к только что доказанной, утверждает, что для каждой пары разбиений можно найти некоторое соотношение с ослабленной зависимостью.

Теорема 3.7. *Если для некоторой ОИВ-таблицы* T *пара* $P(\alpha, \rho_k)$ *есть пара разбиений, то существует*

множество y-переменных σ_k такое, что $\prod_{y_i \in \sigma_k} \rho_i = \alpha$, и пригодное ООТП-кодирование для T такое, что $Y_k = f_k(x, \sigma_k)$.

Доказательство. Намеченное лишь в общих чертах доказательство предусматривает, что для каждой пары переходов $a \to b$ и $c \to d$ в одном и том же столбце таких, что $b \underset{\rho_k}{\neq} d$, мы можем найти разбиение ρ с двумя блоками, причем $\rho \geqslant \alpha$ и ρ покрывает (ab, cd). После того как это сделано, перемножая такие разбиения ρ, получим разбиение $\alpha' \geqslant \alpha$, причем если желательно «приблизить» α' к α, следует включить в рассмотрение бóльшее число разбиений типа ρ. Тогда множество y-переменных, соответствующих результирующему множеству разбиений ρ, удовлетворяет теореме 3.6, а также требованиям на σ_k в условии настоящей теоремы.

Желаемое ρ можно найти путем объединения блоков разбиения α, содержащих a и b, а также путем объединения блоков, содержащих c и d. Оставшиеся блоки можно объединить с каким-либо из только что образованных блоков, и получить, таким образом, разбиение с двумя блоками, покрывающее (ab, cd) и доминирующее над α*). Это, однако, можно сделать лишь тогда, когда ни a, ни b не находятся в одном блоке из α с c или d. Докажем, что это условие действительно выполняется, показав, что $a \underset{\alpha}{\neq} d$ (аналогично показывается для $a \underset{\alpha}{\neq} c$, $b \underset{\alpha}{\neq} c$, $b \underset{\alpha}{\neq} d$). Пусть I — входной столбец, в котором имеют место переходы $a \to b$ и $c \to d$. Тогда, поскольку $P(\alpha, \rho_k)$ — пара разбиений, из определения пары разбиений следует, что $a \underset{\alpha}{=} d$ влечет $N(a, I) \underset{\rho_k}{=} N(d, I)$, что эквивалентно $b \underset{\rho_k}{=} d$. Но по предположению, $b \underset{\rho_k}{\neq} d$, так что $a \underset{\alpha}{\neq} d$.

В главе 5 показывается (теорема 5.1), что для каждой неизбыточной внутренней переменной y_i, используемой в каком-либо кодировании строк асинхронной ОИВ-функции, Y_i является функцией только от y_i. Применяя

*) Т. е. большее, чем α по отношению $\geqslant$. (*Прим. перев.*)

только что доказанную теорему, можно, таким образом, надеяться найти y_k в σ_k.

Для иллюстрации применения теорем 3.6 и 3.7 построим эффективное ООТП-кодирование для таблицы 3.21. Используя стандартный подход, разработанный для синхронных функций, можно найти все пары разбиений и разбиения со свойством подстановки (здесь следует отослать читателя к работе [44]). Одним из подходящих разбиений со свойством подстановки является $S(1235, 4)$. Единственными парами переходов, оканчивающихся в состояниях, входящих в разные блоки разбиения $(1235, 4)$, являются переходы $3 \to 1$ и $4 \to 4$ в столбце 11, а также переходы $5 \to 2$ и $4 \to 4$ в том же столбце. Поскольку $(1235, 4)$ покрывает соответствующие дихотомии $(13,4)$ и $(25,4)$, из теоремы 2.6 следует, что если положить $\rho_1 = (1235, 4)$, то Y_1 представимо в виде $f_1(x_1, x_2, y_1)$ в пригодном ООТП-кодировании. Можно также показать, что $P[(125, 3, 4), (125,34)]$ есть пара разбиений, что определяет выбор $\rho_2 = (125, 34)$. Тогда, поскольку $P[\rho_1\rho_2, \rho_2]$ — пара разбиений и дихотомии $(12, 34)$, $(34, 5)$, $(13, 4)$ и $(25, 4)$ покрываются или разбиением ρ_1, или ρ_2, теорема 3.6 гарантирует нам, что можно записать $Y_2 = f_2(x_1, x_2, y_1, y_2)$. Аналогичные рассуждения, начинающиеся с того, что $S(1234, 5)$ есть разбиение со свойством подстановки, а $P[(134, 2, 5), (134, 25)]$ — пара разбиений, приводят к выбору $\rho_3 = (1234, 5)$ и $\rho_4 = (134, 25)$, так что $S(\rho_3)$ есть разбиение со свойством подстановки, а $P(\rho_3\rho_4, \rho_4)$ — пара разбиений. Вновь получающиеся дихотомии покрываются соответствующим образом, так что Y_3 зависит только от y_3, Y_4 — только от y_3 и y_4. Поскольку $\rho_1\rho_2\rho_3\rho_4 = 0$, мы получаем пригодное кодирование, которое показано в четырех правых y-столбцах таблицы 3.21. Выражениями для Y-функций являются:

$$Y_1 = \bar{x}_1 x_2 + x_2 y_1,$$

$$Y_2 = \bar{x}_1 x_2 + x_2 y_1 + \bar{x}_1 y_2,$$

$$Y_3 = x_1 \bar{x}_2 + \bar{x}_2 y_3,$$

$$Y_4 = x_1 \bar{x}_2 + \bar{x}_2 y_3 + x_1 y_4.$$

В этой задаче существуют и другие варианты кодирований с ослабленной зависимостью, сравнимые по сложности логики. Один из таких вариантов требует трех внутренних переменных.

3.5.3. Декомпозиция асинхронных схем. Рассматриваемая здесь проблема заключается в реализации асинхронной последовательностной функции в виде двух или более отдельных последовательностных схем, связанных таким образом, что никакая обратная связь не охватывает элементы более чем одной схемы. Ниже рассматриваются основные типы соединений таких схем. На

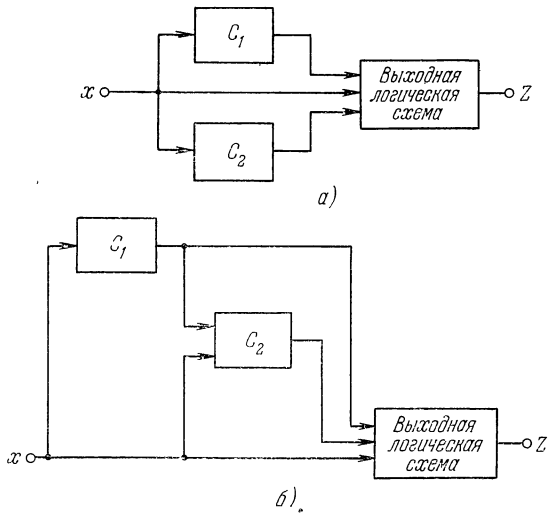


Рис. 3.2. *а*) Параллельная декомпозиция. *б*) Последовательная декомпозиция.

рис. 3.2, *а* показана *параллельная декомпозиция*, где каждая из взаимно независимых последовательностных схем C_1 и C_2 принимает внешний входной сигнал x и передает выходной комбинационной схеме информацию о своем внутреннем состоянии. В *последовательной декомпозиции*, показанной на рис. 3.2, *б*, ситуация отличается лишь тем, что C_2 получает также информацию и от C_1.

Наше рассмотрение вначале будет касаться лишь ситуаций, когда для каждого состояния исходной таблицы переходов существует единственная пара состояний схем C_1 и C_2. Это позволит нам рассматривать лишь пары разбиений и избежать ряда нерешенных задач, связанных с системами множеств. Пример, в котором используются системы множеств, будет приведен позднее.

Мы начнем с простейшего случая — параллельной декомпозиции, когда основная теорема, справедливая для синхронных функций, справедлива также для асинхронных функций.

Теорема 3.8. *ОИВ-таблицу переходов можно реализовать в виде параллельной декомпозиции из двух схем таким образом, что каждому состоянию таблицы переходов будет соответствовать единственная пара состояний этих схем тогда и только тогда, когда существует пара разбиений со свойством подстановки α_1 и α_2 таких, что $\alpha_1\alpha_2=0$*

Доказательство. Доказательство основывается на том соображении, что если рассматривать схемы-компоненты C_1 и C_2 как одну последовательностную схему, то объединенные множества y-переменных можно интерпретировать как множество кодирующих переменных исходной таблицы, разделенное на два взаимно независимых подмножества. Это деление соответствует паре разбиений со свойством подстановки, и если упорядоченная пара состояний из C_1 и C_2 взаимно-однозначно соответствует состоянию исходной таблицы, то произведение этих разбиений должно быть равно 0. Это рассуждение полностью обратимо, откуда следует справедливость обратного утверждения теоремы. Конечно, необходимо, чтобы C_1 и C_2 были реализованы в каком-либо виде, свободном от критических состязаний, но это всегда можно сделать, Если C_1 и C_2 должны быть ООТП-схемами, то эквивалентное кодирование исходной таблицы переходов также должно быть ООТП-кодированием, и выполнение условий теоремы 3.6 эквивалентно выполнению условий пригодности ООТП-кодирований для схем C_1 и C_2.

В качестве примера рассмотрим таблицу 3.22, *а*; заметим, что, если $\alpha_1 = (13, 245)$ и $\alpha_2 = (1, 2, 34, 5)$, то α_1 и α_2-разбиения со свойством подстановки, причем $\alpha_1\alpha_2 = 0$, так что выполняются условия теоремы 3.8.

Таким образом, можно построить таблицы переходов (табл. 3.22, *б* и *в*), соответствующие схемам C_1 и C_2 (рис. 3.2, *а*). Строки этих таблиц соответствуют блокам разбиений α_1 и α_2, что отмечено справа на таблицах. Теперь путем построения пригодных кодирований независимо для каждой таблицы можно синтезировать требуемые схемы (на таблицах показаны ООТП-кодирования).

Таблица 3.22

	x_1x_2 00	01	11	10
1	***1***,0	***1***,0	4,0	2,0
2	***2***,0	3,0	***2***,0	2,0
3	***3***,1	***3***,0	4,0	—
4	***4***,0	—	***4***,0	5,0
5	***5***,1	1,0	—	***5***,0

а) Таблица переходов, имеющая разбиения со свойством подстановки

	x_1x_2 00	01	11	10	y_1	
1	***1***	***1***	2	2	0	(13)
2	***2***	1	***2***	***2***	1	(245)

б) таблица, соответствующая разбиению α_1

	x_1x_2 00	01	11	10	y_1 y_2 y_3
1	***1***	***1***	3	2	0 0 0 (1)
2	***2***	3	***2***	***2***	1 1 0 (2)
3	***3***	***3***	***3***	4	1 0 1 (34)
4	***4***	1	—	***4***	0 1 1 (5)

в) Таблица, соответствующая разбиению α_2

Выходная логическая схема определяет полное внутреннее состояние как функцию состояний схем C_1 и C_2. Так, если C_1 в состоянии 2, а C_2 — в состоянии 3, то состояние исходной таблицы совпадает с пересечением соответствующих блоков 245 и 34, равным 4.

Вернемся к проблеме последовательной декомпозиции и допустим, что α_1 — некоторое разбиение со свойством подстановки на множестве состояний исходной таблицы T. Тогда по α_1 может быть построена новая таб-

лица T_1 (состояния которой соответствуют блокам разбиения α_1). Допустим, что для T_1 найдено ООТП-кодирование с множеством y-переменных σ_1 и что это кодирование используется как частичное кодирование для T в виде приписывания каждому состоянию из T кодовой комбинации, используемой для строки из T_1, соответствующей тому блоку из α_1, который содержит рассматриваемое состояние таблицы T. Тогда с помощью теоремы 3.6 можно показать, что Y-функции для y-переменных из σ_1 зависят только от входа x и этих y-переменных. Допустим, далее, что ООТП-кодирование таблицы T завершается таким выбором второго множества y-переменных σ_2, чтобы покрыть все дихотомии согласно теореме 3.1, которые не были покрыты исходным множеством. Мы можем теперь рассмотреть схему C_1, описываемую таблицей T_1, с множеством σ_1 y-переменных, используемых для ее кодирования, входом этой схемы является вектор x. Вторая схема C_2 определяется множеством σ_2 y-переменных, ее входами являются вектор x и y-переменные из C_1. Таким образом, мы получили последовательную декомпозицию вида, приведенного на рис. 3.2, *б*. Заметим, что если y-переменные из σ_1 не покрывают никакую из требуемых дихотомий, декомпозиция становится тривиальной в том плане, что схема C_2 сама реализует T.

Пусть, например, таблицей T будет таблица 3.23, *а*, имеющая разбиение со свойством подстановки $\alpha_1 = (123, 45, 67, 89)$. Это разбиение соответствует таблице 3.23, *б*, которая имеет пригодное ООТП-кодирование, использующее переменные y_1 и y_2. Дихотомиями, которые должны быть покрыты при пригодном ООТП-кодировании для T, являются: (1,28), (1,39), (1,47), (1,56), (28,39), (28,47), (28,56), (39,47), (39,56), (47,56), (15,24), (15,68), (15,79), (24,35), (24,68), (24,79), (35,68), (35,79) и (68,79). Во всяком кодировании таблицы T_1 блокам разбиения α_1 должны соответствовать разные значения y-переменных, так что, если дана какая-либо дихотомия, образованная из двух блоков разбиения α_1, то имеется некоторая y-переменная в кодировании, покрывающая такую дихотомию. В зависимости от конкретного варианта используемого кодирования могут быть покрыты также и некоторые другие дихотомии.

Таблица 3.23

	x		Кодирование строк таблицы			
	0	1	y_1	y_2	y_3	y_4
1	***1***,0	5,0	0	0	0	0
2	***2***,0	4,0	0	0	1	1
3	***3***,1	5,0	0	0	1	0
4	7,0	***4***,0	0	1	0	1
5	6,0	***5***,0	0	1	1	0
6	***6***,0	8,0	1	1	1	1
7	***7***,0	9,0	1	1	0	0
8	2,0	***8***,0	1	0	1	1
9	3,0	***9***,0	1	0	1	0

а) Таблица переходов T, имеющая разбиение α_1=(123, 45, 67, 89) со свойством подстановки

	x				
	0	1	y_1	y_2	
1	***1***	2	0	0	(123)
2	3	***2***	0	1	(45)
3	***3***	4	1	1	(67)
4	1	***4***	1	0	(89)

б) Матрица переходов для схемы C_1

	xy_1y_2										
	000	001	011	010	110	111	101	100	y_3	y_4	
1	***1***	***1***	***1***	—	3	3	3	3	0	0	(17)
2	***2***	***2***	***2***	***2***	***2***	***2***	4	4	1	1	(268)
3	***3***	2	2	***3***	***3***	***3***	***3***	***3***	1	0	(359)
4	—	1	1	—	—	—	***4***	***4***	0	1	(4)

в) Матрица переходов для схемы C_2

Однако среди выписанных выше дихотомий дихотомия (1, 28) не покрывается ни при каком кодировании, связанном с α_1, поскольку $1 \underset{\alpha_1}{=} 2$. Для покрытия (1, 47) (как и других дихотомий) используется специальное кодирование, хотя можно было бы различить блоки разбиения α_1 и без этого (если бы, например, кодовые комбинации, приписанные парам 67 и 89, были взаимозаменяемы, что в настоящем случае привело бы к появлению критических состязаний в C_1). Никакие из дихотомий в настоящем примере не покрываются парами блоков разбиения α_1 и, следовательно, не покрываются ни при каких вариантах кодирования для C_1. Далее, каждая из перечисленных дихотомий, кроме (1, 28), (1, 39), (28, 39), (47, 56), (15, 24), (24, 35) и (68, 79), покрывается y-пе-

ременными y_1 и y_2. Применяя к этим дихотомиям этапы с третьего по шестой процедуры 3.1, получим минимальную пару покрывающих дихотомий (147, 235689) и (13579, 2468), соответствующих y_3 и y_4 в таблице 3.23, *а*.

При желании по матрице переходов для T можно найти точную таблицу переходов для C_2 путем построения таблицы, строки которой соответствуют блокам $\rho_1\rho_2$, а входами являются x, y_1 и y_2. Клетки такой таблицы заполняются путем рассмотрения необходимых переходов в T с учетом того соображения, что клетки в таблице для C_2 должны (и могут) быть заполнены таким образом, чтобы изменения состояний в таблице для C_2 происходили до или после изменений состояний в таблице для C_1. Например, если T находится в 2 — 0, T_1 — в 1 — 0, то T_2 должна быть в 2—000. Если теперь x меняется, то y_2 становится равным 1 в соответствии с изменением 1—0 → 2—1 в T_1. Таким образом, вход в T_2 сначала изменяется на 100, а лишь потом на 101. Все следующие состояния для 2—100, 2—101, 4—100 и 4—101 должны быть равны 4. Результатом проведенного рассмотрения является следующая теорема.

Теорема 3.9. *ОИВ-таблица переходов T может быть реализована в виде последовательной декомпозиции из двух схем таким образом, что каждому состоянию из T однозначно соответствует пара состояний этих схем тогда и только тогда, когда существует разбиение со свойством подстановки.*

Мы показали, что декомпозиция может быть тривиальной, если никакое из кодирований, соответствующих рассматриваемому разбиению, не покрывает столько дихотомий из T, сколько требуется для уменьшения числа y-переменных, требующихся для покрытия остальных дихотомий.

Проиллюстрируем на примере еще несколько важных моментов. Для таблицы 3.24, *а* не существует разбиения со свойством подстановки, однако система множеств $\beta = (123, 345)$ обладает свойством подстановки. Произведем последовательную декомпозицию с использованием β. Построим сначала по β таблицу переходов B_1 (таблицу 3.24, *б*) и закодируем ее указанным на таблице тривиальным образом. С этого момента можно было бы поступать, как и ранее, заметив, например, что (123, 345)

Таблица 3.24

	x_1x_2 00	01	11	10
1	***1***,0	3,0	***1***,0	3,0
2	***2***,0	4,0	***2***,0	***2***,0
3	***3***,1	***3***,0	2,0	***3***,0
4	2,0	***4***,0	2,0	5,0
5	2,0	***5***,0	1,0	***5***,0

а) Таблица переходов B, имеющая систему множеств со свойством подстановки

	x_1x_2 00	01	11	10	y_1	
1	***1***	2	***1***	***1***	0	(123)
2	1	***2***	1	***2***	1	(345)

б) Таблица переходов B_1, соответствующая системе множеств (123, 345)

	$x_1x_2Y_1$ 000	010	110	100	101	111	011	001
1	***1***	—	***1***	3	—	—	3	—
2	***2***	—	***2***	***2***	—	—	4	—
3	***3***	—	2	***3***	***3***	—	***3***	—
4	2	—	2	—	5	—	***4***	—
5	2	—	1	—	***5***	—	***5***	—

в) Таблица переходов B_2'

	$x_1x_2Y_1$ 000	010	110	100	101	111	011	001	y_3	y_4
1	***1***	—	***1***	3	—	—	3	—	0	0
2	***2***	—	***2***	***2***	4	—	***2***	—	1	1
3	***3***	—	2	***3***	***3***	—	***3***	—	0	1
4	2	—	1	—	***4***	—	***4***	—	1	0

г) Таблица переходов B_2

покрывает $(2, 45)$ — одну из необходимых дихотомий таблицы B. Однако вместо этого мы проиллюстрируем другой подход, который является довольно общим и который можно было бы применить и в предыдущем примере. Построим таблицу B_2' (таблицу 3.24, *в*) с тем же числом состояний, что и в B, одинаковую с B по пове-

дению и отличающуюся от B лишь тем, что ее входами являются x_1, x_2 и Y_1. Для каждого устойчивого полного состояния из B в B_2' имеется по крайней мере одно устойчивое полное состояние, а может быть и несколько. Например, 3—00 устойчиво в B, и соответствующим ему состоянием в B_2' является 3—000. Оба состояния 3—100 и 3—101 из B_2' соответствуют одному и тому же состоянию 3—10 из B в силу того, что этому состоянию соответствуют два устойчивых состояния в B_1. По B и B_1 найдем совокупность последующих состояний для B_2' с учетом того факта, что вход таблицы B_2' включает не текущее, а следующее состояние из B_1.

В B_2' имеется много незаполненных клеток, и следующий шаг состоит в минимизации B_2' в предположении, что состояния внутри каждого блока из β априори являются взаимно несовместимыми. Такое предположение необходимо для того, чтобы в результирующей таблице B_2 различались те состояния, которые неразличимы в B_1. В нашем примере состояния 2 и 4 образуют единственную совместимую пару; минимизированная таблица B_2 представлена таблицей 3.24, *г*. Рядом с этой таблицей указан пригодный вариант ООТП-кодирования, использующий переменные y_3 и y_4. Последовательная композиция схем, реализующих B_1 и B_2, реализует B (с добавлением, естественно, логики на выходе). Заметим, что схема для B_1 может передавать схеме для B_2 либо Y_1, либо y_1, поскольку переменной y_1 с учетом входов x_1 и x_2 достаточно для вычисления Y_1. Схема для B_2 должна быть способна к восприятию почти непрерывных входных изменений, поскольку и x, и сигналы от схемы для B_1 могут меняться довольно быстро (эта проблема рассматривается в главе 4). Иногда возможно объединение в B_2' не только строк, но и столбцов (см. [36]), подобное объединение может привести к уменьшению в B_2 числа необходимых входов.

В общем случае, когда от разбиений переходят к системам множеств, теоремы декомпозиции применимы и при многозначных кодированиях, однако требования, подобные условию $\alpha_1\alpha_2 = 0$ теоремы 3.8, перестают быть необходимыми. Об этом говорится в задаче 3.19, однако

в настоящее время *) проблема еще далека от полного решения.

Можно также модифицировать процедуры декомпозиции, рассматривая, например, в последовательной декомпозиции подачу на схему для B_2 не y или Y, а какой-либо другой информации. В роли подаваемой информации может выступать информация о полном состоянии схемы для B_1, тогда входы x в схеме для B_2 становятся ненужными.

Подводя итог, следует сказать, что так же, как и в синхронном случае, для асинхронных схем возможно проведение декомпозиции. Условия существования декомпозиции того или иного вида определяются свойствами структуры, которая связывает между собой все разбиения со свойством подстановки или системы множеств со свойством подстановки исходной таблицы переходов.

3.5.4. Оценка сложности логики, необходимой при ООТП-кодировании. Представленная здесь процедура синтеза пригодна для ОИВ-функций. Довольно легко реализуемая, она во многих чертах подобна процедуре синтеза схем при позиционном кодировании, рассмотренном в разделе 3.4. При этом используется специальный случай кодирования по Лю (подраздел 3.3.2), в котором каждому отдельному множеству предшественников приписывается единственная y-переменная. Сложность результирующих схем вычисляется довольно легко и служит полезной оценкой, если даже реальные схемы и не являются оптимальными в том или ином смысле. Более того, используемый подход позволил разработать методы синтеза схем, действительно эффективных с точки зрения разумного использования в них логических элементов. Исследования в этом направлении бурно развиваются в настоящее время.

Будем считать, что исходная ОИВ-функция описывается с помощью минимизированной таблицы переходов в стандартной форме (раздел 2.6). Если D_{iq} — множество предшественников, то переменной y_{iq} приписывается значение 1 для каждой строки в D_{iq} и 0 — для всех остальных строк таблицы. Например, в таблице 3.24, *а* мно-

*) Имеется в виду — к моменту написания настоящей книги. (*Прим. перев.*)

жество $\{2, 4, 5\}$ в столбце 00 связано с переменной y_{002}, значение которой равно 1 в строках 2, 4 и 5, и 0 — в строках 1 и 3. Заметим, что множества D_{103} и D_{013}, будучи одинаковыми, порождают только одну y-переменную. Ясно, что результирующее кодирование является пригодным кодированием по Лю, поскольку оно разделяет каждую пару множеств предшественников в каждом столбце.

Легко также найти функцию возбуждения Y_{iq}. Для каждого входного столбца I_j значение Y_{iq} должно быть равно 1 в каждой строке k такой, что $N(k, I_j) \in D_{iq}$. Но каждая строка столбца I_j принадлежит единственному множеству предшественников этого столбца, и если $Y_{iq} = 1$ для состояния $p - I_j$, то Y_{iq} также должно быть равно 1 для каждого состояния $r - I_j$ такого, что r принадлежит тому же множеству предшественников в столбце I_j, что и p. Каждое такое множество характеризует единственная y-переменная, следовательно, каждое логическое произведение в выражении для Y_{iq} состоит из символов тех переменных y и x, которые задают рассматриваемый входной столбец. В нашем примере имеем:

$$Y_{002} = \bar{x}_1\bar{x}_2y_{002} + \bar{x}_1\bar{x}_2y_{104} + \bar{x}_1x_2y_{015} + x_1x_2y_{112} + x_1\bar{x}_2y_{102} + x_1\bar{x}_2y_{105}.$$

Пусть s — число устойчивых состояний в таблице переходов (в нашем случае $s = 11$). Тогда число различных произведений, которые должны быть рассмотрены, равно s, и если m — число входных переменных, то в схеме каждый из элементов И будет иметь $m + 1$ вход (в предположении, что мы не пытаемся уменьшить это число учетом соотношений между соседними столбцами). Следовательно, первый ярус нашей двухуровневой И — ИЛИ-схемы будет содержать $s(m + 1)$ входов элементов.

Пусть $p - I_i$ — устойчивое состояние, а I_j — некоторый входной столбец. Тогда, если все последующие состояния в I_j определены, в I_j имеется единственное множество предшественников D_{jq} такое, что $p \in D_{jq}$. Пусть I_iy_{ip} означает произведение, соответствующее I_i и y_{ip}. Тогда I_iy_{ip} является членом выражения для Y_{jq}. Если таблица переходов полностью определена, если никакие

два столбца не имеют общего множества предшественников и если c — число столбцов, то $I_i y_{ip}$ входит в c выражений для Y, а общее число входов элементов ИЛИ в схеме равно sc. Таким образом, сложность схемы, измеряемая полным числом входов элементов, равна $s(m+1)+sc = s(m+c+1)$.

Для каждой группы из k одинаковых множеств предшественников в таблице эта оценка уменьшается на $k-1$, умноженное на полное число устойчивых состояний в строках, соответствующих этим множествам. Если таблица неполностью определена, то для каждого неопределенного последующего состояния эта оценка уменьшается на 1 для каждого отдельного множества предшественников, устойчивое состояние которого соответствует строке, содержащей это неопределенное состояние.

Для таблицы 3.24, *а* формула дает $11(2+4+1) = 77$, это число должно быть уменьшено на 5, поскольку множество $\{1, 3\}$ появляется дважды, так что окончательная оценка равна 72. Рассмотренная оценка возрастает линейно с ростом числа устойчивых состояний, тогда как оценка из раздела 3.4, основанная на двухтактном позиционном кодировании, растет линейно с ростом числа неустойчивых состояний.

ЗАМЕЧАНИЯ ПО БИБЛИОГРАФИИ

Основные факты, связанные с критическими состязаниями и кодированиями, свободными от критических состязаний, были получены Хаффманом [49]; понятие сцепленных строчных множеств, так же как и кодирование, описанное в разделе 3.1, было введено Хаффманом и Венгертом [50]. Хаффман также предложил подход с совместным использованием строк [49]. Метод построения кодирований с совместным использованием строк и отличный от метода, описанного в разделе 3.2, был предложен Газельтином [45]. Сосье [98] показала, что оценку $2S_0 - 1$ можно понизить по крайней мере для ОИВ-функций, и она же построила обобщенные варианты кодирований, описанные в разделе 3.2. Четкое доказательство пригодности кодирования по Сосье с использованием четырех переменных дано Таном [103]. Кодиро-

вание соседними кодовыми комбинациями, основанное на кодах Хэмминга, также предложено Хаффманом [50]. Лю [63] ввел ООТП-кодирование, называемое здесь кодированием по Лю. Позднее Трэйси [109] разработал общую теорию минимальных ООТП-кодирований. Процедура 3.1 является уточнением подхода Трэйси, причем теорема 3.3 и связанная с ней часть процедуры представлены Ангером [115]. Некоторые подходы к уменьшению объема вычислений, необходимых при выполнении процедуры Трэйси, рассмотрены Фридзом [18]. Джилберт [26] провел интересное исследование свойств кодов Грея. Линейная оценка из подраздела 3.3.3, касающаяся числа переменных, необходимых в универсальном ОТП-кодировании, была получена Фридманом и Грэхемом [22], так же как и конструктивные процедуры, основанные на разделяющих системах и универсальных многозначных ОТП-кодированиях. Уллльман [22] улучшил конструкцию Фридмана — Грэхема, получив оценки $S_0^{1,59}$ и S_0^2. Описанный в подразделе 3.3.4 подход для построения многозначных ОТП-кодирований приведен в работе Фридза [18], в ней же содержится материал по ОТП-кодированиям для МИВ-функций. Тан [104] впервые распространил теорию Хартманиса — Стирнза на асинхронные схемы; на его работе основан весь подраздел 3.5.2. Материал раздела 3.5.3 по декомпозиции взят из работы Фридмана, Менона и Тана [107]. Здесь следует отметить также работы Кинни [56, 57]. Материал подраздела 3.5.4 связан с работой Тана [105], где описан ряд методов минимизации сложности схем. Некоторые новые подходы к синтезу схем, отличные от изложенных здесь, описаны в работах Феррари и Грасселли [17], а также Бжозовски и Сингха [10].

ЗАДАЧИ

3.1. Исходя из таблицы 3.1, *б*, найти три пути завершения кодирования состояний таблицы 3.1, *а*, при котором связанные строчные множества будут другими, нежели в решении, представленном таблицей 3.1, *в*.

3.2. Для матрицы переходов, представленной на таблице 3.25, построить расширенную матрицу переходов, используя исходную матрицу переходов и, если возможно,

Таблица 3.25

	x_1x_2 00	01	11	10	y_1	y_2	y_3
1	***1***,0	***1***,0	3,—	—,—	0	0	0
					0	0	1
2	***2***,1	1,—	***2***,0	—,—	0	1	0
3	2,—	4,—	***3***,1	—,—	0	1	1
					1	1	1
4	1,—	***4***,1	***4***,1	—,—	1	0	1

допуская некритические состязания. Составить (в виде минимальной логической суммы) выражение для Y_1.

3.3$^+$. Построить K-карту для универсального $(2S_0+1)$-кодирования со сцепленными связанными строчными множествами для таблиц с восемью строками, как описано в разделе 3.1 (см. стр. 109).

3.4. Используя подход с рассмотрением строчных множеств, найти соответствующее кодирование для таблицы 3.26, использующее как можно меньше переменных. Здесь, как и в других случаях, когда значения выходов не указаны, предполагается, что эти значения таковы, что они не допускают объединения строк исходной таблицы.

3.5. Применить общее $(2S_0-2)$-кодирование к таблице 3.27. Нанести Y_3 на K-карту, использующую шесть переменных.

3.6. Рассмотреть $(2S_0-1)$-кодирование таблицы 3.3. Сколько тактов необходимо в общем случае для перехода

Таблица 3.26

	x_1x_2 00	01	11	10
1	***1***	4	5	***1***
2	1	5	***2***	***2***
3	***3***	6	***3***	2
4	***4***	***4***	2	7
5	4	***5***	***5***	6
6	3	***6***	***6***	***6***
7	***7***	6	***7***	***7***

Таблица 3.27

	x_1x_2 00	01	11	10
1	***1***	2	5	4
2	***2***	***2***	3	***2***
3	1	4	***3***	4
4	2	***4***	5	***4***
5	1	***5***	***5***	6
6	***6***	4	3	***6***

Таблица 3.28

	x_1x_2 00	01	11	10
1	***1***	3	7	***1***
2	5	***2***	6	***2***
3	1	***3***	7	***3***
4	***4***	7	6	***4***
5	***5***	7	6	1
6	1	2	***6***	3
7	5	***7***	***7***	2

Таблица 3.29

	x_1x_2 00	01	11	10
1	***1***	4	***1***	3
2	1	***2***	***2***	4
3	4	***3***	2	***3***
4	***4***	***4***	5	***4***
5	6	3	***5***	3
6	***6***	2	1	***6***

из состояния 1001011 в строчном множестве 9 к элементу строчного множества 5?

3.7. Найти минимальное подходящее кодирование для таблицы 3.28.

3.8⁺. Найти минимальное подходящее кодирование для таблицы 3.29 по методу совместного использования строк.

3.9. При кодировании по Хэммингу таблицы с восемью строками, какие переменные должны измениться в каждом из следующих переходов? (а) $6 \to 3$; (б) $2 \to 4$; (в) $5 \to 4$.

3.10. По матрице переходов для таблицы 3.30 (соответствующей кодированию по Лю) найти полностью определенную Y-матрицу, доопределяя, где это возможно, неопределенные значения.

3.11. Найти кодирование по Трэйси для таблицы переходов в задаче 3.10.

3.12⁺. Найти кодирование по Трэйси для таблицы переходов, представленной в таблице 3.31.

Таблица 3.30

	A	B	C	y_1	y_2	y_3	y_4
1	***1***	2	***1***	0	0	0	0
2	***2***	***2***	3	1	0	1	0
3	4	***3***	***3***	0	0	1	1
4	***4***	5	1	0	1	0	1
5	2	***5***	3	1	1	1	—

Таблица 3.31

	x_1x_2 00	01	11	10
1	***1***,0	2,0	3,0	***1***,0
2	1,0	***2***,0	3,0	***2***,1
3	***3***,1	5,0	***3***,0	4,0
4	1,0	—,—	***4***,0	***4***,0
5	3,1	***5***,0	***5***,1	4,0

3.13. Найти кодирование по Трейси для таблицы переходов, представленной в таблице 3.32.

3.14. Найти кодирование по Лю для таблицы 3.33.

Таблица 3.32

	x_1x_2 00	01	11	10
1	***1***	2	3	***1***
2	4	***2***	***2***	***2***
3	4	2	***3***	***3***
4	***4***	5	3	***4***
5	1	***5***	***5***	***5***

Таблица 3.33

	A	*B*	*C*
1	***1***	2	3
2	5	***2***	***2***
3	10	***3***	***3***
4	7	***4***	***4***
5	***5***	6	9

	A	*B*	*C*
6	5	***6***	2
7	***7***	8	2
8	1	***8***	***8***
9	***9***	4	***9***
10	***10***	***10***	2

3.15. Синтезировать таблицу 3.34 с точки зрения получения минимальных сумм выражений для переменных z и y. Использовать однозначное позиционное кодирование.

Таблица 3.34

	x_1x_2 00	01	10
1	***1***,0	6,0	2,0
2	1,0	4,0	***2***,0
3	***3***,1	4,0	5,0
4	3,0	***4***,0	2,0
5	3,0	6,0	***5***,0
6	1,0	***6***,0	5,0

Таблица 3.35

	x_1x_2 00	01	10
1	***1***,0	2,0	3,0
2	3,0	***2***,0	4,0
3	***3***,0	4,—	***3***,0
4	1,0	***4***,1	***4***,0

3.16. Таблица 3.35 должна быть реализована в виде релейно-контактной последовательностной схемы. С точки зрения последующего применения желательно минимизировать расход энергии. Для этого кодирование состояний должно быть выбрано таким, чтобы для всех устойчивых состояний минимальное число реле находились в режиме потребления энергии. Это может повлечь использование большего числа внутренних переменных, чем то, которое было бы необходимо формально. Найти подходящее кодирование, и процесс построения схемы довести до получения выражений для переменных z и y. При включении питания схема всегда должна устанавливаться в состояние 1.

Замечание. Эту задачу можно решить таким образом, что в каждом устойчивом состоянии в режиме потребления находится не более одного реле, а при переходах включаются всегда не более двух реле.

3.17. Найти ООТП-кодирование для таблицы 3.36 и построить полную матрицу переходов.

Таблица 3.36

	A	*B*	*C*
1	***1***,0	2,1	4,0
2	***2***,0	3,0	***2***,0
3	1,0	***3***,1	***3***,1
4	***4***,0	5,0	***4***,0
5	2,1	***5***,0	4,1

Таблица 3.37

	A	*B*	*C*	*D*
1	***1***,00	2,00	6,00	***1***,00
2	3,11	***2***,00	5,00	1,00
3	***3***,11	***3***,10	5,00	1,00
4	1,00	3,10	6,00	***4***,11
5	3,11	***5***,01	***5***,00	1,00
6	1,00	2,00	***6***,00	4,11

3.18. Реализовать таблицу переходов из задачи 3.10, используя позиционное кодирование.

3.19^{+}. Провести параллельную декомпозицию таблицы 3.37. (Возможно решение, состоящее из пары схем, реализующих ООТП-кодирование с использованием одной или двух переменных соответственно, но его нелегко найти. При этом используются системы множеств.)

ГЛАВА 4
ЗАДЕРЖКИ, СОСТЯЗАНИЯ И АНАЛИЗ СХЕМ

В этой главе рассматривается поведение последовательностных схем с учетом задержек в логических элементах и проводах при различных предположениях о распределении задержек. Изучается вопрос построения таблицы переходов, описывающей поведение заданной последовательностной схемы. Мы рассмотрим только схемы Хаффмана и начнем с обсуждения соответствующих свойств задержек.

4.1. Типы задержек

Необходимо различать собственно *элементы задержки*, которые сдвигают во времени свой выходной сигнал относительно входного и могут быть сознательно введены в схему конструктором, и *паразитные задержки*, всегда присутствующие в реальной схеме из-за физических свойств ее компонент (например, задержки распространения сигналов в проводах, задержки, связанные с наличием *RC*-цепей, образованных сопротивлениями схемы и паразитными емкостями, а в релейно-контактных схемах — дополнительные задержки, вызываемые дребезгом контактов (см. раздел 4.6)). Обычно мы будем предполагать, что значения задержек элементов задержки ограничены снизу некоторой величиной и что в каждом конкретном случае этот минимум определяет некоторую максимальную величину, ограничивающую значения задержек сверху (и являющуюся функцией разброса значений задержек, а следовательно, и стоимости применяемых компонент). Также предполагается, что значения паразитных задержек лежат в пределах от нуля до некоторого известного значения. Расположение и относительные величины паразитных задержек в нашей модели не ограничиваются нигде, кроме раздела 4.4. В частности, предполагается, что паразитные задержки присущи любому проводнику.

Схемы, не содержащие элементов задержки, будем называть *схемами без задержек*. Следует понимать, что в таких схемах вовсе не обязательно отсутствуют паразитные задержки.

Совершенной задержкой называется такая задержка, которая преобразует входной сигнал $f(t)$ в выходной сигнал $f(t-D)$. Совершенную задержку обычно называют просто задержкой, ее поведение приближенно моделируется линией передачи, но физических устройств, способных точно воспроизвести поведение совершенной задержки, не существует.

Инерциальная задержка, определенная только в терминах двоичных сигналов, реагирует на изменение входного сигнала спустя время D и только в том случае, если за это время не произошло нового изменения (см. рис. 4.1). Таким образом, если вход Y и выход y инерциальной задержки величины D равны 1 в некоторый момент времени, а затем сигнал Y принимает значение 0 и сохраняет его без перерывов, то выход y будет оставаться равным 1 в течение времени D. Заметим, что в силу введенного определения, если $y=1$, а Y меняет свои значения, причем интервалы времени, в течение которых $Y=0$, равны $D-\varepsilon$, где $0<\varepsilon<D$, а интервалы времени, когда $Y=1$, равны $\delta>0$, то выход y продолжает оставаться равным 1. Естественно, что такой же эффект имеет место, если в приведенных рассуждениях соответственно изменить значения сигналов с нуля на единицу.

Построить реальную схему, которая ведет себя в точности так, как описано выше, невозможно, однако схема, изображенная на рис. 4.2, *а*, является хорошим приближением инерциальной задержки. Рассмотрим действие этой схемы, предполагая, что единичный и нулевой сигналы представляются положительными и отрицательными напряжениями, а мажоритарный элемент M содержит в себе пороговое устройство. Отклонение от идеального поведения инерциальной задержки заключается в том, что δ и ε не могут быть сделаны произвольно малыми. Если прямое сопротивление диодов равно r_f, то δ и ε ограничены снизу некоторой величиной, пропорциональной r_f.

Для многих целей в качестве элемента инерциальной задержки можно использовать более простую схему. Дело

в том, что обычно необходимо устройство, фильтрующее случайные одиночные импульсы малой длительности. Таким устройством является простая RC-цепь, изображенная на рис. 4.2, *б*. Для сглаживания реакции на короткие импульсы к выходу RC-цепи следует подсоединить

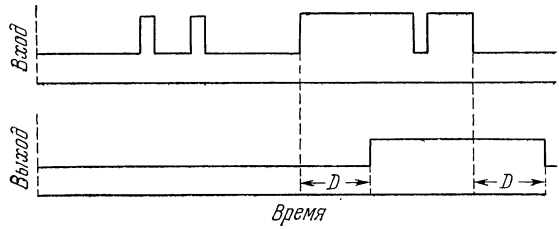


Рис. 4.1. Поведение инерциальной задержки величины D.

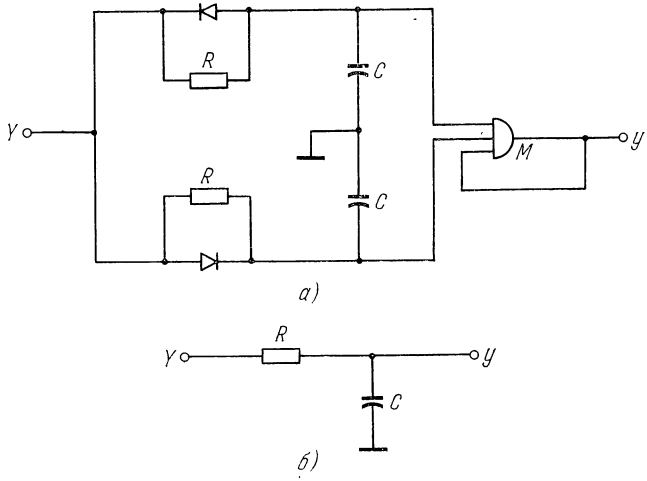


Рис. 4.2. *а*) Почти идеальный элемент инерциальной задержки. *б*) Простая схема элемента инерциальной задержки.

пороговый элемент, в качестве которого можно выбрать, например, триггер Шмитта. Заметим также, что обычное электромагнитное реле с успехом заменяет описанную здесь комбинацию RC-цепи и порогового элемента.

Существует несколько способов построения схем со свойствами инерциальной задержки с помощью элементов

совершенной задержки и логических элементов. Одна из реализаций приведена на рис. 4.3 и содержит мажоритарный элемент, охваченный обратной связью. Входные импульсы (нулевой, если $y = 1$, и единичный, если $y = 0$) длительностью меньше D не проходят на выход y при условии, что период между соседними импульсами

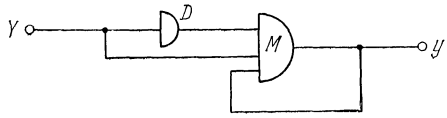


Рис. 4.3. Схема с обратной связью, реализующая инерциальную задержку с помощью элемента совершенной задержки.

не меньше D. Поэтому схема ведет себя почти как идеальная инерциальная задержка с тем ограничением, что δ должно быть больше D.

По некоторым причинам, о которых будет сказано в главе 5, иногда желательно получить аналогичный эффект с помощью схемы без обратной связи. Этому требованию отвечает схема, изображенная на рис. 4.4, хотя

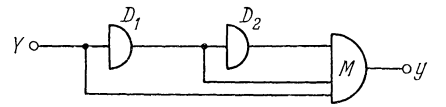


Рис. 4.4. Схема без обратных связей, реализующая инерциальную задержку с помощью элементов совершенной задержки.

ограничения на ее использование более сложные. Входные изменения длительностью, превосходящей $\max(D_1, D_2)$, передаются элементом задержки D_1, в то время как входные импульсы длительностью меньше $\min(D_1, D_2)$ фильтруются при условии, что период между соседними импульсами больше $D_1 + D_2$. Импульсы промежуточной длительности воспроизводятся на выходе с задержкой, а их длительность зависит от соотношения между D_1 и D_2 (см. задачу 4.2).

Обе схемы, содержащие мажоритарные элементы, могут удовлетворительно применяться в качестве элементов инерциальной задержки при условии, что ложные

входные импульсы разделены достаточными временными интервалами. Вопрос о точной классификации различных типов инерциальных задержек в зависимости от их поведения, требований к реализации и применения остается открытым.

В дальнейшем мы обычно будем предполагать, что паразитные задержки могут быть представлены инерциальными задержками или инерциальными задержками, соединенными последовательно с совершенными задержками (см. задачу 4.1).

4.2. Комбинационные состязания

В этом разделе мы обсудим влияние паразитных задержек на переходные процессы в комбинационных переключательных схемах. Как указано в разделе 1.4, паразитные задержки в таких схемах могут вызывать при некоторых изменениях входных сигналов появление на выходе ложных импульсов. Схемы, в которых имеет место описанное явление при некотором распределении паразитных задержек, называются *схемами с комбинационными состязаниями.*

Заметим, что состязания связаны со структурой, а не с физическими особенностями схемы. Данная физическая схема, структура которой допускает возникновение состязаний, может как искажать, так и не искажать выходные сигналы в зависимости от величин и расположения паразитных задержек в данный момент. *Схемой, свободной от состязаний,* называется схема, на выходе которой не появляется искажений независимо от расположения паразитных задержек.

В чисто комбинационной схеме ложные выходные импульсы могут быть вредны в зависимости от того, где используется выход схемы. Например, если схема управляет относительно медленно действующим реле, которое включает или выключает двигатель, то действие ложных импульсов нейтрализуется реле и не наносит ущерба системе. С другой стороны, если схема управляет переключением уличного светофора, то появление на выходе ложных импульсов весьма нежелательно. В случаях, когда комбинационная схема управляет элементами памяти последовательностной схемы, ложные выходные импульсы

могут перевести систему в такое состояние, что дальнейшее ее функционирование будет отличаться от заданного.

Мы начнем с изучения функционирования двухуровневой И — ИЛИ-схемы, считая, что в каждый момент времени допустимы изменения только одной входной переменной, а затем полученные результаты обобщим на более широкий класс схем и на случай одновременного изменения нескольких входных переменных.

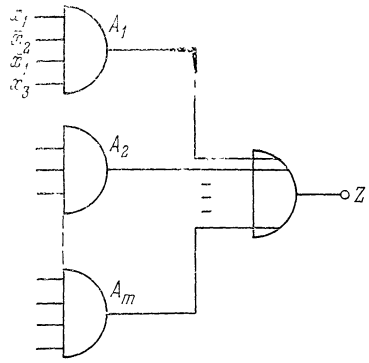


Рис. 4.5. Двухуровневая И — ИЛИ-схема.

На протяжении всей главы мы будем предполагать, что для каждой физической схемы известны пределы изменения значений паразитных задержек, по которым можно установить интервалы следования входных сигналов таким образом, чтобы все изменения, вызванные в схеме некоторым входным импульсом, достигли всех выходов раньше, чем любое изменение, вызванное следующим импульсом, достигнет хотя бы одного выхода. Такое предположение имеет тот же эффект, как и более строгое предположение о том, что в схеме устанавливается устойчивое состояние до того, как начинает вновь изменяться некоторый входной сигнал.

Поставим в соответствие каждому элементу И A_i заданной двухуровневой схемы (см. рис. 4.5) элементарное произведение всех букв, являющихся входами A_i. На рис. 4.5 элементу A_1 соответствует элементарное произведение $p_1 = x_1 x_2 x_1 x_3$. Для дальнейших обобщений нам удобно допустить, чтобы элементарные произведения одного и того же элемента И содержали как переменную, так и ее дополнение, хотя сейчас может показаться, что в этом нет смысла. Комбинационная функция, соответствующая схеме, тогда может быть выражена в виде суммы произведений $\sum_{i=1}^{m} p_i$, где p_i есть элементарное произведение или 0 (например, $p_1 = 0$ в нашем примере).

Пусть два входных состояния I_1 и I_2 отличаются только значениями переменной x_j (т. е. являются соседними по переменной x_j). Тогда будем говорить, что комбинационная схема, реализующая функцию $f(I_k)$, содержит 0-*состязание при переходе между* I_1 *и* I_2 тогда и только тогда, когда (1) $f(I_1) = f(I_2) = 0$; (2) при изменении входной переменной x_j на выходе схемы с начальным входным воздействием I_1 или I_2 может появиться (в зависимости от значений паразитных задержек) сигнал, равный 1.

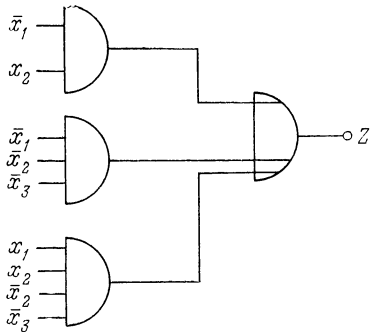


Рис. 4.6. Схема содержащия 0-состязание на переходе между 110 и 100 ($x_j = x_2$) и 1-состязание на переходе между 000 и 010 ($x_j = x_2$).

На рис. 4.6 изображена схема, содержащая 0-состязание.

Для рассматриваемых двухуровневых схем мы можем сформулировать условие существования 0-состязаний.

Лемма 4. 1. *Схема содержит* 0-*состязание при переходе между соседними по переменной* x_j *входными воздействиями* (*наборами*) I_1 *и* I_2 *тогда и только тогда, когда* $f(I_1) = f(I_2) = 0$ *и существует элементарное произведение* p_h, *включающее* x_j *и* $\bar{x}_j$, *а все остальные буквы* p_h *равны* 1 *на наборах* I_1 *и* I_2.

Доказательство. Предположим сначала, что элементарное произведение p_h существует и $f(I_1) = f(I_2) = 0$. Очевидно, что на входных наборах I_1 и I_2 все элементарные произведения равны 0. Без потери общности предположим, что $x_j = 0$ в наборе I_1. Если I_1 — начальный набор и x_j изменяет свое значение, то наличие достаточно большой задержки на входе $\bar{x}_j$ элемента A_h может привести к тому, что на его входах x_j и $\bar{x}_j$ одновременно установится единичное значение. Поскольку остальные входы A_h равны 1, на выходе A_h также появится 1, что в свою очередь приведет к появлению 1 на выходе Z схемы. Такой сигнал будет сохраняться на выходе до тех пор, пока на входе x_j не появится нулевой

сигнал, который вызовет 0 на выходе элемента A_h и на выходе схемы.

Предположим теперь, что схема содержит 0-состязание, т. е. на ее выходе возникает кратковременный единичный импульс после изменения $I_1 \to I_2$. Тогда по крайней мере один элемент должен установиться в 1 во время этого изменения, так как невозможно получить 1 на выходе ИЛИ, если все его входы равны 0. Пусть указанным элементом является A_h. Предположим, что значение некоторой буквы x_q (или $\bar{x}_q$), где $q \neq j$, равно 0 на наборе I_1 и x_q — вход A_h. Ясно, что $x_q = 0$ на наборе I_1, потому что I_1 и I_2 являются соседними по x_j. Тогда, в силу того, что при переходе меняется только x_j, по крайней мере один из входов элемента A_h должен быть равен 0 во время перехода. Следовательно, поскольку мы считаем, что основные логические элементы, составляющие схему, не могут сами по себе генерировать ошибочные импульсы на выходе, выход A_h остается равным 0 на протяжении всего перехода. Но такой вывод противоречит условию, а это означает, что не существует x_q (или $\bar{x}_q$) с указанными свойствами. Таким образом, все входы x_i (или $\bar{x}_i$), где $i \neq j$, элемента A_h имеют значение 1 на наборе I_1 и I_2. Однако элементарное произведение p_h должно быть равно 0 на наборах I_1 и I_2 (так как $f(I_1) = f(I_2) = 0$), а так может случиться только в том случае, когда x_j и $\bar{x}_j$ одновременно являются компонентами p_h.

Простое следствие леммы 4.1 заключается в том, что реализация любой комбинационной функции в виде двухуровневой И — ИЛИ-схемы свободна от 0-состязаний, если на входах любого элемента И не присутствуют вместе переменные и их отрицания.

Введем следующее определение: будем говорить, что схема содержит *1-состязание при переходе между* соседними по x_j *входными наборами* I_1 *и* I_2, если $f(I_1) = f(I_2) = 1$ и на выходе схемы во время перехода между I_1 и I_2 может кратковременно возникнуть сигнал 0. На рис. 4.6 изображена схема, содержащая 1-состязание. По отношению к И—ИЛИ-схемам эта ситуация не является двойственной предыдущей (она будет двойственной, если рассматривать ИЛИ — И-схемы). Лемма 4.2 определяет условия существования 1-состязаний в схеме вида рис. 4.5.

Лемма 4.2. *Схема содержит 1-состязание при переходе между наборами I_1 и I_2, где $f(I_1) = f(I_2) = 1$, тогда и только тогда, когда не существует элементарных произведений, имеющих значение 1 как на наборе I_1, так и на наборе I_2.*

Доказательство. Предположим, что существует такое элементарное произведение p_f, что $p_f = 1$ для входных наборов I_1 и I_2. Тогда переменные x_j и $\bar{x}_j$ не могут быть его компонентами; следовательно, все компоненты p_f не меняются во время перехода между I_1 и I_2. Но это означает, что p_f постоянно обеспечивает единичный сигнал на входе элемента ИЛИ, и поэтому на выходе ИЛИ на протяжении перехода сохраняется сигнал, равный 1, т. е. 1-состязание отсутствует.

Предположим теперь, что не существует элементарного произведения, равного 1 на обоих наборах I_1 и I_2. Тогда множество P_1 всех элементарных произведений, равных 1 на наборе I_1, должно быть непустым и непересекающимся с непустым множеством P_2 всех элементарных произведений, равных 1 на наборе I_2. Пусть начальным входным набором является I_1, тогда на выходах элементов И, соответствующих множеству P_1, появляется 1, а на выходах элементов И, соответствующих множеству P_2, появится 0. При изменении x_j значения выходов элементов указанных множеств соответственно изменятся на 0 и 1. Предположим, что паразитные задержки на выходах всех элементов множества P_2 больше задержек на выходах всех элементов множества P_1. Тогда существует интервал времени, в течение которого все входы элемента ИЛИ, соединенные с элементами из множества P_1, станут равными 0, а все входы того же элемента ИЛИ, связанные с элементами множества P_2, еще не примут значения 1. На протяжении этого интервала все входы ИЛИ, а, следовательно, и выход ИЛИ имеют значение 0, т. е. в схеме возникает 1-состязание.

Если заданная комбинационная функция произвольного вида представлена в виде суммы произведений и при этом не только каждое ее единичное значение покрывается некоторым элементарным произведением (обычное требование), но также каждая пара ее соседних единичных значений покрывается отдельным элементарным произведением, то из леммы 4.2 следует, что И—ИЛИ-схема,

реализующая данную функцию, не содержит 1-состязаний, а из леммы 4.1 вытекает, что такая схема свободна и от 0-состязаний. По тем же причинам в случае, когда интерес представляют лишь установившиеся значения выходных сигналов, наиболее экономичные двухуровневые реализации получаются, если элементарными произведениями являются только простые импликанты. При синтезе схем могут использоваться K-карты или таблицы простых импликантов. При использовании последних для каждой соседней пары единичных значений функции таблицу дополняют столбцом, содержащим 1 в каждой строке, соответствующей простому импликанту, покрывающему данную пару.

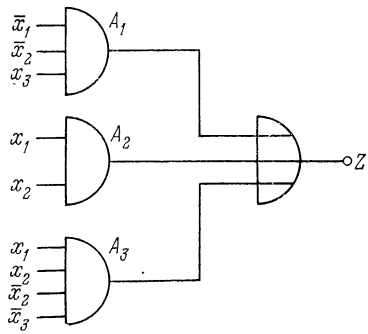


Рис. 4.7. Схема, содержащая динамическое состязание на переходе между 100 и 110.

До сих пор мы рассматривали *статические состязания*, возникающие при входном изменении, в течение которого выход должен оставаться постоянным. Однако возможно также существование состязаний, вызванных входным изменением, при котором изменяется и значение выхода схемы. Этот тип состязаний проявляется в виде дополнительных пар выходных изменений, происходящих до того, как выход примет значение, соответствующее конечному входному набору. Предположим, что I_1 и I_2 являются соседними по переменной x_j входными наборами. Тогда будем говорить, что схема содержит *динамическое состязание при переходе между* I_1 *и* I_2, если $f(I_1) \neq f(I_2)$, и кроме того, из-за наличия паразитных задержек существует возможность трехкратного изменения Z за время перехода от I_1 к I_2. Схема, изображенная на рис. 4.7, содержит такое состязание, проявляющееся на выходе Z, если паразитная задержка на входе x_2 элемента A_2 (или на выходе A_2) больше задержки на входе $\bar{x}_2$ элемента A_3, а задержка на входе x_2 элемента A_3 пренебрежимо мала. В этом случае, если во входном наборе 100 меняется x_2, то выход Z, первоначально равный 0,

изменит свое значение на 1, как только через элемент A_3 пройдет изменение x_2, затем Z снова примет значение 0, поскольку через A_3 пройдет изменение значения входа $\bar{x}_2$, и, наконец, Z установится в конечное состояние, равное 1, так как элемент A_2 передаст изменение $0 \to 1$ его входа x_2. Условие существования динамических состязаний в двухуровневых И—ИЛИ-схемах определяется следующей леммой.

Лемма 4.3. *Двухуровневая И—ИЛИ-схема содержит динамическое состязание при переходе между соседними по переменной x_j входными наборами I_1 и I_2 в том и только в том случае, когда $f(I_1) \neq f(I_2)$ и схема имеет элементарное произведение p_f, содержащее x_j и $\bar{x}_j$, причем остальные переменные, входящие в него, равны 1 на I_1 и I_2.*

Доказательство. Предположим, что такое элементарное произведение p_f существует. Тогда, как и в доказательстве леммы 4.1, на выходе элемента A_f, соответствующего p_f, может произойти изменение $0 \to 1 \to 0$ во время перехода от I_1 к I_2. Предположим сначала, что $f(I_1) = 0$ и $f(I_2) = 1$. Пусть P_2 — множество всех элементарных произведений, принимающих значение 1 на I_2. Тогда на I_1 все элементарные произведения из P_2 имеют значение 0. Если значения паразитных задержек на выходах элементов И из множества P_2 достаточно велики, то последовательность 010 на выходе A_f пройдет на выход Z в то время, как выходы остальных элементов И равны 0, т. е. на выходе Z также появится последовательность 010. Когда единичный сигнал с выхода одного из элементов множества P_2 достигнет Z, произойдет третье и последнее изменение выхода Z. Чтобы завершить доказательство условия необходимости леммы, заметим, что в случае, когда $f(I_1) = 1$ и $f(I_2) = 0$, при наличии относительно больших задержек на входах x_j и $\bar{x}_j$ элемента A_f выходы элементов И, удерживающих Z в значении 1 на наборе I_1, станут равными 0, вызывая тем самым $Z = 0$, раньше, чем последовательность 010 появится на выходе элемента A_f; таким образом, выходная переменная трижды изменится, реализуя последовательность 1010.

Предположим, что при переходе $I_1 \to I_2$ имеет место динамическое состязание. Если $f(I_1) = 0$ и $f(I_1) = 1$, то наличие состязания соответствует появлению последо-

вательности 0101 на выходе Z при $I_1 \to I_2$. Если же $f(I_1) = 1$ и $f(I_2) = 0$, на выходе возникает последовательность 1010. В обоих случаях наличие последовательности 010 в выходном сигнале означает, что некоторый элемент A дважды за время перехода меняет свое значение. Но, как показано во второй части доказательства леммы 4.1, к такому эффекту приводит наличие соответствующего этому элементу элементарного произведения p_f со свойством, заданным в формулировке леммы.

Из леммы 4.3 немедленно следует, что упомянутая выше процедура синтеза двухуровневых схем, свободных от статических состязаний, приводит к построению схем, свободных и от динамических состязаний, поскольку ни один элемент не имеет одновременно входов x_i и $\bar{x}_i$ для любого i. Заметим, однако, что схема, свободная от статических состязаний, может содержать динамические состязания, как, например, схема, изображенная на рис. 4.7.

Наличие двух и более элементарных произведений p_f, покрывающих некоторый переход, может привести к возникновению стольких пар ложных изменений выходного сигнала во время перехода, сколько существует таких элементарных произведений. Это утверждение справедливо для динамических и 0-состязаний. Поэтому после входного изменения $I_1 \longrightarrow I_2$ могут получаться выходные последовательности 01010 или 010101. На переходе, вызывающем 1-состязание, может возникнуть единственный ложный нулевой импульс, если не существует элементарных произведений, покрывающих значения выхода на обоих входных наборах.

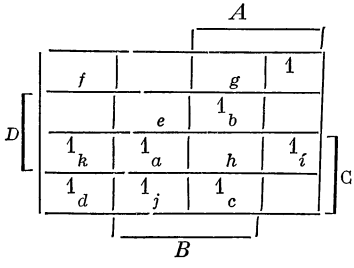
Как показано выше, всегда можно построить И—ИЛИ-схему, свободную от комбинационных состязаний, вызванных изменением одной входной переменной. Единственным недостатком синтезированной схемы является увеличение числа составляющих ее элементов, однако полученная схема защищена от неправильного функционирования вне зависимости от распределения паразитных задержек и не снижает быстродействия. В некоторых случаях можно избежать возникновения состязаний за счет введения элементов задержки, обеспечивающих правильную работу схемы при определенных входных изменениях. Например, элемент задержки, включенный на выходе одного из элементов И из множества P_1 (см. лемму

4.2), предохраняет схему от 1-состязаний, вызываемых данным элементом И. Такое решение не всегда приводит к хорошим результатам, поскольку введенная задержка, защищая схему от одного состязания, может способствовать возникновению другого. Кроме того, задержка может уменьшить быстродействие схемы.

