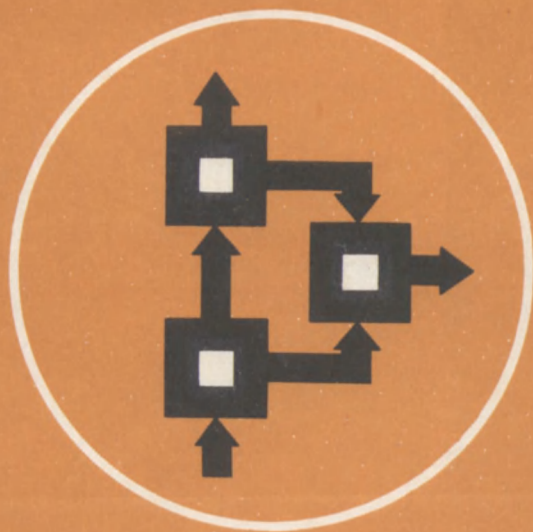


В. В. Сапожников, Вл. В. Сапожников

Самопроверяемые **ДИСКРЕТНЫЕ УСТРОЙСТВА**

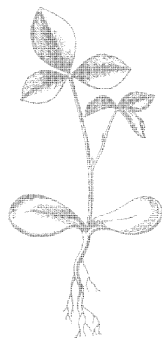


В.В.Сапожников, Вл.В.Сапожников

***Самопроверяемые* ДИСКРЕТНЫЕ УСТРОЙСТВА**



Санкт-Петербург
ЭНЕРГОАТОМИЗДАТ
Санкт-Петербургское отделение
1992



Scan AAW

ББК 32.965.6
С19
УДК 621.327

Сапожников В. В., Сапожников Вл. В.

С19 Самопроверяемые дискретные устройства. — СПб: Энергоатомиздат, Санкт-Петербургское отделение, 1992 — 224 с.: ил.

ISBN 5-283-04605-2

Посвящена вопросам схемотехники самопроверяемых цифровых устройств. Изложены принципы построения самопроверяемых дискретных устройств и их свойства. Рассмотрены основные цифровые схемы, методы синтеза и каталоги тестеров и дешифраторов для кодов с обнаружением ошибок. Описаны система самопроверяемых триггерных устройств и синтез самопроверяемых счетчиков, распределителей, регистров и других типовых узлов вычислительных систем. Приведены методы построения самопроверяемых программных реализаций дискретных устройств в микропроцессорных системах

Для инженерно-технических работников, занимающихся разработкой дискретных управляющих устройств и микропроцессорных систем.

С $\frac{2404020000-126}{051(01)-92}$ 96 — 92

ББК 32.965.6

Производственно-практическое издание

Сапожников Валерий Владимирович
Сапожников Владимир Владимирович

САМОПРОВЕРЯЕМЫЕ ДИСКРЕТНЫЕ УСТРОЙСТВА

Редактор *Ю. В. Долгополова*
Художник обложки *В. Л. Черников*
Художественный редактор *В. М. Мартынов*
Технический редактор *Ю. А. Богданова*
Корректор *Л. А. Мартянова*
ИБ № 3332

Сдано в набор 28.01.92. Подписано в печать 17.11.92. Формат 60×90 1/16.

Бумага типографская № 2. Гарнитура литературная. Высокая печать.

Усл. печ. л. 14 Усл. кр.-отт. 14,25. Уч.-изд. л. 16,77. Доп. тир. 400. Заказ 110 С126.

Энергоатомиздат, Санкт-Петербург, отделение. 191065. С.-Петербург, Д-65, Марсово поле, 1.

Типография ВНИИГ им. Б. Е. Веденеева. 195220 С.-Петербург, Гжатская ул. 21.

ISBN 5-283-04605-2

© В. В. Сапожников,
Вл. В. Сапожников,
1992

ПРЕДИСЛОВИЕ

Самопроверяемость как свойство дискретной аппаратуры все шире используется в инженерной практике при построении управляющих и вычислительных систем. Это свойство позволяет эффективно решать многие проблемы повышения надежности и контролепригодности дискретных устройств.

Самопроверяемой называется система, у которой в процессе ее «служебного» функционирования обеспечивается контроль исправности внутренних элементов и которая в случае возникновения отказа формирует сигнал об этом на специальном контрольном выходе. Таким образом, система обладает некоторыми внутренними ресурсами для того, чтобы контролировать собственную работу. Это качество совместно с отказоустойчивостью (дублированием или троированием) дает возможность добиться высоких показателей надежности (вероятности безотказной работы и установленного коэффициента готовности).

Идеи самопроверяемости появились в литературе по дискретной технике в начале 70-х годов. Это было связано, в частности, с решением проблемы «последнего сторожа». Если некоторое контрольное устройство проверяет исправную работу дискретной системы, то необходимо контролировать и работу этого устройства. Естественным решением данной проблемы является применение в качестве «последнего сторожа» элементов, которые сами себя контролируют, — самопроверяемых элементов.

В настоящее время разработаны методы построения самопроверяемых комбинационных схем и схем с памятью, методы организации самопроверяемого контроля резервированных дискретных систем. Перспективным является создание самопроверяемых интегральных схем различного функционального назначения.

Данная книга посвящена изложению результатов, полученных в области самопроверяемой схмотехники. Наличие самопроверяемых триггерных устройств, счетчиков, распределителей, дешифраторов и других типовых цифровых устройств позволяет сравнительно просто решать проблему построения сложной

самопроверяемой дискретной системы. Особое значение имеет разработка всего разнообразия самопроверяемых контрольных схем, предназначенных для обнаружения отказов. В книге описываются каталоги контрольных схем, позволяющие осуществлять их выбор в соответствии с конкретными требованиями. В связи с широким применением в настоящее время микропроцессорных систем рассмотрен вопрос о свойствах программных реализаций самопроверяемых устройств.

Книга предназначена для инженеров и научных работников, занимающихся проблемами анализа и синтеза дискретных управляющих устройств.

Отзывы о книге, замечания и пожелания просим высылать по адресу: 191065, Санкт-Петербург, Марсово поле, д. 1, Санкт-Петербургское отделение Энергоатомиздата.

Авторы

ПРИНЦИПЫ ПОСТРОЕНИЯ САМОПРОВЕРЯЕМЫХ ДИСКРЕТНЫХ УСТРОЙСТВ

1.1. Способы задания дискретных устройств

Дискретные устройства входят в качестве составных частей в дискретные системы управления, которые широко используются во всех областях техники. Данные системы осуществляют управление разнообразными объектами, назначение которых состоит в реализации некоторых технических процессов. В настоящее время наиболее распространены двухпозиционные, или двоичные, объекты управления, имеющие два состояния: включено и выключено. Сигналы, поступающие на входы дискретного устройства (ДУ), также в большинстве случаев являются двоичными (сигналы от кнопок, рукояток, переключателей, датчиков и т. п.) либо могут быть в противном случае легко преобразованы в двоичные сигналы. В дальнейшем будут рассматриваться ДУ, управляющие двухпозиционными объектами (а следовательно, имеющие двоичные выходные сигналы), на входы которых поступают внешние сигналы и внутренняя структура которых реализована на двоичных элементах, имеющих два внутренних состояния.

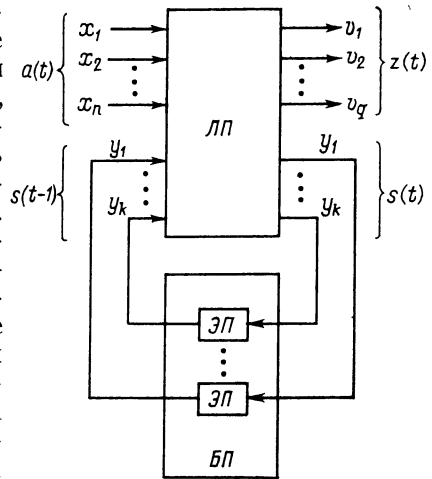


Рис. 1.1

Абстрактной моделью дискретных устройств является конечный автомат (рис. 1.1)

$$M = (A, S, V, \delta, \kappa),$$

где $A = \{a_1, a_2, \dots, a_r\}$ — входной алфавит или множество входных слов, являющихся векторами состояний входных переменных x_1, x_2, \dots, x_n ; $S = \{s_1, s_2, \dots, s_p\}$ — внутренний алфавит или множество состояний автомата, являющихся векторами состояний внутренних переменных y_1, y_2, \dots, y_k ; $Z = \{z_1, z_2, \dots, z_t\}$ — выходной алфавит или множество слов, являющихся

пекторами состояний выходных переменных v_1, v_2, \dots, v_q ; δ — функция переходов, задающая отображение $A \times S \rightarrow S$; κ — функция выходов, задающая отображение $A \times S \rightarrow Z$.

Структура конечного автомата (КА) содержит два основных блока (см. рис. 1.1). Логический преобразователь (ЛП) осуществляет вычисление функций δ и κ . Блок памяти (БП) производит задержку внутренних сигналов y на один такт работы автомата, для чего служат специальные элементы памяти (ЭП).

Подробно с вопросами анализа и синтеза конечных автоматов можно познакомиться в книгах [11, 20, 29, 32]. В данном параграфе мы приведем только основные сведения, необходимые для изложения последующих разделов.

Конечные автоматы подразделяются на две группы: автоматы без памяти, или комбинационные схемы (КС), и автоматы с памятью, или многотактные схемы (МС). Структура КС содержит только один логический преобразователь. При этом каждому состоянию входных переменных $a(t)$ в любой момент времени соответствует одно и то же состояние выходных переменных $v(t)$. Комбинационные устройства описываются функциями алгебры логики (ФАЛ), которые задаются таблицами истинности (ТИ).

Рассмотрим в качестве примера устройство, осуществляющее преобразование двоичных трехразрядных векторов (x_1, x_2, x_3) в двоичные двухразрядные векторы (v_1, v_2) . На выходах v_1 и v_2 формируется вектор, отражающий число единичных разрядов

Таблица 1.1

x_1	x_2	x_3	v_1	v_2
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

в соответствующем трехразрядном векторе. Устройство описывается табл. 1.1, которая представляет собой ТИ, задающую две ФАЛ: $v_1 = f(x_1, x_2, x_3)$ и $v_2 = f(x_1, x_2, x_3)$. Задача синтеза КС состоит из двух этапов. На первом вычисляются и минимизи-

руются функции, описывающие выходы схемы. На втором этапе по полученным формулам реализуется схема.

Вопросы минимизации ФАЛ подробно рассмотрены в [29]. В данном случае имеем

$$\begin{aligned}v_1 &= x_1x_2 \vee x_1x_3 \vee x_2x_3, \\v_2 &= \bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1x_2\bar{x}_3 \vee x_1\bar{x}_2\bar{x}_3 \vee x_1x_2x_3.\end{aligned}$$

Работа конечного автомата с памятью задается двумя уравнениями:

$$s(t) = \delta(a(t), s(t-1)), \quad (1.1)$$

$$z(t) = \kappa(a(t), s(t-1)), \quad (1.2)$$

из которых следует, что внутреннее состояние автомата и состояние его выходов в данный момент времени t (в данный такт) однозначно определяются состоянием входов в данный момент времени и внутренним состоянием в предшествующий момент времени $t-1$.

Наиболее наглядным является табличный способ задания КА. В этом случае функции δ и κ задаются с помощью таблиц переходов (ТП) и таблиц выходов (ТВ) или с помощью одной совмещенной таблицы. Столбцы и строки этих таблиц соответствуют элементам множеств A и S , а клетки — элементам множества $A \times S$. Внутри клеток проставляются состояния s (в ТП) и v (в ТВ), которые определяются функциями δ и κ для соответствующих пар $(a, s) \in A \times S$.

В табл. 1.2 приведен пример совмещенной таблицы для синхронного КА. Автомат имеет два входных (a_1, a_2), два выходных (z_1, z_2) и четыре внутренних (обозначены десятичными числами) состояния. Алгоритм работы автомата предусматривает смену выходного состояния в каждом такте его работы. Таблицы переходов и выходов составляются непосредственно по словесному описанию алгоритма работы автомата. Такая процедура называется абстрактным синтезом. Она подробно рассмотрена в [11, 32]. ТП и ТВ являются исходным заданием на структурный синтез автомата. На первом этапе структурного синтеза производится кодирование входных, внутренних и выходных состояний КА. Задача кодирования состоит в сопоставлении с каждым состоянием определенного набора значений соответствующих переменных автомата таких, чтобы различным состояниям соответствовали различные наборы переменных.

На втором этапе структурного синтеза осуществляется построение ЛП автомата, который представляет собой комбинационную схему. Для этого сначала вычисляются функции включения (ФВ) элементов памяти и функции выходов, после чего производится их минимизация и по полученным выражениям строится схема ЛП. Способы вычисления ФВ определяются видом используемых для построения автомата ЭП.

Минимальное число двоичных переменных, необходимых для кодирования h состояний, вычисляется по формуле

$$\mu = \log_2 H = \lceil \log_2 h \rceil, \quad (1.3)$$

где $H \geq h$ — ближайшее к h целое число, при котором $\log_2 H$ принимает целочисленное значение.

Если при кодировании синхронного КА никаких других дополнительных задач не решается, то процедура кодирования состоит в присвоении каждому входному состоянию некоторого двоичного кодового вектора длины $\lceil \log_2 n \rceil$, каждому внутреннему состоянию — вектора длины $\lceil \log_2 p \rceil$ и каждому выход-

Таблица 1.2

s	a	
	a_1	a_2
1	2, z_1	4, z_2
2	3, z_2	1, z_1
3	4, z_1	2, z_2
4	1, z_2	3, z_1

Таблица 1.3

s	x	
	0	1
1	2, 0	4, 1
2	3, 1	1, 0
3	4, 0	2, 1
4	1, 1	3, 0

ному состоянию — вектора длины $\lceil \log_2 q \rceil$, причем различным состояниям присваиваются различные векторы. Так для кодирования входных и выходных состояний автомата, заданного табл. 1.2, требуется только по одной соответствующей переменной (x и v). Выберем следующий вариант кодирования: $a_1 = 0$, $a_2 = 1$, $v_1 = 0$, $v_2 = 1$. Тогда получаем новую таблицу (табл. 1.3), задающую автомат. Часто при формулировке задачи синтеза автомата входные и выходные состояния даются закодированными и поэтому приходится решать только задачу кодирования внутренних состояний. В этом случае автомат задается таблицей типа табл. 1.3.

Рассматриваемый КА имеет четыре внутренних состояния. Следовательно, для его кодирования требуется $\mu = \lceil \log_2 4 \rceil = 2$ внутренних переменных. Обозначим их через y_1 и y_2 . Возможный вариант кодирования представлен в табл. 1.4. Для выбранного варианта кодирования вычисляются ФВ и выходные функции. Способ вычисления выходных функций не зависит от вида применяемых для построения автомата ЭП. При этом составляется закодированная таблица выходов (табл. 1.5). Она строится непосредственно по заданной совмещенной ТП путем замены в ее первом столбце цифровых обозначений состояний поставленными им в соответствие кодовыми векторами. От закодированной ТВ осуществляется переход к любому известному способу задания ФАЛ, например, к описанной выше ТИ. Затем

Таблица 1.4

s	y	
	y ₁	y ₂
1	0	0
2	0	1
3	1	1
4	1	0

Таблица 1.5

s	x	
	0	1
0 0	0	1
0 1	1	0
1 1	0	1
1 0	1	0

осуществляется минимизация ФАЛ. Табл. 1.5 задает функцию $v = f(x, y_1, y_2)$. Для нее получаем

$$v = \bar{x}(\bar{y}_1 y_2 \vee y_1 \bar{y}_2) \vee x(\bar{y}_1 \bar{y}_2 \vee y_1 y_2).$$

Способы вычисления ФВ рассмотрим для случаев применения различных видов ЭП. Часто для построения автоматов используются ЭП без фиксации воздействия, к которым относятся линии задержки и электромагнитные нейтральные реле. В этом случае для вычисления ФВ строится кодированная таблица переходов (табл. 1.6), которую получают непосредственно из ТП путем замены во всех ее клетках цифровых обозначений состояний поставленными им в соответствие кодовыми векторами. Кодированная ТП задает одновременно все ФВ автомата. По ней так же, как по кодированной ТВ, находят функции, описывающие ЛП автомата:

$$y_1 = \bar{x}y_2 \vee xy_2,$$

$$y_2 = \bar{x}y_1 \vee xy_1.$$

Широкое применение при построении КА находят различного вида триггеры. Триггер представляет собой элементарный авто-

Таблица 1.6

s	x	
	0	1
0 0	0 1	1 0
0 1	1 1	0 0
1 1	1 0	0 1
1 0	0 0	1 1

Таблица 1.7

t^n	t^{n+1}	
D^n T^n	y^{n+1}	
	Тип триггера	
	D	T
0	0	y^n
1	1	\bar{y}^n

мат с двумя внутренними состояниями: 0 и 1. Элементарные триггеры имеют один или два входа и два выхода. Триггер с одним входом переключается из одного состояния в другое по этому входу в различные моменты времени. У триггера с двумя входами один из них используется для перевода его в состояние 1, а другой — в состояние 0. Сигналы на выходах триггера являются взаимно обратными. На прямом выходе y сигнал логической 1 присутствует тогда, когда триггер находится в состоянии 1, а на инверсном выходе — при нахождении его в состоянии 0.

Триггеры различаются по логическим условиям работы, которые задаются специальными таблицами переходов. Основными являются D -, T -, RS -, JK -, E -, R - и S -триггеры. D - и T -триггеры имеют один вход управления, остальные триггеры — два. Таблицы 1.7 и 1.8 являются таблицами переходов указанных триггеров. В табл. 1.7 входные логические сигналы в момент времени t^n обозначены D^n и T^n соответственно для D - и T -триггеров. Аналогично в табл. 1.8 входные логические сигналы на входах S и R для RS -, E -, R - и S -триггеров обозначены S^n и R^n , для JK -триггера — J^n и K^n . Состояния триггера Y в моменты времени t^n и t^{n+1} , следующие один за другим, обозначены Y^n и Y^{n+1} .

Таблица 1.8

t^n		t^{n+1}				
$S^n,$ J^n	$R^n,$ K^n	Y^{n+1}				
		Тип триггера				
		RS	JK	E	R	S
0	0	Y^n	Y^n	Y^n	Y^n	Y^n
1	0	1	1	1	1	1
0	1	0	0	0	0	0
1	1	\sim	\bar{Y}^n	Y^n	0	1

В клетке таблицы переходов указывается состояние, в которое приходит триггер в момент времени t^{n+1} при поступлении на его входы в момент t^n сигналов, указанных в строке, где расположена данная клетка. Из табл. 1.8 видно, что по входам S и J соответствующие триггеры переводятся в состояние 1, а по входам R и K — в состояние 0. Работа триггеров с двумя входами отличается только при одном наборе значений входных сигналов, когда $S = R = 1$, причем поведение RS -триггера на этом наборе не определено.

При синтезе КА на триггерах правила вычисления ФВ определяются логикой работы триггера. Из табл. 1.7 следует, что *D*-триггер представляет собой линию задержки, для которой правила вычисления ФВ сформулированы выше.

Для управления *T*-триггером при синтезе КА необходимо вычислить функцию y_T , определяющую сигналы, которые воздействуют на вход *T*-триггера. Для управления *JK*-триггером вычисляются две функции y_J и y_K , определяющие воздействия соответственно на входы *J* и *K*. Для управления *RS*-, *E*-, *R*- и *S*-триггерами вычисляются функции y_S и y_R . Для данного входа триггера составляется специальная таблица воздействия. Такая таблица для входа *T*-триггера представлена табл. 1.9. В клетке, расположенной на пересечении строки Y^n и столбца Y^{n+1} , проставляется значение сигнала (0 или 1), который необходимо сформировать на данном входе, если триггер изменяет свое состояние со значения Y^n (в такте t^n) на значение Y^{n+1} (в такте t^{n+1}). Например, в табл. 1.9 в двух клетках проставлены сигналы 0 (это соответствует случаю, когда при переходе автомата из такта t^n в такт t^{n+1} триггер сохраняет свое состояние неизменным) и в двух клетках — сигналы 1 (соответствуют случаю изменения состояния триггера).

Таблица 1.9

Y^n	Y^{n+1}	
	0	1
0	0	1
1	1	0

Таблица 1.10

s	x	
	0	1
0 0	0 1	1 0
0 1	1 0	0 1
1 1	0 1	1 0
1 0	1 0	0 1

С помощью таблицы воздействия на вход триггера по кодированной ТП (см. табл. 1.6) составляется таблица задания функции y_T (табл. 1.10). Структура этой таблицы совпадает со структурой кодированной ТП, а в ее клетках проставляются значения функций y_{T1} , y_{T2} , вычисленные с помощью табл. 1.9. При этом в качестве значений Y^n выбираются значения состояний соответствующих ЭП в коде строки, где расположена данная клетка, а в качестве значений Y^{n+1} — значения состояний ЭП в коде состояния, расположенного в данной клетке кодированной ТП. Например, в табл. 1.6 на пересечении строки 00 и столбца 0 проставлено состояние 01. Это значит, что первый ЭП в данном такте работы автомата не изменяет своего состояния и поэтому в соответствии с табл. 1.9 $y_{T1} = 0$, а второй ЭП изменяет свое состояние со значения $Y^n = 0$ на значение $Y^{n+1} = 1$ и поэтому $y_{T2} = 1$.

По табл. 1.10 получаем:

$$y_{T1} = \bar{x}(\bar{y}_1 y_2 \vee y_1 \bar{y}_2) \vee x(\bar{y}_1 \bar{y}_2 \vee y_1 y_2),$$

$$y_{T2} = \bar{x}(\bar{y}_1 \bar{y}_2 \vee y_1 y_2) \vee x(\bar{y}_1 y_2 \vee y_1 \bar{y}_2).$$

Для триггеров с двумя входами управления составляются две таблицы воздействия [34] в соответствии с логикой их работы.

На практике широко распространен класс асинхронных автоматов (АКА), для которых выполняется условие: если $\delta(a, s_i) = s_j$, то $\delta(a, s_j) = s_j$. В них изменение внутреннего состояния всегда связано с изменением входного состояния, что позволяет отказаться от синхронизации сигналов. Длительность такта

Таблица 1.11

s	a	
	a ₁	a ₂
1	(1), 0	2
2	5	(2), 1
3	(3), 1	2
4	3	(4), 0
5	(5), 0	4

в асинхронных автоматах неодинакова и определяется временем сохранения без изменения входных сигналов.

В табл. 1.11 приведен пример совмещенной таблицы переходов АКА, в которой закодированы выходные состояния. Каждая клетка таблицы соответствует полному состоянию автомата, которое определяется значениями входных и внутренних переменных. Полное состояние (a_j, s_i) называется устойчивым, если $\delta(a_j, s_i) = s_i$, т. е. если оно сохраняется сколь угодно долго вплоть до изменения значений входов. Устойчивые состояния

в ТП заключаются в круглые скобки. Остальные состояния являются неустойчивыми. Если в столбце a_j содержится γ_i устойчивых состояний, то на полном множестве состояний S образуется разбиение $L_j = \{\lambda_1, \lambda_2, \dots, \lambda_{\gamma_j}\}$, классы λ которого содержат устойчивое состояние и все неустойчивые состояния, из которых определены переходы в данное устойчивое состояние. Например, для табл. 1.11 имеем: $L_1 = \{\lambda_1 = \{1, 4\}, \lambda_2 = \{2, 3\}, \lambda_3 = \{5\}\}$; $L_2 = \{\lambda_4 = \{1, 2\}, \lambda_5 = \{3, 4, 5\}\}$.

При синтезе АКА необходимо учитывать вероятность состязаний между ЭП, которые могут приводить к искажениям в работе автомата. В синхронных КА проблема состязаний снимается за счет установления с помощью внешних сигналов строгой последовательности срабатывания элементов. Состязания возникают в результате различия во времени срабатывания различных элементов. По этой причине, если в абстрактном автомате запланирован переход между состояниями схемы s_i и s_j , отличающимися друг от друга состояниями двух или более ЭП, то при его реальной реализации результат данного перехода будет зависеть от соотношения между временами переключения указанных ЭП. При соответствующем соотношении возможен переход КА вместо состояния s_j в некоторое незапла-

нированное третье состояние, в результате чего происходит искажение алгоритма работы автомата. Подробно с явлением состязаний и способами их устранения можно познакомиться в [20, 32].

Для синтеза рассматриваемого в данной книге класса устройств наиболее приемлемым способом устранения состязаний является метод кодирования состояния автомата по столбцам таблицы переходов (или метод КСТ), разработанный в [33, 63]. Данный метод сводит все возможные состязания к допустимым, т. е. таким, которые не приводят к искажению алгоритма работы АКА. Особенность метода состоит в том, что классы λ каждого разбиения L_j ($j \in R$, R — множество индексов столбцов ТП) кодируются независимо от классов λ других разбиений L_r ($j \neq r$) автомата. После этого каждому состоянию АКА присваивается кодовое слово, образуемое как совокупность кодовых слов, поставленных в соответствие тем классам λ , в которые входит данное состояние. Метод реализуется с помощью следующего алгоритма.

Алгоритм 1.1:

1. Классы λ каждого разбиения L_j кодируются произвольным образом различными кодовыми словами длиной $\lceil \log_2 \gamma_j \rceil$. Внутренние переменные автомата и ЭП, соответствующие рядам этих слов, называются определяющими по столбцу a_j , а их множество обозначается через Ω_j . При этом не рассматриваются столбцы a_j , не содержащие неустойчивых состояний (множество их индексов обозначим через R_1), и столбцы a_j , содержащие только одно устойчивое состояние.

2. Общее кодовое слово данной строки ТП образуется как совокупность кодовых слов, соответствующих тем классам λ , в которые входит данная строка. Поэтому число ЭП, необходимое для построения автомата,

$$\mu = \sum_{j \in R / R_1} \lceil \log_2 \gamma_j \rceil. \quad (1.4)$$

3. Функции включения ЭП без фиксации воздействия определяются непосредственно по ТП для данного кодирования ее строк по следующей стандартной формуле:

$$y_f = (a_q \vee (\bigvee_{j \in R_1} a_j)) y_f \vee (\bigvee_{j \in R_2} a_j) \vee \left(\bigvee_{j \in R / (q \cup R_1 \cup R_2)} a_j \left(\bigvee_{k \in N_{fj}} \tilde{y}_{i_1} \tilde{y}_{i_2} \dots \tilde{y}_{i_p} \right) \right), \quad (1.5)$$

где q — индекс такого столбца ТП, что $y_f \in \Omega_q$; R_2 — множество индексов столбцов ТП, в которых расположено только одно устойчивое состояние, размещенное в строке, в коде которой $y_f = 1$; N_{fj} — множество индексов строк, в коде которых $y_f = 1$ и на пересечении которых со столбцом a_j расположено устойчи-

вое состояние; i_1, i_2, \dots, i_p — индексы внутренних переменных, которые являются определяющими по столбцу a_j ; $\tilde{y}_{i_1}, \tilde{y}_{i_2}, \dots, \tilde{y}_{i_p}$ — значения определяющих переменных в строке с индексом k .

4. Выходные функции определяются по формуле

$$v_g = (\bigvee_{j \in R_3} a_j) \vee (\bigvee_{j \in R \setminus R_3} a_j (\bigvee_{k \in N_{gj}} \tilde{y}_{i_1} \tilde{y}_{i_2} \dots \tilde{y}_{i_p})), \quad (1.6)$$

где R_3 — множество индексов столбцов ТП, содержащих только одно устойчивое состояние, которому соответствует значение $v_g = 1$; N_{gj} — множество индексов

Таблица 1.12

s	a_1		a_2
	y_1	y_2	y_3
1	0	0	0
2	1	0	0
3	0	1	0
4	0	1	1
5	1	0	1

строк, на пересечении которых со столбцом a_j расположено устойчивое состояние, которому сопоставлено значение $v_g = 1$; для столбцов с индексами $j \in R_1$ предполагается, что определяющими являются все внутренние переменные автомата.

Пример 1.1. Применим алгоритм 1.1 для синтеза АКА, заданного табл. 1.11. Так как $\gamma_1 = 3$ и $\gamma_2 = 2$, то для кодирования автомата требуются две определяющие переменные по столбцу a_1 и одна переменная по

столбцу a_2 . В табл. 1.12 приведено кодирование строк ТП, при котором $\Omega_1 = \{y_1, y_2\}$ и $\Omega_2 = \{y_3\}$. С помощью формул (1.4) и (1.5) находим функции, описывающие структуру автомата:

$$\begin{aligned} y_1 &= a_1 y_1 \vee a_2 \bar{y}_3, & y_3 &= a_1 y_1 \bar{y}_3 \vee a_2 y_3, \\ y_2 &= a_1 y_2 \vee a_2 y_3, & v &= a_1 \bar{y}_1 y_2 \vee a_2 \bar{y}_3. \end{aligned}$$

При синтезе АКА на триггерах формула (1.5) преобразуется в соответствии с особенностями управления триггером, которые зафиксированы в вышеприведенных таблицах воздействия на входы триггеров. Соответствующие формулы приведены в [33].

1.2. Свойства самопроверяемых дискретных устройств

Важнейшим свойством ДУ является надежность функционирования. Неисправности элементов, принадлежащих внутренней структуре ДУ, приводят к искажению его функций переходов и выходов, а следовательно, к выходу из строя самого устройства. Повышение надежности достигается за счет введения в структуру устройства избыточных элементов. При этом может быть достигнут эффект исключения влияния отказов элементов на

работу ДУ, что требует большой избыточности [33]. Другой подход, требующий существенно меньшей избыточности, связан с решением задачи обнаружения отказов. В дальнейшем будет рассматриваться только этот подход.

Для обнаружения отказов элементов ДУ снабжается схемой контроля (СК), назначение которой состоит в формировании специального сигнала при возникновении неисправности (рис. 1.2). Будем называть этот сигнал сигналом ошибки. Он может быть использован для отключения неисправного устройства от объекта управления (ОУ) с помощью специальных устройств переключения (УП). Общая структура системы управления для этого случая приведена на рис. 1.3. Сигнал ошибки может также осуществлять подключение ОУ к резервному устройству управления.

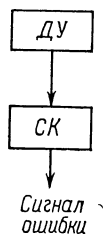


Рис. 1 2

Возможны два подхода к построению ДУ с обнаружением отказов. Первый подход проиллюстрирован на рис. 1.4. Его идея состоит в следующем. Для заданного ДУ строится схема встроенного контроля (СВК), входы которой соединяются со входами и выходами устройства. СВК производит анализ работы ДУ на основе сравнения поступающих на вход устройства последовательностей сигналов с его реакциями на выходе. Практически в этом случае осуществляется контроль работы ДУ в процессе функционирования (функциональное диагностирование). Простейший пример реализации СВК приведен на рис. 1.5.

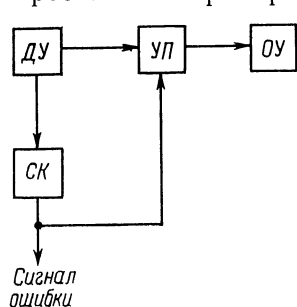


Рис. 1.3

В нее входят второй комплект ДУ и схема сравнения (СС), которая осуществляет сравнение соответствующих входных сигналов двух параллельно работающих устройств. Сигнал ошибки f вырабатывается при несоответствии выходных сигналов.

При организации функционального диагностирования возникает задача контроля исправности СВК (проблема «сторожа над сторожем»). В системах с обнаружением отказов часто надежность СВК оказывает существенное влияние на

надежность всей системы. Подробно с вопросами оценки надежности ДУ с обнаружением отказов можно познакомиться в [47].

Неисправность СВК может приводить, с одной стороны, к тому, что неправильно функционирующее ДУ будет воздействовать на объект управления, а с другой — к отключению ОУ от правильно работающего устройства управления. И тот и другой случай нежелательны при эксплуатации системы управления, особенно тогда, когда осуществляется управление ответственными технологическими процессами.

Задача контроля исправности СВК решается за счет придания ДУ свойства самопроверяемости. В общем плане самопроверяемость есть способность системы обнаруживать отказы как в основном устройстве, так и в схемах контроля в процессе нормального функционирования без дополнительной подачи на входы устройства специальных проверочных тестов или других способов его испытания. При функциональном диагностировании самопроверяемость достигается в том случае, когда по значению

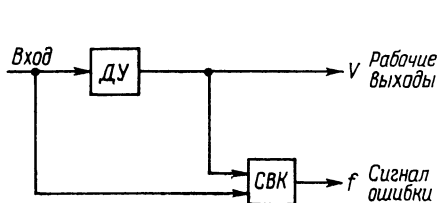


Рис. 1.4

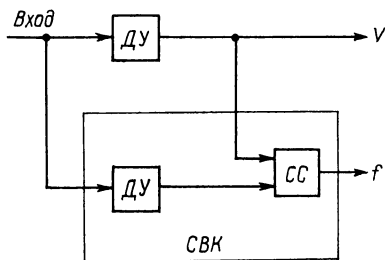


Рис. 1.5

сигнала ошибки f можно судить об исправности не только основного ДУ, но и СВК.

Системы (устройства, схемы), обладающие свойством самопроверяемости, будем называть самопроверяемыми системами. Вопросы построения самопроверяемых ДУ, реализуемых в соответствии со структурной схемой рис. 1.4, всесторонне рассмотрены в [28, 45].

Второй подход к построению ДУ с обнаружением отказов проиллюстрирован на рис. 1.6. Его особенность состоит в том,

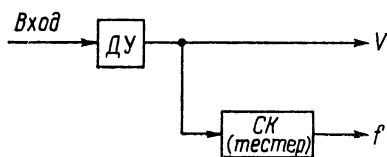


Рис. 1.6

что избыточность, которая является ценой за свойство обнаружения ошибок, вносится во внутреннюю структуру автомата (в системе, показанной на рис. 1.4, избыточность полностью вносится в СВК). В результате улучшается контролепригодность ДУ и оно приобретает такие

свойства, которые позволяют легко фиксировать неисправности. Внешняя СК при этом существенно упрощается или является наперед заданной.

Указанные свойства ДУ достигаются за счет применения при его синтезе идей помехоустойчивого кодирования. При этом в качестве помех рассматриваются отказы внутренних элементов устройства. Для кодирования входных, внутренних и выходных состояний ДУ используются коды с обнаружением ошибок. Отказы элементов ДУ приводят к искажению кодов состояний,

что может быть зафиксировано с помощью специальных схем, определяющих факт принадлежности кодового вектора заданному коду. Будем называть такие схемы тестерами.

Данный подход к контролю работы ДУ определим как тестово-функциональное диагностирование. Он отличается от тестового диагностирования тем, что на вход устройства не подаются специальные тестовые последовательности, формируемые внешним устройством, но в то же время внутренняя структура ДУ строится таким образом, что рабочие воздействия составляют его проверяющий тест. От рассмотренного выше функционального диагностирования отличие состоит в том, что здесь отсутствует анализ значений выходных сигналов ДУ в зависимости от поступающих входных последовательностей, но в то же время проверяющий тест на вход устройства поступает в процессе его нормального функционирования. Тестово-функциональное диагностирование сочетает в себе достоинства как тестового (возможность сокращения длины теста и времени), так и функционального (меньший объем аппаратуры) диагностирования.

Вопросы построения самопроверяемых ДУ в соответствии с принципами тестово-функционального диагностирования рассмотрены в [34]. Зафиксируем свойства таких ДУ в терминах

теории автоматов. Выделим на полном множестве выходных \hat{Z} и внутренних \hat{S} состояний автомата множества основных (рабочих) состояний Z и S и некоторые множества защитных (ошибочных) состояний Z_R и S_R таких, что выполняются условия $Z \cup Z_R = \hat{Z}$, $Z \cap Z_R = \emptyset$, $S \cup S_R \subseteq \hat{S}$, $S \cap S_R = \emptyset$.

Определение 1.1. Конечный автомат называется защищенным от неисправностей, если при возникновении любой неисправности из заданного класса на любой рабочей входной последовательности выходные состояния либо вычисляются правильно в соответствии с функциями переходов и выходов, либо принадлежат множеству защитных состояний Z_R .

Определение 1.2. Конечный автомат называется самотестируемым, если для каждой неисправности из заданного класса существует хотя бы одна рабочая входная последовательность, на которой появляется хотя бы одно выходное состояние, принадлежащее множеству защитных состояний Z_R .

Защищенность от неисправностей исключает неправильные воздействия со стороны управляющего автомата на ОУ. Свойство самотестируемости исключает в схеме автомата необнаруживаемые неисправности заданного класса и их накопление. При этом рабочие входные последовательности составляют одновременно и проверяющий тест. Таким образом, автомат, удовлетворяющий условиям определений 1.1 и 1.2, в указанном смысле сам себя контролирует.

Определение 1.3. Конечный автомат называется полностью самопроверяемым (ПСП-автомат, ПСП КА), если он защищен от неисправностей и является самотестируемым.

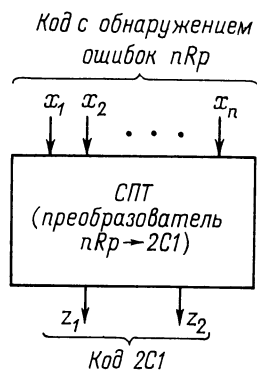


Рис. 1.7

ПСП-автомат в принципе не требует схемы контроля, так как при возникновении отказа его выходы как бы отключаются от ОУ и не оказывают на него неправильного воздействия. Однако на практике, как правило, ПСП-схемы снабжаются СК, которая необходима для выработки сигнала об отказе.

В качестве СК в ПСП-автоматах используются самопроверяемые тестеры (СПТ), которые представляют собой устройства (рис. 1.7), имеющие n входов и два выхода. На входы подаются слова n -разрядного кода с обнаружением ошибок nRp (R — признак кода, p — параметр, характеризующий код), с помощью которых кодируются состояния автомата. СПТ обладает следующими свойствами: 1) контроля входного вектора — выходы z_1 и z_2 принимают значения (1, 0) или (0, 1), если на входе тестера присутствует вектор кода nRp , и принимают значения (0, 0) или (1, 1) в противном случае; 2) самопроверки — для любой неисправности схемы тестера из заданного класса существует входной вектор кода nRp , на котором выходы z_1 и z_2 принимают значения 00 и 11. С функциональной точки зрения СПТ представляет собой комбинационную схему, осуществляющую преобразование кода nRp в код «1 из 2».

Тестер с указанными свойствами позволяет свести контроль исправности ДУ любой сложности к процедуре сравнения двух сигналов (z_1 и z_2) на выходе СПТ. Общая структура ДУ с обнаружением отказов в этом случае показана на рис. 1.8. Сравнение сигналов z_1 и z_2 производится с помощью специальной схемы сравнения СС. Так как такая схема устанавливается только одна для ДУ любой сложности, то достичь ее высокой надежности можно путем многократного резервирования или с помощью специальных высоконадежных элементов.

Важной особенностью тестово-функционального диагностирования является то, что оно позволяет подойти к решению задачи построения типовых цифровых ПСП-устройств (триггеров, счетчиков, дешифраторов и т. п.). Необходимый набор таких устройств позволит вести построение дискретных систем

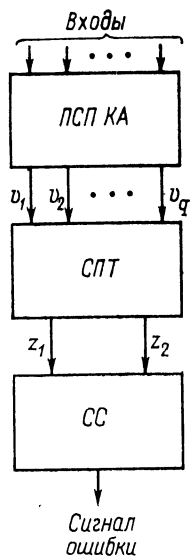


Рис. 1.8

управления обычными методами, используемыми для построения безызбыточных систем, не обладающих свойством обнаружения отказов. При этом данное свойство система приобретает автоматически. Цифровые ПСП-устройства могут быть реализованы в виде типовых микросхем. В этом случае избыточность, вносимая во внутреннюю структуру ПСП-устройств, не имеет существенного значения, так как размеры микросхем при современной технологии определяются в основном не числом внутренних компонентов схемы, а числом входов и выходов.

Таким образом, можно говорить о проблеме создания ПСП-схемотехники. Данная книга является шагом к решению этой проблемы. При этом возникают следующие задачи: определение необходимого набора ПСП-устройств, разработка методик синтеза ПСП-устройств различного типа, получение логических структур основных ПСП-устройств и построение их каталогов, разработка функциональных схем ПСП-устройств с учетом помехозащищенности и состязаний между логическими сигналами, разработка вопросов согласования ПСП-устройств по входам и выходам. Для обычной схемотехники эти задачи рассматриваются в целом ряде книг (например, [8, 30]).

1.3. Принципы построения самопроверяемых схем

Методы синтеза ПСП-автоматов подробно рассмотрены в [34]. В дальнейшем будем говорить, что ПСП-автомат обладает ПСП-свойством, т. е. свойством полной самопроверки (в соответствии с определением 1.3).

Основная идея обеспечения ПСП-свойства автомата состоит в следующем. При возникновении неисправности заданного класса во внутренней структуре автомата обеспечивается его переход из множества основных внутренних состояний S в множество ошибочных состояний S_R (рис. 1.9). На рисунке показано, что обратный переход исключается для всех рабочих входных последовательностей. Защищенность от неисправностей и самотестируемость обеспечивается за счет того, что всем состояниям из S_R приписываются выходные состояния из Z_R .

Задача схемы контроля при этом состоит в том, чтобы зафиксировать переход автомата из множества S в множество S_R , т. е. в конечном счете в том, чтобы отличить друг от друга основные и ошибочные состояния. Эта задача решается достаточно просто при кодировании внутренних состояний КА кодом с обнаружением ошибок. Тогда СК является декодером (тестером) и подключается к выходам ЭП.

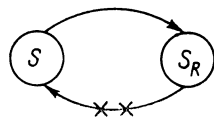


Рис. 1.9

При реализации данного подхода к синтезу ПСП-автоматов существенное значение имеет вид неисправностей. В дальней-

шем рассматриваются два вида неисправностей логических элементов и ЭП: сбой и катастрофические отказы. Сбой представляет собой кратковременную ложную фиксацию логического сигнала 0 или 1 (обозначается: $C \rightarrow 0$ и $C \rightarrow 1$) на входе или выходе некоторого элемента в течение одного такта работы КА. Катастрофический отказ есть постоянная фиксация логического сигнала (обозначается: $K \rightarrow 0$ и $K \rightarrow 1$). Класс указанных неисправностей принято называть классом константных логических неисправностей.

Неисправности бывают одиночные и кратные. Одиночная неисправность есть неисправность одного элемента (его выхода или одного из входов). Кратная неисправность состоит из нескольких одиночных. Кратная однонаправленная неисправ-

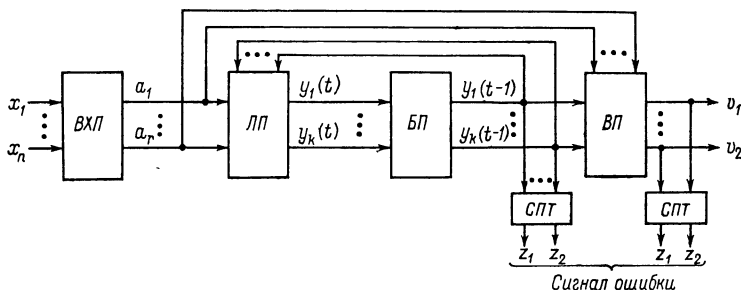


Рис. 1.10

ность состоит из нескольких одиночных неисправностей одного вида (либо только $C \rightarrow 0$, либо только $C \rightarrow 1$). Так как вероятность кратных неисправностей существенно меньше вероятности одиночных, то на практике чаще всего ограничиваются задачей обнаружения только одиночных неисправностей. В ПСП-автоматах возможно решение задачи обнаружения всех кратных однонаправленных неисправностей. Будем обозначать класс одиночных неисправностей как ОН, а класс кратных однонаправленных неисправностей — как КОН. Очевидно, что неисправности класса ОН полностью входят в класс неисправностей КОН.

Существенное значение имеет также и место возникновения неисправности. На рис. 1.10 приведена детальная структура КА, в которой по сравнению с рис. 1.1 выделены еще два блока: входной преобразователь (ВХП), формирующий наборы входных переменных x_1, \dots, x_n , и выходной преобразователь (ВП), реализующий функции выходов. Из данной структуры следует, что контроль автомата можно осуществлять как по внутреннему состоянию (в этом случае тестер подключается к выходам блока БП), так и по выходному состоянию (тестер подключается к выходам блока ВП). При этом необходимо иметь в виду, что неисправности в блоках ВХП, ЛП и БП приводят

к искажению функций переходов и выходов, а неисправности в блоке ВП — только к искажению функции выходов.

В ПСП-автоматах необходимо обеспечить обнаружение неисправностей во всех блоках структуры. Для этого требуется выполнение ряда условий [34]. Переход КА из множества S в множество S_R обеспечивается за счет кодирования внутренних состояний КА кодом с обнаружением ошибок. Наиболее часто для построения ПСП-автоматов используются равновесные коды [34, 78], коды с суммированием [54, 73], с повторением [35] и с контролем на нечетность (четность) [14]. Равновесный код относится к классу неразделимых кодов, в которых нельзя отличить друг от друга необходимые и избыточные разряды. Коды с суммированием, с повторением и с контролем на нечетность являются разделимыми кодами, в которых выделяются информационные и контрольные разряды.

Равновесный код nCm (m — вес кодовых слов) содержит C_n^m кодовых комбинаций (слов, векторов), каждая из которых

Таблица 1.13

x_1	x_2	x_3	x_4
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	1

Таблица 1.14

Информационное слово			Вспомогательное слово		Контрольное слово	
x_1	x_2	x_3	x_4^*	x_5^*	x_4'	x_5'
0	0	0	0	0	1	1
0	0	1	0	1	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	1	1	0	0	1
1	1	0	1	0	0	1
1	1	1	1	1	0	0

состоит из m разрядов, равных 1, и $n - m$ разрядов, равных 0. Например, код 4C2 содержит $C_4^2 = 6$ слов, приведенных в табл. 1.13. Код nCm обладает рядом достоинств, которые делают его удобным для кодирования состояний полностью самопроверяемых конечных автоматов. Для данных чисел n и m он дает достаточно большое число кодовых слов. Алгоритм построения кода несложен. Самым существенным является то, что код nCm обеспечивает простой принцип контроля авто-

мата — по весу кода внутреннего состояния. Контрольные схемы при этом получаются достаточно простыми.

Код с суммированием nSk [5] (k — число информационных разрядов) содержит 2^k слов. Строится он следующим образом (см. табл. 1.14, где представлен код 5S3). Образуется множество из 2^k k -разрядных информационных слов. Каждому информационному слову присваивается вспомогательное слово из $\lceil \log_2(k+1) \rceil$ разрядов. Вспомогательное слово образуется как двоичное число, выражающее количество единиц, содержащихся среди k символов соответствующего информационного слова. Каждое вспомогательное слово преобразуется в контрольное слово путем замены значения каждого символа кода на противоположное. Слова кода nSk образуются приписыванием справа к информационным словам полученных указанным способом соответствующих контрольных слов. С точки зрения построения ПСП-устройств коды nCm и nSk обладают многими общими свойствами. Коды nSk применяют в тех случаях, когда существенным является свойство делимости избыточного кода. Однако принцип контроля кода nSk более сложен, так как отсутствует единый параметр контроля и требуется сравнение информационной и контрольной частей кодовых слов.

Код с повторением nPk ($k = n/2$) имеет двойную избыточность. Контрольные слова этого кода образуются путем повторения соответствующих им информационных слов. Несмотря на большую избыточность код nPk часто используется на практике, так как с ним связаны методы повышения надежности, основанные на дублировании устройств.

Разновидностью кода с повторением является парафазный код (обозначим его $n\bar{P}k$). В нем каждый информационный разряд дополняется контрольным разрядом, имеющим противоположное по сравнению с информационным разрядом значение.

Код с проверкой на нечетность $nH1$ (четность) обладает наименьшей избыточностью. При его построении информационное слово удлиняется на один контрольный разряд, значение которого выбирается таким, чтобы число единиц в слове было нечетным (четным).

Другие известные избыточные коды (например, коды Хемминга или остаточные коды) не нашли широкого применения при построении ПСП-автоматов. Не используются они и при реализации стандартных цифровых ПСП-устройств. По этой причине дальнейшее рассмотрение ограничено только указанными выше кодами. Вопросы использования более широкого класса кодов освещены в [45].

Свойства ПСП-автомата существенным образом определяются свойствами используемого для кодирования его состояний кода. Существует прямая связь между классом обнаруживаемых в автомате неисправностей и классом ошибок, которые обнаруживаются избыточным кодом. Коды nCm , nSk и nPk

относятся к классу несравнимых кодов. В таких кодах два любых кодовых слова являются несравнимыми, т. е. содержат два разряда со значением 10 в одном слове и 01 в другом. Данное свойство обеспечивает обнаружение в указанных кодах всех одиночных и всех кратных однонаправленных ошибок. В ПСП-автоматах при этом могут быть достигнуты следующие два свойства [34]:

- 1) ФВ автомата являются монотонными ФАЛ;
- 2) в автомате обнаруживаются все неисправности класса КОН.

Код $nH1$ относится к классу сравнимых кодов. Такие коды не позволяют получать автоматы, описываемые монотонными ФАЛ. Свойство же монотонности является важным свойством ПСП-автоматов. Оно позволяет осуществлять совместную реализацию ФВ различных ЭП. При этом одиночная неисправность элемента ЛП может трансформироваться в кратную неисправность ЭП, но только в кратную однонаправленную неисправность, которая относится к классу обнаруживаемых неисправностей. Совместная реализация ФВ позволяет существенным образом уменьшить сложность автомата. Как правило, в автоматах, реализованных на основе кодов $nH1$, обеспечивается обнаружение только одиночных неисправностей. При их построении используется хорошо разработанная теория линейных схем.

В ПСП-автоматах помимо кодирования состояний кодом nRp требуется также специальное построение ЛП, при котором обеспечивается невозможность перехода автомата из множества S_R в множество S (см. рис. 1.9). Например, для КА, кодирование которого произведено несравнимым кодом, можно выделить два способа построения ЛП: 1- и 0-реализации. При 1-реализации ФВ автомата вычисляются по формуле (для синхронного автомата) [78]

$$y_f = \bigvee_{j \in R} a_j \left(\bigvee_{k \in N_{ff}} y_{i_1} y_{i_2} \dots y_{i_m} \right), \quad (1.7)$$

где R — множество индексов столбцов ТП; N_{ff} — множество индексов строк ТП, на пересечении которых со столбцом a_j в клетке расположено состояние, в коде которого $y_f = 1$; i_1, i_2, \dots, i_m — индексы внутренних переменных, которые принимают значение 1 в коде строки с индексом k .

При 0-реализации используется формула

$$y_f = \bigwedge_{j \in R} (\bar{a}_j \vee (\bigwedge_{k \in P_{ff}} (y_{i_1} \vee y_{i_2} \vee \dots \vee y_{i_{n-m}}))), \quad (1.8)$$

где P_{ff} — множество индексов строк ТП, на пересечении которых со столбцом a_j в клетке расположено состояние, в коде которого $y_f = 0$; i_1, i_2, \dots, i_{n-m} — индексы внутренних переменных, которые принимают значение 0 в коде строки с индексом k .

Свойства ПСП-автомата выполняются, если при определении ФВ часть из них вычисляется по формуле (1.7), а другая часть — по формуле (1.8) [78]. При этом строится (0, 1)-реализация КА. Для построения 1-реализации автомата (или 0-реализации) с выполнением ПСП-условий к заданной ТП предъявляются определенные требования [34]. Если они не выполняются, то ТП соответствующим образом преобразуется. Имеет место также проблема исключения в ЛП элементов пассивной избыточности. Отказы таких элементов, с одной стороны, не фиксируются в работе КА и могут накапливаться с течением времени, а с другой — могут приводить к потере автоматом ПСП-свойств. Задача исключения решается с помощью преобразования исходной ТП и ее специального кодирования.

Обнаружение отказов блока ВП (см. рис. 1.10) обеспечивается организацией контроля по выходному состоянию. В этом случае при построении автомата выполняются условия контроля по внутреннему состоянию, а выходные состояния кодируются словами кода с обнаружением ошибок.

Следует отметить, что при построении сложных дискретных устройств, имеющих большое число входов и выходов, а также сложные функции переходов и выходов, выполнение всего комплекса условий реализации ПСП-автоматов может вызывать определенные затруднения. В таких случаях более приемлемыми могут оказаться методы построения самопроверяемых устройств, развитые в [45]. При реализации же типовых цифровых схем, обладающих достаточно простыми описаниями, методы синтеза ПСП-автоматов оказываются эффективными. Это, как уже отмечалось, позволяет создать типовые интегральные ПСП-схемы, с помощью которых могут быть построены ПСП-устройства любой сложности.

ГЛАВА ВТОРАЯ

КОНТРОЛЬНЫЕ СХЕМЫ В САМОПРОВЕРЯЕМЫХ УСТРОЙСТВАХ

2.1. Свойства и характеристики самопроверяемых тестеров

Рассмотрение проблемы построения типовых цифровых ПСП-устройств начнем с изучения вопросов синтеза самопроверяемых тестеров для различных кодов с обнаружением ошибок. Тестеры в классе ПСП-устройств выступают как схемы контроля. В то же время они являются неотъемлемой частью этих устройств и поэтому оказывают существенное влияние на такие

важнейшие характеристики ДУ, как надежность, сложность и быстродействие. Большое разнообразие кодов nRp делает задачу синтеза СПТ достаточно сложной.

Самопроверяемый тестер (см. рис. 1.10) по своему основному назначению представляет собой кодовый детектор (или схему кодового разделения), задача которого состоит в том, чтобы отличать кодовые векторы, принадлежащие рассматриваемому коду, от всех остальных возможных векторов. Обычный детектор [7] имеет один выход и описывается функцией, определяемой через дизъюнкцию всех векторов кода. Для получения свойства самопроверки (или самоконтролируемости) детектор снабжается вторым выходом, как это показано на рис. 1.10. В принципе число выходов может быть и более двух. Это позволяет строить детектор в виде преобразователя кодов, на выходе которого формируются слова какого-либо произвольного кода. Тогда каждое слово заданного кода на входе детектора преобразуется в некоторое (любое) слово выходного кода, а некодовое слово на входе преобразуется в некодовое слово на выходе. В этом состоит свойство контроля входного вектора.

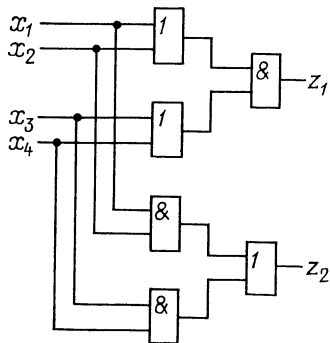


Рис. 2.1

При этом появляется возможность фиксировать неисправности внутренней структуры детектора путем установления на выходе некодового слова. Известен [45] подход к построению СПТ с одним выходом. В этом случае используется дополнительный управляющий сигнал. В дальнейшем рассматриваются только СПТ с двумя выходами. Они представляют собой преобразователи кода nRp в код $2C1$.

На рис. 2.1 показана схема СПТ для кода $4C2$. На входы x_1, x_2, x_3 и x_4 подаются четырехразрядные двоичные векторы. Шесть из них принадлежат коду $4C2$. В табл. 2.1 представлено осуществляемое схемой СПТ преобразование слов кода $4C2$ в слова кода $2C1$, которые формируются на выходах z_1 и z_2 . В табл. 2.2 представлены все четырехразрядные векторы, не принадлежащие коду $4C2$. При поступлении этих векторов на вход СПТ на его выходе формируются двухразрядные векторы 00 или 11, которые не принадлежат коду $2C1$. Таким образом, схема рис. 2.1 отличает друг от друга слова кода $4C2$ от слов, не принадлежащих этому коду. Свойство самопроверки СПТ на рис. 2.1 состоит в том, что для любой неисправности класса КОН существует такое слово кода $4C2$, при котором на выходе схемы устанавливается некодовое слово 00 или 11. Например, неисправность «обрыв входа x_1 » обнаруживается при поступлении слова 1100 путем установления на выходе значений 00.

Для сравнения тестеров между собой, а также для оценки возможности применения СПТ в структурах конкретных ДУ вводятся специальные характеристики СПТ [36]. Полное множество характеристик разобьем на четыре группы.

Таблица 2.1

Код 4C2				Код 2C1	
x_1	x_2	x_3	x_4	z_1	z_2
1	1	0	0	0	1
1	0	1	0	1	0
1	0	0	1	1	0
0	1	1	0	1	0
0	1	0	1	1	0
0	0	1	1	0	1

Таблица 2.2

x_1	x_2	x_3	x_4	z_1	z_2
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	0	1	1	1	1
1	1	1	1	1	1

1. Характеристики сложности.