Другим методом обеспечения правильной работы схемы является присоединение к ее выходу элемента инерциальной задержки, который фильтрует выходной сигнал, подавляя возникающие в нем импульсы, вызванные состязаниями. Недостаток этого метода заключается в снижении быстродействия, так как увеличиваются времена нарастания и спадания выходного сигнала, если в состав инерциальной задержки не входит пороговый элемент. Однако пороговые элементы дороги и могут также еще больше замедлять действие схемы. На основании приведенных соображений наиболее целесообразным способом борьбы с состязаниями является построение схем, свободных от состязаний.

Таблица 4.1

K-карта функции, содержащей состязания различных типов при изменениях нескольких входных переменных



Рассмотрим действие одновременного изменения двух и более входных переменных. В частности, будем предполагать, что функция, заданная K-картой (табл. 4.1), реализуется И—ИЛИ-схемой. Очевидно, что не существует ни одного элементарного произведения функции, а, следовательно, ни одного элемента И, покрывающего одновременно две единицы, помеченные на K-карте буквами a и c. Если паразитные задержки на выходах элементов И, покрывающих c, больше, чем задержки на выходах И, покрывающих a, то ошибочный сигнал 0 появится на выходе при входном изменении от a к c (т. е. при одновременном изменении переменных A и D). Поэтому для данной функции 1-состязание неустранимо.

Создавшуюся ситуацию удобно анализировать, учитывая влияние порядка изменения отдельных входных пере-

менных на выходной сигнал и принимая во внимание паразитные задержки. При таком подходе рассматриваются кажущиеся последовательности входных наборов, которые можно выявить по реакции схемы. В рассматриваемом нами случае входное изменение $a \longrightarrow c$ выход может воспринять как последовательность $a \longrightarrow h \longrightarrow c$, в результате которой на выходе появится ложный сигнал в ответ на кажущийся входной набор h. Никакого ложного сигнала не было бы, если бы паразитные задержки были распределены таким образом, чтобы кажущейся была входная последовательность $a \longrightarrow j \longrightarrow c$. Это свидетельствует о том, что состязание между переменными A и D устранимо за счет введения элементов задержки так, чтобы при рассматриваемом переходе процессы, вызываемые в схеме изменением переменной D, достигали выхода первыми, делая, таким образом, кажущимся входным набором набор j.

В некоторых случаях рассмотренный метод выявления состязаний может быть полезен, хотя те же соображения применимы и для устранения состязаний при изменении одной входной переменной. Для некоторых переходов, например, таких, как $a \longrightarrow b$, метод не дает никаких результатов, поскольку вне зависимости от порядка изменения входных переменных промежуточному входному набору (e или h) соответствует значение выхода, равное 0. Другими словами, на K-карте не существует пути через взаимно соседние клетки со значениями 1, соединяющего клетки a и b. Так как не всегда возможно введение в схему элементов задержки так, чтобы все входные изменения воздействовали на выход одновременно, то остается одно средство борьбы с появлением ложных импульсов: использование инерциальной задержки для фильтрации выходного сигнала. Процессы, происходящие в схеме в обоих рассмотренных случаях (при переходах $a \longrightarrow b$ и $a \longrightarrow c$), классифицируются как *функциональные состязания,* чтобы подчеркнуть тот факт, что такие состязания присущи собственно функции, а не какой-либо ее схемной реализации.

В третьем случае при переходе $a \longrightarrow d$ каждый путь минимальной длины, соединяющий клетки 1_a и 1_d, проходит только через клетки с единичным значением функции. Иначе говоря, подкуб, натянутый на две вершины

n-мерного куба, соответствует элементарному произведению функции. Поэтому рассматриваемый участок карты можно покрыть единственным элементом И, выходной сигнал которого постоянно равен 1 на переходе $a \longrightarrow d$, и тем самым устранить возможность появления на выходе схемы ложного сигнала. Действительно, очевидно, существует простой импликант, покрывающий рассматриваемый подкуб; следовательно, если функция реализуется И—ИЛИ-схемой, причем элементы И соответствуют простым импликантам функции, то не существует неустранимого 1-состязания. Если некоторая схема содержит такое устранимое 1-состязание (в нашем примере оно может иметь место, если в схеме нет элемента, реализующего произведение $\bar{A}C$, а клетки a и d покрываются элементами, соответствующими произведениям $\bar{A}BC$ и $\bar{A}\bar{B}C$), то говорят, что существует *логическое* 1-*состязание*, определяя этим, что ложное поведение присуще построенной схеме, а не реализуемой функции. Нетрудно видеть, что наличие логического 1-состязания означает отсутствие элемента И, соответствующего некоторому простому импликанту.

Аналогичная ситуация существует и для 0-состязаний. Переходы $e \longrightarrow h$, $e \longrightarrow g$, $e \longrightarrow f$ иллюстрируют случаи, когда соответственно ни один, несколько (но не все) и все пути на K-карте свободны от возникновения ложных выходных импульсов. В первых двух случаях имеют место *функциональные* 0-*состязания*, тогда как возможность появления на выходе последовательности 010 при переходе $e \longrightarrow f$ означает наличие *логического* 0-*состязания*. Заметим, что последнее указывает на наличие элемента И с прямыми и инверсными входами, соответствующими одной или более переменным, изменяющимся на данном переходе (в нашем примере — A и D). Все результаты, связанные с изменением нескольких входных переменных, объединяются следующими двумя теоремами, доказательством которых являются приведенные выше рассуждения.

Теорема 4.1. *Функциональное состязание имеет место при входном переходе $p \longrightarrow q$ тогда и только тогда, когда на K-карте существует путь минимальной длины от p к q, вдоль которого значение функции меняется более одного раза.*

Теорема 4.2. И—ИЛИ-*схема свободна от статических логических состязаний, если реализуемые ее элементами* И *элементарные произведения взаимнооднозначно соответствуют простым импликантам реализуемой функции. Схема содержит логическое* 1-*состязание тогда и только тогда, когда существует некоторый простой импликант функции, не представленный элементарным произведением, реализуемым элементом* И. *Схема содержит логическое* 0-*состязание тогда и только тогда, когда для некоторой пары входных наборов* I_1 *и* I_2 *таких, что* $f(I_1) = f(I_2) = 0$, *существует реализуемое элементом* И *элементарное произведение, не имеющее букв, равных* 0 *как на* I_1, *так и на* I_2, *и имеющее буквы в прямом и инверсном виде, представляющие входные переменные, которые меняют свои значения при переходе между* I_1 *и* I_2 (см. задачу 4.20).

Исследование динамических состязаний сильно усложняется, если допустимы изменения нескольких входных переменных. К счастью, они не слишком важны, и поэтому мы ограничимся указанием некоторых возможностей. В таблице 4.1 все пути минимальной длины, соединяющие g и i, включают три изменения значения функции и, кроме того, существует три пути, соединяющих g и a и обладающих тем же свойством. Поэтому переходы между указанными парами входных наборов вызывают динамические функциональные состязания. Путей, соединяющих d и e, с аналогичным свойством нет, так что мы можем заключить, что при переходе $d \longrightarrow e$ не существует динамического функционального состязания. Смысл полученного вывода заключается в том, что данная функция может быть реализована без указанного состязания. Однако, как мы сейчас покажем, эта реализация может быть построена только за счет введения в других случаях устранимых статических состязаний при различных переходах.

Рассмотрим клетку i с единичным значением, которую можно покрыть существенно простым импликантом $\bar{B}CD$. Мы обязаны использовать этот импликант для того, чтобы избежать 1-состязания на переходе $i \longrightarrow k$. Но тогда во время перехода $d \longrightarrow e$ элемент И, реализующий импликант $\bar{B}CD$, может выдать значение 1, и, если на его выходе задержка относительно велика, то это значение появится на выходе схемы только после того, как выход

элемента И, реализующего простой импликант $\bar{A}C$, станет равным 0. Итак, выход Z может стать равным 0, когда $\bar{A}C = 0$; равным 1, когда $\bar{B}CD = 1$, и, наконец, снова стать равным 0 при $\bar{B}CD = 0$. Аналогичная ситуация вызывается элементарным произведением $BC\bar{D}$, необходимым для покрытия клетки c.

Кроме уже обсужденных типов динамических состязаний, имеется более распространенный тип, обусловленный наличием элементарных произведений, содержащих входные переменные и их дополнения.

Таким образом, любая функция реализуема схемой, свободной от всех комбинационных состязаний, вызываемых изменением одной переменной. Такая схема называется *свободной от состязаний*. Однако при возможности изменения нескольких входных переменных любая функция, имеющая более одного простого импликанта, содержит функциональные состязания, которые невозможно устранить при синтезе логической схемы. В таких случаях количество статических состязаний можно уменьшить, используя все простые импликанты функции; гораздо сложнее разрешить проблему исключения динамических состязаний, вызванных изменением нескольких входных переменных,— об этом речь будет идти позже.

Обобщим теперь результаты, полученные для И—ИЛИ-схем, на схемы произвольного вида. Для этого изучим, как различные преобразования схем влияют на различные виды состязаний. Кроме того, выделим множество преобразований, приводящих любую схему к эквивалентной И—ИЛИ-схеме с тем же множеством состязаний. Далее будут представлены специальный метод для выявления динамических состязаний и совокупность преобразований двухуровневой схемы в многоуровневую без добавления новых состязаний. Докажем сначала две леммы, позволяющие нам оперировать с частями схем.

Лемма 4.4. *Преобразования схемы* (см. рис. 4.8), *соответствующие для элементов ИЛИ замене подформулы* $A + (B + C)$ *на подформулу* $A + B + C$ *и наоборот, не приводят к появлению новых состязаний, так же как и соответствующие преобразования для элементов И.*

Доказательство. При изучении рис. 4.8 становится очевидным, что единственным отличием двух подсхем

является длина некоторых путей прохождения сигналов, а здесь длины путей не могут вызывать или уничтожать состязания.

Определим *подсхему* заданной схемы C как схему, содержащуюся в C, имеющую единственный выход, который может являться выходом схемы C или вести к другим элементам схемы C, и имеющую входами либо входы схемы C, либо выходы элементов схемы C.

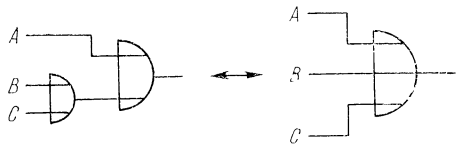


Рис. 4.8. Преобразование схемы, соответствующее $A + (B + C) = |= A + B + C$.

Лемма 4.5. *Преобразование схемы C, состоящее в применении к ее подсхеме преобразования, сохраняющего состязания в подсхеме, сохраняет состязания в C.*

Доказательство. Пусть S — первоначальная подсхема и S^* — преобразованная. Поскольку множество выходов S^* всегда совпадает с множеством выходов S, замена S на S^* в схеме C не оказывает никакого влияния на выход C.

Лемма 4.6. *Схема, двойственная заданной, т. е. схема, в которой осуществлена замена элементов И на ИЛИ и наоборот, реализует двойственную функцию с двойственными состязаниями.*

Доказательство. При решении любого вопроса, рассматриваемого в этой главе, мы используем взаимную замену элементов И и ИЛИ и значений 0 и 1 для получения двойственных утверждений, которые так же истинны, как и прямые утверждения. Этот факт вытекает из аксиом двойственности булевой алгебры. Таким образом, если заданная И—ИЛИ-схема содержит статическое 1-состязание при переходе $I_1 \longrightarrow I_2$, то двойственная схема, т. е. ИЛИ — И-схема, содержит 0-состязание при переходе между входными наборами, двойственными I_1 и I_2 (полученными взаимной заменой 0 и 1 во всех компонентах наборов I_1 и I_2).

Лемма 4.7. *Введение или удаление инверторов на выходе схемы не приводит к возникновению новых состязаний, а также не исключает существующие, но приводит к взаимной замене* 0- *и* 1-*состязаний. Введение или удаление инверторов на входах схемы не исключает существующие состязания и не приводит к возникновению новых, но существующие состязания возникают на переходах, двойственных переходам, первоначально вызывавшим состязания.*

Доказательство очевидно.

Лемма 4.8. *Преобразование схемы, соответствующее применению обобщенной теоремы де Моргана (взаимная замена элементов И и ИЛИ и инвертирование входов и выходов), является сохраняющим состязания.*

Доказательство. Инвертирование в заданной схеме C входов, согласно лемме 4.7, сохранит состязания, присущие схеме C, на переходах, двойственных первоначально вызывавшим эти состязания. Переход к схеме C^*, двойственной C, приведет к возникновению на указанных переходах состязаний, двойственных исходным, в силу леммы 4.6. И наконец, инвертирование выходов схемы C^*, согласно лемме 4.7, восстановит в схеме состязания исходного вида. Очевидно, что результирующая схема реализует исходную функцию.

Лемма 4.9. *Преобразование схемы, соответствующее применению дистрибутивного закона, сохраняет статические состязания.*

Доказательство. Мы должны показать, что при предположении, что каждая переменная A, B и C изменяется не более одного раза, схемам, представляющим функции $A(B+C)$ и $AB+AC$, свойственно одинаковое поведение на переходах, если установившееся значение $A(B+C)$ не меняется при входном изменении. (Мы, конечно, знаем, что в установившемся состоянии обе схемы ведут себя одинаково.)

1. Рассмотрим сначала 0-состязания. На переходах, в которых A зафиксировано в 0, ни в одной из схем не может возникнуть состязаний. Если значение A зафиксировано в 1, обе схемы соответствуют $B+C$, и поэтому не могут иметь 0-состязаний. Если A изменяется из 0 в 1, то в схеме, представляющей $A(B+C)$, возможно 0-состязание, если $B+C$ изменяется из 1 в 0 одновременно с A.

Изучение схемы, представляющей $AB + AC$, показывает, что ей также присуще такое состязание. Кроме того, в обеих схемах имеет место 0-состязание, если A изменяется из 1 в 0, а $B + C$ изменяется из 0 в 1. Следовательно, во всех случаях выражения $A(B + C)$ и $AB + AC$ соответствуют схемам, содержащим одинаковые 0-состязания.

2. Что касается 1-состязаний, то совершенно очевидно, что они возникают в обеих схемах только в том случае, когда значение A постоянно равно 1, а B и C изменяются в противоположных направлениях. Поэтому преобразование, соответствующее применению дистрибутивного закона, ни устраняет, ни добавляет новых 1-состязаний, что устанавливает справедливость леммы. Заметим, что лемма верна только по отношению к статическим состязаниям, так как в схеме, соответствующей $A\bar{A} + AB$, возникает динамическое состязание при $B = 1$ и изменении A, в то время как схема, соответствующая скобочной форме $A(\bar{A} + B)$, свободна от динамического состязания на этом переходе.

Однако лемма 4.9 частично справедлива и для динамических состязаний.

Лемма 4.10. *Если в схеме, соответствующей* $A(B + C)$, *на некотором переходе возможно динамическое состязание, то в любой схеме, соответствующей* $AB + AC$, *также возможно динамическое состязание на этом переходе* *).

Доказательство. Как уже показано, на любом переходе, когда значение A фиксировано, схема, соответствующая $A(B + C)$, воспроизводит на выходе либо константу 0 (при $A = 0$), либо $B + C$ (при $A = 1$). В каждом случае динамического состязания возникать не должно. Если A изменяется из 0 в 1, то условие возникновения динамического состязания требует, чтобы равенство $B + C = 1$ выполнялось после окончания этого изменения, а во время перехода $B + C$ изменялось по закону $1 \longrightarrow 0 \longrightarrow 1$. Отсюда следует, что при этом B и C должны изменяться в противоположных направлениях, что в свою очередь позволяет положить $B = A$ и $C = \bar{A}$. Но в этом случае схема, соответствующая $AB + AC$, ведет себя как

*) Как уже показано, обратное утверждение неверно.

схема, представляющая $AA + A\bar{A}$, в которой также возможно динамическое состязание. Поскольку сходные аргументы могут быть приведены для доказательства существования динамического состязания при другом направлении изменения A, то справедливость леммы можно считать установленной.

Обе предшествующих леммы, как и лемма 4.4, останутся справедливы, если заменить $B + C$ суммой более чем двух слагаемых, так как основные идеи доказательства при этом не изменятся.

Аналогичным способом докажем следующую лемму.

Лемма 4.11. *Если в схемах, соответствующих выражениям A или $A + B$, возможны состязания на любых переходах, то в схемах, соответствующих $A+AB$ или $A+\bar{A}B$, также возможны состязания на этих же переходах.*

Доказательство. Часть доказательства, относящаяся к преобразованию $A + AB \longrightarrow A$, тривиальна, поскольку очевидно, что в схеме, соответствующей A, невозможны никакие состязания. Обратное преобразование может внести новое состязание, как показано в задаче 4.10. В схеме, представляющей $A + B$, возникает состязание только при $B = \bar{A}$, что для $A + \bar{A}B$ приводит к $A + A\bar{A}$, откуда следует, что в схеме, соответствующей $A + \bar{A}B$, также возможно такое же 1-состязание. Следовательно, немедленно устанавливается факт того, что преобразование $A + \bar{A}B \longrightarrow A + B$ новых состязаний не вносит.

Теперь мы можем сформулировать две важные теоремы, касающиеся анализа схем произвольного вида на возникновение в них статических состязаний и синтеза многоуровневых логических схем, свободных от логических состязаний. Доказательства этих теорем непосредственно следуют из семи предыдущих лемм, начиная с леммы 4.4.

Теорема 4.3. *Если преобразовать заданное алгебраическое выражение $\mathscr{E}$ произвольного вида к выражению $\mathscr{G}$ в форме суммы произведений с помощью ассоциативного и дистрибутивного законов и правила де Моргана (что всегда возможно), то схемам, соответствующим $\mathscr{E}$ и $\mathscr{G}$, присущи одни и те же статические состязания.*

Поэтому многоуровневые схемы можно анализировать на возникновение в них статических состязаний, приводя

соответствующие алгебраические выражения к выражению в виде суммы произведений с последующим применением лемм 4.1 и 4.2. Последовательно-параллельные контактные схемы могут быть проанализированы аналогичным образом после получения соответствующих алгебраических выражений. Для анализа мостиковых контактных схем следует сначала построить элементарные произведения для каждого пути схемы и затем снова использовать леммы 4.1 и 4.2. Выявление динамических состязаний мы обсудим после того, как сформулируем теорему синтеза.

Теорема 4.4. *Если для заданного выражения* $\mathscr{E}_1$ *произвольного вида построить выражение* $\mathscr{E}_2$, *используя только обобщенное правило де Моргана, ассоциативный закон, вынесение общих множителей за скобки* (*но не раскрытие скобок*) *и преобразования вида* $A+AB \longrightarrow A$ *и* $A+\bar{A}B \longrightarrow A+B$, *то схема, соответствующая* $\mathscr{E}_2$, *будет свободна от комбинационных состязаний, которые невозможны в схемах, соответствующих* $\mathscr{E}_1$.

Таким образом, при использовании описанной ранее процедуры синтеза И—ИЛИ-схем, свободных от логических состязаний (включая динамические состязания), мы можем применять указанные преобразования для построения многоуровневых схем с требуемыми свойствами.

Хотя наши исследования проведены в терминах схем, состоящих из элементов типа И, ИЛИ и инверторов, полученные результаты применимы к логическим схемам, содержащим и другие более сложные элементы, например, типа И—НЕ. Для анализа такие схемы могут быть преобразованы простой заменой каждого сложного элемента эквивалентной схемой, свободной от состояний, построенной в базисе И, ИЛИ, НЕ. При синтезе следует построить схему, использующую элементы И, ИЛИ, НЕ, а затем, применяя теорему 4.3, преобразовать ее в схему с элементами требуемого базиса. Процедура оптимального синтеза свободных от состязаний схем в произвольном базисе в общем случае является сложным делом, однако возникающие трудности имеют тот же характер, что и в случае, когда мы не заботимся о состязаниях.

Последовательно-параллельные контактные схемы можно конструировать аналогичным образом, но синтез мостиковых контактных схем, по-видимому, не поддается

эффективной формализации, независимо от того, интересуемся мы состязаниями или нет.

Для выявления динамических состязаний необходимо различать пути распространения сигналов по схеме, определяя связи между теми множествами путей, по которым сигналы могут распространяться до определенного выхода. Недостаточно знать, например, что в случае распространения сигналов с входов A, $\bar{B}$ и C до выхода Z значение Z изменяется. Мы должны знать, по каким именно путям происходит распространение сигналов A, $\bar{B}$ и C. Выяснить это можно за счет приписывания числовых индексов в порядке возрастания слева направо каждой букве, встречающейся в выражении, соответствующем схеме; например, выражение

$$AB + \bar{B}C(\bar{A} + D) + [(\bar{A} + D)E + B](\overline{CD})$$

принимает вид

$$A^1B^2 + \bar{B}^3C^4(\bar{A}^5 + D^6) + [(\bar{A}^7 + D^8)E^9 + B^{10}](\overline{C^{11}D^{12}}).$$

Два элемента с различными индексами обозначают различные пути распространения сигналов в схеме. В нашем примере $\bar{A}^5$ относится к пути, начинающемуся от входа $\bar{A}$, проходящему через элемент ИЛИ, на другом входе которого действует сигнал D, затем через элемент И, на который также воздействуют сигналы $\bar{B}$ и C, и, наконец, через элемент ИЛИ, получающий сигналы от двух других подсхем. Сигнал $\bar{A}^7$ представляет совершенно другой путь между теми же двумя полюсами схемы. Этот путь может сначала проходить через тот же элемент ИЛИ, если $\bar{A} + D$ воспроизводится как подфункция, а затем подается в две различные точки схемы, но в любом случае рассматриваемый путь проходит через элемент И, на втором входе которого действует сигнал E, потом через элемент ИЛИ с сигналом B на другом входе, через элемент И, к входу которого подведен сигнал $\overline{CD}$, и, наконец, через последний трехвходовый элемент ИЛИ на выход Z.

При обсуждении перехода между двумя входными наборами I_1 и I_2 (необязательно соседними) будем разделять буквы на четыре типа: f_0- и f_1-буквы, равные соответственно 0 и 1 во время перехода, и c_0- и c_1-буквы, из-

меняющиеся из 1 в 0 и из 0 в 1 соответственно на переходе $I_1 \longrightarrow I_2$. Теперь мы в состоянии сформулировать основную теорему для выявления динамических состязаний.

Теорема 4.5. *Пусть заданной схеме соответствует выражение $\mathscr{E}$, реализующее функцию f такую, что $f(I_1) = 0$ и $f(I_2) = 1$, и пусть $\mathscr{G}$ есть представление $\mathscr{E}$ в виде суммы произведений, полученное путем приписывания индексов буквам выражения $\mathscr{E}$ и раскрытия скобок с применением, если это необходимо, правила де Моргана. При этом динамическое состязание возможно на переходе $I_1 \longrightarrow I_2$ тогда и только тогда, когда существует элементарное произведение p_j, не содержащее f_0-букв, но содержащее по меньшей мере по одной c_0- и c_1-букве, такое, что каждое элементарное произведение, равное 1 на наборе I_2 (т. е. не содержащее ни c_0-, ни f_0-букв), содержит не менее одной c_1-буквы, не входящей в p_j.*

Заметим, что эта теорема применима для выявления всех динамических состязаний, как логических, так и функциональных, на переходах с изменением любого числа входных переменных.

Доказательство. 1. Предположим сначала, что условия теоремы выполняются. Тогда, если задержки некоторого пути схемы, соответствующего одной из c_0-букв элементарного произведения p_j, достаточно велики по сравнению с задержками каждого из путей, соответствующих c_1-буквам p_j, то сигнал 1 достигнет выхода Z за счет равенства единице произведения p_j. Следовательно, Z изменяется из 0 в 1. Теперь для каждого p_k, равного 1 на наборе I_2, пусть c_{1k} является одной из c_1-букв, содержащихся в p_k и не входящих в p_j (согласно условию теоремы, хотя бы одна такая буква существует), пусть задержка пути, соответствующего c_{1k}, больше, чем задержки путей, соответствующих c_0-буквам, принадлежащим p_j. Тогда распространение единичных сигналов до выхода Z за счет произведений p_k произойдет только после того, как c_0-сигналы элементарного произведения p_j достигнут выхода Z и значение Z станет равным 0. Наконец, если некоторое другое элементарное произведение p_t также может воспроизвести кратковременный единичный импульс, то длительность этого импульса либо за счет увеличения задержек путей, соответствующих c_1-буквам p_t, либо за счет уменьшения задержек путей, соответствующих c_0-

буквам p_t, может быть сделана такой, что импульс не распространится до выхода, пока p_t не примет нулевого значения, при допущении, что p_t содержит хотя бы одну c-букву, не входящую в p_j; или же p_t содержит то же множество c-букв, что и p_j, и тогда импульсы, воспроизводимые p_t и p_j, точно совпадают. Следовательно, сигнал Z под действием p_j примет сначала значение 1 и затем изменится в 0 и, наконец, установится в 1 под действием некоторого p_k. Таким образом, динамическое состязание возможно, что доказывает условие необходимости.

2. Допустим теперь, что динамическое состязание возможно на переходе $I_1 \longrightarrow I_2$. Тогда некоторое элементарное произведение должно измениться из 0 в 1, а затем из 1 в 0. Так может изменяться только такое элементарное произведение, которое удовлетворяет условиям на p_j в формулировке теоремы. Для того чтобы действие p_j не было подавлено единичным сигналом, воспроизводимым другим элементарным произведением, должно существовать такое распределение паразитных задержек, которое препятствует установлению в 1 любого элементарного произведения, равного 1 на наборе I_2, прежде чем p_j примет значение 0. Так может случиться только в том случае, если такое произведение содержит c_1-букву, не входящую в p_j, так как c_1-буквы соответствуют путям, задержки которых должны быть достаточно малы. Этим завершается доказательство условия достаточности.

В контактных схемах индексы при буквах обозначают различные контакты, а элементарные произведения — пути прохождения сигналов в схеме. Довольно легко привести доводы, показывающие, что теорема 4.5 справедлива и для контактных схем.

Продолжим пример, начатый перед рассмотрением формулировки теоремы. Применяя правило де Моргана и дистрибутивный закон, получим для последнего выражения примера его представление в виде суммы произведений:

$$A^1B^2 + \bar{B}^3C^4\bar{A}^5 + B^3\bar{C}^4D^6 + A^7\bar{E}^9\bar{C}^{11} + A^7E^9\bar{D}^{12} + \\ + D^8E^9\bar{C}^{11} + D^8E^9\bar{D}^{12} + B^{10}\bar{C}^{11} + B^{10}\bar{D}^{12}.$$

Находим, что для перехода с изменением одной входной переменной в качестве p_j может выступать только произ-

ведение $D^8E^9\bar{D}^{12}$. В этом случае E является f_1-буквой, а D и $\bar{D}$ — c-буквами. Будем считать, что $\bar{D}$ есть c_1-буква, а $\bar{D}$ — c_0-буква; тогда $\bar{D}=1$ на наборе I_1. Поскольку $f(I_1)=0$, $B^{10}\bar{D}^{12}$ равно 0 на наборе I_1; поэтому B должна быть f_0-буквой. Аналогично, так как $\bar{A}^7E^9\bar{D}^{12}=0$ на наборе I_1, A должна быть f_1-буквой. После удаления букв, зафиксированных в одном и том же значении на обоих наборах перехода, получим выражение

$$C^4D^6+D^8\bar{C}^{11}+D^8\bar{D}^{12}.$$

Полагая C f_0-буквой, получим

$$D^8+D^8\bar{D}^{12}.$$

Таким образом, динамическое состязание невозможно, так как элементарное произведение D^8 не содержит буквы D, не принадлежащей элементарному произведению $p_j=D^8\bar{D}^{12}$. Следовательно, D^8 (или, вернее, первоначальное элементарное произведение $D^8E^9\bar{C}^{11}$) принимает значение 1, когда $D^8E^9\bar{D}^{12}$ становится равным 1, и больше не изменяется. Однако, если допустить, что C есть f_1-буква, то получим выражение

$$D^6+D^8\bar{D}^{12},$$

удовлетворяющее условиям теоремы 4.5, и при изменении переменной D и фиксации переменных A, B, C, E значениями 1, 0, 1, 1 соответственно в схеме возможно динамическое состязание.

Динамические состязания при изменении только одной переменной X часто могут быть выявлены непосредственно по исходному выражению, в котором еще не проведена индексация. Для этого необходимо определить, существуют ли переменные, отличные от X, которые можно зафиксировать значениями 1 и 0 таким образом, чтобы после выполнения операций сложения и умножения, применения правила де Моргана и преобразований вида $X+X\longrightarrow X$ и $XX\longrightarrow X$ результирующее выражение содержало либо $X+X\bar{X}$, либо $X(X+X)$. В нашем примере, фиксируя переменные A, B, C, E значениями 1, 0, 1, 1

соответственно, получим выражение

$$1 \cdot 0 + 1 \cdot 1 \cdot (0 + D) + ((0 + D) \cdot 1 + 0)\,\overline{(1 \cdot D)},$$

равное $D + D\bar{D}$.

При фиксации переменных A, B, C, E значениями 1, 0, 0, 1 мы получили бы

$$1 \cdot 0 + 1 \cdot 0(0 + D) + ((0 + D) \cdot 1 + 0)\,\overline{(0 \cdot D)} = D.$$

Заметим, что этот способ мы не будем доказывать формально.

Возвращаясь вновь к нашей основной процедуре и к нашему примеру, рассмотрим переход $00000 \longrightarrow 00101$ $(ABCDE)$. Здесь после освобождения от неизменяющихся переменных A, B и D получим

$$C^4 + E^9\bar{C}^{11} + E^9.$$

Элементарное произведение $E^9\bar{C}^{11}$ выступает в роли p_j, но элементарное произведение E^9 не подчиняется ограничению, наложенному на p_h, так как не содержит букв C или E, не входящих в $E^9\bar{C}^{11}$; следовательно, на переходе $00000 \longrightarrow 00101$ динамическое состязание невозможно. Рассмотрим переход $00000 \longrightarrow 11001$. После фиксации значением 0 переменных C и D получим

$$A^1B^2 + \bar{A}^7E^9 + B^{10}.$$

Поскольку $\bar{A}^7$ является c_0-буквой, а E^9 — c_1-буквой, то $\bar{A}^7E^9$ представляет собой p_j. B^{10} и A^1B^2 содержат c_1-буквы, не входящие в $\bar{A}^7E^9$, поэтому на данном переходе динамическое состязание возможно.

Закончим этот раздел двумя примерами. Сначала синтезируем функцию, заданную в таблице 4.1, в виде многоуровневой схемы, свободной от логических состязаний. Напишем выражение в виде суммы произведений, элементарными произведениями которого являются все простые импликанты функции:

$$\bar{A}C + BC\bar{D} + \bar{B}CD + AB\bar{C}D + A\bar{B}\bar{C}\bar{D}.$$

Это выражение соответствует двухуровневой схеме, сво-

бодной от логических состязаний. Вынесем за скобки C и $A\bar{C}$:

$$C(\bar{A}+B\bar{D}+\bar{B}D)+A\bar{C}(BD+\bar{B}\bar{D}).$$

И, наконец, применяя правило де Моргана к первой скобке, получим эквивалентное выражение, в котором $BD+\bar{B}\bar{D}$ присутствует дважды:

$$C(\bar{A}+\overline{(BD+\bar{B}\bar{D})})+A\bar{C}(BD+\bar{B}\bar{D}).$$

Схема, соответствующая последнему выражению, свободна от логических состязаний и изображена на рис. 4.9.

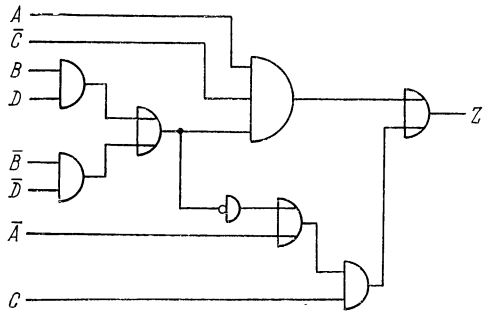


Рис. 4.9. Схема, свободная от логических состязаний.

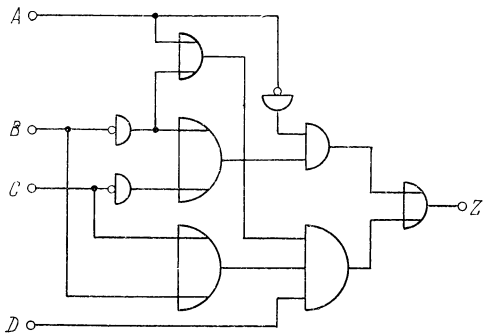


Рис. 4.10. Схема, анализируемая на комбинационные состязания.

Теперь проанализируем схему, приведенную на рис. 4.10, на возможность появления в ней статических

состязаний. Запишем выражение, непосредственно соответствующее схеме:

$$\bar{A}(\bar{B}+\bar{C})+(A+\bar{B})(B+C)D.$$

После раскрытия скобок получим выражение в виде суммы произведений, соответствующее двухуровневой схеме:

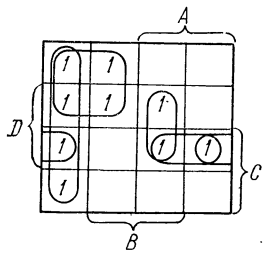
$$\bar{A}\bar{B}+\bar{A}\bar{C}+ABD+ACD+\bar{B}CD+\bar{B}BD.$$

Эта схема содержит те же состязания, что и исходная, так как мы не проводим сокращение $\bar{B}B=0$, поскольку такое преобразование не является сохраняющим состязания.

Из таблицы 4.2, в которой указаны элементарные произведения последнего выражения, за исключением

Таблица 4.2

Элементарные произведения функции, реализуемой схемой рис. 4.10



произведения $\bar{B}BD$, немедленно становится очевидным, что 1-состязание возможно на переходе между 0101 и 1101, так как указанное выражение не содержит простого импликанта $B\bar{C}D$. Возможны также многочисленные функциональные состязания, из которых мы перечислим несколько:

0-состязания: 1001 ↔ 1010, 0111 ↔ 1010.

1-состязания: 1101 ↔ 1011, 0011 ↔ 0100.

В подразделе 4.5.3 мы рассмотрим процедуру, полезную для выявления состязаний в сложных схемах на конкретных переходах.

4.3. Состязания в последовательностных схемах с произвольно расположенными паразитными задержками

В этом разделе *) мы рассмотрим нарушения функционирования последовательностных схем из-за наличия в них паразитных задержек и обратных связей, присущих такого рода схемам. Более подробно вопросы, связанные с наличием обратных связей, будут рассмотрены в главе 5. Мы будем предполагать, что паразитные инерциальные задержки могут присутствовать в любой ветви схемы, даже если эта ветвь состоит только из проводов, соединяющих два логических элемента. Даже если задержки в проводах пренебрежимо малы, сделанное предположение иногда является необходимым в случаях, когда сигнал с ненулевым временем нарастания подается на два входа логического элемента с различными порогами срабатывания. Даже если этот сигнал приходит одновременно на оба входа, достижение порога срабатывания происходит в разные моменты времени, так что создавшаяся ситуация аналогична той, которая складывается, когда паразитные задержки на входах различны (см. работу Робертсона [97]). Относительные величины этих задержек не ограничиваются, предполагается лишь то, что для каждой физической реализации известна верхняя граница значений паразитных задержек. Такое предположение необходимо для того, чтобы между последовательными входными изменениями проходило достаточно времени и схема успевала отрабатывать каждое входное воздействие. В разделе 4.4 мы рассмотрим схемы, относительные величины паразитных задержек которых ограничены в соответствии со сделанным здесь предположением о том, что задержки в логических элементах значительно больше, чем в соединяющих проводах. Там же будет показано, что из этого предположения вытекают совершенно иные результаты.

Поясним теперь, что мы будем понимать под *правильной по Хаффману последовательностной переключательной схемой*. Прежде всего, если верхние границы значений паразитных задержек и величины задержек элемен-

*) До тех пор, пока не оговорено обратное, мы будем иметь дело лишь с ОИВ-функциями.

тов задержки известны, то должна существовать возможность задания такого наименьшего интервала следования последовательных входных изменений, что, если промежутки времени между входными изменениями не ниже этого минимума, то схема работает без отклонений от наперед заданного закона функционирования независимо от того, насколько велики интервалы времени между смежными входными сигналами. Соотношение между временем срабатывания схемы и допустимой скоростью подачи входных воздействий является основным для любой реальной системы, если в ней не предусмотрено управление с помощью сигналов обратной связи скоростью подачи входных данных *).

Второе основное требование касается элементов задержки. Как будет показано ниже, элементы задержки нужны, если в этом есть необходимость, для подавления некоторых эффектов, создаваемых паразитными задержками. Таким образом, обязательно существует связь между величинами паразитных задержек и задержек элементов задержки. Хотелось бы, чтобы эта связь была минимальной, с тем чтобы значения задержек элементов были сведены до минимума, в результате чего снизилась бы стоимость и повысилась надежность схемы. Такая минимальная связь состоит в задании нижней границы (одинаковой для всех элементов схемы) значений задержки элементов задержки как функции верхней границы значений паразитных задержек. Пока величина задержки каждого элемента задержки удовлетворяет указанному условию, а входные изменения разделены достаточными интервалами времени, как указано выше, схема будет функционировать по заданному закону.

Построение правильной в описанном выше смысле схемы можно рассматривать как игру конструктора с чертенком. Чертенок $\mathcal{C}$ делает первый ход, представляя конструктору $\mathcal{D}$ таблицу переходов. Ответом конструктора $\mathcal{D}$ является построение схемы, реализующей таблицу переходов, причем схема может как содержать, так и не содержать элементы задержки. После изучения схемы чертенок делает второй ход, которым он устанавливает

*) Системы с такими управляющими сигналами рассматриваются в главе 6.

верхнюю границу значений паразитных задержек (во второстепенном варианте игры вместо одной общей верхней границы чертенок может задавать границы значений для каждой паразитной задержки). Конструктор $\mathcal{D}$ отвечает вычислением нижней границы задержки элементов задержки. Третий ход чертенка $\mathcal{C}$ состоит в установлении допустимых отклонений значений задержки элементов задержки, что эквивалентно заданию верхней границы значений задержек. В ответ $\mathcal{D}$ задает минимальный интервал времени между последовательными входными изменениями. Заключительный ход в игре делает чертенок $\mathcal{C}$, выбирающий конкретные значения задержек для каждого элемента паразитной задержки в пределах от нуля до верхней границы и для каждого элемента задержки внутри соответствующего диапазона.

Чертенок выбирает также полное начальное состояние схемы и последовательность входных изменений конечной длины, точно указывая, когда должно произойти каждое изменение с учетом ограничений на интервал следования. В этот момент начинает действовать судья $\mathcal{R}$, который соблюдая все выдвинутые противниками условия, конструирует схему, придает задержкам значения, указанные чертенком, устанавливает систему в начальное состояние и подает на нее входную последовательность, несомненно, построенную чертенком $\mathcal{C}$ по дьявольскому плану. Если схема будет работать в соответствии с таблицей переходов, то побеждает конструктор $\mathcal{D}$. В противном же случае... Схема, построенная $\mathcal{D}$ на его первом ходе, является *правильной*, если $\mathcal{D}$ побеждает вне зависимости от стратегии чертенка $\mathcal{C}$. Если неправильность функционирования схемы сказывается лишь в том, что непосредственно после подачи отдельных входных воздействий на некоторых выходах изредка появляются пары ложных изменений, то говорят, что в схеме могут иметь место *переходные состязания*. Если чертенок $\mathcal{C}$ может заставить систему прийти в неправильное устойчивое состояние, то говорят, что имеет место *состязание устойчивых состояний*. Штраф, накладываемый на конструктора $\mathcal{D}$, может при определенных условиях определяться тем, какое из указанных нарушений недопустимо. Схему, в которой невозможно существование состязаний устойчивых состояний, мы будем называть *S-правильной*

схемой независимо от того, имеют ли место переходные состязания.

Игра зачастую происходит при наложенном на действия $\mathcal{C}$ предварительном ограничении, согласно которому каждое входное изменение должно представлять собой изменение лишь одной входной переменной (*функционирование с изменением одной входной переменной*). Если допустимы изменения нескольких входных переменных, то на втором ходе обычно чертенок $\mathcal{C}$ должен также задать интервал t_s такой, что, если несколько входных переменных изменяются за время t_s, их следует рассматривать изменяющимися одновременно. В этом случае $\mathcal{C}$ может включить задание таких почти одновременных входных изменений в качестве своего последнего хода. До подраздела 4.3.4 мы ограничимся игрой с изменением одной входной переменной.

Основная цель книги дать возможность добросовестному читателю стать мастером в любом варианте игры, независимо от выбранной им роли конструктора $\mathcal{D}$ или чертенка $\mathcal{C}$. В этом разделе объясняется, какое значение имеют элементы задержки. Здесь будет показано, что для специального класса последовательностных функций правильные схемы всегда могут быть построены без элементов задержки при условии, что входные изменения допустимы только по одной переменной. При том же ограничении на входные изменения для более широкого класса функций схемы, не содержащие элементов задержки, могут быть построены так, что будут свободными от состязаний устойчивых состояний, хотя в них могут возникать переходные состязания. Любые другие последовательностные функции требуют для своей реализации по меньшей мере одного элемента задержки, и, как будет показано, всегда можно сконструировать реализации, использующие только один элемент задержки (даже в том случае, если допустимы изменения нескольких входных переменных).

Следует понимать, что хотя в общем случае желательно строить правильные схемы, нас иногда будут удовлетворять также схемы, не являющиеся правильными. Иногда может быть разумным, например, увеличение задержки одного элемента задержки по сравнению с другим для того, чтобы зафиксировать некоторое критическое состязание, но зато сократить число элементов в схеме. Или

может случиться, что возникновение отдельных состязаний никогда не приведет к неправильному функционированию схемы из-за того, что значения паразитных задержек не могут быть такими, какими они выбраны чертенком $\mathcal{C}$. Понимание излагаемой далее теории весьма упрощает распознавание указанных ситуаций.

4.3.1. Функции, реализуемые правильными схемами без задержек. Рассмотрим ООТП-матрицу переходов $\mathcal{M}$ для ОИВ-функции, считая, что в отдельные моменты времени допустимы изменения только одной входной переменной. Определим условие, которое позволит нам реализовать заданную матрицу $\mathcal{M}$ с помощью S-правильной схемы без задержек. При этом станет очевидным, какой класс таблиц переходов реализуется такими схемами, а также какова полная процедура синтеза.

Пусть $1 — B$ является устойчивым полным состоянием матрицы $\mathcal{M}$ и пусть E представляет собой входное состояние, отличающееся от B только значением входной переменной x. Перечислим все варианты заполнения столбцов B и E матрицы $\mathcal{M}$, которые могут возникнуть при трехкратном изменении x, считая, что начальное состояние $\mathcal{M}$ равно $1 — B$. Входное воздействие в начале первого перехода равно B, а в конце равно E.

Таблица 4.3

Варианты заполнения матрицы переходов при трехкратном изменении одной входной переменной

	B	E
1	***1***	***1***

а)

	B	E
1	***1***	2
2	***2***	***2***

б)

	B	E
1	***1***	2
2	1	***2***

в)

	B	E
1	***1***	2
2	3	***2***
3	***3***	2

г)

	B	E	
1	***1***	2	
2	3	***2***	
3	***3***	k	$k \neq 2$

д)

Если состояние $1 — E$ устойчиво, как показано в таблице 4.3, *а*, то трехкратное изменение x не требует рассмотрения других состояний, кроме $1 — B$ и $1 — E$. Если

же состояние $1 - E$ неустойчиво и изменение x переводит $1 - E$ в $2 - E$, то в зависимости от состояния, записанного в клетке $2 - B$, можно разобрать три основных случая, Если $N(2, B) = 2$, то получаем вариант, изображенный в таблице 4.3, *б*. Второй случай приведен в таблице 4.3, *в*, где $N(2, B) = 1$. Обе ситуации при дальнейшем изменении x не требуют рассмотрения состояний, отличных от 1 и 2. Третий случай соответствует переходу из $2 - B$ в некоторое новое состояние 3. Этот случай может быть разделен на два, в зависимости от состояния $N(3, E)$. Условию $N(3, E) = 2$ отвечает сегмент матрицы переходов, изображенный в таблице 4.3, *г*. Если $N(3, E) = k$, где k отлично от 1 и 2, то интересующим нас сегментом является сегмент, представленный в таблице 4,3, *д*. Для любой матрицы переходов изучаемого типа или произвольной ОИВ-таблицы переходов сегмент, выделенный относительно любого устойчивого состояния и любого соседнего входного столбца, может быть отнесен к одному из пяти видов, представленных в таблице 4.3, с точностью до переобозначения внутренних состояний.

Предположим теперь, что матрица $\mathcal{M}$ реализуется схемой без задержек, свободной от логических состязаний в комбинационной части. Исследуем для каждого сегмента из таблицы 4.3 последствия однократного изменения x при начальном состоянии $1 - B$.

а) Поскольку в таблице 4.3, *а* изменений внутренних состояний нет, то предполагается, что нет изменений значений внутренних переменных (т. е. Y). Изменяется значение только одного входа x свободной от состязаний логической схемы; следовательно, так как статические состязания невозможны, действительно, никакие Y-переменные не меняются. Любые изменения выходных переменных происходят в схеме так, как определено в таблице переходов, потому что при изменении лишь на одном входе динамические состязания также не возникают. Таким образом, реализация данного перехода не вызывает трудностей.

б) На переходе*) $1 \longrightarrow 2$ схема может пройти через любое y-состояние из $T(1, 2)$. Как было показано ранее

*) Основные используемые здесь обозначения и понятия введены в подразделе 3.3.2.

при обсуждении ОТП-кодирования, если существуют какие-либо переходы $p \longrightarrow q$ в столбце E, когда $q \neq 2$, необходимо, чтобы при выбранном варианте кодирования $T(1, 2)$ и $T(p, q)$ различались. В таком случае, поскольку элементы задержки отсутствуют, возможно, что в некоторых частях схемы изменение x, вызывающее смену внутреннего состояния, из-за наличия паразитных задержек не будет воспринято до того, как проявятся изменения одной или более y-переменных. Поэтому некоторые внутренние переменные Y или выход Z могут принять значения, соответствующие временному переходу системы в столбце B в некоторое y-состояние, принадлежащее $T(1, 2)$. Отсюда следует, что последовательность состояний может иметь два варианта. Во-первых, если любая выходная переменная Z_t постоянна в $1 - B$ и $2 - E$, а в $2 - B$ имеет другое значение, то на данном переходе возникает неустранимое переходное состязание, поскольку узел Z_t может воспринять следующую последовательность полных состояний: $1 - B \longrightarrow 2 - B \longrightarrow 2 - E$. Во-вторых, если для любого y-состояния из $T(1, 2)$ следующее состояние в столбце B является y-состоянием, не принадлежащим $T(1, 2)$, то некоторые внутренние переменные Y могут изменить свои значения не так, как требуется, и конечное устойчивое состояние не будет равно $2 - E$.

в) Как и в предыдущем случае, наблюдаемое в различных точках состояние системы может принадлежать B или E, причем внутренним состоянием является любой элемент множества $T(1, 2)$. Однако, так как для ОИВ-функции $Z(2, B)$ можно выбрать равным $Z(1, B)$, а $Z(1, E)$ равным $Z(2, E)$, переходное состязание не должно возникнуть. Кроме того, поскольку переход $2 \longrightarrow 1$ в столбце B должен соответствовать переходу $1 \longrightarrow 2$ в столбце E, ограничения, наложенные в подразделе 3.3.2 при кодировании состояний, выполняются, препятствуя появлению в любом Y-узле ошибочного сигнала, как это описано для случая (б).

г) Это более сложный случай. Поскольку переменные, по которым y^2 отличается от y^3 (y^2 и y^3 — y-состояния, приписанные соответственно строкам 2 и 3), могут иметь значения, соответствующие состоянию $2 - B$ системы, то, как показано в предыдущих двух случаях, эти значения могут измениться. Избежать этого нет возможности —

единственным способом является использование элементов задержки, которые сейчас исключены из рассмотрения. Следовательно, после изменения x внутренним состоянием может стать (или проявиться в некоторых точках схемы) любое состояние из $T(1, 2, 3)$. Таким образом, приводя аргументы, аналогичные высказанным в случае (б), заключаем, что следующие состояния в столбце B матрицы $\mathcal{M}$ для любого состояния из $T(1, 2, 3)$ должны соответствовать состояниям из $T(1, 2, 3)$, чтобы избежать появления дальнейших ошибочных изменений y-переменных. Если, кроме того, следующее состояние в столбце E для любого состояния становится состоянием 2, то, как только изменение x распространится до Y-узлов схемы, все y-переменные приобретут значения, соответствующие y^2, и система перейдет в правильное устойчивое состояние. Однако, как и в случае (б), возможно переходное состязание, если только каждый сигнал Z не имеет того же значения в состоянии $3 - B$ (и $2 - B$), что и в состоянии $1 - B$ или $2 - E$.

д) Как и в случае (г), система может находиться в любом состоянии из $T(1, 2, 3)$ при входах B или E. В частности, некоторые или все переменные, различающие y^3 и y^k, могут принять значения, соответствующие состоянию $3 - E$. Если бы эти переменные изменились, то система могла бы перейти в устойчивое состояние $k - E$, не являющееся заданным на данном переходе. Нет способов предотвратить эту возможность при построении ООТП-кодирования (с учетом того, что не используются элементы задержки), и, действительно, как будет показано в следующем подразделе, если таблица переходов содержит сегмент, указанный в таблице 4.3, ∂, нет способа реализовать эту таблицу без применения элементов задержки так, чтобы избежать появления состязаний устойчивых состояний. Наличие в таблице переходов такого сегмента приводит к возникновению существенного состязания, принципиально присущего последовательностной функции (или таблице переходов): для некоторого начального полного состояния и входной переменной x три последовательных изменения x переводят систему в состояние, отличное (и неэквивалентное) от состояния, достигаемого после однократного изменения x. В таблице 4.3, ∂ начальное полное состояние есть $1 - B$, состоя-

ние, достигаемое после однократного изменения x, есть $2 - E$, а состояние, достижимое после трехкратного изменения x, есть $k - E$.

Конфигурацию, изображенную в таблице 4.3, *г*, будем называть *г*-тройкой *), а переход $1 - B \longrightarrow 2 - E$ в этой таблице — *г-переходом*. *г*-тройка в таблице состоит из трех строк 1, 2 и 3, а *г*-переход начинается из B и заканчивается в E. Формально такая конфигурация определяется следующим образом: существуют начальное состояние и входная переменная x такие, что после трехкратного изменения x внутреннее состояние совпадает с внутренним состоянием, достигаемым после однократного изменения x, а внутреннее состояние, достигнутое после двукратного изменения x, отличается от начального состояния и от состояния, достигаемого после однократного изменения x.

Наши предыдущие исследования показали, что, если в схеме не используются элементы задержки, нельзя рассматривать взаимодействия между переходами только в одном столбце, как это делалось в разделе 3.3.2. Тот факт, что входное изменение, вызывающее изменение внутреннего состояния, может быть воспринято схемой после окончания изменений y-переменных, означает, что следует рассматривать пересечения T-множеств, соответствующих переходам в соседних столбцах. Таким образом, должны выполняться условия более общие, нежели условия теоремы 3.2, особенно в тех случаях, когда рассматриваются *г*-тройки. Приведем теперь процедуру синтеза, определяющую необходимые обобщенные условия Трэйси, а затем докажем, что построенная с помощью этой процедуры схема свободна от состязаний устойчивых состояний.

Пусть запись $i, j, \ldots \neq p, q, \ldots$ обозначает, что каждый левый элемент отличен от любого правого. Переходы типа *б*, *в* и *г* относятся к соответствующим сегментам (б), (в) и (г) таблицы 4.3. Таблица 4.4 используется для иллюстрации процедуры.

Процедура 4.1. *Построение ООТП-кодирования для S-правильной, без элементов задержки и без существенных состязаний реализации таблицы переходов.*

*) Эта конфигурация соответствует состязанию, называемому *несущественным*.

1. Для каждого перехода в таблице переходов построить дихотомии так, как указано ниже, удаляя каждую дихотомию, покрываемую любой другой, построенной ранее.

1.1. Если переход $i - B \longrightarrow j - E$ является $г$-переходом, причем k — третья строка $г$-тройки, то построить следующие дихотомии:

1.1.1. (ijk, pqr), если $p, q, r \neq i, j, k$ и pqr является $г$-тройкой, начинающейся или кончающейся в B или E. В нашем примере, полагая $B = 00$, $E = 01$, $ijk = 123$, $pqr = 567$ ($г$-тройка, начинающаяся в 11 и кончающаяся в 01), получаем дихотомию (123, 567).

Таблица 4.4

Таблица переходов, свободная от существенных состязаний, но содержащая четыре $г$-тройки

	x_1x_2 00	01	11	10
1	***1***,0	2,0	7,0	***1***,0
2	3,0	***2***,0	***2***,0	3,0
3	***3***,0	2,0	4,0	***3***,0
4	5,0	***4***,0	***4***,0	3,0
5	***5***,0	6,1	***5***,0	***5***,0
6	5,0	***6***,1	7,0	***6***,0
7	1,0	6,1	***7***,0	***7***,0

1.1.2. (ijk, pq), если $p, q \neq i, j, k$ и переход $p \longrightarrow q$ является $б$- или $в$-переходом (но не $г$-переходом) в B или E. Например, в таблице 4.4, если $B = 11$, $E = 10$, $ijk = 234$, то переход $6 \longrightarrow 7$ в 11 приводит к дихотомии (234, 67).

1.1.3. (ijk, p), если $p \neq i, j, k$, состояние $p - E$ устойчиво и никакие переходы в E не заканчиваются состоянием $p - E$, т. е. не существует $q \neq p$ таких, что $N(q, E) = p$. Например, $г$-тройка 234, начинающаяся в 11 и кончающаяся в 10, и устойчивое состояние 1 — 10 приводят к дихотомии (1, 234).

1.2. Если переход $i - B \longrightarrow j - E$ является $б$- или $в$-переходом (но не $г$-переходом) в B или E, то добавить следующие дихотомии:

1.2.1. (ij, pq), если $p, q \neq i, j$ и переход $p \longrightarrow q$ является $б$- или $в$-переходом (но не $г$-переходом) в B или E. Например, $7—10 \rightarrow 1—00$ и $4—11 \rightarrow 3—10$ приводят к дихотомии (17, 34).

1.2.2. (ij, p), если $p \neq i, j$, состояние $p - E$ устойчиво и никакие переходы в E не заканчиваются состоянием $p - E$.

2. Найти, выполняя шаги 3 — 6 процедуры 3.1, минимальное множество, покрывающее дихотомии, полученные на предыдущем шаге.

3. Сформировать матрицу переходов, сопоставляя внутреннюю переменную каждому элементу множества, полученного на шаге 2. На каждом переходе $i - B \longrightarrow j - E$, если он является $г$-переходом, для каждого y-состояния из множеств $T(i, j)$ или $T(i, j, k)$ следующее состояние и выход в столбце E должны совпадать со следующим состоянием и выходом для $j - E$. Следующие состояния и выходы в столбце B для состояний из этих множеств могут соответствовать любому состоянию из $T(i, j)$ или $T(i, j, k)$.

Докажем справедливость описанной процедуры.

Теорема 4.6. *Для любой заданной ОИВ-функции, свободной от существенных состязаний, процедура 4.1 приводит к построению матрицы переходов, которая может быть реализована с помощью S-правильной схемы без задержек в предположении, что допустимы изменения лишь одной входной переменной. В некоторых случаях могут возникать переходные состязания.*

Доказательство. Непосредственно перед описанием процедуры 4.1 было показано, что для $а$-переходов (табл. 4.3, $а$), при которых внутреннее состояние не меняется, возможно произвольное кодирование строк. Для $б$-, $в$- и $г$-переходов показано, что справедливо ООТП-кодирование при условии, что отдельным множествам y-состояний, определенных как функции тех y-состояний, которые приписаны строкам, участвующим в переходах, приписываются значения соответствующих следующих состояний для входных столбцов, в которых начинаются и заканчиваются переходы.

Чтобы доказать справедливость нашей процедуры, покажем, что, если покрываются все дихотомии, полученные на шаге 1, то множества y-переменных, которые должны иметь разные клетки в некотором определенном столбце, не пересекаются и, следовательно, процесс заполнения клеток матрицы переходов непротиворечив.

1. Рассмотрим сначала $г$-переход $i - B \longrightarrow j - E$, в предположении, что k есть третье состояние $г$-тройки. Ранее показано, что y-состояниям из множества $T(i, j, k)$ должны быть приписаны следующие состояния, соответствующие состоянию j в столбце E и любому состоянию из множества $T(i, j, k)$ в столбце B. Дихотомии, построен-

ные на шаге 1.1.1 *), гарантируют, что, если существуют любые другие z-тройки, охватывающие столбец B или E и некоторое непересекающееся множество строк, то множество $T(i, j, k)$ не пересекается с соответствующими этим z-тройкам y-множествами.

Любой другой z-переход, z-тройка которого включает **) одну или более строк ijk и который оканчивается в E, должен иметь в качестве конечного состояния $j - E$ и, таким образом, j должно быть записано в качестве следующего состояния в столбце E y-множества, т. е. противоречий в этом столбце не возникает.

Если некоторый другой z-переход оканчивается в столбце B и частично совпадает с ijk, то его конечным состоянием является $i - B$ или $k - B$; следовательно, следующим состоянием в его столбце B y-множества должно быть, соответственно, состояние i или k. Каждое из этих состояний не противоречит записанным в столбце B состояниям множества $T(i, j, k)$, соответствующего ijk-тройке, т. е. и в этом случае нет противоречий.

Предположим, что некоторая новая z-тройка pqr начинается в столбце B и частично совпадает с ijk. Тогда $T(i, j, k)$ имеет непустое пересечение с $T(p, q, r)$ и для y-состояний, принадлежащих пересечению, ограничения, накладываемые обеими z-тройками, удовлетворяются, если следующие состояния выбраны соответствующими любому состоянию из пересечения.

И, наконец, допустим, что некоторая z-тройка pqr начинается в столбце E и частично совпадает с ijk. Тогда состояние $j - E$ должно принадлежать этой z-тройке, а, значит, либо p, либо r равно j. Таким образом, j удовлетворяет ограничениям, накладываемым pqr на состояния из $T(p, q, r)$ в столбце E, которые частично совпадают с $T(i, j, k)$, так как j принадлежит $T(p, q, r)$.

Таким образом, процедура 4.1 обеспечивает отсутствие пар z-троек, накладывающих противоречивые требования на следующие состояния в матрице переходов.

Теперь рассмотрим противоречия между ijk и $б$- или $в$-переходами. Выполнение шага 1.1.2 обеспечивает пустое

*) Здесь и далее в доказательстве теоремы мы будем иметь в виду шаги процедуры 4.1.

**) В дальнейшем будем говорить, что при этом z-переход *частично совпадает* с ijk.

пересечение множества $T(i, j, k)$ и y-множества, соответствующего любому *б*- или *в*-переходу, не совпадающему с ijk. Для *б*- или *в*-переходов, оканчивающихся в B или E, с помощью точно таких же доводов, как в соответствующих случаях частичного совпадения *г*-троек, можно установить, что процедура 4.1 непротиворечива по отношению к требованиям, налагаемым *г*-переходами и *б*- или *в*-переходами.

Не произойдет ничего вредного, если во время *г*-перехода некоторый узел Y примет значение, соответствующее устойчивому состоянию системы из столбца B, так как никакая y-переменная, сохраняющая устойчивое значение на протяжении перехода, не может измениться. Шаг 1.1.3 гарантирует, что множество $T(i, j, k)$ не содержит y-состояния, приписанного изолированному устойчивому состоянию из столбца E. Поэтому требования, наложенные произвольным *г*-переходом, могут быть всегда выполнены без нарушения любых других требований.

2. Чтобы показать, что при выполнении шага 1.2 процедуры 4.1 требования, налагаемые *б*- и *в*-переходами, выполняются без противоречий, можно привести сходные, хотя и более простые доводы.

Таким образом, теорема доказана.

Вопрос о том, когда могут появиться неустранимые переходные состязания, был обсужден непосредственно перед формулировкой процедуры 4.1.

Следует отметить, что, в отличие от довольно похожей теоремы 3.2, в теореме 4.6 доказано достаточное, но не необходимое условие. В некоторых случаях отдельные дихотомии, полученные на первом шаге процедуры 4.1, могут быть заменены другими дихотомиями, и при этом не возникнет состязаний устойчивых состояний (см. задачу 4.12). Однако систематически использовать указанное свойство для получения каких-либо выгод, по-видимому, затруднительно.

Покажем, как, применяя процедуру 4.1, построить матрицу переходов, соответствующую таблице 4.4. На шаге 1.1.1 *г*-тройки 123, 456, 234 и 567 приводят к дихотомиям (123, 456), (123, 567) и (234, 567). На шаге 1.1.2 добавляются дихотомии (17, 456) и (17, 234). На шагах 1.1.3 и 1.2.1 не появляется новых дихотомий, не покрываемых уже полученными. Например, (123, 4), получаю-

щаяся на шаге 1.1.3, покрывается (123, 456), а (17, 23), получающаяся на шаге 1.2.1, покрывается (17, 234). Завершающий эту часть процедуры шаг 1.2.2 приводит к построению дихотомий (2, 34) и (5, 67). По окончании первого шага удобно разделить все переходы и соответствующие им столбцы по категориям.

На втором шаге процедуры 4.1 построим карту пар (см. табл. 4.5), с помощью которой определим минимальное множество совместимых дихотомий $\{ad, bc, e, fg\}$ *), задающее в свою очередь множество покрывающих дихотомий: (1237, 456), (1234, 567), (17, 234) и (25, 3467). Чтобы избежать ненужных усложнений, возникающих при неполном определении некоторых y-столбцов, для целей минимизации логической схемы в дальнейшем мы заменим последние две дихотомии на (17, 23456) и (125, 3467) и получим, тем самым, кодирование, приведенное в таблице 4.6.