1.1. Сложность L — суммарное число входов логических элементов, входящих в структуру тестера. При этом число входов элементов не ограничивается. Данная характеристика рассматривается как основная обобщенная характеристика, определяющая сложность реализации тестера независимо от выбранной элементной базы. Для СПТ на рис. 2.1 $L = 12$.

1.2. Сложность N_1 — число логических элементов, входящих в структуру тестера и не имеющих ограничений на число входов. N_1 определяет сложность реализации тестеров на микроэлектронной базе, когда имеются элементы с различным числом входов, в том числе многовходовые элементы. Для СПТ на рис. 2.1 $N_1 = 6$.

1.3. Сложность N_2 — число двухвходовых логических элементов, требующихся для реализации тестера. Данная характеристика введена по двум причинам. Во-первых, использование типовых двухвходовых элементов полностью унифицирует схемы тестеров, что позволяет сравнивать их между собой более точно, чем с помощью характеристик L и N_1 . Во-вторых, характеристика N_2 позволяет оценивать сложность программной реализации

тестеров, так как каждая команда программы осуществляет операцию только над двумя операндами. Для СПТ на рис. 2.1 $N_2 = 6$.

2. Характеристики быстродействия. Следует отметить, что так как тестеры включаются в состав ДУ, то они не должны снижать их быстродействие.

2.1. Число уровней r_1 — максимальное число элементов, через которые проходит путь в схеме тестера, связывающий его вход с выходом, при условии реализации схемы на элементах с неограниченным числом входов. Данная характеристика используется совместно с характеристиками сложности L и N_1 .

2.2. Число уровней r_2 — аналогичная характеристика для схемы тестера, реализованного на двухвходовых элементах; используется совместно с характеристикой N_2 . Для СПТ на рис. 2.1 $r_1 = r_2 = 2$.

Для программной реализации тестеров аналогом характеристики быстродействия является время выполнения программы, которое определяется характеристикой N_2 , так как все логические операции при программной реализации выполняются последовательно во времени [27].

3. Характеристики контролепригодности. Такие характеристики имеют важное значение, так как сами тестеры по своей идее являются контролепригодными схемами.

3.1. Длина проверяющего теста t — число слов кода nRp , поступление которых на вход тестера обеспечивает обнаружение в его структуре всех неисправностей из заданного класса на основе свойства самопроверки. Это — важнейшая характеристика СПТ. Проверяющий тест СПТ, входящего в структуру ПСП-автомата, составляется из внутренних (или выходных) состоя-

Таблица 2 3

x_1	x_2	x_3	x_4	z_1	z_2
1	1	0	0	0	1
1	0	0	1	1	0
0	1	1	0	1	0
0	0	1	1	0	1

ний автомата и формируется в процессе его функционирования. По этой причине уменьшение числа векторов, необходимых для полной проверки тестера, позволяет решить задачу увеличения вероятности обнаружения в нем неисправностей за счет более частого приложения к схеме всего проверяющего теста. Кроме того, это же дает возможность контроля кодовых систем, содержащих неполное множество слов кода nRp . Например, тестер

для кода 4C2 (рис. 2.1) требует для своей проверки только 4 из 6 возможных кодовых слов (проверяющий тест представлен в табл. 2.3). Данное обстоятельство позволяет использовать код 4C2 при синтезе ПСП-автоматов с числом состояний $p=4, 5, 6$. При числе состояний $p \leq 3$ код 4C2 применить нельзя, так как в этом случае не будет обеспечена полная проверка тестера, включенного в структуру автомата.

3.2. Коэффициент симметрии $k = b_1/b_2$, где $b_1(b_2)$ — число слов кода nRp , на которых равна 1 функция $z_1(z_2)$. Смысл введения данной характеристики следующий. Каждый элемент схе-

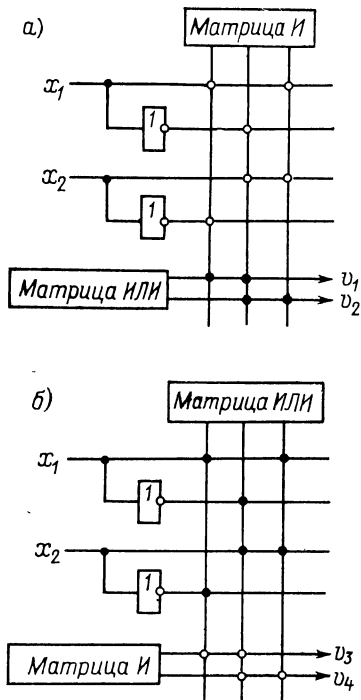


Рис. 2.2

мы тестера проверяется на одном или нескольких словах кода nRp . Разные слова поступают на вход тестера с различной частотой. Поэтому необходимым требованием для обеспечения свойства самопроверки является поступление на вход тестера всех слов проверяющего теста в промежутке времени между моментами появления двух таких неисправностей, которые совместно не обнаруживаются на основе свойства самопроверки. Если принять естественное предположение о равновероятности слов кода nRp , то равенство $k = 1$ будет наилучшим условием для выполнения указанного требования.

4. Характеристики реализуемости тестера на программируемых логических матрицах (ПЛМ). Матрицы находят широкое применение при построении ДУ с помощью больших и сверхбольших интегральных схем. В рамках ПСП-схемотехники ПЛМ целесообразно использовать как раз для реализации СПТ, учитывая их

большое многообразие. ПЛМ представляет собой устройство стандартного вида [21], состоящее из двух матриц типа И и ИЛИ (рис. 2.2). ФАЛ реализуется на ПЛМ по своим дизъюнктивным нормальным формам (ДНФ). Матрица типа И на своих вертикальных шинах реализует конъюнкции ДНФ, а матрица типа ИЛИ на горизонтальных шинах — операции общей дизъюнкции. ПЛМ на рис. 2.2, а реализует систему функций $v_1 = x_1x_2 \vee \bar{x}_1\bar{x}_2$, $v_2 = \bar{x}_1x_2 \vee x_1x_2$. Для описания ПЛМ удобно указывать число входных (a_1) и выходных (a_3) шин, число промежуточных (вертикальных на рис. 2.2, а) шин (a_2)

и обозначать ПЛМ как $M(a_1, a_2, a_3)$. Так для ПЛМ на рис. 2.2, а $a_1 = 4, a_2 = 3, a_3 = 2$ и поэтому ее обозначение имеет вид $M(4, 3, 2)$.

Вторая модификация ПЛМ реализует ФАЛ по конъюнктивным нормальным формам (КНФ). В этом случае матрицы И и ИЛИ в структуре на рис. 2.2, а меняются местами. На рис. 2.2, б представлена ПЛМ, реализующая систему функций $v_3 = (x_1 \vee x_2)(\bar{x}_1 \vee x_2), v_4 = (\bar{x}_1 \vee x_2)(x_1 \vee x_2)$.

Введем три характеристики тестеров, реализованных на ПЛМ.

4.1. Число ПЛМ h , необходимое для построения тестера. Оно определяется структурой формул, описывающих схему СПТ. Так тестер на рис. 2.1 описывается двумя функциями: $z_1 =$

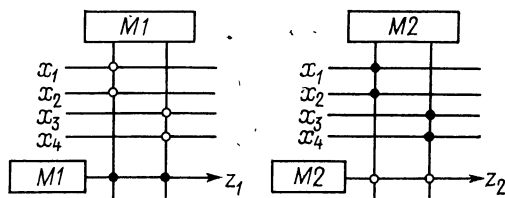


Рис. 2.3

$= (x_1 \vee x_2) \wedge (x_3 \vee x_4)$ и $z_2 = x_1 x_2 \vee x_3 x_4$. Первая функция представлена в КНФ, а вторая — в ДНФ. Поэтому для реализации тестера требуются две матрицы различных модификаций (рис. 2.3): $M1(4, 2, 1)$ и $M2(4, 2, 1)$. Очевидно, что структуры тестеров, требующие для своей реализации большого числа ПЛМ, неэффективны. В дальнейшем рассматриваются только такие модификации тестеров, которые могут быть построены не более чем на четырех ПЛМ.

4.2. Площадь S — суммарная площадь всех ПЛМ, входящих в структуру тестера, которая вычисляется по формуле

$$S = \sum_{i \in W} S_i = \sum_{i \in W} a_2^i (a_1^i + a_3^i), \quad (2.1)$$

где W — множество индексов ПЛМ; a_j^i ($j \in \{1, 2, 3\}$) — характеристика ПЛМ с индексом i .

Необходимость введения данной характеристики определяется тем, что при практической реализации тестеров всегда существуют ограничения на размеры ПЛМ, имеющиеся в распоряжении разработчика. С этим же связана целесообразность указания параметров a_1, a_2 и a_3 всех ПЛМ, входящих в структуру тестера.

4.3. Число ПЛМ r_3 , включенных последовательно в схему тестера (характеризует быстродействие). Введение данной характеристики обусловлено тем, что схемы с одним и тем же

числом уровней r_1 могут потребовать различное число последовательно включенных ПЛМ.

Для схемы СПТ на рис. 2.3 имеем $S = S_1 + S_2 = 10 + 10 = 20$, $r_3 = 1$.

Указанные десять характеристик позволяют определять направления, в которых следует разрабатывать структуры тестеров. При этом надо учитывать, что для ряда характеристик можно указать их минимальные значения, которые определяются видом кода nRp и его конкретными параметрами. Для других характеристик неизвестны минимальные значения. Поэтому следует вести поиск СПТ с наименьшими значениями

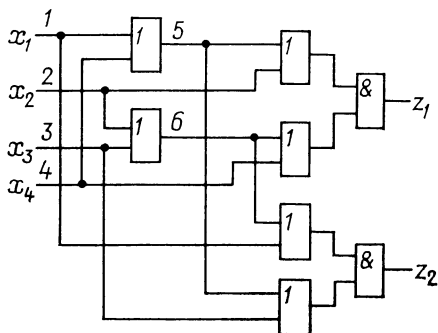


Рис. 2.4

этих характеристик. Наконец, возникает задача поиска структур тестеров с наилучшим сочетанием ряда характеристик (например, с наименьшей сложностью при заданном быстродействии или с наименьшей площадью S при заданном числе ПЛМ h и др.). Конкретные требования к СПТ вытекают из свойств ДУ, в состав которого включается тестер.

Таким образом, можно говорить о многообразии задач, возникающих при разработке структур СПТ. Целесообразно также создавать специальные каталоги для конкретных кодов, в которые включать только те тестеры, которые обладают наилучшими характеристиками. Вопросы разработки каталогов будут рассмотрены в дальнейшем.

Для описания схем СПТ будем использовать системы функций алгебры логики (ФАЛ), применяя при этом принцип суперпозиции функций. Входные переменные тестера обозначаются цифрами от 1 до n , соответствующими индексам этих переменных, а выходные переменные — буквами z_1 и z_2 . Промежуточные функции, реализуемые на внутренних узлах схемы, имеющих разветвления, также обозначаются цифрами, но от $n+1$ и далее. Логическая операция дизъюнкции обозначается при этом знаком «+», операция конъюнкции — знаком «X», а операция отрицания — чертой над соответствующей цифрой. Если с помощью конъюнкции связываются два скобочных выражения, то знак «X» опускается. Промежуточные функции выделяются так, чтобы получаемая при этом система функций однозначно отражала структуру схемы. Рассмотрим, например, СПТ для кода 4С1 (рис. 2.4). Для описания схемы необходимо ввести две промежуточные функции, реализуемые на выходах элементов ИЛИ первого уровня схемы (обозначим их цифрами

5 и 6). Тогда тестер описывается следующей системой функций: $5 = 1 + 4$, $6 = 2 + 3$, $z_1 = (2 + 5) \times (4 + 6)$, $z_2 = (1 + 6) \times (3 + 5)$.

2.2. Тестеры для равновесных кодов

Для кодов nCm тестеры представляют собой преобразователи $nCm \rightarrow 2C1$. Будем обозначать их как m/n -тестеры, или m/n -СПТ. С точки зрения особенностей построения m/n -тестеров все коды nCm целесообразно разделить на два класса: $nC1m$ и $nC2m$. Для кодов класса $nC1m$ выполняется соотношение $n \geq 2m$, а для кодов класса $nC2m$ — соотношение $n < 2m$. В дальнейшем будут рассматриваться в основном коды класса $nC1m$ (они будут обозначаться просто как nCm). Случаи рассмотрения кодов класса $nC2m$ будут специально оговариваться. Каждый конкретный код из класса $nC1m$ имеет обратный код в классе $nC2m$. А именно, коду nCm обратным является код $(n-m)Cm$, который может быть получен из кода nCm заменой значений разрядов кодовых слов на противоположные. По этой причине $(m-n)/n$ -СПТ можно получить из m/n -СПТ установкой инверторов на всех входах последнего. Кроме того, в ряде случаев $(m-n)/n$ -СПТ образуется из m/n -СПТ заменой в нем каждого элемента И на элемент ИЛИ и наоборот (обратные схемы тестеров). Такие случаи будут указываться специально.

Самопроверяемый m/n -тестер описывается функциями z_1 и z_2 . Для их определения уточним свойство контроля входного вектора. Будем считать, что в данном случае оно состоит в следующем: на векторах кода nCm выходы z_1 , z_2 принимают значения 01 или 10, на векторах с весом $\omega < m$ — значения 00, а на векторах с весом $\omega > m$ — значения 11. Такое уточнение вытекает из свойств кодов nCm . Оно позволяет более просто решать задачу синтеза m/n -СПТ.

Введем в рассмотрение базовую функцию m/n -СПТ вида

$$f^m(x_1, x_2, \dots, x_n) = f^m(x_1 \div x_n) = \bigvee_{i_1, i_2, \dots, i_m \in \{1, 2, \dots, n\}} x_{i_1} x_{i_2} \dots x_{i_m}, \quad (2.2)$$

которая образуется путем объединения знаком дизъюнкции конъюнкций ранга m , соответствующих всем словам (их число равно C_n^m) кода nCm . Между конъюнкциями $x_{i_1} x_{i_2} \dots x_{i_m}$ и словами кода nCm существует взаимно однозначное соответствие. Конъюнкция содержит в себе те переменные, которым в соответствующем слове кода отвечают разряды, равные 1. По этой причине в дальнейшем для обозначения слов кода мы будем пользоваться также и понятием конъюнкций. Например, для кода $4C2$ имеем

$$\begin{aligned} f^2(x_1, x_2, x_3, x_4) = & x_1 x_2 \vee x_1 x_3 \vee x_1 x_4 \vee \\ & \vee x_2 x_3 \vee x_2 x_4 \vee x_3 x_4. \end{aligned} \quad (2.3)$$

Функция (2.3) содержит шесть конъюнкций ранга 2, соответствующих шести словам кода 4C2, представленным в табл. 1.13. Слову 1100 соответствует конъюнкция x_1x_2 , слову 1010 — конъюнкция x_1x_3 и т. д.

Методы вычисления функций z_1 и z_2 предусматривают использование базовой функции $f^m(x_1 \div x_n)$. При этом между ними существует следующее соотношение. Каждая конъюнкция базовой функции входит в ДНФ одной из функций z_1 или z_2 . Однако последние могут содержать конъюнкции, не входящие в базовую функцию.

В настоящее время разработаны четыре подхода к построению m/n -СПТ. Основная идея первого подхода состоит в разложении базовой функции (2.2) на две функции z_1 и z_2 , каждая

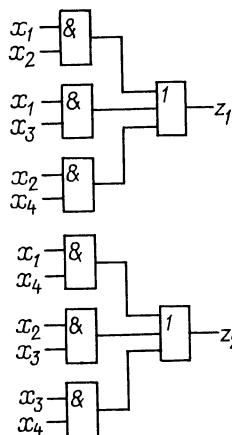


Рис. 2.5

из которых также представляет собой объединение конъюнкций ранга m . Другими словами, конъюнкции функции $f^m(x_1 \div x_n)$ определенным образом распределяются между функциями z_1 и z_2 . Например, из выражения (2.3) можно получить следующие функции:

$$z_1 = x_1x_2 \vee x_1x_3 \vee x_3x_4, \quad (2.4)$$

$$z_2 = x_1x_4 \vee x_2x_3 \vee x_2x_4.$$

В данном случае система (2.4) описывает 2/4-СПТ, показанный на рис. 2.5. Но, очевидно, что функции z_1 и z_2 выбираются не произвольно, а должны быть выбраны так, чтобы реализованная по ним схема тестера обладала свойствами контроля входного вектора и самопроверки.

Рассмотрим свойство контроля входного вектора. Можно указать необходимые и достаточные условия, которым должно удовлетворять построение функций z_1 и z_2 .

Обозначим через p_i слово кода nCm , а через M_i — соответствующую ему конъюнкцию. Через W_m , W_{m-1} и W_{m+1} обозначим множество всех слов или конъюнкций кодов соответственно nCm ; $nC(m-1)$ и $nC(m+1)$. Будем также говорить, что конъюнкция M_1 ранга m покрывает конъюнкцию M_2 ранга $m+1$, если M_1 составляет часть M_2 .

Сформулируем указанные условия:

Н1. Функции z_1 и z_2 должны представлять собой дизъюнкцию конъюнкций ранга m , входящих в базовую функцию тестера (множество W_m).

Н2. Каждая конъюнкция ранга m из множества W_m должна входить в одну из функций z_1 и z_2 , и причем только в одну.

Н3. Каждая конъюнкция ранга $m+1$, входящая в множество

W_{m+1} , должна покрываться хотя бы одной конъюнкцией, включенной в функцию z_1 , и хотя бы одной конъюнкцией, включенной в функцию z_2 .

Докажем необходимость данных условий. Необходимость первого условия очевидна. Рассмотрим условие Н2. Возможны два случая, когда оно не выполняется. В первом случае имеется некоторая конъюнкция M_i , которая не включена в функции z_1 и z_2 . Тогда очевидно, что при поступлении на вход тестера слова p_i , соответствующего конъюнкции M_i , устанавливается равенство $z_1 = z_2 = 0$, что приводит к нарушению свойства контроля входного вектора. Во втором случае конъюнкция M_i включена в обе функции z_1 и z_2 . Тогда при поступлении на вход тестера слова p_i устанавливается равенство $z_1 = z_2 = 1$, что также нарушает свойство контроля входного вектора. Рассмотрим условие Н3. Оно не выполняется, если имеется такая конъюнкция кода $nC(m+1)$, которая покрывается конъюнкциями ранга m , входящими, например, в функцию z_1 , и не покрывается ни одной конъюнкцией, входящей в функцию z_2 . При этом подача на вход m/n -тестера некодового слова с весом $m+1$, соответствующего указанной конъюнкции, приводит к установлению равенств $z_1 = 1$ и $z_2 = 0$, что нарушает свойство контроля вектора.

Докажем достаточность условий Н1—Н3. Для этого надо рассмотреть случаи, когда на вход тестера поступают слова с весами $w = m$, $w < m$ и $w > m$. Условия Н1 и Н2 обеспечивают выполнение свойства контроля входного вектора при поступлении слова с $w = m$, так как при этом одна из функций z_1 , z_2 и только одна из них принимает значение 1. Если вес входного вектора $w < m$, то в силу выполнения условия Н1 устанавливается равенство $z_1 = z_2 = 0$, что отвечает свойству контроля входного вектора. Рассмотрим случай, когда вес входного слова $w > m$. Пусть $w = m+1$. Тогда ввиду выполнения условия Н3 устанавливается равенство $z_1 = z_2 = 1$ и свойство контроля входного вектора также выполняется. Если вес входного слова p_i $w > m+1$, то можно указать несколько слов с весом $m+1$, объединение которых дает слово p_i . А так как для каждого из этих слов свойство контроля входного вектора выполняется, то оно выполняется и для слова p_i .

Система (2.4) отвечает условиям Н1—Н3. Выполнение условия Н3 проиллюстрируем с помощью специальной таблицы покрытий (табл. 2.4). Такая таблица для кода nCm содержит C_n^m строк, соответствующих конъюнкциям ранга m , и C_n^{m+1} столбцов, соответствующих конъюнкциям ранга $m+1$. На пересечении строки с конъюнкцией M_i и столбца с конъюнкцией M_j ставится знак покрытия (*), если конъюнкция M_i покрывает M_j . Из табл. 2.4 видно, что при поступлении на вход тестера (см. рис. 2.5) любого слова кода $4C3$ на его выходах устанавливаются значения $z_1 = z_2 = 1$. При подаче на вход тестера слова с весом $w < 2$ устанавливаются значения $z_1 = z_2 = 0$, так

как элементы первого уровня схемы реализуют конъюнкции ранга 2.

Перейдем к рассмотрению свойства самопроверки. Его реализация представляет собой более сложную задачу по сравнению с задачей обеспечения свойства контроля входного вектора. Это связано с тем, что если свойство контроля входного вектора определяется только составом функций z_1 и z_2 (в соответствии с условиями Н1 и Н2), то свойство самопроверки, кроме этого, определяется также и структурой схемы, реализующей функции z_1 и z_2 . Очевидно, что по функциям z_1 и z_2 за счет их эквива-

Таблица 2.4

Функция	Код 4С2	Код 4С3			
		$x_1x_2x_3$	$x_1x_2x_4$	$x_1x_3x_4$	$x_2x_3x_4$
z_1	x_1x_2	*	*		
	x_1x_3	*		*	
	x_3x_4			*	*
z_2	x_1x_4		*	*	
	x_2x_3	*			*
	x_2x_4		*		*

лентных преобразований могут быть получены различные схемы тестера. При этом эквивалентные преобразования не влияют на реализацию свойства контроля входного вектора, но оказывают существенное влияние на реализацию свойства самопроверки, а также на длину проверяющего теста.

В связи с этим можно указать только достаточное условие, которому должны удовлетворять функции z_1 и z_2 для выполнения свойства самопроверки. Оно дополняет условия Н1—Н3 и формулируется следующим образом.

Н4. Каждая конъюнкция ранга $m-1$, входящая в множество W_{m-1} , должна покрывать хотя бы одну конъюнкцию, включенную в функцию z_1 , и хотя бы одну конъюнкцию, включенную в функцию z_2 .

Выполнение условия Н4 для системы (2.4) проиллюстрировано табл. 2.5, которая является таблицей покрытий для кода 4С1.

Докажем достаточность условия Н4 для выполнения свойства самопроверки (учитывая, что выполняются также условия Н1 и Н2). Рассматриваем неисправности класса КОИ. Функции z_1 и z_2 по построению (см. формулу (2.2)) являются монотонными ФАЛ, представленными в виде минимальных ДНФ. Они не содержат переменных со знаком инверсии и поэтому при-

надлежат к узкому классу простых симметричных функций [29]. Реализации таких функций представляют собой двухуровневые схемы вида И — ИЛИ (см. рис. 2.5), которые не содержат инверторов. Для полной проверки схем данного вида достаточно проверить только неисправности входов элементов И первого уровня. Неисправности элементов можно интерпретировать как фиксацию в 0 или 1 отдельных переменных или целых конъюнкций в функциях z_1 и z_2 . Например, если в схеме на рис. 2.5 происходит неисправность вида $K \rightarrow 1$ (или $C \rightarrow 1$, что в данном случае, когда рассматриваются комбинационные схемы, несус-

Таблица 2.5

Функция	z_1			z_2		
	Код 4C2					
Код 4C1	x_1x_2	x_1x_3	x_3x_4	x_1x_4	x_2x_3	x_2x_4
x_1	*	*		*		
x_2	*				*	*
x_3		*	*		*	
x_4			*	*		*

щественно) входа x_1 элемента И, реализующего конъюнкцию x_1x_2 , это соответствует фиксации в единицу переменной x_1 (будет обозначать $x_i \equiv 1$) в конъюнкции x_1x_2 функции z_1 системы (2.4). В этом случае схема вместо функции z_1 реализует ошибочную функцию $z_1^* = x_2 \vee x_1x_3 \vee x_3x_4$. Если происходит неисправность вида $K \rightarrow 0$ указанного входа, то схема реализует ошибочную функцию $z_1^* = x_1x_3 \vee x_3x_4$.

Рассмотрим одиночные неисправности. Пусть в функцию z_p ($p \in \{1, 2\}$) входит конъюнкция $M_j = x_{i_1} \dots x_{i_{t-1}} x_{i_t} x_{i_{t+1}} \dots x_{i_m}$. Проанализируем два противоположных случая фиксации ее переменной x_{i_t} . Если $x_{i_t} \equiv 0$, то фиксируется в 0 и вся конъюнкция M_j . В результате вместо функции z_p реализуется функция z_p^* , не содержащая M_j . Очевидно, что в данном случае неисправность фиксируется при поступлении на вход тестера слова ρ , соответствующего конъюнкции M_j , так как при этом устанавливается равенство $z_1 = z_2 = 0$. Если $x_{i_t} \equiv 1$, то на выходе z_p вместо конъюнкции M_j реализуется конъюнкция $M_j^* = x_{i_1} \dots x_{i_{t-1}} x_{i_{t+1}} \dots x_{i_m}$ ранга $m-1$. В соответствии с условием Н4 функция z_q ($q \in \{1, 2\}, q \neq p$) содержит конъюнкцию M_s , которая покрывается конъюнкцией M_j^* ($M_s = x_{i_1} \dots x_{i_{t-1}} x_{i_r} x_{i_{t+1}} \dots x_{i_m}, x_{i_r} \neq x_{i_t}$). По этой причине неисправность фиксируется

на слове ρ , соответствующем конъюнкции M_s , так как при этом $z_1 = z_2 = 1$.

Рассмотрим кратные однонаправленные неисправности. Они соответствуют одновременной фиксации одного вида ($K \rightarrow 1$ или $K \rightarrow 0$) некоторых переменных в различных конъюнкциях. Если такая фиксация имеет вид $K \rightarrow 0$, то она приводит к обращению в 0 нескольких конъюнкций в функциях z_1 и z_2 . В данном случае неисправность обнаруживается на слове, соответствующем любой из этих конъюнкций. Если фиксация имеет вид $K \rightarrow 1$, то она приводит к замене в функциях z_1 и z_2 ряда конъюнкций ранга m на конъюнкции ранга $m-t$ ($m > t \geq 1$). Согласно условию Н4 каждая из образовавшихся конъюнкций M_j^* ранга $m-t$ в функции $z_1(z_2)$ покрывает хотя бы одну конъюнкцию M_s ранга m функции $z_2(z_1)$, что позволяет фиксировать неисправность на слове ρ_s , соответствующем конъюнкции M_s . Если же в результате кратности неисправности конъюнкция M_s также преобразуется в конъюнкцию ранга $m-t$, то условие однонаправленности неисправностей обеспечивает их обнаружение на слове ρ_s .

Проведенное доказательство относится к тестеру, реализованному в виде двухуровневой схемы вида И — ИЛИ без инверторов. Однако известно [12], что проверяющий тест такой схемы является тестом и для любой образованной из нее путем эквивалентных преобразований схемы. Поэтому условия Н1 — Н4 являются достаточными условиями построения произвольных структур m/n -СПТ, полученных на основе эквивалентных преобразований функций z_1 и z_2 .

Таким образом, описанный метод построения m/n -СПТ состоит в следующем. Строятся таблицы покрытий для кодов nCm и $nC(m-1)$, по которым путем перебора вариантов определяются функции z_1 и z_2 , отвечающие условиям Н1 — Н4. Структура тестера строится либо непосредственно по функциям z_1 и z_2 , либо по полученным из них эквивалентным формулам. Однако описанный подход не является универсальным, так как установлено [77], что для ряда кодов условие Н4 не может быть выполнено.

Кроме того, метод перебора вариантов не позволяет решать задачу целенаправленного улучшения характеристик тестеров. При решении этих задач и построении СПТ для указанных кодов используются другие подходы к реализации тестеров.

Второй подход к построению m/n -СПТ отличается от первого тем, что в функции z_1 и z_2 помимо конъюнкций ранга m включаются также определенные конъюнкции большего ранга. Будем называть такие конъюнкции избыточными. Они обеспечивают в необходимых случаях выполнение свойства контроля входного вектора, но требуют для проверки элементов тестера специальной его структуры. Выбор избыточных конъюнкций обосновывается в каждом конкретном случае построения m/n -СПТ.