Проверяя по очереди каждый переход в таблице 4.6, заполним строки 8, 9, ..., 16 так, как предписывается третьим шагом процедуры 4.1. Например, состояние 8—00 может появиться на $г$-переходе $1 — 00 \longrightarrow 2 — 01$ или на $б$-переходе $7 — 10 \longrightarrow 1 — 00$. В первом случае для правильного функционирования необходимо, чтобы следующее состояние для $8 — 00$ соответствовало строке, y-состояние которой принадлежит 00— —. Во втором случае это состояние должно быть состоянием 1. Так как состояние 1 удовлетворяет обоим условиям, оно и используется (значение выхода, равное 0, также удовлетворяет условиям на обоих переходах). Все остальные подобные случаи состояний, участвующих в нескольких переходах, также имеют удовлетворяющие нас решения, что и гарантируется теоремой 4.6. Такими состояниями являются полные состояния 11—11, 16—11, 8—10 и 9—10. На переходе $4—01 \longrightarrow 5—00$ возможно переходное состязание, но оно является неотъемлемым свойством таблицы переходов и не может быть устранено без применения элементов задержки. Другие матрицы переходов можно построить, выбирая способы кодирования строк, основанные на других множествах дихотомий, покрывающих таблицу 4.5.

*) Существуют и другие минимальные множества.

Таблица 4.5

Карта пар, построенная на шаге 2 процедуры 4.1

a (123, 456)	b (123, 567)	c (234, 567)	d (17, 456)	$\bar{d}$ (456, 17)	e (234, 17)	f (5, 67)	$\bar{f}$ (67, 5)	g (2, 34)
	×							
×		×	×	×	×	×	×	×
	×	×		×	×	×	×	×
×	×		×					
×	×	×	×	×				
×	×	×	×	×	×			
×	×	×		×	×			

Таблица 4.6

Матрица переходов для таблицы 4.4

	x_1x_2 00	01	11	10	y_1	y_2	y_3	y_4
1	***1***,0	2,0	7,0	***1***,0	0	0	0	0
2	3,0	***2***,0	***2***,0	3,0	0	0	1	0
3	***3***,0	2,0	4,0	***3***,0	0	0	1	1
4	5,0	***4***,0	***4***,0	3,0	1	0	1	1
5	***5***,0	6,1	***5***,0	***5***,0	1	1	1	0
6	5,0	***6***,1	7,0	***6***,0	1	1	1	1
7	1,0	6,1	***7***,0	***7***,0	0	1	0	1
8	1,0	2,0	7,0	1/7/8/9,0	0	0	0	1
9	1,0	6,1	7,0	1/7/8/9,0	0	1	0	0
10	—	6,1	5/6/7/9/10/ 11/15/16,0	—	0	1	1	0
11	—	6,1	7,0	6/7/11/16,0	0	1	1	1
12	—	—	—	—	1	0	1	0
13	—	—	—	—	1	0	0	1
14	5,0	4/5/6/14,0	2/3/4/14,0	3,0	1	0	0	0
15	—	6,1	5/6/7/9/10 11/15/16,0	—	1	1	0	0
16	—	6,1	7,0	6/7/11/16,0	1	1	0	1

4.3.2. Функции, требующие элементов задержки для правильной реализации. В этом подразделе доказывается, что таблицы переходов с существенными состязаниями не могут быть S-правильно реализованы без применения элементов задержки. Доказательство ведется от противного: для заданной схемы без элементов задержки, реализующей функцию с существенными состязаниями, показывается, как приписать значения паразитным задержкам, чтобы система достигла состояния $k - E$ вместо состояния $2 - E$ при переходе типа, показанного в таблице 4.3, *д*.

Доказательство не зависит от того, какой вид кодирования строк используется, даже если некоторые устойчивые состояния таблицы переходов представляются циклическими множествами y-состояний. Следующая лемма имеет отношение к этому вопросу, но читатель может не обращаться к ней до тех пор, пока не прояснится ее место в полном доказательстве.

Лемма 4.12. *Предположим, что в некоторой S-правильной реализации без задержек таблицы переходов множе-*

ство S_i y-состояний приписано строке i и что во входном столбце I подмножество C_{ij} состояний является замкнутым (не имеющим выхода из цикла) циклическим подмножеством множества S_i. Тогда в каждом столбце I те состояния из множества $T(C_{ij})$, для которых значения y-переменных постоянны в C_{ij}, имеют те же сигналы возбуждения, что и в C_{ij}, а значениями выхода должны быть значения $Z(i, I)$.

Доказательство. Если $\mathbf{y}^p$ является устойчивым состоянием из $T(C_{ij})$, то оно может быть воспринято в некотором узле Y или узле Z как текущее y-состояние в любой момент цикла C_{ij}, так как не существует элементов задержки, устанавливающих порядок изменения y-состояний. Следовательно, значение Z в состоянии $\mathbf{y}^p$ должно быть равным $Z(i, I)$ для того, чтобы избежать возможности бесконечного повторения ошибочных выходных сигналов, а любое Y_j, имеющее постоянное значение в каждом состоянии из C_{ij}, должно иметь то же значение и в состоянии $\mathbf{y}^p - I$, иначе y_i может измениться, вызывая выход системы из цикла, что противоречит условию.

Следующая лемма значительно упрощает задачу анализа поведения схем без задержек. В лемме показано, что всегда можно выбрать такое множество y-переменных, что с каждым Y_i будет связана отдельная логическая схема. Таким образом, только за счет соответствующего выбора y-переменных заданную схему всегда можно представить в виде схемы, изображенной на рис. 4.11 и называемой *схемой с раздельным возбуждением*. Выходы схемы на рисунке не показаны, поскольку рассматриваемое преобразование их не затрагивает.

Лемма 4.13. *Внутренние переменные y произвольной S-правильной последовательностной схемы без задержек могут быть выбраны таким образом, что схему можно представить в виде схемы с раздельным возбуждением.*

Доказательство. Переменные y могут быть выбраны следующим образом:

1. Выбрать множество узлов, достаточное, чтобы разорвать каждый замкнутый путь в схеме, и приписать этим узлам начальное множество y-переменных. Теперь сигнал в любой точке схемы может быть задан как комбинационная функция входных переменных и выбранных y-переменных.

2. Если в схеме существуют два y-узла y_i и y_j и узел n, не являющийся входом или y-узлом и такой, что существуют пути прохождения сигнала от n к y_i и y_j, не проходящие через другие y-узлы, то отметить узел n как новый y-узел.

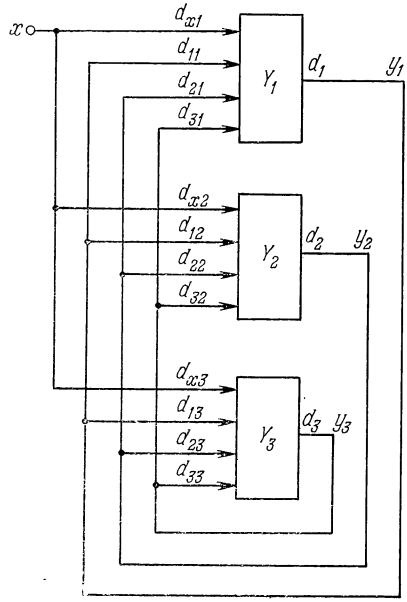


Рис. 4.11. Схема с раздельным возбуждением.

3. Повторять шаг 2 до тех пор, пока это возможно. Так как схема является конечной и процедура не добавляет новых узлов, последняя завершится, когда не останется элементов, выходы которых не помечены y-переменными и ведут более чем к одному y-узлу. Матрица переходов может быть построена с использованием переменных, выбранных на первом шаге или после любого числа повторений второго шага. Соответствующие сжатые таблицы переходов должны быть эквивалентны, так как все они описывают поведение одной и той же схемы, и это поведение определяется единственным образом с точ-

ностью, быть может, до переходных неустойчивых состояний, поскольку схема является S-правильной.

Теперь перейдем к основному результату.

Теорема 4.7. *Никакая таблица переходов, несвободная от существенных состязаний, не может быть реализована с помощью S-правильной схемы без задержек.*

Доказательство. Допустим, что такая схема существует и что мы представили ее с помощью леммы 4.13 в виде схемы с раздельным возбуждением. Предположим, что таблица 4.7, в которой столбцы B и E отличаются только значениями переменной x, представляет собой часть таблицы переходов, несвободной от существенных состязаний. Тогда, если входное состояние и начальное y-состояние (y^{1i}) соответствуют полному состоянию $1 - B$ и изменяется x, то по крайней мере одна y-переменная должна изменить свое значение. Если должны измениться одновременно несколько y-переменных, то будем считать, что первой изменится y_1. Затем может стать неустойчивым некоторое другое множество y-переменных — мы отметим одну из y-переменных, а именно y_2, и будем считать, что y_2 меняется второй. Продолжая этот процесс, получим последовательность y-изменений *) как реакцию на изменение x, причем конечное состояние будет соответствовать одному из состояний, приписанных строке 2. Может существовать много таких последовательностей, кончающихся различными состояниями, приписанными строке 2, причем вид последовательности, получаемой в каждом отдельном случае, зависит от распределения паразитных задержек. Мы выбираем просто одну из них, отмечая y-переменные так, чтобы, если $i < j$, то первое изменение y_i в последовательности предшествовало первому изменению y_j. Основное требование к структуре логической схемы заключается в том, что значение выхода подсхемы Y_1 (см. рис. 4.11) должно измениться, если под-

Таблица 4.7

Фрагмент таблицы переходов, содержащий существенное состязание

	B	E
1	***1***	2
2	3	***2***
3	***3***	4
4	***4***	***4***

*) Возможно, что изменения некоторых y-переменных появятся в этой последовательности не один раз.

схема Y_1 воспринимает изменение x; затем, если подсхема Y_2 воспринимает изменения x и y_1, то ее выход изменяется и т. д. В общем случае, если начальным отрезком последовательности является $xy_{i_1}y_{i_2}\ldots y_{i_{k-1}}y_{i_k}$ и подсхема Y_{i_k} воспринимает все изменения переменных x, y_{i_1}, y_{i_2}, ..., $x_{i_{k-1}}$ из этого отрезка, то выход Y_{i_k} будет менять свое значение. Такую последовательность, описывающую переход $1-B \longrightarrow 2-E$, назовем C_{12}-*последовательностью*, и для любой схемы, реализующей рассматриваемую таблицу переходов, должна существовать по крайней мере одна C_{12}-последовательность.

Аналогичным образом должны существовать C_{23}-последовательность, начинающаяся с изменения x и переводящая систему из $2-B$ в $3-E$, и C_{34}-последовательность, также начинающаяся с изменения x и переводящая систему из $3-B$ в $4-E$.

Предположим, что при начальном состоянии $\mathbf{y}^{1i}-B$ мы получили C_{12}-последовательность, заканчивающуюся в некотором y-состоянии $\mathbf{y}^{2j}$, C_{23}-последовательность, начинающуюся в $\mathbf{y}^{2j}$ и заканчивающуюся в $\mathbf{y}^{3k}$, C_{34}-последовательность, начинающуюся в $\mathbf{y}^{3k}$ и заканчивающуюся в $\mathbf{y}^{4m}$, причем состояние $\mathbf{y}^{4m}-E$ — либо устойчивое состояние, либо состояние цикла, не выходящего из столбца E. Мы всегда можем выбрать $\mathbf{y}^{4m}$ указанным образом, потому что состояние $4-E$ устойчиво. Тогда последовательность C, объединяющая эти три последовательности, т. е. $C = C_{12}C_{23}C_{34}$, переводит систему из $\mathbf{y}^{1i}-B$ в $\mathbf{y}^{4m}-E$.

Допустим, что y-переменными, входящими в некоторую C-последовательность, являются переменные $y_1, y_2, \ldots, y_n$. Если переменная y_i подвергается «чистому» изменению в C-последовательности, т. е. изменяется нечетное число раз, то будем обозначать этот факт предикатной записью $\mathrm{Ch}(y_i)$, в противном случае будем писать $\overline{\mathrm{Ch}}(y_i)$. Очевидно, что, если система находилась в начальном состоянии $\mathbf{y}^{1i}-B$, x и каждая переменная y_i, для которой $\mathrm{Ch}(y_i)$, изменились только один раз, а другие y-переменные не менялись, то конечным состоянием системы будет $\mathbf{y}^{4m}-E$. Теперь покажем, как должны быть распределены паразитные задержки, чтобы вызвать следующие события, происходящие в системе, если при $t=0$ изменяется x, а система находится в начальном состоянии $\mathbf{y}^{11}-B$:

а) при $t = i$ $(i = 1, 2, \ldots, n)$ изменяется y_i;

б) при $t = n + i$ $(i = 1, 2, \ldots, n)$ изменяются те и только те y_i, для которых Ch (y_i).

Если произойдет такая последовательность событий, то конечным состоянием системы будет $\mathrm{y}^{4m} - E$, соответствующее состоянию $4 - E$, которое не является состоянием, определенным таблицей переходов. Следовательно, мы должны показать, что неверное поведение системы возможно из-за действия паразитных задержек.

Определим сначала другой предикат Prec(a, b), обозначающий, что в рассматриваемой C-последовательности нечетное число изменений a, т. е. «чистое» изменение a, предшествует первому изменению b. Пусть значение паразитной задержки между входом x и входом подсхемы Y_j равно d_{xj}, а значение паразитной задержки между узлом y_i и входом подсхемы Y_j равно d_{ij} (см. рис 4.11). Предположим, что значения всех паразитных задержек пренебрежимо малы, кроме определенных ниже:

1. Если $\overline{\text{Prec}}(x, y_i)$, то $d_{xi} = i$;
2. Если $\overline{\text{Prec}}(x, y_i)$, то $d_{xi} = n + i$;
3. Если $\overline{\text{Prec}}(y_j, y_i)$, то $d_{ji} = i - j$;
4. Если $\overline{\text{Prec}}(y_j, y_i)$ и $\overline{\text{Ch}}\ (y_j)$, то $d_{ji} = n + i - j$;
5. Если $\overline{\text{Prec}}(y_j, y_i)$ и $\overline{\text{Ch}}\ (y_j)$, то $d_{ji} = 2n$.

Тогда, если система находилась первоначально в состоянии $\mathrm{y}^{1i} - B$ и при $t = 0$ изменилась переменная x, то это изменение воспринимается при $t = 1$ подсхемой Y_1 (в силу п. 1), после чего изменяется y_1. Допустим теперь, что при $t = i$ $(i = 1, 2, \ldots, j - 1)$ изменяется y_i. Тогда при $t = j$ подсхема Y_j воспримет согласно пп. 1 и 2 изменение x, если Prec(x, y_i), а согласно пп. 3, 4 и 5 — изменения именно тех y_j, для которых Prec(y_i, y_j). Таким образом, подсхема Y_j воспринимает состояние системы, предшествующее первому изменению y_j, а, следовательно, меняет значение y_j. Последний вывод завершает индуктивное доказательство того, что y_i изменяется при $t = i$ $(1 \leqslant i \leqslant n)$.

Выясним, что произойдет в схеме при $t = n + 1$. Если подсхема Y_1 еще не восприняла изменение x, то она воспримет его при $t = n + 1$, согласно п. 2. Более того, если для каждого j y_j совершает «чистое» изменение на C-последовательности, то в силу п. 4 изменение y_j воспри-

нимается подсхемой Y_1 в этот же момент. Поэтому подсхема Y_1 воспринимает систему в состоянии $\mathbf{y}^{4m} - E$. Если это состояние устойчиво, то y_1 остается неизменной при $\mathrm{Ch}\,(y_1)$ и меняется второй раз при $\overline{\mathrm{Ch}}\,(y_1)$. Согласно лемме 4.12 возможны такие же варианты, если состояние $\mathbf{y}^{4m}$ принадлежит замкнутому циклическому множеству и y_1 представляет собой одну из неизменяющихся переменных. Если y_1 является одной из изменяющихся переменных такого множества, то при $t = n + 1$ она может как изменяться, так и оставаться постоянной.

Допустим теперь, что при $t = n + i$ $(i = 1, 2, \ldots, j-1)$ подсхема Y_i воспринимает систему в состоянии $\mathbf{y}^{4m} - E$ или в некотором другом состоянии, принадлежащем циклу, который содержит состояние $\mathbf{y}^{4m} - E$. Кроме того, допустим, что затем y_i изменяется так, как описано выше для случая изменения y_1. Тогда при $t = n + j$ подсхема Y_j воспримет изменение x, если она не восприняла его при $t = j$, а также вторые изменения, если они происходят, тех y-переменных, первое изменение которых было воспринято подсхемой Y_j при $t = j$ (согласно п. 3). Кроме того, согласно п. 4 Y_j воспринимает первые изменения тех y-переменных, которые претерпевают «чистые» изменения на C-последовательности, но изменения которых подсхема Y_j не восприняла при $t = j$. Согласно п. 5 Y_j не воспринимает в момент $t = n + j$ никакие изменения тех переменных y_i, для которых $\mathrm{Ch}\,(y_i)$ и $\overline{\mathrm{Prec}}\,(y_i, y_j)$. Если паразитные задержки инерциальны, то Y_j никогда не воспримет никакие изменения указанных переменных y_i. Следовательно, при $t = n + j$ в Y_j воспринимается каждая переменная состояния $\mathbf{y}^{4m} - E$, если $\mathbf{y}^{4m} - E$ не является циклическим состоянием; в случае, когда состояние $\mathbf{y}^{4m} - E$ циклическое, Y_j может воспринять некоторые циклически меняющие свои значения переменные других состояний. Тем не менее в последнем случае подсхема Y_j воспринимает систему в некотором состоянии из циклического множества, содержащего $\mathbf{y}^{4m} - E$, а поэтому в обоих случаях поведение переменной y_j полностью соответствует принятой гипотезе. Таким образом, мы установили, что при определенном распределении паразитных задержек последовательности изменения y-состояний происходят в системе так, как мы и предполагали, причем при $t = 2n - 1$ система переходит в состояние, соответ-

ствующее состоянию $4 - E$, в котором все существенные y-переменные устойчивы. Этим мы заканчиваем доказательство теоремы.

Заметим, что в случае, когда отдельные паразитные задержки являются задержками совершенного типа, то согласно п. 5, начиная с момента $t = 2n$, в системе могут происходить некоторые дальнейшие изменения. Хотя интуитивное предположение, что совершенные паразитные задержки вызывают, вероятно, больше затруднений, чем инерциальные задержки, доказательство, приведенное здесь, требует, чтобы паразитные задержки имели инерциальные компоненты для обеспечения уверенности, что система остановится в неправильном состоянии. Вероятно, можно построить доказательство так, чтобы не использовать этого допущения.

Следует понимать, что нарушение правильного функционирования может быть вызвано не только таким распределением паразитных задержек, какое определено в доказательстве теоремы. К тому же небольшие отклонения от заданных значений задержек не влияют на конечный результат, так что вероятность неблагоприятного распределения задержек, имеющего место в схеме, отлична от нуля. Следует, однако, ясно представлять себе, что в отдельных случаях можно выяснить, что распределение задержек не является неблагоприятным, и поэтому схема, не принадлежащая классу S-правильных, тем не менее работает правильно.

Несмотря на то что доказательство проведено для бесконтактных логических схем, теорема 4.7 справедлива и для релейно-контактных схем, в которых дребезг контактов и действие паразитных реактивностей играют роль паразитных задержек.

4.3.3. Схемы, содержащие элементы задержки. Мы уже видели, что последовательностные функции, которым свойственны существенные состязания, могут быть правильно реализованы только с помощью элементов задержки. В некоторых случаях элементы задержки можно вводить в схему даже тогда, когда функция свободна от существенных состязаний, для использования более эффективного варианта кодирования строк таблицы, для уменьшения числа требуемых внутренних переменных или по другим причинам. Хотя мы продолжаем рассматривать

изменения лишь одной входной переменной и ОИВ-функции до тех пор, пока не будет оговорено противное, в этом подразделе мы отметим несколько моментов, в которых использование элементов задержки допускает ослабление некоторых из указанных ограничений. В подразделе 4.3.4 будет отдельно проведен анализ схем для случая изменения нескольких входных переменных.

Допустим сначала, что элементы задержки инерциального типа включаются в каждую обратную связь и в каждый выход. Если d_M — максимальный интервал времени, необходимый для получения реакции комбинационной части схемы на любое изменение любой входной переменной, т. е. максимальная сумма паразитных задержек некоторого пути в схеме, и d_m — минимальный такой интервал, то можно определить нижнюю границу величины задержки элемента задержки в виде $D = d_M - d_m$. Кроме того, допустим, что используется некоторое свободное от критических состязаний пригодное кодирование строк (см. главу 3).

Предположим, что происходит входное изменение. Из-за возможных в схеме комбинационных состязаний оно может вызвать появление на некоторых Z-выходах совокупности кратковременных импульсов, после чего распространятся сигналы, обусловливающие устойчивое изменение состояния некоторых выходов. Инерциальные задержки, включенные на Z-выходах, отфильтруют эти импульсы. Аналогичным образом, отдельные Y-сигналы могут ошибочно измениться несколько раз, и некоторые из внутренних переменных примут значения, соответствующие новому устойчивому состоянию. Элементы инерциальной задержки, содержащиеся в обратных связях, допустят распространение изменения устойчивого состояния на y-входы только после того, как на каждом из Y-полюсов проявилось входное изменение. Если имеют место изменения более чем одной y-переменной, то в общем случае эти изменения происходят в различные моменты времени; однако, поскольку критические состязания отсутствуют, порядок изменений не влияет на множество вновь установившихся значений Z- и Y-переменных. Как и раньше, кратковременные импульсы фильтруются элементами инерциальной задержки, и процесс продолжается с новым множеством y-сигналов, возникших на входах

логического блока, только после того, как предыдущие значения распространятся до всех Y- и Z-полюсов. В конце концов в системе устанавливается устойчивое состояние, и спустя D единиц времени после окончания последнего y-изменения на входе может начаться новое изменение. Ни комбинационные, ни существенные состязания не могут вызвать нарушений функционирования.

Теперь удалим задержки на Z-выходах и заменим задержки в обратных связях элементами задержки совершенного типа с теми же величинами задержек. Допустим кроме того, что логические схемы свободны от логических состязаний. Тогда начальное входное изменение вызовет только определенные изменения на некоторых Y- и Z-полюсах. В случае возникновения некритических состязаний значения Z-выходов и функций возбуждения каждой y-переменной останутся неизменными внутри подкуба, натянутого на начальное и конечное y-состояния, так что кратковременных ошибок не появится. Как и в предыдущем случае, последовательность y-состояний может быть пройдена без ошибок.

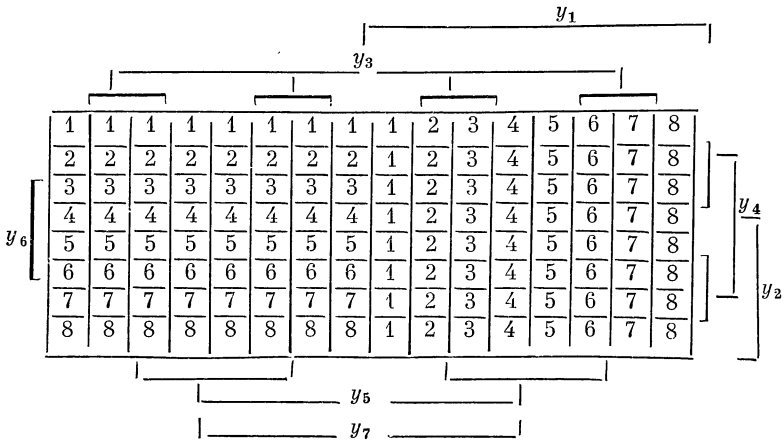
Как МИВ-, так и НИВ-функции могут быть реализованы при допущениях, принятых в предыдущем подразделе, поскольку значения Z-выходов могут быть приписаны так, чтобы их изменения не зависели от конкретного пути в подкубах переходов. ООТП-кодирование (см. подраздел 3.3.5) может быть использовано, если желательно иметь функционирование, зависящее от времени. В общем случае необходимым условием реализации МИВ- или НИВ-функций является использование логических схем, свободных от состязаний, потому что включить на выходах инерциальные задержки нельзя, так как эти элементы будут сглаживать не только ошибочные, но и правильные выходные импульсы.

Предположим, что желательно реализовать заданную функцию с минимальным числом элементов задержки по причине высокой стоимости их производства. Здесь мы опишем несколько методов, приводящих нас к этой цели, включая методы, при которых для реализации необходим лишь один элемент задержки.

Сначала рассмотрим вариант кодирования состояний, приведенный в таблице 4.8 и являющийся универсальным кодированием со сцепленными связанными строчными

Таблица 4.8

$(2S_0+1)$-кодирование



1	1	1	1	1	1	1	1	1	2	3	4	5	6	7	8
2	2	2	2	2	2	2	2	1	2	3	4	5	6	7	8
3	3	3	3	3	3	3	3	1	2	3	4	5	6	7	8
4	4	4	4	4	4	4	4	1	2	3	4	5	6	7	8
5	5	5	5	5	5	5	5	1	2	3	4	5	6	7	8
6	6	6	6	6	6	6	6	1	2	3	4	5	6	7	8
7	7	7	7	7	7	7	7	1	2	3	4	5	6	7	8
8	8	8	8	8	8	8	8	1	2	3	4	5	6	7	8

множествами (подробное обсуждение см. в разделе 3.1). Приведенное здесь кодирование является $(2S_0+1)$-кодированием, при котором пространство значений y-переменных разделено на два сектора, различающихся значением переменной y_1. Все возможные переходы происходят в два этапа: некритические состязания в строке (или в столбце), принадлежащей начальному множеству строк, возникают после изменения y_1, изменяющего y-состояние в столбце (или в строке) из другого сектора, соответствующего следующему состоянию.

Допустим теперь, что единственный элемент задержки помещен только в цепи возбуждения y_1 и что происходящее входное изменение должно перевести систему из строки i в строку j. Тогда, вообще говоря, в строчном множестве R_i, приписанном строке i, возникнет указанное выше некритическое состязание. Независимо от порядка следования вызванных y-изменений и входного изменения, не могут измениться никакие y-переменные, кроме тех, которые обозначают y-состояния из начальной строки (столбца) множества R_i (предполагается, что логических состязаний нет). Тогда Y_1 может измениться только

после того, как все меняющиеся y-переменные приобретут предполагаемые значения, соответствующие R_i-состоянию s_{ij}, соседнему с R_j. Следующее событие произойдет спустя время D после изменения y_1. При допущении, что D выбрано подходящим образом, это изменение будет иметь место только после распространения до всех Y- и Z-полюсов всех предыдущих входных изменений и y-изменений. Таким образом, система обязательно достигнет состояния j, и при этом не возникнет переходных состязаний.

Если состояние j неустойчиво в новом входном столбце, а k является следующим состоянием, то значения функций возбуждения для всех R_j-состояний из этого столбца, за исключением R_j-состояния s_{jk}, соседнего с R_k в этом секторе, будут соответствовать состоянию s_{jk}. В состоянии s_{jk} неустойчиво только y_1, так что за задержанным изменением y_1 еще раз последует некритическое состязание.

Итак, поскольку единственный элемент задержки участвует во всех переходах по внутренним состояниям, не возникает никаких вопросов о том, каков порядок изменения соответствующих переменных, даже тогда, когда происходит последовательность переходов. Это означает, что $(2S_0 + 1)$-кодирование можно использовать для построения свободной от состязаний логической схемы с одним элементом задержки, реализующей как ОИВ-, так и МИВ- и НИВ-функции.

Другим подходом к синтезу схем, содержащих один элемент задержки, является построение блока задержки — схемы, условно представленной на рис. 4.12 и имеющей n входов и n выходов. Основное свойство блока задержки состоит в том, что y_i $(i = 1, 2, \ldots, n)$ повторяет значения Y_i через элемент задержки. Мы представим две реализации блока задержки, требующие применения одного элемента задержки, независимо от значения n. Различие этих реализаций заключается только в ограничениях, при которых они должны использоваться.

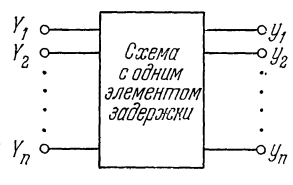


Рис. 4.12. Блок задержки.

Первая схема изображена на рис. 4.13 для $n = 3$ (обобщение станет очевидным из дальнейшего) и содержит в качестве компонент мажоритарные элементы и сумматоры по модулю 2 *). Мажоритарный элемент M_i и сумматор A_i соответствуют каждой переменной y_i. Отметим, что A_i получает входные сигналы с каждого выхода y_j $(j \neq i)$ и с выхода элемента задержки, включенного

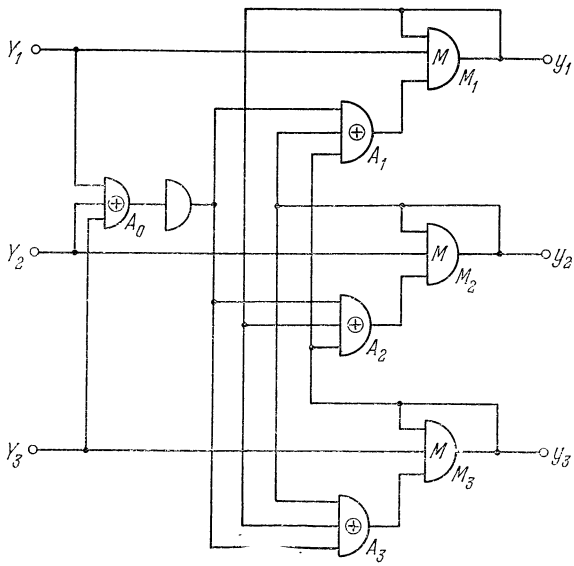


Рис. 4.13. Реализация блока задержки.

последовательно с сумматором A_0. На элементе A_0 суммируются все входные сигналы Y_i. Предположим теперь, что $Y_i = y_i$ для всех i и что Y-входы остаются постоянными в течение долгого времени. Тогда выходной сигнал A_0 равен сумме по модулю 2 всех y-выходов. Следовательно, выходная функция каждого A_i имеет вид

$$\sum_{j=1}^{n} y_j \oplus \sum_{j \neq i} y_j = \sum_{j \neq i} (y_j \oplus y_j) \oplus y_i = 0 + y_i = y_i.$$

Тогда значения всех трех входов M_i есть $y_i = Y_i$, и си-

*) Очевидно, что можно использовать и другие типы элементов.

стема находится в состоянии равновесия. Если Y_j изменится, то действие этого изменения коснется только A_0 и M_i-элементов, непосредственно соединенных с полюсом Y_j. Через некоторое время, определяемое паразитными задержками, изменится выход A_0, однако его изменение распространится дальше в течение интервала времени D, равного величине задержки элемента задержки. Элемент M_j вообще не отреагирует на изменение Y_j, так как два других его входных сигнала y_j и A_j останутся постоянными. Таким образом, все y-сигналы сохранят свои значения на интервале, не меньшем D. И, наконец, когда изменение A_0 спустя время D поступит на входы сумматоров, каждый сумматор A_i обработает это изменение по прошествии интервала времени, равного величине другой паразитной задержки. Изменение значения выхода A_i вызовет в свою очередь изменение одного из входных сигналов каждого M_i. Но поскольку такое изменение является изменением на втором входе M_j, то выход M_j изменится и также изменится значение y_j, приходя в соответствие с Y_j. Ни один из выходов остальных M_i не изменится, так как для каждого из них входы Y_i и y_i все еще будут сохранять первоначальные значения Когда изменение y_j распространится до входов A_i (для всех $i \neq j$), выходы этих элементов установятся в исходные значения, и система снова придет в состояние равновесия, причем $Y_i = y_i$ для всех i. Поэтому единственным результатом изменения Y_j явится изменение y_j с задержкой, равной сумме D и значений некоторых паразитных задержек. Следовательно, поведение блока задержки совпадает с поведением системы, содержащей элемент задержки между парами Y_j и y_j.

Рассмотрим условия, при которых блок задержки функционирует правильно. Допустим сначала, что элементы свободны от комбинационных состязаний. Тогда легко видеть, что в случае изменения четного числа Y-входов выходной сигнал A_0 останется постоянным и не вызовет изменений никаких y-выходов. Если изменятся три и более Y-входов, то схема может вести себя правильно, но появится возможность состязаний вследствие того, что некоторые изменения y-выходов будут состязаться с изменениями выходов элементов A_i, соединенных с другими y-выходами. Достигнуть надежного функциони-

рования можно только за счет ограничения, состоящего в допустимости изменения только одного Y-входа в каждый момент времени.

Если максимальная паразитная задержка между y_i и выходом любого A_i $(i \neq j)$ равна d, то необходимо выждать d единиц времени после распространения изменения Y_i до выхода y_i прежде, чем разрешить новое входное изменение.

Если даже элемент задержки является элементом совершенного типа, то блок задержки моделирует до некоторой степени инерциальные задержки, поскольку в случае, когда Y_i изменяется четное число раз за интервал, меньший D, значение y_i остается прежним. Причина этого явления заключается в том, что при $Y_i = y_i$ в момент изменения значения выхода A_i выход M_i сохранит свое значение из-за наличия на двух его входах первоначальных сигналов.

Описанный блок задержки может быть использован в совокупности с любым вариантом кодирования строк соседними кодовыми комбинациями и свободной от состояний логической схемой для правильной реализации ОИВ-функций произвольного вида. Все изменения значений y-переменных задерживаются блоком задержки, а применение указанного способа кодирования гарантирует, что в каждый момент времени изменится лишь одна входная переменная блока задержки и что входные изменения не будут происходить слишком часто. Использование свободных от состязаний логических схем ликвидирует возможность нового изменения y_i вслед за ошибочным импульсом Y_i, который может возникнуть прежде, чем изменение y_i проявится на элементе A_i.

Вторая реализация блока задержки с одним элементом задержки изображена на рис. 4.14, *а*. Кроме сумматоров по модулю 2, элемента ИЛИ и элемента задержки, блок задержки содержит блок переключения. Этот блок, показанный слева на рис. 4.14, *б*, действует следующим образом: когда управляющий сигнал C равен 1, выход E подключается к входу B (нижняя пунктирная линия), а когда $C = 0$, выход E принимает значение входа A. Таким образом, блок переключений действует как переключатель с одним перекидным контактом, управляемый сигналом C. Свободная от состязаний комбинационная

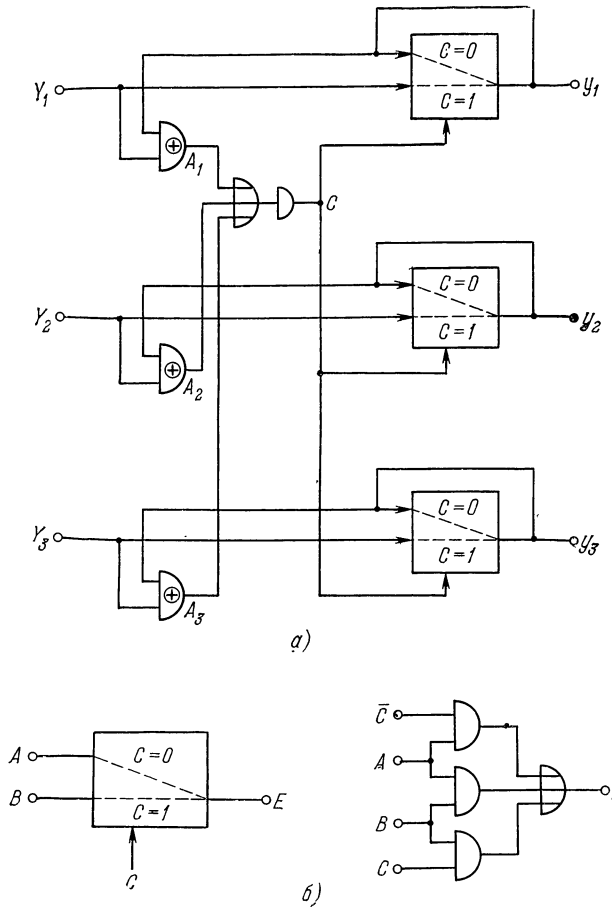


Рис. 4.14. *а*) Другой вид реализации блока задержки. *б*) Реализация переключателя, свободная от состязаний.

схема, реализующая блок переключений, изображена в правой части рис. 4.14, *б*.

Рассмотрим теперь действие блока задержки. Прежде всего заметим: если $Y_i = y_i$ для каждого i, то значение выхода каждого элемента A_i равно 0 и в случае сохранения такой ситуации C также примет нулевое значение. Тогда все блоки переключений будут находиться в состоянии блокировки, соответствующем соединению A и E с замкнутой петлей обратной связи для каждого y_i; поэтому выходные сигналы блока задержки будут оставаться неизменными, а система примет устойчивое состояние. После изменения некоторого входа Y_j выход элемента A_j примет единичное значение, в результате чего элемент ИЛИ также установится в 1. Отметим, что элемент ИЛИ устанавливается в 1 только тогда, когда $Y_i \neq y_i$ для некоторого i. Никаких дальнейших изменений не произойдет, и, в частности, все y-выходы останутся в первоначальных значениях до тех пор, пока единичный сигнал не распространится через элемент задержки и не изменит значение C. После изменения C все блоки переключения примут состояние, соответствующее соединению y_i и Y_i. Спустя немного времени после того, как y_j примет значение, согласующееся с Y_j, выход элемента A_j изменится и станет равным 0, что вызовет установку в 0 элемента ИЛИ. После распространения нулевого сигнала через элемент задержки сигнал C вновь примет значение 0, блоки переключения снова примут состояние блокировки до появления нового входного изменения.

Важной особенностью этой схемы является допустимость одновременного изменения нескольких входных переменных. Если на протяжении интервала D изменяется несколько Y-переменных, то все соответствующие y-выходы изменятся спустя D единиц времени после первого Y-изменения. Таким образом, блок задержки сдвигает во времени любое число одновременных входных изменений. Кроме того, если входные изменения имеют некоторый разброс по времени, выходные изменения тем не менее произойдут одновременно, т. е. схема обладает свойством синхронизации, которое может быть использовано в отдельных случаях.

Однако после окончания изменения выходного сигнала для последующих входных изменений система пред-

ставляет собой только паразитную задержку до тех пор, пока сигнал C не установится в 0, а это происходит спустя промежуток времени, несколько больший, чем D, и зависящий от величин паразитных задержек. Поэтому входные изменения должны следовать друг за другом с интервалом $2D + d$, в то время как этот интервал для первой реализации блока задержки равен $D + d$. Как и в предыдущем случае, задержки, представленные блоком, также обладают некоторыми инерциальными свойствами.

Блок задержки, изображенный на рис. 4.14, как и блок задержки, схема которого приведена на рис. 4.13, может быть использован для реализации ОИВ-функций при рассмотренных ранее ограничениях, за тем исключением, что кодирование состояний требует применения ОТП-кодирования, но вовсе необязательно, чтобы этот способ являлся кодированием соседними кодовыми комбинациями. Ослабление ограничений связано с допустимостью одновременного изменения нескольких Y-входов. Однако в этом случае возникает требование увеличения интервала следования x-изменений из-за увеличения времени восстановления.

И, наконец, отметим, что двухвходовый вариант любого блока задержки в совокупности с $2S_0$-кодированием состояний (см. таблицу 3.2 в главе 3) может быть использован для реализации ОИВ-функций произвольного вида. Блок задержки следует применять для задержки только двух квадрантных переменных y_1 и y_2. Эта схема аналогична ранее использовавшейся для тех же целей в совокупности с $(2S_0 + 1)$-кодированием (табл. 4.8). Она работоспособна, поскольку при использовании $2S_0$-кодирования каждый переход по внутренним состояниям вызывает всегда изменение только одной из двух квадрантных переменных и никогда не вызывает изменения обеих. Использовать схему с трехвходовыми блоками задержки вместе с $(2S_0 - 1)$-кодированием для реализации ОИВ-функций нельзя, так как в этом случае во время одного перехода по состояниям могут измениться несколько октантных переменных, в результате чего блоки задержки будут функционировать неправильно (после изменения первой октантной переменной блоки задержки не смогут задержать последующие изменения, так как невозможно

обеспечить достаточного времени для восстановления состояния блоков).

Теперь мы представим метод построения правильных схем, не содержащих элементов задержки в обратных связях. Единственный элемент задержки связывается только с каждым входным сигналом. Рассмотрим, например, функцию, заданную в таблице 4.9, *а* и содержащую

Таблица 4.9

	x_1x_2 00	01	11	10
1	***1***,0	2,0	4,0	***1***,0
2	3,1	***2***,0	3,0	***2***,0
3	***3***,1	4,0	***3***,0	2,0
4	***4***,0	***4***,0	***4***,0	1,0

а) Таблица переходов, содержащая существенные состязания

	$x_1x_2x_1Dx_2D$ 0000	0100	0101	0001	1101	1111
1	***1***,0	***1***,0	2,0	***1***,0	***1***,0	4,0
2	3,1	***2***,0	***2***,0	***2***,1	***2***,0	3,0
3	***3***,1	***3***,0	4,0	***3***,1	***3***,0	***3***,0
4	***4***,0	***4***,0	***4***,0	***4***,0	***4***,0	***4***,0

б) Фрагмент расширения таблицы а) с задержанными входами

несколько существенных состязаний. Предположим, что для каждого сигнала x_i мы ввели дополнительный сигнал x_{iD}, полученный сдвигом x_i во времени с помощью элемента задержки. Расширенная таблица переходов может быть построена, если кроме x_i независимыми входными переменными считать также x_{iD}. Все элементы каждого столбца, где $x_j \neq x_{jD}$ для некоторого j, должны быть сделаны устойчивыми. Те столбцы, в которых $x_i = x_{iD}$ для всех i, должны в точности повторять столбцы исходной таблицы, содержащие соответствующие значения x_i. Таблица 4.9, *б* содержит шесть из шестнадцати столбцов расширенной таблицы 4.9, *а*. Заметим, что столбцы 0000, 0101 и 1111 таблицы 4.9, *а* соответствуют столбцам 00, 01 и 11 таблицы 4.9, *а*.

Допустим, что начальным состоянием таблицы 4.9, *а* является 1—00. Тогда, если входные сигналы остаются неизменными в течение D единиц времени, соответствующим состоянием таблицы 4.9, *б* будет 1—0000. Если x_2 изменится из 0 в 1, то в таблице 4.9, *б* сначала произойдет переход 1—0000 → 1—0100, причем состояние 1—0100 сохранится до того момента, когда x_{2D} примет значение, равное 1, спустя D единиц времени. За это время на всех Y- и Z-полюсах установятся значения, соответству-

ющие переходу системы в столбец 0100, поэтому система будет вести себя так, будто произошли два отдельных входных изменения: 0000 → 0100, а затем 0100 → 0101. Каждое из них включает изменение только одной входной переменной, существенных состязаний или g-троек не возникает на обоих переходах, поскольку каждая строка промежуточного столбца 0100, вовлеченного в оба перехода, содержит устойчивое состояние. Таким образом, при использовании любого пригодного варианта ОТП-кодирования (см., например, раздел 3.3) и схемы, свободной от логических состязаний, обеспечивается правильная реализация таблицы переходов. Такая же картина наблюдается при всех переходах, на которых изменяется лишь одна переменная x_i. Заметим, что каждый промежуточный столбец единственным образом связан с некоторой упорядоченной парой соседних входных состояний. Выходные состояния в клетках промежуточных столбцов выбираются равными выходным состояниям в следующих устойчивых состояниях, достигаемых из соответствующих столбцов-предшественников. Такой выбор устраняет собственные переходные состязания (см. подраздел 4.3.1).

Синтез схем с задержками на входах может быть выполнен без построения довольно громоздких расширенных таблиц. Пусть

$$C = \sum_{i=1}^{m} (x_i \oplus x_{iD}),$$

где $\sum$ обозначает булево суммирование. Тогда $C = 1$ только для тех входных состояний, которые соответствуют промежуточным столбцам; кроме того, пусть $\hat{Y}_i$ есть функции возбуждения в терминах y-переменных и незадержанных x-переменных, воспроизводимые непосредственно по исходной таблице переходов (при задании некоторого пригодного ОТП-кодирования). Тогда, поскольку каждое состояние y_i устойчиво в каждом промежуточном столбце, а в остальных столбцах совпадает с соответствующим состоянием исходной таблицы, для каждого Y_i имеем:

$$Y_i = Cy_i + \overline{C}\hat{Y}_i.$$

Это выражение задает соответствующую функцию, которую следует реализовать с помощью свободной от состя-

заний схемы, причем ее внешними входами должны быть x-, x_D- и y-переменные. Z-функции легко вычисляются по исходной таблице, так как они не зависят от задержанных входных сигналов. Иногда можно получить некоторую экономию, если не требовать, чтобы все элементы промежуточных столбцов были устойчивыми.

В некоторых случаях после детального анализа матрицы переходов можно придавать элементы задержки одной или нескольким y-переменным. Мы продемонстрируем такую возможность на примере таблицы 4.7 — сегмента таблицы переходов, содержащего существенное состязание. Допустим, что кодирование строк построено так, что на переходе $1-E \longrightarrow 2-E$ изменяется только y_{12}, а на переходе $2-B \longrightarrow 3-B$ изменяется только y_{23}. Ранее мы показали, что на переходе $1-B \longrightarrow 1-E \longrightarrow 2-E$ существенное состязание проявляется, если существует возможность того, что изменение y_{12} распространится до узла Y_{23} раньше, чем входное изменение, т. е. возможно ошибочное изменение y_{23}. Есть два способа разрешения этой ситуации с помощью элементов задержки.

1. Ввести элемент задержки инерциального или совершенного типа в ветвь y_{12}, чтобы изменение y_{12} не происходило раньше, чем входное изменение распространится по всей схеме.

2. Ввести инерциальную задержку в ветвь y_{23}, чтобы, если даже Y_{23} изменится на короткое время, значение y_{23} осталось постоянным.

В некоторых случаях может случиться, что кодирование состояний таково или выбрано таким, что достаточно узкое подмножество y-переменных участвует во всех переходах, на которых возможны состязания. Это обстоятельство допускает синтез правильной схемы, экономичной с точки зрения числа внутренних переменных, количества логических элементов схемы и числа необходимых элементов задержки. Важный пример такой ситуации представлен в таблице 4.10, представляющей матрицу переходов счетчика с четырьмя состояниями и закодированной с помощью отраженного кода Грея (см. подраздел 3.3.1). Заметим, что каждый переход содержит существенное состязание.

Выбранный вариант кодирования строк является кодированием соседними кодовыми комбинациями, перемен-

ная y_3 изменяется на половине всех переходов и является переменной, которая может ошибочно изменяться на остальных переходах. Поэтому единственный элемент задержки инерциального типа, включенный в обратную связь y_3, обеспечивает правильность реализации. Сказанное справедливо и для счетчика с 2^n состояниями для любого n, реализующего матрицу, аналогичную приведенной в таблице 4.10 и закодированную с помощью отраженного кода Грея.

Таблица 4.10

Таблица переходов счетчика и кодирование отраженным кодом Грея

	x				
	0	1	y_1	y_2	y_3
1	***1***,00	2,00	0	0	0
2	3,01	***2***,00	0	0	1
3	***3***,01	4,01	0	1	1
4	5,10	***4***,01	0	1	0
5	***5***,10	6,10	1	1	0
6	7,11	***6***,10	1	1	1
7	***7***,11	8,11	1	0	1
8	1,00	***8***,11	1	0	0

Множество представленных здесь реализаций, содержащих минимальное число элементов задержки, иллюстрирует разнообразие возможностей. Например, при использовании $(2S_0 + 1)$-кодирования достаточно одного элемента задержки, а число внутренних переменных, даже в наихудших случаях, не превосходит двоичного логарифма от числа строк. Хотя заметим, что переходы по внутренним состояниям могут потребовать столько времени, сколько необходимо, чтобы сигнал распространился, во-первых, через элемент задержки и, во-вторых, через логическую схему. При использовании блока задержки требуется время лишь на прохождение сигнала через логическую схему. Характеристики других рассмотренных методов для использования в конкретных ситуациях предоставляется выяснить читателю. Весьма вероятно, что другие методы и их разновидности могут быть еще более полезными.

4.3.4. Реализации, допускающие изменения нескольких входных переменных. Предположим, в системе, реализующей функцию из таблицы 4.11, *а*, при начальном состоянии 1—00 одновременно изменяются обе входные переменные x_1 и x_2. В этом случае система должна перейти в устойчивое состояние 1—11. Однако, если изменения x_1 и x_2 происходят не точно в один и тот же момент или если из-за наличия паразитных задержек эти изменения проявляются в некоторых характеристи-

Таблица 4.11

	x_1x_2 00	01	11	10	y_1	y_2
1	***1***,0	2,1	***1***,0	4,1	0	0
2	3,1	***2***,1	***2***,0	***2***,0	0	1
3	***3***,1	***3***,0	4,0	***3***,0	1	1
4	1,0	3,0	***4***,0	***4***,1	1	0

а) ОИВ-таблица переходов

	x_1x_2 00	01	11	10
1	***1***,0	2*B*,0	***1***,0	4*B*,0
2*A*	3*B*,0	2*B*,0	***2A***,0	***2A***,0
2*B*	3*B*,1	***2B***,***1***	2*A*,1	2*A*,***1***
3*A*	3*B*,0	***3A***,0	4*A*,0	***3A***,0
3*B*	***3B***,1	3*A*,1	4*A*,1	3*A*,1
4*A*	1,0	3*A*,0	***4A***,0	4*B*,0
4*B*	1,1	3*A*,1	4*A*,1	***4B***,1

б) Таблица Мура, эквивалентная таблице а)

ческих узлах системы так, что кажется, будто они происходят неодновременно, то система может вести себя так, будто ее состоянием, непосредственно следующим за 1—00, является 1—01 или 1—10. Поскольку эти состояния неустойчивы, на выходах Y_1 или Y_2 соответственно должны появиться ошибочные сигналы. Если ошибочные сигналы сохраняются в течение времени, большего, чем задержка соответствующей петли обратной связи $Y - y$, то система никогда не достигнет состояния 1—11 и перейдет либо в 2—11, либо в 4—11.

Если элемент совершенной задержки величины D включен в обратную связь и интервал d_e сохранения ошибочного сигнала меньше D, то система придет в 1—11, а затем перейдет в 2—11 (или 4—11), когда ошибочный сигнал распространится через задержку. Тогда в системе будут происходить колебания между состояниями 1—11 и 2—11 (или 4—11) до тех пор, пока не изменится входное состояние.

Описанная ситуация не может быть исправлена за счет выбора иного варианта кодирования строк или модификации схемы, реализующей таблицу. Она свойственна данной таблице переходов. Решение проблемы заключается в использовании в обратных связях элементов инерциальной задержки. В таком случае, пока $d_e < D$, ошибочный сигнал будет отфильтрован инерциальными задержками и никогда не вызовет непредусмотренного изменения y-состояния.

Для использования пригоден любой из элементов инерциальной задержки, описанных в разделе 4.1. Кроме того, при выборе ОТП-кодирования можно использовать блок задержки, изображенный на рис. 4.14, *а*, обладающий требуемым свойством инерциальности. Если используется кодирование соседними кодовыми комбинациями, можно применять схему, приведенную на рис. 4.13, и, наконец, $(2S_0+1)$-кодирование строк (табл. 4.8) используется в совокупности с элементом инерциальной задержки в обратной связи y_1. Таким образом, S-правильные реализации существуют для ОИВ-функций, требующих использования только одного элемента задержки (при использовании блока задержки допустим элемент совершенного типа) даже для случая изменения нескольких входных переменных. Заметим, что величины задержек этих элементов зависят как от разнообразия значений паразитных задержек в путях от x-входов до Y-выходов, так и от максимально допустимого значения интервала времени между изменениями входных переменных, внутри которого изменения считаются происходящими одновременно.

Допуская, что таблица 4.11, *а* реализуется указанным способом кодирования строк и что в обеих обратных связях используются элементы инерциальной задержки, видим, что остается возможность появления переходных состязаний. На протяжении интервала существования ошибочного сигнала, который может иметь место при переходе $1—00 \longrightarrow 1—11$, сигнал Z может принять значение 1, являющееся выходным значением в промежуточных состояниях 1—01 и 1—10.

Естественным средством борьбы с переходными состязаниями является введение инерциальной задержки в каждую выходную линию, аналогично тому, как показано в разделе 4.2 для борьбы с комбинационными состязаниями. Недостатки этого способа указаны ранее. Другое решение, свободное от таких недостатков, не требующее в общем случае дополнительного количества логических элементов, заключается в преобразовании таблицы переходов в эквивалентную таблицу Мура, в которой выходные сигналы постоянны для каждого внутреннего состояния. Если такое преобразование выполнено, то после любого изменения нескольких входных

переменных выходной сигнал не меняется и не происходит никаких изменений на выходе, пока не изменится внутреннее состояние.

Таблица 4.11, *б* представляет собой таблицу Мура, эквивалентную таблице 4.11, *а*. Преобразование выполнено по следующему правилу. Если i-я строка содержит k различных выходных значений, связанных с ее устойчивыми состояниями, то она расщепляется на k строк iA, iB, ..., каждая из которых содержит устойчивые состояния i-й строки с одним из k выходных состояний. В клетках каждой строки все выходные состояния одинаковы, и в клетках всех строк, полученных из i-й, одинаковы следующие состояния. Для полного понимания процесса преобразования достаточно сопоставить таблицы 4.11, *а* и *б*.

Поскольку в таблице Мура не всегда выполняется соотношение $Z(N(s, I), I) = Z(s, I)$, процедура построения ООТП-кодирования должна быть слегка изменена. В частности, если в некотором столбце I имеют место переходы $i \longrightarrow j$ и $k \longrightarrow j$, для которых $Z(i, I) = Z(j, I) \neq Z(k, I)$, то должна быть покрыта дихотомия (ij, k)*). В таблице 4.11, *б*, например, дихотомия $(2B3B, 3A)$ должна быть покрыта из-за наличия в столбце 00 переходов $2B \longrightarrow 3B$ и $3A \longrightarrow 3B$.

С помощью указанного выше метода любая ОИВ-функция может быть правильно реализована, даже если допустимы изменения нескольких входных переменных. Другой интересный подход заключается в использовании «блока источника», схемы, которая предварительно обрабатывает входные сигналы, реализуемые некоторым модифицированным вариантом заданной таблицы переходов.

Общая идея этого подхода состоит в следующем. Добавляем к заданной таблице один устойчивый дополнительный столбец (см. таблицу 4.12, представляющую собой модифицированный вариант таблицы 4.11, *а*) и заново кодируем входы с помощью позиционного метода кодирования, приписывая нулевое состояние дополнительному столбцу. X-входы подключены к блоку источника,

*) С такой ситуацией мы сталкивались при построении ООТП-кодирования для МИВ-таблиц. См. подраздел 3.3.5.

который в установившемся состоянии воспроизводит единичный сигнал только на одном своем выходе в соответствии с кодовой комбинацией, приписанной столбцу таблицы переходов. В нашем примере, если x-входом является 11, на выходе блока источника появляется 0010. Полученный сигнал выступает в роли входного воздействия для схемы, реализующей модифицированную таблицу переходов; поэтому в установившемся состоянии действие такой схемы ничем не отличается от функционирования схемы, реализующей исходную таблицу переходов, с непосредственно подключенными к ней входными линиями x. Однако сразу после изменений x-входов (независимо от того, сколько входных переменных x_i изменяется) одиночный единичный сигнал на выходе блока источника изменяет свое значение на нулевое, и все выходные сигналы сохраняют нулевые значения в течение времени D, соответствующего величине задержки блока. По окончании этого «промежуточного» интервала на выходе блока источника появляется сигнал, соответствующий новому входному состоянию. На протяжении промежуточного интервала внутреннее состояние последовательностной схемы не меняется так же, как и ее выход *).

Таблица 4.12

Вариант таблицы 4.11, *а*

	$v_1v_2v_3v_4$ 0000	1000	0100	0010	0001
1	***1***,*S*	***1***,0	2,1	***1***,0	4,1
2	***2***,*S*	3,1	***2***,1	***2***,0	***2***,0
3	***3***,*S*	***3***,1	***3***,0	4,0	***3***,0
4	***4***,*S*	1,0	3,0	***4***,0	***4***,1

Таким образом, переходу 1—11 $\longrightarrow$ 1—10 $\longrightarrow$ 4—10 в таблице 4.11, *а* соответствует переход 1—0010 $\longrightarrow$ $\longrightarrow$ 1—0000 $\longrightarrow$ 4—0001 в таблице 4.12, причем система находится в промежуточном состоянии 1—0000 примерно D единиц времени. Рассмотренная ситуация совпадает с описанной в подразделе 4.3.3 с точки зрения использования входных задержек (см. табл. 4.9, *б*). В настоящем случае имеется единственный дополнительный столбец,

*) Символ S в устойчивом столбце таблицы 4.11 обозначает, что состояние выхода в каждом состоянии этого столбца остается равным выходному состоянию, реализованному непосредственно перед последним входным изменением.

который используется для связи переходов между любыми двумя столбцами, в то время как ранее существовал отдельный дополнительный столбец для каждой упорядоченной пары входных состояний. Обычный способ ОТП-кодирования, основанный на методах раздела 3.3, приводит к правильному функционированию, даже если не используется ни одного элемента задержки, так как каждый переход преобразуется в последовательность двух

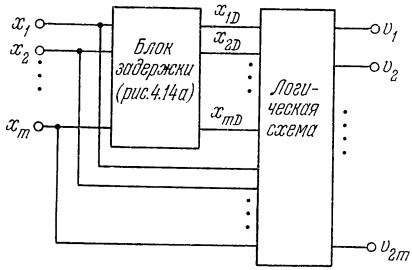


Рис. 4.15. Реализация блока источника.

переходов, один из которых начинается, а другой оканчивается в устойчивом столбце, так что существенных состязаний и ε-переходов не возникает. Выходные сигналы системы снимаются с триггеров, установленных в 0 или 1 только в состояниях, которые не находятся в дополнительном столбце. Триггеры могут быть исключены из схемы, если таблица переходов является таблицей Мура или преобразована в нее. Заметим, что метод входных задержек, описанный выше, может быть применен непосредственно (без использования блока источника) в сочетании с выходными триггерами или таблицей в форме Мура для правильной реализации функций при условии допустимости изменения нескольких входных переменных.

Рассмотрим теперь, как можно реализовать блок источника. Один из способов, приведенный на рис. 4.15, использует блок задержки в любой форме (рис. 4.14), описанный в подразделе 4.3.3 и который, напомним, моделирует с помощью одного элемента задержки поведение множества независимых элементов задержки, даже в случае когда одновременно изменяются несколько вход-

ных переменных. Логический блок (рис. 4.15) содержит элемент И на каждом выходном полюсе v, а каждому входному состоянию x соответствует отдельный выходной полюс. Элемент И устанавливается в 1 тогда и только тогда, когда x и x_D представляют соответствующее этому элементу входное состояние. Например, если $m = 3$, в схеме есть элемент для реализации состояния 101, входами которого являются x_1, $\bar{x}_2$, x_3, x_{1D}, $\bar{x}_{2D}$ и x_{3D}. В установившемся состоянии, когда $x_1x_2x_3 = 101$, значения x_{1D}, $\bar{x}_{2D}$ и x_{3D} равны 1, 0 и 1 соответственно, в результате чего этот элемент устанавливается в 1. Никакие другие сигналы v не приобретают единичных значений. Если произойдет изменение входного состояния, например, $101 \longrightarrow 110$, то 101-элемент И перейдет в нулевое состояние, так как сначала изменяются x_1, x_2 и x_3. Однако 110-элемент И еще не приобретает единичного значения на выходе, так как $x_{1D}\bar{x}_{2D}x_{3D}$ остаются в 101. Поэтому все выходы v примут нулевые значения, как это и должно быть, и будут оставаться в них в течение промежуточного интервала длительности D, равной номинальному значению задержки блока задержки. По окончании этого периода времени $x_{1D}\bar{x}_{2D}x_{3D}$ изменяется в 110 (в этот момент сигналы в любой точке системы соответствуют входному воздействию $x_1x_2x_3 = 110$), и когда действие указанного изменения распространится до 110-элемента, последний перейдет в состояние 1. Через промежуток времени, несколько превышающий D, блок готов воспринять следующее входное изменение. Существуют и другие реализации блока источника.

Использование блока источника удобно в тех случаях, когда таблица переходов содержит много строк и относительно мало столбцов, в то время как метод, обсужденный ранее и использующий блок задержек для переменных состояния, более экономичен в противоположной ситуации.

При комбинированном использовании $(2S_0 + 1)$-кодирования строк и элемента инерциальной задержки для секторной переменной, блока логики, свободного от состязаний, и преобразования Мура МИВ-функции могут быть правильно реализованы схемами, не содержащими других элементов задержки, даже при условии допустимости изменения нескольких входных переменных.

4.4. Схемы, в которых паразитные задержки сосредоточены в логических элементах

Вместо предположения о том, что величины паразитных задержек только ограничены, введем более строгое ограничение: паразитные задержки сосредоточены лишь в логических элементах, а задержки соединительных проводов равны нулю. Иначе говоря, будем считать, что одиночные инерциальные паразитные задержки включены на выходных полюсах элементов схемы и только на них. Это означает, что, если сигнал в данном узле, которым может быть входной полюс или выход элемента, распространяется к нескольким входам элементов, то изменение этого сигнала проявится одновременно во всех указанных точках. Позже мы покажем, что полученные здесь результаты останутся справедливы и для схем, содержащих задержки в проводах, при условии, что величина задержки в любом проводе не превосходит минимальной задержки любого пути в логической схеме.

Если такое ограничение на паразитные задержки соблюдено, то любая ОИВ-функция может быть правильно реализована с помощью схем без задержек. Следует отметить, что одно из предположений, сделанных в теории схем, не зависящих от скорости (см. ссылки на работы Мюллера), заключается в том, что паразитные задержки сосредоточены в логических элементах. Естественно, справедливость такого допущения должна быть проверена при выборе используемой технологии производства. Здесь уместны также замечания, сделанные в начале раздела 4.3.