Универсальный характер имеет третий подход к построению тестеров, основанный на многокаскадном преобразовании кодов (в отличие от первого подхода, при котором осуществляется непосредственное преобразование кода n_1Cm_1 в код $2C1$), которое осуществляется по схеме $n_1Cm_1 \rightarrow n_2Cm_2 \rightarrow n_3Cm_3 \rightarrow \dots \rightarrow n_kCm_k \rightarrow 2C1$. На рис. 2.6 представлена обобщенная структура тестера, основанного на преобразовании кодов. Число каскадов в преобразовании кодов в общем случае не ограничено. Определим требования, которым должны удовлетворять преобразователи, входящие

в СПТ. В первую очередь, каждый преобразователь должен обладать свойствами контроля входного вектора и самопроверки. Данные свойства аналогичны соответствующим свойствам СПТ. А именно, на выходе преобразователя $n_iCm_i \rightarrow n_jCm_j$ формируется слово кода n_jCm_j , если на его входе присутствует вектор кода n_iCm_i ; в противном случае формируется вектор, не принадлежащий коду n_jCm_j . Для любой неисправности преобразователя класса КОН существует слово кода n_iCm_i , при котором на выходе устанавливается вектор, не принадлежащий коду n_jCm_j .

Если сформулированное требование выполняется, то структура рис. 2.6 обладает свойством трансляции некодового вектора от входа к выходу. В самом деле, при поступлении на входы x_1, x_2, \dots, x_{n_1} некодового слова (не принадлежащего коду n_1Cm_1) на выходе первого преобразователя $n_1Cm_1 \rightarrow n_2Cm_2$ формируется также некодовое слово (не принадлежащее коду n_2Cm_2). Поэтому некодовое слово появляется на входе второго преобразователя $n_2Cm_2 \rightarrow n_3Cm_3$, что приводит к установлению на его выходе также некодового слова, и т. д.

В результате на выходах структуры z_1, z_2 формируется некодовый вектор 00 или 11. Если в каком-либо преобразователе возникает неисправность, то она фиксируется путем установления некодового слова на выходе этого преобразователя, которое затем транслируется на выходы z_1, z_2 . Для того чтобы выполнить такую фиксацию, необходимо обеспечить поступление на вход преобразователя проверяющего теста.

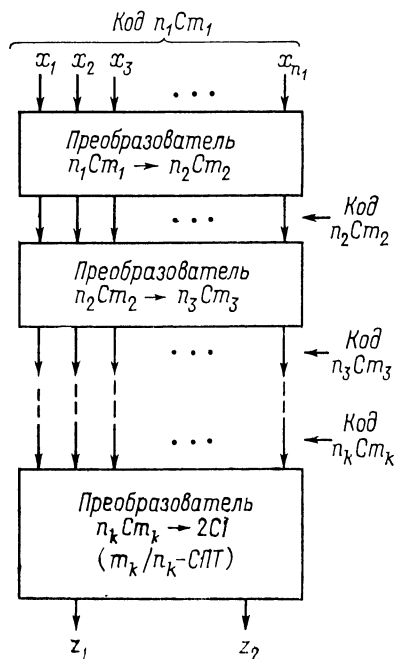


Рис. 2.6

Определим понятие проверяющего теста для некоторого блока A , состоящего из функциональных элементов и реализующего некоторую логическую вектор-функцию $\tilde{\varphi}(\tilde{x})$. Пусть $G(A)$ — совокупность всех неисправностей блока A из рассматриваемого класса, а $\tilde{\varphi}_g(\tilde{x})$ — вектор-функция, реализуемая блоком A с неисправностью $g \in G(A)$. Тогда под проверяющим тестом блока A понимается совокупность входных векторов $T(A)$, такая, что для любой неисправности $g \in G(A)$ существует такой входной вектор $x \in T(A)$, для которого $\tilde{\varphi}_g(\tilde{x}) \neq \tilde{\varphi}(\tilde{x})$.

Обозначим через A и B блоки из функциональных элементов, реализующие функции $\tilde{\varphi}(\tilde{x})$ и $\tilde{\psi}(\tilde{x})$ соответственно, а через AB — схему, полученную последовательным соединением блоков A и B и реализующую функцию $\tilde{f}(\tilde{x}) = \tilde{\psi}(\tilde{\varphi}(\tilde{x}))$. Для того чтобы совокупность входных векторов $T(AB)$ составляла проверяющий тест схемы AB , необходимо выполнение следующих трех условий согласования блоков A и B :

1. $T(A) \subseteq T(AB)$.
2. При подаче на вход схемы AB проверяющего теста $T(AB)$ на выходе блока A должно формироваться множество векторов, включающее в себя тест $T(B)$, т. е. $T(B) \subset \{\tilde{\varphi}(\tilde{x}), \tilde{x} \in T(AB)\}$.

3. Для каждой неисправности $g \in G(A)$ должен существовать вектор $\tilde{x} \in T(A)$, такой, что $\tilde{x} \in T(AB)$ и $\tilde{\psi}(\tilde{\varphi}_g(\tilde{x})) \neq \tilde{f}(\tilde{x})$.

С точки зрения построения минимального теста целесообразно строить такие тесты $T(AB)$, для которых $t(AB) = l(A)$ или $t(AB) = t(B)$. Указанные условия очевидным образом обобщаются и на случай последовательного соединения нескольких блоков.

Для структуры на рис. 2.6 третье условие выполняется автоматически ввиду наличия свойства трансляции некодового вектора. Первые два условия должны обеспечиваться при реализации СПТ специальным образом.

Четвертый подход к построению m/n -СПТ заключается в реализации их структур из типовых блоков. Будем называть набор блоков функционально полным (ФПН), если с помощью соединения входящих в него блоков между собой может быть реализован любой тестер из рассматриваемого класса m/n -СПТ. ФПН целесообразно строить для класса всех m/n -СПТ, а также для классов СПТ с фиксированными значениями m или n . В ФПН включаются m/n -СПТ с небольшими значениями m и n , а также некоторые логические блоки, используемые для объединения отдельных m/n -СПТ в общую структуру.

Далее рассматриваются различные методы синтеза m/n -СПТ, относящиеся к указанным подходам и позволяющие получать тестеры с определенными характеристиками. С точки

зрения реализуемости m/n -СПТ все коды nCm разобьем на три класса: $2mCm$, $nC1$ и nCm ($n/2 \geq m > 1$), которые будут рассмотрены по отдельности.

2.3. Синтез тестеров для кодов $2mCm$

Коды $2mCm$ в связи со свойствами симметрии позволяют получать эффективные схемы СПТ. Первый метод реализации $m/2m$ -СПТ был описан в [48], где предложено для построения тестеров использовать пороговые цепи, проектируемые в базисе типовых логических элементов. Будем обозначать через $F_n(k \geq m)$ пороговую функцию n переменных, принимающую значение 1 на наборах входных переменных, имеющих вес m и более. Через B обозначим множество входных переменных пороговой сети.

Двухуровневая пороговая цепь на элементах И и ИЛИ строится по следующим правилам. Устанавливается C_n^m элементов И с m входами. Каждый элемент И реализует конъюнкцию, соответствующую одной из комбинаций m переменных. Выходы всех элементов И подаются на входы элемента ИЛИ. Выход последнего является выходом пороговой цепи. Очевидно, что базовая функция m/n -СПТ (2.2) является пороговой функцией, а выражение (2.3) описывает двухуровневую пороговую цепь, для которой $B = \{x_1, x_2, x_3, x_4\}$ и $m = 2$.

Многоуровневая пороговая цепь строится следующим образом. Множество входных переменных B разбивается на два непустых подмножества B_1 и B_2 , содержащих соответственно n_1 и n_2 переменных ($n_1 + n_2 = n$). Функция, описывающая пороговую цепь, определяется по формуле

$$F_n(k \geq m) = \bigvee_{j \in \{r_1, \dots, r_2\}} F_{n_1}(k_1 \geq j) F_{n_2}(k_2 \geq m - j), \quad (2.5)$$

где $r_1 = \max(m - n_2; 0)$, $r_2 = \min(n_1; m)$.

Например, для случая, когда $n = 4$ и $m = 2$, образуем подмножества $B_1 = \{x_1, x_2\}$ и $B_2 = \{x_3, x_4\}$. Тогда $n_1 = n_2 = 2$, $r_1 = \max(0; 0) = 0$ и $r_2 = \min(2; 2) = 2$. В соответствии с (2.5) пороговая цепь описывается функцией

$$\begin{aligned} F_4(k \geq 2) &= \bigvee_{j \in \{0, 1, 2\}} F_2(k_1 \geq j) F_2(k_2 \geq 2 - j) = \\ &= F_2(k_1 \geq 0) F_2(k_2 \geq 2) \vee F_2(k_1 \geq 1) F_2(k_2 \geq 1) \vee \\ &\quad \vee F_2(k_1 \geq 2) F_2(k_2 \geq 0) = \\ &= x_3 x_4 \vee (x_1 \vee x_2) (x_3 \vee x_4) \vee x_1 x_2. \end{aligned} \quad (2.6)$$

Данная цепь реализуется трехуровневой схемой. Для получения пороговых цепей с большим числом уровней (с целью построения более экономичных по расходу элементов схем)

формула (2.5) должна применяться последовательно несколько раз. Проиллюстрируем такой процесс на примере вычисления функции $F_6(k \geq 4)$. Выделим множества $B_1 = \{x_1 \div x_4\}$ и $B_2 = \{x_5, x_6\}$. В этом случае $m = n_1 = 4$, $n_2 = 2$, $r_1 = \max(2; 0) = 2$, $r_2 = \min(4; 4) = 4$. Тогда имеем

$$\begin{aligned} F_6(k \geq 4) &= \bigvee_{j \in \{2, 3, 4\}} F_4(k_1 \geq j) F_2(k_2 \geq 4 - j) = \\ &= F_4(k_1 \geq 2) F_2(k_2 \geq 2) \vee F_4(k_1 \geq 3) F_2(k_2 \geq 1) \vee \\ &\vee F_4(k_1 \geq 4) F_2(k_2 \geq 0). \end{aligned}$$

В полученном выражении для представления функций $F_4(k_1 \geq 2)$ и $F_4(k_1 \geq 3)$ снова применим формулу (2.5). Функция $F_4(k_1 \geq 2)$ представляется в виде выражения (2.6). Для вычисления функции $F_4(k \geq 3)$ множество B_1 разобьем на два подмножества $B_{11} = \{x_1, x_2\}$ и $B_{12} = \{x_3, x_4\}$. При этом $m = 3$, $n_1 = n_2 = 2$, $r_1 = \max(1; 0) = 1$, $r_2 = \min(2; 3) = 2$. Следовательно,

$$\begin{aligned} F_4(k_1 \geq 3) &= \bigvee_{j \in \{1, 2\}} F_2(k_1 \geq j) F_2(k_2 \geq 3 - j) = \\ &= F_2(k_1 \geq 1) F_2(k_2 \geq 2) \vee F_2(k_1 \geq 2) F_2(k_2 \geq 1) = \\ &= (y_1 \vee y_2) y_3 y_4 \vee y_1 y_2 (y_3 \vee y_4). \end{aligned}$$

Подставляя полученные выражения в исходное, получаем

$$\begin{aligned} F_6(k \geq 4) &= (x_3 x_4 \vee (x_1 \vee x_2) (x_3 \vee x_4) \vee x_1 x_2) x_5 x_6 \vee \\ &\vee ((x_1 \vee x_2) x_3 x_4 \vee x_1 x_2 (x_3 \vee x_4)) (x_5 \vee x_6) \vee x_1 x_2 x_3 x_4. \end{aligned}$$

Метод синтеза СПТ, обладающих свойством контроля входного вектора, состоит в следующем. Множество входных переменных тестера разбивается на два непересекающихся подмножества B_1 и B_2 , содержащих соответственно n_1 и n_2 переменных. Функции z_1 и z_2 находятся по следующим формулам:

$$z_1 = \bigvee_{j \in \{r_1, \dots, r_2\}} F_{n_1}(k_1 \geq j) F_{n_2}(k_2 \geq m - j); \quad (2.7)$$

$$z_2 = \bigvee_{i \in \{r_1, \dots, r_2\}} F_{n_1}(k_1 \geq i) F_{n_2}(k_2 \geq m - i), \quad (2.8)$$

где переменная j принимает значения, соответствующие нечетным числам, а переменная i — четным числам; $r_1 = \max(m - n_2; -1)$; $r_2 = \min(n_1; m + 1)$.

Определим функции, описывающие 2/5-СПТ. Множество $B = \{x_1 \div x_5\}$ разделим на два подмножества: $B_1 = \{x_1, x_2, x_3\}$ и $B_2 = \{x_4, x_5\}$. В этом случае имеем: $m = 2$, $n_1 = 3$, $n_2 = 2$, $r_1 = \max(0; -1) = 0$, $r_2 = \min(3; 3) = 3$. Используя формулы (2.7) и (2.8), получаем

$$\begin{aligned} z_1 &= \bigvee_{j \in \{1, 3\}} F_3(k_1 \geq j) F_2(k_2 \geq 2 - j) = F_3(k_1 \geq 1) F_2(k_2 \geq 1) \vee \\ &\vee F_3(k_1 \geq 3) F_2(k_2 \geq -1) = (x_1 \vee x_2 \vee x_3) (x_4 \vee x_5) \vee x_1 x_2 x_3; \quad (2.9) \end{aligned}$$

$$\begin{aligned}
z_2 &= \bigvee_{j \in \{0, 2\}} F_3(k_1 \geq i) F_2(k_2 \geq 2 - i) = \\
&= F_3(k_1 \geq 0) F_2(k_2 \geq 2) \vee F_3(k_1 \geq 2) F_2(k_2 \geq 0) = \\
&= x_4 x_5 \vee x_1 x_2 \vee x_1 x_3 \vee x_2 x_3. \quad (2.10)
\end{aligned}$$

Функции z_1 и z_2 отвечают условиям Н1 и Н2, но не отвечают условию Н3. Последнее не выполняется в том случае, если в одну из функций z входят все $m+1$ конъюнкций ранга m , покрывающих какую-либо конъюнкцию ранга $m+1$. Например, в функцию (2.10) входят конъюнкции $x_1 x_2$, $x_1 x_3$ и $x_2 x_3$, покрывающие конъюнкцию $x_1 x_2 x_3$. Свойство контроля входного вектора выполняется при этом за счет того, что указанная конъюнкция ранга $m+1$ включается в другую функцию z . Так конъюнкция $x_1 x_2 x_3$ присутствует в функции (2.9). Однако наличие избыточных конъюнкций ранга $m+1$ в функциях z_1 и z_2 приводит к тому, что построенные по ним тестеры не обладают свойством самопроверки, так как неисправности элементов, реализующих избыточные конъюнкции, не могут быть обнаружены на векторах с весом m .

Отсутствие избыточных конъюнкций обеспечивается только в одном случае, когда $m = n_1 = n_2 = n/2$. Этот случай соответствует коду $2mCm$, при этом множество B делится на два равных подмножества. Тогда функции z_1 и z_2 вычисляются по формулам:

$$z_1 = \bigvee_{j \in \{0, \dots, n/2\}} F_{n_1}(k_1 \geq j) F_{n_2}\left(k_2 \geq \frac{n}{2} - j\right); \quad (2.11)$$

$$z_2 = \bigvee_{i \in \{0, \dots, n/2\}} F_{n_1}(k_1 \geq i) F_{n_2}\left(k_2 \geq \frac{n}{2} - i\right), \quad (2.12)$$

где i и j имеют тот же смысл, что и в формулах (2.7) и (2.8).

Функции (2.11) и (2.12) позволяют строить $m/2m$ -СПТ. В них содержатся только конъюнкции ранга m , для них выполняются условия Н1—Н4. Определим функции для 3/6-СПТ. Множество $B = \{x_1 \div x_6\}$ делим на подмножества $B_1 = \{x_1, x_2, x_3\}$ и $B_2 = \{x_4, x_5, x_6\}$. Используя формулы (2.11) и (2.12), получаем

$$\begin{aligned}
z_1 &= \bigvee_{j \in \{1, 3\}} F_{n_1}(k_1 \geq j) F_{n_2}(k_2 \geq 3 - j) = \\
&= F_3(k_1 \geq 1) F_3(k_2 \geq 2) \vee F_3(k_1 \geq 3) F_3(k_2 \geq 0) = \\
&= (x_1 \vee x_2 \vee x_3) (x_4 x_5 \vee x_4 x_6 \vee x_5 x_6) \vee x_1 x_2 x_3; \quad (2.13)
\end{aligned}$$

$$\begin{aligned}
z_2 &= \bigvee_{i \in \{0, 2\}} F_{n_1}(k_1 \geq i) F_{n_2}(k_2 \geq 3 - i) = \\
&= F_3(k_1 \geq 0) F_3(k_2 \geq 3) \vee F_3(k_1 \geq 2) F_3(k_2 \geq 1) = \\
&= x_4 x_5 x_6 \vee (x_1 x_2 \vee x_1 x_3 \vee x_2 x_3) (x_4 \vee x_5 \vee x_6). \quad (2.14)
\end{aligned}$$

Проверяющий тест $m/2m$ -СПТ состоит из 2^m векторов, отвечающих следующему требованию: часть вектора, образованная переменными $x_i \in B_1$, является инверсной по отношению к части вектора, образованного переменными $x_i \in B_2$. На рис. 2.1 показана схема 2/4-СПТ, реализованная по функциям, полученным с помощью формул (2.11) и (2.12) (при этом $B_1 = \{x_1, x_2\}$, $B_2 = \{x_3, x_4\}$). В табл. 2.3 представлен проверяющий тест для этой схемы.

Эквивалентное преобразование функций (2.11) и (2.12) дает формулы

$$z_1 = \bigwedge_{j \in \{0, \dots, n/2\}} \left(F_{n_1}(k_1 \geq j+1) \vee F_{n_2}\left(k_2 \geq \frac{n}{2} - j + 1\right) \right); \quad (2.15)$$

$$z_2 = \bigwedge_{i \in \{0, \dots, n/2\}} \left(F_{n_1}(k_1 \geq i+1) \vee F_{n_2}\left(k_2 \geq \frac{n}{2} - i + 1\right) \right), \quad (2.16)$$

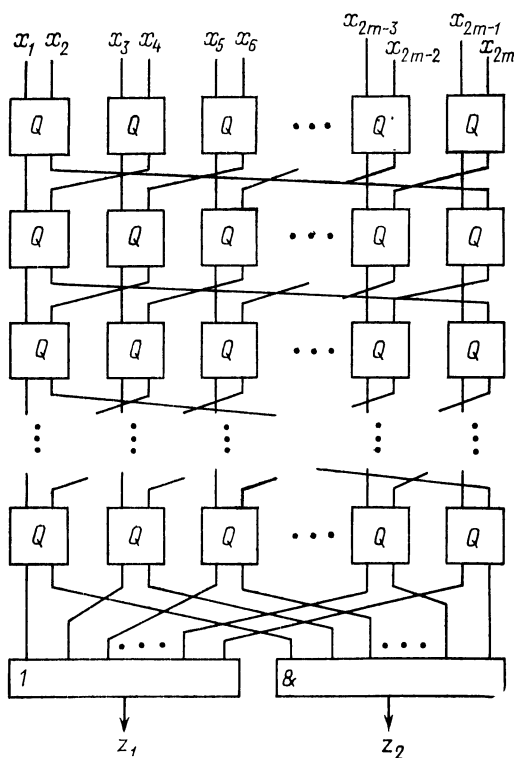


Рис. 2.7

с помощью которых могут быть получены тестеры с меньшим числом уровней r .

Например, для 3/6-СПТ получаем:

$$z_1 = (x_1 x_2 \vee x_1 x_3 \vee x_2 x_3 \vee x_4 x_5 x_6) (x_4 \vee x_5 \vee x_6); \quad (2.17)$$

$$z_2 = (x_1 \vee x_2 \vee x_3) (x_1 x_2 \times x_3 \vee x_4 x_5 \vee x_4 x_6 \vee x_5 x_6).$$

Симметрия кода $2mCm$ позволяет получить регулярную стандартную структуру $m/2m$ -СПТ. Такая структура была предложена в [75], а затем усовершенствована в [77], где показано, что любой $m/2m$ -СПТ может быть реализован с помощью структуры, показанной на рис. 2.7. Каждый блок Q содержит два элемента, реализующие дизъюнкцию

и конъюнкцию, причем дизъюнкция реализуется на правом, а конъюнкция на левом выходе элемента. На рис. 2.8 приведена схема 3/6-СПТ. Стандартная структура имеет следующие ха-

Таблица 2.6

x_1	x_2	x_3	\dots	x_{m-1}	x_m	x_{m+1}	x_{m+2}	\dots	x_{2m-1}	x_{2m}
1	1	1	\dots	1	1	0	0	\dots	0	0
0	1	1	\dots	1	1	1	0	\dots	0	0
0	0	1	\dots	1	1	1	1	\dots	0	0
\dots	\dots	\dots	\dots	\dots	\dots	\dots	\dots	\dots	\dots	\dots
0	0	0	\dots	0	1	1	1	\dots	1	0
1	1	1	\dots	1	0	0	0	\dots	0	1
1	1	1	\dots	0	0	0	0	\dots	1	1
\dots	\dots	\dots	\dots	\dots	\dots	\dots	\dots	\dots	\dots	\dots
1	0	0	\dots	0	0	0	1	\dots	1	1
0	0	0	\dots	0	0	1	1	\dots	1	1

рактеристики: $L = 4m^2 - 2m$, $N_1 = 2m^2 - 2m + 2$, $N_2 = 2m^2 - 2$, $r_1 = m$, $r_2 = m + \lceil \log_2 m \rceil$ и $t = 2m$. Множество векторов, включаемых в проверяющий тест, представлено в табл. 2.6. Для схемы на рис. 2.8 тест представлен в табл. 2.7. В табл. 2.8 показаны характеристики основных $m/2m$ -СПТ, причем схемы СПТ1 получены по формулам (2.11) и (2.12), схемы СПТ2 — по формулам (2.15) и (2.16), а схемы СПТ3 соответствуют стандартной структуре тестера.

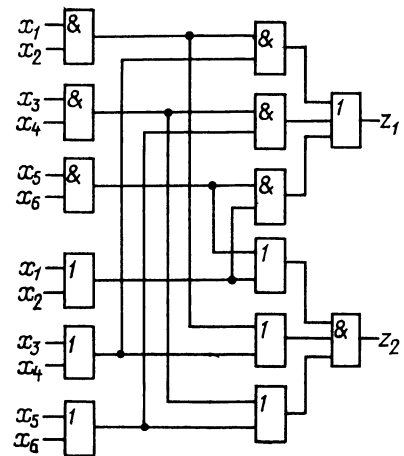


Рис. 2.8

2.4. Тестеры для кодов $nC1$

Коды $nC1$ с точки зрения построения m/n -СПТ также составляют особый класс равновесного кода. Словам кода $nC1$ соответствуют конъюнкции ранга 1, для реализации которых не требуется специальных элементов. В связи с этим первый подход к синтезу m/n -СПТ, основанный на разложении базовой функции, для кодов $nC1$ не может быть использован,

Таблица 27

x_1	x_2	x_3	x_4	x_5	x_6
1	1	1	0	0	0
0	1	1	1	0	0
0	0	1	1	1	0
1	1	0	0	0	1
1	0	0	0	1	1
0	0	0	1	1	1

Таблица 28

m/n	Тип тестера	Характеристики						
		L	N_1	N_2	r_1	r_2	t	k
2/4	СПТ1	12	6	6	2	2	4	2,0
3/6	СПТ1	38	16	22	4	5	8	1,0
	СПТ2	36	14	22	3	6	8	1,0
	СПТ3	30	14	16	3	4	6	2,33
4/8	СПТ1	95	33	62	4	7	16	1,19
	СПТ2	91	29	62	3	7	16	1,19
	СПТ3	56	26	30	4	6	8	3,5
5/10	СПТ1	226	66	158	4	9	32	1,0
	СПТ2	212	60	158	3	9	32	1,0
	СПТ3	90	42	48	5	8	10	9,1

2.4.1. Универсальные методы синтеза $1/n$ -СПТ

Рассмотрим метод построения $1/n$ -СПТ, основанный на преобразовании кодов [48]. Используется двухкаскадное преобразование по схеме $nC1 \rightarrow n'Cm \rightarrow 2C1$. В этом случае тестер состоит из двух блоков: преобразователя $nC1 \rightarrow n'Cm$ и тестера m/n' -СПТ. Параметры n' и m выбираются так, чтобы выполнялось условие: $n' > t$ (m/n' -СПТ). Преобразователь $nC1 \rightarrow n'Cm$ строится следующим образом. Устанавливается n' элементов

ИЛИ, не связанных между собой и имеющих по m входов. Каждый из n входов преобразователя соединяется со входами m элементов ИЛИ так, чтобы на его выходах были сформированы все t слов проверяющего теста m/n' -СПТ. Наиболее эффективным является преобразование по схеме $nC1 \rightarrow 2mCm \rightarrow \rightarrow 2C1$.

На рис. 2.9. показан преобразователь $4C1 \rightarrow 4C2$. Он осуществляет преобразование четырех слов кода $4C1$ в четыре слова кода $4C2$, которые входят в проверяющий тест $2/4$ -СПТ (см. табл. 2.3). В табл. 2.9 проиллюстрировано это преобразование. Очевидно, что описанный преобразователь обладает свойствами контроля входного вектора и самопроверки. Из кодов вида $nC1$ только для кода $3C1$ не может быть построен описанный преобразователь. Код $3C1$ имеет только три кодовых слова и поэтому не может быть преобразован в код $4C2$, так как $2/4$ -СПТ требует для проверки четыре кодовых слова. Для всех остальных кодов $nC1$ подобные преобразователи существуют. При $4 \leq n \leq 6$ необходимо осуществлять преобразование в код $4C2$, при $7 \leq n \leq 20$ — в код $6C3$, при $20 \leq n \leq 70$ — в код $8C4$ и т. д. В табл. 2.10 представлены характеристики для ряда $1/n$ -СПТ. Тестеры, реализованные описанным методом, обозначены как СПТ1. В них в качестве $m/2m$ -СПТ используются СПТ3 из табл. 2.8. Для всех тестеров $t = n$.

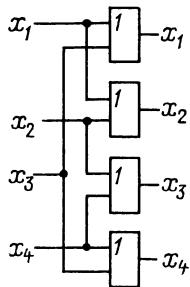


Рис. 2.9

Таблица 2.9

Код 4C1				Код 4C2			
x_1	x_2	x_3	x_4	x_1	x_2	x_3	x_4
1	0	0	0	1	1	0	0
0	1	0	0	0	1	1	0
0	0	1	0	1	0	0	1
0	0	0	1	0	0	1	1

Метод синтеза $1/n$ -СПТ, относящийся ко второму из указанных выше подходов к синтезу тестеров, основан на введении в базовую функцию избыточных конъюнкций [37]. Рассмотрим идею метода. Разобьем множество входных переменных $\{x_1, x_2, \dots, x_n\}$ на два подмножества A и B (в данном случае их удобно обозначить разными буквами). В подмножество A включим все переменные с нечетными индексами, а в подмножество

Таблица 2.10

$1/n$	Тип тестера	Характеристики					
		L	N_1	N_2	r_1	r_2	k
1/4	СПТ1	20	10	10	3	3	1,0
	СПТ2	16	8	8	3	3	1,0
	СПТ3	20	10	10	3	3	1,0
1/5	СПТ1	22	10	12	3	4	1,5
	СПТ2	22	10	12	3	4	1,5
	СПТ3	25	11	14	3	5	1,5
1/6	СПТ1	24	10	14	3	4	2,0
	СПТ2	24	10	14	3	4	1,0
	СПТ3	30	12	18	3	5	1,0
1/7	СПТ1	51	20	31	4	5	1,33
	СПТ2	33	14	19	4	6	1,33
	СПТ3	49	15	34	3	6	1,33
	СПТ4	38	18	20	6	7	1,33
1/8	СПТ1	54	20	34	4	5	1,0
	СПТ2	36	15	21	4	7	1,0
	СПТ3	56	16	40	3	6	1,0
	СПТ4	44	20	24	6	8	1,0
1/10	СПТ1	60	20	40	4	6	1,0
	СПТ2	44	17	27	4	7	1,5
	СПТ3	50	17	33	3	6	1,0
	СПТ4	48	20	28	6	8	1,0
1/12	СПТ1	66	20	46	4	6	1,0
	СПТ2	48	17	31	4	7	1,0
	СПТ3	84	20	64	3	8	1,0
	СПТ4	60	25	35	7	9	1,0

1/n	Тип тестера	Характеристики					
		L	N_1	N_2	r_1	r_2	k
1/24	СПТ1	152	34	118	5	8	1,0
	СПТ2	96	30	66	4	7	1,0
	СПТ3	168	38	130	3	9	1,0
	СПТ4	108	40	72	7	11	1,0

B — все переменные с четными индексами. Через n_a и n_b обозначим соответственно мощности множеств A и B . Переменные $x_j \in A$ будем также обозначать буквами a_j ($j \in \{1, 2, \dots, n_a\}$), а переменные $x_j \in B$ — буквами b_j ($j \in \{1, 2, \dots, n_b\}$). Тогда можно записать, что $A = \{a_1, a_2, \dots, a_{n_a}\}$ и $B = \{b_1, b_2, \dots, b_{n_b}\}$.

Составим функции следующего вида:

$$f_1 = \bigvee_{j \in \{1, 2, \dots, n_a\}} a_j \vee \left(\bigvee_{\substack{p, k \in \{1, 2, \dots, n_b\} \\ p \neq k}} b_p b_k \right) = f_1' \vee f_1'', \quad (2.18)$$

$$f_2 = \bigvee_{j \in \{1, 2, \dots, n_b\}} b_j \vee \left(\bigvee_{\substack{p, k \in \{1, 2, \dots, n_a\} \\ p \neq k}} a_p a_k \right) = f_2' \vee f_2''.$$

Например, для кода 7C1 образуем подмножества $A = \{x_1, x_3, x_5, x_7\}$ и $B = \{x_2, x_4, x_6\}$. Тогда получим

$$f_1 = x_1 \vee x_3 \vee x_5 \vee x_7 \vee x_2 x_4 \vee x_2 x_6 \vee x_4 x_6,$$

$$f_2 = x_2 \vee x_4 \vee x_6 \vee x_1 x_3 \vee x_1 x_5 \vee x_1 x_7 \vee x_3 x_5 \vee x_3 x_7 \vee x_5 x_7.$$

Тестер, реализованный по формулам (2.18), обладает свойством контроля входного вектора. Если на его вход поступает слово кода $nC1$, то только одна из переменных a_1, \dots, a_{n_a} , b_1, \dots, b_{n_b} принимает значение 1. Поэтому только одна из функций, f_1 или f_2 , равна 1. Если входное слово имеет вес $w = 0$, то, очевидно, что $f_1 = f_2 = 0$. Если вес входного слова $w > 1$, то $f_1 = f_2 = 1$. Например, если $\rho = a_i a_j$ ($i \neq j$), то функция $f_1 = 1$, так как содержит все конъюнкции первого ранга вида a_j , а $f_2 = 1$, так как она содержит все конъюнкции второго ранга из переменных a_j . Однако тестер не обладает свойством самопроверки, поскольку функции f_1 и f_2 содержат избыточные конъюнкции.

Специальное представление системы (2.18) обеспечивает выполнение всех свойств СПТ [26]. При этом функции f_1' и f_2' заменяются функциями z_1' и z_2' следующего вида:

$$z_i' = \bigvee_{\substack{u, v \in w \\ u \neq v}} G_u G_v, \quad (2.19)$$

где $t \in \{1, 2\}$; ω — множество индексов логических сумм вида $G_p (p \in \{u, v\})$; $G_p = S_p \vee Q_p$;

$$S_p = a_{j_1} \vee a_{j_2} \vee \dots \vee a_{j_q} (j_1, j_2, \dots, j_q \in \{1, 2, \dots, n_a\});$$

$$Q_p = b_{j_1} \vee b_{j_2} \vee \dots \vee b_{j_r} (j_1, j_2, \dots, j_r \in \{1, 2, \dots, n_b\}).$$

Данные функции находятся по некоторому фиксированному множеству логических сумм G_p с выполнением следующего требования: каждая входная переменная должна входить в две и только в две суммы G_p . Тогда логическое произведение двух таких сумм дает конъюнкцию, состоящую из одной логической переменной. Такие конъюнкции входят в функции f_1' и f_2' . Функции z_1' и z_2' находятся по следующим правилам. Функция z_1' содержит n_a произведений вида $G_u G_v$, каждое из которых дает конъюнкцию, состоящую из переменной $x_j \in A$. Функция z_2' содержит n_b произведений $G_u G_v$, каждое из которых включает в себя конъюнкцию, состоящую из переменных $x_i \in B$. Обозначим через $G^i_{a_j} (G^i_{b_j})$ сумму, в которую входит переменная $a_j (b_j)$. Верхний индекс i указывает номер суммы ($i \in \{1, 2\}$). Тогда

$$\begin{aligned} z_1' &= G^1_{a_1} G^2_{a_1} \vee G^1_{a_2} G^2_{a_2} \vee \dots \vee G^1_{n_a} G^2_{n_a}, \\ z_2' &= G^1_{b_1} G^2_{b_1} \vee G^1_{b_2} G^2_{b_2} \vee \dots \vee G^1_{n_b} G^2_{n_b}. \end{aligned} \quad (2.20)$$

Например, для кода 7C1 можно образовать следующие суммы: $G_1 = x_1 \vee x_2$, $G_2 = x_2 \vee x_3$, $G_3 = x_3 \vee x_4$, $G_4 = x_4 \vee x_5 \vee x_7$, $G_5 = x_5 \vee x_6$, $G_6 = x_1 \vee x_6 \vee x_7$. С помощью (2.20) получаем

$$\begin{aligned} z_1' &= (x_1 \vee x_2) (x_1 \vee x_6 \vee x_7) \vee (x_2 \vee x_3) (x_3 \vee x_4) \vee \\ &\vee (x_4 \vee x_5 \vee x_7) (x_5 \vee x_6) \vee (x_4 \vee x_5 \vee x_7) (x_1 \vee x_6 \vee x_7), \\ z_2' &= (x_1 \vee x_2) (x_2 \vee x_3) \vee (x_3 \vee x_4) (x_4 \vee x_5 \vee x_7) \vee \\ &\vee (x_5 \vee x_6) (x_1 \vee x_6 \vee x_7). \end{aligned}$$

Тестер, реализованный по системе (2.20), будет обладать свойством самопроверки при условии, что каждая сумма вида G_p реализуется одним элементом. Для обеспечения свойства контроля входного вектора необходимо также выполнение условий

$$f_1'' \rightarrow z_1', \quad f_2'' \rightarrow z_2', \quad (2.21)$$

в соответствии с которыми функции z_1' и z_2' должны содержать все соответствующие конъюнкции второго ранга вида $a_p a_k$ и $b_p b_k$. В рассматриваемом примере данное условие выполняется для функции z_1' , но не выполняется для функции z_2' , так как в ней не содержится конъюнкция $x_1 x_7$. Выполнение условий (2.21) обеспечивается за счет специального расширения функций z_1' и z_2' .

Способ расширения функций поясним на следующем примере. Для того чтобы функция z_2' содержала конъюнкцию x_1x_7 , заменим в ней сумму $G_3 = x_3 \vee x_4$ расширенной суммой $G_7 = G_3 + G_1 = (x_3 \vee x_4) \vee (x_1 \vee x_2)$. Круглые скобки в этом выражении указывают на то, что каждая из исходных сумм реализуется на отдельном элементе. Получаем выражение

$$z_2 = (x_1 \vee x_2)(x_2 \vee x_3) \vee [(x_3 \vee x_4) \vee (x_1 \vee x_2)] \wedge \wedge (x_4 \vee x_5 \vee x_7) \vee (x_5 \vee x_6)(x_1 \vee x_6 \vee x_7). \quad (2.22)$$

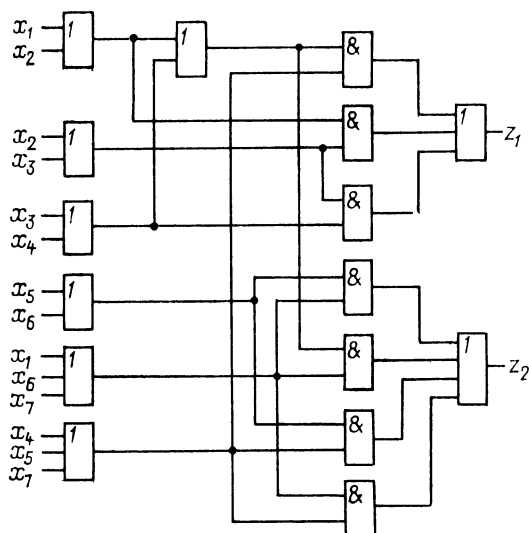


Рис. 2.10

При построении схемы по формуле (2.22) для реализации суммы G_7 используется элемент ИЛИ с двумя входами, один из которых соединен с элементом, реализующим сумму $G_3 = x_3 \vee x_4$, а второй — с элементом, реализующим сумму $G_1 = x_1 \vee x_2$. Очевидно, что первый вход является избыточным. Для решения задачи проверки этого входа произведем расширение функции z_1' так, чтобы она также содержала сумму G_7 и чтобы эта сумма была образована путем расширения суммы G_1 . При этом необходимо следить, чтобы не было нарушено уже достигнутое свойство контроля входного вектора, т. е. чтобы в конечном счете выполнялось условие

$$z_1 f_2 = 0, \quad z_2 f_1 = 0.$$

Для рассматриваемого случая получаем

$$z_1 = [(x_3 \vee x_4) \vee (x_1 \vee x_2)] (x_1 \vee x_6 \vee x_7) \vee (x_2 \vee x_3) (x_3 \vee x_4) \vee \vee (x_4 \vee x_5 \vee x_7) (x_5 \vee x_6) \vee (x_4 \vee x_5 \vee x_7) (x_1 \vee x_6 \vee x_7). \quad (2.23)$$

Выражения (2.22) и (2.23) описывают 1/7-СПТ, обладающий свойством самопроверки при условии, что каждая сумма, заключенная в круглые или квадратные скобки, реализуется отдельным элементом. Одинаковые суммы реализуются одним и тем же элементом (рис. 2.10). Поскольку элемент ИЛИ, вычисляющий сумму G_7 , входит в схему, реализующую как функцию z_1 , так и функцию z_2 , то неисправность обоих его входов обнаруживается на соответствующих словах кода 7С1.

Структура 1/ n -СПТ зависит от вида логических сумм G_p и способа расширения функций z_1' и z_2' . В [37] приведен алгоритм, позволяющий получать простые структуры тестеров. В табл. 2.10 они обозначены как СПТ2. Особенность таких тестеров состоит в том, что схемы для меньших значений n могут быть получены из схем для больших значений n путем подачи сигнала «логический 0» на определенные входы этих схем. Например, тестеры для $n = 13 \div 23$ целесообразно получать из 1/24-СПТ, который описывается следующей системой функций: $25 = 1+3+11+12$, $26 = 16+18+20+21$, $27 = 2+8+17+23$, $28 = 2+3+4+6$, $29 = 14+15+22+24$, $30 = 1+7+13+19$, $31 = 19+21+23+24$, $32 = 5+11+14+20$, $33 = 5+6+7+9$, $34 = 13+15+17+18$, $35 = 4+10+16+22$, $36 = 8+9+10+12$, $37 = 25+26+27$, $38 = 28+29+30$, $39 = 28+31+32$, $40 =$

Таблица 2.11

n	Исключаемые переменные
24	
23	11
22	9, 12
21	9, 11, 12
20	9, 12, 15, 24
19	9, 11, 12, 15, 24
18	4, 9, 12, 15, 16, 24
17	4, 9, 11, 12, 15, 16, 24
16	2, 4, 9, 12, 15, 16, 17, 24
15	1, 4, 9, 12, 13, 15, 16, 19, 24
14	1, 4, 7, 9, 12, 13, 15, 16, 19, 24
13	1, 2, 4, 9, 12, 13, 15, 16, 17, 19, 24

$= 33+34+35$, $41 = 33+29+27$, $42 = 36+26+30$, $43 = 36+34+32$, $44 = 25+31+35$, $z_1 = 37 \times 38 + 39 \times 40 + 41 \times 42 + 43 \times 44$, $z_2 = 37 \times 40 + 39 \times 42 + 41 \times 44 + 43 \times 38$.

В табл. 2.11 для каждого $n = 13 \div 23$ указаны номера тех переменных, простое исключение которых из приведенных функций дает функции, описывающие соответствующий $1/n$ -СПТ. Тестеры для $n = 8 \div 11$ получают из $1/12$ -СПТ (табл. 2.12), который описывается следующей системой функций:

Таблица 2.12

$13 = 2+5+9+12$, $14 = 1+4+8+11$,
 $15 = 3+6+7+10$, $16 = 1+10+13$,
 $17 = 2+7+14$, $18 = 3+12+14$, $19 =$
 $= 4+9+15$, $20 = 5+8+15$, $21 =$
 $= 6+11+13$, $z_1 = 16 \times 17 + 18 \times 19 +$
 $+ 20 \times 21$, $z_2 = 16 \times 19 + 18 \times 21 +$
 $+ 20 \times 17$.

Для основных тестеров с $n = 4 \div 7$ получены структуры, описываемые следующими функциями: тестер $1/5$ -СПТ

n	Исключаемые переменные
12	
11	4
10	4, 9
9	2, 4, 8
8	3, 4, 8, 9

$$z_1 = (x_1 \vee x_2)(x_2 \vee x_3 \vee x_5) \vee (x_3 \vee x_4)(x_1 \vee x_4 \vee x_5), \quad (2.24)$$

$z_2 = [(x_3 \vee x_4) \vee (x_1 \vee x_4 \vee x_5)][(x_2 \vee x_3 \vee x_5) \vee (x_1 \vee x_2)]$;
тестер $1/6$ -СПТ

$$z_1 = (x_1 \vee x_2 \vee x_6)(x_2 \vee x_3 \vee x_5) \vee (x_3 \vee x_4 \vee x_6)(x_1 \vee x_4 \vee x_5),$$

$$z_2 = [(x_3 \vee x_4 \vee x_6) \vee (x_1 \vee x_4 \vee x_5)][(x_2 \vee x_3 \vee x_5) \vee (x_1 \vee x_2 \vee x_6)];$$

тестер $1/7$ -СПТ

$$z_1 = [(x_2 \vee x_4) \vee x_1 \vee x_5][(x_1 \vee x_3) \vee x_2 \vee x_7] \vee$$

$$\vee (x_5 \vee x_6 \vee x_7)[(x_2 \vee x_4) \vee x_3 \vee x_6],$$

$$z_2 = [(x_2 \vee x_4) \vee x_1 \vee x_5](x_5 \vee x_6 \vee x_7) \vee [(x_1 \vee x_3) \vee x_4] \wedge$$

$$\wedge [(x_2 \vee x_4) \vee x_3 \vee x_6] \vee (x_4 \vee x_6 \vee x_7)[(x_1 \vee x_3) \vee x_2 \vee x_7].$$

Тестер $1/4$ -СПТ приведен на рис. 2.4.

Рассмотренный метод модифицирован с целью получения быстродействующих $1/n$ -СПТ с трехуровневой структурой [73]. В этом случае исключается возможность использования указанного выше метода расширения функций z_1' и z_2' , так как он приводит к увеличению числа уровней. Функции z_1 и z_2 , описывающие тестер, находятся как дизъюнкции произведений логических сумм, причем каждая логическая сумма не может содержать в себе какую-либо другую логическую сумму. В [73] приведен алгоритм, согласно которому для $n \in \{4, 5, 6, 10\}$ функции z_1 и z_2 вычисляются в соответствии с выражением

(2.19), а для остальных значений n — в соответствии с выражением следующего вида:

$$z_t = \bigvee_{\substack{u, v, s \in \omega \\ u \neq v, u \neq s, v \neq s}} G_u G_v G_s. \quad (2.25)$$

В табл. 2.10 данные тестеры обозначены как СПТЗ. Увеличение быстродействия приводит к увеличению сложности. До значений $n \leq 6$ тестеры СПТ2 имеют меньшую сложность при том же быстродействии, чем СПТЗ.

Для основных тестеров получаем следующие структуры: тестер 1/4-СПТ

$$z_1 = (x_1 \vee x_2) (x_2 \vee x_3), \quad z_2 = (x_3 \vee x_4) (x_1 \vee x_4);$$

тестер 1/5-СПТ

$$\begin{aligned} z_1 &= (x_1 \vee x_2) (x_2 \vee x_3 \vee x_5) \vee (x_3 \vee x_4) (x_1 \vee x_4 \vee x_5), \\ z_2 &= (x_1 \vee x_2) (x_1 \vee x_4 \vee x_5) \vee (x_3 \vee x_4) (x_2 \vee x_3 \vee x_5) \vee \\ &\quad \vee (x_1 \vee x_4 \vee x_5) (x_2 \vee x_3 \vee x_5); \end{aligned}$$

тестер 1/6-СПТ

$$\begin{aligned} z_1 &= (x_1 \vee x_2 \vee x_6) (x_2 \vee x_3 \vee x_5) \vee (x_3 \vee x_4 \vee x_6) (x_1 \vee x_4 \vee x_5) \vee \\ &\quad \vee (x_1 \vee x_2 \vee x_6) (x_3 \vee x_4 \vee x_6), \\ z_2 &= (x_1 \vee x_2 \vee x_6) (x_1 \vee x_4 \vee x_5) \vee (x_3 \vee x_4 \vee x_6) (x_2 \vee x_3 \vee x_5) \vee \\ &\quad \vee (x_1 \vee x_4 \vee x_5) (x_2 \vee x_3 \vee x_5); \end{aligned}$$

тестер 1/7-СПТ

$$\begin{aligned} 8 &= 1+2+5+6, \quad 9 = 2+3+6+7, \quad 10 = 1+3+6, \quad 11 = 3+4+7, \\ 12 &= 2+4+5+7, \quad 13 = 1+4+5, \quad z_1 = 8 \times 9 \times 10 + 9 \times 11 \times 12 + \\ &+ 8 \times 12 \times 13, \quad z_2 = 8 \times 9 \times 12 + 9 \times 10 \times 11 + 11 \times 12 \times 13 + 8 \times \\ &\times 10 \times 13; \end{aligned}$$

тестер 1/8-СПТ

$$\begin{aligned} 9 &= 1+2+5+6, \quad 10 = 2+3+6+7, \quad 11 = 1+3+6+8, \quad 12 = 3+4+ \\ &+ 7+8, \quad 13 = 1+4+5+8, \quad 14 = 2+4+5+7, \quad z_1 = 9 \times 10 \times 11 + \\ &+ 10 \times 12 \times 14 + 11 \times 12 \times 13 + 9 \times 13 \times 14, \quad z_2 = 9 \times 10 \times 14 + 10 \times \\ &\times 11 \times 12 + 12 \times 13 \times 14 + 9 \times 11 \times 13; \end{aligned}$$

тестер 1/10-СПТ

$$\begin{aligned} 11 &= 1+2+6+8, \quad 12 = 2+3+7+9, \quad 13 = 3+4+8+10, \quad 14 = 4+ \\ &+ 5+6+9, \quad 15 = 1+5+7+10, \quad z_1 = 11 \times 12 + 12 \times 13 + 13 \times 14 + \\ &+ 14 \times 15 + 11 \times 15, \quad z_2 = 11 \times 13 + 12 \times 14 + 13 \times 15 + 11 \times 14 + \\ &+ 12 \times 15; \end{aligned}$$

тестер 1/12-СПТ

$$\begin{aligned} 13 &= 1+2+5+7+8+10, \quad 14 = 2+3+6+8+9+11, \quad 15 = 1+3+ \\ &+ 4+9+10+12, \quad 16 = 2+4+5+7+10+11, \quad 17 = 3+5+6+8+ \\ &+ 11+12, \quad 18 = 1+4+6+7+9+12, \quad z_1 = 13 \times 14 \times 16 + 14 \times 15 \times \\ &\times 17 + 15 \times 16 \times 18 + 13 \times 16 \times 17 + 14 \times 17 \times 18 + 13 \times 15 \times 18, \quad z_2 = \end{aligned}$$

$$= 13 \times 14 \times 17 + 14 \times 15 \times 18 + 13 \times 15 \times 16 + 14 \times 16 \times 17 + 15 \times 17 \times 18 + 13 \times 16 \times 18.$$

Любой СПТЗ представляется в виде трехуровневой структуры типа ИЛИ — И — ИЛИ и может быть реализован с помощью двух ПЛМ. На рис. 2.11 показана реализация на ПЛМ 1/4-СПТ, в табл. 2.13 приведены характеристики ПЛМ для основных тестеров.

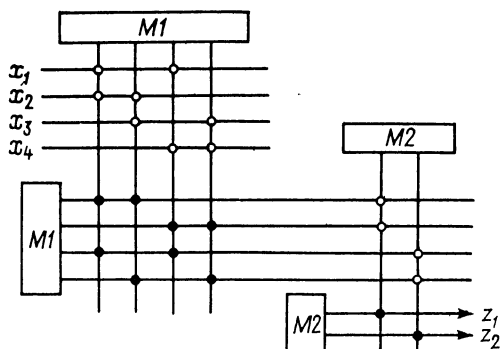


Рис. 2.11

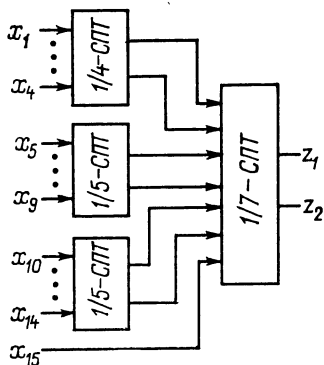


Рис. 2.12

Таблица 2.13

1/n-СПТ						
1/4	1/6	1/7	1/8	1/10	1/12	1/24
Х а р а к т е р и с т и к и П Л М						
M1 [4, 4, 4]	M1 [6, 4, 6]	M1 [7, 6, 7]	M1 [8, 6, 8]	M1 [10, 5, 10]	M1 [12, 6, 12]	M1 [24, 12, 24]
M2 [4, 2, 2]	M2 [6, 2, 2]	M2 [7, 2, 2]	M2 [8, 2, 2]	M2 [10, 2, 2]	M2 [12, 2, 2]	M2 [24, 2, 2]

В [24] предложен каскадный принцип построения 1/n-СПТ. Показано, что любой 1/n-тестер может быть реализован в виде структуры типа «дерево», в вершинах которого расположены тестеры со значениями $n' < n$. В качестве примера на рис. 2.12 показана одна из возможных реализаций 1/15-СПТ, состоящая из двух уровней тестеров. Так как каждый из тестеров, входящих в такую структуру, также может быть реализован на основе каскадного принципа, то возможно большое число модификаций одного и того же 1/n-СПТ, в том числе с большим числом уровней тестеров. Так на рис. 2.13 представлена структура 1/15-СПТ, полученная из структуры рис. 2.12 путем замены 1/7-СПТ каскадным соединением 1/4- и 1/5-тестеров. Фактически каскадный принцип также реализует идею преобразования кодов. Например, структура на рис. 2.13 осуществляет преобразование кодов по схеме $15C1 \rightarrow 7C1 \rightarrow 5C1 \rightarrow 2C1$. В табл. 2.10

данные тестеры обозначены как СПТ4. Приведены характеристики тех структур, которые имеют наименьшую сложность. При этом в качестве $1/n$ -тестеров, входящих в каскадные структуры, использованы СПТ2.

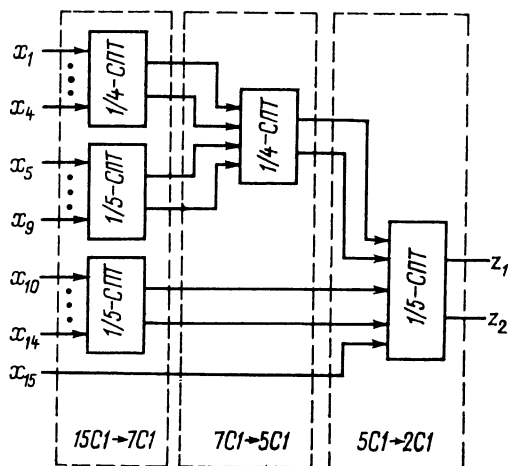


Рис. 2.13

2.4.2. $1/3$ -тестеры

Вышеприведенные методы позволяют строить $1/n$ -СПТ со значениями $n \geq 4$. Для определения способов реализации $1/3$ -СПТ рассмотрим следующие положения.

Утверждение 2.1. При построении m/n -СПТ в виде КС вход тестера не может быть соединен с выходом непосредственно или через элементы ИЛИ.

Данное положение достаточно очевидно и легко доказывается путем раздельного рассмотрения случаев, когда $m = 1$ и $m \geq 2$.

Утверждение 2.2. Для проверки любого m/n -СПТ, реализованного в виде КС, требуется проверяющий тест длиной $t \geq 4$.

Доказательство. Из утверждения 2.1 следует, что на выходах m/n -СПТ должны быть установлены два элемента, один из которых соединяется с выходом z_1 , а другой — с выходом z_2 . При этом возможны два случая: когда элементы реализуют одну и ту же функцию и когда — разные функции. Первый случай проиллюстрирован на рис. 2.14, а. Определим минимальное множество тестовых наборов, необходимое для проверки выходных элементов ИЛИ в схеме m/n -СПТ. Элементы имеют минимальное число входов. Известно [12], что для проверки двухвходового элемента ИЛИ требуется три набора: 00, 01, 10. Однако оба элемента ИЛИ в схеме m/n -СПТ не могут

быть проверены только тремя наборами, так как требуется согласование наборов, проверяющих первый элемент, с наборами, проверяющими второй элемент. Принцип согласования наборов показан на рис. 2.14, а. При подаче на вход элемента Э1 набора 10 или 01 на входе Э2 должен быть зафиксирован набор 00, так как проверка схемы осуществляется на словах *n*Ст-кода, при которых на выходах $z_1 z_2$ устанавливаются значения 10 или 01. Поэтому минимальное множество кодовых слов, необходимое для проверки выходных элементов тестера, равно 4.

На рис. 2.14, б показан случай, когда выходные элементы тестера реализуют различные функции. В соответствии с утвер-

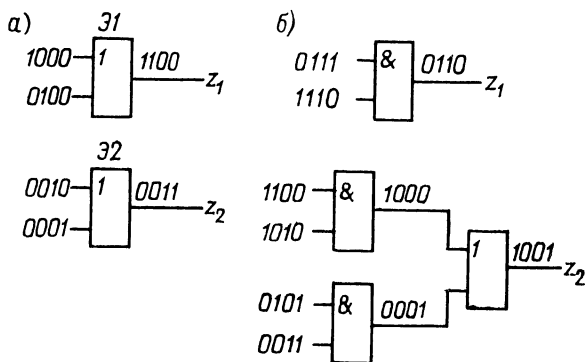


Рис. 2.14

ждением 2.1 входы выходного элемента ИЛИ должны быть соединены с выходами элементов И. Эти соединения могут осуществляться непосредственно (как это показано на рисунке) или через дополнительные элементы ИЛИ, что не имеет принципиального значения. На рис. 2.14, б показаны множества тестовых наборов, необходимые для проверки выходных элементов тестера. Схема, реализующая функцию z_2 , требует для проверки не менее четырех наборов, так как каждый из элементов схемы проверяется тремя наборами, но наборы 11 проверки элементов И не могут быть объединены в один тестовый набор проверки схемы, поскольку при нем на входах элемента ИЛИ образуется набор 11, не входящий в проверяющий тест элемента. Таким образом, утверждение доказано.

Следствие. Самопроверяемый 1/3-тестер не может быть реализован в виде КС.

Следствие непосредственно вытекает из утверждения 2.2, так как код ЗС1 содержит только три кодовых слова, а любой комбинационный m/n -СПТ требует для своей проверки не менее четырех кодовых слов.