Рассмотрим теперь конкретный пример для иллюстрации введенных допущений. Таблица 4.13, *а* задает матрицу переходов в ОТП-кодирование ОИВ-функции с существенным состязанием, возникающим при изменении x_2 и начальном состоянии 3—11. Карты Карно функций Y_1 и Y_2 заданы в таблице 4.13, *б* и *в*. Мы не будем рассматривать функцию Z (в нашем случае $Z = Y_1$), поскольку она не имеет отношения к стоящим перед нами вопросам. Реализация Y_1 и Y_2 в виде двухуровневой логической схемы приведена на рис. 4.16, *а*.

Предположим, что в системе с начальным состоянием 3—11 входная переменная x_2 изменила свое значение из

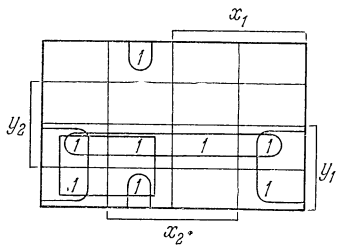
1 в 0. Тогда $x_2y_1y_2$-элемент (отмечен буквой A_3 на рис. 4.16, *а*), являющийся одним из элементов И, ведущих к полюсу Y_2, установится в 0 (см. также табл. 4.13, *в*).

Если инвертор, преобразующий x_2 в $\bar{x}_2$, содержит достаточно большую задержку, то изменение y_2 может достигнуть входа элемента A_1 (связанного с Y_1) раньше, чем сигнал $\bar{x}_2$ вызовет появление единичного сигнала на входе элемента A_2 (см. табл. 4.13, *б*). В действительности,

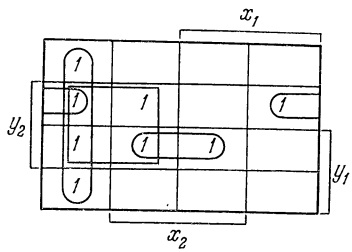
Таблица 4.13

	x_1x_2 00	01	11	10	y_1	y_2
1	2,0	4,1	***1***,0	***1***,0	0	0
2	***2***,0	***2***,0	1,0	***2***,0	0	1
3	***3***,1	***3***,1	***3***,1	4,1	1	1
4	3,1	***4***,1	1,0	***4***,1	1	0

а) Матрица переходов



б) Y_1-карта



в) Y_2-карта

как показано в разделе 1.4 при обсуждении рис. 1.2 и в разделе 4.3.2, сигнал Y_1 ведет себя так, будто состояние системы меняется из 3—11 в 4—11. Иначе говоря, создается такая ситуация, как будто изменение y_2 проявляется в узле Y_1 раньше, чем в Y_1 проявится изменение x_2. В результате Y_1 установится в 0. Если вход $\bar{y}_2$ элемента A_2 принимает значение 0, в то время как x_2 еще остается равным 0, то A_2 никогда не установится в 1 и система придет в состояние 1—10 вместо 4—10, как это предполагается. Мы видим, что даже при ограничении, введенном

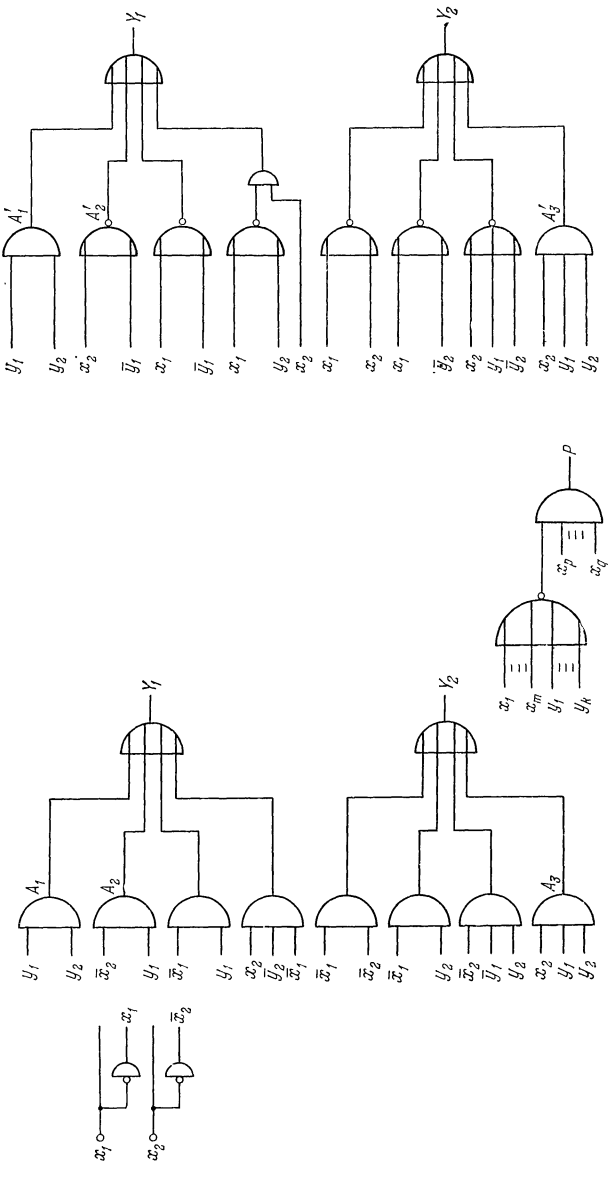


Рис. 4.16. *а*) Первая неудачная реализация таблицы 4.13, *а* без элементов задержки. *б*) Схема общего вида, заменяющая элементы И. *в*) Вторая неудачная реализация таблицы 4.13, *а* без элементов задержки.

в этом разделе, существенное состязание может вызвать тот же вид нарушения работы схемы, встречавшийся, когда чертенок, расставляющий паразитные задержки, обладает бо́льшей свободой.

На первом шаге, который мы сделаем для того, чтобы избавиться от неправильного функционирования, следует удостовериться, что последовательность полных состояний на любом переходе воспринимается на всех полюсах схемы в правильном порядке. Отметим, что о текущем полном состоянии можно судить по выходным сигналам элементов И, изображенных на рис. 4.16, *а*. Наш подход заключается в замене каждого элемента И логической схемой, реализующей ту же функцию, причем x-входы этих схем не должны содержать дополнений $\bar{x}$. Кроме того, значения y-входов должны проявляться тогда, когда их изменения не могут повлиять на логические значения, определяемые изменениями x-входов, происшедшими первыми.

Общий вид такой схемы, реализующей элементарное произведение $\bar{x}_1 \ldots \bar{x}_m y_1 \ldots y_k x_p \ldots x_q$, показан на рис. 4.16, *б*. $\bar{x}$-сигналы и все y-сигналы являются входными для элемента ИЛИ — НЕ, в то время как x-сигналы воздействуют непосредственно на элемент И. В тех случаях, когда элементарное произведение составлено не из всех трех указанных типов переменных, используется часть схемы рис. 4.16, *б*. При нашем допущении, что задержки в соединениях пренебрежимо малы, проявление любого x-изменения на выходе рассматриваемого составного элемента И завершается прежде, чем начнется распространение последующего y-изменения. Предположим, например, что на рис. 4.16, *б* $x_p = \ldots = x_q = 1$, $x_1 = \ldots = x_m = 0$ и все остальные входы (y-переменные или их отрицания) равны 0, за исключением y_1. Тогда выход $P = 0$. Если x_1 изменяется в 1 и вызывает изменение в 0 переменной y_1, то непосредственным эффектом изменения x_1 явится поддержание выхода элемента ИЛИ — НЕ (и, следовательно, P) в 0, поскольку изменение y_1, которое стремится вызвать на выходе ИЛИ — НЕ единичное значение, происходит на входе того же элемента позже. Мы предполагаем, что изменение x_1 на этом элементе происходит одновременно с изменением x_1 на элементе, вызываю-

щем установку в 0 переменной y_1. Никаких особых паразитных задержек на входе любого элемента появиться не может.

На рис. 4.16, *в* элементы И схемы, приведенной на рис. 4.16, *а*, заменены составными элементами И. Теперь во время перехода $3—11 \longrightarrow 3—10 \longrightarrow 4—10$ элемент A_3' устанавливается в 0, вызывая изменение в 0 сигнала y_2, тогда как A_2' приобретает единичное значение. И когда изменение y_2 распространится до входов элементов первого яруса, элемент A_1' установится в 0. Это соответствует правильному порядку распространения изменений x_2 и y_2. Теперь элемент, соответствующий A_2 на рис. 4.16, *а*, не может воспринять изменение x_2 позже изменений y_1 и y_2 и тем самым сохранить нулевой сигнал на своем выходе.

К сожалению, предотвращение такого типа нарушений функционирования недостаточно для обеспечения правильного поведения всей схемы. Хотя A_1' не устанавливается в 0 до тех пор, пока A_2' не примет единичного выходного значения (этот факт здесь может быть успешно проверен по K-карте Y_2), задержка на выходе A_2' может быть достаточно велика, чтобы сохранить равным 0 соответствующий входной сигнал элемента ИЛИ на выходе Y_1 до того момента, пока входное воздействие с выхода A_1' имеет нулевое значение. Таким образом, Y_1 может все-таки ошибочно принять нулевое значение. Предотвратив нарушения на первом ярусе логической схемы, мы перенесли его на второй.

Для того чтобы понять, как избавиться от неправильного функционирования второго яруса, необходимо рассмотреть переходный процесс в элементе ИЛИ при изменении его входных сигналов. Допустим, что каждый входной сигнал изменяется не более одного раза в течение некоторого интервала времени. Если начальное значение выхода равно 0, то независимо от того, является ли конечное значение выхода нулем или единицей, во время перехода не может возникнуть никаких кратковременных ошибочных выходных сигналов. Этот факт очевиден, если принять во внимание доводы, приведенные в разделе 4.2 при обсуждении 0-состязаний. Также справедливо следующее: если выходной сигнал меняется из 1 в 0, то

изменение происходит, как только последнее единичное входное значение стало нулевым, и в этом случае не возникает кратковременных ошибочных сигналов, поскольку ни один вход не принимает снова единичного значения. Однако, если начальный и конечный выходной сигнал равен 1, во время перехода возможно появление нулевого импульса. Так может случиться, если все сигналы, первоначально равные 1, станут нулевыми раньше, чем любой из сигналов, изменяющихся из 0 в 1, установится в 1. Итак, ложными сигналами могут быть только нулевые импульсы.

Этот факт может быть использован, если нам удастся разработать систему, в которой никакие изменения состояний не могут быть вызваны нулевыми сигналами.

Рассмотрим RS-триггер, приведенный на рис. 4.17, *а*. Допустим, что сигналы, равные 1, никогда одновременно не появляются на общих входах R и S. Тогда на выходах y и $\bar{y}$ установятся соответственно 1 и 0 (это состояние называется *установкой в* 1), если на вход S последним подан сигнал 1. Если триггер установлен в 1, то подача 1 на вход R установит его в нулевое состояние, т. е. y изменит свое значение на 0, а $\bar{y}$ — на 1. Если $S = R = 0$, состояние триггера не изменится. Мимоходом заметим: если задержки отсутствуют, то во время изменения состояния выходы y и $\bar{y}$ никогда одновременно не будут равны 1. Для наших целей очень удобно, что кратковременный нулевой импульс на входе RS-триггера никогда не вызывает изменения состояния.

Хорошо известно, что RS-триггеры можно использовать в качестве элементов памяти как в асинхронных, так и в синхронных последовательностных схемах (см., например, [74]). Те же критерии применимы для пригодных способов кодирования строк, и как только кодирование строк выполнено, необходимо определить соответствующие R- и S-функции для каждого триггера (одного для каждого сигнала y). Например, если содержимое клетки следующего состояния в некотором полном состоянии, для которого $y_i = 1$, соответствует состоянию, в котором $y_i = 0$, то мы задаем $R_i = 1$ и $S_i = 0$. Если в текущем и в следующем состояниях $y_i = 1$, то мы задаем $R_i = 0$ и оставляем S_i без изменений. Отметим, что

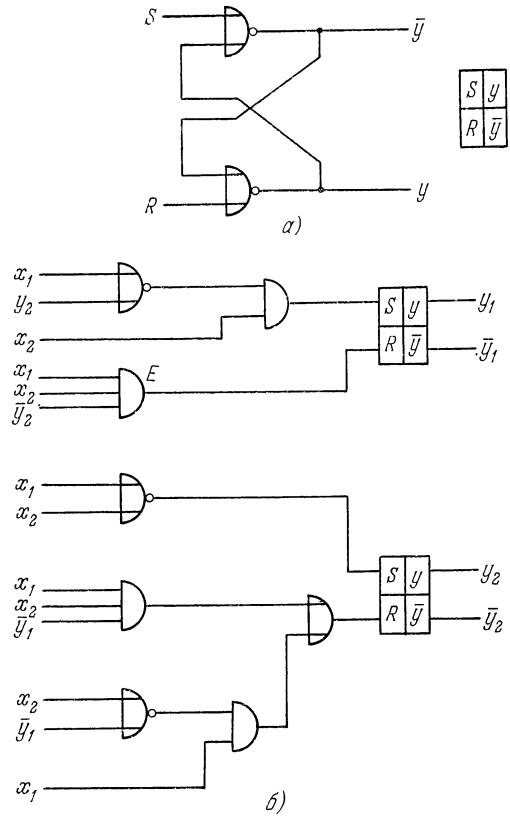
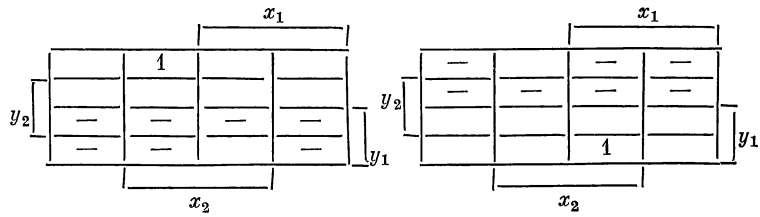
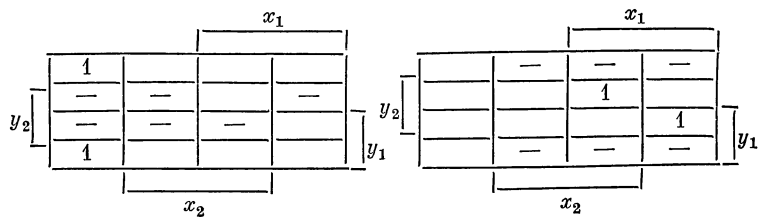


Рис. 4.17. *а*) *RS*-триггер. *б*) Реализация таблицы 4.13, *а* без задержек.

Таблица 4.14



а) S_1-карта б) R_1-карта



в) S_2-карта г) R_2-карта

сигнал установки в 1, приложенный к триггеру, установленному в состояние 1, не изменяет этого состояния, и подобным образом, если $\bar{y} = 1$ и R получает сигнал 1, то триггер остается в состоянии 0. В таблице 4.14 приведены карты для управляющих сигналов, необходимых для реализации матрицы переходов из таблицы 4.13, *а*. Результирующими логическими выражениями являются

$$S_1 = \bar{x}_1 x_2 \bar{y}_2, \quad R_1 = x_1 x_2 \bar{y}_2,$$

$$S_2 = \bar{x}_1 \bar{x}_2, \quad R_2 = x_1 x_2 \bar{y}_1 + x_1 \bar{x}_2 y_1.$$

Эти выражения реализованы на рис. 4.17, *б* в специальном виде, обсуждавшемся ранее.

Поскольку, как уже было показано, при нашем предположении о паразитных задержках ошибочные единичные сигналы не могут быть порождены логической схемой и поскольку триггеры не реагируют на ложные нулевые сигналы, схема без задержек, изображенная на рис. 4.17, *б*, правильно реализует таблицу 4.13, *а*. Такая же процедура может быть использована для реализации произвольной ОИВ-функции в предположении, что допустимы изменения одной входной переменной и что задержки в проводах пренебрежимо малы. Заметим, что в силу отсут-

ствия необходимости бороться со статическими 1-состязаниями, нет нужды в покрытии каждой соседней пары единичных значений Y-карт простыми импликантами. Хотя мы предполагали использовать ОТП-кодирование состояний, существуют другие методы синтеза, справедливые для любого варианта кодирования состояний, свободного от критических состязаний.

Выясним, насколько можно ослабить наше ограничение, наложенное на паразитные задержки, чтобы стало возможным неправильное функционирование схемы, изображенной на рис. 4.17, *б*. Рассмотрим снова переход $3—11 \longrightarrow 3—10 \longrightarrow 4—10$ с существенным состязанием, допуская теперь, что паразитная задержка величины d содержится в проводе, ведущем к входу x_2 элемента E, соединенного с входом R_1 триггера. В таком случае (задержки в других проводах пока пренебрежимо малы) изменение x_2 $1 \longrightarrow 0$ приведет к изменению выхода нижнего элемента ИЛИ — НЕ, поэтому связанные с ним элементы И и ИЛИ также изменят свои выходные сигналы, что в свою очередь вызовет установку в 0 триггера (y_2 и $\bar{y}_2$ изменятся). Если d достаточно велико, чтобы изменение x_2 не изменило значения E прежде, чем E изменится под действием изменения $\bar{y}_2$, то выход триггера y_1 установится в 0 ошибочным сигналом, и конечным состоянием будет 1—10. Для того чтобы произошло такое событие, d должно превосходить полную задержку в пути между x_2 и $\bar{y}_2$, которая включает как задержку в логике, так и время, необходимое для изменения состояния триггера.

В общем случае мы можем сказать, что, если задержка в каждом проводе меньше суммы минимальных задержек логики и триггера, то схемы типа, рассмотренного в этом разделе, будут работать правильно. В действительности для того чтобы имело место нарушение правильности функционирования, d должно превосходить значение, определенное выше, на величину, превосходящую сумму величины наибольшей инерциальной задержки между элементом E и триггером R_2 и времени, необходимого для изменения состояния триггера. Таким образом, некоторый запас безопасности неявно заложен в указанную нами оценку.

4.5 Анализ асинхронных последовательностных схем

В этом разделе излагается материал, связанный с решением основных вопросов, встающих перед исследователем при анализе последовательностных схем: как построить таблицу переходов заданной последовательностной схемы произвольного вида, имеющей несколько входных и выходных полюсов (пример такой схемы приведен на рис. 4.18), и каким образом выбрать соотношения между величинами паразитных задержек, элементов задержки и скоростью следования входных сигналов, чтобы построенная схема работала правильно.

4.5.1. Выбор внутренних переменных и построение таблицы переходов. Основная проблема, с которой при-

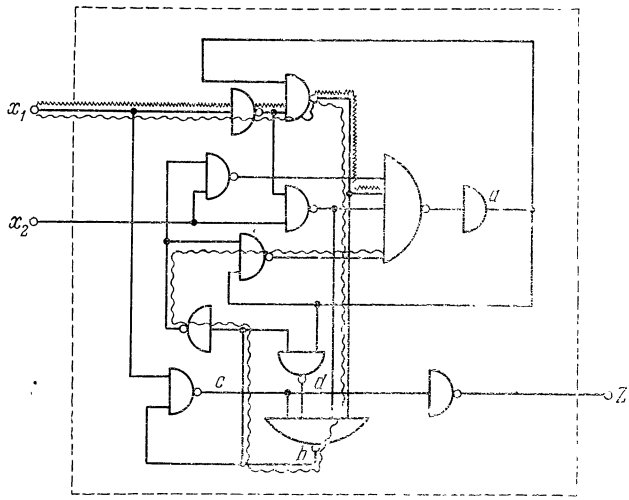


Рис. 4.18. Исследуемая схема.

ходится сталкиваться при анализе схем (в качестве примера будем рассматривать схему, изображенную на рис. 4.18 и содержащую для разнообразия элементы И — НЕ), заключается в определении множества *ветвей состояний*, воспроизводящих сигналы, которые можно использовать как внутренние переменные. Если схема содержит элементы задержки, решение этого вопроса не-

сколько упрощается, поскольку каждый элемент задержки должен быть выбран как ветвь состояния. В противном случае, вопреки нашим основным допущениям, внутри логического блока оказались бы элементы задержки, когда схема приведена к стандартному виду.

В нашем примере присутствует только одна задержка, и поэтому мы можем обозначить ее выход (точка a на рис. 4.18) как y_1, тогда ее входом будет, конечно, Y_1. Появилась ли теперь возможность выразить выход Z как функцию от x_1, x_2 и y_1? Выходной сигнал Z, снимаемый с выхода инвертора, зависит от сигнала c. Этот сигнал в свою очередь является выходным сигналом элемента, на входах которого действуют x_1 и сигнал b, поэтому $Z = \overline{\overline{x_1 b}} = x_1 b$. Однако сигнал b является функцией c; следовательно, мы не в состоянии получить комбинационную функцию, описывающую Z только через y_1 и входы x_1, x_2. Такое же затруднение возникло бы, если бы мы попытались определить Y_1.

Эти затруднения происходят из-за того, что в схеме присутствуют цепи обратной связи, которые не содержат ветви состояния y_1. Необходимо, чтобы множество ветвей состояний заключало по меньшей мере по одному узлу каждого замкнутого пути схемы. Таким образом, в нашем примере мы должны выбрать по крайней мере еще одну ветвь состояния. При внимательном изучении схемы становится ясно, что множество ветвей состояний должно быть дополнено ветвью, содержащей точку b. Другими словами, если разорвать схему в точках a и b, все замкнутые пути схемы распадутся. Поэтому обозначим сигнал, выходящий из точки b, через y_2, а сигнал, поступающий в точку b, через Y_2. Так как в этой ветви нет элемента задержки, то $y_2 = Y_2$.

Теперь уже можно выразить Y_1, Y_2 и Z в виде функций от x_1, x_2, y_1 и y_2:

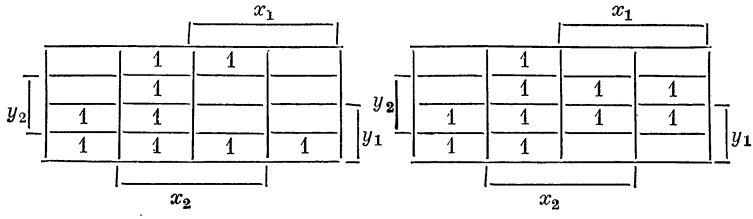
$$Y_1 = \overline{\overline{(y_1\bar{y}_2)\,(\bar{x}_1 x_2)\,(\bar{x}_1 y_1)\,(x_2 \bar{y}_2)}} = y_1\bar{y}_2 + \bar{x}_1 x_2 + \bar{x}_1 y_1 + x_2\bar{y}_2,$$

$$Y_2 = \overline{\overline{(x_1 y_2)\,(y_1 y_2)\,(\bar{x}_1 x_2)\,(\bar{x}_1 y_1)}} = x_1 y_2 + y_1 y_2 + \bar{x}_1 x_2 + \bar{x}_1 y_1,$$

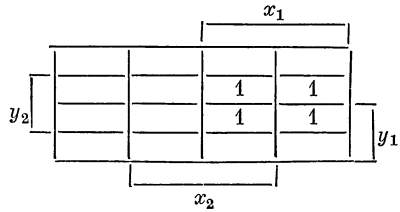
$$Z = \overline{\overline{x_1 y_2}} = x_1 y_2.$$

Из полученных выражений легко получаются K-карты

Таблица 4.15



а) Y_1-карта б) Y_2-карта



в) Z-карта

г) Y_1Y_2-матрица

y_1 y_2 \ x_1x_2	00	01	11	10
0 0	00	11	10	00
0 1	00	11	01	01
1 1	11	11	01	01
1 0	11	11	10	10

д) Матрица переходов

	x_1x_2 00	01	11	10	y_1	y_2
1	***1***,0	3,0	4,0	***1***,0	0	0
2	1,0	3,0	***2***,1	***2***,1	0	1
3	***3***,0	***3***,0	2,1	2,1	1	1
4	3,0	3,0	***4***,0	***4***,0	1	0

(см. табл. 4.15, *а*, *б* и *в*). Объединяя карты Y_1 и Y_2, находим матрицу возбуждения (табл. 4.15, *г*), по которой составляем матрицу переходов (табл. 4.15, *д*). Точность, с которой матрица переходов описывает реальное поведение любой физической схемы, соответствующей рис. 4.18, должна быть определена при дальнейшем анализе с учетом задержек и состязаний. Но сначала добавим несколько слов о выборе ветвей состояний.

Каждая ветвь, содержащая элементы задержки, выбирается в качестве ветви состояния. В множество ветвей состояний включается, кроме того, не меньше одного узла, принадлежащего каждому замкнутому пути схемы. Вообще говоря, задача составления множества ветвей состояний имеет неоднозначное решение, причем в каждом

случае построенное множество может содержать различное число ветвей. Если схема свободна от состязаний, то всевозможные таблицы переходов, соответствующие различным множествам ветвей состояний, подобны и отличаются только наличием строк, в которых нет устойчивых состояний и которые недостижимы из любой строки, содержащей устойчивые состояния, множествами строк, которые могут быть совмещены и, наконец, выходными сигналами в неустойчивых полных состояниях. Например, в результате некоторого другого выбора ветвей состояний можно построить таблицу переходов, в которой выходной сигнал при полном состоянии, соответствующем состоянию 2—00 из таблицы 4.15, *е*, равен 1.

Определенное удобство состоит в составлении минимального множества обрываемых ветвей обратной связи, поскольку в противном случае таблица переходов получается неминимальной. Проблема поиска минимального множества привлекала внимание специалистов по теории графов (см., например, [121]).

При анализе релейно-контактных последовательностных схем выбор ветвей состояний существенно упрощается (см. раздел 4.6).

4.5.2. Определение условий наличия состязаний. После построения таблицы переходов перейдем к определению условий, при которых поведение схемы согласуется с матрицей переходов. Сначала необходимо проверить кодирование строк на критические состязания, учитывая, что некоторые строки матрицы могут быть эквивалентными. В нашем примере критических состязаний нет.

Затем для выявления комбинационных состязаний, возникающих при изменениях одной входной переменной, применим методы, изложенные в разделе 4.2. В представленном примере из-за наличия элементарного произведения x_2y_2 возможно 1-состязание в Y_2 на переходе $2—11 \longrightarrow 2—01$. Хотя при этом может произойти кратковременный ошибочный переход в 1—01, конечное состояние всегда будет правильным, поскольку столбец 01 содержит только одно устойчивое состояние. Следует всегда помнить о такой особенности последовательностных схем, так как иногда она может предотвратить существенное нарушение функционирования, вызванное комбинационным состязанием.

Теперь необходима проверка на существование s-троек и существенных состязаний. В нашем примере есть две s-тройки, причем обе заканчиваются в столбце 01, содержащем единственное устойчивое состояние. Поэтому состязания устойчивых состояний невозможны. Однако на выходе могут возникнуть переходные состязания при переходе $4—11 \longrightarrow 4—01 \longrightarrow 3—01$ из-за наличия единичного выходного сигнала в состоянии 3—11 и отсутствия задержки в линии $Y_2 — y_2$. В общем случае, когда по внешнему виду матрицы переходов нельзя сказать ничего определенного, для выявления существенных состязаний следует проверить задержки путей схемы с помощью метода, излагаемого ниже.

Таблица 4.15, д содержит существенное состязание на переходе, начинающемся с $3—00 \longrightarrow 3—10$, когда изменяющейся внутренней переменной является y_1. Элемент задержки, включенный в ветвь y_1, может предотвратить нарушение функционирования на этом переходе, если его величина задержки достаточно велика. Вместо вычисления необходимой нижней границы значения задержки этого элемента используем другой пример возникновения иного существенного состязания, присущего нашей таблице на переходе, начинающемся с $4—10 \longrightarrow 4—00$. Здесь «незадержанная» переменная y_2 является одной из изменяющихся переменных. При рассмотрении K-карт для Y и алгебраических выражений для Y-переменных легко видеть, что после изменения x_1 элементарное произведение $\bar{x}_1 y_1$ принимает значение 1, что в свою очередь приводит к установлению Y_2 в 1. Как только y_2 станет равной 1, элементарное произведение $y_1 \bar{y}_2$ примет значение 0, а произведение $x_1 y_1$ приведет к выполнению равенства $Y_1 = 1$. Опасность заключается в том, что действие изменения $\bar{x}_1 \bar{y}_1$ на Y_1 окончится прежде, чем действие изменения $y_1 \bar{y}_2$ вызовет установление Y_1 в 0. (Мы считаем, что задержка y_1 не является инерциальной, так что ошибочный сигнал Y_1 обязательно повлияет на y_1.) Поэтому необходимо сравнить задержки пути, отмеченного волнистой линией на рис. 4.18, от x_1 через $\bar{x}_1 y_1$-элемент, Y_2-элемент, $\bar{y}_1 y_2$-элемент до входа Y_1-элемента и пути, отмеченного более частой волнистой линией, от x_1 через $\bar{x}_1 y_1$-элемент до входа Y_1-элемента.

Поскольку эти пути начинаются в одной точке, достаточно сравнить задержки отрезков путей, начинающихся с точки разветвления, т. е. с выхода $x_1\bar{y}_1$-элемента. Если паразитные задержки первого пути больше, чем задержки второго, то нарушений не происходит. Однако, если задержка более короткого пути на схеме (не следует забывать, что изображение схемы отражает только топологические связи, а не размеры схемы) больше, то ошибочный сигнал Y_1 появится на время, равное разнице значений задержек путей. В результате в схеме начнется колебательный процесс по состояниям из столбца 00.

Как указано в разделе 4.3, такие нарушения ликвидируются введением элемента задержки в ветвь состояния y_2 с величиной задержки, достаточной для восстановления баланса. В действительности этот элемент задержки может быть расположен в любой точке отрезка пути, отмеченного волнистой линией, не являющегося общим с другим путем, однако в общем случае включение задержки не в ветви y_2 может создать неприятности на других переходах, что потребует принятия дополнительных мер предосторожности. Рассматриваемая проблема также разрешается, если задержка представляет собой инерциальную задержку той же величины.

Аналогичный анализ может быть проведен для каждого r-перехода и существенного состязания матрицы переходов. Вообще говоря, мы должны определить два пути распространения сигнала, начинающихся на входном полюсе, соответствующем x-переменной, с изменения которой начинается переход. Один путь (*путь возникновения нарушения*) представляет собой последовательное соединение пути распространения сигнала, изменяющего y-переменную, которая реагирует на этот сигнал, и пути распространения сигнала от указанного y-полюса к Y-полюсу, на котором может возникнуть ошибочная реакция. Второй путь (*путь исправления нарушения*) есть путь распространения сигнала, который исправляет ошибочную реакцию, передавая новое x-значение на второй Y-полюс. Неприятности могут возникнуть, если задержка в пути исправления нарушения превышает задержку пути возникновения нарушения.

Ситуация усложняется, хотя и не является совершенно иной, в случае возникновения некритических состя-

заний. В этом случае необходимо анализировать большее множество путей. Если же ряд y-изменений происходит в последовательности, соответствующей многотактному кодированию строк или МИВ-функции, анализ следует проводить на каждом такте отдельно. Например, если вслед за изменением y_i происходит изменение y_j, то переменная y_i и все остальные y-переменные, изменившиеся раньше y_i в последовательности, должны рассматриваться в качестве переменных, с изменения которых начинается переход.

Часто полезно определить, выполняется ли менее сильное, но более удобное для проверки достаточное условие правильности реализации, особенно в тех случаях, когда ситуация усложнена из-за причин, рассмотренных в предыдущем подразделе. Пусть d_{Lm} и d_{LM} — минимальная и максимальная величины задержки путей схемы, начинающихся от x- или y-полюсов, а d_{fm} — наименьшая задержка некоторой ветви состояния (цепи обратной связи). Тогда, поскольку пути возникновения нарушения требуют двойного прохода сигнала по схеме, а пути исправления нарушения — только одного, то рассматриваемые нарушения всегда устранимы, если выполняется неравенство

$$2d_{Lm} + d_{fm} > d_{LM}.$$

Это весьма удобный критерий, который можно использовать при синтезе последовательностных схем.

4.5.3. Выявление состязаний с помощью троичной логики. При анализе сложных схем часто полезно выяснить, возможны ли состязания при конкретных переходах. В этом разделе будет рассмотрен подход к решению такой задачи, удобство которого заключается в возможности использования вычислительной машины для моделирования как комбинационных, так и последовательностных цифровых схем.

Основная идея подхода состоит в введении третьего значения, что дает возможность перейти от двоичных переменных к троичным. Новое значение, обозначаемое 1/2, является промежуточным между 0 и 1, т. е. предполагается, что при изменении переменной x между значениями 0 и 1 (в любом направлении) на протяжении всего интервала изменения $x = 1/2$. Также предполагается, что переменные имеют значения 1/2, если неизвестно

их действительное значение. Изучение переходных процессов в схеме ведется следующим образом: вычисляются значения выходов для начального и конечного входных воздействий, а также для промежуточных, когда изменяющимся переменным приписано значение 1/2.

Сначала рассмотрим комбинационные схемы для выявления в них статических состязаний, возникающих из-за изменения одной или нескольких входных переменных. Заметим, что в случае, когда хотя бы один из входов элемента ИЛИ равен 1, выход элемента равен 1, независимо от значений остальных входов. Тогда, если предположить, что состязания не могут возникать в самих логических элементах, изменение любого другого входа ИЛИ не изменяет его выхода. Следовательно, выход этого элемента равен 1 и тогда, когда некоторые входы принимают значения 1/2. С другой стороны, если все входы, за исключением одного, имеют значение 0, то на выходе устанавливается 0 или 1 в зависимости от того, равен оставшийся вход 0 или 1. Поэтому, если на этом входе появилась 1/2, то и выход примет значение 1/2. Аналогичные рассуждения справедливы для элемента И при замене нуля на единицу и наоборот. В общем случае, если элемент реализует функцию $f(x_1, x_2, \ldots, x_n)$ и значение функции $f(x_1, x_2, \ldots, x_p, a_{p+1}, a_{p+2}, \ldots, a_n)$ (где $x_i = a_i$ при $i = p+1, p+2, \ldots, n$, a_i равны 0 или 1) равно 0 (или 1) независимо от значений $x_i (i = 1, 2, \ldots, p)$, то будем считать, что $f(1/2, \ldots, 1/2, a_{p+1}, a_{p+2}, \ldots, a_n) = 0$ (или 1). В противном случае $f(1/2, \ldots, 1/2, a_{p+1}, a_{p+2}, \ldots, a_n) = 1/2$. Используя такое предположение, легко вычислить значение выхода любого элемента для произвольного входного воздействия. Например, выход элемента «сумма по модулю 2» равен 1/2, если его вход принимает значение 1/2, так как изменение любого входа всегда меняет значение выхода. Нетрудно видеть, что, если произвольный вход произвольного элемента меняется на 1/2, выход этого элемента либо не меняется, либо также принимает значение 1/2. И наоборот, если произвольный вход изменяет свое значение с 1/2 на 0 или 1, выход элемента либо не меняется, либо меняется с 1/2 на 0 или 1 в зависимости от типа элемента.

Рассмотрим теперь комбинационную схему. Предположим, что некоторый входной сигнал принял значение

1/2. Тогда выходы тех элементов, которые соединены с этим входом, могут измениться лишь в 1/2; распространяя это изменение по всей схеме, получим, что выходной сигнал схемы либо останется в 0 или 1, либо изменится на 1/2. Аналогично, если входное воздействие меняется с 1/2 на 0 или 1, выход должен измениться с 1/2 на 0 или 1 или остаться неизменным.

Данный выходной сигнал может измениться при изменении некоторого подмножества входных сигналов в том и только том случае, когда справедливо следующее условие: данный выход принимает значение 1/2, если указанное множество входов имеет значение 1/2, а остальные входы зафиксированы в их первоначальных значениях (0 или 1). Такое утверждение очевидно для одноуровневых схем и может быть распространено по индукции на схемы общего вида. Теперь мы в состоянии сформулировать следующую процедуру.

Процедура 4.2. *Выявление статических состязаний в комбинационных схемах для заданного изменения входного воздействия.*

1. Определить, совпадают ли значения на заданном выходе при начальном и конечном входных воздействиях.

2. При совпадении начального и конечного выходных значений состязание возможно, если при установлении изменяющихся входов в 1/2 выход равен 1/2.

Эта процедура вытекает из предыдущих рассуждений. Рассмотрим действие процедуры 4.2 на примере, применяя ее к схеме (рис. 4.19), исследованной в разделе 4.2. Пусть на входах $ABCD$ происходит изменение $0011 \longrightarrow \longrightarrow 1111$. Начальное и конечное значения выхода равны 1, а на входном воздействии 1/2 1/2 11 выход равен 1/2. Следовательно, в схеме возникает статическое 1-состязание при данном входном изменении. Такой вывод подтверждается приведенными ранее исследованиями, указывающими на существование функционального 1-состязания. Анализ изменения $0101 \longrightarrow 1101$ показывает, что $Z = 1$ при начальном и конечном воздействиях и $Z = 1/2$ при $ABCD = 1/2\ 101$. Мы выявили еще одно статическое 1-состязание, которое, как было показано ранее, является логическим.

Обратимся теперь к последовательностным схемам, для которых обсуждаемая процедура должна указывать, при-

обретают ли при заданном входном изменении (не обязательно по одной переменной) y-переменные конкретное устойчивое состояние независимо от величины задержек. Предположим сначала, что рассматриваемые схемы не имеют задержек*). Процедура состоит из двух частей А и Б.

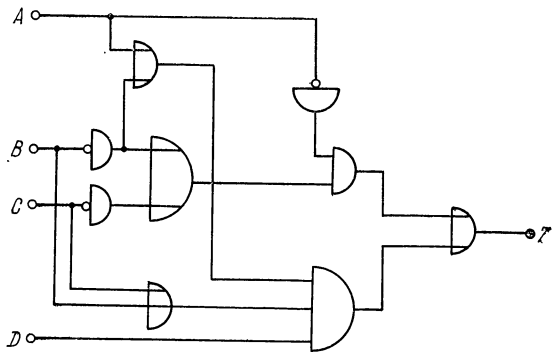


Рис. 4.19. Схема, анализируемая на комбинационные состязания.

Процедура 4.3А. *Построение множества y-переменных, которые могут измениться при заданном изменении входных переменных.*

1. Установить y-переменные и не меняющиеся на заданном изменении x-переменные в их начальные значения (0 или 1) и присвоить изменяющимся x-переменным значение 1/2.

2. Вычислить Y-функции и определить множество Y-переменных, принимающих значение 1/2.

3. Присвоить значение 1/2 тем y-переменным, которые соответствуют множеству Y-переменных, установившихся в 1/2.

4. Повторять шаги 2 и 3 до тех пор, пока множество Y-переменных, полученное на шаге 2, не перестает пополняться. Тогда совокупность y-переменных, равных 1/2, представит собой искомое множество.

Справедливость этой процедуры следует из предыдущих рассуждений. Проиллюстрируем процедуру 4.3А на примере. Рассмотрим схему, изображенную на рис. 4.20,

*) Предлагаемый метод несправедлив в том случае, когда осцилляция y-состояний предусмотрена при создании схемы.

и определим для нее множество y-переменных, которые могут измениться после одновременного изменения x_1 и x_2 при начальном состоянии $x_1x_2y_1y_2=0000$. Сначала отметим, что начальное состояние действительно устойчивое. Полагая $x_1x_2y_1y_2=1/2\,1/2\,0\,0$, получаем, что Y_2 приобретает значение 1/2. Повторяя процедуру, полагаем

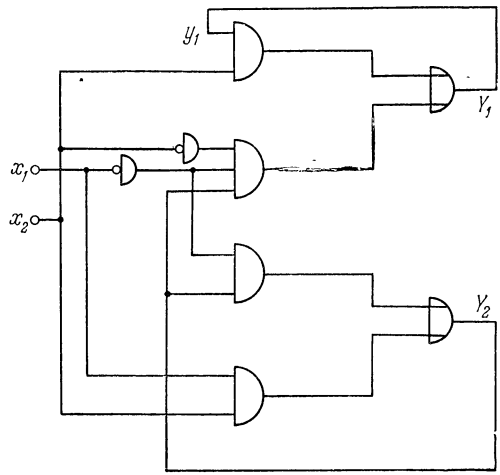


Рис. 4.20. Исследуемая последовательностная схема.

$x_1x_2y_1y_2=1/2\,1/2\,0\,1/2$ и вычисляем, что $Y_1=1/2$. Заметим, что нам не нужно вновь вычислять значение Y_2, поскольку, как было показано ранее, изменение значения входа, в данном случае y_2, из 0 в 1/2 не может привести к изменению какого-либо выхода из 1/2 в 0 или 1. Итак, мы определили, что на данном изменении изменяются обе y-переменные.

Теперь для определения множества тех y-переменных, которые примут определенные значения 0 или 1 по окончании входного изменения, введем следующую процедуру.

Процедура 4.3Б. *Определение множества y-переменных, принимающих определенные значения.*

1. Присвоить y-переменным значения, соответствующие их значениям по окончании процедуры A, присвоить x-переменным значения, соответствующие новому входному воздействию.

2. Вычислить Y-функции (измениться могут лишь те Y-переменные, которые по окончании процедуры А имели значения 1/2).

3. Если некоторые Y-переменные установились в 0 или 1, изменить соответствующие y-переменные и повторить шаг 2. Продолжать процедуру до тех пор, пока не прекратятся изменения. В результате y-переменные, оставшиеся в 1/2, либо находятся в режиме генерации, либо могут установиться в 0 или 1 в зависимости от распределения паразитных задержек.

Как и раньше, справедливость процедуры следует из приведенных выше рассуждений. В нашем примере (см. рис. 4.20), полагая $x_1x_2y_1y_2 = 1\,1\,1/2\,1/2$, получим $Y_2 = 1$. Повторяя шаг 2 с новым значением $x_1x_2y_1y_2 = 1\,1\,1/2\,1$, находим, что никаких других изменений не происходит, т. е. y_1 не может установиться ни в 0, ни в 1.

Если схема содержит элементы задержки, то процедуру следует дополнить: после изменения сигналов на входах y-переменные обратных связей без задержек принимают свои новые значения до того, как изменится любая y-переменная обратной связи с задержкой. Таким образом, мы применяем процедуры А и Б, полагая, что y-переменные обратных связей с задержками зафиксированы в своих начальных значениях. Если входной сигнал (значение Y-переменной) элемента совершенной задержки устанавливается в 1/2 при процедуре А, то значение остается постоянным до окончания процедуры Б и не проверяется вновь, поскольку, если вход совершенной задержки изменился, в конце концов изменится и ее выход независимо от величины задержки. Входы элемента инерциальной задержки не проверяются до окончания процедуры Б, так как кратковременные изменения на входе сглаживаются. Если некоторые Y-переменные, связанные с элементами задержки, изменят свои значения на 1/2, 0 или 1 во время выполнения описанных процедур, следует присвоить эти новые значения соответствующим y-переменным и повторять все действия до тех пор, пока не прекратятся изменения Y-переменных или пока не встретится y-состояние, вычисленное ранее. Последняя ситуация определяет генерацию в схеме; в этом случае все y-переменные, меняющиеся в течение цикла, устанавливаются в 1/2, и работа процедуры завершается.

Представленные в этом подразделе процедуры можно запрограммировать для вычислительной машины, а с помощью такой программы удобно выявлять возможность возникновения состязаний в достаточно сложных схемах при подаче последовательностей входных воздействий. После того как выявлено состязание, следует найти причину, его определяющую, и средства его устранения с помощью более детального исследования. Прослеживание по схеме путей распространения сигналов, равных 1/2, является очень удобным методом для понимания идей, обсуждаемых в других разделах настоящей главы.

4.6. Специальные вопросы, связанные с релейно-контактными последовательностными схемами

Хотя до сих пор мы рассматривали схемы, состоящие из бесконтактных логических элементов, развитая нами теория применима и к релейно-контактным устройствам. Однако при конструировании релейно-контактных схем существуют некоторые вопросы, заслуживающие особого внимания. Они будут обсуждены после короткого вступления, в котором мы покажем, как развитые ранее систему обозначений и методы можно применить к релейно-контактным схемам.

Типичная релейно-контактная последовательностная схема изображена на рис. 4.21. Важно обратить внимание на следующие особенности, присущие релейно-контактным схемам.

1. Один конец каждой обмотки реле соединен с источником напряжения, а другой — с контактной схемой, обеспечивающей при определенных условиях связь этого конца с земляной шиной (или вторым полюсом источника напряжения).

2. Если обмотка реле R находится под напряжением, то значение *переменной возбуждения* R равно 1.

3. Нормально-открытые (*прямые*) контакты r замыкаются ($r \longrightarrow 1$) спустя некоторое время после того, как R принимает значение 1. Таким образом, между переменными R и r реализуется такое же соотношение, как между входом и выходом элемента инерциальной задержки.

Нормально-закрытые (*обратные*) контакты $\bar{r}$ размыкаются ($\bar{r} \longrightarrow 0$) через некоторое время после $R \longrightarrow 1$.

4. Различные контакты одного и того же реле могут реагировать на возбуждение через различные промежутки времени. Этот эффект, называемый *дребезгом контактов*, аналогичен действию паразитных задержек в логических схемах.

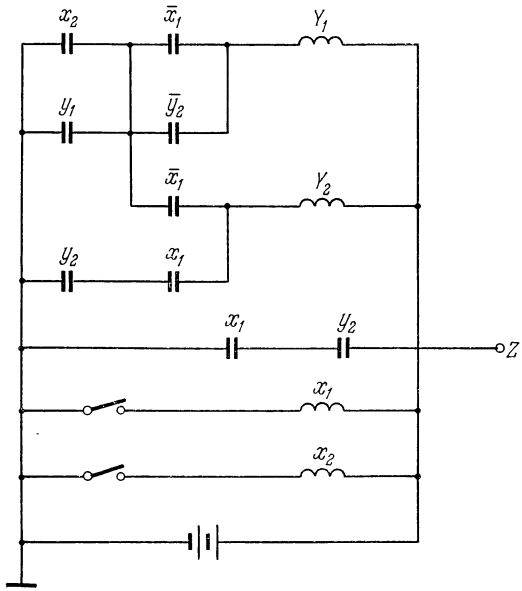


Рис. 4.21. Релейно-контактная последовательностная схема, реализующая таблицу 4.15, *д*.

Входные переменные обычно представляются состояниями механических переключателей с несколькими прямыми и обратными контактами или реле (называемыми *первичными реле*), управляемых непосредственно входными переключателями. Внутренним переменным соответствуют состояния Y-реле (*вторичных реле*), управляемых схемами из контактов первичных и вторичных реле (x- и y-контактов соответственно).

Синтез релейно-контактных схем ведется в соответствии с принципами, изложенными выше. После выполне-

ния процедур кодирования строк и определения логических Y- и Z-функций контактная схема, реализующая эти функции, синтезируется с помощью любого метода, развитого для таких устройств [12], но так, чтобы в схеме не возникали комбинационные состязания.

Известно, что релейно-контактные схемы весьма уязвимы с точки зрения возникновения в них существенных состязаний, так как явление дребезга особенно заметно при срабатывании прямых и обратных контактов. Свойство y-реле, аналогичное действию элемента инерциальной задержки, может быть недостаточным для исключения таких состязаний, поэтому в схемах следует применять реле с замедленным срабатыванием. Наилучшим методом во многих случаях считается использование управления некоторыми вторичными реле *шунтированием* их обмоток. При этом выполняется последовательное соединение обмотки реле с источником напряжения и сопротивлением, а параллельно обмотке подключается контактная схема, реализующая дополнение Y-функции.

Таким способом часто возможно заменить состязания между разнородными контактами одного и того же x-реле состязаниями между однородными контактами, разброс времен срабатывания которых обычно гораздо меньше.

Некоторые проблемы возникают в релейно-контактных схемах в связи с использованием источника входных сигналов. В тех случаях, когда входные сигналы подаются в схему через первичные реле, действие источника аналогично действию элементов инерциальной задержки, включенных последовательно с каждым входным полюсом. Строгий анализ таких схем (см. подраздел 4.5.1) требует, чтобы каждая из указанных задержек рассматривалась как ветвь состояний схемы. Действительно, следует считать, что первичные реле управляются основными входами (сигналами возбуждения, поступающими на обмотки), а реакцию этих реле рассматривать как y-сигналы. Если же в данный момент времени возможно изменение лишь одного сигнала x, то этот анализ не дает ничего нового, поскольку действие первичных реле заключается только в задержке входных сигналов.

Однако, если одновременно меняется несколько входов, то условие возникновения состязаний, обычно крити-

ческих, имеет место из-за наличия первичных реле. Действие первичных реле увеличивает возможное отклонение от одновременности до величины, сравнимой с величиной задержек соответствующих реле; поэтому инерциальность вторичных реле не может поглотить ошибку. Одно очевидное средство состоит в использовании двух типов реле: относительно быстродействующих первичных реле и медленнодействующих вторичных. Это, конечно, является отступлением от нашего соглашения о правильных схемах (см. раздел 4.3).

Иногда входные воздействия релейно-контактных последовательностных схем вырабатываются другими релейно-контактными схемами и передаются через контактную часть схемы (выход Z схемы, изображенной на рис. 4.21, может, например, использоваться в качестве формирователя входного сигнала для другой схемы). В таких случаях обычно необходимо употреблять первичные реле для получения необходимого множества прямых и обратных контактов схем, реализующих Y- и Z-функции. Однако, если эти схемы могут быть построены так, что для каждой входной переменной достаточно единственного прямого контакта, заземленного одним концом, то можно непосредственно использовать контактные входы и обойтись без первичных реле; например, на рис. 4.21 контакт x_2 в верхнем левом углу схемы является единственным используемым контактом реле x_2, и, следовательно, реле x_2 может быть удалено из схемы, а выключатель, используемый для управления реле x_2, должен быть включен на место контакта x_2. Иногда к такой ситуации можно придти преднамеренно, соответствующим образом кодируя строки таблицы переходов. Но этот подход довольно труден и содержит много нерешенных вопросов.

ЗАМЕЧАНИЯ ПО БИБЛИОГРАФИИ

Идея введения инерциальных задержек и их отдельные реализации, изображенные на рис. 4.2, *б* и 4.3, были предложены Хаффманом ([49] и в неопубликованных заметках). Фридман [19, 20] разработал реализации, изображенные на рис. 4.3 и 4.4. Основы теории комбинационных состязаний, представленные здесь, разработаны Хаффманом [51]. Леммы 4.1—4.3 о состязаниях в двухуров-

невых логических схемах сформулированы Маккласки [73, 74]. Эйхельбергер [15] впервые ввел и рассмотрел состязания, возникающие при изменении сигналов на нескольких входах. Ему также принадлежит метод обнаружения состязаний с помощью троичной логики (см. раздел 4.5.3). Несколько позднее этот метод развивался в работе Йоели и Рино [120]. Определения таких понятий, как правильные реализации, существенные состязания, s-тройки, и результаты, приведенные в разделах 4.3.1 и 4.3.2, относящиеся к синтезу схем без задержек и необходимости введения элементов задержки при наличии существенных состязаний, принадлежат Ангеру [111, 112, 115], который также установил, что достаточно одного элемента задержки для реализации произвольной ОИВ-функции, и разработал схему, приведенную на рис. 4.13. Другие доказательства теоремы 4.7 (о необходимости введения элементов задержки) получены Халлом [40] и Лангдоном [60]. Работа Лангдона содержит, кроме того, обобщение определения существенных состязаний при изменении нескольких входных переменных. Схема, изображенная на рис. 4.14, построена Хаффманом, а материал из раздела 4.3.3, относящийся к использованию элементов задержки на входных линиях, заимствован из работы Климана и Лоуенштюсса [58]. Фридман и Менон [23] построили источник сигналов и показали, как использовать преобразования таблиц Мили в таблицы Мура для построения схем с одной задержкой, допускающих изменения на нескольких входах (раздел 4.3.4). Содержание раздела 4.4, в котором рассматриваются схемные реализации без задержек в случае локализации паразитных задержек на выходах логических элементов, взято у Армстронга, Фридмана и Менона [2]. Другое решение этого вопроса (здесь не рассмотренное, однако весьма интересное) получено Лангдоном [60, 61], по-видимому, несколько позднее, но независимо от предыдущих авторов. Процедура построения таблицы переходов для логических последовательностных схем является обобщением известной работы Хаффмана [49] и выполнена Ангером [111]. Дальнейшее развитие эта процедура получила в работе Лернера [62], выявившего, как и где должны вводиться в схему элементы задержки для предотвращения неправильного функционирования, вызванного наличием

паразитных задержек. Исследование существенных состязаний в релейно-контактных последовательностных схемах и предложение использовать управление шунтированием выполнены Маркусом [65]. Вопросы, связанные с входами релейно-контактных схем, рассмотрены Бжозовски [9], построившим также некоторые методы, не освещенные здесь. Проблемам синтеза релейно-контактных схем посвящены монографии Колдуэла [12] и Кейстера, Ричи и Уошборна [55]. Исследование различных проблем, связанных с состязаниями, выполнено Мюллером [87].

ЗАДАЧИ

4.1 *. В таблице 4.16 описаны четыре элемента задержки. Входные сигналы s_1, s_2, s_3 и s_4 представляют собой

Таблица 4.16

Обозначение задержки	Тип задержки	Величина
$D_{с1}$	совершенная	4
$D_{с2}$	совершенная	6
$D_{и1}$	инерциальная	4
$D_{и2}$	инерциальная	6

отдельные импульсы длительностью 2, 5, 8 и 14 соответственно. Вычислить реакции на каждый из входных сигналов следующих соединений элементов задержки:

(а) $D_{с_1}$ последовательно соединен с $D_{с_2}$;

(б) $D_{с_2}$ последовательно соединен с $D_{и_2}$;

(в) $D_{и_1}$ последовательно соединен с $D_{и_2}$.

Какие другие соединения заданных элементов задержки эквивалентны соединению (в)? Сформулировать общее правило для этих случаев.

4.2 *. Найти реакцию схемы, изображенной на рис. 4.4, если входным сигналом является импульс длительности 5 и (а) $D_1=4$, $D_2=6$; (б) $D_1=6$, $D_2=4$.

4.3. Написать уравнения для свободных от состязаний двухуровневых комбинационных схем, реализующих восьмеричную функцию четырех переменных $F=\Sigma(0, 4, 5, 7, 11, 13, 17)$. Использовать представления в виде суммы и в виде произведения. (Замечание: числа в условии задачи представляют собой состояния в восьмеричной форме. Например, 13 определяет четверку $x_1x_2x_3x_4=1011$.)

4.4. Задана функция из задачи 4.3. (а) Изменить построенное представление в виде суммы так, чтобы в схеме возникло состязание на переходах $4 \longrightarrow 5$ и $11 \longrightarrow 13$. Начертить схему. Ввести в схему элемент совершенной задержки в точку, где с его помощью можно устранить ложный выходной сигнал на переходе $4 \longrightarrow 5$. Можно ли с помощью этого элемента задержки также устранить ложный выходной сигнал на переходе $5 \longrightarrow 4$? (б) Изменить представление в виде произведения так, чтобы в схеме возникло состязание на переходе $2 \longrightarrow 3$.

4.5. Начертить схему, соответствующую выражению

$$F=(\bar{A}+C)(B+\bar{D})+(A\bar{D}+B)(\bar{B}+\bar{C}).$$

Найти все комбинационные состязания, возможные при изменении одной входной переменной, и два таких же состязания, возможных при изменении нескольких входных переменных. Написать простое выражение, соответствующее свободной от состязаний схеме, реализующей ту же функцию.

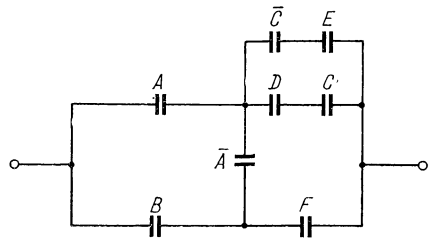


Рис. 4.22.

4.6. На каких переходах (если их несколько) при условии, что допустимо изменение только одной входной переменной, возможны состязания в контактной схеме, изображенной на рис. 4.22?

4.7. Существуют ли двухуровневые И—ИЛИ-схемы, реализующие функции без состязаний при изменении одной входной переменной и не покрывающие каждый простой импликант реализуемой функции? Ответ доказать.

4.8. Построить функцию четырех переменных и такую реализацию этой функции, которая содержит 0-состязания при изменениях одной входной переменной, но свободной от логических состязаний.

4.9. Доказать, что любая функция, имеющая более одного простого импликанта, содержит статическое функциональное состязание.

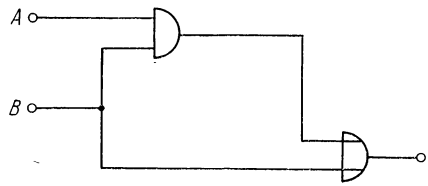


Рис. 4.23.

4.10 *. Показать, что в схеме, изображенной на рис. 4.23, существует динамическое логическое состязание при изменении нескольких входных переменных.

4.11. Выявить все существенные состязания для функции, заданной в таблице 4.17.

Таблица 4.17

	x_1x_2 00	01	11	10
1	***1***	2	3	***1***
2	1	***2***	4	***2***
3	1	***3***	***3***	***3***
4	***4***	3	***4***	2

4.12 *. Определить, правильно ли выбран способ кодирования строк в таблице 4.18 для того, чтобы реализация была свободна от состязаний, считая, что в каждый момент времени может изменяться только одна входная переменная.

Таблица 4.18

	$x_1x_2x_3$ 000	001	011	010	100	y_1	y_2	y_3	y_4
1	***1***,0	2,0	3,0	***1***,0	5,0	0	0	0	0
2	***2***,1	***2***,0	3,0	***2***,0	5,0	0	1	1	0
3	4,0	2,0	***3***,0	***3***,0	5,0	0	1	0	0
4	***4***,0	2,0	3,0	***4***,1	5,0	0	1	0	1
5	***5***,0	***5***,0	3,0	***5***,0	***5***,0	1	0	1	1

4.13. Показать, как таблица 4.19 может быть реализована с использованием блока задержки, изображенного на рис. 4.14. Допустимы изменения нескольких входных переменных.

4.14. Предположим, что последовательностная схема должна быть построена с использованием элементов задержки с таким разбросом значений задержек, что для того, чтобы обеспечить минимальную величину задержки D_m, необходимо взять элементы, значения задержки которых доходят до $(1+E)D_m$. Допустим, что некоторые переходы содержат последовательности, состоящие не более чем из

Таблица 4.19

	x_1x_2 00	01	11	10
1	***1***,0	2,0	***1***,0	***1***,1
2	3,0	***2***,0	3,1	***2***,0
3	***3***,0	2,0	***3***,1	4,1
4	***4***,0	***4***,0	1,0	***4***,1

Таблица 4.20

	x 0	1	y_1	y_2	y_3
1	***1***,0	2,0	0	0	0
2	3,0	***2***,0	0	0	1
3	***3***,0	4,0	0	1	1
4	5,0	***4***,0	0	1	0
5	***5***,0	6,1	1	1	0
6	1,0	***6***,1	1	0	0

n изменений y-состояний. Считая, что D и d — максимальная и минимальная задержка в путях логической схемы, определить, какое время должно проходить между последовательными входными изменениями, чтобы схема функционировала правильно.

4.15. Матрица переходов задана в таблице 4.20. (а) Какое количество задержек только совершенного типа нужно для обеспечения S-правильного функционирования? (б) Пусть цена инерциальной задержки равна

50 центам, а цена совершенной задержки — 40 центов. Как наиболее дешевым образом обеспечить S-правильное функционирование? (в) В каком соотношении могут находиться скорость и задержки логических элементов реализации этой таблицы? (г) Как будет ограничен наш выбор в задаче (б), если желательно избежать состязаний как переходных, так и устойчивых состояний?

Таблица 4.21

	x_1x_2 00	01	11
1	***1***,0	2,0	***1***,0
2	3,1	***2***,0	***2***,0
3	***3***,1	***3***,0	1,0

4.16. Считая, что в схеме паразитные задержки могут содержаться только на выходах логических элементов, синтезировать функцию, заданную в таблице 4.21, не используя элементов задержки.

4.17. Провести анализ схемы, изображенной на рис. 4.18, считая сигналы в точках a, c и d переменными y_1, y_2 и y_3 соответственно. Сравнить результат с таблицей 4.15, *д*.

4.18. Используя метод троичной логики (см. подраздел 4.5.3), провести анализ последовательностной схемы, изображенной на рис. 1.1 (построенной по таблице 1.2), для того чтобы определить последствия изменения x_2 при начальном состоянии системы 1—10.

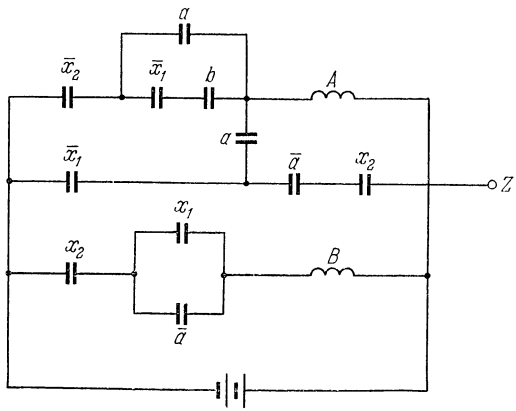


Рис. 4.24.

4.19. Построить таблицу переходов, определяющую поведение схемы, изображенной на рис. 4.24. Начальным состоянием является 00—00.

4.20 *. Доказать, что существование в И—ИЛИ-схеме логического 0-состязания означает наличие элементарного произведения с отрицанием входящих в него букв.

4.21 *. Показать, что функция, заданная выражением $\bar{A}B+B\bar{C}$, может быть реализована И—ИЛИ-схемой, свободной от состязания на переходе 010 ⟶ 110 или на переходе 000 ⟶ 011, но нет таких схем, свободных от обоих этих состязаний. Существует ли схема произвольного вида, реализующая эту функцию без обоих состязаний?

Таблица 4.22

	x_1x_2 00	01	11	10
1	***1***,0	2,0	***1***,0	***1***,0
2	3,1	***2***,0	***2***,0	5,0
3	***3***,1	2,0	***3***,0	***3***,0
4	3,1	***4***,1	***4***,0	5,0
5	***5***,0	***5***,1	4,0	***5***,0

4.22. Построить полную матрицу переходов для реализации без задержек таблицы 4.22. Не использовать внутренних переменных больше, чем это необходимо. Считать, что допустимы изменения одной входной переменной.

4.23+. Каждая из таблиц 4.23, *а* и *б* задает ОИВ-функцию, для которой допустимы изменения нескольких

Таблица 4.23

	x_1x_2 00	01	11	10
1	***1***,0	2,1	***1***,0	***1***,0
2	3,0	***2***,1	1,0	***2***,0
3	***3***,0	***3***,0	1,0	***3***,1

а)

	x_1x_2 00	01	11	10
1	***1***,0	2,0	***1***,0	***1***,0
2	3,0	***2***,0	***2***,1	3,1
3	***3***,0	4,0	2,1	***3***,1
4	***4***,1	***4***,0	1,0	3,1

б)

входных переменных. Построить матрицу переходов для каждой таблицы, свободную от состязаний устойчивых или переходных состояний. Использовать минимальное число внутренних переменных и указать, когда необходимы элементы задержки.

ГЛАВА 5
ОБРАТНЫЕ СВЯЗИ

В этой главе будет обсуждено значение обратных связей в асинхронных последовательностных схемах в различных условиях, физические эффекты, к которым приводит наличие обратных связей, и, наконец, методы синтеза схем с минимальным числом обратных связей. Основные результаты справедливы для схем Хаффмана, хотя часть из них применима и к схемам Мюллера.

5.1. Необходимость введения обратных связей

В этом разделе будет показано, что для большого класса последовательностных переключательных функций положительная обратная связь соответствует каждой неизбыточной внутренней переменной в любой схеме Хаффмана, состоящей из логических элементов и элементов задержки.

Теорема 5.1. *Пусть матрица M является ОИВ-матрицей переходов* *), *свободной от критических состязаний. Тогда для каждой существенной внутренней переменной y_i, используемой при кодировании состояний, существует такое состояние других y-переменных и входных переменных, при котором $Y_i = y_i$.*

Теорема 5.1 эквивалентна утверждению о том, что каждая существенная переменная Y_i не может быть ни независимой от y_i, ни монотонно убывающей функцией y_i.

---

*) Заметим, что единственное y-состояние присваивается каждой строке такой матрицы переходов, а следующее состояние в каждом полном состоянии представляется в матрице номером строки, y-состояние которой определяет Y-значения. Как показано в главе 3, матрица получается из таблицы переходов приписыванием некоторых y-состояний каждой строке с последующим расширением таблицы, при котором каждому y-состоянию отводится отдельная строка.

Иначе говоря, должна существовать некоторая «положительная» обратная связь от y_i к Y_i. Физическая интерпретация теоремы 5.1 заключается в том, что, если в схеме Хаффмана, реализующей ОИВ-функцию, существует такая внутренняя переменная y_j, что каждый путь, ведущий от y_j к Y_j, содержит инвертор (или нечетное число инверторов), то можно сказать, что y_j избыточна.

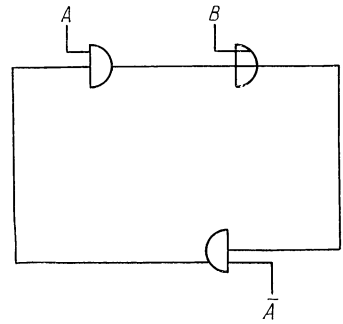


Рис. 5.1. Псевдоконтур.

Важно понять, что указанный выше путь через обратную связь должен действительно проводить сигналы, а не представлять собой лишь топологическую связь в схеме. Например, путь, изображенный на рис. 5.1, является *псевдоконтуром*, поскольку в любой момент либо один, либо другой элемент И заперт. Таким образом, если схему разорвать в любой точке, сигнал на выходе разрыва не будет зависеть от значения сигнала на входе разрыва. Для рассмотренного псевдоконтура, содержащего Y_i и y_i, не выполняется условие теоремы о том, что Y_i есть функция от y_i.

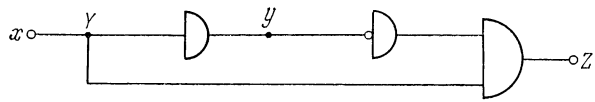


Рис. 5.2. Реализация таблицы 5.1 без обратных связей.

Необходимость по отношению к ОИВ-условию иллюстрируется схемой без обратных связей (рис. 5.2), реализующей функцию, представленную в таблице 5.1. Двукратное изменение выхода $0 \longrightarrow 1 \longrightarrow 0$ вызывается однократным входным изменением $0 \longrightarrow 1$, что противоречит нашему условию. Перейдем теперь к доказательству теоремы.

Доказательство. В каждой клетке матрицы сделаем замену: $Z(s, I) = Z(N(s, I), I)$ для всех s и I. Тогда выход в каждом состоянии окажется равным выходу в

следующем состоянии (или на множестве состояний цикла, представляющих устойчивое состояние таблицы переходов). Такая замена не изменит поведения системы, когда реализуется ОИВ-функция (см. раздел 2.6). Теперь предположим, что Y_j никогда не увеличивается с увеличением y_j и что через r_0 и r_1 обозначены две строки, кодирование которых различается только тем, что $y_j = 0$ в строке r_0 и $y_j = 1$ в строке r_1. Покажем, что r_0 и r_1 эквивалентны по существу и r_1 можно удалить без изменения реализуемой функции и без возникновения критических состязаний. Если удались все такие строки каждой пары, то y_j окажется избыточной, что и доказывается ниже.

Таблица 5.1

Таблица переходов для функции, вырабатывающей выходной импульс

	x = 0	x = 1
1	*1*,0	2,1
2	1,0	*2*,0

1. Для любого заданного входа I существует только четыре способа задания значений Y_j в клетках $r_0 — I$ и $r_1 — I$. Один из них, заключающийся в задании нулевого и единичного значений Y_j в $r_0 — I$ и $r_1 — I$ соответственно, вызвал бы увеличение Y_j при увеличении y_j от строки r_0 к строке r_1 и, следовательно, исключается по предположению. В каждом другом случае y_j является неустойчивым по крайней мере в одном из двух полных состояний. Предположим без потери общности, что y_j неустойчиво ($Y_j = 1$) в состоянии $r_0 — I$. Тогда независимо от того, какие значения приписаны другим Y-функциям в $r_0 — I$, возможно, что после перехода в $r_0 — I$ система может перейти в состояние $r_1 — I$.

2. Однако $N(r_0, I)$ также является таким состоянием, в которое система может перейти из $r_0 — I$; поэтому, если это состояние не эквивалентно состоянию $r_1 — I$, могут возникнуть критические состязания. Поскольку такая возможность исключается в формулировке теоремы, состояния $r_1 — I$ и $N(r_0, I)$ эквивалентны.

3. Тогда $Z(r_1, I) = Z(N(r_0, I), I) = Z(r_0, I)$.

4. Следовательно, r_0 и r_1 имеют одинаковые выходы в каждом столбце.

5. По тем же соображениям, что использованы в п. 2, устойчивые состояния (или множества состояний цикла), достижимые из $r_1 — I$ и $N(r_0, I)$ (и поэтому из $r_0 — I$), также должны быть эквивалентны в предположении, что,

как это обычно бывает в асинхронных схемах, входной сигнал не изменяется прежде, чем установится устойчивое состояние.

6. Тогда, если начальным состоянием является r_0 или r_1, все входные сигналы дают один и тот же выход и ведут в эквивалентные строки. Следовательно, r_0 и r_1 эквивалентны (предполагается, что схема асинхронная — см. раздел 2.6).

7. Вычеркнем из таблицы каждую строку r_1 и y_j из множества y-переменных, так как y_j необходимо только для различения строк r_0 и r_1. Полученная таблица переходов, рассматриваемая как представление асинхронной функции, эквивалентна исходной таблице, поскольку каждое состояние r_0 и r_1 заданной таблицы эквивалентно состоянию r_0 новой таблицы.

8. Предположим, что в новой матрице на некотором переходе $s - I \longrightarrow N(s, I) = d - I$ существует критическое состязание между переменными y_p и y_q (и, возможно, также между другими y-переменными). Тогда соответствующее состязание между переменными y_p и y_q существует и в исходной матрице на переходе $r_0 - I \longrightarrow N(r_0, I)$ (r_0 соответствует s), поскольку мы не изменили кодирования переменных, отличных от y_j. Тогда либо в состязании должно участвовать то же множество переменных, либо переменная y_j также должна состязаться. Каждый элемент переходного множества $T(s, d)$ должен был бы иметь своего «двойника» в состязании в исходной матрице.

9. Однако, в силу того, что по предположению в исходной матрице нет критических состязаний, их не должно быть и в новой матрице. Таким образом, теорема доказана.

Хотя в приведенной теореме мы показали, что кодирование, свободное от критических состязаний, может быть получено из первоначального кодирования удалением y_j, существуют случаи, когда использование y_j полезно. Например, это было бы в случае введения в логику элемента задержки для исключения возможности статического состязания (см. раздел 4.2). При анализе схемы выход этой задержки должен быть отождествлен с внутренней переменной (см. подраздел 4.5.1), встречающейся при описании y_j. Так как по предположению вовсе необязательно, что наличие y_j предотвращает критические состязания, то

можно обойтись без этой переменной (как указано в теореме), если схема перестроена в свободную от состязаний за счет добавления одного или более логических элементов.

Отметим, что согласно приведенному доказательству, если в состояниях $r_0 - I$ и $r_1 - I$ переменной Y_j приписаны соответственно значения 1 и 0 так, что Y_j убывает с возрастанием y_j, в схеме могут, по крайней мере временно, возникнуть колебания между состояниями $r_0 - I$ и $r_1 - I$. Эти колебания обязательно будут временными, если некоторая другая y-переменная является неустойчивой и в $r_0 - I$, и в $r_1 - I$, так что в конце концов после ее изменения система перейдет в некоторое другое состояние. Существование «отрицательной» обратной связи, отвечающее наличию нечетного числа инверторов в некотором пути от y_i к Y_i, всегда создает такие колебания, которые могут являться как желательными, так и нет. Тогда мы видим, что, если схема, реализующая функцию рассматриваемого типа, не имеет избыточных внутренних переменных и в ней невозможны колебания, то каждая переменная Y_i является монотонно возрастающей функцией от y_i. Другими словами, любое неприведенное выражение для Y_i, содержащее только отрицания отдельных букв (как в минимальной форме представления в виде суммы произведений), будет включать по меньшей мере один терм с y_i и не иметь ни одного терма с $\bar{y}_i$.

Заметим, что в отношении колебаний утверждение, обратное доказанному, в общем случае неверно. Функции $Y_1 = y_2$ и $Y_2 = \bar{y}_1$, входящие как подфункции в другие функции, в которых Y_1 и Y_2 являются соответственно возрастающими функциями от y_1 и y_2, могут привести к колебаниям между четырьмя состояниями 00, 01, 11, 10. В разделе 5.6 приведена интересная схема, в которой устойчивые состояния таблицы переходов реализуются как колебания между множествами y-состояний. (Теорема 5.1 верна и для таких случаев.) Следует также подчеркнуть, что хотя могут существовать состояния схемы, между которыми возможны колебания, можно ввести ограничения на начальные состояния схемы или на входные последовательности так, чтобы состояния, участвующие в колебательном процессе, были недостижимы при нормальном функционировании.

5.2. Усиление в последовательностных схемах

Важным следствием теоремы 5.1 обратной связи является необходимость наличия в каждом контуре обратной связи усилителей того или иного типа.

Допустим, что логические переменные наших схем представляются напряжениями. (Аналогичные аргументы можно привести, если некоторые физические параметры представляются током или давлением.) Тогда для представления нулевого и единичного значения логических переменных естественно ввести неперекрывающиеся уровни напряжений; например, в некоторой части схемы $e < 10$ может соответствовать нулевому значению, а $e > 10$ — единичному значению. В других частях схемы могут использоваться иные уровни напряжений.

На рис. 5.3, *а* изображена реализация, удовлетворяющая условиям теоремы 5.1; допустим, что в схеме нет избыточных внутренних переменных. Схема рис. 5.3, *б* является частью предыдущей схемы с выделенной ветвью состояния y_k, причем предполагается, что значения входных переменных и остальных внутренних переменных таковы, что $Y_k = y_k$. Согласно теореме 5.1 такое равенство справедливо, и в этом случае y_k принимает устойчивое значение, равное 0 или 1. На рис. 5.3, *б* пунктирная линия через логическую схему означает, что существует путь распространения сигнала от y_k до Y_k. Тогда по отношению к логической схеме наше условие указывает, что схема остается в статическом состоянии до тех пор, пока остальные переменные не изменятся. Если теперь разорвать цепь в некоторой точке B (см. рис. 5.3, *б*), схему можно рассматривать как статический преобразователь напряжения с передаточной функцией T, связывающей реакцию e_r (рис. 5.3, *в*) справа от разрыва с напряжением e_i сигнала, подаваемого на схему.

Заметим, что в исходной схеме без разрыва при $y_k = 0$ напряжение в точке B должно быть меньше 10, а при $y_k = 1$ это напряжение должно превосходить 20. Так как в этом случае e_i равно e_r, то график функции T должен проходить через точки, в которых $e_i = e_r < 10$ и $e_i = e_r > 20$. При естественном физическом допущении о том, что величина напряжения в точке B конечна, названные

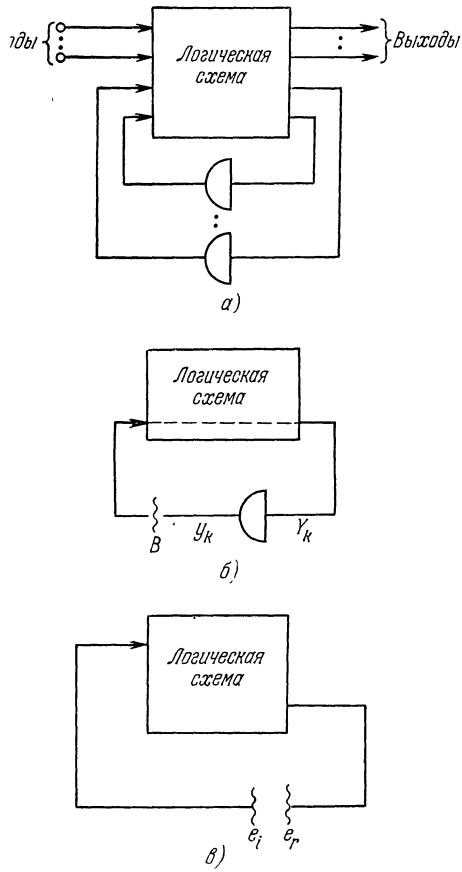


Рис. 5.3. *а*) Модель последовательностной схемы. *б*) Часть схемы а) с выделенной ветвью состояния y_k (значения остальных переменных таковы, что $Y_k = y_k$). *в*) Схема с разорванным контуром, эквивалентная схеме б).

точки должны находиться в устойчивом равновесии. Однако в системе с положительной обратной связью точка на графике $e_r = T(e_i)$ может быть точкой равновесия лишь тогда, когда отношение приращений $\Delta e_r/\Delta e_i$ меньше 1 в некоторой окрестности этой точки.

Учитывая сказанное, построим на рис. 5.4, *а* часть функции T в координатах e_i и e_r с точками равновесия v_0 и v_1. Для того чтобы завершить построение функции T, необходимо определить характер ее изменения между точками p и q. Независимо от того, как доопределена функция T (на рис. 5.4, *б* приведен один из способов доопределения), по виду полученной функции передачи можно сказать, что изменения выходного сигнала превышают изменения входного, когда значения входного сигнала меняются между p и q. Таким образом, в некотором месте контура должно иметь место усиление напряжения. Кроме того, поскольку выходной сигнал логической схемы (см. рис. 5.3, *в*) подводится к ее входу, при наличии усиления по напряжению выходной ток равен входному. Поэтому в контуре происходит не только усиление напряжения, но и усиление мощности.

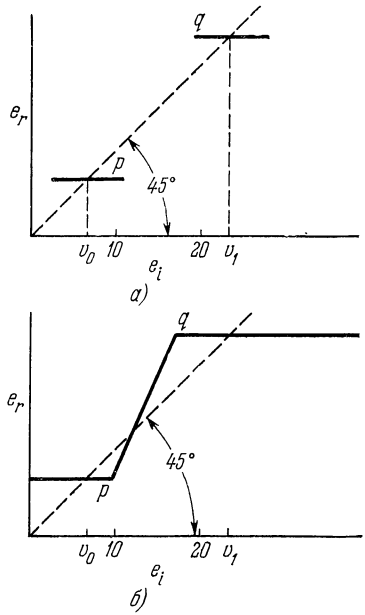


Рис. 5.4. *а*) Часть графика функции T. *б*) Пример доопределения функции T.

Итак, если мы реализуем последовательностную схему с помощью логических блоков, не обеспечивающих усиления по напряжению и мощности, например, на диодных элементах или в отдельных случаях даже на элементах, содержащих транзисторы, мы обязаны ввести в схему усилители так, чтобы каждый контур обратной связи, не

являющийся псевдоконтуром, имел коэффициент усиления по напряжению, превосходящий 1. Очевидным и всегда удобным местом введения усилителей является каждая ветвь состояния. Однако в следующем разделе будет показано, что возможно построение схем, требующих меньше усилителей, если не настаивать на их включении в ветви состояний.

5.3. Минимизация числа обратных связей

Пусть задана структурная схема последовательностной схемы, тогда, если предположить, что в нее обязательно должны быть введены усилители, проблема минимизации числа усилителей сводится к нахождению минимального *множества разрывов обратных связей*, т. е. множества узлов, из которого каждому контуру принадлежит по меньшей мере один элемент. Следует помнить, что псевдоконтуры здесь не рассматриваются.

Таким образом, задача синтеза схем с минимальным числом усилителей соответствует задаче синтеза схем с минимальным индексом обратной связи (см. раздел 1.5 гл. 1).

С другой стороны, синтез схем с минимальным числом обратных связей вызван необходимостью проверки, функционирует ли реализованная схема в соответствии с заданными условиями. Последняя задача может быть существенно упрощена для асинхронных схем, если мы в состоянии временно разорвать все обратные связи и проверить полученную комбинационную схему. Естественно, что число специальных элементов, необходимых для создания разрывов, пропорционально индексу обратной связи.

Докажем, прежде всего, теорему, согласно которой легко определить минимальное значение индекса обратной связи произвольной схемой реализации заданной таблицы переходов.

Теорема 5.2. *Пусть задана произвольная сжатая таблица переходов, содержащая s_i устойчивых состояний в i-м столбце входов. Тогда индекс обратной связи f любой последовательностной схемы, реализующей эту таблицу так, что каждому устойчивому состоянию таблицы соот-*

ветствует устойчивое состояние схемы, удовлетворяет неравенству $f \geqslant \mathrm{E}[\log_2(\max_i) s_i)]$*).

Доказательство. Предположим, что для произвольной последовательностной схемы разрывы должны быть сделаны в каждом из элементов минимального множества разрывов обратных связей, причем входы этих разрывов рассматриваются как дополнительные входные полюсы схемы. Тогда, если новым входам так же, как входам исходной схемы, присвоить фиксированные значения, во всех точках схемы можно вычислить значения сигналов, соответствующие установившемуся состоянию, поскольку схемы, получающиеся после проведения процедуры разрыва обратных связей, содержат конечное число элементов, среди которых могут быть и элементы задержки, но не имеют обратных связей. В частности, можно вычислить компоненты вектора y, определяющего внутреннее состояние схемы. Однако это равносильно тому, что значения входных сигналов и сигналов точек, принадлежащих множеству разрывов, определяют y, если исходная схема находилась в устойчивом состоянии. Если множество точек разрыва содержит f элементов, то любому входному состоянию соответствует 2^f различных y-состояний, так как в указанных точках можно наблюдать не более 2^f различных комбинаций значений сигналов и при заданных входных сигналах каждому f-состоянию может соответствовать не более одного y-состояния. Следовательно, для любого входного состояния существует не более 2^f устойчивых состояний схемы, что и доказывает нашу теорему.

Теперь рассмотрим задачу реализации таблицы переходов в виде схем с минимальным числом обратных связей. Обратимся к рис. 5.3, *а*, на котором приведена блок-схема последовательностной схемы общего вида. Предположим, что существует n внутренних переменных и все 2^n y-состояний реализуются при функционировании схемы. Тогда, поскольку на выходе комбинационной части может появиться любое из 2^n Y-состояний, нельзя так выбрать точки разрыва в схеме, чтобы, с одной стороны, полностью отделить входы от выходов, а с другой — разорвать мень-

*) Как и раньше, $\mathrm{E}[x]$ обозначает целую часть числа. (*Прим. перев.*)

ше чем n связей схемы. Доводы, устанавливающие этот факт, аналогичны использованным в доказательстве теоремы 5.2, поэтому на первый взгляд кажется невозможным включить в множество точек разрыва меньше элементов, чем число внутренних переменных (если только не создать иллюзию успеха введением избыточных внутренних переменных!).

Теперь предположим, что в каждом входном столбце множеством следующих состояний является часть из 2^n y-состояний. В этом случае для каждого входного столбца множество возможных следующих состояний можно закодировать с помощью m двоичных переменных, причем $m < n$. При заданных x и y комбинационная схема L_1 могла бы кодировать следующее состояние в виде компактного m-разрядного кодового вектора и передавать его по m проводам на вторую логическую схему L_2. Если на схему L_2 также воздействует входной сигнал x, то m-разрядный кодовый вектор декодируется схемой L_2 в требуемые Y-сигналы, которые подводятся к входам элементов задержки.

Блок-схема такого устройства приведена на рис. 5.5. Часть блок-схемы, обведенная пунктиром, соответствует логическому блоку, изображенному на рис. 5.3, *а*. Теперь заметим, что, разорвав m q-проводов, соединяющих L_1 и L_2, мы получим схему, в которой все контуры обратной связи разорваны. Таким образом, если ни один входной столбец матрицы переходов не содержит более 2^{n-1} различных следующих состояний, то индекс обратной связи реализующей схемы меньше, чем число внутренних переменных.

После выполнения указанной процедуры разрыва всех обратных связей логическая схема не делится на две совершенно отдельные части, поскольку входной сигнал x воздействует на схему L_2. Как указано в начале нашего исследования, разрыв всех путей в схеме достигается разрывом не менее чем n проводов. Поэтому при использовании изложенного метода, в противоположность обычному методу синтеза с тем же способом кодирования строк, индекс обратной связи сокращается по крайней мере до числа проводов, необходимого для кодирования входных состояний.

Как показано на рис. 5.5, выходные сигналы системы z могут воспроизводиться как схемой L_1, так и схемой L_2,

так как необходимая информация есть на входах обеих схем.

В качестве примера рассмотрим функцию, заданную таблицей 5.2, *а*. Свободное от состязаний кодирование

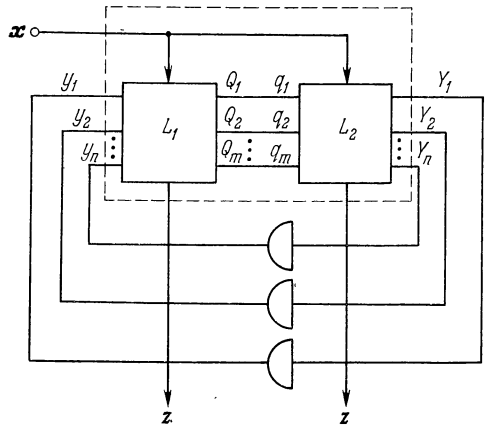


Рис. 5.5. Блок-схема системы с минимальным числом обратных связей.

строк этой таблицы приведено в таблице 5.2, *б*, где строка, отмеченная символом α, используется как промежуточное состояние при переходе $4 \longrightarrow 5$ в столбце 00. Столбец 00 содержит три устойчивых состояния, так что в силу теоремы 5.2 при реализации этой матрицы значение индекса обратной связи не может быть меньше двух. Соответствующий параметр в нашей процедуре равен максимальному числу различных следующих состояний любого столбца матрицы переходов: в данном случае в столбце 00 имеется четыре различных состояния. Поэтому достаточно двух Q-переменных для кодирования всех следующих состояний в каждом столбце; таким образом, применение нашей процедуры приводит к схеме, индекс обратной связи которой равен 2.

Следующий шаг состоит в кодировании следующих состояний каждого столбца с помощью переменных Q_1 и Q_2. В столбце 00 следующими состояниями являются 1, 3, α и 5. Возможен любой способ кодирования этих состояний с помощью Q_1 и Q_2, хотя, вообще говоря, сложность

Таблица 5.2

	x_1x_2 00	01	11
1	***1***,0	2,0	***1***,0
2	3,1	***2***,0	1,0
3	***3***,1	2,0	4,1
4	5,1	2,0	***4***,1
5	***5***,1	2,0	1,0

а) Таблица переходов

	x_1x_2 00	01	11	10	y_1	y_2	y_3
1	***1***,0	2,0	***1***,0	—	0	0	0
2	3,1	***2***,0	1,0	—	0	1	0
3	***3***,1	2,0	4,1	—	0	1	1
4	α,1	2,0	***4***,1	—	1	1	1
5	***5***,1	2,0	1,0	—	0	0	1
α	5,1	—	—	—	1	0	1
	—	—	—	—	1	0	0
	—	—	—	—	1	1	0

б) Матрица переходов

	x_1x_2 00	01	11	10	y_1	y_2	y_3
1	00	—	1—	—	0	0	0
2	01	—	1—	—	0	1	0
3	01	—	0—	—	0	1	1
4	11	—	0—	—	1	1	1
5	10	—	1—	—	0	0	1
α	10	—	—	—	1	0	1
	—	—	—	—	1	0	0
	—	—	—	—	1	1	0

в) Q-матрица

x_1x_2 00	01	11	10	q_1	q_2
000	010	111	—	0	0
011	010	111	—	0	1
101	010	000	—	1	1
001	010	000	—	1	0

г) Y-матрица

x_1x_2 00	01	11	10	q_1	q_2
0	0	1	—	0	0
1	0	1	—	0	1
1	0	0	—	1	1
1	0	0	—	1	0

д) z-матрица

схемы зависит от нашего выбора. Припишем, например, состояниям 1, 3, α и 5 комбинации 00, 01, 11 и 10 соответственно и заменим этими кодовыми словами соответствующие состояния в столбце 00 таблицы 5.2, *в*. Так как при входном сигнале 01 следующим состоянием всегда является 2, то нет нужды передавать какую-либо информацию схеме L_2, кроме входного воздействия. Поэтому значения Q_1 и Q_2 в столбце 01 произвольны. Столбец 11 содержит два различных следующих состояния, так что достаточно задать значения только одной Q-переменной. Для этой цели выберем, например, переменную Q_1. Построенная таким образом таблица 5.2, *в* определяет схему L_1.

Схему L_2 можно теперь определить, указывая для каждого входного состояния внутреннее состояние, соответствующее каждому q-состоянию. (Хотя $q_i = Q_i$ для любого i, что и показано на рис. 5.5, в дальнейшем окажется удобным различать выходы L_1 и соответствующие входы L_2.) При заполнении клеток таблицы 5.2, *г* в столбце 00 для обозначения следующих Y-состояний мы используем Q-состояния 00, 01, 11 и 10 в виде 000, 011, 101 и 001 соответственно, определяя $Y_1Y_2Y_3$ как функцию от $x_1x_2q_1q_2$. Равенство всех элементов второго столбца указывает, что следующим состоянием является состояние 2, а в третий столбец занесены кодовые комбинации строк 4 или 1 в зависимости от того, равно значение q_1 0 или 1.

Выход z может быть задан в терминах y-переменных непосредственно по матрице переходов, а если выход является функцией следующего состояния, как в нашем случае, удобно определить z через q-переменные. (Более подробно о выходных схемах будет сказано позднее). Этот выбор является последним в нашем примере, и функция z, заданная таблицей 5.2, *д*, может быть реализована схемой L_2. Заметим, что выход z оказался равным Y_3.

Стандартным приемом по таблице 5.2, *в* и *г* получим следующие выражения:

$$Q_1 = \bar{y}_2y_3 + \bar{x}_1y_1 + x_1\bar{y}_3, \tag{5.1}$$

$$Q_2 = y_2, \tag{5.2}$$

$$Y_1 = \bar{x}_2q_1q_2 + x_1\bar{q}_1, \tag{5.3}$$

$$Y_2 = \bar{q}_1q_2 + \bar{x}_1x_2 + x_1\bar{q}_1, \tag{5.4}$$

$$Y_3 = z = \bar{x}_2q_2 + \bar{x}_2q_1 + x_1\bar{q}_1. \tag{5.5}$$

Полученная схема изображена на рис. 5.6. Из рисунка видно, что разрыв проводов, отмеченных символами Q_1 и Q_2, приводит к разрыву всех обратных связей.

В нашем примере число необходимых для реализации логических элементов может быть уменьшено примерно

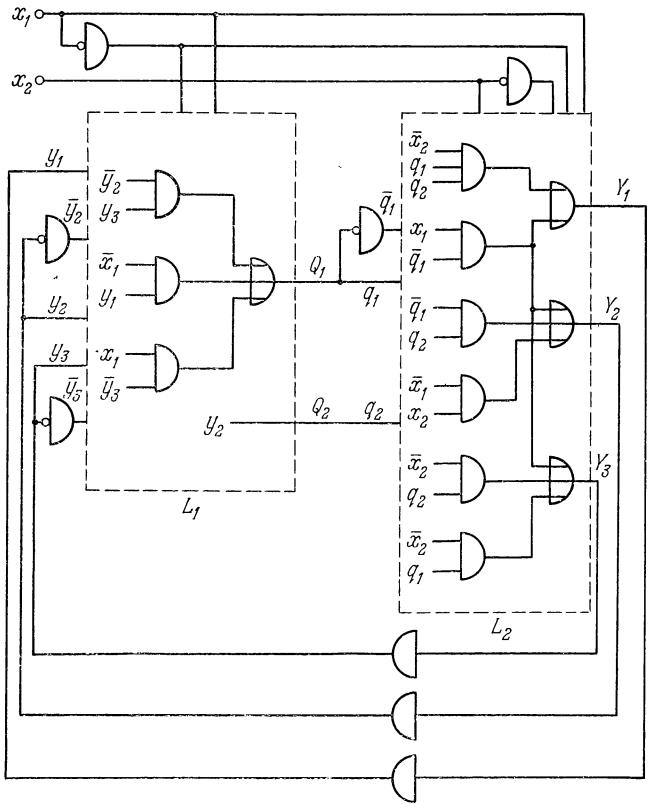


Рис. 5.6. Логическая схема, соответствующая таблице 5.2.

на треть, если ту же матрицу переходов реализовать обычным образом в виде двухуровневой логической схемы с индексом обратной связи, равным 3. Однако во многих случаях используемый здесь процесс «разложения» приводит к более простой логической схеме, чем стандартная

процедура синтеза. Естественно, что обычные погрешности многоуровневых схем, а именно, задержки и ухудшение сигналов, должны быть приняты во внимание.

Очевидно, что при допустимости одновременного изменения нескольких входных сигналов возможны кратковременные ошибочные Y-сигналы, однако следует ожидать их появления даже в тех случаях, когда в каждый момент времени допустимы изменения лишь одной входной переменной или когда схемы L_1 и L_2 свободны от состязаний. Это явление возникает из-за того, что даже при изменении одной x-переменной могут последовательно с малым интервалом времени произойти два или более изменения сигналов на входе схемы L_2. Одним из них является x-изменение, а другим q-изменение, вызванное x-изменением на входе схемы L_1. В таком случае, как указано в разделе 4.2, невозможно избежать комбинационных состязаний только за счет того или иного выбора структуры схемы. Таким образом, платой за построение свободных от состязаний схем возбуждения Q- и Y-переменных является обязательное применение инерциальных y-задержек, которым свойственна фильтрация отдельных коротких импульсов.

Воспроизведение z-сигнала с помощью схемы L_1 необходимо выполнять так, чтобы избежать состязаний, вызываемых изменением нескольких входных переменных схемы L_2. Если допустимы изменения нескольких входных переменных, то таблицу переходов следует преобразовать к форме Мура (см. раздел 4.3). Реализации МИВ- и НИВ-функций будут рассмотрены в разделе 5.5.

Следующая процедура представляет собой алгоритм синтеза схем, удовлетворяющих условиям теоремы 5.2, для важного случая реализации ОИВ-функций.

Процедура 5.1. *Синтез схем Хаффмана с минимальным индексом обратной связи, реализующих ОИВ-функции.*

1. Закодировать строки таблицы переходов, используя любое ООТП-кодирование (например, кодирование по Лю или Трейси, рассмотренное в подразделе 3.3.2). Особенность этих способов кодирования такова, что при появлении r_i в качестве следующего состояния в некотором столбце матрицы переходов в этом столбце и строке r_i полное состояние устойчиво.

2. В полученной матрице переходов элементы следующих состояний каждого столбца закодировать с помощью множества Q-переменных, мощность которого равна нижней границе индекса обратной связи, определяемой теоремой 5.2 ($\mathrm{E}[\log_2 S_{\max}]$, где $S_{\max}$ равно максимальному числу устойчивых состояний, содержащихся в отдельных столбцах матрицы). Как и в предыдущем примере, синтезировать логическую схему L_1, реализующую Q-переменные.

3. Используя кодирование, обратное принятому в п. 2, построить схему L_2, воспроизводящую Y-переменные.

4. Логическую схему, воспроизводящую выходные z-сигналы, ввести в схему L_1.

Большая свобода выбора вариантов в приведенной процедуре заключается в выборе способа кодирования строк и кодирования Q-переменных, что позволяет конструктору оптимизировать другие параметры, например, сложность схемы. Хотя в общем случае, как указано выше, в каждой ветви состояния необходимо использовать инерциальные задержки, иногда в некоторые ветви состояний можно вводить совершенные задержки или вообще обходиться без задержек. Этот аспект проблемы еще не полностью изучен. Конечно, для отдельных физических схем существующее распределение паразитных задержек может быть таким, что позволит обойтись без некоторых, а то и вообще без элементов задержки.

В следующем разделе процедура 5.1 конкретизируется для построения схем с минимальным числом обратных связей, чтобы принять во внимание число необходимых инверторов.

5.4. Схемы с минимальным числом транзисторов

Не так много времени прошло с тех пор, когда для построения элементов И и ИЛИ начали применяться полупроводниковые диоды, а для построения инверторов или усилителей — транзисторы. При прежней технологии, поскольку транзисторы стоили гораздо дороже диодов, было весьма желательно при построении схем использовать относительно малое число инверторов и усилителей. В этом разделе мы усовершенствуем процедуру синтеза, сформулированную в разделе 5.3, для минимизации не только

числа необходимых усилителей, но и числа инверторов. Таким образом, если используются прямые и инверсные выходы и применяется диодно-транзисторная логика, наша задача состоит в сокращении числа требуемых для синтеза транзисторов при условии, что в общем случае

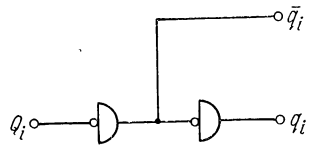


Рис. 5.7. Схема усилителя-инвертора на двух транзисторах.

стоимость затрачиваемых диодов не должна резко возрасти.

Развитие технологии производства элементов резко сократило область, в которой применение этого метода дает значительную экономию. Транзисторные И — НЕ и ИЛИ — НЕ элементы вошли в число используемых дискретных компонент, а при развивающейся в настоящее время технологии производства интегральных схем диоды ненамного дешевле, транзисторов. Однако всегда существует вероятность того, что дальнейшее развитие искусства создания компонент может снова потребовать проведения такого рода оптимизации. Кроме того, метод достаточно прост и красив, и нет смысла отказываться от его изложения, поскольку он представляет некоторый интерес.

Q-ветви изображенной на рис. 5.5 синтезируемой схемы содержат усилители. Предполагается, что усилитель состоит из двух последовательно соединенных инверторов, каждый из которых построен на одном транзисторе, причем выход первого инвертора доступен. Следовательно, как показано на рис. 5.7, обе переменные q_i и $\bar{q}_i$ могут использоваться как входы схемы L_2. Кроме того, предполагается, что каждая переменная x_i может использоваться вместе со своим отрицанием. Допустим, что задержки построены на пассивных элементах, например, на емкостях и сопротивлениях. Ясно, что схема L_2, все входы которой парные (прямые и инверсные), может быть полностью реализована на элементах И и ИЛИ, построенных

на диодах. Существо метода заключается в выборе такого способа кодирования строк и Q-переменных, чтобы каждая Q-переменная была монотонно возрастающей функцией y-переменных. При этом можно избежать инвертирования, если схема L_1 собрана на элементах И и ИЛИ. Для уменьшения числа требуемых дополнительных инверторов выходы всей схемы воспроизводятся схемой L_2. Если из-за возможности изменения нескольких входных переменных на выходе схемы L_2 существуют переходные состязания, то на соответствующих z-полюсах должны быть включены инерциальные задержки. Задержки могут быть простыми RC-схемами (см. рис. 4.2, *б*), если не требуется большой крутизны переднего и заднего фронтов z-сигнала. В ином случае необходимо использовать более сложные элементы, один из которых изображен на рис. 4.4.

Кодирование строк ведется по методу Лю, описанному в подразделе 3.3.2. Поглощения множеств при этом не проводится, так что в каждом столбце каждому устойчивому состоянию приписывается отдельное подсостояние.

Таблица 5.3

Укороченная матрица переходов, определяющая модифицированное кодирование по Лю для таблицы 5.2, *а*

	x_1x_2 00	01	11	y_1	y_2	y_3
1	***1***,0	2,0	***1***,0	0	0	0
2	3,1	***2***,0	1,0	0	1	0
3	***3***,1	2,0	4,1	0	1	1
4	5,1	2,0	***4***,1	1	0	1
5	***5***,1	2,0	1,0	1	0	0

Процедура 5.2. *Синтез схем, реализующих ОИВ-функции, с минимальным числом транзисторов.*

(Допустимы одновременные изменения нескольких входных переменных.) Таблица 5.2, *а* будет использоваться здесь в качестве примера. Прежде всего применим модифицированное кодирование по Лю для построения матрицы переходов, приведенной в таблице 5.3 (в этой таблице столбец 10 и промежуточные y-состояния отброшены). Значения переменных y_1 и y_2 выбраны в соответ-

ствии с содержимым столбца 00, а значения y_3 выбраны по столбцу 11. Поскольку в столбце 01 существует только одно устойчивое состояние, этот столбец не требует y-переменных.

Заметим теперь, что основное свойство модифицированного кодирования по Лю заключается в том, что для заданного входного воздействия I_j y-переменные, связанные с I_j, остаются неизменными, а все остальные y-переменные принимают значения, определяемые значениями неизменившихся переменных. Например, в таблице 5.3, если при начальном состоянии системы 4—11 значения x_1 и x_2 изменятся, то y_1 и y_2 сохранят свои значения 1 и 0, а для y_3 допустимо значение 0, которое принимает y_3 в устойчивом состоянии столбца 00 при $y_1y_2 = 10$. Следовательно, для каждого входного состояния пути обратных связей замыкаются через связанные с ним множества y-переменных, а остальные y-переменные не принадлежат никаким контурам. Поэтому мы можем приписать Q-ветви, содержащие усилители, соответствующим y-переменным в зависимости от входного состояния, и схема L_2, входами которой являются q- и x-переменные, будет воспроизводить все Y-сигналы.

В нашем примере мы припишем переменные Q_1 и Q_2 переменным y_1 и y_2 в столбце 00 и переменную Q_1 — переменной y_3 в столбце 11. Для воспроизведения всех значений Y-переменных в столбце 01 достаточно только входного состояния. Таким образом, уравнения для Q_i примут следующий вид:

$$Q_1 = \bar{x}_1\bar{x}_2y_1 + x_1x_2y_3,$$
$$Q_2 = \bar{x}_1\bar{x}_2y_2.$$

Легко видеть, что схема L_1 является простым соединением y-полюсов с Q-полюсами, управляемым входным состоянием. Это означает, что никогда нет необходимости в получении отрицаний y-переменных.

Чтобы обеспечить «блокировку» переменных y_1 и y_2 при входном состоянии 00, переменные Y_1 и Y_2 устанавливаются в значения, равные соответственно значениям q_1 и q_2 при данном состоянии. Аналогичным образом, блокировка y_3 при входном состоянии 11 достигается установкой значения Y_3, равного q_1 при $x_1x_2=11$. При вход-

ном воздействии 00 Y_3 становится функцией от y_1 и y_2, значения которых передаются в схему L_2 через полюсы q_1 и q_2. Точно так же при $x_1x_2=11$ значения Y_1 и Y_2 определяются значением q_1, равным значению y_3. При $x_1x_2=01$ знания значений q-переменных не требуется для вычисления значений Y-переменных. Поэтому можно записать следующие выражения для Y-переменных:

$$Y_1 = \bar{x}_1\bar{x}_2q_1+x_1x_2q_1,$$
$$Y_2 = \bar{x}_1\bar{x}_2q_2+x_1\bar{x}_2,$$
$$Y_3=\bar{x}_1\bar{x}_2\bar{q}_1q_2+x_1x_2q_1.$$

Учитывая, что входное состояние 10 запрещено, можно определить, имеет ли место входное воздействие 00, наблюдая только, выполняется ли равенство $x_2=0$. Аналогичным образом входное состояние равно 11 тогда и только тогда, когда $x_2=1$. Отсюда следует упрощение приведенных выше уравнений:

$$Q_1 = \bar{x}_2y_1+x_1y_3, \quad Q_2 = \bar{x}_2y_2,$$
$$Y_1 = \bar{x}_2q_1 + x_1q_1, \quad Y_2 = \bar{x}_1q_2 + \bar{x}_1x_2, \quad Y_3 = \bar{x}_2\bar{q}_1q_2 + x_1q_1.$$

По таблице 5.3 можно вычислить z:

$$z=\bar{x}_2q_1+\bar{x}_2y_2+x_1y_3=Q_1+Q_2=q_1+q_2.$$

Соответствующая схема изображена на рис. 5.8. Так как только четыре транзистора необходимо для реализации четырех инверторов и четыре транзистора для реализации двух усилителей, без включения которых в схему обойтись невозможно *), то построенная схема является реализацией с минимальным числом транзисторов.

Хотя представленная здесь процедура приводит к построению схем с минимальным числом транзисторов, иногда можно создать другие реализации с минимальным числом транзисторов для тех же функций, несколько сокращая число внутренних переменных; таким образом,

*) Общепринято, что необходимо два транзистора для построения усилителя, не инвертирующего сигнал, с коэффициентом усиления по напряжению, большим 1, и коэффициентом усиления по току, по крайней мере равным 1.

существует возможность дальнейшего упрощения схемы и уменьшения числа элементов задержки. До настоящего времени не предложена процедура, обеспечивающая синтез схем с минимальным числом транзисторов при минимальном числе y-переменных, хотя известен метод,

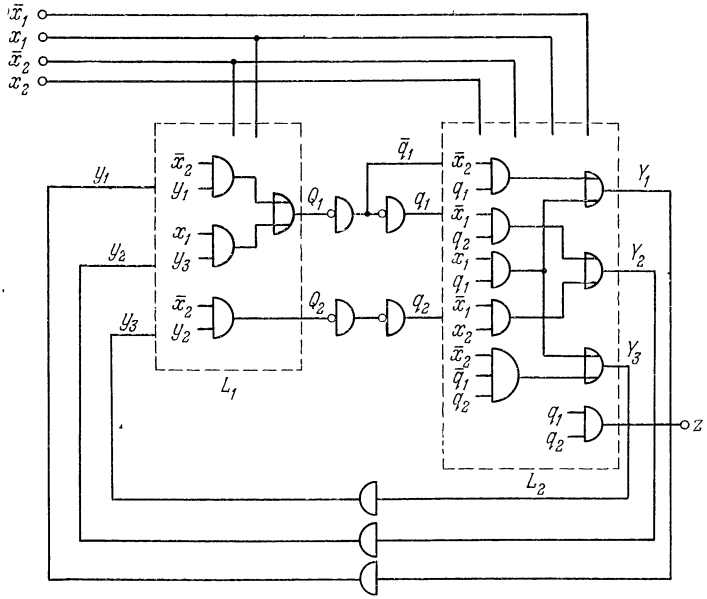


Рис. 5.8. Схема с минимальным числом транзисторов

приводящий иногда к цели за счет исследования тех подмножеств y-переменных, выбираемых описанным выше методом, которые удовлетворяют всем условиям Трейси (см. подраздел 3.3.2) для таблицы переходов.

Вместо подробного описания такого метода мы ограничимся примером. Читатель может проверить, что для кодирования таблицы 5.4, *а* в соответствии с приведенной здесь процедурой синтеза необходимо четыре y-переменных. Однако используя показанное справа от таблицы 5.4, *а* кодирование строк и представленное в таблице 5.4, *б* кодирование Q-переменных, получаем следующие выражения:

Таблица 5.4

	I_1	I_2	I_3	I_4	y_1 y_2 y_3
1	***1***,0	4,0	***1***,0	2,0	0 0 0
2	1,0	***2***,0	1,0	***2***,0	0 1 0
3	***3***,1	4,0	5,0	6,0	1 0 1
4	3,1	***4***,0	1,0	6,0	1 0 0
5	3,1	2,0	***5***,0	6,0	1 1 1
6	3,1	2,0	***6***,1	***6***,0	1 1 0

а) Таблица переходов и вариант кодирования строк, приводящий к реализации с минимальным числом транзисторов

	I_1	I_2	I_3	I_4	y_1 y_2 y_3
1	0—	0—	00	0—	0 0 0
	—	—	—	—	0 0 1
	—	1—	—	—	0 1 1
2	0—	1—	00	0—	0 1 0
6	1—	1—	10	1—	1 1 0
5	1—	1—	—1	1—	1 1 1
3	1—	0—	—1	1—	1 0 1
4	1—	0—	00	1—	1 0 0

б) Матрица переходов для переменных Q_1 и Q_2

$$Q_1 = I_1y_1 + I_2y_2 + I_3y_1y_2 + I_4y_1, \quad Q_2 = y_3,$$

$$Y_1 = I_1q_1 + I_2\bar{q}_1 + I_3q_1 + I_3q_2 + I_4q_1,$$

$$Y_2 = I_2q_1 + I_3q_1 + I_3q_2 + I_4, \quad Y_3 = I_1q_1 + I_3q_2.$$

Поскольку ни одна из y-переменных не входит в найденные выражения с отрицанием, эти уравнения можно использовать для синтеза схем L_1 и L_2, причем других инверторов, кроме входящих в два необходимых усилителя, не требуется.

Процедура 5.2 имеет следующие преимущества:

1. Число транзисторов и индекс обратной связи минимальны.

2. Стоимость построенной схемы в смысле числа используемых логических элементов невелика.

3. Процедура синтеза проста.

4. Быстродействие схемы относительно велико, так как используется ОТП-кодирование.

Однако на практике эти преимущества нелегко реализовать (допуская, естественно, что используется соответствующая технология) из-за того, что синтезируемая схема является обычно четырехуровневой. Это свойство схемы не только снижает быстродействие схемы (п. 4), но может затруднить управление схемой с помощью одной ступени усиления.

Изложенный метод применим только в ОИВ-функциям, и в настоящее время не известно общего метода синтеза схем с минимальным числом транзисторов при ослаблении ограничений, наложенных на выходные последовательности. Однако в следующем разделе мы опишем метод синтеза схем с минимальным числом обратных связей для МИВ- и НИВ-функций.

5.5. Синтез схем с минимальным числом обратных связей, реализующих МИВ- и НИВ-функции

Если допустимы одновременные изменения нескольких входных переменных, таблицу переходов прежде всего следует привести к форме Мура, как показано в разделе 4.3. Рассмотрим теперь МИВ-функции. Покажем, что для таких функций достижима оценка, найденная в теореме 5.2.

Основная идея заключается в выборе двух множеств y-переменных: одно множество (y_S-переменные) для определения следующего устойчивого состояния, а другое (y_T-переменные) — для определения промежуточных (неустойчивых) состояний. Управление y_S-переменными очень похоже на управление y-переменными из процедуры 5.2 и требует введения контуров обратной связи, в то время как y_T-переменные реализуются цепочками задержек, не включенными ни в один контур обратной связи. Теперь полезно обобщить определение множества предшественников, термин, введенный в разделе 3.2 при изучении ОИВ-функций. Применительно к МИВ-функциям множеством предшественников D_{qi} назовем множество внутренних состояний p таких, что из начального состояния p под

действием входного сигнала I_i система в конце концов перейдет в устойчивое состояние q (D_{qi} часто называют *множеством предшественников относительно входа* I_i). В таблице 5.5, которую мы будем использовать ниже в качестве примера, $D_{62}=\{1, 5, 6, 8\}$. Представим, наконец, следующую процедуру.

Процедура 5.3. *Синтез схем, реализующих МИВ-функции, с минимальным числом обратных связей.*

1. Как и в процедуре 5.2, приписать каждому столбцу I_i множество y_S-переменных путем одинакового кодирования каждого множества предшественников относительно входа I_i. В нашем примере значение y_{S_1} различается для тех состояний, из которых система при входе I_i переходит в состояние 1, и для тех состояний, из которых система при том же воздействии переходит в состояние 8. Аналогичным образом значения y_{S_2} и y_{S_3} различны для трех множеств предшественников относительно входа I_2, а y_{S_4} и y_{S_5} используются для кодирования множеств предшественников относительно входа I_3.

2. Пусть в некотором столбце последовательность состояний p_1 является последовательностью максимальной длины, переводящей систему в устойчивое состояние s_s. Если эта последовательность содержит k состояний, то выбрать для кодирования k y_T-переменных, так называемых p_1-переменных. Для последовательности i одинаковых значений α введем обозначение α^i и определим значения p_1-переменных как $0^{k-i}1^i$ для состояний из p_1, находящихся на расстоянии i от s_s. (В нашем примере в столбце I_1 выберем $s_s=1$ и $p_1=4\longrightarrow 3\longrightarrow 2\longrightarrow 1$. Тогда $k=3$, и p_1-переменными являются y_{T_1}, y_{T_2} и y_{T_3}.)

Предположим, что некоторая другая последовательность p_{11} сливается с p_1 на состоянии $s_m\neq s_s$ (в нашем примере такой последовательностью является $p_{11}=6\longrightarrow 5\longrightarrow 2\longrightarrow 1$, сливающаяся с p_1 на $s_m=2$). Тогда членам из p_{11} приписать p_1-переменные с теми же значениями, которые приписаны состояниям из p_1, равноотстоящим от s_m. Для различения соответствующих состояний p_1 и p_{11} вводится новое множество y_T-переменных мощностью t, равной длине оканчивающейся в s_m подпоследовательности последовательности p_{11}. p_{11}-переменным придаются значения $0^{t-j}1^j$ для состояний из p_{11}, отстоящих от s_m на расстояние j, и значение 0^t всем состояниям из

Таблица 5.5

Пример для синтеза МИВ-функции

	I_1	I_2	I_3	I_1 y_{S_1}	I_2 $y_{S_2}y_{S_3}$	I_3 $y_{S_1}y_{S_5}$	I_1 $y_{T_1}y_{T_2}y_{T_3}y_{T_4}y_{T_5}$	I_2 $y_{T_6}y_{T_7}y_{T_8}$
1	***1***,00	5,00	***1***,00	0	0 0	0 0	0 0 0 0 0	1 1 0
2	1,10	3,00	***2***,01	0	0 1	0 1	0 0 1 0 0	0 0 1
3	2,00	***3***,01	7,00	0	0 1	1 1	0 1 1 0 0	0 0 0
4	3,11	***4***,10	2,01	0	1 0	0 1	1 1 1 0 0	0 0 0
5	2,01	6,01	***5***,11	0	0 0	1 0	0 1 1 0 1	0 1 0
6	5,11	***6***,11	7,00	0	0 0	1 1	1 1 1 1 1	0 0 0
7	8,00	4,00	***7***,00	1	1 0	1 1	0 0 1 0 0	0 0 1
8	***8***,01	6,10	7,00	1	0 0	1 1	0 0 0 0 0	0 0 1

p_1 (В нашем примере дополнительными переменными являются y_{T_4} и y_{T_5}.)

Если на состоянии s_m сливаются другие последовательности, то для каждой из них вводится дополнительное множество y_T-переменных, причем значения этих переменных для состояний всех остальных последовательностей равны 0. Если какие-либо последовательности сливаются с p_{11} на состояниях, из которых достижимо s_m, то эти последовательности рассматриваются аналогично тому, как рассматривалась последовательность p_{11} по отношению к пути p_1. Последовательности, пересекающие p_1 только на s_s, рассматриваются отдельно с введением новых множеств y_T-переменных. Переменным, связанным с p_1 и ветвями p_1, придаются нулевые значения для всех состояний новых последовательностей и, наоборот, новые переменные полагаются равными 0 для всех состояний из p_1 и из ветвей p_1.

Поскольку в данном столбце q-состояния различны для разных множеств предшественников, то, если это необходимо, для последовательностей с различными множествами предшественников могут использоваться одни и те же y_T-переменные в различных комбинациях. (Например, в столбце I_1 переменная y_{T_3} используется также для кодирования состояний последовательности $7 \longrightarrow 8$.) В общем случае необходимо использовать различные y_T-переменные для каждого входного воздействия. (В нашем примере $y_{T_6} y_{T_7} y_{T_8}$ использованы для входа I_2, и поскольку в столбце I_3 не происходит никаких выходных изменений ни в одном множестве предшественников, для этого входа вообще нет нужды вводить y_T-переменные.)

3. Как и в процедуре 5.2, y_S-переменные сопоставляются Q-переменным и вычисляются Y_S и Q-функции для реализации с минимальным числом обратных связей. Поэтому в нашем примере необходимо ввести две Q-переменные, чтобы передать соответствующие y_S-переменные в каждом столбце, а именно y_{S_1} в I_1, y_{S_2} и y_{S_3} в I_2, y_{S_4} и y_{S_5} в I_3. В результате получаются следующие выражения:

$$Q_1 = I_1 y_{S_1} + I_2 y_{S_2} + I_3 y_{S_4},$$
$$Q_2 = I_2 y_{S_3} + I_3 y_{S_5}.$$

Тогда контуры обратной связи, блокируемые устойчивыми переменными, замыкаются, и y-переменные в тех

столбцах, где они не являются блокирующими, приписываются значениям следующих состояний, как функции устойчивых переменных (которые передаются с помощью q-переменных). Тогда

$$Y_{S_1} = I_1 q_1 + I_3 q_1 q_2,$$
$$Y_{S_2} = I_2 q_1 + I_3 q_1 q_2,$$
$$Y_{S_3} = I_2 q_2 + I_3 \bar{q}_1 q_2,$$
$$Y_{S_4} = I_1 q_1 + I_2 \bar{q}_1 + I_3 q_1,$$
$$Y_{S_5} = I_1 q_1 + I_2 + I_3 q_2.$$

4. Вычислить Y_T-функции. Легко видеть, что каждая переменная Y_{T_i} не зависит от y_{T_j}, где $j \geqslant 1$, так что не существует обратных связей, содержащих y_T-переменные. В столбце, воспроизводящем Y_{T_i}, Y_{Ti} является функцией от q-переменных (переводящих систему в устойчивые y_S-состояния) и, возможно, от $y_{T_{i-1}}$. В других столбцах Y_{Ti} является функцией лишь q-переменных. Поэтому для таблицы 5.5 получаем, что

$$Y_{T_1} = I_2 q_2,$$
$$Y_{T_2} = I_1 y_{T_1} + I_2 + I_3 q_1 \bar{q}_2,$$
$$Y_{T_3} = I_1 y_{T_2} + I_2 + I_3 q_1 + I_3 q_2,$$
$$Y_{T_4} = I_2 \bar{q}_1 \bar{q}_2,$$
$$Y_{T_5} = I_1 y_{T_4} + I_2 \bar{q}_1 \bar{q}_2 + I_3 q_1 \bar{q}_2,$$
$$Y_{T_6} = I_1 \bar{q}_1 + I_3 \bar{q}_1 \bar{q}_2,$$
$$Y_{T_7} = I_1 \bar{q}_1 + I_2 y_{T_6} + I_3 \bar{q}_2,$$
$$Y_{T_8} = I_1 q_1 + I_3 q_2.$$

Заметим, что в приведенных уравнениях с отрицаниями появляются только символы q_i, и этот факт имеет место в общем случае для данного метода.

5. Выходные сигналы должны быть определены в терминах входных воздействий и соответствующих y_S- и y_T переменных. q-переменные не могут входить в выражения для Z_i, так как их использование может привести к возникновению кратковременных ошибочных сигналов.

В нашем примере

$$Z_1 = I_1 y_{T_1} + I_1 \bar{y}_{S_1} \bar{y}_{T_2} y_{T_3} + I_2 \bar{y}_{S_3} \bar{y}_{T_7} \bar{y}_{T_8} + \\ + I_2 \bar{y}_{S_2} \bar{y}_{S_3} y_{T_8} + I_3 y_{S_4} \bar{y}_{S_5},$$

$$Z_2 = I_1 y_{T_1} + I_1 y_{T_5} + I_1 y_{S_1} \bar{y}_{T_3} + I_2 \bar{y}_{T_6} y_{T_7} + \\ + I_2 \bar{y}_{S_2} \bar{y}_{T_7} \bar{y}_{T_8} + I_3 \bar{y}_{S_4} y_{S_5} + I_3 y_{S_4} \bar{y}_{S_5}.$$

Заметим, что Z-выражения могут содержать отрицания y_S и y_T-переменных.

В приведенной процедуре не предпринимается особых попыток сократить число используемых внутренних переменных, и весьма вероятно, что даже в нашем примере существует решение с меньшим числом y_T-переменных. Индекс обратной связи полученной схемы равен $\mathrm{E}[\log_2 \max_i s_i]$, как определено в теореме 5.2.

Применение инерциальных задержек необходимо и для y_S-, и для y_T-переменных, поскольку на входах схем, воспроизводящих Y_S- и Y_T-переменные, возможны изменения нескольких входных переменных, т. е. x-переменных и q-переменных. Как и в случае, рассмотренном в разделе 5.4, для получения y_S-переменных пригодны RC-задержки. Однако в силу того, что на схемы, реализующие Y_T- и Z-функции, воздействуют изменяющиеся y_T-сигналы, в качестве инерциальных задержек желательно использовать приборы, выходные сигналы которых имеют более крутые фронты. По-видимому, схемы, изображенные на рис. 4.4, удовлетворяют этому условию. И все-таки сомнительно, что при последовательном включении достаточно большого числа таких приборов без введения промежуточных усилителей можно получить удовлетворительные результаты.

Теперь перейдем к синтезу схем с минимальным числом обратных связей, реализующих НИВ-функции. Для таких функций характерно, что при определенных условиях выходные сигналы циклически изменяются, принимая значения из некоторой последовательности все время, пока входное состояние остается неизменным. (См., например, табл. 1.7, *б* и *в*). Это означает, что внутренее состояние системы также изменяется.

Предположим, что система циклически проходит последовательность состояний $s_1, s_2, \ldots, s_n$ и что y-перемен-

ными, изменяющимися во время этого процесса, являются $y_1, y_2, \ldots, y_m$. Если одна из функций Y_i $(1 \leqslant i \leqslant m)$ не зависит ни от одного из y-изменений, то следует допустить, что переменная y_i установится в конце концов в некоторое постоянное значение в противовес предположению о том, что этой установки не происходит. Следовательно, каждая переменная y_i $(1 \leqslant i \leqslant m)$ должна принадлежать контуру обратной связи, содержащему по меньшей мере одноэлементное множество y-переменных (возможно, только y_i). Основой для процедуры синтеза является простой класс схем с индексом обратной связи, равным 1, реализующих циклы с произвольным периодом без критических состязаний.

Рассмотрим первый случай, когда период равен четному числу, а именно, $n=2k$. Схема, изображенная на рис. 5.9

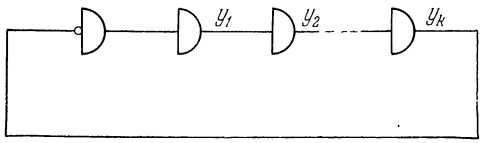


Рис. 5.9. Реализация цикла с периодом $2k$.

и состоящая из k элементов задержки, последовательно соединенных с инвертором, при подходящем начальном состоянии (например, $y_1=y_2=\ldots=y_k=0$) циклически проходит $2k$ состояний, причем в каждый момент времени изменяется только одна переменная. Например, если $k=3$, то последовательность будет иметь вид: 000, 100, 110, 111, 011, 001, 000, ...

При нечетной длине цикла простейшая процедура, сохраняющая периодичность, заключается в использовании n задержек и приписывании пары прямого и инверсного y-состояний каждому состоянию системы. Например, при $n=3$ состоянием s_1, s_2 и s_3 можно приписать кодовые комбинации (000, 111), (100, 011) и (110, 001) соответственно

Контур должен содержать усилитель для предотвращения затухания сигнала. Те задержки цепочки, на которые воздействуют выходные сигналы предшествующих задержек, могут быть выполнены на элементах задержки совершенного типа. Как станет очевидно ниже, по мень-

шей мере один элемент цепочки получает входное воздействие от q-переменных, которое может быть ошибочным в течение короткого отрезка времени. Поэтому в этом случае необходим элемент инерциальной задержки некоторого типа. Схема, приведенная на рис. 4.4, здесь не подходит, поскольку нет способа (из-за малой величины задержки этого элемента) обеспечить достаточно большой интервал следования входных изменений прибора во время цикла. Нас могла бы удовлетворить RC-задержка, однако, как замечено ранее, выбор типа инерциальной задержки требует дальнейших исследований.

Метод, описываемый ниже, представляет собой обобщение процедуры 5.3 с добавлением y_C-переменных для создания цикличности изменения состояний. Указанные переменные под воздействием входных переменных и y_S-переменных принимают такие значения, что контур обратной связи замыкается через некоторую Q-переменную.