Самопроверяемые 1/3-тестеры реализуются в виде многотактных схем. В [55] предложена структура 1/3-СПТ, в кото-

рой осуществляется двухкаскадное преобразование кода. Код 3С1 преобразуется в код 4С1 с помощью схемы с памятью, а затем код 4С1 переводится в код 2С1 с помощью комбинационного 1/4-СПТ. Преобразователь 3С1 → 4С1 строится следующим образом. Два входа преобразователя соединяются непо-

Таблица 2.14

s	$x_1 x_2 x_3$		
	100	010	001
1	(1), 01	2	(1), 01
2	3	(2), 10	3
3	(3), 10	4	(3), 10
4	1	(4), 01	1

Таблица 2.15

s	$x_1x_2x_3$			
	$\begin{matrix} 1 & 0 & 0, \\ 0 & 0 & 1 \end{matrix}$		0 1 0	
	y			
	y_1	y_2	y_3	y_4
1	0	1	1	0
2	1	0	1	0
3	1	0	0	1
4	0	1	0	1

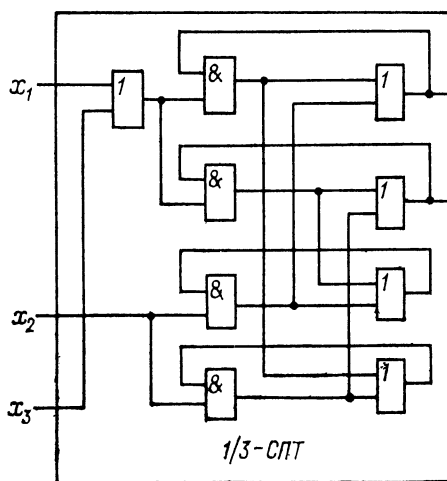


Рис. 2.15

средственно с двумя его выходами. Сигнал с третьего входа преобразуется в парафазный сигнал с помощью схемы с памятью, выходы которой соединяются с двумя другими выходами преобразователя. Недостаток данной схемы тестера состоит в наличии состязаний между сигналами.

С целью исключения этого недостатка целесообразно 1/3-СПТ строить в виде АКА, имеющего три входных состояния, которые кодируются словами кода 3С1, и два выходных состоя-

ния 01 и 10. Каждому входному состоянию ставятся в соответствие два устойчивых внутренних состояний с противоположными значениями выходов. В табл. 2.14 представлена ТП автомата, моделирующего 1/3-СПТ. Для получения ПСИ-свойств автомат кодируется кодом 4С2 с помощью метода кодирования состояний по столбцам ТП. Особенности ТП позволяют реали-

зовать ПСП-автомат с помощью 1-реализации [34], при этом возможно совмещение определяющих разрядов по первому и третьему столбцам ТП [33]. В табл. 2.15 приведено кодирование строк ТП, которое позволяет также осуществить полное совмещение преобразователей ВП и ЛП автомата по методу [38]. С помощью формул (1.5) и (1.6) находим систему функций, описывающую структуру АКА, реализующего 1/3-СПТ:

$$\begin{aligned} y_1 &= x_1 y_1 \vee x_2 y_3 \vee x_3 y_1, & y_3 &= x_1 y_2 \vee x_2 y_3 \vee x_3 y_2, \\ y_2 &= x_1 y_2 \vee x_2 y_4 \vee x_3 y_2, & y_4 &= x_1 y_1 \vee x_2 y_4 \vee x_3 y_4, \\ z_1 &= y_1, & z_2 &= y_2. \end{aligned} \quad (2.26)$$

На рис. 2.15 представлена схема 1/3-СПТ с характеристиками: $L = 18$, $N_1 = N_2 = 9$, $r_1 = r_2 = 3$. Для проверки схемы

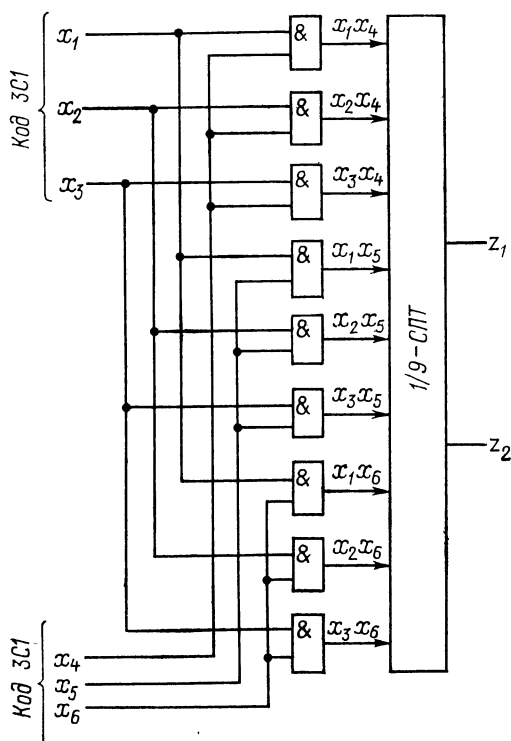


Рис. 2.16

требуется последовательность из пяти слов кода 3C1. Схема может быть реализована на одной ПЛМ.

Комбинационная схема может быть использована для контроля кода 3C1 в сочетании с каким-либо другим кодом nCm [61]. В этом случае строится преобразователь сокращенного кода $(n+3)C(n+1)$ в код $qC1$ ($q = 3C_n^m$) путем логического

умножения каждой переменной 3С1-кода на каждую конъюнкцию, соответствующую одному из слов nCm -кода. Последовательно с этим преобразователем устанавливается $1/q$ -СПТ. На рис. 2.16 показана схема контроля двух кодов 3С1.

2.5. Универсальные методы синтеза тестеров для кодов « m из n »

Универсальные методы обеспечивают построение m/n -СПТ для любых кодов nCm при $m \geq 2$. Все эти методы основаны на принципе преобразования кодов. Первый универсальный метод был предложен в [48]. Он базируется на трехкаскадном преобразовании кодов по схеме $nCm \rightarrow qC1 \rightarrow 2kCk \rightarrow 2C1$, где $q =$

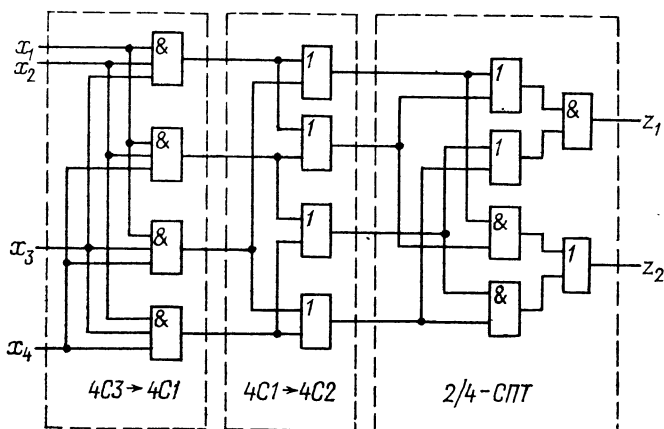


Рис. 2.17

$= C_n^m$. Первое преобразование $nCm \rightarrow qC1$ осуществляется при помощи схемы, состоящей из C_n^m независимых элементов И, на выходе каждого из которых реализуется конъюнкция, соответствующая одному из слов кода nCm . Такая схема обладает свойствами контроля входного вектора и самопроверки. Два других преобразования реализуются при помощи $1/q$ -СПТ, синтезированного методом, описанным на стр. 44 для СПТ1. На рис. 2.17 приведена полная схема $3/4$ -СПТ, состоящая из двух преобразователей кода ($4C3 \rightarrow 4C1$, $4C1 \rightarrow 4C2$) и $2/4$ -СПТ. В табл. 2.16 приведены основные характеристики тестеров, построенных универсальными методами. Тестеры, реализованные методом [48], обозначены как СПТ4.

Тестеры СПТ4 имеют большую сложность. Уменьшение сложности достигнуто в [65], где используется та же схема преобразования, но при этом $q < C_n^m$. Например, для $n \leq 4m$ $q \in \{4, 5, 6\}$. В этом случае тестеры строятся на основе схемы

Таблица 2.16

m/n	Тип тестера	Характеристика			
		L	N_t	r_1	t
2/5	СПТ4	88	32	6	10
	СПТ5	35	17	7	7
	СПТ6	30	15	6	6
	СПТ7	39	20	7	8
	СПТ8	34	17	6	5
3/8	СПТ4	487	97	6	56
	СПТ5	96	40	7	20
	СПТ6	62	30	7	10
	СПТ7	77	34	7	17
	СПТ8	68	34	8	8
4/8	СПТ4	95	33	4	16
	СПТ5	95	33	4	16
	СПТ6	57	27	5	9
	СПТ7	87	37	7	16
	СПТ8	60	30	7	8
5/14	СПТ4	12012	2254	6	2002
	СПТ5	930	260	7	166
	СПТ6	164	73	10	21
	СПТ7	304	105	7	76
	СПТ8	184	91	14	14
6/24	СПТ4	830040	136640	6	134600
	СПТ5	17714	3471	7	2966
	СПТ6	369	130	12	40
	СПТ7	916	322	7	208
	СПТ8	444	223	25	24

преобразования $nCm \rightarrow qC1 \rightarrow 4C2 \rightarrow 2C1$. Преобразователь $nCm \rightarrow qC1$ реализуется путем разложения базовой функции (2.2) на q функций $y_1 \div y_q$, которые должны удовлетворять следующим условиям, аналогичным условиям Н1 — Н4.

Н1*. Функции $y_1 \div y_q$ должны представлять собой дизъюнкцию конъюнкций ранга m .

Н2*. Каждая конъюнкция ранга m должна входить в одну и только в одну из функций $y_1 \div y_q$.

Н3*. Каждая конъюнкция ранга $m+1$ должна покрываться хотя бы одной конъюнкцией, включенной в функцию y_i , и хотя бы одной конъюнкцией, включенной в функцию y_j ($i \neq j$; $i, j \in \{1, 2, \dots, q\}$).

Н4*. Каждая конъюнкция ранга $m-1$ должна покрывать хотя бы одну конъюнкцию, включенную в функцию y_i , и хотя бы одну конъюнкцию, включенную в функцию y_j ($i \neq j$).

Метод [65] является развитием метода [48] синтеза $m/2m$ -СПТ и основан на многоэтапном разбиении множества $B = \{x_1, x_2, \dots, x_n\}$ переменных тестера. Обозначим через $]a[$ ($[a]$) наибольшее целое число, меньшее (большее) или равное a . Для синтеза преобразователей $nCm \rightarrow qC1$ разработаны следующие алгоритмы.

Алгоритм 2.1 (случай $2m+2 \leq n \leq 4m$):

1. Множество входных переменных B разбивается на два подмножества, B_1 и B_2 , таких, что $n_1 =]n/2[$ и $n_2 = n - n_1$. Вычисляются функции y_1 и y_2 по формулам:

$$\begin{aligned} y_1 &= \bigvee_{j \in \{1, \dots, m-1\}} F_{n_1}(k_1 \geq j) F_{n_2}(k_2 \geq m - j), \\ y_2 &= \bigvee_{i \in \{1, \dots, m-1\}} F_{n_1}(k_1 \geq i) F_{n_2}(k_2 \geq m - i), \end{aligned} \quad (2.27)$$

где обозначения имеют тот же смысл, что и в формулах (2.7) и (2.8).

2. Множество B_1 разбивается на два подмножества, B_{11} и B_{12} , таких, что $n_{11} =]n_1/2[$ и $n_{12} = n_1 - n_{11}$. Вычисляются функции y_3 и y_4 по формулам:

$$\begin{aligned} y_3 &= \bigvee_{j \in \{m-n_{12}, \dots, n_{11}\}} F_{n_{11}}(k_{11} \geq j) F_{n_{12}}(k_{12} \geq m - j), \\ y_4 &= \bigvee_{i \in \{m-n_{12}, \dots, n_{11}\}} F_{n_{11}}(k_{11} \geq i) F_{n_{12}}(k_{12} \geq m - i). \end{aligned} \quad (2.28)$$

3. Множество B_2 разбивается на два подмножества, B_{21} и B_{22} , таких, что $n_{21} =]n_2/2[$ и $n_{22} = n_2 - n_{21}$. Вычисляются функции y_5 и y_6 по формулам:

$$\begin{aligned} y_5 &= \bigvee_{j \in \{m-n_{22}, \dots, n_{21}\}} F_{n_{21}}(k_{21} \geq j) F_{n_{22}}(k_{22} \geq m - j), \\ y_6 &= \bigvee_{i \in \{m-n_{22}, \dots, n_{21}\}} F_{n_{21}}(k_{21} \geq i) F_{n_{22}}(k_{22} \geq m - i). \end{aligned} \quad (2.29)$$

Пример 2.1. Определим функции, описывающие преобразователь $8C3 \rightarrow qC1$. В этом случае имеем: $n = 8$, $n_1 = n_2 = 4$, $n_{11} = n_{12} = n_{21} = n_{22} = 4$, $B_1 = \{x_1, x_2, x_3, x_4\}$, $B_2 = \{x_5, x_6, x_7, x_8\}$, $B_{11} = \{x_1, x_2\}$, $B_{12} = \{x_3, x_4\}$, $B_{21} = \{x_5, x_6\}$, $B_{22} = \{x_7, x_8\}$. Используя вышеприведенные формулы, получаем:

$$\begin{aligned} y_1 &= (x_1 \vee x_2 \vee x_3 \vee x_4) (x_5 x_6 \vee x_5 x_7 \vee x_5 x_8 \vee x_6 x_7 \vee x_6 x_8 \vee x_7 x_8), \\ y_2 &= (x_1 x_2 \vee x_1 x_3 \vee x_1 x_4 \vee x_2 x_3 \vee x_2 x_4 \vee x_3 x_4) (x_5 \vee x_6 \vee x_7 \vee x_8), \\ y_3 &= (x_1 \vee x_2) x_3 x_4, \quad y_5 = (x_5 \vee x_6) x_7 x_8, \\ y_4 &= x_1 x_2 (x_3 \vee x_4), \quad y_6 = x_5 x_6 (x_7 \vee x_8). \end{aligned}$$

Для реализации 3/8-СПТ преобразователь $8C3 \rightarrow 6C1$, построенный по полученным формулам, соединяется последовательно с преобразователем $6C1 \rightarrow 4C2$ и 2/4-СПТ.

Алгоритм 2.2 (случай $n = 2m + 1$):

1. Множество входных переменных B разбивается на два подмножества B_1 и B_2 , таких, что $n_1 = \lfloor n/2 \rfloor$ и $n_2 = n - n_1$. Вычисляются функции y_1 и y_2 по формулам:

$$\begin{aligned} y_1 &= \bigvee_{j \in \{1, \dots, m\}} F_{n_1}(k_1 \geq j) F_{n_2}(k_2 \geq m - j), \\ y_2 &= \bigvee_{i \in \{1, \dots, m\}} F_{n_1}(k_1 \geq i) F_{n_2}(k_2 \geq m - i). \end{aligned} \quad (2.30)$$

2. Множество B_2 разбивается на два подмножества B_{21} и B_{22} , таких, что $n_{21} = \lfloor n_2/2 \rfloor$ и $n_{22} = n_2 - n_{21}$. Функции y_3 и y_4 вычисляются по формулам (2.29).

Пример 2.2. Определим функции, описывающие преобразователь $7C3 \rightarrow 4C1$. В этом случае имеем: $n = 7$, $n_1 = 3$, $n_2 = 4$, $n_{21} = n_{22} = 2$, $B_1 = \{x_1, x_2, x_3\}$, $B_2 = \{x_1, x_5, x_6, x_7\}$, $B_{21} = \{x_4, x_5\}$, $B_{22} = \{x_6, x_7\}$. Используя формулы (2.30) и (2.29), получаем:

$$\begin{aligned} y_1 &= (x_1 \vee x_2 \vee x_3) (x_4 x_5 \vee x_4 x_6 \vee x_4 x_7 \vee x_5 x_6 \vee x_5 x_7 \vee \\ &\quad \vee x_6 x_7 \vee x_1 x_2 x_3), \\ y_2 &= (x_1 x_2 \vee x_1 x_3 \vee x_2 x_3) (x_4 \vee x_5 \vee x_6 \vee x_7), \\ y_3 &= (x_4 \vee x_5) x_6 x_7, \quad y_4 = x_4 x_5 (x_6 \vee x_7). \end{aligned}$$

Для реализации 3/7-СПТ преобразователь $7C3 \rightarrow 4C1$, построенный по полученным формулам, соединяется последовательно с преобразователем $4C1 \rightarrow 4C2$ и 2/4-СПТ.

Формулы (2.27) — (2.30) обеспечивают построение преобразователей $nCm \rightarrow qC1$ с выполнением условий Н1* — Н4*. Тестеры, реализуемые на основе алгоритма 2.1, требуют для своей проверки t кодовых слов, где

$$t = \sum_{i=1}^{m-1} \max [C_{n_1}^i, C_{n_2}^{m-i}] + \sum_{i=m-n_{12}}^{n_{11}} \max [C_{n_{11}}^i, C_{n_{12}}^{m-i}] +$$

$$+ \sum_{i=m-n_{22}}^{n_{21}} \max [C_{n_{21}}^i, C_{n_{22}}^{m-i}]. \quad (2.31)$$

Число кодовых слов, необходимое для проверки тестеров, реализуемых на основе алгоритма 2.2,

$$t = \sum_{i=1}^m C_{m+1}^{m-i} + \sum_{i=m-n_{22}}^{n_{21}} \max [C_{n_{21}}^i, C_{n_{22}}^{m-i}]. \quad (2.32)$$

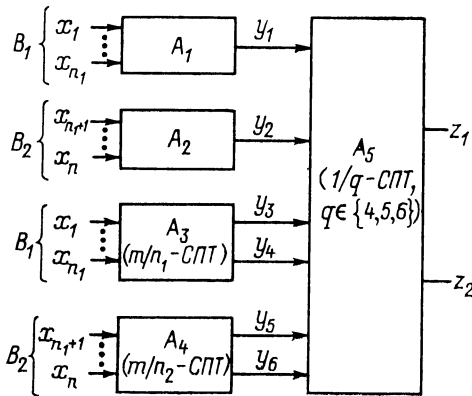


Рис. 2.18

Для случая $n > 4m$ в [65] предложена рекурсивная процедура построения m/n -СПТ в соответствии со структурой, показанной на рис. 2.18. Структура содержит пять блоков. Блоки A_1 и A_2 реализуются по формулам (2.27), получаемым в результате выполнения п. 1 алгоритма 2.1. Блоки A_3 и A_4 представляют собой полные схемы m/n_1 -СПТ и m/n_2 -СПТ, где $n_1 = \lfloor n/2 \rfloor$ и $n_2 = n - n_1$. Они реализуются

либо с помощью алгоритмов 2.1 и 2.2, либо в виде структуры, соответствующей рис. 2.18 (если $n_1 > 4m$ или $n_2 > 4m$).

Пример 2.3. Требуется получить структуру 3/14-СПТ. Множество $B = \{x_1 \div x_{14}\}$. Выполняем п. 1 алгоритма 2.1. Имеем $n = 14$, $n_1 = n_2 = 7$, $B_1 = \{x_1 \div x_7\}$, $B_2 = \{x_8 \div x_{14}\}$,

$$y_1 = (x_1 \vee x_2 \vee \dots \vee x_7) \quad (x_8 x_9 \vee x_8 x_{10} \vee \dots \vee x_{13} x_{14}),$$

$$y_2 = (x_1 x_2 \vee x_1 x_3 \vee \dots \vee x_6 x_7) \quad (x_8 \vee x_9 \vee \dots \vee x_{14}).$$

По полученным формулам синтезируются блоки A_1 и A_2 . Блоки A_3 и A_4 представляют собой 3/7-СПТ. Каждый из них состоит из последовательно включенных преобразователя $7C3 \rightarrow 4C1$, описываемого функциями, полученными в примере 2.2, преобразователя $4C1 \rightarrow 4C2$ и 2/4-СПТ. Схемы последних представлены на рис. 2.17. Блок A_5 представляет собой 1/6-СПТ, синтезированный по соответствующим функциям, приведенным в § 2.4.

В табл. 2.16 тестеры, реализованные методом [65], обозначены как СПТ5. В табл. 2.17 проведено сравнение по сложности различных m/n -СПТ при $n > 4m$.

Дальнейшее развитие метод, основанный на разбиении множества входных переменных тестера, получил в [39], где пред-

ложен универсальный алгоритм, позволяющий синтезировать m/n -СПТ с любым значением n . Тестеры строятся на основе двухкаскадного преобразования по схеме $nCm \rightarrow qC1 \rightarrow 2C1$ и состоят из двух блоков (преобразователя $nCm \rightarrow qC1$ и $1/q$ -СПТ). В качестве $1/q$ -СПТ используются тестеры типа СПТ2 (см. табл. 2.10).

Таблица 2.17

Тип тестера	m/n			
	2/10	3/15	3/24	4/18
	Характеристика L			
СПТ4	345	6678	21394	22154
СПТ5	105	361	1008	1266
СПТ6	57	130	210	199
СПТ7	77	205	582	400
СПТ8	64	146	264	224

Для базовой функции тестера (2.2), которая относится к классу простых симметричных функций [29], имеет место равенство

$$\begin{aligned}
 f^m(x_1, \dots, x_k, \dots, x_n) = & f^m(x_1, \dots, x_k) \vee f^m(x_{k+1}, \dots, x_n) \vee \\
 & \vee f^{m-1}(x_1, \dots, x_k) f^1(x_{k+1}, \dots, x_n) \vee f^{m-2}(x_1, \dots, x_k) \wedge \\
 & \wedge f^2(x_{k+1}, \dots, x_n) \vee \dots \vee f^1(x_1, \dots, x_k) f^{m-1}(x_{k+1}, \dots, x_n).
 \end{aligned}
 \tag{2.33}$$

Заметим, что $f^m(x_{i_1}, x_{i_2}, \dots, x_{i_{m-j}}) = 0$ ($j \geq 1$). Назовем функции вида $f^1(x_{i_1}) = x_{i_1}$, $f^1(x_{i_1}, x_{i_2}) = x_{i_1} \vee x_{i_2}$ и $f^2(x_{i_1}, x_{i_2}) = x_{i_1} x_{i_2}$ элементарными.

Реализация функции (2.2) представляет собой двухуровневую схему, имеющую сложность $L = (m+1)C_n^m$. Применение к ней преобразования (2.33) приводит к замене указанной схемы трехуровневой схемой, которая описывается некоторой скобочной формулой. Следствием этого является уменьшение сложности схемы. Правая часть равенства (2.33) в качестве составляющих содержит функции, также являющиеся простыми симметричными функциями. Поэтому возможно дальнейшее применение преобразования (2.33). При этом в формуле появляется несколько обозначений одной и той же функции вида (2.2), что позволяет перейти к многоуровневой реализации с разветвлениями. Многократное применение преобразования (2.33) приводит к формуле, содержащей обозначения только элементар-

ных функций. Назовем такую формулу и ее схемную реализацию тупиковыми. Очевидно, что тупиковая реализация обладает наименьшей сложностью и наибольшим числом уровней.

Преобразование (2.33) основано на разбиении множества переменных B на два подмножества $B_1 = \{x_1, \dots, x_k\}$ и $B_2 = \{x_{k+1}, \dots, x_n\}$. Вид и сложность тупиковой реализации зависят от выбора числа k на каждом этапе преобразования (2.32). Определим два способа выбора этого числа. В первом случае $k = \lfloor n/2 \rfloor$, во втором $k = \langle n/2 \rangle$, где $\langle n/2 \rangle$ — ближайшее к $n/2$ четное число (если таких чисел два, то выбирается любое из них). Будем обозначать четное и нечетное число n соответственно как $n = \varphi(2)$ и $n = \varphi(1)$. При построении тупиковой реализации осуществляется последовательное разбиение множества переменных B до образования множества элементарных подмножеств, состоящих из одной или двух переменных. Поэтому каждой тупиковой реализации соответствует некоторое множество подмножеств переменных, образуемых на всех этапах преобразования (2.33). Будем называть такое множество базовым множеством для функции вида (2.2) и соответственно для m/n -СПТ.

Пример 2.4. Вычислим тупиковую формулу для функции $f^3(x_1 \div x_7)$. Для этого определим базовое множество V . На первом этапе разбиения множества $B = \{x_1 \div x_7\}$ получаем $B_1^1 = \{x_1, x_2, x_3, x_4\}$ и $B_2^1 = \{x_5, x_6, x_7\}$ (верхний индекс в обозначениях указывает номер этапа разбиения). На втором этапе разбиения имеем: $B_1^2 = \{x_1, x_2\}$, $B_2^2 = \{x_3, x_4\}$, $B_3^2 = \{x_5, x_6\}$, $B_4^2 = \{x_7\}$. Используя (2.33), получаем

$$\begin{aligned} f^3(x_1 \div x_7) &= f^3(x_1 \div x_4) \vee f^3(x_5 \div x_7) \vee f^2(x_1 \div x_4) \wedge \\ &\wedge f^1(x_5 \div x_7) \vee f^1(x_1 \div x_4) f^2(x_5 \div x_7) = f^2(x_1, x_2) \wedge \\ &\wedge f^1(x_3, x_4) \vee f^1(x_1, x_2) f^2(x_3, x_4) \vee f^2(x_5, x_6) f^1(x_7) \vee \\ &\vee [f^2(x_1, x_2) \vee f^2(x_3, x_4) \vee f^1(x_1, x_2) f^1(x_3, x_4)] \times \\ &\times [f^1(x_5, x_6) \vee f^1(x_7)] \vee [f^1(x_1, x_2) \vee f^1(x_3, x_4)] \times \\ &\times [f^2(x_5, x_6) \vee f^1(x_5, x_6) f^1(x_7)]. \end{aligned}$$

Сложность тупиковой реализации $L = 41$, в то время как сложность двухуровневой реализации данной функции $L = 140$.

Преобразование $nCm \rightarrow qC1$ строится на основе представления базовой функции в виде

$$f^m(x_1, \dots, x_k, \dots, x_n) = y_1 \vee y_2 \vee \dots \vee y_q. \quad (2.34)$$

С использованием составляющих правой части равенства (2.33) составим функции F_1 , F_2 и F_3 с помощью следующих правил (для случая $m \geq 3$):

$$f^m(x_1, \dots, x_k, \dots, x_n) = F_1 \vee F_2 \vee F_3, \quad (2.35)$$

$$F_1 = \begin{cases} 0, \text{ если } k = n-k = m, \\ f^m(x_1, \dots, x_k), \text{ если } k \neq m \text{ и } n-k = m, \\ f^m(x_{k+1}, \dots, x_n), \text{ если } k = m \text{ и } n-k \neq m, \\ f^m(x_1, \dots, x_k) \vee f^m(x_{k+1}, \dots, x_n), \text{ если } k \neq m \text{ и } n-k \neq m; \end{cases} \quad (2.36)$$

$$F_2 = F_2' \vee f^{m-1}(x_1, \dots, x_k) f^1(x_{k+1}, \dots, x_n) \vee f^{m-3}(x_1, \dots, x_k) \wedge \wedge f^3(x_{k+1}, \dots, x_n) \vee \dots \vee f^s(x_1, \dots, x_k) f^{m-s}(x_{k+1}, \dots, x_n); \quad (2.37)$$

$$F_3 = F_3' \vee f^{m-2}(x_1, \dots, x_k) f^2(x_{k+1}, \dots, x_n) \vee f^{m-4}(x_1, \dots, x_k) \wedge \wedge f^4(x_{k+1}, \dots, x_n) \vee \dots \vee f^t(x_1, \dots, x_k) f^{m-t}(x_{k+1}, \dots, x_n), \quad (2.38)$$

где $s = 1$ и $t = 2$ при $m = \varphi(2)$; $s = 2$ и $t = 1$ при $m = \varphi(1)$;

$$F_2' = \begin{cases} 0, \text{ если } n-k \neq m \text{ и } m = \varphi(2), \\ f^m(x_{k+1}, \dots, x_n), \text{ если } n-k = m \text{ и } m = \varphi(1); \end{cases}$$

$$F_3' = \begin{cases} 0, \text{ если } k \neq m \text{ и } n-k \neq m \text{ или } k \neq m \text{ и } m = \varphi(1), \\ f^m(x_1, \dots, x_k), \text{ если } k = m \text{ и } n-k \neq m \text{ или} \\ k = m \text{ и } m = \varphi(1), \\ f^m(x_1, \dots, x_k) \vee f^m(x_{k+1}, \dots, x_n), \text{ если} \\ k = n-k = m \text{ и } m = \varphi(2). \end{cases}$$

Для случая $m = 2$ используются выражения (2.35)—(2.37), а также следующие соотношения:

$$F_3 = \begin{cases} 0, \text{ если } F_1 \neq 0, \\ f^2(x_1, x_2) \vee f^2(x_3, x_4) \vee \dots \vee f^2(x_5, x_t), \\ \text{если } F_1 = 0, \end{cases} \quad (2.39)$$

где $s = n-1$ и $t = n$ при $n = \varphi(2)$; $s = n-2$ и $t = n-1$ при $n = \varphi(1)$.