Для дальнейшего обсуждения полезно расширить определение *множества предшественников относительно входа* I_k. Будем называть два состояния принадлежащими одному и тому же множеству предшественников относительно входа I_i, если система, первоначально находившаяся в одном из этих состояний, под воздействием I_i переходит в одно и то же устойчивое состояние или в одно и то же множество циклических состояний. Пусть D_i обозначает число множеств предшественников в столбце I_i и пусть

$$K_i = \begin{cases} \mathrm{E}[\log D_i], & \text{если в столбце } I_i \text{ нет циклов,} \\ 1 + \mathrm{E}[\log D_i], & \text{если столбец } I_i \text{ содержит} \\ & \text{хотя бы один цикл.} \end{cases}$$

Тогда индекс обратной связи синтезируемой схемы равен $f = \max\limits_i K_i$. Возможно, что минимальное значение f достижимо, однако этот факт не доказан.

Процедура 5.4. *Синтез схем, реализующих НИВ-функции, с минимальным числом обратных связей.*

1. Для каждого входа выбрать множество y_S-переменных для разбиения состояний на множества предшественников относительно данного входа, как в процедуре 5.3. (Для функции, заданной в таблице 5.6, которую мы будем использовать в качестве примера, столбец I_1 содержит

Таблица 5.6

Пример для синтеза НИВ-функции

	I_1	I_2	I_3	I_1 y_{S_1}	I_3 $y_{S_2}y_{S_3}$	I_3 y_{S_4}	I_1 $y_{T_1}y_{T_2}y_{C_1}y_{C_2}y_{C_3}$
1	***1***,00	***1***,00	5,00	0	0 0	0	0 0 0 0 0
2	1,01	4,00	***2***,00	0	1 1	1	0 1 0 0 0
3	2,11	***3***,00	5,00	0	0 1	0	1 1 0 0 0
4	5,11	***4***,00	5,00	1	1 1	0	1 1 0 0 0
5	6,10	3,00	***5***,00	1	0 1	0	0 1 0 0 0
6	7,00	***6***,00	2,00	1	1 0	1	0 0 {0 0 0 / 1 1 1}
7	8,01	6,00	2,00	1	1 0	1	0 0 {1 0 0 / 0 1 1}
8	6,11	6,00	2,00	1	1 0	1	0 0 {1 1 0 / 0 0 1}

переход, заканчивающийся в устойчивом состоянии 1, и переход, заканчивающийся в состоянии 6, принадлежащем циклу, содержащему, кроме 6, состояния 7 и 8. Поэтому y_{S_1} различает множества предшественников {1, 2, 3} и {4, 5, 6, 7, 8}.)

2. Каждое множество предшественников устойчивого состояния рассматривается, как в процедуре 5.3. Кроме того, все состояния циклов трактуются как устойчивые состояния для приписывания y_T-переменных состояниям последовательностей, оканчивающихся в циклах. (В нашем примере y_{T_1} и y_{T_2} используются, как и в предыдущей процедуре, для кодирования состояний последовательности $3 \to 2 \to 1$ столбца I_1. Кроме того, они же используются для кодирования состояний последовательности $4 \to 5 \to 6$, оканчивающейся в цикле.)

3. Если множество предшественников включает в себя цикл с периодом n, то:

3.1. Если n — четно ($n=2k$), закодировать состояния цикла с помощью k y_C-переменных. Нумеруя состояния цикла, начиная с некоторого произвольного состояния, приписать y_C-состояние $1^{i-1}0^{k+1-i}$ i-му состоянию при $1 \leqslant i \leqslant k$ и y_C-состояние $0^{i-k-1}1^{2k+1-i}$ i-му состоянию при $k+1 \leqslant i \leqslant 2k$.

3.2. Если n — нечетно, закодировать состояния цикла с помощью n y_C-переменных, приписывая y_C-состояние $1^{i-1}0^{n+1-i}$ и инверсное ему состояние i-му состоянию цикла. (В нашем примере в столбце I_1 парами состояний $y_{C_1}y_{C_2}$ y_{C_3} закодирован цикл, содержащий состояния 6, 7 и 8.)

Переменные, используемые в качестве y_C- или y_T-переменных в некотором множестве предшественников, могут быть также использованы для кодирования других множеств предшественников в том же столбце.

4. Число Q-переменных, используемых в столбце I_i, равно K_i. В столбцах без циклов эти переменные, как и в процедуре 5.3, используются для переходов по y_S-состояниям. В столбцах, содержащих один или более циклов, одна из Q-переменных используется для замыкания контура обратной связи, составленного из y_C-задержек; поэтому в столбцах с несколькими контурами одна из Q-переменных зависит и от y_C-, и от y_S-переменных. Например,

$$Q_1 = I_1 y_{S_1} + I_2 y_{S_2} + I_3 y_{S_4},$$
$$Q_2 = I_1 y_{C_3} + I_2 y_{S_3}.$$

5. Выражения для Y_S-, Y_T- и Z-функций строятся, как и раньше. При входном воздействии, для которого сделан выбор Y_{C_i}, и во множестве предшественников, где Y_{C_i} действует как y_C-переменная (как определяемая устойчивыми y_S-переменными), $Y_{C_i}=\bar{q}_t$, если i — наименьший индекс состояния этого цикла (q_t равно значению y_{C_j} с максимальным индексом из того же цикла), в противном случае $Y_{C_i}=y_{C_{i-1}}$.

В нашем примере, поскольку столбцы y_{S_3} и y_{T_2} одинаковы, можно использовать одну переменную $y_{S_3T_2}$ для обоих столбцов, вычисляя выражение для $Y_{S_3T_2}$ так же, как и для Y_{T_2}. В результате получаем:

$$
\begin{aligned}
Z_1 &= I_1 y_{T_1} + I_1 y_{S_1} y_{S_3T_2} + I_1 y_{C_2}\bar{y}_{C_3} + I_1\bar{y}_{C_2}y_{C_3},\\
Z_2 &= I_1\bar{y}_{S_1}y_{S_3T_2} + I_1 y_{T_1} + I_1 y_{C_1}\bar{y}_{C_3} + I_1\bar{y}_{C_1}y_{C_3}.\\
Y_{S_1} &= I_1 q_1 + I_2 q_1 + I_3\bar{q}_1,\\
Y_{S_2} &= I_1 q_1 + I_2 q_1 + I_3 q_1,\\
Y_{S_4} &= I_1 q_1 + I_2 q_1\bar{q}_2 + I_3 q_1,\\
Y_{T_1} &= I_2 q_2,\\
Y_{S_3T_2} &= I_1 y_{T_1} + I_2 q_2 + I_3,\\
Y_{C_1} &= I_1\bar{q}_2,\\
Y_{C_2} &= I_1 y_{C_1},\\
Y_{C_3} &= I_1 y_{C_2}.
\end{aligned}
$$

Заметим, что буква y_{S_3} в выражении для Q_2 должна быть заменена на $y_{S_3T_2}$.

Как отмечено в конце процедуры 5.3, мы не предпринимали никаких попыток для выяснения того, как минимизировать число y-переменных. Весьма вероятно, что можно, например, использовать одни и те же переменные для различных целей в различных столбцах (как в нашем примере). Такое многократное использование может привести к появлению в схеме псевдоконтуров, но это не вызовет никаких неприятностей.

5.6. Схемы с единственной обратной связью

В предыдущих разделах проводилась неявная аналогия между циклом и устойчивым состоянием. Предположим теперь, что мы исследуем эту аналогию, сняв ограничение, введенное в теореме 5.2, о том, что устойчивые состояния таблицы переходов представляются устойчивыми состояниями схемы, и допуская, что устойчивое состояние таблицы представляется множеством неустойчивых

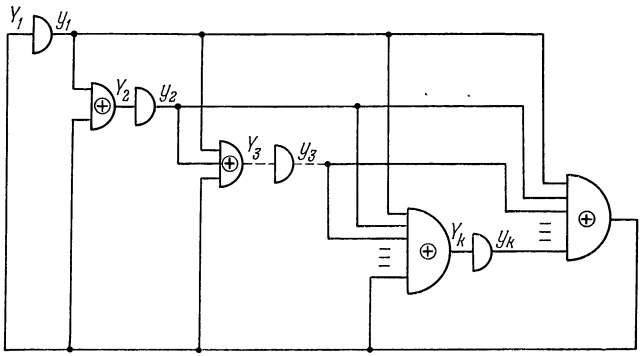


Рис. 5.10. Схема с единственной обратной связью, имеющая $k+1$ состояние.

состояний схемы. Весьма удивительным результатом устранения такого ограничения является метод реализации произвольной последовательности функции с помощью схемы, индекс обратной связи которой равен 1. К сожалению, этот метод требует использования элементов инерциальной задержки, имеющих пороговые свойства, а эти элементы могут быть реализованы только в виде схем, содержащих усилители и обратную связь.

Рассмотрим схему, изображенную на рис. 5.10 (в качестве логических элементов выбраны сумматоры по модулю 2), допуская в первом приближении, что паразитные задержки отсутствуют. Легко видеть, что каждый y-сигнал передается по двум различным путям к сумматору, управляющему Y-сигналом. Одно соединение является прямым, другое проходит через последний сумматор и контур обратной связи. Так как сумма по модулю 2 некото-

рой переменной с самой собой равна нулю, то не существует Y-сигналов, зависящих от приходящих слева y-сигналов. Из-за наличия обратной связи каждая переменная Y_i зависит от всех y-переменных (включая y_i), действующих справа от Y_i, так что при $1 \leqslant i \leqslant k$

$$Y_i = \sum_{j=i}^{k} y_j, \mod 2.$$

Следовательно, если справа от y_i действует четное число y-сигналов, равных 1, то $Y_i = y_i$ и y_i устойчиво в обоих состояниях. С другой стороны, если нечетное число y-сигналов, действующих справа от y_i, равно 1, то $Y_i = y_i \oplus 1$, т. е. y_i неустойчиво.

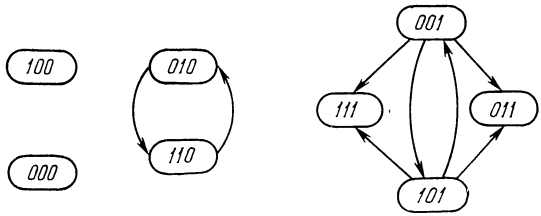


Рис. 5.11. Финальные множества состояний $k = 3$.

Предположим теперь, что самым правым y-сигналом, равным 1, является y_j, т. е. $y_j = 1$ и $y_i = 0$ при $j < i \leqslant k$. Тогда y_j и все y-переменные, действующие справа от y_j, устойчивы и находятся в состоянии $1000\ldots0$, y_{j-1} неустойчива, y_{j-2} устойчива, если $y_{j-1} = 1$, и неустойчива, если $y_{j-1} = 0$, и т. д. Таким образом, выходные сигналы всех задержек с номерами j, $j+1$, ..., k остаются равными $10\ldots0$, в то время как сигналы, действующие слева от j-й задержки, изменяются некоторым образом, в зависимости от относительных величин задержки элементов задержки. Отметим, что система имеет два устойчивых состояния: 1) все y-переменные равны 0; 2) все y-переменные, кроме y_1, равны 0.

В качестве примера рассмотрим случай, когда $k = 3$. На рис. 5.11 изображен направленный граф, на котором дуга ведет от состояния p к состоянию q, если состояние q является возможным последователем состояния p. Состояние имеет более одного последователя, если несколько

y-переменных неустойчивы в этом состоянии, т. е. выполняется условие состязаний. Отметим, что существует четыре множества состояний, названных Мюллером *финальными множествами*, таких, что если система находится в состоянии, принадлежащем одному из этих множеств, следующее ее состояние также будет принадлежать тому же множеству. Для класса рассматриваемых схем финальные множества состояний можно классифицировать в соответствии со значениями их неизменяющихся переменных, а именно 000, 100, $c10$, $cc1$ *). Более компактный способ записи финальных множеств заключается в задании числа переменных, имеющих значение 0 (это всегда самые правые переменные), в нашем примере 3, 2, 1 и 0 соответственно. В общем случае финальными множествами являются k, $k-1$, ..., 1, 0. Заметим, что переменная, действующая слева от имеющей значение 0 переменной, равна 1, поэтому такое описание финальных множеств показывает, что они взаимно не пересекаются. Каждое финальное множество используется для представления некоторого устойчивого состояния заданной таблицы переходов.

Продемонстрируем процедуру синтеза, применяя ее к ОИВ-таблице переходов, заданной в таблице 5.7. Процедура является непосредственным расширением процедуры 5.1 с применением правил, описанных в предыдущем разделе.

Как и раньше, каждому столбцу приписывается множество y-переменных. Финальное множество состояний этих переменных сопоставляется следующему состоянию. В таблице 5.7 переменные y_1 и y_2 приписаны столбцу 00, а финальные множества 00, 10 и $C1$ (в терминах тех же переменных) приписаны строкам со следующими состояниями, равными соответственно 1, 3, 5. Для кодирования столбца 01 не требуется никаких y-переменных; переменные y_3 и y_4 приписаны столбцу 11.

Когда на систему воздействует вход I_i, y-переменные, связанные с I_i (I_i-переменные), соединяются, как показано на рис. 5.10, причем сигнал в линии обратной связи определяется единственной Q-переменной. Остальным y-переменным приписываются значения, заданные как

*) Символ c обозначает изменяющуюся переменную.

Таблица 5.7

Пример для синтеза схемы с единственной обратной связью

	x_1x_2 00	01	11	y_1	y_2	y_3	y_4
1	***1***,0	2,0	***1***,0	0	0	0	0
2	3,1	***2***,0	1,0	1	0	0	0
3	***3***,1	2,0	4,1	1	0	C	1
4	5,1	2,0	***4***,1	C	1	C	1
5	***5***,1	2,0	***5***,0	C	1	1	0

функции I_i-переменных, а выход системы задается как функция входа и y-переменных. Символ C обозначает, что значение переменной произвольно. Для нашего примера выражения принимают вид:

$$Z=\bar{x}_2y_1+\bar{x}_2y_2+x_1y_4,$$

$$Q=\bar{x}_2(y_1\oplus y_2)+x_1(y_3\oplus y_4),$$

$$Y_1=\bar{x}_2q+\bar{x}_1x_2,$$

$$Y_2=\bar{x}_2(y_1\oplus q)+x_1y_4+x_1y_3,$$

$$Y_3=\bar{x}_2y_2+x_1q,$$

$$Y_4=\bar{x}_1y_1\bar{y}_2+x_2(y_3\oplus q).$$

Отметим, что Y_2 является функцией от y_3, а Y_3 — функцией от y_2, однако так как входной сигнал не может принять такого значения, чтобы оба соотношения выполнялись одновременно, схема содержит псевдоконтур. Действительные контуры замыкаются через Q-переменную.

Теперь допустим, что в схеме рис. 5.10 присутствуют паразитные задержки, так что действие изменения y_i достигает Y_j ($j>i$, y_j — устойчивая переменная) по одному пути раньше, чем по второму. Тогда на протяжении интервала времени между моментами распространения изменения y_i до узла Y_j значение Y_j изменяется неправильно. Эта ошибка проявляется в узле y_j, если задержка y_j не обладает свойством инерциаль-

ности. Предположим, что все элементы задержки инерциальны с минимальным значением задержки D_m и что максимальный интервал существования ошибки, определенный выше, равен $\mathscr{E}$ (как функция значений паразитных задержек). Тогда всякий раз, когда y-переменная, действующая слева от y_j, изменяет значение, на входе задержки y_j может появляться ошибочный сигнал, длительность которого не превышает $\mathscr{E}$. Так как слева от y_j может быть не больше $k-1$ y-переменной (наибольшее число достигается, когда y_j — самая правая переменная, т. е. $j=k$) и поскольку ни одна инерциальная задержка не может изменить свой выходной сигнал больше одного раза на любом интервале длительности D_m, то максимальное полное время, на протяжении которого значение Y_j ошибочно внутри интервала длиной D_m, равно $(k-1)\mathscr{E}$. Таким образом, значение D_m можно выбрать достаточно большим, чтобы полное время, в течение которого входной сигнал произвольной задержки ошибочен, уменьшилось настолько, насколько это необходимо. Однако отдельные ошибочные импульсы могут появиться в любой момент, а управлять интервалом их следования невозможно. Следовательно, задержка, изображенная на рис. 4.4, в данном случае неприменима.

В схеме, приведенной на рис. 5.10, любой изменяющийся y-сигнал во время изменения непосредственно воздействует на собственный сигнал возбуждения и на другие Y-сигналы. (Это утверждение неверно, например, для класса схем, аналогичных изображенной на рис. 5.8.) Поэтому важно, чтобы интервал, во время которого сигнал имеет неопределенное значение, был мал по сравнению с временем реакции задержек схемы. Таким образом, RC-задержки также неприемлемы. Для наших целей необходим пороговый прибор некоторой разновидности (см. раздел 4.1), который быстро изменяет свой выходной сигнал после принятия решения. По-видимому, наличие усиления и обратной связи свойственно схемам с указанным поведением, хотя задача не сформулирована с достаточной точностью для того, чтобы получить определенный ответ. Конечно, если обратная связь должна вводиться в каждый элемент задержки, то нет смысла применять описанный подход. Тем не менее считается, что такой подход полезен для построения остроумной схемы рис. 5.10.

ЗАМЕЧАНИЯ ПО БИБЛИОГРАФИИ

Теорема 5.1 является обобщением теоремы, доказанной Ангером [110, 111], причем в последней работе также представлен материал раздела 5.2. Отметим, что необходимость введения усилителей в замкнутые контуры замечена ранее Хаффманом [50]. Пример схемы, индекс обратной связи которой меньше числа внутренних переменных, построен Ангером [111]. Теорема 5.2 доказана Эйхельбергером [13], общий метод синтеза с Q-переменными принадлежит Хаффману [52]. Идея совместного использования этого метода и ОТП-кодирования для построения схем с минимальным индексом обратной связи, метод синтеза схем с минимальным числом транзисторов, схемы с единственной обратной связью и идея метода синтеза схем с минимальным числом обратных связей, реализующих функции с ограниченным числом выходных последовательностей (раздел 5.5), разработаны Фридманом [19, 20]. Изложенный в последней части раздела 5.4 метод, в результате применения которого иногда сокращается число внутренних переменных, требуемых в методе Фридмана построения схем с минимальным числом транзисторов, принадлежит Фридзу [18]. Известна интересная работа по синтезу синхронных или импульсных схем с минимальным числом обратных связей. Эта задача довольно сильно отличается от синтеза асинхронных схем, и здесь Фридман показал, что достижимо решение с единственной обратной связью [19, 21].

ЗАДАЧИ

5.1 *. Синтезировать RS-триггер с помощью элементов типа И, ИЛИ, НЕ, предполагая, что элементы И и ИЛИ ослабляют сигналы. Найти решение с минимальным числом инверторов.

5.2$^+$. Синтезировать τ-триггер, изменяющий свой выходной сигнал после изменения значения входа из 1 в 0, т. е. триггер должен переключаться под действием заднего фронта входного импульса. Предполагается, что входы непарные; попытаться минимизировать число используемых транзисторов.

5.3. Построить выражения для реализации Y- и Q-функций с минимальным числом транзисторов по таблице переходов, использованной в задаче 3.15.

5.4. Построить выражения для реализации Y- и Q-функций с минимальным числом транзисторов по таблице переходов, использованной в задаче 3.12.

Таблица 5.8

	x_1x_2 00	01	11
1	***1***,0	2,1	***1***,0
2	***2***,0	3,0	4,1
3	2,0	***3***,1	1,0
4	***4***,1	3,0	***4***,1

5.5. Построить реализацию с минимальным числом обратных связей функции, заданной в таблице 5.8.

ГЛАВА 6
ДРУГИЕ СПОСОБЫ РАБОТЫ СХЕМ

Рассмотренные в главе 1 некоторые классы схем здесь будут исследованы более детально наряду со схемами других типов. Поведение этих схем различно в зависимости от условий, при которых допустимы входные изменения, а также в зависимости от внутренней структуры. Наши исследования охватывают широкий класс задач: от максимизации средней скорости функционирования до минимизации структур в процессе их построения.

6.1. Схемы, генерирующие сигнал завершения

Схемы, генерирующие сигналы завершения, названные в разделе 1.6 *схемами Мюллера* (или *схемами, не зависящими от скорости*), посылают источнику входных сигналов сигналы о своей готовности принять новое входное состояние. В случаях, когда время, необходимое для обработки входного сигнала, изменяется в широких пределах в зависимости, естественно, от конкретного воздействия или внутреннего состояния, возможно существенное увеличение средней скорости подачи входных сигналов.

Такой подход возможен лишь тогда, когда система управляет источником входных сигналов или когда между источником и системой может быть помещено буферное устройство, управляемое системой. Если система должна немедленно вырабатывать реакции на неуправляемые входные изменения, как это случается во многих случаях при работе системы в реальном времени, то построение схем Мюллера невозможно. Очевидно, что наиболее естественным местом применения этих схем является цифровая вычислительная машина, представляющая собой после загрузки ее памяти программой и исходными данными для решения задачи замкнутую

систему. Схемы с сигналом завершения могут быть использованы также в отдельных частях системы или для управления линиями связи между основными блоками система в то время, как в остальных подсистемах будет применен иной подход.

Как станет ясно позднее, указанные в разделе 4.4 ограничения, наложенные на распределение паразитных задержек, необходимы и здесь, т. е. задержки в проводах малы по сравнению с задержками на выходах элементов. Однако, хотя в предыдущих исследованиях предполагалось, что верхняя граница значений паразитных задержек известна (это условие необходимо для вычисления минимального интервала следования входных изменений), в настоящей главе мы откажемся от знания такой величины, и кроме того, в дальнейшем не будем использовать элементы задержки. Мы будем также считать, что паразитные задержки могут быть как совершенного, так и инерциального типа, хотя некоторые исследователи постулируют наличие только инерциальных задержек.

Сначала мы рассмотрим методы синтеза комбинационных схем, а затем полученные результаты распространим на последовательностные схемы, реализующие ОИВ-функции. В последнем подразделе будут обсуждены вопросы построения сетей, состоящих из таких схем. С самого начала следует отметить, что методы, представленные в этом разделе, отражают лишь один из многих подходов, и наши исследования гораздо менее исчерпывающи, чем в предыдущих главах.

6.1.1. Комбинационные схемы с сигналом завершения. По-видимому, впервые необходимость в генерации сигнала завершения появилась при построении такой итеративной комбинационной схемы, как параллельный двоичный сумматор. Конструкция сумматора имеет особенность, заключающуюся в том, что время, необходимое для получения результата, зависит от величин суммируемых чисел. Наибольшее время затрачивается, когда сигнал переноса распространяется по всей схеме. Если же переноса нет, то время вычисления может быть меньше максимального примерно в n раз для n-разрядного сумматора. При случайном выборе слагаемых среднее время, затрачиваемое на суммирование, оценивается числом

$\log_2(5/4)$*). Однако в случае, когда сумматор не генерирует сигнала, свидетельствующего об окончании вычислений, для правильного функционирования необходимо отводить максимальное время.

Чтобы избежать лишних потерь времени, сумматор был сконструирован так, что он вырабатывал сигнал завершения для ведения вычислений со средней, а не минимальной скоростью. Такой сумматор может содержать вдвое больше элементов и работает медленнее, чем простой сумматор; но тем не менее при соответствующих условиях его использование дает значительный выигрыш в эффективности. Метод, применяемый при синтезе таких сумматоров [27], был обобщен на широкий класс итеративных схем [117], но вместо обсуждения указанного метода мы рассмотрим метод, используемый для реализации произвольной комбинационной функции. Заметим однако, что существует два до некоторой степени сходных метода. Будем предполагать, что, если не оговорено особо, паразитные задержки полностью сосредоточены на выходах элементов и задержки в проводах отсутствуют.

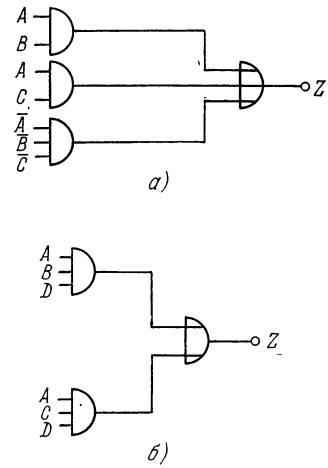


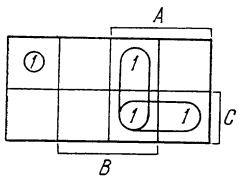
Рис. 6.1. *а*) Схема, содержащая 0101-состязание задержек на переходе 011 ⟶ ⟶ 111 ⟶ 101. *б*) Схема, содержащая 01010-состязание задержек на переходе 0111 ⟶ 1111 ⟶ 1110.

Сначала мы опишем явление, составляющее принципиальное препятствие при синтезе схем Мюллера, а именно, *состязание задержек*. Рассмотрим рис. 6.1, *а*, на котором изображена двухуровневая И—ИЛИ-схема, реализующая функцию $A\bar{B}\bar{C} + AB + AC$ без логических состязаний (в K-карте функции (см. табл. 6.1, *а*) отмечены используемые элементарные произведения). Входная

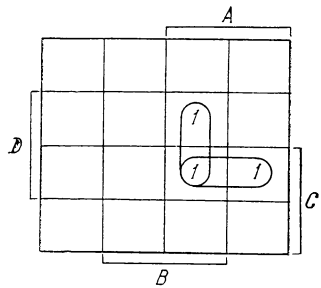
*) Этот результат получен Хендриксоном [46].

последовательность 011, 111, 101 должна вызывать выходную последовательность 0, 1, 1. Предположим, что мы наблюдаем входные переменные (A, B и C) и выходную переменную Z для определения факта окончания действия первого входного изменения и готовности схемы к второму изменению. Можно считать, что после изменения Z

Таблица 6.1



а) Функция, реализуемая схемой рис. 6.1, *а*



б) Функция, реализуемая схемой рис. 6.1, *б*

из 0 в 1, происходящего в результате входного изменения $011 \longrightarrow 111$ (первое изменение), схема готова к второму изменению на входах ABC. Однако при таком предположении на выходе может возникнуть ложный импульс. Начальное входное изменение воздействует на два элемента И со входами AB и AC. Предположим, что задержка AC-элемента И достаточно велика. Тогда AB-элемент изменит свою реакцию первым, и элемент ИЛИ ответит на нее раньше, чем изменится выход AC-элемента. Если теперь произойдет второе входное изменение ($111 \longrightarrow 101$), вновь изменятся выходные значения AB-элемента и элемента ИЛИ, в то время как выход AC-элемента все еще будет сохранять нулевое значение. И, наконец, после срабатывания AC-элемента выход Z окончательно установится в 1.

Таким образом, выходной последовательностью на самом деле будет 0101 вместо 011. Заметим, что этот процесс аналогичен рассмотренному в разделе 4.2 при наличии динамического состояния в случае изменения нескольких входных переменных.

В общем случае мы определяем *состязание задержек* как такое состояние схемы, свободной от логических состязаний, при котором последовательность двух входных изменений $I_1 \longrightarrow I_2 \longrightarrow I_3$ может вызвать одну из двух выходных последовательностей:

1. $f(I_1)$, $f(I_2)$, $\overline{f(I_2)}$, $f(I_3)$, где $f(I_2) = f(I_3)$;
2. $f(I_1)$, $f(I_2)$, $f(I_3)$, $f(I_2)$, $f(I_3)$, где $f(I_2) \neq f(I_3)$.

Рис. 6.1, *а* и таблица 6.1, *а* иллюстрируют первую возможность, а рис. 6.1, *б* и таблица 6.1, *б* — вторую (предполагается, что паразитные задержки являются задержками совершенного типа).

Если диапазон величин паразитных задержек известен, то мы можем выждать достаточное время после очередного входного изменения для того, чтобы действие этого изменения распространилось по всей схеме и можно было допустить следующее входное изменение. Если указанный диапазон не известен или мы не расположены «предполагать худшее» в каждом случае, для обеспечения надежной работы схемы необходимо специальное управление элементами, составляющими схему, а также входными сигналами. Заметим, однако, что причина состязаний задержек заключается в том, что во время перехода изменяются сигналы на нескольких входах элемента ИЛИ. Как только один из входных сигналов элемента ИЛИ принимает единичное значение, по наблюдению выхода схемы невозможно выяснить, установился ли в 1 какой-либо другой его вход. Следовательно, мы не можем сказать, когда схема полностью переработает входное изменение. Если же в схеме при переходе может изменяться лишь один вход выходного элемента ИЛИ, то появляется возможность определить по выходу схемы, когда она приходит в устойчивое состояние после входного изменения.

Двойственная картина имеет место в ИЛИ—И-схемах. Здесь возникают аналогичные явления, когда более чем один входной сигнал может принять значение 0, так как первое входное изменение в 0 замаскирует состояние остальных входов.

Нашей основной целью является синтез схемы, для которой изменения входных состояний допустимы после установления схемы в устойчивое состояние и последова-

тельности выходных сигналов точно соответствуют входным последовательностям. В случаях, когда схема имеет несколько выходов, выходное состояние может, конечно, во время переходов принимать промежуточные значения между желаемыми.

Наш подход состоит в том, чтобы заставить входной сигнал изменяться между входным состоянием, соответствующим существенному состоянию входов, называемому *словом данных*, и фиксированным *промежуточным словом*. Эти слова должны быть выбраны так, чтобы выполнялись следующие условия:

1. Имеется возможность легко определить, является ли входное состояние словом данных или промежуточным словом.

2. Никакое слово данных не может лежать в области, натянутой на промежуточное слово и другое слово данных.

Согласно второму условию между двумя последовательными промежуточными словами должно содержаться точно одно слово данных. Выходные состояния могут быть закодированы аналогичным образом, хотя, как будет показано далее, соблюдение этого требования необязательно.

Существует много способов кодирования, удовлетворяющих указанным условиям, однако здесь будет рассмотрен лишь один из них, так называемый *метод парных входов*. Каждый сигнал x передается при таком методе по двум линиям x_0 и x_1. Если $x = 0$, то x_0 придается значение 1, а $x_1 — 0$. Условию $x = 1$ соответствуют равенства $x_0 = 0$, $x_1 = 1$. Промежуточное состояние кодируется нулевыми значениями сигналов x_0 и x_1 для каждой переменной. Условий, при которых $x_0 = x_1 = 1$, не существует. Удобно считать, что x_1 представляет x, а $x_0 — \bar{x}$, хотя следует помнить, что, поскольку оба сигнала могут быть одновременно равны 0, в действительности они не являются отрицаниями друг друга.

Такой способ кодирования удовлетворяет нашим условиям. Действительно,

1. Легко определить, когда входное состояние является словом данных (на одной линии каждой пары присутствует 1), а когда промежуточным словом (сигналы на всех входных линиях равны 0).

2. При изменении входного состояния от промежуточного слова к слову данных ни одно из промежуточных входных состояний не является словом данных, при этом предполагается, что ни один сигнал, равный нулю в начале и в конце входного изменения, не может временно принимать единичное значение. Аналогичное утверждение

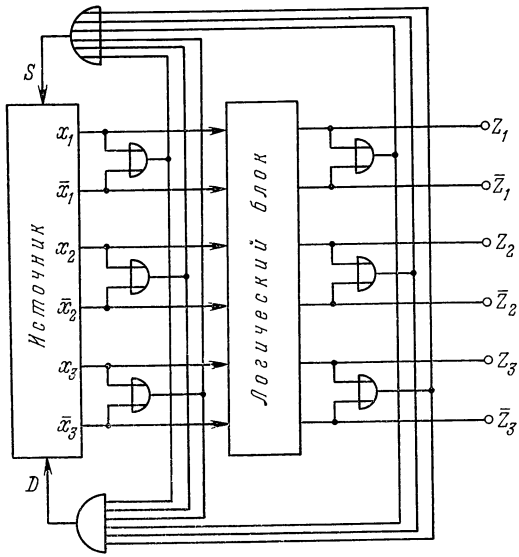


Рис. 6.2. Комбинационная схема с сигналом завершения.

справедливо и для перехода от слова данных к промежуточному слову.

Рассмотрим теперь систему, блок-схема которой изображена на рис. 6.2, считая, что сигналы на выходах Z_i и $\bar{Z}_i$ представляют собой реакции двухуровневой И—ИЛИ-схемы. Предположим, что в начальном состоянии источник генерирует промежуточное слово, т. е. устанавливает все x_i и $\bar{x}_i$ в 0, и что это слово сохраняется до тех пор, пока все элементы И и ИЛИ логического блока имеют нулевую реакцию. Тогда выход S верхнего элемента ИЛИ и выход D нижнего элемента И также равны 0. Эти сигналы воспринимаются источником как

разрешение подать на схему очередное слово данных. В результате подачи слова данных некоторые элементы И логического блока установятся в 1, что в свою очередь приведет к изменению одного из значений Z_i или $\overline{Z}_i$ для каждого i. После завершения описанного процесса на выходе появится реакция на слово данных и изменятся сигналы D и S. Условие $D = S = 1$ источник воспринимает как разрешение подать промежуточное слово, придав всем сигналам x_i и $\overline{x}_i$ нулевые значения. Тогда все сигналы логического блока, в том числе и выходные, также станут нулевыми, причем установка в 0 сигналов D и S будет свидетельствовать об окончании обработки промежуточного слова. После этого источник вырабатывает новое слово данных, и весь процесс повторится.

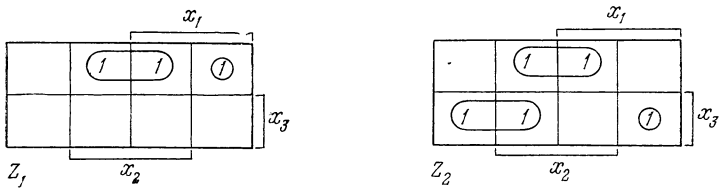
Рассмотрим функционирование схемы более детально для определения условий, при которых схема работает правильно. Предположим, что на некотором слове данных некоторый выход Z_j принял единичное значение, и кроме того, изменили свои выходные значения два элемента И, соединенные с элементом ИЛИ, имеющим выход Z_j. Если задержка на выходе одного из элементов И велика, то единичный сигнал может не распространиться через задержку до тех пор, пока на схему не будут поданы следующие промежуточное слово и слово данных. Это условие определяет состязание задержек и может привести к появлению неправильного выходного сигнала.

Для устранения такой возможности введем ограничение: логический блок должен быть построен так, чтобы на любом слове данных для любого i только один элемент И из подсхемы, вырабатывающей выход Z_i или $\overline{Z}_i$, мог принимать единичное значение. Тогда если оба сигнала D и S меняются из 1 в 0, то в схеме не останется единичных сигналов, скрытых паразитными задержками, так как только элементы И, принимавшие значение 1, должны изменить свои выходы на нулевые. Если на схему подано слово данных, то только один элемент И принимает значение 0 для каждой пары выходов Z_i и $\overline{Z}_i$. Заметим, что в такой ситуации невозможно появление ложных импульсов, т. е. отсутствуют комбинационные состязания, поскольку схема не содержит инверторов. (Предполагается, что источник генерирует лишь слова данных и проме-

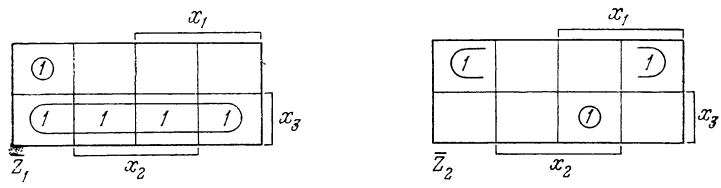
жуточные слова, никогда не воспроизводя ложных сигналов).

Требование, согласно которому для каждой пары выходов в некоторый момент времени изменяется значение выхода лишь одного элемента И, означает, что каждое единичное значение каждой выходной функции должно

Таблица 6.2



а) K-карты функций Z_1 и Z_2



б) K-карты дополнений функций Z_1 и Z_2

покрываться точно одним элементарным произведением. Такое же утверждение справедливо и для дополнения каждой функции. Это значит, что в общем случае нельзя представлять требуемые функции целиком с помощью сумм простых импликантов, т. е. существует тенденция к увеличению числа элементов, необходимых для построения таких схем. Однако, как показано ранее, чтобы обезопасить схему от комбинационных состязаний, не требуется лишних элементов (см. раздел 4.2), благодаря чему достигается некоторое сокращение числа элементов в реализуемой схеме.

Рассмотрим в качестве примера функции, заданные таблицей 6.2, *а*. Дополнительные к ним функции представлены таблицей 6.2, *б*. Отдельные элементарные произведения, отмеченные в этих таблицах, приводят

к построению следующих выражений:

$$Z_1 = x_2\bar{x}_3 + x_1\bar{x}_2\bar{x}_3,$$
$$\bar{Z}_1 = \bar{x}_1\bar{x}_2\bar{x}_3 + x_3,$$
$$Z_2 = x_2\bar{x}_3 + \bar{x}_1x_3 + x_1\bar{x}_2x_3,$$
$$\bar{Z}_2 = \bar{x}_2\bar{x}_3 + x_1x_2x_3.$$

Схема, реализующая эти функции, изображена на рис. 6.3.

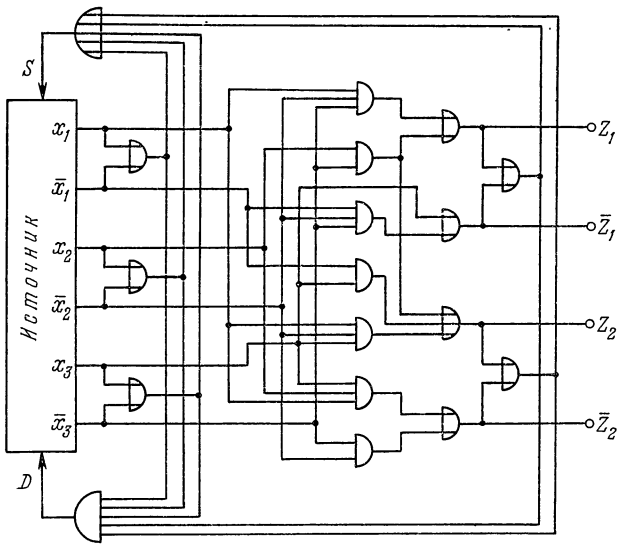


Рис. 6.3. Реализация таблицы 6.2.

Читателю полезно проследить действие схемы на входной последовательности

$$001 \longrightarrow 110 \longrightarrow 111.$$

Требуемой последовательностью выходных сигналов при этом является

$$01 \longrightarrow 11 \longrightarrow 00.$$

Для нашей схемы входная последовательность, воспроизведенная источником, и выходная последовательность

имеют вид:

$$x_1\bar{x}_1x_2\bar{x}_2x_3\bar{x}_3 = 000000 \longrightarrow 010110 \longrightarrow 000000 \longrightarrow$$
$$\longrightarrow 101001 \longrightarrow 000000 \longrightarrow 101010,$$

$$Z_1\bar{Z}_1Z_2\bar{Z}_2 =$$
$$= 0000 \longrightarrow 0110 \longrightarrow 0000 \longrightarrow 1010 \longrightarrow 0000 \longrightarrow 0101.$$

Можно обойтись без выходных промежуточных слов, если ввести триггеры на выходах. Этот способ построения схемы

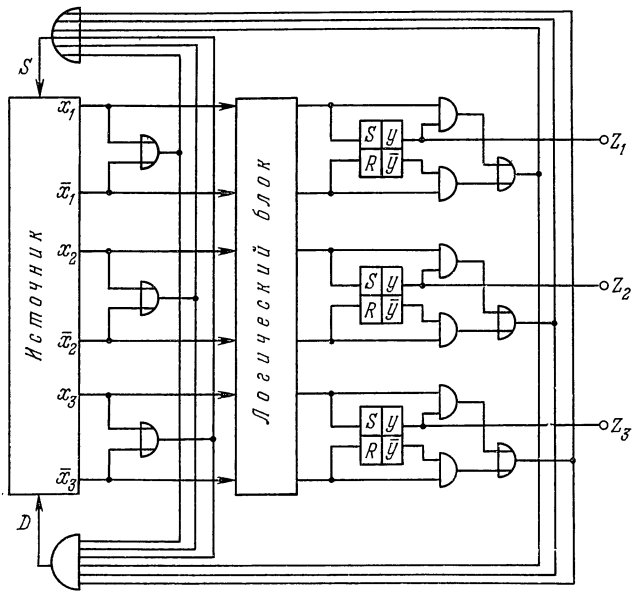


Рис. 6.4. Схема, не вырабатывающая выходных промежуточных слов.

представляет особый интерес, так как в дальнейшем составит основу метода синтеза последовательностных схем.

Как показано на рис. 6.4, для реализации каждой выходной переменной используется отдельный RS-триггер. На каждом слове данных в зависимости от значения выхода Z_i (1 или 0), определенного на подаваемом слове, возбуждается либо вход S, либо вход R триггера, связан-

ного с этим выходом. Логический блок вырабатывает сигналы возбуждения так же, как логический блок схемы, изображенной на рис. 6.2. Триггер реагирует на возбуждение, после чего элемент ИЛИ, соединенный с ним, устанавливается в 1. Когда все выходные триггеры обработают свои входные сигналы, соответствующие слову данных, элемент И, нижний на схеме, выдаст сигнал $D = 1$. Верхний элемент ИЛИ, устанавливающий тип входного слова и факт равенства нулю сигналов возбуждения всех триггеров, воспроизводит сигнал S, который вместе с сигналом D управляет входным источником, как описано раньше.

Для схемы, изображенной на рис. 6.4, реализующей функции, заданной в таблице 6.2, легко видеть, что входная последовательность

$$x_1x_2x_3 = 001 \longrightarrow 110 \longrightarrow 111$$

представляется источником в виде

$$x_1\bar{x}_1x_2\bar{x}_2x_3\bar{x}_3 = 000000 \longrightarrow 010110 \longrightarrow 000000 \longrightarrow$$
$$\longrightarrow 101001 \longrightarrow 000000 \longrightarrow 101010.$$

Выходной последовательностью Z_1Z_2 в этом случае является

$$01 \longrightarrow 11 \longrightarrow 00.$$

Заметим, что когда, как в нашем примере, на одном переходе изменяются несколько выходов, мы можем предположить, что имеют место промежуточные выходные состояния, вид которых зависит от значений паразитных задержек. Так, в действительности выходными последовательностями системы могут быть

$$01 \longrightarrow 11 \longrightarrow 01 \longrightarrow 00 \quad \text{или} \quad 01 \longrightarrow 11 \longrightarrow 10 \longrightarrow 00.$$

Теперь кратко изложим полученные результаты. В начальный момент все элементы схемы находятся в состоянии покоя (выходы всех элементов равны нулю); затем источник подает на схему входное слово. Независимо от значения паразитных задержек и их случайного распределения (считаем, что задержек в проводах нет) входной сигнал не меняется до тех пор, пока на выходах не возникнут правильные значения. Затем источник получает сигнал о том, что к схеме может быть приложен

промежуточный сигнал для перевода схемы в состояние покоя. После прохождения по схеме промежуточного сигнала и обнуления всех единичных сигналов в источник поступает разрешение приложить к схеме следующее входное воздействие.

До сих пор предполагалось, что в проводах нет задержек. Выясним, до какой степени можно ослабить это ограничение. Если считать, что в проводах (или на входах элементов — см. раздел 4.3) содержатся ненулевые задержки, изменение сигнала в узле, которым может быть входной полюс или выход элемента, непосредственно соединенный с несколькими другими точками схемы, может воздействовать на эти точки в различные моменты времени. При этом может произойти нарушение правильности функционирования, если некоторое входное изменение распространится до различных входов одного из элементов в то время, как на других его входах будут продолжать действовать сигналы, обусловленные предыдущим входным состоянием. Указанное нарушение всегда устранимо, если задержка в линиях, соединяющих пары узлов, никогда не будет превосходить задержку в двух уровнях логической схемы, вырабатывающей сигналы S и D. Более умеренное требование заключается в том, чтобы наибольшая задержка в линии никогда не превосходила половины наименьшей задержки элемента. Такое соотношение напоминает ограничение, введенное при рассмотрении в разделе 4.4 схем без задержек.

Очень полезная особеность обсуждаемых схем заключается в том, что широкий класс неисправностей схемы вызывает немедленное прерывание функционирования системы. Предположим, что в некотором узле схемы сигнал принял фиксированное значение, равное 0 или 1. Если во время правильной работы сигнал в этом узле должен иметь противоположное значение, то возникшая ситуация может классифицироваться как появление бесконечной задержки на выходе элемента (или, возможно, источника, вырабатывающего сигнал в данном узле. В результате входное состояние будет заморожено. Так, например, если в любом узле схемы, изображенной на рис. 6.3, будет постоянно сохраняться единичный сигнал, то сигнал S также постоянно будет иметь значение, равное 1, и источник перестанет получать разрешение на

подачу следующего слова данных. Если же в некотором узле будет зафиксировано нулевое значение, то при подаче на систему входного состояния, вызывающего появление в данном узле единичного значения, которое существенно для установки в 1 некоторого выхода Z_i или $\overline{Z}_i$, на указанном выходе также будет сохраняться 0, и поэтому источник не сможет перейти к генерации следующего промежуточного входного слова. Такое поведение системы очень удобно для человека, с ней работающего.

В заключение следует указать, что метод реализации источника с элементами, имеющими неограниченные паразитные задержки, пока не известен. Но синтез таких схем несложен, если используются компоненты с ограниченными паразитными задержками.

6.1.2. Последовательностные схемы с сигналами завершения. Принципы, изложенные на примере модели комбинационной схемы (см. рис. 6.4), могут служить основой для синтеза последовательностных ОИВ-функций, если охватить логический блок обратными связями, идущими с выходов некоторых триггеров, как показано на рис. 6.5. Эти y-сигналы, взятые с выходов триггеров, можно рассматривать как состояния, остальные Z-полюсы удобно считать, как и прежде, выходами. Основным вопросом является построение удовлетворительного способа кодирования строк.

При использовании ОТП-кодирования факт окончания изменения внутреннего состояния, вызванного входным изменением, однозначно определяется по установке всех триггеров в состояния, соответствующие сигналам возбуждения. (Такой же принцип использовался в схеме, изображенной на рис. 6.4.) Остается, однако, ограничить класс способов кодирования строк, чтобы в дальнейшем избежать состязаний задержек.

Как и при синтезе комбинационных схем, подсхемы, реализующие выходы Z_i и Y_i логического блока, следует создавать так, чтобы во время любого перехода единичное значение принимало не более одного элемента И из числа соединенных с любым выходным элементом ИЛИ. Это означает, что не только каждое единичное значение каждой Z- или Y-функции должно покрываться единственным элементарным произведением, но кроме того, если существует переход $i \longrightarrow j$ в некотором входном столбце, где

Y_k (или Z_k) равно 1 в i-й и j-й строках, то единственное элементарное произведение должно покрывать обе единицы в реализации Y_k (или Z_k). Но, как указано в разделе 3.3, кодирование по Лю имеет такое свойство, что в

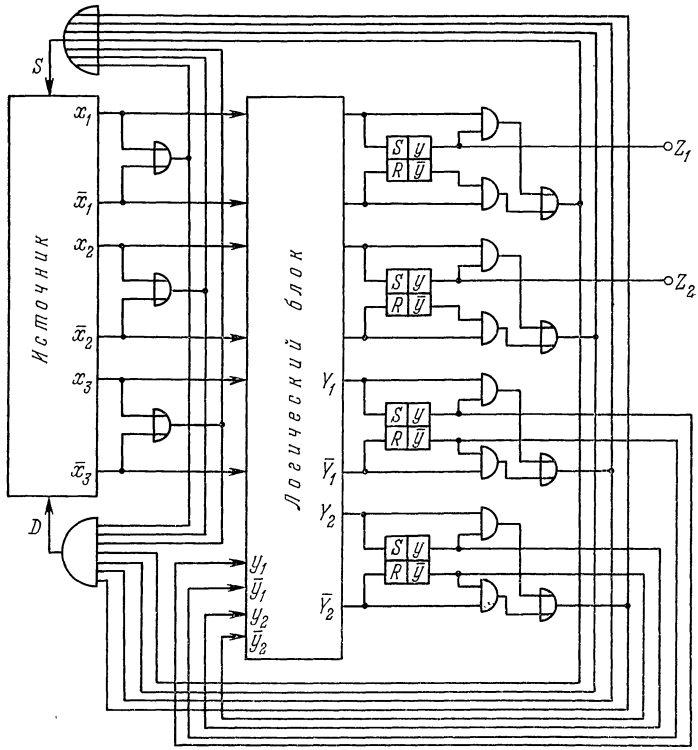


Рис. 6.5. Последовательностная схема с сигналом завершения.

каждом входном столбце единственный y-подкуб связан с каждым множеством предшественников и этот подкуб не пересекается со всеми другими подкубами, связанными с остальными множествами предшественников того же столбца. Так как значения функций возбуждения Y постоянны для каждого множества предшественников, то для каждой Y-функции множество непересекающихся элементарных произведений может покрывать единичные

значения функции так, что на любом переходе в данном столбце всегда будет изменяться значение не более чем одного элементарного произведения. Поскольку значения Z-функций постоянны на каждом множестве предшественников, они также могут быть реализованы аналогичным образом.

Итак, приемлемой конструкцией схемы является схема, изображенная на рис. 6.5, при условии, что используется кодирование по Лю, и Y- и Z-функции реализуются с помощью максимальных подкубов в каждом столбце, но без многократного покрытия какого-либо единичного значения. Как и в случае комбинационной схемы типа приведенной на рис. 6.4, необходимо вырабатывать сигналы Z_i, $\bar{Z}_i$, Y_j и $\bar{Y}_j$ для каждого i и j и каждого входного слова данных. Когда на схему воздействует промежуточное слово, все Y-, $\bar{Y}$-, Z- и $\bar{Z}$-сигналы полагаются равными нулю.

Таблица 6.3

Матрица переходов, закодированная по методу Лю

	x_1x_2 00	01	11	y_1	y_2	y_3
1	***1***,0	2,0	***1***,0	0	0	0
2	3,0	***2***,0	***2***,0	1	0	0
3	***3***,0	5,1	4,0	1	0	1
4	1,0	5,1	***4***,0	0	0	1
5	***5***,1	***5***,1	4,0	0	1	1

В качестве примера рассмотрим таблицу переходов, закодированную по Лю (см. табл. 6.3). Выражения для функций возбуждения триггеров получены в предположении, что $S = Y$ и $R = \bar{Y}$, с использованием элементарных произведений, покрывающих максимальные подкубы в каждом столбце. В каждом выражении выбраны взаимно непересекающиеся элементарные произведения:

$$S_1 = \bar{x}_2 y_1 + \bar{x}_1 x_2 \bar{y}_3 + x_1 x_2 y_1 \bar{y}_3,$$
$$R_1 = \bar{x}_1 \bar{x}_2 \bar{y}_1 + \bar{x}_1 x_2 y_3 + x_1 \bar{y}_1 + x_1 y_1 x_3,$$
$$S_2 = \bar{x}_2 y_2 + \bar{x}_1 x_2 y_3,$$
$$R_2 = \bar{x}_1 \bar{x}_2 \bar{y}_2 + \bar{x}_1 x_2 \bar{y}_3 + x_1,$$
$$S_3 = \bar{x}_2 y_1 \bar{y}_2 + \bar{x}_2 y_2 + x_2 y_3,$$
$$R_3 = \bar{x}_2 \bar{y}_1 \bar{y}_2 + x_2 \bar{y}_3,$$
$$Z = \bar{x}_2 y_2 + \bar{x}_1 x_2 y_3, \qquad \bar{Z} = \bar{x}_1 \bar{x}_2 \bar{y}_2 + \bar{x}_1 x_2 \bar{y}_3 + x_1.$$

Изложенный подход может быть модифицирован, если желательно получить выходные изменения, разделенные промежуточными словами, как это делалось в комбинационной схеме, изображенной на рис. 6.2. Такое изменение может быть достигнуто за счет исключения выходных триггеров и получения сигналов Z и $\overline{Z}$, как и раньше, с помощью элементов ИЛИ, определяющих окончание выработки каждого выходного сигнала (см. рис. 6.6).

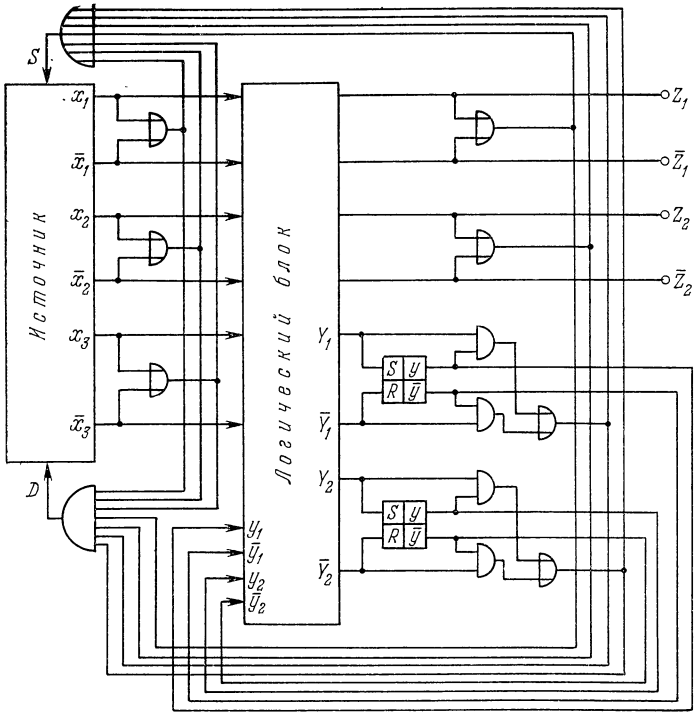


Рис. 6.6 Последовательностная схема, вырабатывающая выходные промежуточные слова.

Комбинационные и последовательностные состязания не составляют проблемы для схем, построенных по описанным выше правилам, а прекращение работы в случае появления неисправностей в схеме (см. предыдущий подраздел) является характерной особенностью этих схем.

6.1.3. Сети схем. В сложных цифровых системах может появиться необходимость в соединении нескольких комбинационных и последовательностных схем, структура которых обсуждалась в двух предыдущих подразделах. Такое соединение требует некоторого изменения схем для обеспечения запоминания их выходных состояний и введения средств для получения сигналов завершения.

Простейшее соединение схем изображено на рис. 6.7, где на самую левую схему воздействуют входные сигналы,

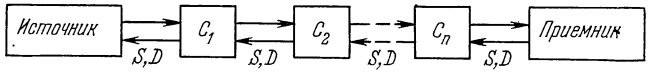


Рис. 6.7. Цепочка схем.

воспроизводимые источником, управление которым, как описано раньше, осуществляется с помощью сигналов S и D. Каждая внутренняя схема цепочки получает входное слово данных с выхода предшествующей ей (левой) схемы и своим выходом воздействует на следующую за ней (правую) схему. Когда правая схема разрешает подачу на свой вход слова данных или промежуточного слова, сигнал разрешения должен быть подан на левую схему. Приемник, включенный на правом конце цепочки, воспринимает результат работы всей системы и воспроизводит сигнал разрешения на подачу очередного слова данных или промежуточного слова в соответствии с собственной скоростью функционирования. Можно считать, что приемником является, например, печатающее устройство, а источником — промежуточное запоминающее устройство.

Каждая схема цепочки разрешает подачу на нее промежуточного слова (слова данных), когда ее выходным состоянием является слово данных (промежуточное слово), причем сигнал разрешения действует, когда выходным состоянием левой схемы является промежуточное слово (слово данных) и правая схема разрешает подачу промежуточного входного слова (слова данных). Таким образом, в длинной цепочке схем несколько последовательно поданных на систему различных входных состояний могут проявляться на разных схемах, разделенных, одна-

ко, схемами, выходными состояниями которых являются промежуточные слова. Заметим, что более позднее входное воздействие никогда не может исказить результатов действия более ранних входных сигналов.

Модель комбинационной схемы, пригодной для включения в рассмотренную цепочку в качестве i-го модуля, изображена на рис. 6.8 и построена на основе схемы рис. 6.2, в которую добавлены триггеры для каждого из выходов Z_i и $\overline{Z}_i$. Когда $(i+1)$-й модуль требует подачи промежуточного входного слова (на рис. 6.8 вход $SD = 10$) и на выходе $(i-1)$-го модуля появилось промежуточное слово, верхний элемент И (см. рис. 6.8) возбуждается, вызывая установку в 0 всех триггеров i-го модуля, в результате чего на выходе системы появляется промежуточное слово. После этого (заметим, что $\overline{y}$-выходы не могут принять единичных значений, пока входы S триггеров не установятся в 0) возбуждается нижний элемент И, устанавливая в 0 *триггер разрешения* Q, который вырабатывает сигнал (выход $SD = 01$), являющийся запросом слова данных от $(i-1)$-го модуля.

Сигнал разрешения ($SD = 01$), поступающий в i-й модуль из $(i+1)$-го модуля, воздействует на блок логики (сигнал D является входным сигналом для каждого элемента И блока логики), и, если триггер Q установлен в 0, на блок логики будет подано слово данных из $(i-1)$-го модуля. Таким образом, каждый модуль обрабатывает последовательность сигналов, приходящую слева на его входы, и передает результаты вправо по сигналу разрешения, разделяя слова данных промежуточными словами.

Рассмотрим теперь, как следует изменить структуру последовательностной схемы, изображенной на рис. 6.6, чтобы появилась возможность включать ее совместно с комбинационными схемами в систему, приведенную на рис. 6.7. Для этой цели (см. рис. 6.9) необходимо добавить по два триггера для каждой выходной переменной и схему, определяющую, когда на выходе появилось промежуточное слово (как и на рис. 6.8). Один триггер этой схемы запоминает тип последнего выходного слова, а другой триггер определяет, являются ли все триггеры, управляемые внутренними переменными, одновременно возбужденными и находящимися в устойчивых состояниях, т. е.

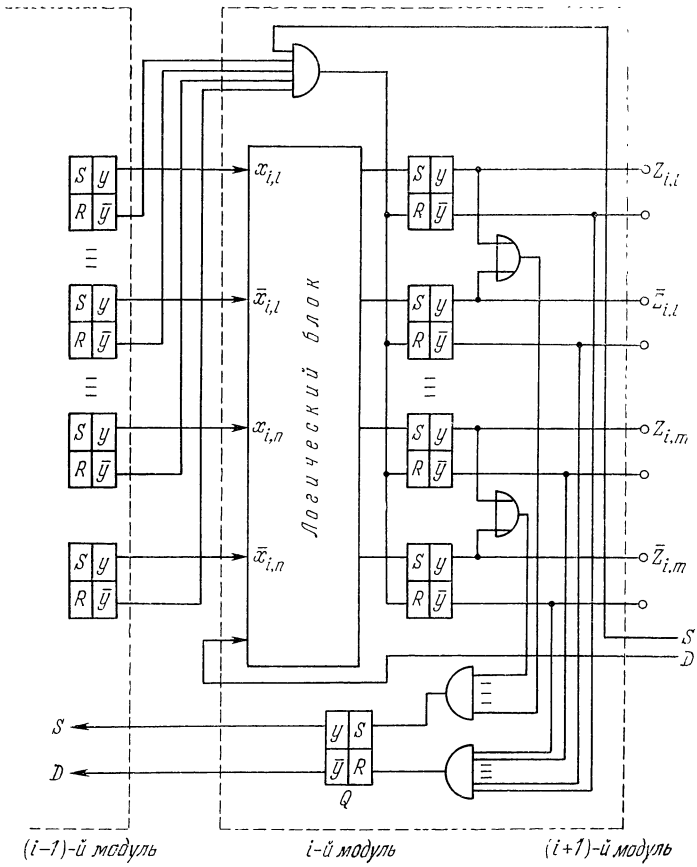


Рис. 6.8. Комбинационный модуль, пригодный для включения в цепочку.

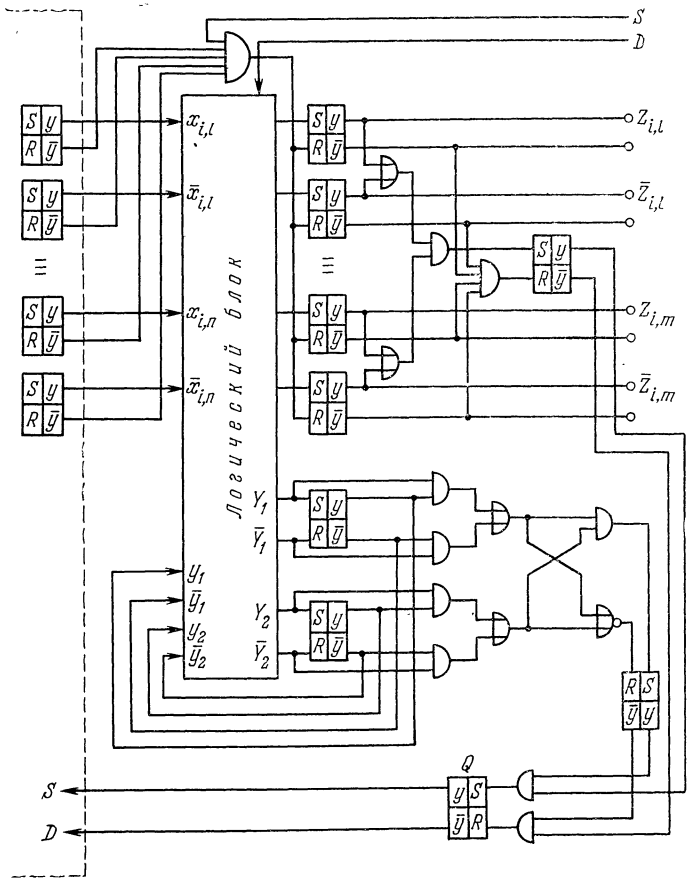


Рис. 6.9. Последовательностный модуль, пригодный для включения в цепочку.

соблюдается ли одновременно для каждого триггера условие наличия входного сигнала $S = 1$ и установки в 1 или $R = 1$ и установки в 0. Указанные два триггера управляют триггером разрешения Q, который устанавливается в 1, когда оба триггера установлены в 1, или устанавливается в 0, когда оба триггера установлены в 0. Подача промежуточного слова или слова данных из $(i-1)$-го модуля происходит, когда триггер Q установлен в 1 или 0 соответственно. Сигнал S, приходящий из $(i+1)$-го модуля, управляет установкой в 0 выходных триггеров, а сигнал D управляет всеми элементами И блока логики, как в схеме, изображенной на рис. 6.8.