Разложение базовой функции в соответствии с формулами (2.35)—(2.39) назовем разложением типа F . В [39] показано, что функции F_2 и F_3 могут быть непосредственно использованы в качестве функций y_i в разложении (2.34). К функциям $f^m(x_1, \dots, x_k)$ и $f^m(x_{k+1}, \dots, x_n)$, входящим в F_1 (возможны случаи, когда в F_1 входит только одна из них), необходимо снова применить разложение типа F . При этом образуются новые функции вида F_1 , F_2 и F_3 , в состав которых включаются соответствующие составляющие разложения как функции $f^m(x_1, \dots, x_k)$, так и функции $f^m(x_{k+1}, \dots, x_n)$. После каждого применения разложения типа F возможны два случая:

1. $F_1 = 0$; при этом процесс разложения базовой функции заканчивается и все полученные в процессе разложения функции вида (2.37) и (2.38) образуют множество функций вида y_1, y_2, \dots, y_q ;

2. $F_1 \neq 0$; при этом к функции F_1 снова применяется разложение типа F .

Таким образом, процесс получения функций y_1, y_2, \dots, y_q , описывающих преобразователь $nCm \rightarrow qC1$, состоит в последовательном применении разложения типа F к базовой функции m/n -СПТ до тех пор, пока не будет $F_1 = 0$. При этом обеспечивается выполнение условий Н1* — Н4*. Наиболее простая структура тестера получается при разложении базовой функции (2.34) с наименьшим числом q . С этой точки зрения наилучший результат дает первый вид разбиения множества B (при котором $k = [n/2]$), используемый при образовании базового множества V . Однако с точки зрения простоты реализации конкретной функции вида (2.2) лучший результат дает второй вид разбиения (при котором $k = \langle n/2 \rangle$). По этой причине при разложении типа F целесообразно использовать второй вид разбиения, кроме тех случаев, когда это приводит к увеличению числа q . Последнее вычисляется с помощью решения следующей системы:

$$\begin{aligned} 2^{p-1}m < n < 2^p m, \\ q = \begin{cases} 2p & \text{при } m \geq 3, \\ p+1 & \text{при } m = 2. \end{cases} \end{aligned} \quad (2.40)$$

Пример 2.5. Определим функции, описывающие преобразователь $18C4 \rightarrow qC1$. Так как $2^2 \times 4 < 18 < 2^3 \times 4$, то $q = 6$. Применяем разложение типа F к базовой функции $f^m(x_1 \div x_{18})$, используя второй вид разбиения множества входных переменных (функциям F_1, F_2 и F_3 будем присваивать верхний индекс, указывающий число применений преобразования типа F):

$$\begin{aligned} F_1^1 &= f^4(x_1 \div x_{10}) \vee f^4(x_{11} \div x_{18}), \\ F_2^1 &= f^3(x_1 \div x_{10}) f^1(x_{11} \div x_{18}) \vee f^1(x_1 \div x_{10}) f^3(x_{11} \div x_{18}), \\ F_3^1 &= f^2(x_1 \div x_{10}) f^2(x_{11} \div x_{18}). \end{aligned}$$

Так как $F_1^1 \neq 0$, то применяем к ней разложение типа F :

$$\begin{aligned} F_1^2 &= f^4(x_1 \div x_6), \\ F_2^2 &= f^3(x_1 \div x_6) f^1(x_7 \div x_{10}) \vee f^1(x_1 \div x_6) f^3(x_7 \div x_{10}) \vee \\ &\vee f^3(x_{11} \div x_{14}) f^1(x_{15} \div x_{18}) \vee f^1(x_{11} \div x_{14}) f^3(x_{15} \div x_{18}), \\ F_3^2 &= f^4(x_7 \div x_{10}) \vee f^2(x_1 \div x_6) f^2(x_7 \div x_{10}) \vee f^2(x_{11} \div x_{14}) \times \\ &\times f^2(x_{15} \div x_{18}) \vee f^4(x_{11} \div x_{14}) \vee f^4(x_{15} \div x_{16}). \end{aligned}$$

Так как $F_1^2 \neq 0$, то снова применяем разложение типа F :

$$\begin{aligned} F_1^3 &= 0, \\ F_2^3 &= f^5(x_1 \div x_4) f^1(x_5, x_6), \\ F_3^3 &= f^2(x_1 \div x_4) f^2(x_5, x_6) \vee f^4(x_1 \div x_4). \end{aligned}$$

Так как $F_1^3 = 0$, то процесс разложения базовой функции заканчивается. В результате получены шесть функций: $y_1 = F_2^1$, $y_2 = F_3^1$, $y_3 = F_2^2$, $y_4 = F_3^2$, $y_5 = F_2^3$, $y_6 = F_3^3$, которые списывают преобразователь $18C4 \rightarrow 6C1$.

Сформулируем алгоритм синтеза m/n -СПТ.

Алгоритм 2.3:

1. Записывается базовая функция m/n -СПТ.
2. С помощью решения системы уравнений (2.40) определяется число q , характеризующее преобразователь $nCm \rightarrow qC1$.
3. Для базовой функции тестера находится базовое множество V , соответствующее одной из тупиковых реализаций функции и полученному в предыдущем пункте числу q .
4. Определяются функции y_1, y_2, \dots, y_q , которые описывают преобразователь $nCm \rightarrow qC1$ и соответствуют полученному базовому множеству V .
5. Каждая из функций y_1, y_2, \dots, y_q преобразуется с целью получения тупиковой формулы.
6. Строится преобразователь $nCm \rightarrow qC1$ путем реализации функций y_1, y_2, \dots, y_q в виде связанных многоуровневых схем с разветвлениями; при этом каждая функция вида (2.2) реализуется в соответствии с выражением (2.33).
7. Выходы преобразователя $nCm \rightarrow qC1$ произвольным образом соединяются со входами $1/q$ -СПТ, синтезированных методом [37].

Пример 2.6. Синтезируем $3/8$ -СПТ. Базовая функция имеет вид $f^3(x_1 \div x_8)$. С помощью системы (2.40) находим число q . Так как $2^1 \times 3 < 8 < 2^2 \times 3$, $q = 4$. Строим базовое множество V :

$$B = \{x_1 \div x_8\}, B_1^1 = \{x_1 \div x_4\}, B_2^1 = \{x_5 \div x_8\}, B_1^2 = \{x_1, x_2\}, \\ B_2^2 = \{x_3, x_4\}, B_3^2 = \{x_5, x_6\}, B_4^2 = \{x_7, x_8\}.$$

В результате применения к базовой функции разложения типа F получаем:

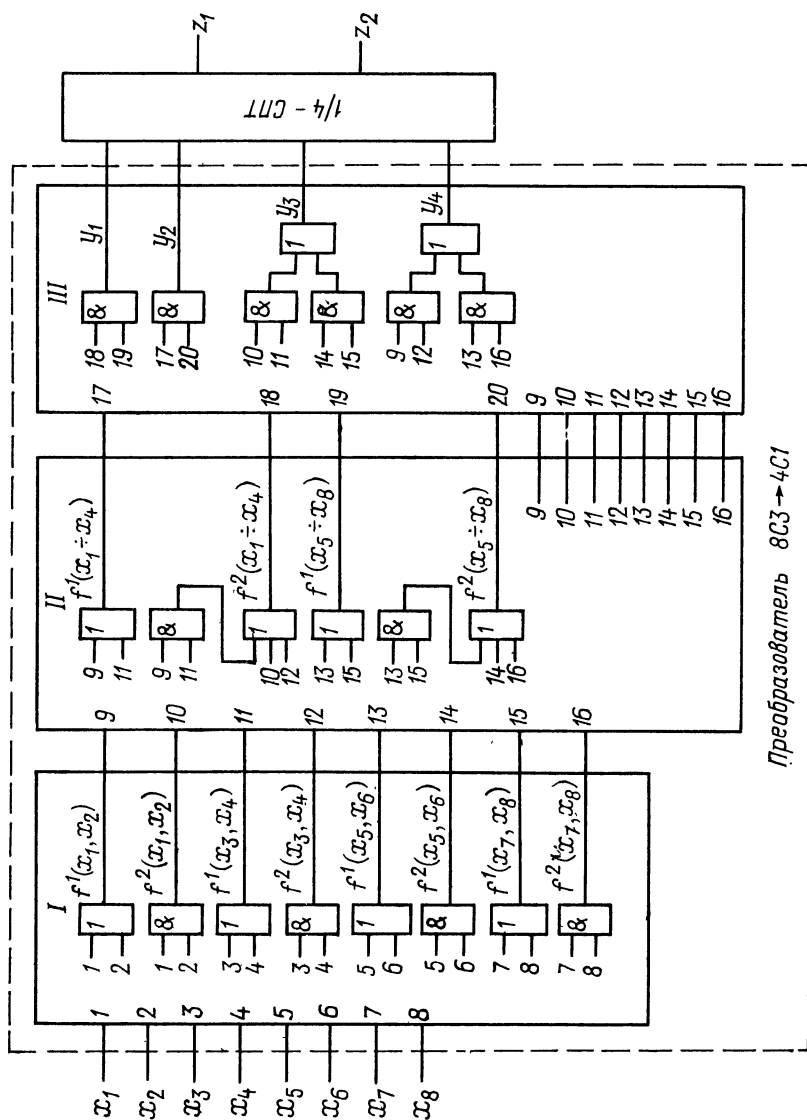
$$y_1 = f^2(x_1 \div x_4) f^1(x_5 \div x_8), \quad y_2 = f^1(x_1 \div x_4) f^2(x_5 \div x_8), \\ y_3 = f^2(x_1, x_2) f^1(x_3, x_4) \vee f^2(x_5, x_6) f^1(x_7, x_8), \\ y_4 = f^1(x_1, x_2) f^2(x_3, x_4) \vee f^1(x_5, x_6) f^2(x_7, x_8).$$

Функции y_3 и y_4 записаны в виде тупиковых формул. Представим в виде тупиковых формул функции y_1 и y_2 :

$$y_1 = [f^2(x_1, x_2) \vee f^2(x_3, x_4) \vee f^1(x_1, x_2) f^1(x_3, x_4)] \wedge \\ \wedge [f^1(x_5, x_6) \vee f^1(x_7, x_8)], \\ y_2 = [f^1(x_1, x_2) \vee f^1(x_3, x_4)] [f^2(x_5, x_6) \vee f^2(x_7, x_8) \vee \\ \vee f^1(x_5, x_6) f^1(x_7, x_8)].$$

Схемная реализация $3/8$ -СПТ представлена на рис. 2.19.

Свойства тестеров полностью определяются базовым множеством V и системой функций y_1, y_2, \dots, y_q . Это позволяет



Преобразователь 8C3 → 4C1

Рис. 2.19

вычислять характеристики тестера без построения его схемы и ее анализа.

Представим сложность тестера L в виде

$$L = L_1 + L_2, \quad (2.41)$$

где L_1 — сложность преобразователя $nCm \rightarrow qC1$, $L_1 = L_1' + L_1''$; L_2 — сложность $1/q$ -СПТ; L_1' — сложность блока, реализующего систему функций вида $f^a(x_{i_1}, \dots, x_{i_p})$ ($a \in \{1, 2, \dots, m\}$); $i_1, i_2, \dots, i_p \in \{1, 2, \dots, n\}$, необходимых для представления функций y_1, y_2, \dots, y_q (блоки I и II на рис. 2.19); L_1'' — сложность блока преобразователя $nCm \rightarrow qC1$, реализующего функции y_1, y_2, \dots, y_q в том виде, в каком они получены в результате применения к базовой функции разложения типа F (блок III на рис. 2.19).

Схемы, реализующие функции $f^a(x_{i_1}, \dots, x_{i_p})$, имеют стандартную структуру, которая полностью определяется значениями чисел d и p . С другой стороны, необходимое число таких схем однозначно определяется базовым множеством V . Это дает возможность вычислять характеристику L_1' с помощью следующей формулы:

$$\begin{aligned} L' = & 4S_2 + S_3(2 + 4a_2^3 + 2a_3^3) + \sum_{t \in \{4, 6, 8, 10, \dots\}} S_t \{2 + a_2^t p_2 + \\ & + a_3^t p_3 + \dots + a_{t/2-1}^t p_{t/2-1} + a_{t/2}^t [b_1^t p_{t/2} + b_2^t (p_{t/2} - 1)] + \\ & + a_{t/2+1}^t [b_1^t (p_{t/2} + 1) + b_2^t (p_{t/2} - 1)] + a_{t/2+2}^t p_{t/2+2} + \\ & + a_{t/2+3}^t p_{t/2+3} + \dots + a_{t-1}^t p_{t-1} + 2a_t^t\} + \\ & + \sum_{t \in \{5, 7, 9, 11, \dots\}} S_t [2 + a_2^t g_2 + a_3^t g_3 + \dots + a_{t/2-1/2}^t g_{t/2-1/2} + \\ & + a_{t/2+1/2}^t (g_{t/2-1/2} + 2) + a_{t/2+3/2}^t (g_{t/2-1/2} + 1) + \\ & + a_{t/2+5/2}^t g_{t/2+5/2} + a_{t/2+7/2}^t g_{t/2+7/2} + \dots + a_{t-1}^t g_{t-1} + 2a_t^t], \quad (2.42) \end{aligned}$$

где S_l ($l \in \{2, 3, 4, 5, \dots, t, \dots\}$) — число подмножеств переменных с l элементами, входящих в базовое множество тестера V ; $a_h^r = 1$ ($r \in \{3, 4, 5, 6, \dots\}$, $h \in \{2, 3, 4, \dots\}$), если $h < m$ и $r \leq h$ или $h = r = m$, в других случаях $a_h^r = 0$; $p_2 = 5$, $p_{i+1} = p_i + 3$ ($i \in \{2, 3, \dots, t/2\}$); $p_{t/2+2} = p_{t/2} - 2$; $p_{j+1} = p_j - 3$ ($j \in \{t/2+2, t/2+3, \dots, t-1\}$); $g_2 = 5$; $g_{k+1} = g_k + 3$ ($k \in \{2, 3, \dots, t/2-1/2\}$); $g_{t/2+5/2} = g_{t/2-1/2} + 2$; $g_{z+1} = g_z - 3$ ($z \in \{t/2+5/2, t/2+7/2, \dots, t-1\}$); $b_1^t = 1$ и $b_2^t = 0$, если $t/2 = \varphi(1)$ или $t/2 = \varphi(2)$ и к рассматриваемому подмножеству применено разбиение первого вида, в остальных случаях $b_1^t = 0$ и $b_2^t = 1$.

Характеристика L_1'' вычисляется непосредственно по системе функций y_1, y_2, \dots, y_q . Представим эти функции в следующем виде ($i \in \{1, 2, \dots, q\}$):

$$y_i = \bigvee_{j \in Q_i} y_j^*, \quad (2.43)$$

где y_j^* — функция вида $f^m(x_{t_1}, \dots, x_{t_m})$ или $f^{m-d}(x_{t_1}, \dots, x_{t_k}) \times \times f^d(x_{t_{k+1}}, \dots, x_{t_h})$; Q_i — множество номеров дизъюнктивных членов правой части (2.43), имеющее мощность $\mu(Q_i)$.

Аналогично функцию $f^d(x_{t_1}, \dots, x_{t_h})$, представленную в соответствии с преобразованием (2.33), запишем в виде

$$f^d(x_{t_1}, \dots, x_{t_h}) = \bigvee_{j \in R} f_j^*, \quad (2.44)$$

где f_j^* — функция вида $f^d(x_{t_1}, \dots, x_{t_k})$ или $f^d(x_{t_{k+1}}, \dots, x_{t_h})$ или $f^{d-p}(x_{t_1}, \dots, x_{t_k}) f^p(x_{t_{k+1}}, \dots, x_{t_h})$, R — множество индексов дизъюнктивных членов правой части (2.44).

Введем следующие обозначения: $\mu[f]$ — число дизъюнктивных членов функции f , представляющей собой общее дизъюнктивное выражение;

$$\mu[y_j^*] = \begin{cases} 1, & \text{если } y_j^* = f^m(x_{t_1}, \dots, x_{t_m}), \\ \max\{\mu[f^{m-d}(x_{t_1}, \dots, x_{t_k})], \mu[f^d(x_{t_{k+1}}, \dots, x_{t_h})]\}, & \\ \text{если } y_j^* = f^{m-d}(x_{t_1}, \dots, x_{t_k}) f^d(x_{t_{k+1}}, \dots, x_{t_h}), & \end{cases}$$

$$\mu[y_i] = \sum_{j \in Q_i} \mu[y_j^*].$$

Длина проверяющего теста m/n -СПТ определяется по следующей формуле:

$$t = \mu[y_1] + \mu[y_2] + \mu[Q_3] + \mu[Q_4] + \dots + \mu[Q_q] + \\ + \sum_S \sum_{d \in \{1, 2, \dots, m-2\}} \{\mu[f^{m-d}(x_{t_1}, x_{t_2}, \dots, x_{t_h})] - (d+1)\}, \quad (2.45)$$

где сумма Σ берется по тем входящим в базовое множество V подмножествам $\{x_{t_1}, x_{t_2}, \dots, x_{t_h}\}$, которые получены на втором и дальнейших этапах преобразования базовой функции и для которых $h > m$.

Число уровней r схемы m/n -СПТ определяется на основании решения следующей системы уравнений:

$$2^{b-1} < n \leq 2^b, \quad (2.46)$$

$$r = \begin{cases} p+l+3+r & (1/q\text{-СПТ}) \text{ при } p > l, \\ 2p+3+r & (1/q\text{-СПТ}) \text{ при } p \leq l, \end{cases}$$

где r ($1/q$ -СПТ) — число уровней $1/q$ -СПТ, входящего в структуру рассматриваемого тестера;

для $m = 2$ и $n \leq 6$ $r = b+r$ ($1/q$ -СПТ),
 для $m = 2$ и $n \geq 7$ $r = b+1+r$ ($1/q$ -СПТ),
 для $m = 3$ $r = b+1+r$ ($1/q$ -СПТ),
 для $m \geq 4$ $p = b-2$, $l = m-2$.

Пример 2.7. Определим характеристики 6/24-СПТ. С помощью системы (2.40) находим число q . Так как $2^1 \times 6 \leq 24 \leq 2^2 \times 6$, то $q = 4$. Строим базовое множество V : $B = \{x_1 \div x_{24}\}$, 1-й этап разбиения множества B :

$$B_1 = \{x_1 \div x_{12}\}, \quad B_2^1 = \{x_{13} \div x_{24}\};$$

2-й этап:

$$B_1^2 = \{x_1 \div x_6\}, \quad B_2^2 = \{x_7 \div x_{12}\}, \quad B_3^2 = \{x_{13} \div x_{18}\}, \quad B_4^2 = \{x_{19} \div x_{24}\};$$

3-й этап:

$$B_1^3 = \{x_1 \div x_4\}, \quad B_2^3 = \{x_5, x_6\}, \quad B_3^3 = \{x_7 \div x_{10}\}, \quad B_4^3 = \{x_{11}, x_{12}\}, \\ B_5^3 = \{x_{13} \div x_{16}\}, \quad B_6^3 = \{x_{17}, x_{18}\}, \quad B_7^3 = \{x_{19} \div x_{22}\}, \quad B_8^3 = \{x_{23}, x_{24}\},$$

4-й этап:

$$B_1^4 = \{x_1, x_2\}, \quad B_2^4 = \{x_3, x_4\}, \quad B_3^4 = \{x_7, x_8\}, \quad B_4^4 = \{x_9, x_{10}\}, \\ B_5^4 = \{x_{13}, x_{14}\}, \quad B_6^4 = \{x_{15}, x_{16}\}, \quad B_7^4 = \{x_{19}, x_{20}\}, \quad B_8^4 = \{x_{21}, x_{22}\}.$$

В результате применения к базовой функции $f^6(x_1 \div x_{24})$ разложения типа F получаем

$$y_1 = f^5(x_1 \div x_{12}) f^1(x_{13} \div x_{24}) \vee f^3(x_1 \div x_{12}) f^3(x_{13} \div x_{24}) \vee \\ \vee f^1(x_1 \div x_{12}) f^5(x_{13} \div x_{24}), \\ y_2 = f^4(x_1 \div x_{12}) f^2(x_{13} \div x_{24}) \vee f^2(x_1 \div x_{12}) f^4(x_{13} \div x_{24}), \\ y_3 = f^5(x_1 \div x_6) f^1(x_7 \div x_{12}) \vee f^3(x_1 \div x_6) f^3(x_7 \div x_{12}) \vee \\ \vee f^1(x_1 \div x_6) f^5(x_7 \div x_{12}) \vee f^5(x_{13} \div x_{18}) f^1(x_{19} \div x_{24}) \vee \\ \vee f^3(x_{13} \div x_{18}) f^3(x_{19} \div x_{24}) \vee f^1(x_{13} \div x_{18}) f^5(x_{19} \div x_{24}), \\ y_4 = f^6(x_1 \div x_6) \vee f^6(x_7 \div x_{12}) \vee f^4(x_1 \div x_6) f^2(x_7 \div x_{12}) \vee \\ \vee f^2(x_1 \div x_6) f^4(x_7 \div x_{12}) \vee f^6(x_{13} \div x_{18}) \vee f^6(x_{19} \div x_{24}) \vee \\ \vee f^4(x_{13} \div x_{18}) f^2(x_{19} \div x_{24}) \vee f^2(x_{13} \div x_{18}) f^4(x_{19} \div x_{24}).$$

Определяем сложность тестера. Так как $q = 4$, то $L_2 = L(1/4\text{-СПТ}) = 16$ (см. табл. 2.10). Значение L_1'' определяется суммарной сложностью реализации функций y_1, y_2, y_3 и y_4 , т. е. $L'' = L(y_1) + L(y_2) + L(y_3) + L(y_4) = 9 + 6 + 18 + 16 = 49$. Базовое множество V включает в себя подмножества, содержащие 2, 4, 6 и 12 элементов, причем $S_2 = 12, S_4 = 4, S_6 = 4, S_{12} = 2$. Поэтому согласно (2.42) имеем $L_1' = 4S_2 + S_4(2 + 5 + 6 + 2) + S_6(2 + 5 + 7 + 7 + 6 + 2) + S_{12}(2 + 5 + 8 + 11 + 14) = 304$. В результате получаем $L = L_1' + L_1'' + L_2 = 304 + 49 + 16 = 369$.

Вычисляем длину проверяющего теста. В соответствии с (2.45) имеем $t = \mu[y_1] + \mu[y_2] + \mu(Q_3) + \mu(Q_4)$. При этом $\Sigma = 0$, так как на втором и дальнейших этапах разбиения множества B образованы подмножества с числом элементов, равным m и менее. Определяем

$$\begin{aligned} \mu[y_1] = & \max \{ \mu[f^5(x_1 \div x_{12})], \mu[f^1(x_{13} \div x_{24})] \} + \\ & + \max \{ \mu[f^3(x_1 \div x_{12})], \mu[f^3(x_{13} \div x_{24})] \} + \max \{ \mu[f^1(x_1 \div x_{12})], \\ & \mu[f^5(x_{13} \div x_{24})] \} = \max \{6, 2\} + \max \{4, 4\} + \max \{2, 6\} = 16. \end{aligned}$$

Аналогично находим, что $\mu[y_2] = 10$, $\mu(Q_3) = 6$ и $\mu(Q_4) = 8$. Тогда $t = 16 + 10 + 6 + 8 = 40$.

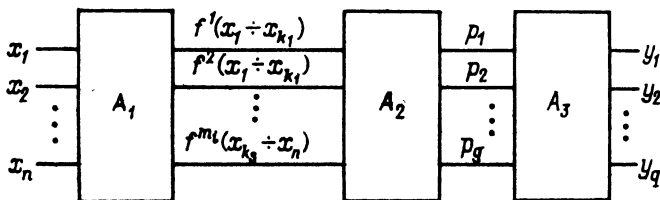


Рис. 2.20

С помощью решения системы (2.46) вычисляем число уровней схемы тестера: $b = 5$, так как $2^{5-1} < 24 < 2^5$; $p = 5 - 2 = 3$; $l = 6 - 2 = 4$; $p < l$; $r(1/4\text{-СПТ}) = 3$; $r = 2p + 3 + r(1/4\text{-СПТ}) = 6 + 3 + 3 = 12$.

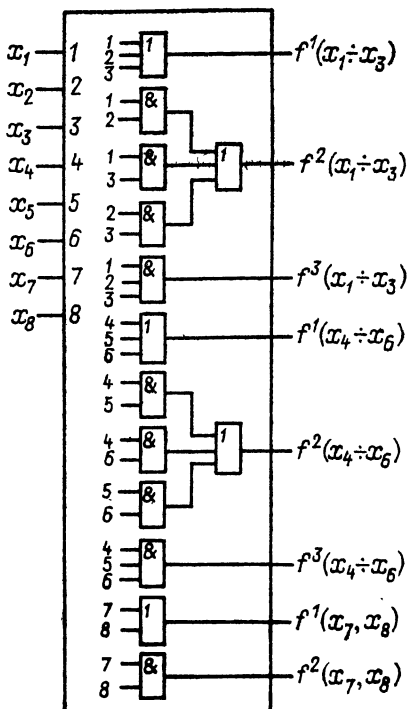


Рис. 2.21

В табл. 2.16 и 2.17 тестеры, синтезированные методом [39] обозначены как СПТ6. В [58] предложен четвертый универсальный метод синтеза m/n -СПТ, также основанный на двухкаскадном преобразовании кодов по схеме $nCm \rightarrow qC1 \rightarrow 2C1$. Преобразователь $nCm \rightarrow qC1$ реализуется в виде структуры (рис. 2.20), состоящей из трех включенных последовательно блоков. При этом множество входных переменных $B = \{x_1, \dots, x_n\}$ разбивается на p подмножеств B_1, B_2, \dots, B_p . На основе этого разбиения осуществляется разложение базовой функции $f^m(x_1, \dots, x_n)$, аналогичное разложению вида (2.33).

Например, для кода 8СЗ имеем: $B = \{x_1 \div x_8\}$, $B_1 = \{x_1 \div x_3\}$, $B_2 = \{x_4 \div x_6\}$, $B_3 = \{x_7, x_8\}$ (при этом $p = 3$),

$$f^3(x_1 \div x_8) = f^3(x_1 \div x_3) \vee$$

$$\vee f^3(x_4 \div x_6) \vee f^2(x_1 \div x_3) \wedge$$

$$\wedge f^1(x_4 \div x_6) \vee f^2(x_1 \div x_3) f^1(x_7, x_8) \vee f^1(x_1 \div x_3) f^2(x_4 \div x_6) \vee$$

$$\begin{aligned} & \vee f^1(x_1 \div x_3) f^2(x_7, x_8) \vee f^2(x_4 \div x_6) f^1(x_7, x_8) \vee f^1(x_4 \div x_6) \wedge \\ & \wedge f^2(x_7, x_8) \vee f^1(x_1 \div x_3) f^1(x_4 \div x_6) f^1(x_7, x_8). \end{aligned} \quad (2.47)$$

Блок A_1 на рис. 2.20 реализует все функции вида $f^i(x_{i_1} \dots x_{i_s})$, входящие в правую часть разложения (2.47), т. е. реализует набор простых симметричных функций. Будем называть такой блок S -блоком. В структурах СПТ целесообразно применение S -блоков трех типов. В блоках первого типа предусматривается реализация каждой симметричной функции в виде отдельной одно- или двухуровневой схемы по ее дизъюнктивной форме. Такой блок имеет максимальное быстродействие. Для случая кода 8СЗ S -блок данного типа показан на рис. 2.21.

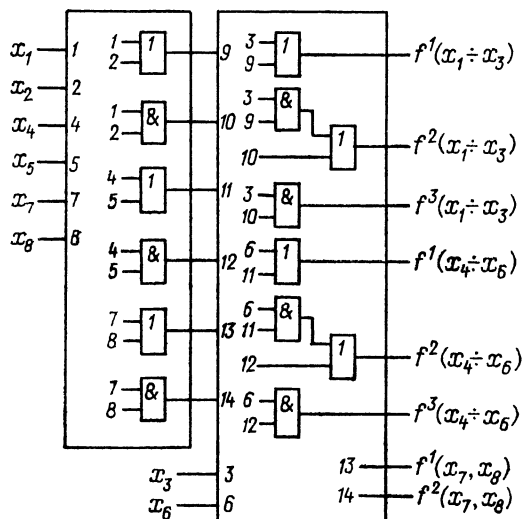


Рис. 2.22

В S -блоках второго типа симметричные функции реализуются по тупиковым формулам. Такой блок имеет минимальную сложность. Рассмотренные выше тестеры СПТ6 также фактически содержат S -блоки. Например, в приведенной на рис. 2.19 схеме 3/8-СПТ S -блок образован блоками I и II . Он реализует все симметричные функции, входящие в разложение (2.33) базовой функции $f^3(x_1 \div x_8)$. S -блок для разложения (2.47) показан на рис. 2.22.