В тех случаях, когда к выходу одного модуля подсоединено несколько модулей-последователей, сигналы S и D, приходящие из всех этих модулей, должны воздействовать на элементы И так, чтобы смена промежуточных слов и слов данных на выходе рассматриваемого модуля могла происходить только тогда, когда разрешение на подачу очередного слова пришло из всех модулей-последователей.

Описанные выше схемы указывают возможные решения обсуждаемых задач. Цена за получение способа функционирования, не зависящего от скорости (т. е. за построение схем, работающих правильно независимо от относительных и абсолютных скоростей работы используемых компонент), определяется примерно утроением числа применяемых компонент, и поскольку пути прохождения сигналов удлиняются за счет добавления различных схем, генерирующих сигналы завершения и определяющих типы слов, максимальная скорость работы уменьшается. Однако, если требуется длинная цепочка модулей, времена срабатывания которых значительно различаются, эффективная скорость функционирования может быть увеличена с одновременным повышением надежности в эксплуатации. Можно также показать, что существуют схемы, реализующие рассмотренный способ работы с более простой структурой.

Теория схем, не зависящих от скорости, главным образом касается реализации автономных функций (функций, не имеющих входных переменных), и далее мы покажем, как представленные здесь идеи приложимы к этой задаче. Таблица 6.4, *а* описывает автономную функцию, на-

чальным состоянием которой предполагается состояние 1. В этом примере выходная последовательность состоит из переходной последовательности и следующей за ней периодической последовательности.

Наш подход заключается в преобразовании таблицы в двухстолбцовую таблицу, воспроизводящую желаемую

Таблица 6.4

1	2,00
2	3,01
3	4,11
4	5,01
5	6,00
6	7,10
7	4,11

а) Таблица переходов для автономной функции

	x = 0	x = 1
1	***1***,00	***1′***,00
1′	2,01	***1′***,00
2	***2***,01	***2′***,01
2′	3,11	***2′***,01
3	***3***,11	3′,11
3′	4,01	***3′***,11
4	***4***,01	4′,01

	x = 0	x = 1
4′	5,00	***4′***,01
5	***5***,00	5′,00
5′	6,10	***5′***,00
6	***6***,10	6′,10
6′	7,11	***6′***,10
7	***7***,11	7′,11
7′	4,01	***7′***,11

б) Расширение таблицы а)

выходную последовательность при непрерывном изменении входного сигнала. В иллюстрирующем примере таблица 6.4, *б* представляет собой расширенный вариант таблицы 6.4, *а* и имеет два столбца и вдвое больше строк. Расширенная таблица является ОИВ-таблицей и поэтому может быть реализована в виде схемы, изображенной на рис. 6.9. Если начальное состояние таблицы 6.4, *б* равно 1—0, то многократное изменение значения x от 0 к 1 и обратно вызовет такую же выходную последовательность, какая предусмотрена исходной таблицей. При реализации таблицы 6.4, *б* в виде схемы, изображенной на рис. 6.5, выходная последовательность не будет отличаться от предусмотренной из-за отсутствия промежуточных слов, разделяющих последовательность выходных слов данных. Для завершения синтеза следует позаботиться о средствах получения непрерывно изменяющегося входного сигнала x с помощью модулей рассматриваемого типа. Решением этой задачи является замкнутая цепь из трех модулей комбинационного типа, показанная на рис. 6.10.

Выходы Z_i и $\bar{Z}_i$ каждой схемы поданы на последующую схему; каждая схема посылает в предшествующую ей схему сигналы S и D, о которых говорилось ранее. В частности, каждый модуль ведет себя как внутренний модуль системы рис. 6.7, воспроизводя на выходе промежуточные

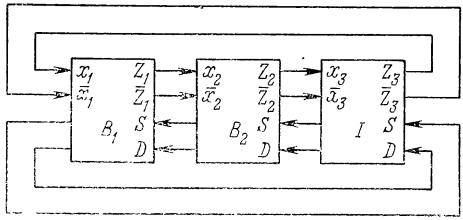


Рис. 6.10. Схема, вырабатывающая последовательность 010101...

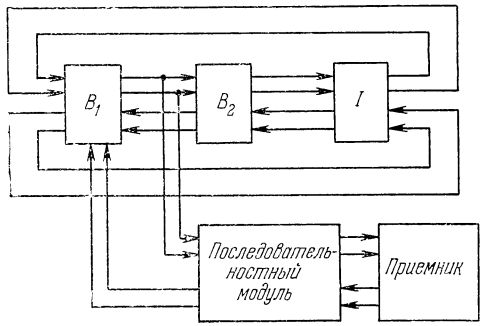


Рис. 6.11. Реализация автономной функции.

слова и слова данных при тех же ограничениях. Модуль I на рис. 6.10 представляет собой инвертор ($Z_3 = \bar{x}_3$), а модули B_1 и B_2 являются повторителями ($Z_1 = x_1$ и $Z_2 = x_2$). Изучение этой системы при допущении, что первоначально на выходах двух модулей действуют промежуточные слова, а на выходе третьего — 0 или 1, показывает: в каждый момент времени изменяется выход только одного модуля, и в каждом модуле последовательные значения выходов в словах данных равны то 0, то 1. (Поведение такой схемы напоминает работу схемы на

рис. 5.9). Например, если $Z_1 = \overline{Z}_1 = Z_2 = \overline{Z}_2 = Z_3 = 0$ и $\overline{Z}_3 = 1$, то модуль I будет запрашивать (и немедленно получит) промежуточное слово с выхода B_2, модуль B_2 запросит слово данных от B_1 и модуль B_1 запросит (и получит) слово данных от I. Следовательно, только модуль B_1 сможет изменить свой выход, и после того, как изменение произойдет, на его выходе будет слово данных $Z_1\overline{Z}_1 = 01$.

Выход любого модуля схемы, изображенной на рис. 6.10, может быть соединен с входом последовательностной схемы (рис. 6.9) для реализации таблицы 6.4, *б*. Целиком система приведена на рис. 6.11. Сигналы S и D передаются от последовательностной схемы в тот модуль, от которого она получает входные воздействия, так что система работает по представленным выше правилам.

6.2. Импульсные и синхронные схемы

В этом разделе кратко излагаются преимущества и недостатки импульсных схем и схем более общего вида, а именно, синхронных схем.

Рассмотрим матрицу переходов (табл. 6.5), определяющую функцию с тремя входными состояниями I_1, I_2 и I_3

Таблица 6.5

Таблица переходов для последовательностной функции, реализуемой с помощью импульсной схемы

	I_1	I_2	I_3	y_1	y_2
1	$3,O_2$	$1,O_2$	$3,O_2$	0	0
2	$1,O_2$	$2,O_2$	$2,O_2$	0	1
3	$2,O_1$	$2,O_2$	$3,O_2$	1	1

и двумя выходными состояниями O_1 и O_2. Наша цель — реализовать эту функцию с помощью импульсной схемы, имеющей три входных и два выходных полюса. Входной сигнал L_i представим импульсом на входном полюсе I_i, а выходной сигнал O_i — импульсом, воспроизведенным на

выходном полюсе O_i. В произвольный момент времени импульс может действовать не более чем на одном входном и на одном выходном полюсе. Любому одиночному входному импульсу должно соответствовать не более одного изменения внутреннего состояния, и между двумя входными импульсами внутреннее состояние, а также все выходные сигналы должны сохраняться неизменными (все сказанное в разделе 1.6 относительно рис. 1.3 и таблицы 1.6, *д* применимо и здесь с тем исключением, что в настоящем случае нет синхронизирующего импульса).

Блок-схема системы, которую мы пытаемся реализовать, приведена на рис. 6.12. Используя выбранный способ кодирования строк, получаем следующие выражения:

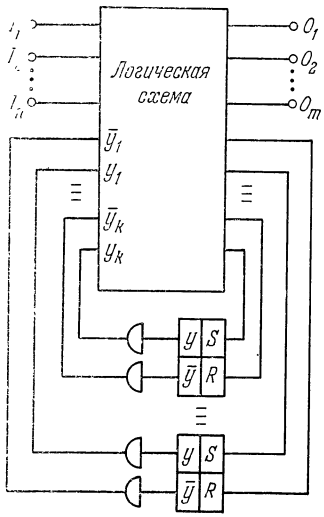


Рис. 6.12. Импульсная последовательностная схема.

$$S_1 = I_1\bar{y}_2 + I_3\bar{y}_2,$$
$$R_1 = I_1y_1 + I_2,$$
$$S_2 = I_1\bar{y}_2 + I_3,$$
$$R_2 = I_1\bar{y}_1y_2,$$
$$O_1 = I_1y_1,$$
$$O_2 = I_1\bar{y}_1 + I_2 + I_3.$$

Заметим, что каждое элементарное произведение выражений для O_1 и O_2 должно содержать символ I_i, чтобы входной импульс приводил к возникновению импульса на выходе. Основное свойство рассматриваемого импульсного способа работы состоит в том, что на кодирование состояний наложено единственное ограничение: никакое y-состояние не может быть приписано более чем одной строке таблицы.

Предположим теперь, что в рассматриваемом примере входной импульс I_1 прикладывается к системе, находящейся в начальный момент времени в состоянии 3. Тогда на триггер y_1 подается сигнал, устанавливающий триггер в 0; в результате y-состояние меняется на 01, что

соответствует строке 2 таблицы переходов. Однако, если ширина импульса I_1 настолько велика, что импульс продолжает действовать на входе и после того, как сигнал y_1 распространится через элемент задержки и начнет воздействовать на блок логики, система будет вести себя так, будто импульс I_1 вновь подан на систему, находящуюся в состоянии 2. В результате вход R триггера y_2 приобретет единичное значение, устанавливая в 0 триггер y_2 и тем самым переводя систему в состояние 1. Даже если бы состояние $2 - I_1$ было устойчивым, эта ситуация привела бы к появлению ошибочного импульса O_2. Таким образом, если следующее состояние неустойчиво или значения выходной переменной в текущем и следующем состояниях различны, длительность входного импульса должна быть меньше, чем сумма времени прохождения сигнала через логику, времени переключения триггера и времени вторичного распространения сигнала через логику.

Конечно, существует минимально допустимая длительность входного импульса, которая определяется временем, необходимым для переключения наиболее «медленного» триггера. Как станет очевидно из дальнейшего, сочетание этих двух ограничений делает обычно необходимыми элементы задержки, показанные на рис. 6.12.

Для получения количественных характеристик введем некоторые обозначения: D — номинальное значение задержки элемента задержки, Ts — номинальное время переключения триггера, W — длительность импульса, E_S — нормализованный допуск на время T_S (иными словами, время переключения отдельного триггера меняется от $((1-E_S)T_S$ до $(1+E_S)Ts)Ts)$, E_W — нормализованный допуск на длительность импульса W. Мы пренебрегаем задержками в логике, которые могут быть легко учтены позже. Тогда первое ограничение, упоминавшееся выше, имеет вид

$$W(1-E_W) < T_S(1-E_S) + D. \tag{6.1}$$

Вторым ограничением является

$$W(1-E_W) > T_S(1+E_S). \tag{6.2}$$

В наихудшем случае, когда (6.2) превращается в равенство,

$$W = \frac{T_S(1+E_S)}{1-E_W}. \tag{6.3}$$

Подставляя (6.3) в (6.1), получаем

$$\frac{D}{T_S} > \frac{2(E_S+E_W)}{1-E_W}. \tag{6.4}$$

Очевидно, что может потребоваться введение задержки, существенно большей, чем время срабатывания триггера, если времена срабатывания триггеров и длительность импульсов нельзя определить точно. Это в свою очередь означает, что входные импульсы должны следовать друг за другом с бóльшей паузой. Следовательно, чтобы добиться правильного функционирования схемы, скорость ее работы следует выбирать в зависимости от многообразия используемых компонент и скоростей их работы.

Отметим еще одно обстоятельство. В нашем примере, если к системе, находящейся в состоянии 1, приложить входной сигнал I_3, на оба триггера будут воздействовать сигналы, устанавливающие триггеры в 1. Если триггер y_2 изменит свое состояние первым и его выходной сигнал распространится через задержку и логику раньше, чем переключится триггер y_1, то сигнал S_1 может принять нулевое значение, и система придет не в состояние 1, а в состояние 2. Избежать такого критического состязания можно, если

$$(1-E_S)T_S + D > (1+E_S)\ T_S. \tag{6.5}$$

Отсюда мы получаем второе ограничение для D:

$$\frac{D}{T_S} > 2E_S. \tag{6.6}$$

Если неравенства (6.4) и (6.6) выполняются одновременно, то работу схемы определяет только неравенство (6.4).

Важно понимать, что величина T_S определяется значениями задержек во внутренних обратных связях триггеров (см. задачу 5.1); эти задержки не предназначены для выполнения каких-либо полезных функций и лишь замедляют работу схемы. D-элементы включены во внешние петли обратной связи и должны выбираться так, чтобы

удовлетворялось условие (6.4). Поскольку значения задержки этих элементов имеют разброс E_D, номинальное значение D, используемое для вычисления допустимой скорости следования входных сигналов, должно удовлетворять неравенству

$$\frac{D}{T_S} > \frac{2(E_S + E_W)}{(1 - E_W)(1 - E_D)}. \tag{6.7}$$

Зная величины D и T_S, мы можем теперь определить минимальную длительность паузы T_I между последовательными входными сигналами. Пауза должна быть такой, чтобы все изменения в схеме, вызванные предшествующим входным сигналом, полностью завершились до подачи нового входного импульса; поэтому

$$T_I > (1 + E_D)D + (1 + E_S)T_S. \tag{6.8}$$

Приведенные вычисления можно взять за основу при решении рассматриваемой задачи, а полученные результаты приводят нас к мысли о том, что, вообще говоря, скорость функционирования существенно зависит от разброса различных параметров схемы и значений задержек. Подробности, присущие схемам с тем или иным способом работы, отражаются, естественно, в различии ограничений.

Синхронные последовательностные схемы можно рассматривать совершенно аналогичным образом, считая лишь, что через W обозначена длительность синхроимпульса.

В примере, приведенном в разделе 1.6, проиллюстрирован случай, когда на входах схемы действуют двоичные сигналы, изменяющие свои значения только при отсутствии синхроимпульса. В некоторых случаях (см. рис. 1.3) используемые элементы памяти (в этом примере — триггеры со счетным входом) допускают подачу входных импульсов возбуждения с ограничениями на амплитуду, длительность и крутизну фронтов. Последнее из ограничений усложняет проблему конструирования надежных схем. При использовании емкостных связей необходимо учитывать различные временные константы, влияющие на поведение системы в разных состояниях, и поэтому проблема борьбы с паразитными импульсами

шумов становится весьма актуальной. Эти вопросы рассматриваются в монографиях по электронной технике и цифровым схемам (см., например, книги Миллмана и Тауба [78] или Мэйли и Эрла [64]).

6.3. Двухранговые последовательностные схемы

В этом разделе будет идти речь о схемах, в которых элементы задержки заменены особыми подсхемами, состоящими из нескольких триггеров, управляемых парой синхронизирующих импульсов.

Рассмотрим функцию, заданную таблицей 6.6, *а*, и блок-схему устройства, приведенную на рис. 6.13, *а*. При наличии синхроимпульса C_1 (см. задачу 6.4) логическая подсхема переводит триггеры первого ранга в новое состояние, являющееся функцией действующего входного сигнала x и состояния триггеров второго ранга. При тех же входных сигналах выходная логическая подсхема вырабатывает синфазно с C_1 выходные импульсы. Предполагается, что входной сигнал x остается постоянным, пока синхроимпульс C_1 воздействует на схему. Как показано на рис. 6.13, *б*, синхроимпульс C_2 подается на схему спустя некоторое время после снятия C_1. После этого переключаются триггеры второго ранга, переходя в те же состояния, что и триггеры первого ранга.

Любая из схем, изображенных на рис. 6.13, *в* и *г*, может быть использована для получения пары импульсов C_1 и C_2 из одного импульса C.

Основная особенность рассматриваемого способа работы заключается в том, что во время фазы 1, когда изменяет состояние подсхема первого ранга, подсхема второго ранга, управляющая этим процессом изменения, свое состояние не меняет, а во время фазы 2 происходит такой же процесс, только подсхемы первого и второго ранга меняются ролями. Необходимость в элементах задержки исчезает, поскольку в схеме нет путей обратной связи, по которым изменение состояния некоторого триггера могло бы изменить возбуждение какого-либо другого триггера. (Заметим, что импульс C_1 управляет элементами И подсхемы первого ранга таким образом, что при $C_1 = 0$ никакой триггер первого ранга не может переключиться.

Таблица 6.6

	x_1x_2 00	01	11	10	y_1	y_2	y_3
1	3,0	5,0	2,0	1,0	0	0	0
2	2,0	5,0	4,0	1,0	0	0	1
3	5,0	4,0	3,0	3,0	0	1	0
4	5,0	4,0	5,0	4,0	0	1	1
5	3,0	2,0	2,1	3,0	1	0	0

а) Матрица переходов

	x_1x_2 00	01	11	10	y_1	y_2	y_3
1	$\overline{S}S\overline{S}$	$S\overline{S}\overline{S}$	$\overline{S}S\overline{S}$	$S\overline{S}\overline{S}$	0	0	0
2	$\overline{S}S\overline{R}$	$S\overline{S}R$	$\overline{S}S\overline{R}$	$S\overline{S}R$	0	0	1
3	$SR\overline{S}$	$\overline{S}RS$	$\overline{S}R\overline{S}$	$\overline{S}RS$	0	1	0
4	SRR	$\overline{S}R\overline{R}$	SRR	$\overline{S}R\overline{R}$	0	1	1
5	$RS\overline{S}$	$R\overline{S}S$	$R\overline{S}S$	$RS\overline{S}$	1	0	0

б) Матрица изменений

	x_1x_2 00	01	11	10	y_1	y_2	y_3
1	0—100—	100—0—	0—0—10	0—0—0—	0	0	0
2	0—0——0	100—01	0—10—0	0—0—01	0	0	1
3	10010—	0——010	0——00—	0——00—	0	1	0
4	100101	0——0—0	100101	0——0—0	0	1	1
5	01100—	010—10	010—10	01100—	1	0	0
		S_1R_1	S_2R_2	S_3R_3			

в) Матрица возбуждения триггеров

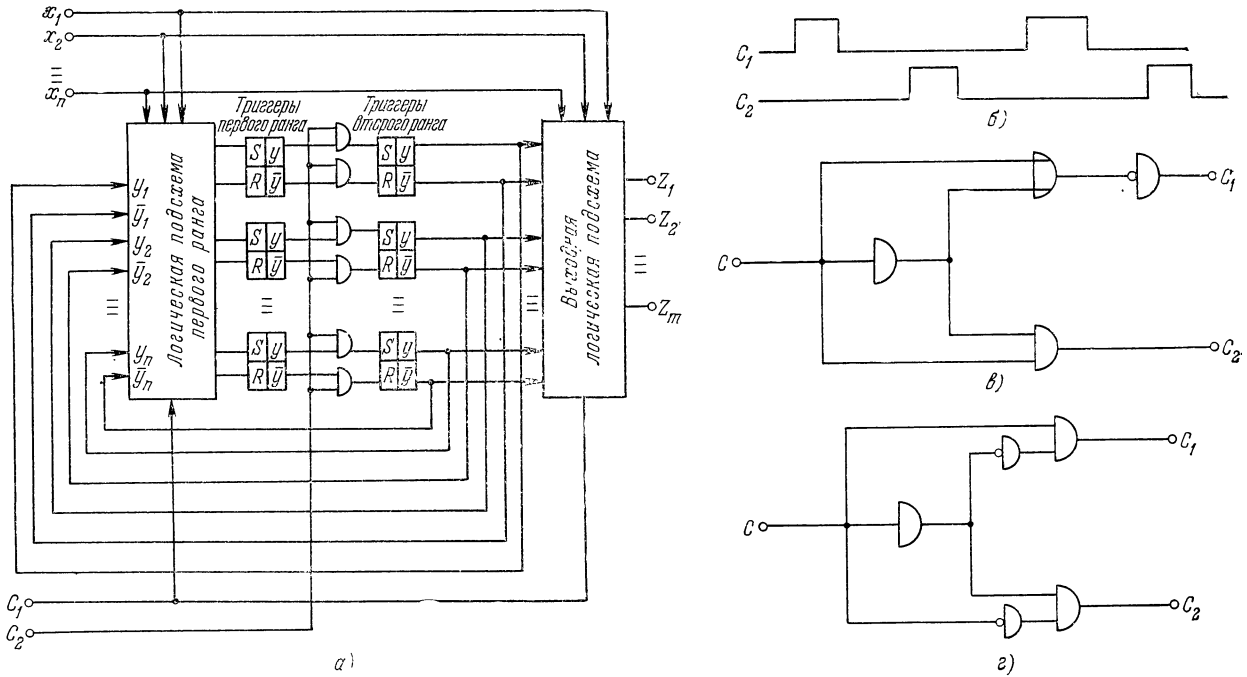


Рис. 6.13. *а*) Двухранговая схема первого типа. *б*) Фазовая диаграмма синхроимпульсов. *в*) Первый вариант схемы получения двух синхроимпульсов из одного. *г*) Второй вариант схемы получения двух синхроимпульсов из одного.

Аналогичным образом импульс C_2 управляет выходной подсхемой).

Длительность импульса синхронизации должна быть достаточно большой, чтобы успели переключиться наиболее медленно действующие триггеры. В противоположность ситуации, описанной в предыдущем разделе, максимальную длительность синхроимпульса можно не ограничивать при единственном условии: импульсы C_1 и C_2 не должны подаваться на схему одновременно. Импульс C_2 и триггеры второго ранга играют роль элементов задержки, используемых в импульсных или синхронных схемах, рассмотренных в предыдущем разделе.

При синтезе логики изучаемой схемы комбинационные состязания можно не принимать во внимание, так как все выходные сигналы и импульсы, прикладываемые к триггерам, являются следствием действия синхроимпульсов, подаваемых на схемы, все входные сигналы которых имеют постоянные значения на протяжении всего интервала действия синхроимпульсов.

Процесс синтеза двухранговых схем совершенно не отличается от синтеза схем с одним синхроимпульсом. Сначала выбирается способ кодирования строк таблицы переходов (единственное требование: каждой строке должно быть приписано единственное состояние). Вообще говоря, различные способы кодирования влияют на сложность синтезируемой логики, но методы выбора наилучших кодирований, разработанные для синхронных схем с одним синхроимпульсом, пригодны и здесь. Подходящее кодирование для нашего примера указано в таблице 6.6, *а*. Следует также подумать о способе кодирования состояний подсхемы второго ранга.

Введем полезное для построения схем с RS-триггерами понятие *матрицы изменений*, обобщение которого понадобится нам в дальнейшем. Матрица изменений для нашего примера приведена в таблице 6.6, *б*. В ней отражены сигналы возбуждения, которые следует подавать на каждый y-триггер, чтобы переходы из любого полного состояния в следующее состояние совершались согласно матрице переходов. Для каждого триггера существует четыре возможности.

1. Если $y_i = 0$ в текущем состоянии и $y_i = 1$ в следующем состоянии, то должен быть возбужден вход

установки триггера в 1, поэтому элементом матрицы является S.

2. Если $y_i = 1$ в текущем состоянии и $y_i = 0$ в следующем состоянии, то в качестве элемента матрицы выбирается R, т. е. необходима подача сигнала установки в 0.

3. Если $y_i = 0$ в текущем и следующем состояниях, на вход триггера может быть подан сигнал R, однако вход S не должен быть возбужден. Этот факт отражается выбором символа $\bar{S}$.

4. Если y_i равно 1 и должно сохранить это значение в следующем состоянии, подача сигнала R недопустима, так что элементом матрицы является $\bar{R}$.

С помощью матрицы изменений легко построить матрицу возбуждения триггеров (для нашего примера см. таблицу 6.6, *в*). В таблице записаны пары значений SR для каждой y-переменной, причем комбинации 10, 01, 0— и —0 соответствуют случаям 1, 2, 3 и 4. Обычным способом по таблице 6.6, *в* получаем аналитические выражения для сигналов возбуждения триггеров первого ранга (C_1 добавляется в каждое элементарное произведение):

$$S_1 = \bar{x}_1\bar{x}_2y_2C_1 + x_1\bar{x}_2\bar{y}_1\bar{y}_2C_1 + x_1x_2y_2y_3C_1,$$

$$R_1 = y_1C_1,$$

$$S_2 = \bar{x}_1\bar{x}_2\bar{y}_2\bar{y}_3C_1 + x_1x_2\bar{y}_2y_3C_1 + \bar{x}_2\bar{y}_1C_1,$$

$$R_2 = \bar{x}_1\bar{x}_2y_2C_1 + x_1x_2y_2y_3C_1,$$

$$S_3 = x_2y_1C_1 + \bar{x}_1x_2y_2C_1 + x_1x_2\bar{y}_1\bar{y}_2C_1,$$

$$R_3 = \bar{x}_1\bar{x}_2y_2C_1 + \bar{x}_1x_2\bar{y}_1\bar{y}_2C_1 + x_1x_2y_1C_1 + x_1\bar{x}_2\bar{y}_1\bar{y}_2C_1.$$

Функция, реализуемая выходной логической подсхемой, получается по таблице 6.6, *а* и имеет вид (C_1 является входным сигналом для элемента И)

$$Z = x_1x_2y_1C_1.$$

В общем случае необходимо также построить специальную схему для установки последовательностной схемы в начальное состояние. Поскольку в рассмотренном выше случае начальное состояние определяется только состоянием триггеров второго ранга, представляется возможным и желательным уметь приводить такую си-

стему в начальное состояние лишь за счет приведения в начальное состояние триггеров второго ранга. Однако это сделать невозможно, если схема уже построена, из-за того, что в каждом полном состоянии при наличии импульса C_1 возбуждаются не все триггеры первого ранга. Например, триггер y_1 не может быть установлен ни в 1, ни в 0 в состоянии 1—00. Это означает, что для состояний такого рода следующее состояние триггеров первого ранга частично зависит от предыдущего состояния этих триггеров, а не только от входного сигнала и состояния триггеров второго ранга. Если же мы в конкретном случае убеждаемся в том, что каждый триггер первого ранга возбуждается всякий раз, когда подается импульс C_1, то достаточно приводить в начальное состояние только триггеры второго ранга, так как триггеры первого ранга приобретут правильные значения после подачи начального импульса C_1, который по предположению предшествует первому импульсу C_2. Платой за такое свойство системы, называемое *свойством самосинхронизации*, является некоторая потеря свободы при построении логики первого ранга, что в свою очередь часто приводит к несколько более сложному схемному решению. В матрице возбуждения триггеров неопределенные значения их входов R (или S) должны быть доопределены как отрицания значений входов S (или R) тех же триггеров. Таким образом, функции возбуждения S и R каждого триггера являются инверсными по отношению друг к другу.

Часто можно сократить число триггеров первого ранга, заменяя, как показано на рис. 6.14, второй логической схемой элементы И, управляемые синхроимпульсом C_2. Поскольку x-сигналы не подаются на этот блок, разным заполнениям клеток матрицы возбуждения триггеров второго ранга должны соответствовать разные состояния триггеров первого ранга. Если мы задались целью построить схему с минимальным числом триггеров, то в матрице возбуждения следует найти минимальное множество значений содержимого ее клеток*) такое, чтобы каждому значению клеток матрицы изменений соответ-

*) В дальнейшем ради краткости вместо термина «значение содержимого клетки» будем употреблять «значение клетки». (*Прим. ред.*)

ствовал хотя бы один элемент этого множества. Например, значениям $\bar{S}SR$, $\bar{S}S\bar{S}$, $RS\bar{S}$, RSR клеток матрицы изменений соответствует один элемент 011001. Два значения клеток матрицы изменений могут быть названы *совместимыми*, если в них нет позиций, в которых значение первой клетки равно S, а второй — $\bar{S}$ или R, и нет

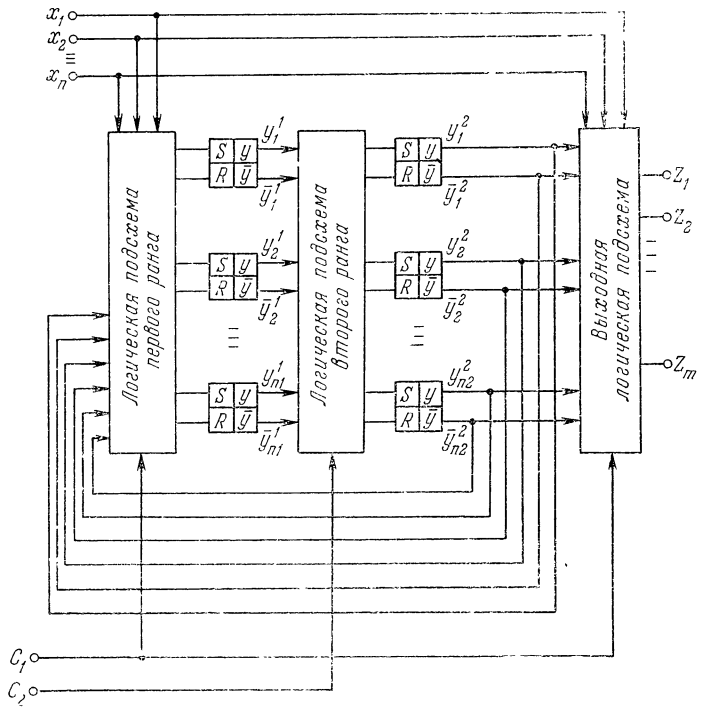


Рис. 6.14. Двухранговая схема второго типа.

позиций, в которых значение первой клетки равно R, а второй — $\bar{R}$. Нетрудно заметить, что множеству взаимно совместимых значений клеток матрицы изменений может соответствовать одно и то же значение клеток матрицы возбуждения. Таким образом, значениям $\bar{S}S\bar{R}$, $R\bar{S}S$ и $\bar{S}\bar{R}S$ из таблицы 6.6, *б* соответствует элемент 010010.

Поскольку этот вид совместимости аналогичен принципу совместимости, введенному при изучении в разделе 2.2 минимизации таблиц переходов, то представленные ранее методы нахождения максимальных совместимых множеств применимы и здесь. После нахождения максимальных совместимых множеств строится минимальное множество МС-множеств так, чтобы каждое значение клеток матрицы изменений покрывалось по крайней мере одним элементом этого множества — вопрос, изучавшийся в подразделе 3.3.2, при обсуждении кодирования Трэйси. Для каждого МС-множества будет получено одно значение клеток матрицы возбуждения, возможно, не полностью определенное. При заданном кодировании состояний подсхемы второго ранга предложенная последовательность действий приводит нас к минимальному числу триггеров первого ранга.

Проиллюстрируем эту процедуру, реализовав таблицу 6.6, *а* в виде схемы, изображенной на рис. 6.14. Прежде чем начать построение карты пар для значений клеток таблицы 6.6, *б*, отметим, что некоторые значения можно не рассматривать; например, любое возбуждение, соответствующее значению RRS, также соответствует значению $R\bar{S}S$. Поэтому, если RRS и $R\bar{S}S$ являются значениями клеток матрицы изменений, то нет нужды рассматривать последнее из них при построении минимального множества значений. Вообще говоря, значение e_2 доминирует над значением e_1, и, следовательно, e_1 может не приниматься во внимание при построении минимального множества значений, если компоненты e_1 и e_2 совпадают, за исключением некоторых компонент $\bar{R}$ или $\bar{S}$ значения e_1, соответствующих компонентам S или R значения e_2.

Применяя это свойство к таблице 6.6, *б*, получаем значения, над которыми не доминируют никакие другие: SRR, $RS\bar{S}$, $\bar{S}R\bar{S}$, $R\bar{S}S$, $\bar{S}S\bar{R}$, $\bar{S}\bar{S}R$. С помощью карты пар (табл. 6.7, *а*), где эти значения отмечены буквами *a*, *b*, *c*, *d*, *e* и *f* соответственно, находим максимальные совместимые множества: *a*, *be*, *cd*, *ce*, *f*. Минимальным покрывающим множеством является $\{a, be, cd, f\}$. Следующие значения клеток матрицы возбуждений соответствуют элементам этого множества: 100101, 011000, 010010 и 0—0—01. Каждое значение клеток таблицы 6.6, *б* определяется одним из указанных возбуждений: обозначая их

Таблица 6.7

a SRR	b $RS\bar{S}$	c $\bar{S}R\bar{S}$	d $R\bar{S}S$	e $\bar{S}S\bar{R}$	f $\bar{S}S\bar{R}$
×					
×	×				
×	×				
×			×		
×	×	×	×	×	

а) Карта пар для таблицы 6.6,*б*

	x_1x_2 00	01	11	10	y_1^1	y_2^1	y_3^1
1	e_2	e_1	e_3	e_4	0	0	0
2	e_3	e_1	e_2	e_4	0	0	1
3	e_1	e_3	e_2	e_2	0	1	0
4	e_1	e_2	e_1	e_2	0	1	1
5	e_2	e_3	e_3	e_2	1	0	0

б) Матрица возбуждения для таблицы 6.6, *б* с минимальным числом различных возбуждений

	y_1^1	y_2^1	S_1^2	R_1^2	S_2^2	R_2^2	S_3^2	R_3^2
e_1	0	0	1	0	0	1	0	1
e_2	0	1	0	1	1	0	0	0
e_3	1	0	0	1	0	0	1	0
e_4	1	1	0	—	0	—	0	1

в) Задание на синтез логической подсхемы второго ранга

x_1x_2 00	01	11	10	y_1^2	y_2^2	y_3^2
0110	0101	1001	1010	0	0	0
1001	0101	0110	1010	0	0	1
0101	1001	0110	0110	0	1	0
0101	0110	0101	0110	0	1	1
0110	1001	1001	0110	1	0	0

$S_1^1R_1^1S_2^1R_2^1$

г) Задание на синтез логической подсхемы первого ранга

через e_1, e_2, e_3 и e_4, мы получим таблицу 6.7, *б* (теперь переменные разных рангов удобно различать с помощью верхних индексов). Если теперь закодировать e_1, e_2, e_3 и e_4 значениями переменных $y_1^1 y_2^1$ первого ранга в виде 00, 01, 10 и 11 соответственно (возможны и другие кодирования, хотя сложность синтезируемой логики при этом может измениться), мы получим задание на синтез логической подсхемы второго ранга в виде таблицы 6.7, *в*. Логическую подсхему первого ранга задает таблица 6.7, *г*. Отметим, что y-переменные первого ранга сильно напоминают q-переменные, используемые при построении схем с минимальным числом обратных связей (см. раздел 5.3).

Логическая подсхема второго ранга задается следующими выражениями:

$$S_1^2 = \bar{y}_1^1 \bar{y}_2^1 C_2,$$
$$R_1^2 = y_1^1 C_2 + y_2^1 C_2,$$
$$S_2^2 = \bar{y}_1^1 y_2^1 C_2,$$
$$R_2^2 = \bar{y}_1^1 \bar{y}_2^1 C_2,$$
$$S_3^2 = y_1^1 \bar{y}_2^1 C_2,$$
$$R_3^2 = \bar{y}_1^1 \bar{y}_2^1 C_2 + y_1^1 y_2^1 C_2.$$

Для логики первого ранга такие выражения имеют вид:

$$S_1^1 = \bar{x}_2 \bar{y}_2^2 y_3^2 C_1 + \bar{x}_1 x_2 y_2^2 \bar{y}_3^2 C_1 + x_2 y_1^2 C_1 + x_1 \bar{y}_1^2 \bar{y}_2^2 \bar{y}_3^2 C_1,$$
$$R_1^1 = \bar{x}_1 \bar{x}_2 \bar{y}_3^2 C_1 + \bar{x}_2 y_1^2 C_1 + y_2^2 y_3^2 C_1 + \bar{x}_1 x_2 \bar{y}_1^2 \bar{y}_2^2 C_1 + x_1 y_2^2 C_1 + x_2 y_3^2 C_1,$$
$$S_2^1 = \bar{x}_2 \bar{y}_2^2 \bar{y}_3^2 C_1 + \bar{x}_1 x_2 y_2^2 y_3^2 C_1 + x_1 \bar{y}_2^2 \bar{y}_3^2 C_1 + x_1 \bar{x}_2 C_1 + x_1 \bar{y}_2^2 y_3^2 C_1,$$
$$R_2^1 = \bar{x}_1 \bar{x}_2 y_3^2 C_1 + \bar{x}_1 y_2^2 \bar{y}_3^2 C_1 + \bar{x}_1 x_2 \bar{y}_2^2 C_1 + x_1 x_2 \bar{y}_2^2 \bar{y}_3^2 C_1 + x_1 x_2 y_2^2 y_3^2 C_1.$$

Дальнейшим развитием двухранговых схем должно явиться введение x-входов в логическую подсхему второго ранга. Возможно, что в результате такой операции логи-

ческая схема упростится и потребуется меньше триггеров первого ранга, однако добавится ограничение, в силу которого входные изменения не должны происходить во время действия как синхроимпульса C_1, так и C_2.

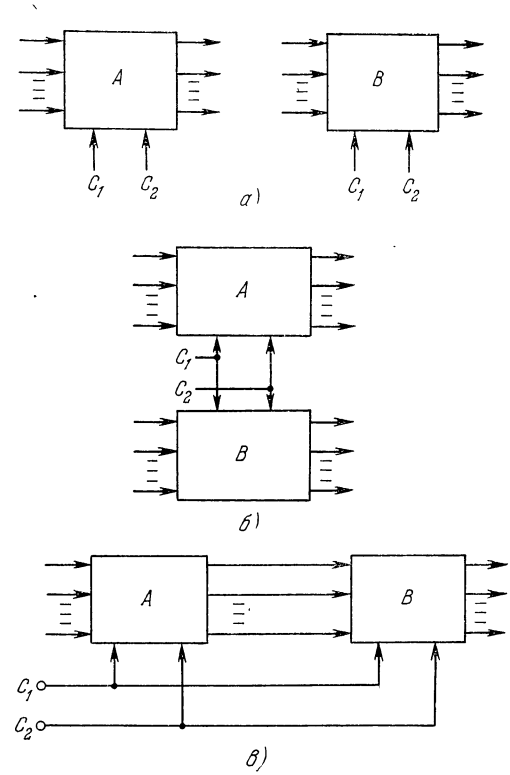


Рис. 6.15. *а*) Двухранговые схемы A и B. *б*) Параллельное соединение схем A и B. *в*) Последовательное соединение схем A и B.

Интересная особенность рассмотренных схем состоит в том, что они легко могут быть соединены как последовательно, так и параллельно, образуя более крупные двухранговые системы (см. рис. 6.15). Это свойство особенно полезно для синтеза итеративных схем, например, счетчиков и регистров сдвига.

ЗАМЕЧАНИЯ ПО БИБЛИОГРАФИИ

Идея создания схем, не зависящих от скорости, в которых сигналы завершения устраняют необходимость вести синтез схем в предположении наихудшего случая, оценивая максимальное значение внутренних задержек, принадлежит Мюллеру и его коллегам [83—86, 88, 42, 97, 16].

В этих работах, непростых для чтения, рассматривались только автономные схемы, а общих методов синтеза не приведено. Глава 10 книги Миллера [80], возможно, является более простым пособием для изучения схем Мюллера. Лаконичное изложение этих результатов дано Холлом [37]. Отметим ссылку в разделе 6.1 на работу Гилхриста, Померайна и Вонга [27], в которой сигналы завершения использованы при построении сумматоров, и обобщение этой работы Вейтом [117]. Материал раздела 6.1, касающийся синтеза, допустимых входных сигналов и связи подходов Мюллера и Хаффмана, заимствован у Армстронга, Фридмана и Менона [1]. Симз и Грей [100] и Макнотон [76] также изучали различные аспекты этих вопросов, причем последняя работа является исчерпывающим исследованием.

Решение проблем, связанных с временными характеристиками синхронных и импульсных схем (раздел 6.2), дано Ангером [110], хотя конструкторы-практики могут выполнить аналогичные вычисления с большим искусством.

Общий подход к этой проблеме можно найти в различных книгах [74, 80, 119]. Более подробная трактовка инженерных аспектов представлена, как отмечалось ранее, в [64, 78].

Идея двухранговых схем, представленная в разделе 6.3, введена Воре [118] применительно к счетчикам и регистрам сдвига. Примерно в то же время, но в другом контексте Хаффман [49] рассмотрел похожее понятие в более общем виде, используя термин «спусковые схемы». Некоторые идеи, представленные здесь, заимствованы у Холла [38, 40].

Два интересных способа работы, не рассматривавшиеся здесь, были развиты в работах Джерачи и Джестри [24, 25] и Эйхельбергера [13, 14].

ЗАДАЧИ

6.1*. Найти подходящий способ кодирования строк для таблицы 4.9 и выписать логические выражения для блока логики схемы Мюллера, изображенной на рис. 6.5 и реализующей эту таблицу.

6.2. Разновидностью триггера, часто используемого в синхронных системах, является JK-триггер, имеющий три входа. Один из них — вход синхронизации C, а два других — вход S установки в 1 и вход R установки в 0. Изменение состояния JK-триггера может произойти только при наличии импульса C. Если в этот момент на входе S (или R) действует единичный сигнал, то выходным состоянием является установка в 1 (или 0). Если при наличии C возбуждены оба входа S и R, то триггер изменяет свое состояние. Таким образом, JK-триггер сочетает свойства RS-триггера и триггера со счетным входом. Реализовать таблицу 1.6, *д*, используя JK-триггеры.

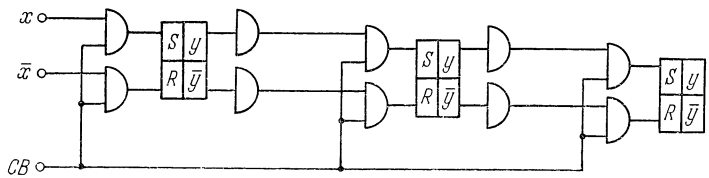


Рис. 6.16.

6.3. Трехразрядный синхронный регистр сдвига изображен на рис. 6.16. Построить трехразрядный двухранговый регистр сдвига, используя минимальное число триггеров.

6.4. Построить таблицы переходов для схем, изображенных на рис. 6.13, *в* и *г*, служащих для получения двух синхроимпульсов из одного.

ГЛАВА 7
СЧЕТЧИКИ

Двоичные счетчики давно уже входят как компоненты в цифровые системы различных типов. Еще до того, как подобные системы начали бурно развиваться, двоичные электрические счетчики были важным инструментом в экспериментальной физике. Такая сравнительно длинная (по отношению к молодой еще области исследований) технологическая «предыстория» и широкая область применения обусловили то, что в проектировании и построении таких устройств был достигнут значительный прогресс и разработано довольно много разнообразных конструкций.

Хотя здесь мы и рассматриваем детально несколько различных вариантов счетчиков, наша цель не в том, чтобы дать обзор достижений в этой области. На примере этих устройств нам хотелось бы проиллюстрировать большинство из методов и принципов, изложенных в предыдущих главах.

Понятие счетчика допускает несколько интерпретаций. В каждой из них рассматривается схема с одним входом, вырабатывающая для целого положительного n в зависимости от принятой интерпретации:

1) выходной сигнал после каждого n-го входного сигнала;

2) различные выходные сигналы после каждого из первых n символов, после чего последовательность выходных сигналов повторяется при последующей подаче входных последовательностей из n символов;

3) выходные сигналы, являющиеся двоичными представлениями чисел от 0 до $n-1$ для первых n выходных символов.

Мы сфокусируем наше внимание на третьем определении, являющемся частным случаем второго. В наших примерах n будет степенью числа 2, хотя очень просто

преобразовать логику таким образом, чтобы сделать счетчик циклическим для любой желаемой длины цикла.

Для понимания работы различных рассматриваемых здесь счетчиков необходимо четко представлять, как меняются значения разрядов в двоичном представлении

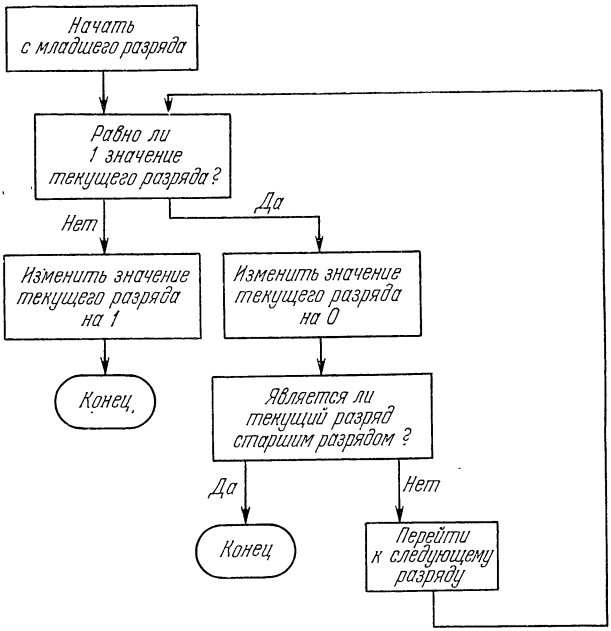


Рис. 7.1. Блок-схема работы счетчика.

числа при увеличении числа на 1. Для лучшего представления этого процесса можно воспользоваться блок-схемой алгоритма, показанной на рис. 7.1, и мы советуем читателю внимательно ее изучить.

7.1. Счетчик «Домино»

Один из самых первых и простейших видов двоичных электрических счетчиков, показанный на рис. 7.2, состоит из «линейки» триггеров со счетным входом (см. раздел 1.6), связанных через конденсаторы, причем входной

сигнал A поступает лишь на первый триггер этой линейки. Двоичные сигналы, снимаемые с y-выходов этих триггеров, образуют выходной сигнал.

При рассмотрении работы счетчика будем предполагать, что логическому значению 1 соответствует большее значение положительного напряжения и что триггер реагирует только на положительные импульсы. При каждом поступлении на его вход импульса A триггер меняет свое состояние. Когда $\bar{y}_1$ меняется от 1 до 0, через конденсатор передается отрицательный импульс на триггер τ_2 и, по допущению, не оказывает на τ_2 никакого влияния. Но когда триггер τ_1 возвращается к первоначальному состоянию, так что $\bar{y}_1$ меняется на 1, на τ_2 поступает положительный импульс, приводящий к изменению состояния

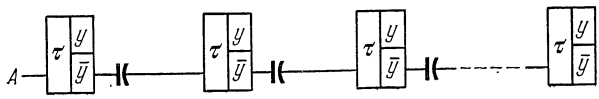


Рис. 7.2. Счетчик «домино».

триггера τ_2. Таким образом, y_2 меняет свое значение всякий раз, когда входной импульс вызывает изменение y_1 из 1 в 0. Аналогичная ситуация повторяется для триггера τ_2. Допустим, что первые i триггеров представляют i младших двоичных разрядов в выходном сигнале счетчика. Тогда $(i+1)$-й триггер получит входной импульс лишь тогда, когда на входе счетчика имеется входной импульс A, а каждый из первых i триггеров находится в состоянии 1. Именно так и должен вести себя этот триггер, если его выход y представляет $(i+1)$-й разряд счетчика, тогда по индукции получаем, что описанная схема действительно является счетчиком. При наличии m каскадов максимальным представляемым числом является 2^m-1. Название «домино» обусловлено тем обстоятельством, что, если k младших триггеров находятся в состояниях 1, приходящий входной сигнал «сваливает» первый триггер (в состояние 0), который, в свою очередь, «сваливает» следующий и т. д., что подобно поведению поставленных рядом костяшек домино, если толкнуть первую из них.

Заметим, что передача единицы в m-каскадном счетчике может потребовать по времени увеличенной в m раз длительности переходного процесса в отдельном триггере и связанном с ним конденсаторе. Если необходимо выполнять какие-либо операции с учетом внутреннего состояния счетчика, то последовательные входные импульсы должны быть разделены интервалом, величина которого не меньше вышеприведенного значения. Если же нас интересует показание счетчика после серии поданных входных импульсов (как, например, в случае, когда желательно знать число поданных импульсов), то импульсы в такой серии должны быть разделены интервалами, равными длительности переходного процесса лишь в самом «медленном» триггере и связанном с ним конденсаторе. Это объясняется тем, что после того, как первый импульс переключил первый триггер и, возможно, возбудил цепочку последующих переключений, второй импульс можно подавать и при наличии таких переключений. Указанное минимальное разделение гарантирует, что действие последующего импульса не «наложится» на действие предыдущего импульса.

В задачах, возникающих при построении подобных счетчиков, в частности, в тех, которые касаются процессов зарядки и разрядки связующих конденсаторов в плане получения импульсов нужной формы и величины, требуется, как правило, уменьшение скорости функционирования, если должна быть гарантирована надежность функционирования. Это требование усложняется еще и тем, что такие схемы имеют каскадную структуру.

7.2. Импульсные счетчики с параллельным управлением

При использовании в качестве базисных тех же триггеров, что и в рассмотренном выше счетчике, и при замене связующих конденсаторов логическими элементами можно так перестроить схему, что все триггеры, которые, по предположению, должны переключаться, получают входные импульсы одновременно (если не рассматривать логические задержки). Такой счетчик показан на рис. 7.3, *а*; ясно, что приходящий импульс доходит до некоторого

триггера тогда и только тогда, когда каждый из более левых по отношению к нему триггеров находится в состоянии 1. Логические элементы необходимы по причине, изложенной в разделе 6.2, а именно — чтобы предотвратить влияние переключения триггера на последующие элементы схемы на временном интервале присутствия входного импульса. Рассмотрение, проведенное для общего случая

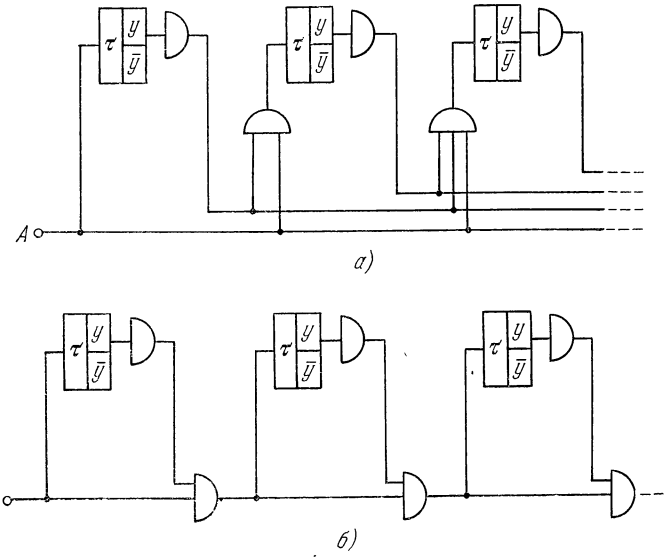


Рис. 7.3. *а*) Счетчик с параллельным управлением при минимальной длине путей прохождения сигналов. *б*) Счетчик с параллельным управлением при минимальном числе разветвлений.

в разделе 6.2, определяет ограничения на ширину приходящих импульсов, значения задержек и т. п. Настоящая схема работает на значительно большей скорости, чем предыдущий счетчик.

Имеется, однако, ряд серьезных проблем, связанных с разветвлением внешних и внутренних сигналов в схеме и присущих счетчикам этого типа, когда число каскадов m превышает 4 или 5. Элемент И для k-го триггера имеет k входов, приходящий входной импульс поступает на m схем и выходной сигнал от k-го триггера поступает

на $m - k$ схем. Одно из решений, свободных от указанных недостатков, показано на рис. 7.3, *б* с последовательным соединением одинаковых подсхем, причем число входов и ветвей разветвления выхода каждого элемента не превышает двух.

Такое решение приводит, однако, к двум новым трудностям: к снижению скорости операций, связанному с удлинением путей для прохождения сигналов, и к затуханию входного сигнала, если элементы И не содержат усилительных элементов. Возможен, конечно, компромисс между схемами рис. 7.3, *а* и *б*, при котором схемы удовлетворяют требуемым ограничениям на число входов и ветвей разветвления выхода каждого элемента (см. задачу 7.1), такие «компромиссные» схемы широко используются.

Недостатки импульсных счетчиков такие же, которые, как отмечено в разделе 6.2, присущи широкому классу импульсных и синхронных схем.

7.3. Схемы, реагирующие на задние фронты

Схемы из предыдущего раздела можно было бы сделать асинхронными, если использовать триггеры со счетным входом, которые ведут себя согласно таблице 7.1; это

Таблица 7.1

Таблица переходов асинхронного триггера со счетным входом

	τ	
	0	1
1	***1***,01	2,01
2	3,10	***2***,01
3	***3***,10	4,10
4	1,01	***4***,10

избавило бы от необходимости находить верхнюю оценку ширины приходящих импульсов. Такой асинхронный триггер со счетным входом характеризуется тем, что его выход обусловлен только задним фронтом входного импульса,

так что при изменении $0 \longrightarrow 1$ выход триггера остается неизменным и меняется лишь при прохождении заднего фронта импульса (при изменении входа от 1 до 0).

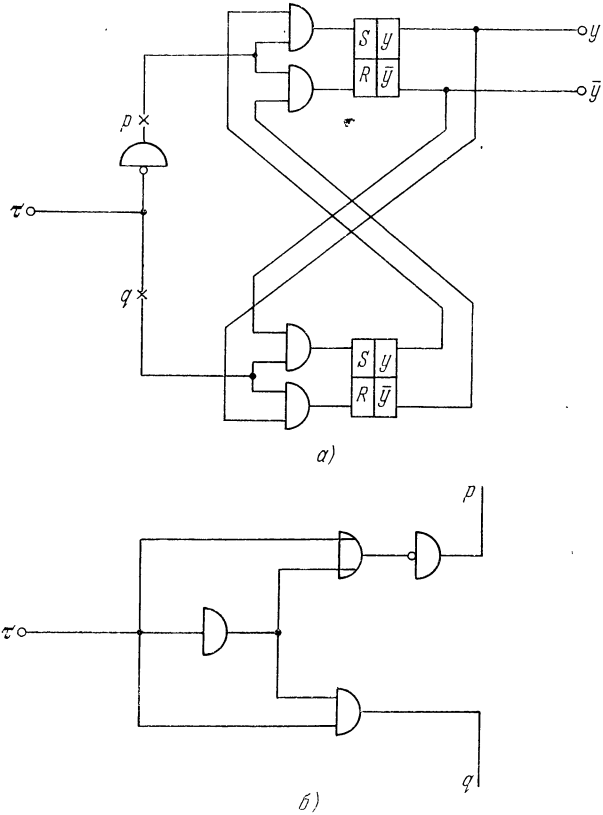


Рис. 7.4. *а*) Асинхронный триггер со счетным входом. *б*) Входная схема для устранения существенных состязаний в схеме *а*).

Если используются такие триггеры, то безотносительно к тому, как долго подается сигнал A на какую-либо из схем рис. 7.3, входные τ-сигналы получат лишь те триггеры, которые, по предположению, должны измениться, поскольку и моменту изменения значений выходов

триггеров входной сигнал A примет значение 0. При этом можно учесть и существенные состязания, если быть уверенным, что для каждого каскада задержка в пути через триггер больше задержки в пути,

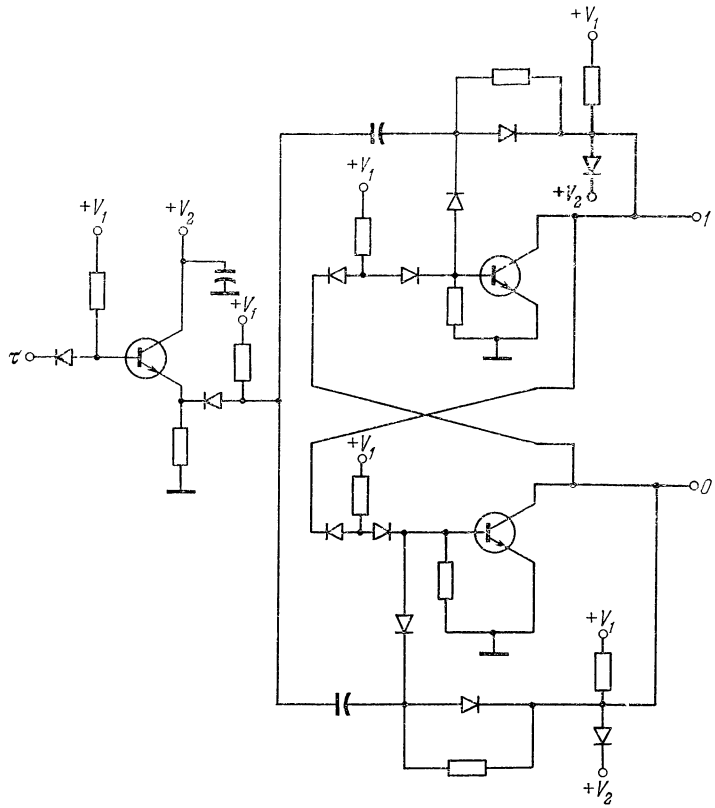


Рис. 7.5. Триггер, срабатывающий от задних фронтов входных импульсов.

минующем триггер. Это достигается с помощью элемента задержки на выходе каждого триггера (хотя такое включение задержки может быть и не обязательным), и кроме них не требуется никаких других внешних (по отношению к триггерам) элементов задержки.

Асинхронные триггеры можно синтезировать точным методом. Одна из реализаций уже была представлена (см. задачу 5.2). Другая реализация, включающая пару RS-триггеров, показана на рис. 7.4, *а*. Если $\tau = 0$, то состояние верхнего триггера повторяет состояние нижнего, если же $\tau = 1$, то состояние нижнего триггера противоположно состоянию верхнего триггера. Для устранения существенных состязаний, присущих таблице 7.1, внутри одного из триггеров можно использовать инерциальную задержку. Другой подход, который также устраняет необходимость внешних элементов задержки в счетчике, состоит в обеспечении определенного интервала, на котором оба сигнала p и q равны 0 после каждого изменения τ. Это достигается с помощью схемы рис. 7.4, *б* (см. также рис. 6.13, *в* и *г*), помещаемой между узлами τ, p и q вместо инвертора в схеме на рис. 7.4, *а*.

Читателям, знакомым с электроникой, вероятно, интересно будет проанализировать триггер, показанный на рис. 7.5, являющийся реализацией таблицы 7.1 с помощью только трех транзисторов. Счетчики, использующие схемы такого типа, считаются практичными.

7.4. Счетчик Грея

Допустим теперь, что нам желательно разработать счетчик, в скорости и надежности которого используются преимущества асинхронных операций. Непосредственный подход состоит в построении соответствующей таблицы переходов и выборе подходящего варианта кодирования строк. В последней части подраздела 4.3.3 это проделано для счетчика с четырьмя состояниями (см. табл. 4.10). Для кодирования используется код Грея; подобный процесс применим при построении счетчиков, максимальное вычисляемое число для которых является некоторой степенью числа 2. Выше было отмечено, что в схеме, построенной по матрице переходов с таким кодированием, необходим лишь один элемент задержки, а именно — инерциальная задержка, связанная с y-переменной, имеющей наибольшую частоту изменения.

К сожалению, такой подход не позволяет получать простые схемы, которые можно было бы соединять в произвольное число каскадов. Более того, поскольку выход-

ные переменные не соответствуют внутренним переменным (кроме переменной старшего разряда), для получения выходных значений в двоичном коде необходима специальная логика. При реализации счетчика в итеративной форме (схема преобразования кода Грея в двоичный код), являющейся одной из наиболее дешевых, схема содержит длинные последовательности логических элементов, что, конечно, приводит к существенному снижению эффективной скорости операций. По этой причине счетчики подобного типа если и используются, то крайне редко.

7.5. Счетчик Воре

Подход, изложенный в разделе 7.4, можно модифицировать с учетом следующих обстоятельств. Показанный в таблице 7.2 вариант кодирования является кодированием по Лю (см. подраздел 3.3.2), в котором частичные

Таблица 7.2

Матрица переходов счетчика Воре

	A					
	0	1	y_2^0	y_1^0	y_2^1	y_1^1
1	***1***	2	0	0	0	0
2	3	***2***	0	1	0	0
3	***3***	4	0	1	0	1
4	5	***4***	1	0	0	1
5	***5***	6	1	0	1	0
6	7	***6***	1	1	1	0
7	***7***	8	1	1	1	1
8	1	***8***	0	0	1	1

состояния (подкубы), приписанные множествам предшественников каждого столбца, расположены в порядке неубывания двоичных чисел.

Таким образом, устранена необходимость в логике для выработки выходных сигналов, поскольку выходные сигналы можно снимать непосредственно с триггеров, представляющих y-переменные, выбранные для столбца 0.

Пусть в качестве элементов памяти используются RS-триггеры, тогда можно заметить, что при использовании кодирования по Лю, как в нашем случае, переменные, связанные с данным столбцом, остаются устойчивыми в этом столбце. Следовательно, y^0-триггеры не возбуждаются, когда $x = 0$, а y^1-триггеры не возбуждаются, когда $x = 1$. Более того, возбуждения y^1-триггеров в столбце 0 определяются только y^0-переменными, а возбуждения y^0-триггеров в столбце 1 зависят лишь от y^1-переменных. Проверка таблицы 7.2 показывает, что в устойчивых состояниях столбца 0 $y_i^1 = y_i^0$ для $i = 1, 2$. Следовательно, условия возбуждения для y^1-триггеров можно определить так:

$$S_i^1 = \bar{A} y_i^0, \qquad R_i^1 = \overline{A y_i^0},$$

где $i = 1, 2$. Дальнейшая проверка показывает, что в устойчивых состояниях столбца 1 y^0-состояние, рассматриваемое как число в двоичной форме, превышает на 1 y^1-состояние, представленное в такой же форме, и что y^1-состояние совпадает с предшествующим y^0-состоянием. Таким образом, y^0-триггеры, которые должны изменить свое состояние, в точности соответствуют y^1-триггерам, состояния которых должны были бы измениться, если бы счетчик состоял лишь из одних y^1-триггеров. Отсюда следует, что y_1^0 должно всегда принимать значение $\bar{y}_1^1$, когда $A = 1$, так что имеем:

$$S_1^0 = A\bar{y}_1^1, \qquad R_1^0 = A y_1^1.$$

Поскольку y_2^0 должно оставаться фиксированным или принимать значение переменной $\bar{y}_2^1$, если $y_1^1 = 1$, получим:

$$S_2^0 = A y_1^1 \bar{y}_2^1, \qquad R_2^0 = A y_1^1 y_2^1.$$

Удлиняя таблицу 7.2 и убеждаясь в том, что все ранее использованные аргументы сохраняют силу, видим, что этот подход можно применить также к счетчикам с 2^m состояниями, где m — некоторое целое число. Выражения для S_i^1 и R_i^1 остаются неизменными с ростом i, а

выражения для S^0 и R^0 можно обобщить в следующем виде:

$$S_1^0 = A\bar{y}_1^1,$$

$$S_i^0 = A\bar{y}_i^1 \prod_{j=1}^{i-1} y_j^1 \quad \text{для} \quad i = 2, 3, \ldots, m,$$

$$R_i^0 = A \prod_{j=1}^{i} y_j^1 \quad \text{для} \quad i = 1, 2, 3, \ldots, m.$$

Таким образом, решена вторая проблема, указанная в разделе 7.4, поскольку с увеличением размеров счетчика объем логики возрастает линейно, хотя ценой этому является удвоение числа используемых триггеров. Результирующая логическая схема показана на рис. 7.6.

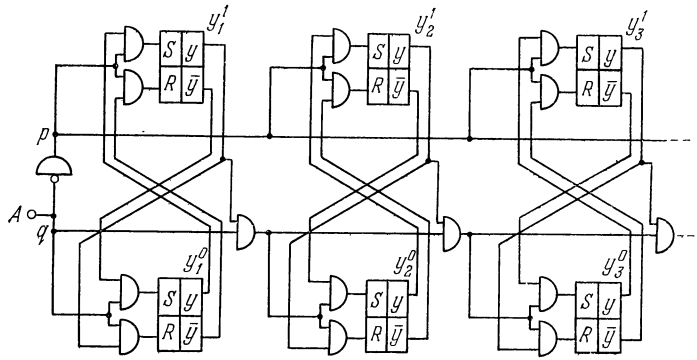


Рис. 7.6. Счетчик Воре.

Анализ этой схемы показывает, что она фактически совпадает со схемой, описанной в разделе 7.3, в которой асинхронные триггеры, такие, как на рис. 7.4 *а*, соединены в соответствии со схемой на рис. 7.3, *б*. (Мы могли бы, конечно, реализовать нашу схему в виде схемы, показанной на рис. 7.3, *а* или любой другой, реализующей те же самые уравнения возбуждения.) Для устранения состязаний применимы те же методы, что и раньше. В частности, можно заменить входной инвертор схемой рис. 7.4, *б*, устранив тем самым необходимость во всех других задержках в схеме. Такой подход является специальным случаем

описанного в подразделе 4.3.3 метода с «задержанным входом» для учета состязаний.

Проведя еще раз рассуждения, приводящие к рис. 7.6, и обращая особое внимание на роль двух множеств триггеров, можно заметить следующую уже знакомую нам картину. В течение одного состояния входа состояния y^1-триггеров повторяют состояния y^0-триггеров, а в течение другого состояния входа состояния y^0-триггеров зависят от y^1-значений. Это в точности совпадает с принципом работы двухранговой схемы первого типа, показанной на рис. 6.13 в разделе 6.3. y^1-триггеры соответствуют второму рангу, а y^0-триггеры — первому рангу, тогда как в качестве синхронизирующих импульсов C_1 и C_2 выступают соответственно сигналы q и p со входной схемы, показанной на рис. 7.4, *б* (которая совпадает, по существу, со схемой на рис. 6.13, *в*, предназначенной для получения связанных соответствующим образом импульсов C_1 и C_2 от единственного источника). Заметим, что наш счетчик не имеет входов, соответствующих x-входам общей модели. Выходные сигналы можно брать с триггеров любого ранга, хотя, если снимать их с y^0-триггеров, то необходимо обеспечить требуемые начальные условия, положив вначале $A = 0$.

Счетчик Воре обязан своим появлением (задолго до развития представленной здесь теории) подходу к синтезу с точки зрения двухранговых схем и был прототипом целого класса таких схем. Он считается в применении быстрым, эффективным и надежным.

7.6. Счетчик Мэйни

Как уже было показано в разделе 6.3, иногда можно уменьшить число триггеров в одном ранге двухранговой схемы с помощью использования нетривиальной логики для возбуждения обоих рангов. Здесь возникает вопрос: нельзя ли построить двухранговый счетчик таким образом, чтобы только один из рангов представлял двоичное число, являющееся выходом счетчика, а другой содержал меньшее число триггеров.

Перед непосредственным разрешением этого вопроса исследуем его кратко в терминах кодирования состояний. Обратившись к методу Трэйси для нахождения ООТП-

кодирования (подраздел 3.3.2), заметим, что в таблице переходов для счетчика (подобной таблице 7.2) кодирование, при котором состояния соответствуют желаемым выходам, т. е. упорядоченной последовательности двоичных чисел (как состояния y^0-переменных в рассмотренном примере), покрывает все дихотомии в одном столбце (в нашем случае в столбце 0) и лишь некоторые дихотомии в других столбцах (например, (12.56)). Логично предположить, что оставшиеся дихотомии удастся покрыть с помощью меньшего числа дополнительных внутренних переменных. Поскольку в общем случае этот вопрос, повидимому, не решается каким-либо очевидным образом, а проверка этой гипотезы для частных случаев является довольно утомительным, хотя и нетрудным занятием, мы не будем далее рассматривать это направление.

Покажем теперь, как можно синтезировать счетчик с восемью состояниями, описываемый таблицей 7.3, *а*, в форме двухранговой схемы второго типа (рис. 6.14). Начнем с кодирования состояний триггеров второго ранга, как показано в таблице 7.3, *б*. Затем построим матрицу изменений, представленную в таблице 7.3, *в*, и применим метод из раздела 6.3 для нахождения минимального множества различных возбуждений, которые удовлетворяют всем клеткам матрицы изменений, а именно: $S_1R_1S_2R_2S_3R_3 = 000010$, 001001, 100101, 010101. Так получаем матрицу возбуждения, представленную в таблице 7.3, *г*. Далее, полученные четыре состояния возбуждения кодируются с помощью переменных $y_1^1y_2^1$ второго ранга соответственно как 00, 01, 10 и 11, что показано в таблице 7.3, *д*. После этого значения клеток матрицы возбуждения (табл. 7.3,*г*) заменяются на полученные возбуждения для триггеров первого ранга, приводящие в каждом случае эти триггеры в такие состояния, которые в соответствии с таблицей 7.3, *д* порождают возбуждения таблицы 7.3, *г*. Результатом является матрица возбуждения триггеров первого ранга, представленная в таблице 7.3, *е*. Подобным образом можно было бы продолжить этот процесс для определения логики обоих рангов по таблицам 7.3, *д* и *е*, учитывая, что сигналы, соответствующие $\overline{A}$ (или p) и A (или q) на рис. 7.6 должны быть поданы на выходные элементы И первого и второго рангов соответственно.

Таблица 7.3

				y_1^2	y_2^2	y_3^2				y_1^2	y_2^2	y_3^2
1	2,000	1	2,000	0	0	0	1	$\overline{S}\,\overline{S}S$	1	0	0	0
2	3,001	2	3,001	0	0	1	2	$\overline{S}SR$	2	0	0	1
3	4,010	3	4,010	0	1	0	3	$\overline{S}\,\overline{R}S$	3	0	1	0
4	5,011	4	5,011	0	1	1	4	SRR	4	0	1	1
5	6,100	5	6,100	1	0	0	5	$\overline{R}\,\overline{S}S$	5	1	0	0
6	7,101	6	7,101	1	0	1	6	$\overline{R}SR$	6	1	0	1
7	8,110	7	8,110	1	1	0	7	$\overline{R}\,\overline{R}S$	7	1	1	0
8	1,111	8	1,111	1	1	1	8	RRR	8	1	1	1

а) Таблица переходов б)Матрица переходов в) Матрица изменений

	S_1^2	R_1^2	S_2^2	R_2^2	S_3^2	R_3^2	y_1^2	y_2^2	y_3^2
1	0	0	0	0	1	0	0	0	0
2	0	0	1	0	0	1	0	0	1
3	0	0	0	0	1	0	0	1	0
4	1	0	0	1	0	1	0	1	1
5	0	0	0	0	1	0	1	0	0
6	0	0	1	0	0	1	1	0	1
7	0	0	0	0	1	0	1	1	0
8	0	1	0	1	0	1	1	1	1

г) Матрица возбуждения

	S_1^2	R_1^2	S_2^2	R_2^2	S_3^2	R_3^2	y_1^1	y_2^1
1	0	0	0	0	1	0	0	0
2	0	0	1	0	0	1	0	1
3	1	0	0	1	0	1	1	0
4	0	1	0	1	0	1	1	1

д) Матрица возбуждения для логики второго ранга

	S_1^1	R_1^1	S_2^1	R_2^1	y_1^2	y_2^2	y_3^2
1	0	1	0	1	0	0	0
2	0	1	1	0	0	0	1
3	0	1	0	1	0	1	0
4	1	0	0	1	0	1	1
5	0	1	0	1	1	0	0
6	0	1	1	0	1	0	1
7	0	1	0	1	1	1	0
8	1	0	1	0	1	1	1

е) Матрица возбуждения для логики первого ранга

Вместо такого продолжения процесса попытаемся обобщить нашу процедуру на счетчики с любым числом состояний. Проверка матрицы изменений (табл. 7.3, *в*) показывает, что для очередного увеличения показания счетчика, когда $y_3^2 = 0$ (младший разряд), мы должны переключить y_3^2 в 1. Если же $y_3^2 = 1$ и $y_2^2 = 0$, мы должны переключить y_3^2 в 0, а y_2^2 в 1. А когда $y_3^2 = y_2^2 = 1$, а $y_1^2 = 0$, первые две переменные нужно переключить в 0, а y_1^2 — в 1. Если же все три переменные равны 1, то все они должны переключиться в 0. Эта ситуация, конечно же, соответствует нашему словесному описанию, приведенному на рис. 7.1, а именно, что для увеличения показания счетчика на единицу мы просто переключаем в 0 все рядом стоящие единичные разряды, начиная с младшего разряда, а первый встретившийся 0 заменяем на 1. Отсюда следует, что для увеличения показания m-разрядного счетчика мы должны проделать одну из $m + 1$ операций, в зависимости от текущего состояния счетчика (имеется, конечно, 2^m возможных состояний). Какая из этих операций должна быть выполнена — определяется числом рядом стоящих единиц, начинающихся с младшего разряда. В нашем примере, где $m = 3$, число возможных операций, таким образом, равно четырем, это объясняет тот факт, почему матрица возбуждения из таблицы 7.3, *г* может быть построена лишь с четырьмя различными значениями клеток. Функция триггеров первого ранга состоит в записи в определенной форме информации о том, какая из $m + 1$ операций должна выполняться на триггерах второго ранга. Эта информация кодируется во втором ранге в виде двоичного представления числа рядом стоящих единиц (хотя нет никаких причин считать такое кодирование наилучшим из возможных).

Таким образом, логика первого ранга должна считывать это число с выходов триггеров второго ранга и соответствующим образом возбуждать триггеры первого ранга.

В нашем примере описанный процесс иллюстрируется рис. 7.7. Сначала вычисляются сигналы B_i ($i = 0, 1, 2, 3$) — число рядом стоящих единиц. Эти сигналы устанавливают триггеры первого ранга в состояния, соответствующие двоичному представлению числа i. Например, если

$y_1^2 y_2^2 y_3^2 = 010$, то возбуждается B_2, что в свою очередь приводит к значениям $y_1^1 = 1$ и $y_2^1 = 0$.

В течение второй фазы ($A = 1$ или $q = 1$) изменяются состояния определенных триггеров второго ранга в соответствии с информацией, запасенной на триггерах первого ранга. В частности, если в первом ранге запасено

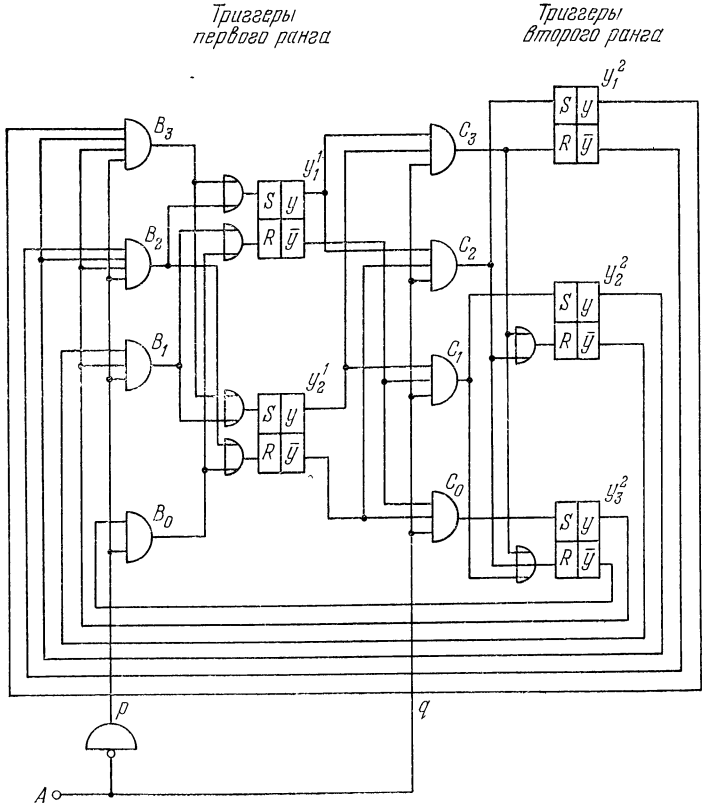


Рис. 7.7. Счетчик Мэйни.

число i, то выход C_i становится равным 1, и i триггеров младших разрядов второго ранга устанавливаются в 0, тогда как $(i+1)$-й триггер (если $i+1 \leqslant m$) устанавливается в 1. В рассматриваемом примере необходимая для

выполнения указанных операций логика показана на рис. 7.7. Таким образом, если $i = 2$ $(y_1^1 = 1, y_2^1 = 0)$, то в течение второй фазы C_2 становится равным 1, обусловливая следующие значения: $y_2^2 = y_3^2 = 0$, $y_1^2 = 1$.

В общем случае для счетчика с 2^m состояниями (с m триггерами второго ранга) необходимо $\mathrm{E}[\log_2(m+1)]$ триггеров первого ранга. Так, для счетчика Мэйни необходимо $m + \mathrm{E}[\log_2(m+1)]$ триггеров для счета от 0 до 2^m, тогда как для счетчика Воре их число равно $2m$, для счетчика Грея — $m + 1$ и, наконец, m — для счетчика «домино» или импульсного счетчика.

Счетчик Мэйни обладает теми же преимуществами в скорости и надежности, что и счетчик Воре. Сравнение рис. 7.6 и 7.7 показывает, что счетчик Мэйни требует большего объема логики, чем счетчик Воре, но меньшего числа триггеров. (Автору было указано А. Фридзом на то, что на схеме рис. 7.7 на нулевые входы триггеров первого ранга достаточно подавать лишь сигнал B_0. При этом отпадает нужда в элементах ИЛИ на входах этих триггеров, но появляется необходимость установки триггеров первого ранга, так же как и триггеров второго ранга, в начальные состояния, если счет начинается с нечетного числа.) Ясно, что сложность логики для счетчика Мэйни также может быть существенно уменьшена при использовании итеративных схем, однако эта простота потребует удлинения путей прохождения сигналов (как это было ранее в случае, показанном на рис. 7.3). Можно также рассмотреть различные варианты кодирования для логики первого ранга аналогично тому, как это делалось ранее.

ЗАМЕЧАНИЯ ПО БИБЛИОГРАФИИ

Автор, к сожалению, не смог отыскать первоисточники, касающиеся счетчиков с параллельным управлением, описанных в разделе 7.2. Триггер, приведенный на рис. 7.5, был описан Кэглом [11], который использовал его для построения счетчиков. Счетчик Воре был разработан Воре [118]. Мэйни является создателем счетчика Мэйни [69], он разработал также вариант этого счетчика, использующий проверку на четность [70]. Другие инте-

ресные сведения по счетчикам, не описанные здесь, содержатся в работах Кэйна [54] и Маркуса [67]. Дополнительный материал по счетчикам можно найти также в книге Ричардса [96].

ЗАДАЧИ

7.1*. Построить импульсный пятиразрядный счетчик с параллельным управлением, в котором число входов и число ветвей разветвления выхода для любого элемента И не превышают 3 и в котором никакой сигнал не проходит более чем через два каскада логики.

7.2*. Построить импульсный счетчик с параллельным управлением, который считает от 0 до 5, а затем возвращается в 0 для следующего цикла счета.

7.3. Построить счетчик «домино», использующий триггеры, реагирующие на задние фронты приходящих входных импульсов.

1.1. Выходной сигнал становится равным 1 после второго последовательного изменения X_2 и не меняет своего значения, пока не изменится X_1. Изменение X_1 всегда приводит к установлению Z в 0, а для установления Z в 1 необходимо два последовательных изменения X_2.

1.4.

	X = 0	X = 1	C_1	C_2
1	***1***	2	0	0
2	3	***2***	1	0
3	***3***	4	0	0
4	1	***4***	0	1

2.2.

а)

	X = 0	X = 1
1	1,0	2,0
2	3,0	1,0
3	2,1	1,0

б)

	X = 0	X = 1
1	1,1	2,0
2	3,0	1,0
3	2,1	1,0

в)

	X = 0	X = 1	Строки исходной таблицы
1	2,0	1,0	12
2	1,1	1,0	13

Таблицы *а*) и *б*) не могут быть подвергнуты дальнейшей минимизации. Однако исходную таблицу можно минимизировать до таблицы *в*) с двумя строками.

2.6. Карта финальных пар имеет следующий вид:

1					
36	2				
××	14××	3			
13××	16	××	4		
	12	××	15	5	
12	××	46	45	14 25 26××	6

МС-множествами являются: 245, 46, 36, 125, 16, граф импликаций приведен на рис. Р2.6.

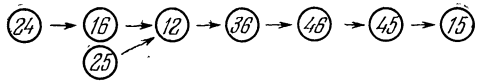


Рис. Р2.6.

Таким образом, получаем минимальное замкнутое покрытие: 45, 46, 36, 125, так что решением является следующая таблица:

	X_1X_2 00	01	11	10	
1	2,0	1,0	1,1	1,1	125
2	—,1	1,1	4,0	3/4,1	36
3	1,0	1/3,—	1,1	1,1	45
4	1,0	1,1	2/4,—	3,1	46

2.8. а) Карта финальных пар имеет следующий вид:

1	2	3	4	5	6	7	8	9
23 45								
46	56							
×	×	×						
×	×	×	13					
×	×	×		12				
×	×	×			×			
×	×	×	×			×		
×	×	×		×		×	×	

МС-множествами являются: 568, 469, 457, 456, 123. Граф импликаций приведен на рис. Р2.8 а.

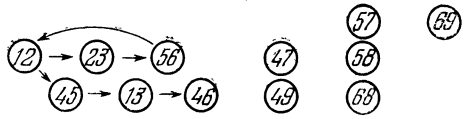


Рис. Р2.8а.

Заметим, что хотя МС-множество 123 порождает 45, 46 и 56, оно не порождает 456, так что 456 не входит в наше замкнутое покрытие. Минимизированная таблица приведена ниже:

	A	B	C	D	
1	1,0	2,0	4,0	3,0	123
2	1,0	1,1	4,1	2,0	457
3	1,1	1,1	—,1	1,1	469
4	1,1	1,1	3,0	2,0	469

б) Карта финальных пар имеет следующий вид:

1								
23 45	2							
12 46	13 56	3						
×	×	×	4					
×	×	×	13	5				
×	×	×			6			
×	×	×			×	7		
×	×	×	×			×	8	
×	×	×		×		×	×	9

MC-множествами являются: 568, 469, 457, 456, 123. Граф импликаций приведен на рис. Р2.8 б.

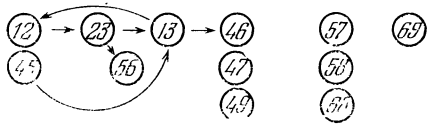


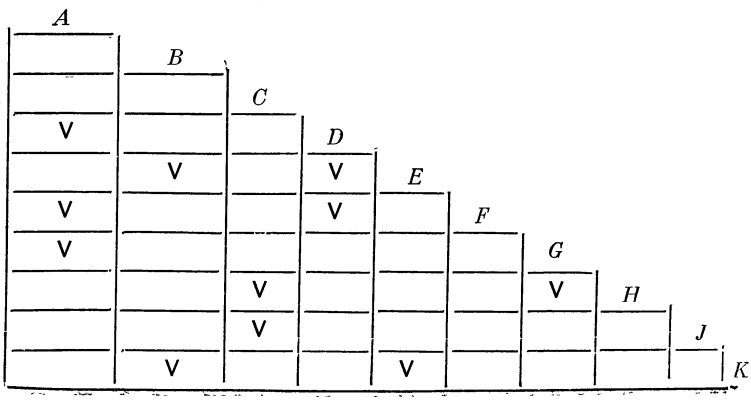
Рис. Р2.8б.

Если рассматривать лишь попарную совместимость, то, как и в задаче 2.8 а, замкнутой совокупностью МС-множеств будет {123, 457, 469, 568}. Однако при этом 123 ⟶ 456, так что указанная совокупность не будет замкнутой в обычном смысле. Одним из решений является {123, 457, 469, 568, 456}. Другое решение — {123, 456, 7, 8, 9}. Минимизированная таблица приведена ниже:

	A	*B*	*C*	*D*	
1	1,0	2,0	—,0	3,—	123
2	1,1	1,1	—,1	1,0	456
3	—,0	—,1	4,—	—	7
4	—,1	—,0	5,—	—	8
5	—,1	—,1	—	1,1	9

2.11а. Пусть $i_а$ и $i_б$ — i-е строки таблиц $а$ или $б$ соответственно, и запись $i_а$ с $j_б$ означает, что строка $i_а$ покрывает строку $j_б$. С учетом значений выходов получаем, что лишь строка $3_а$ может покрыть строку $2_б$, так что $3_а$ с $2_б$ ⟶ $1_а$ с $3_б$ ⟶ $1_а$ с $1_б$. Но строка $1_а$ не может покрыть строку $1_б$, так как выход для $1_б$ не определен в столбце B, тогда как соответствующий выход для $1_а$ определен. Следовательно, таблица $а$ не покрывает таблицу $б$.

2.15. Найти наибольшие множества классов, для которых допустимо одновременное проведение зачетов (см. [39]). Проверка указывает на следующие совместимые классы;

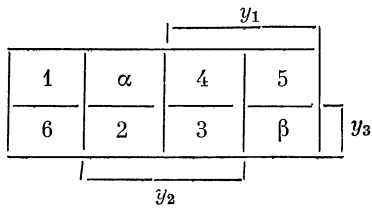


МС-множествами являются *GH*, *BEK*, *DE*, *ADF*, *AG*, *CH*, *CJ*. Минимальная покрывающая совокупность: *BEK*, *CJ*, *ADF*, *GH*.

Понедельник	Вторник	Среда	Четверг
BEK	*CJ*	*ADF*	*GH*

3.3. См. табл. 4.8.

3.8. Множествами предшественников в четырех столбцах являются (12, 34, 56), (14, 26, 35), (16, 23, 45), (135, 24, 6). Соответствующее им кодирование задается следующей таблицей:



Результирующая матрица переходов приведена ниже:

	00	01	11	10	y_1	y_2	y_3
1	***1***	α	***1***	5	0	0	0
2	α	***2***	***2***	α	0	1	1
3	4	***3***	2	***3***	1	1	1
4	***4***	***4***	5	***4***	1	1	0
5	β	β	***5***	β	1	0	0
6	***6***	2	1	***6***	0	0	1
α	1	4	—	4	0	1	0
β	6	3	—	3	1	0	1

3.12. Необходимо покрыть следующие дихотомии:
Столбцы

00: (12, 35) (14, 35)
01: (12, 35)
11: (13, 4) (13,5) (23,4) (23, 5)
10: (1, 34) (1, 45) (2, 34) (2, 45) (1, 2)

Выберем строку 3, как наиболее часто встречающуюся в этих дихотомиях, и построим 14 различных упорядоченных дихотомий по отношению к строке 3:

$A[12, 35]$, $B[14, 35]$, $C[4, 13]$, $D[5, 13]$,
$E[4, 23]$, $F[5, 23]$, $G[1, 34]$, $H[2, 34]$, $I[1, 45]$,
$\overline{I}[45, 1]$, $J[2, 45]$, $\overline{J}[45, 2]$, $K[1, 2]$, $\overline{K}[2, 1]$.

Покрывающей совокупностью МС-множеств является *BEK*, $CDEF\overline{I}\overline{J}$, *AGHIJ*. Эти МС-множества соответствуют дихотомиям [14, 235], [45, 123], [12, 345], и если перейти к дополнению второй из

них (приписав комбинацию 000 строке 1), получим следующую матрицу переходов:

	X_1X_2 00	01	11	10	y_1	y_2	y_3
1	***1***,0	2,0	3,0	***1***,0	0	0	0
2	1,0	***2***,0	3,0	***2***,1	1	0	0
3	***3***,1	5,0	***3***,0	4,0	1	0	1
4	1,0	—	***4***,0	***4***,0	0	1	1
5	3,1	***5***,0	***5***,1	4,0	1	1	1
	1,0	—	3,0	4,0	0	0	1
	1,0	—	—	—	0	1	0

3.19. Используя метод, описанный, например, в [44], построим разбиение со свойством подстановки (146, 235); других разбиений со свойством подстановки менее чем с пятью блоками, разделяющих состояния 2 и 3, не существует. Следовательно, не существует решения задачи, основанного на разбиениях. В то же время существуют системы множеств со свойством подстановки.

Рассмотрим, например, разбиения $\alpha_1 = (1234, 256)$ и $\alpha_2 = (1, 25, 23, 46)$. И α_1, и α_2 являются системами множеств со свойством подстановки, причем их произведение не равно нулю. Допустим,

Таблица Р3.19

	A	*B*	*C*	*D*	
1	***1***	***1***	2	***1***	0 (12_p34)
2	1	***2***	***2***	1	1 (2_q56)

а) Таблица T_1

	A	*B*	*C*	*D*	y_2 y_3
1	***1***	2/3	4	***1***	00 (1)
2	3	***2***	***2***	1	01 (2_p5)
3	***3***	***3***	2	2	11 (2_q3)
4	1	3	***4***	***4***	10 (46)

б) Таблица T_2

что мы преобразовали α_1 и α_2 в разбиения $\alpha_1' = (12_p34, 2_q56)$ и $\alpha_2' = (1, 2_p5, 2_q3, 46)$ соответственно, расщепив строку 2 на две строки 2_p и 2_q. Тогда $\alpha_1'\alpha_2' = 0$. С учетом такого расщепления можно найти расширенное эквивалентное представление исходной таблицы такое, что α_1' и α_2' суть разбиения со свойством подстановки в этой новой таблице. Тогда таблицы переходов T_1 и T_2, соответствующие α_1' и α_2' (см. табл. Р3.19, *а* и *б*), определяют компоненты в параллельной декомпозиции исходной таблицы при условии, что $N(1, B) = 2$ в таблице T_2. При использовании кодирования, показанного для таблиц T_1 и T_2, состояния исходной таблицы переходов для обеспечения требуемой декомпозиции кодируются следующим

образом:

1(000), 2(001 и 111), 3(011), 4(010), 5(101), 6(110).

Для использования кодирования двумя переменными, свободного от критических состязаний, элемент $N(3, D)$ в таблице T_2 заменен на 2. Заметим, что возможна последовательная декомпозиция таблицы T_2, использующая разбиение со свойством подстановки (14, 23), соответствующее разбиению (146, 235) на состояниях исходной таблицы.

4.1. а) Все сигналы задерживаются на 10 единиц времени;

б) Сигнал s_1 не проходит на выход (блокируется с помощью задержки $D_{и1}$). Другие сигналы задерживаются на 10 единиц времени.

в) Сигналы s_1 и s_2 блокируются, тогда как s_3 и s_4 задерживаются на 10 единиц времени. Последовательное соединение элементов инерциальной и совершенной задержки полностью характеризуется величиной наибольшей инерциальной задержки и суммой значений задержки всех элементов задержки. Следовательно, в нашем случае меньшую инерциальную задержку $D_{и1}$ можно заменить совершенной задержкой $D_{с1}$ той же величины без изменения поведения схемы.

4.2. а) Реакцией схемы является импульс длительностью в одну единицу времени, начинающийся спустя 4 единицы времени после прохождения переднего фронта входного импульса;

б) Реакцией схемы является импульс длительностью в одну единицу времени, начинающийся спустя 6 единиц времени после прохождения переднего фронта входного импульса.

4.10. Схема задается выражением $AB + B$. Допустим, что сигналы B и A изменяются в противоположных направлениях. Тогда полагая $B = \overline{A}$, получим выражение $A\overline{A} + \overline{A}$, которое обязательно содержит состязание (при изменениях между 01 и 10 в исходном выражении). Этот же вывод следует непосредственно из теоремы 4.5, если положить $I_1 = 01$ и $I_2 = 10$ и рассматривать выражение $A^1B^2 + B^3$.

4.12. В соответствии с процедурой 4.1 и теоремой 4.6 достаточно покрыть следующие дихотомии:

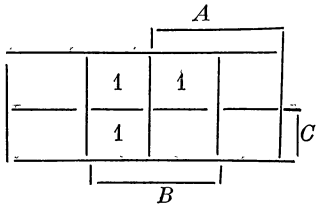
(12, 5) (23, 5) (24, 5) (1, 34) (2, 34) (34, 5) (12, 34).

Все дихотомии, кроме последней, покрываются исходным кодированием. Дихотомия (12, 34) связана с переходом 1—000 ⟶ 1—001. В этом случае опасность состоит в том, что $T(1, 2)$ будет пересекаться с $T(3, 4)$, где в столбце 000 следующим состоянием должно быть состояние 4. Из-за наличия паразитных задержек некоторые из Y-переменных приняли бы значения, соответствующие этому состоянию в столбце 000, в результате чего система перешла бы в состояние 5—001, а не в состояние 2—001. Такой исход был бы возможен в нашем примере, если бы не переменная y_1. Поскольку y_1 разделяет 1234 и 5, подобная возможность устраняется, и, следовательно, используемое кодирование является пригодным.

4.20. Если при переходах между входными состояниями I_1 и I_2 существует логическое 0-состязание, то $f(x) = 0$ для каждого x из $T(I_1, I_2)$ и должно существовать элементарное произведение p_j,

равное единице в течение перехода и нулю после его окончания. Но если p_j не содержит отрицаний какой-либо буквы, то p_j должно быть равным единице в устойчивом состоянии при некотором x' между I_1 и I_2. Следовательно, на K-карте функции f в области, соответствующей переходу между I_1 и I_2, должна существовать клетка с единичным значением, что противоречит нашему исходному предположению (т. е. состязание становится функциональным, а не логическим).

4.21. Рассмотрим K-карту заданной функции:



Если мы не будем использовать элементарное произведение $B\bar{C}$, то на переходе 010 ⟶ 110 будет существовать 1-состязание. При использовании этого произведения в течение перехода 000 ⟶ 011 соответствующий элемент может принять сначала единичное, а затем нулевое значения до того, как установится в 1 элемент, соответствующий элементарному произведению, покрывающему 011, так что в схеме произойдет динамическое состязание. Таким образом, в И — ИЛИ-схеме должно иметь место по крайней мере одно из указанных состязаний, тогда как других состязаний можно избежать. В схеме, соответствующей выражению $B(\bar{A}+\bar{C})$, никогда не возникает состязаний, в чем можно убедиться, применив методы, рассмотренные в разделе 4.2.

4.23а. Прежде всего преобразуем таблицу, изменив на 0 значение выхода для состояния 1—01 и расщепив строку 3 на две строки. Заметим, что строку 2 расщеплять не обязательно, поскольку при пребывании системы в каком-либо из устойчивых состояний в начальный момент и при многократных изменениях входов невозможно возникновение ошибочных переходов. Для модифицированной таблицы легко найти ОТП-кодирование. Каждая y-переменная должна быть приписана выходу инерциальной задержки, что достигается использованием блока задержки, показанного на рис. 4.14.

	x_1x_2 00	01	11	10	Y_1	Y_2
1	***1***,0	2,0	***1***,0	***1***,0	0	0
2	3*A*,0	***2***,1	1,0	***2***,0	0	1
3*A*	***3A***,0	***3A***,0	1,0	3*B*,1	1	1
3*B*	3*A*,1	3*A*,1	1,1	***3B***,1	1	0

5.1. Матрица переходов приведена ниже:

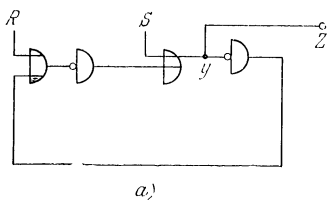
	SR 00	01	11	10	y
1	***1***,0	***1***,0	—	2,—	1
2	***2***,1	1,—	—	***2***,1	0

$$Y = S + \bar{R}y, \quad Z = y.$$

Если использовать непосредственно Y-выражение, то потребуется один инвертор для получения $\bar{R}$ из сигнала R и два инвертора для поддержания усиления в петле обратной связи и воспроизведения выходного сигнала Z. Вместо этого можно переписать Y-выражение в следующем виде:

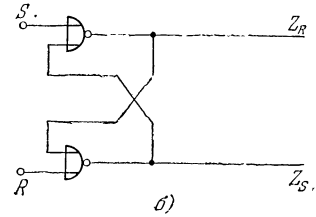
$$Y = S + \overline{(R + \bar{y})}.$$

Такое преобразование устраняет необходимость инвертирования сигнала R и размещения двух инверторов между Y и y. Результирующая схема приведена на рис. Р5.1, *а*. При наличии такой



а)

Рис. Р5.1а.



б)

Рис. Р5.1б.

схемы можно рассмотреть вариант, когда выходной сигнал снимается с выхода левого инвертора, т. е.

$$Z = \overline{(R + \bar{y})} = \bar{R}y.$$

Это выражение также удовлетворяет исходным условиям, и другим выходом триггера можно считать выход второго инвертора. В этом случае схема преобразуется в пару элементов ИЛИ — НЕ, соединенных, как показано на рис. Р5.1, *б*. Выходы Z_S и Z_R не являются взаимно дополнительными, поскольку при $R = 1$ они одновременно могут принимать нулевые значения (однако в этой книге везде, где состояния рассматриваемых триггеров определяют состояние системы, их выходы Z_S и Z_R обозначаются через y и $\bar{y}$). Поскольку в схеме имеется лишь одна y-переменная, элементов задержки не требуется.

5.2. Матрица переходов триггера со счетным входом приведена ниже:

	τ = 0	τ = 1	y_1	y_2
1	***1***,0	2,0	0	0
2	3,1	***2***,0	0	1
3	***3***,1	4,1	1	1
4	1,0	***4***,1	1	0

$$Q = \bar{\tau} y_2 + \tau y_1, \quad Y_1 = \bar{\tau} q + \tau \bar{q} = q,$$

$$Y_2 = \bar{\tau} q + \tau \bar{q}, \quad Z = \bar{\tau} q + \tau q = q.$$

Таким образом, необходимо три транзистора, в том числе один для инвертирования τ. (Другое решение также с тремя транзисторами приведено в разделе 7.3. В этой схеме используются конденсаторы или эффект полного запоминания информации.)

6.1. Множествами предшественников для четырех столбцов являются соответственно (1, 23, 4), (12, 34), (14, 23), (14, 23). Состояния 1 и 4 в столбце 1 можно объединить, следовательно, кодирование по Лю, соответствующее множествам (12, 34) и (14, 23), является пригодным. Строки 1, 2, 3, 4 закодированы следующими комбинациями значений переменных $y_1 y_2$: 00, 01, 11, 10. Схема должна быть построена так, чтобы на протяжении любого перехода внутри столбца лишь одно элементарное произведение равнялось единице. (Это запрещает использование в выражении для Y_1 элементарного произведения $\bar{x}_1 y_1$.) Результирующие выражения приведены ниже:

$$Y_1 = \bar{\bar{x}}_1 x_2 y_2 + \bar{\bar{x}}_1 x_2 y_1 \bar{y}_2 + \bar{x}_1 x_2 y_1 + x_1 x_2$$

$$Y_1 = \bar{\bar{x}}_1 x_2 \bar{y}_1 y_2 + \bar{x}_1 x_2 \bar{y}_1 + x_1 \bar{x}_2,$$

$$Y_2 = \bar{\bar{x}}_1 x_2 \bar{y}_2 + \bar{x}_1 x_2 \bar{y}_1 + x_1 y_2,$$

$$\bar{Y}_2 = \bar{\bar{x}}_1 x_2 y_2 + \bar{x}_1 x_2 y_1 + x_1 \bar{y}_2,$$

$$Z = \bar{\bar{x}}_1 x_2 y_1,$$

$$\bar{Z} = \bar{\bar{x}}_1 x_2 \bar{y}_2 + \bar{x}_1 x_2 + x_1.$$

7.1. Решение приведено на рис. Р7.1.

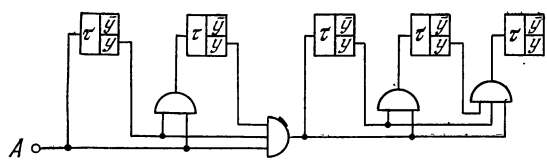


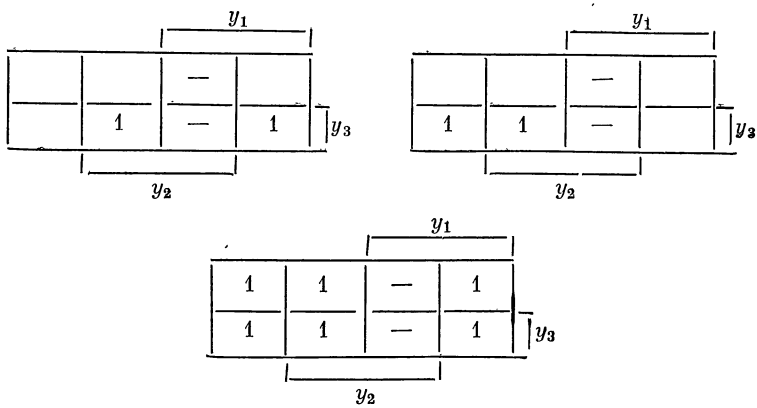
Рис. Р7.1.

7.2.

		y_1	y_2	y_3
1	2	0	0	0
2	3	0	0	1
3	4	0	1	0
4	5	0	1	1
5	6	1	0	0
6	1	1	0	1

			y_1	y_2	y_3
0	0	1	0	0	0
0	1	1	0	0	1
0	0	1	0	1	0
1	1	1	0	1	1
0	0	1	1	0	0
1	0	1	1	0	1

K-карты для τ_1, τ_2 и τ_3:



$$\tau_1 = Ay_2y_3 + Ay_1y_3, \quad \tau_2 = A\bar{y}_1y_3, \quad \tau_3 = A.$$

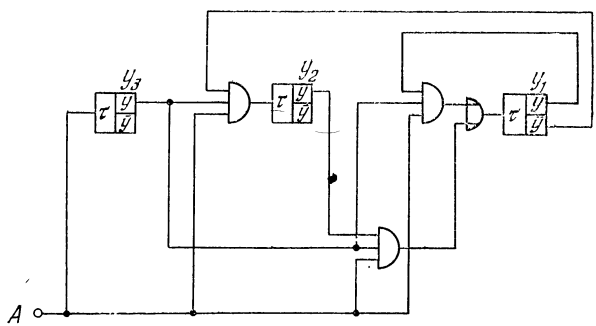


Рис. Р7.2.

ЛИТЕРАТУРА

Литература, приведенная здесь, содержит (без претензий на полноту) как основные работы, связанные с материалом настоящей книги, так и ряд статей и книг по смежным вопросам. Непосредственные ссылки на соответствующие работы приводятся в разделах «Замечания по библиографии», заканчивающих каждую главу. Вместе с тем много ссылок и в самом тексте.

Вниманию читателя можно предложить несколько вводных пособий по теории переключательных схем, в которых затрагиваются вопросы и асинхронных схем. Здесь следует назвать (без какого-либо предпочтения) работы Пратера [94], Маркуса [68], Торнга [108], Вуда [119], Кригера [59], Фистера [93], Хамфри [53], Бартн, Лебова и Рида [3], Хилла и Петерсона [48]. Мур был автором интересного сборника статей [82] по рассмотренным здесь вопросам. Ряд интересных статей содержится также в книгах [75] и [6] под редакцией (соответственно) Маккласки и Барти, а также Бьёрчи. В некоторых книгах рассматриваются «конечные автоматы»— синхронизированные последовательностные схемы, описываемые с помощью соотношений между входными и выходными последовательностями. К таким книгам следует отнести книги Бута [7], Хенни [47], Гилла [28], Гинзбурга [33], Харрисона [43].

Основным журналом, публикующим работы по теории переключательных схем, является IEEE Transactions on Computers, называвшийся ранее IEEE (IRE) Transactions on Electronic Computers, а также Journal of the Association for Computing Machinery. Другим важным органом для публикации, в частности, предварительных сообщений о полученных результатах является Proceedings of the Annual Symposia on Switching and Automata Theory.

1. Armstrong D. B., Friedman A. D., Menon P. R., Design of asynchronous circuits assuming unbounded gate delays. IEEE Trans. Computers, v. C-18, No. 12, 1969.
2. Armstrong D. B., Friedman A. D., Menon P. R., Realization of asynchronous sequential circuits without inserted delay elements. IEEE Trans. Computers, v. C — 17, No. 2, pp. 129—134, 1968.
3. Bartee T. C., Lebow I. L., Reed I. C., Theory and design of digital machines. McGraw-Hill, N. Y., 1962.
4. Beatty J., Miller R., Some theorems for incompletely specified sequential machines with applications to state minimization. In: Proceedings of the 3rd Annual Symposium on Switching Circuits Theory and Logical Design, pp. 123—136, 1962.

5. Beatty J., Miller R., An approach to state minimization for incompletely specified sequential machines. IBM Research Report, RC-1055, 1963.
6. Biorci G. (ed.), Network and switching theory. A NATO Advanced Study Institute, Academic Press, N. Y., 1968.
7. Booth T. L., Sequential machines and automata theory. John Wiley and Sons, N. Y., 1967.
8. Bredt T. H., McCluskey E. J., Analysis and synthesis of control mechanisms for parallel processes. In: Symposia on Parallel Processing Systems and Applications, Naval Postgraduate School, Monterey, Calif., 1969.
9. Brzozowski J. A., Some problems in relay circuit design. IEEE Trans. Electron. Computers, v. EC-14, No. 4, pp. 630—634, 1965.
10. Brzozowski J. A., Singh S., Definite asynchronous sequential circuits. IEEE Trans. Computers, v. C-17, No. 1, pp. 18—26, 1968.
11. Cagle W. B., Menne R. S., Skinner R. S., Staehler R. E., Underwood M. D., No. 1 ESS logic circuits and their application to the design of the central control. Bell System Techn. Journal, v. 43, pp. 2055—2095, 1964.
12. Caldwell S. H., Switching circuits and logical design. John Wiley and Sons, N. Y., 1958. [Русский перевод: Колдуэлл С., Логический синтез релейных устройств. ИЛ, М., 1962.]
13. Eichelberger E. B., Sequential circuits synthesis using hazard and delays. Ph. D. Dissertation, Dept. of Electrical Engineering, Princeton University, 1963.
14. Eichelberger E. B., Sequential circuits synthesis using input delays. In: Proc. 4th IEEE Annual Symp. on Switching Circuit Theory and Logical Design, v. S-156, 1963.
15. Eichelberger E. B., Hazard detection in combinational and sequential switching circuits. IBM Journal, v. 9, No. 2, pp. 90—99, 1965.
16. Elspas B., Goldberg J., Short R. A., Stone H. S., Investigation of propagation — limited computer networks. Final Report — Phase II, Stanford Research Institute, 1965.
17. Ferrari D., Grasselli A., A cellular structure for sequential networks. In: IEEE Conference Record of 8th Annual Symp. on Switching and Automata Theory, pp. 210—225, 1967.
18. Friedes A., State reduction and state assignments for asynchronous sequential machines. Ph. D. Dissertation, Dept. of Electrical Engineering, Columbia University, 1965.
19. Friedman A. D., Feedback in synchronous and asynchronous sequential switching circuits. Ph. D. Dissertation, Dept. of Electrical Engineering, Columbia University, 1965.
20. Friedman A. D., Feedback in asynchronous sequential circuits. IEEE Trans. Electron. Computers, v. EC-15, No. 5, pp. 740—749, 1966.
21. Friedman A. D., Feedback in synchronous sequential circuits. IEEE Trans. Electron. Computers, v. EC-15, No. 3, pp. 354—367, 1966.

22. Friedman A. D., Graham R. L., Ullman J. D., Universal single transition time asynchronous state assignments. IEEE Trans. Computers, v. C-18, No. 6, pp. 541—547, 1969.
23. Friedman A. D., Menon P. R., Synthesis of asynchronous sequential circuits with multiple input changes. IEEE Trans. Computers, v. C-17, No. 6, pp. 559—566, 1968.
24. Gerace G. B., Gestri G., State assignments for reducing the number of delay elements in sequential machines. Information and Control, v. 10, No. 3, pp. 223—253, 1967.
25. Gerace G. B., Gestri G., Sequential machines with less delay elements than feedback paths. IEEE Trans. Computers, v. C-18, No. 2, 1969.
26. Gilbert E. N., Gray codes and path on the n-cube.— Bell System Techn. Journal, v. 37, pp. 815—826, 1958.
27. Gilchirst B., Pomerine J. H., Wong S. Y., Fast carry logic for digital computers. IRE Trans. Electron. Computers, v. EC-4, pp. 133—136, 1955.
28. Gill A., Introduction to the theory of finite state machines. McGraw-Hill, N. Y., 1962. [Русский перевод: Гилл А., Введение в теорию конечных автоматов. «Наука», М., 1966.]
29. Gimpel J. F., A reduction technique for prime implicant tables. IEEE Trans. Electron. Computers, v. EC-14, No. 4, pp. 542—552, 1965.
30. Ginsburg S., A synthesis technique for minimal-state sequential machines. Natl. Cash Register Co. Internal Report, July 1958.
31. Ginsburg S., On the reduction of superfluous states in a sequential machine. Journal of ACM, v. 6, pp. 259—282, 1959.
32. Ginsburg S., A technique for the reduction of a given machine to a minimal-state machine. IRE Trans. Electron. Computers, v. EC-8, No. 3, pp. 346—355, 1959.
33. Ginsburg S., An introduction to mathematical machine theory. Addison-Wesley, Reading, Mass., 1962.
34. Grasselli A., Luccio F., A method for minimizing the number of internal states in incompletely specified sequential networks. IEEE Trans. Electron. Computers, v. EC-14, No. 3, pp. 350—359, 1965.
35. Grasselli A., Minimal closed partitions incompletely specified flow tables. IEEE Trans. Electron. Computers, v. EC-15, No. 2, pp. 245—249, 1966.
36. Grasselli A., Luccio F., A method for combined row-column reduction of flow tables. In: Proc. 7th Annual Symp. Switching Theory and Automata, 1966.
37. Hall A. D., Introduction to the theory of speed independent asynchronous switching circuits. Report No. 50, Digital Systems Laboratory, Princeton University, 1966.
38. Hall A. D., Synthesis of double rank sequential circuits. Report No. 53, Digital Systems Laboratory, Princeton University, 1966.
39. Hall A. D., Acton F. S., Scheduling university course examinations by computer. Communications of ACM, v. 10, No. 4, pp. 235—238, 1967.

40. Hall A. D., Treatment of delays in asynchronous networks. Ph. D. Dissertation, Dept. of Electrical Engineering, Princeton University, 1966.
41. Hamming R. W., Error detecting and correcting codes. Bell System Techn. Journal, v. 29, pp. 147—160, 1950.
42. Hammel D., Ideas on asynchronous feedback networks. In: Proc. 5th Annual Symp. on Switching Theory and Logical Design, Princeton University, pp. 4—11, 1964.
43. Harrison M. A., Introduction to switching and automata theory. McGraw-Hill, N. Y., 1965.
44. Hartmanis J., Stearns R., Algebraic structure theory of sequential machines. Prentice-Hall, Englewood Cliffs, N. J., 1966.
45. Hazeltine B., Encoding of asynchronous sequential circuits. IEEE Trans. Electron. Computers, v. EC-14, pp. 727—729, 1965.
46. Hendrickson H. C., Fast high-accuracy binary parallel addition. IRE Trans. Electron. Computers, v. EC-9, No. 4, pp. 465—469, 1960.
47. Hennie F. C., Finite state models for logical machines. John Wiley and Sons, N. Y., 1968.
48. Hill F. J., Peterson G. R., Introduction to switching theory and logical design. John Wiley and Sons, N. Y., 1968.
49. Huffman D. A., The synthesis of sequential switching circuits. Journal of the Franklin Institute, v. 257, No. 3, pp. 161—190, 1954, and No. 4. pp. 275—303, 1954. (См. также:) Moore E. F., Sequential machines: selected papers. Addison-Wesley, Reading, Mass., 1964.
50. Huffman D. A., A study of the memory requirements of sequential switching circuits. Techn. Report No. 293, Research Laboratory of Electronics, MIT, 1955.
51. Huffman D. A., Design of hazard-free switching circuits. Journal of ACM, v. 4, pp. 47—62, 1957.
52. Huffman D. A. (unpublished paper delivered at Princeton University Symposium on Switching Theory. Princeton, N. J., 1962.)
53. Humphrey W. S., Switching circuits with computer applications. McGraw-Hill, N. Y., 1958.
54. Kain R. Y., Synthesis of up-down counters. IEEE Trans. Electron. Computers, v. EC-16, No. 2, pp. 146—151, 1967.
55. Keister W., Ritchie A. E., Washburn S. H., The design of switching circuits. D. Van Nostrand, Princeton, N. J., 1951.
56. Kinney L. L., A characterization of some asynchronous state assignments. In: IEEE Conf. Record of 9th Annual Symp. on Switching and Automata Theory, pp. 20—27, 1968.
57. Kinney L. L., Decomposition of asynchronous sequential switching circuits. Ph. D. Dissertation, Dept. of Electrical Engineering, University of Iowa, 1968.
58. Kliman M., Lowenschuss O., Asynchronous electronic switching circuits. In: 1959 IRE National Conventional Record, Pt. 4, pp. 267—271, 1959.
59. Krieger M., Basic switching circuit theory. The Macmillan Co., N. Y., 1967.

60. Langdon C. G., Jr., Analysis and synthesis of asynchronous circuits under different delay assumptions. Ph. D. Dissertation, Dept. of Electrical Engineering, Syracuse University, 1967.
61. Langdon C. G., Jr., Delay-free asynchronous circuits with constrained line delays. IEEE Trans. Computers, v. C-17, pp. 1131—1143, 1968.
62. Lerner S. B., Hazard correction in asynchronous sequential circuits. IEEE Trans. Electron. Computers, v. EC-14, No. 2, pp. 265—267, 1965.
63. Liu C. N., A state variable assignment method for asynchronous sequential switching circuits. Journal of ACM, v. 10, pp. 209—216, 1963.
64. Maley G. A., Earle J., The logic design of transistor digital computers. Prentice-Hall, Englewood Cliffs, N. J., 1963.
65. Marcus M. P., Relay essential hazards. IEEE Trans. Electron. Computers, v. EC-12, pp. 405—407, 1963.
66. Markus M. P., Derivation of maximal compatibles using Boolean algebra. IBM Journal, v. 8, No. 5, pp. 537 — 538, 1964.
67. Markus M. P., Cascaded binary counters with feedback.— IEEE Trans. Electron. Computers, v. EC-12, pp. 361—364, 1963.
68. Markus M. P., Switching circuits for engineers. Prentice-Hall, Englewood Cliffs, N. J., 1962.
69. Mayne R. H., The binary counter: a novel lognical design. Unpublished Bell Laboratories memorandum, 1958.
70. Mayne R. H., A binary counter with associated parity bit. Unpublished Bell Laboratories memorandum, 1958.
71. McCluskey E. J., Fundamental mode and pulse mode sequential circuits. In: Proc. IFIP Congress 1962, Münich, Germany, Information Processing, North-Holland Publishing Co., Amsterdam, pp. 725—730, 1962.
72. McCluskey E. J., Minimum-state sequential circuits for a restricted class of incompletely specified flow tables. Bell System Techn. Journal, v. 4, pp. 1759—1768, 1962.
73. McCluskey E. J., Transient behavior of combinational logic networks. In: Redundancy techniques for computing systems. Spartan Books, pp. 9—46, 1962.
74. McCluskey E. J., Introduction to the theory of switching circuits. McGraw-Hill, N. Y., 1965.
75. McCluskey E. J., Bartee T. C. (eds.)., A survey of modern switching circuit theory. McGraw-Hill, N. Y., 1962.
76. McNaughton R., Badly timed elements and well timed nets. Moore School Report No. 65-02, University of Pennsylvania, 1964.
77. Mealy G. H., A method for synthesizing sequential circuits. Bell System Techn. Journal, v. 34, pp. 1045—1079, 1955.
78. Millman J., Taub H., Pulse, digital, and switching waveforms. McGraw-Hill, N. Y., 1965.
79. Miller R., Switching theory, vol. I: Combinational circuits. John Wiley and Sons, N. Y., 1965. [Русский перевод: Миллер Р., Теория переключательных схем, т. 1: Комбинационные схемы. «Наука», М., 1970.]

80. Miller R., Switching theory, vol. II: Sequential circuits and machines. John Wiley and Sons, N. Y., 1965. [Русский перевод: Миллер Р., Теория переключательных схем, т. 2: Последовательностные схемы и машины. «Наука», М.,1970.]
81. Moore E. F., Gedanken experiments on sequential machines. In: Shannon C. T., McCarthy J. (eds.), Automata Studies, Princeton University Press, 1956. [Русский перевод: Мур Е. Ф., Умозрительные эксперименты с последовательностными машинами. В сб.: Автоматы, под ред. К. Э. Шеннона и Дж. Маккарти. ИЛ, М., 1963.]
82. Moore E. F., Sequential machines: selected papers. Addison-Wesley, Reading, Mass., 1964.
83. Muller D. E., Bartky W. S., A theory of asynchronous circuits I. Report No. 75, University of Illinois, Digital Computer Laboratory, 1956.
84. Muller D. E., Bartky W. S., A theory of asynchronous circuits II. Report No. 78, University of Illinois, Digital Computer Laboratory, 1957.
85. Muller D. E., Bartky W. S., A theory of asynchronous circuits. In: Proc. of an Intern. Symp. on the Theory of Switching, vol. 29, Harvard University Press, pp. 204—243, 1959.
86. Muller D. E., The general synthesis problem for asynchronous digital networks. In: IEEE Conference Record of 8th Annual Symp. on Switching and Automata Theory, pp. 71—82, 1967.
87. Muller D. E., Treatment of transition signals in electronic sequential circuits by algebraic methods. IRE Trans. Electron. Computers, v. EC-8, No. 3, p. 401, 1959.
88. Muller D. E., Asynchronous logics and application to information technology. In: Proc. of a Symp. on the Application of Switching Theory in Space Technology, Stanford University Press, 1962.
89. Narasimhan R., Minimizing incompletely specified sequential switching functions. IRE Trans. Electron. Computers, v. EC-10, No. 3, pp. 531—532, 1961.
90. Paull M. C., Unger S. H., Minimizing the number of states in incompletely specified sequential switching functions. IRE Trans. Electron. Computers, v. EC-8, No. 3, pp. 356—367, 1959.
91. Paull M. C., Waldbaum G., A note on state minimization of asyncronous sequential functions. IEEE Trans. Electron. Computers, v. EC-16, No. 1, pp. 94—97, 1967.
92. Paull M. C., The minimization of the number of states of a completely specified sequential function having restrictions on its allowable input sequences. Unpublished Bell Telephone Laboratories memorandum, 1966.
93. Phister M., Logic design of digital computers. John Wiley and Sons, N. Y., 1956. [Русский перевод: Фистер М., Логическое проектирование цифровых вычислительных машин, «Техніка», Киев, 1964.]
94. Prather R. E., Introduction to switching theory: a mathematical approach. Allyn and Bacon Inc., Boston, 1967.

95. Reed I. S., Some remarks on state reduction of asynchronous circuits by the Paull — Unger method. IEEE Trans. Electron. Computers, v. EC-14, No. 2, pp. 262—265, 1965.
96. Richards R. K., Arithmetic operations in digital computers. D. Van Nostrand, Princeton, N. J., 1955. [Русский перевод: Ричардс Р., Арифметические операции на цифровых вычислительных машинах. ИЛ, М., 1957.]
97. Robertson J. E., Problems in the physical realization of speed independent circuits. In: Proc. 2nd AIEE Symp. on Switching Circuit Theory and Logical Design, Detroit, Michigan, vol. S-134, pp. 106—108, 1961.
98. Saucier G., Encoding of asynchronous sequential networks. IEEE Trans. Electron. Computers, v. EC-16, No. 3, pp. 365—369, 1967.
99. Shannon C. E., McCarthy J. (eds.), Automata Studies. Princeton University Press, 1956. [Русский перевод: Автоматы, под ред. К. Э. Шеннона и Дж. Маккарти. ИЛ, М., 1963.]
100. Sims I. C., Gray H. J., Design criteria for asynchronous circuits. In: Proc. Eastern Joint Computer Conf., pp. 94—99, 1958.
101. Singh S., Asynchronous sequential switching circuits with feedback. Dept. Electrical Engineering, University of Ottawa, Techn. Report No. 68-9, 1968.
102. Smith R. J., Tracey J. H., Schoeffel W. L., Maki G. K., Automation in the design of asynchronous sequential circuits. In: Proc. Spring Joint Computer Conf., pp. 55—60, 1968.
103. Tan C. J., Note on general race-free assignment for asynchronous circuits. Unpublished note, Dept. Electrical Engineering, Columbia University, 1967.
104. Tan C. J., State assignment for asynchronous sequential machines. Bell Telephone Laboratory Report, 1967.
105. Tan C. J., Synthesis of asynchronous sequential switching circuits. Ph. D. Dissertation, Dept. Electrical Engineering, Columbia University, 1969.
106. Tan C. J., Computer aided design of asynchronous sequential switching circuits. In: First Annual Houston Conf. on Circuits, Systems and Computers, University of Houston, 1969.
107. Tan C. J., Menon P. R., Friedman A. W., Structural simplification and decomposition of asynchronous sequential circuits. IEEE Trans. Computers, v. C-18, No. 9, 1969.
108. Torng H. C., Introduction to the logical design of switching systems. Addison-Wesley, Reading, Mass., 1964.
109. Tracey J. H., Internal state assignments for asynchronous sequential machines. IEEE Trans. Electron. Computers, v. EC-15, No. 4, pp. 551—560, 1966.
110. Unger S. H., A theorem on feedback in sequential switching circuits. Quarterly Progress Report, Research Laboratory of Electronics, MIT, 1955.
111. Unger S. H., A study of asynchronous logical feedback networks. Techn. Report No. 320, Research Laboratory of Electro-

nics, MIT, 1957,См. также: Sc. D. Dissertation, Dept. of Electrical Engineering, MIT, 1957.
112. Unger S. H., Hazards and delays in asynchronous sequential switching circuits. IRE Trans. Circuit Theory, v. CT-6, pp. 12—25, 1959.
113. Unger S. H., Simplification of state tables. In: AIEE Fall Meeting, Chicago, 1960. См. также: McCluskey E. J., Bartee T. C. (eds.), A survey of modern switching circuit theory. McGraw-Hill, N. Y., 1962, pp. 145—170.
114. Unger S. H., Flow table simplification — some useful aids. IEEE Trans. Electron. Computers, v. EC-14, No. 3, pp. 472—475, 1965.
115. Unger S. H., A row assignment for delay-free realizations of flow tables without essential hazards. In: Proc. 7th Annual Symp. on Switching and Automata Theory, Berkeley California, 1966. (См. также:) IEEE Trans. Computers, v. C-17, No. 2, pp. 146—158, 1968.
116. Unger S. H., Timing considerations in clocked switching circuits. Quarterly Progress Report, Research Laboratory of Electronics, MIT, pp. 113—115, 1956.
117. Waite W. M., The production of completion signals by asynchronous iterative networks. IEEE Trans. Electron. Computers, v. EC-13, pp. 83—86, 1964.
118. Ware W. H., The logical principles of a new kind of binary counter. Proc. IRE, v. 41, No. 10, pp. 1429—1437, 1953.
119. Wood P. E., Switching theory. McGraw-Hill, N. Y., 1968.
120. Yoeli M., Rino S., Application of ternary algebra to the study of static hazards. Journal of ACM, v. 11, No. 1, pp. 84—97, 1964.
121. Younger D. H., Minimum feedback arc sets for a directed graph. IEEE Trans. Circuit Theory, v. CT-10, 1963.

Литература, добавленная при издании перевода

1. Кузнецов О. П., Временной анализ автономных логических сетей. Автоматика и телемеханика, № 5, 1965.
2. Левин В. И., Введение в динамическую теорию конечных автоматов. Рига, «Зинатне», 1975.
3. Лупанов О. Б., О схемах из функциональных элементов с задержками. Проблемы кибернетики, вып. 23, 1970, 43—81.
4. Миллерова М. Г., О временном анализе асинхронной логической сети. Автоматика и телемеханика, № 6, 1973, 162—165.
5. Рогинский В. Н., Динамические автоматы и временные булевы функции. Изв. АН СССР, сер. Техническая кибернетика, № 2, 1970.
6. Томфельд Ю. Л., О режимах работы контура обратной связи асинхронной логической сети. Автоматика и телемеханика, № 3, 1973.
7. Якубайтис Э. А., Синтез асинхронных конечных автоматов. Рига, «Зинатне», 1970.
8. Thayse A., Transient analysis of logical networks applied to hazard detection, Philips Research Reports, v. 25, No. 5, 1970.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