В S -блоках третьего типа симметричные функции реализуются с помощью итеративной схемы, описанной в [76]. Схема позволяет реализовать множество всех функций $f^m(x_1, x_2, \dots, x_k)$ ($m \in \{1, 2, \dots, k\}$) и требует для полного тестирования $2k$ входных наборов. Множество последних имеет вид

$$\begin{array}{r}
 000 \dots 000 \\
 000 \dots 001 \\
 000 \dots 011 \\
 \dots \dots \dots \\
 T = 111 \dots 111 \\
 111 \dots 110 \\
 \dots \dots \dots \\
 110 \dots 000 \\
 100 \dots 000
 \end{array}$$

Итеративная схема строится с помощью двухвходовых элементов И и ИЛИ. При этом требуется $k(k-1)$ элементов, т. е. $L = 2k(k-1)$. Следующие формулы позволяют вычислять число уровней итеративной схемы для каждого ее выхода: для функций f^1 и f^r $r = k-1$, для функций f^i ($i \in \{2, 3, \dots, k-1\}$) $r = k+i-2$. Максимальное число уровней (для выхода f^{k-1}) $r_{\max} = 2k-3$. Простой принцип соединения элементов в итера-

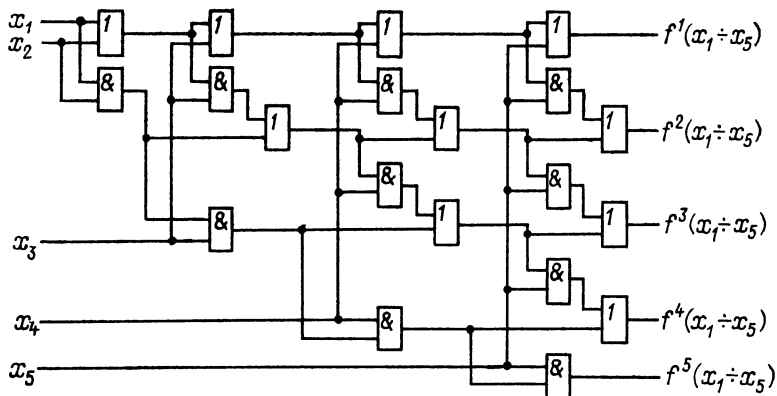


Рис. 2.23

тивной схеме показан на рис. 2.23 для случая реализации системы функций $f^m(x_1 \div x_5)$ ($m \in \{1, 2, \dots, 5\}$). На рис. 2.24 показана итеративная схема S-блока для разложения (2.47).

Обозначим дизъюнктивные элементы правой части разложения (2.47) как $P_j = P_{i_1 i_2 \dots i_p}$ ($j \in \{1, 2, \dots, g\}$; $i_1, i_2, \dots, i_p \in \{1, 2, \dots, m\}$), где индекс i_s ($s \in \{1, 2, \dots, p\}$) соответствует верхнему индексу функции $f^{i_s}(x_{r_1} \div x_{r_d})$, входящей в дизъюнктивный элемент. Например, $f^3(x_1 \div x_8) = f^3(x_1 \div x_3) f^0(x_4 \div x_6) \times f^0(x_7, x_8) = P_{300}$; $f^2(x_1 \div x_3) \wedge f^1(x_4 \div x_6) = P_{210}$; $f^1(x_1 \div x_3) \times f^1(x_4 \div x_6) f^1(x_7, x_8) = P_{111}$ и т. д. Тогда имеем:

$$\begin{aligned}
 f^3(x_1 \div x_8) = & P_{300} \vee P_{030} \vee P_{210} \vee P_{201} \vee P_{120} \vee \\
 & \vee P_{102} \vee P_{021} \vee P_{012} \vee P_{111}.
 \end{aligned} \quad (2.48)$$

Блок A_2 на рис. 2.20 реализует все функции вида $P_{i_1 \dots i_p}$, входящие в разложение (2.48), и представляет собой одноуровневую схему, состоящую из независимых элементов И. Последовательным соединением блоков A_1 и A_2 составляется преобразователь $nCm \rightarrow gC1$, где g — число дизъюнктивных элементов в выражении (2.48). В [58] показано, что для любого кода nCm ($n \geq 4$, $m \geq 2$) преобразователь $nCm \rightarrow gC1$ обладает свойствами контроля входного вектора и самопроверки, если и только если для каждого $i \in \{1, 2, \dots, p\}$

$$n_i \leq m \leq n - n_i, \quad (2.49)$$

где n_i — число элементов множества B_i .

Из условия (2.49) вытекает неравенство, определяющее минимальное число p блоков разбиения множества переменных B :

$$n \leq pm \leq (p-1)n. \quad (2.50)$$

Для $n \leq 4m$ $p \in \{2, 3, 4\}$, для $n > 4m$ $p > 4$.

Блок A_3 на рис. 2.20 реализует функции y_1, y_2, \dots, y_q , описывающие преобразователь $nCm \rightarrow qC1$. Эти функции образуются как дизъюнкции функций $P_{i_1 i_2 \dots i_p}$, входящих в разложение (2.48). Они должны удовлетворять условиям Н1* — Н4*, но так как преобразователь $nCm \rightarrow gC1$, к выходам которого подключается блок A_3 , обладает свойствами контроля входного вектора и самопроверки, то указанные условия могут быть заменены более простыми.

Рассмотрим разложения типа (2.48) для базовых функций кодов $nC(m+1)$ и $nC(m-1)$ при том же варианте разбиения множества B , что и для кода nCm . При этом для обозначения элементов разложения будем использовать соответственно буквы $E_{i_1 i_2 \dots i_p}$ и $R_{i_1 i_2 \dots i_p}$. Например, для рассматриваемого случая имеем:

$$f^4(x_1 \div x_8) = E_{310} \vee E_{301} \vee E_{031} \vee E_{130} \vee E_{211} \vee$$

$$\vee E_{121} \vee E_{112} \vee E_{220} \vee E_{202} \vee E_{022},$$

$$f^2(x_1 \div x_8) = R_{200} \vee R_{020} \vee R_{002} \vee R_{110} \vee R_{101} \vee R_{011}.$$

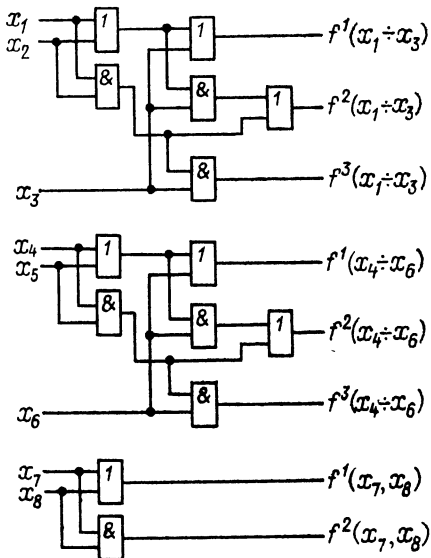


Рис. 2.24

Каждая функция $E_{i_1 i_2 \dots i_p}$ покрывается несколькими функциями $P_{i_1 i_2 \dots i_p}$, а каждая функция $R_{i_1 i_2 \dots i_p}$ покрывает несколько функций $P_{i_1 i_2 \dots i_p}$. Например, функция E_{310} покрывается функциями P_{300} и P_{210} ; функция R_{200} покрывает функции P_{210} и P_{201} и т. д. Условия Н1* — Н4* заменяются аналогичными условиями, в формулировках которых конъюнкция ранга m заменяется функцией $P_{i_1 i_2 \dots i_p}$, конъюнкция ранга $m+1$ — функцией $E_{i_1 i_2 \dots i_p}$, а конъюнкция ранга $m-1$ — функцией $R_{i_1 i_2 \dots i_p}$.

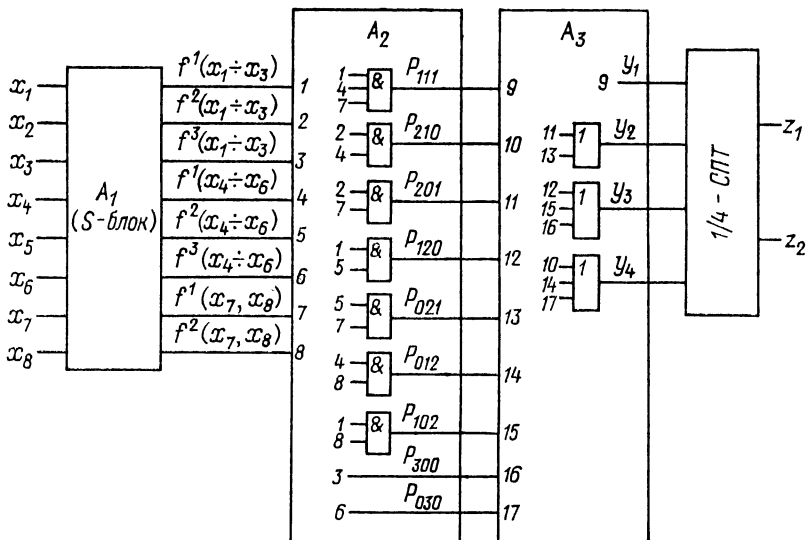


Рис. 2.25

В рассматриваемом примере данные условия выполняются для следующей системы функций:

$$y_1 = P_{111}, \quad y_3 = P_{120} \vee P_{102} \vee P_{300},$$

$$y_2 = P_{201} \vee P_{021}, \quad y_4 = P_{210} \vee P_{012} \vee P_{030}.$$

На рис. 2.25 представлена полная схема 3/8-СПТ.

В [58] получено простое условие построения функций $y_1 \div y_q$. Пусть имеется две функции $P_{i_1 i_2 \dots i_p}$ и $P_{i_1^* i_2^* \dots i_p^*}$. Определим алгебраическую сумму

$$d = \sum_{k=1}^p |i_k - i_k^*|.$$

Условия построения функций $y_1 \div y_q$ выполняются, если для каждой пары функций $P_{i_1 i_2 \dots i_p}$, принадлежащей одной функции y_j ($j \in \{1, 2, \dots, q\}$), выполняется условие $d \geq 2$.

Число q зависит от значения p . Если $p \in \{2, 3, 4\}$, то $q \in \{2, 3, 4\}$. Если $p > 4$, то

$$\begin{aligned} q &= 3 \times 2^{t-1} & \text{при } p = 2t+1 = \varphi(1), \\ q &= 2^t & \text{при } p = 2t = \varphi(2). \end{aligned}$$

Характеристики тестеров зависят от способа реализации S -блока и принятого значения p . В табл. 2.16 и 2.17 m/n -СПТ, реализованные данным методом, обозначены как СПТ7. Приведены характеристики для случая двухуровневой реализации S -блока и минимального значения p .

Рассмотрим метод синтеза m/n -СПТ, позволяющий получать тестеры с минимальным множеством тестов ($t = n$) [9]. В методе осуществляется преобразование кодов по схеме $nCm \rightarrow qC1 \rightarrow 2C1$, где $q = k - m + 3$, $k = \lfloor n/2 \rfloor$. Преобразователь $nCm \rightarrow qC1$, описываемый функциями y_1, y_2, \dots, y_q , строится с помощью следующего алгоритма.

Алгоритм 2.4:

1. Множество переменных B делится на два подмножества $B_1 = \{x_1, \dots, x_k\}$ и $B_2 = \{x_{k+1}, \dots, x_n\}$, где $k = \lfloor n/2 \rfloor$.

2. Методом, предложенным в [76], составляются функции

$$\begin{aligned} f^1(x_1, \dots, x_k), f^2(x_1, \dots, x_k), \dots, f^{m-1}(x_1, \dots, x_k), \\ f^1(x_{k+1}, \dots, x_n), f^2(x_{k+1}, \dots, x_n), \dots, f^{m-1}(x_{k+1}, \dots, x_n), \end{aligned}$$

отражающие реализацию их в виде итеративной схемы.

3. Аналогичным образом находятся функции $f^m(x_1, \dots, x_k)$ и $f^m(x_{k+1}, \dots, x_n)$, причем осуществляется их следующее представление:

$$f^m(x_1, \dots, x_k) = \bigvee_{j=1}^t v_j, \quad f^m(x_{k+1}, \dots, x_n) = \bigvee_{j=1}^{t^*} w_j,$$

где t — число дизъюнктивных элементов выражения $f^m(x_1, \dots, x_k)$; t^* — число дизъюнктивных элементов выражения $f^m(x_{k+1}, \dots, x_n)$; $t^* = t$ или $t-1$; $t = q-2$.

4. Определяются функции:

$$\begin{aligned} y_1 &= f^{m-1}(x_1, \dots, x_k) f^1(x_{k+1}, \dots, x_n) \vee f^{m-3}(x_1, \dots, x_k) \wedge \\ &\wedge f^3(x_{k+1}, \dots, x_n) \vee \dots \vee f^s(x_1, \dots, x_k) f^{m-s}(x_{k+1}, \dots, x_n), \\ y_2 &= f^{m-2}(x_1, \dots, x_k) f^2(x_{k+1}, \dots, x_n) \vee f^{m-4}(x_1, \dots, x_k) \wedge \\ &\wedge f^4(x_{k+1}, \dots, x_n) \vee \dots \vee f^g(x_1, \dots, x_k) f^{m-g}(x_{k+1}, \dots, x_n), \\ y_3 &= v_1 \vee w_1, y_4 = v_2 \vee w_2, \dots, y_q = v_t \vee w_{t^*}, \end{aligned}$$

где $s = 1$ и $g = 2$ при $m = \varphi(2)$; $s = 2$ и $g = 1$ при $m = \varphi(1)$; если $t^* = t-1$, то $w_{t^*} = 1$.

5. Функции y_1, y_2, \dots, y_q реализуются по полученным формулам, при этом входящие в них симметричные функции реализуются в виде итеративного S -блока.

Пример 2.8. Синтезируем 3/8-СПТ. Множество $B = \{x_1 \div x_8\}$ делим на два подмножества $B_1 = \{x_1 \div x_4\}$ и $B_2 = \{x_5 \div x_8\}$. Находим представления симметричных функций, соответствующие итеративной схеме (см. рис. 2.23):

$$f^1(x_1 \div x_4) = x_1 \vee x_2 \vee x_3 \vee x_4,$$

$$f^2(x_1 \div x_4) = x_1 x_2 \vee (x_1 \vee x_2) x_3 \vee (x_1 \vee x_2 \vee x_3) x_4,$$

$$f^1(x_5 \div x_8) = x_5 \vee x_6 \vee x_7 \vee x_8,$$

$$f^2(x_5 \div x_8) = x_5 x_6 \vee (x_5 \vee x_6) x_7 \vee (x_5 \vee x_6 \vee x_7) x_8,$$

$$f^3(x_1 \div x_4) = x_1 x_2 x_3 \vee [x_1 x_2 \vee (x_1 \vee x_2) x_3] x_4 = v_1 \vee v_2,$$

$$f^3(x_5 \div x_8) = x_5 x_6 x_7 \vee [x_5 x_6 \vee (x_5 \vee x_6) x_7] x_8 = w_1 \vee w_2.$$

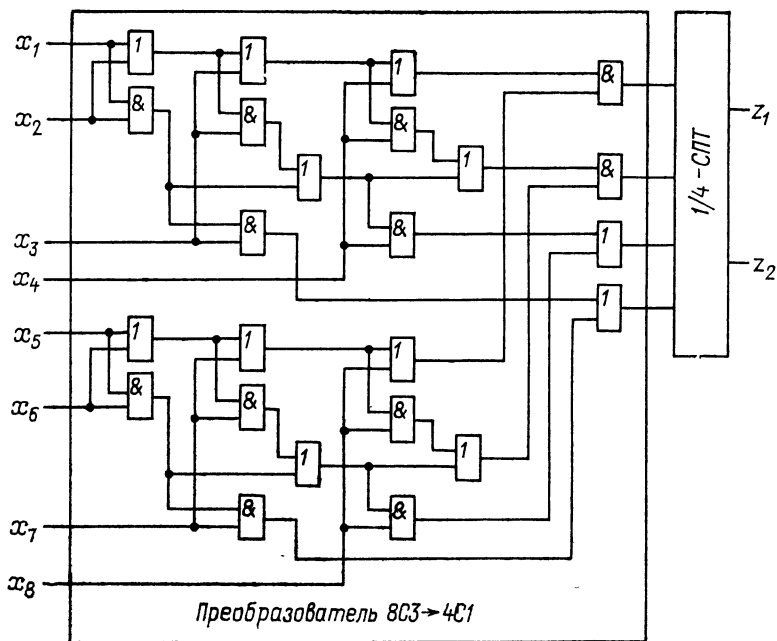


Рис 2.26

Находим функции $y_1 \div y_4$:

$$y_1 = f^2(x_1 \div x_4) f^1(x_5 \div x_8), \quad y_2 = f^1(x_1 \div x_4) f^2(x_5 \div x_8),$$

$$y_3 = v_1 \vee w_1, \quad y_4 = v_2 \vee w_2.$$

На рис. 2.26 представлена схема 3/8-СПТ. В качестве 1/4-СПТ может быть использован любой известный тестер.

Преобразователь $nCm \rightarrow qC1$ имеет следующие характеристики: $L = 2[2(m-1)(n-m) + [n/2] - m + 1] + m - 1$; $N_1 = 2(m-1)(n-m) + [n/2] + 2$; $t = n$; $r = 2[n/2] - 4$. В табл. 2.16 и 2.17 тестеры, реализованные данным методом, обозначены как

СПТ8. Проверяющий тест СПТ8 имеет следующую стандартную структуру (приведен пример для 3/12-СПТ):

$$T = \left. \begin{array}{cc} 111000 & 000000 \\ 110000 & 100000 \\ 100000 & 110000 \\ 000000 & 111000 \end{array} \right\} T'$$

$$T = \left. \begin{array}{cc} 000111 & 000000 \\ 000011 & 000001 \\ 000001 & 000011 \\ 000000 & 000111 \end{array} \right\} T'',$$

$$\left. \begin{array}{cc} 011100 & 000000 \\ 001110 & 000000 \\ 000000 & 011100 \\ 000000 & 001110 \end{array} \right\} T'''$$

которая содержит три блока. Блоки T' и T'' имеют стандартное построение, а блок T''' содержит все слова, имеющие m расположенных подряд разрядов, равных 1, в пределах множеств B_1 и B_2 .

Сравнительный анализ универсальных методов синтеза m/n -СПТ позволяет сделать следующие выводы. Оптимальные структуры тестеров позволяет получать метод [39]. Тестеры СПТ6 имеют сложность, близкую к минимальной, при достаточно высоком быстродействии и требуют для проверки незначительного числа тестовых наборов. Выигрыш по отношению к тестерам других типов особенно проявляется при значениях $n > 4m$. Метод [9] позволяет получать тестеры, требующие для проверки минимального числа тестовых наборов, но при этом резко ухудшается быстродействие. Кроме того, по сравнению с СПТ6 уменьшение числа тестовых наборов незначительно в процентном отношении ко всему множеству кодовых слов. Например, 6/24-СПТ8 требует для своей проверки 0,18 % слов кода 24С6 (см. табл. 2.16), в то время как 6/24-СПТ6 требует 0,28 % слов, но имеет в 1,2 раза большую сложность L и в 2,1 раза больше уровней схемы. Недостатками СПТ7 (метод [58]) являются сравнительно большая сложность и большая длина проверяющего теста, а достоинством — высокое быстродействие. Уменьшение сложности может быть достигнуто за счет применения в структуре СПТ7 S-блоков, реализованных по тупиковым или ите-

Таблица 2.18

m/n	Тип тестера		
	СПТ6	СПТ7	СПТ8
	q		
3/8	4	4	4
3/12	4	4	6
3/24	6	16	12
3/30	8	32	16

ративным схемам, но при этом ухудшается быстродействие. Тестеры СПТ4 и СПТ5 имеют большую сложность и поэтому не могут найти практического применения для тестеров с большими значениями m и n .

Тестеры СПТ6, СПТ7 и СПТ8 строятся с использованием одной и той же схемы преобразования кодов $nCm \rightarrow qC1 \rightarrow 2C1$. Существенным преимуществом СПТ6 является то, что в них с ростом отношения n/m наблюдается значительно более медленный рост числа q по сравнению с СПТ7 и СПТ8 (табл. 2.18). От значения числа q зависит сложность $1/q$ -СПТ, входящего в структуру тестера. В связи с этим при больших значениях отношений n/m тестеры СПТ7 и СПТ8 существенно проигрывают по сложности тестерам СПТ6.

2.6. Синтез быстродействующих m/n -тестеров

Быстродействие m/n -тестеров определяется числом уровней r его схемы. Число уровней имеет важное значение также и с точки зрения построения тестеров на ПЛМ. Реализуются на ПЛМ только тестеры с небольшим числом уровней. Для тестеров с максимальным быстродействием $r = 2$. Двухуровневые тестеры строятся в виде схем И—ИЛИ либо ИЛИ—И (см. рис. 2.5), которые описываются функциями z_1 и z_2 , отвечающими условиям Н1—Н4. Данные условия могут быть выполнены для широкого класса кодов, и поэтому для них существуют двухуровневые реализации тестеров. Однако для целого ряда кодов условия Н1—Н4 не выполняются. В [77] определены некоторые границы параметров кодов nCm , для которых существуют двухуровневые тестеры. Например, они не могут быть построены для кодов $nC1$, $nC2$ (при $n \geq 6$) и $nC3$ (при $n \geq 17$). Для таких кодов возможно построение трехуровневых тестеров.

Двухуровневые тестеры существуют для $2mCm$ -кодов. Для их построения могут быть использованы формулы (2.11), (2.12), (2.15) и (2.16), при этом функции z_1 и z_2 реализуются по своим ДНФ. Для остальных кодов может быть использован описанный выше метод таблиц покрытий для кодов nCm [16, 76]. Он содержит перебор вариантов и поэтому громоздок.

Рассмотрим алгоритмический метод синтеза двухуровневых m/n -СПТ, предложенный в [15]. Код nCm можно разбить на непересекающиеся подмножества кодовых слов, каждое из которых содержит все циклические сдвиги одного слова. Такие подмножества представляют собой строго циклические подкоды рассматриваемого кода. Подкод называется полным, если его мощность равна числу разрядов кода. Порождающим словом p строго циклического подкода называется слово, для которого двоичное число, соответствующее слову p , минимально. Для

построения множества всех слов кода nCm достаточно иметь список порождающих слов. В [19] показано, что число μ порождающих слов полных подкодов удовлетворяет неравенству $\mu \leq] C_n^m/n [$, и предложены правила построения порождающих слов.

Обозначим: $G_m = \{\rho_1^m, \rho_2^m, \dots, \rho_s^m\}$, $G_{m+1} = \{\rho_1^{m+1}, \rho_2^{m+1}, \dots, \rho_s^{m+1}\}$, $G_{m-1} = \{\rho_1^{m-1}, \rho_2^{m-1}, \dots, \rho_s^{m-1}\}$ — множества порождающих слов для кодов nCm , $nC(m+1)$ и $nC(m-1)$ соответственно; $Q(\rho)$ — множество всех кодовых слов циклического подхода с порождающим словом ρ ; G_{m1} и G_{m2} — два непересекающихся подмножества G_m ; W_{m1} и W_{m2} — множества всех кодовых слов циклических подкодов с порождающими словами из G_{m1} и G_{m2} соответственно.

Будем говорить, что циклический подкод $Q(\rho_i)$ покрывает циклический подкод $Q(\rho_j)$, если $\rho_i \geq \rho_j$ или существует слово $g \in Q(\rho_i)$ такое, что $g \geq \rho_j$. Обозначим это как $Q(\rho_i) \geq Q(\rho_j)$.

Функции z_1 и z_2 , описывающие двухуровневые m/n -СПТ, строятся на основе разбиения множества всех кодовых слов W_m (или соответствующих им конъюнкций) на два подмножества, удовлетворяющих условиям Н1 — Н4. В данном случае появляется возможность упростить решение задачи на основе следующего утверждения.

Утверждение 2.3. Если множество G_m разбито на два непересекающихся подмножества G_{m1} и G_{m2} так, что выполняются требования:

- 1) $(\forall \rho_i^{m+1} \in G_{m+1}) (\exists \rho_1 \in G_{m1} \wedge \exists \rho_2 \in G_{m2} \rightarrow Q(\rho_i^{m+1}) \geq Q(\rho_1) \wedge Q(\rho_i^{m+1}) \geq Q(\rho_2))$,
- 2) $(\forall \rho_i^{m-1} \in G_{m-1}) (\exists \rho_1 \in G_{m1} \wedge \exists \rho_2 \in G_{m2} \rightarrow Q(\rho_i) \geq Q(\rho_i^{m+1}) \wedge Q(\rho_i) \geq Q(\rho_i^{m-1}))$,

то разбиение множества кодовых слов W_m на подмножества W_{m1} и W_{m2} удовлетворяет условиям Н1 — Н4.

Поясним метод [15] на примере.

Пример 2.9. Рассмотрим код 6СЗ. Он имеет четыре циклических подкода: $G = \{\rho_1^3 = 111000, \rho_2^3 = 110100, \rho_3^3 = 101100, \rho_4^3 = 101010\}$. Находим множества $G_{m+1} = G_4 = \{\rho_1^4 = 111100, \rho_2^4 = 111010, \rho_3^4 = 110110\}$ и $G_{m-1} = G_2 = \{\rho_1^2 = 110000, \rho_2^2 = 101000, \rho_3^2 = 100100\}$. Для определения соотношений между циклическими подкодами строим таблицы покрытий и переходов. В таблице покрытий (табл. 2.19) представлены порождающие слова подкодов nCm -кода, которые покрывают подкод $Q(\rho_i^{m+1})$, а в таблице переходов (табл. 2.20) — те слова, которые покрывают подкод $Q(\rho_i^{m-1})$. Из таблиц видно, что для выполнения требования утверждения 2.3 слова ρ_2^3 и ρ_3^3 необходимо расположить в различных подмножествах G_{m1} и G_{m2} . Поскольку ρ_2^3 и ρ_3^3 имеются в каждом столбце обеих таблиц, то условия утверждения 2.3 будут выполняться независимо, от

Таблица 2.19

ρ_1^4	ρ_2^4	ρ_3^4
ρ_1^3	ρ_1^3	ρ_2^3
ρ_2^3	ρ_2^3	ρ_3^3
ρ_3^3	ρ_3^3	
	ρ_4^3	

Таблица 2.20

ρ_1^2	ρ_2^2	ρ_3^2
ρ_1^3	ρ_1^3	ρ_2^3
ρ_2^3	ρ_2^3	ρ_3^3
ρ_3^3	ρ_3^3	
	ρ_4^3	

распределения по блокам остальных порождающих слов. Пусть, например, $G_{m1} = \{\rho_1^3, \rho_2^3\}$ и $G_{m2} = \{\rho_3^3, \rho_4^3\}$. Для каждого порождающего слова строится циклический подкод. Например, по слову $\rho_1^3 = 111000$ находим $Q(\rho_1^3) = \{111000, 011100, 001110, 000111, 100011, 110001\}$ и т. д. Затем по множествам G_{m1} и G_{m2} определяются множества $W_{m1} = \{Q(\rho_1^3) \cup Q(\rho_2^3)\}$, $W_{m2} = \{Q(\rho_3^3) \cup Q(\rho_4^3)\}$. Функции z_1 и z_2 , описывающие 3/6-СПТ, определяются как дизъюнкция конъюнкций, соответствующих словам кода 6СЗ, входящим соответственно в множества W_{m1} и W_{m2} .

Двухуровневые тестеры реализуются на одной ПЛМ вида $M[n, C_n^m, 2]$. Для ПЛМ типичными являются следующие неисправности [51, 79]: а) константные неисправности шин и полюсов типа $K \rightarrow 1$ и $K \rightarrow 0$; б) мостиковые неисправности, соответствующие замыканиям между шинами; в) коммутационные неисправности, отражающие дефекты соединений промежуточных шин с входными и выходными шинами (исчезновение и появление этих соединений). Методы построения m/n -СПТ учитывают только константные неисправности. Необходимым условием обнаружения мостиковых и коммутационных неисправностей является использование ПЛМ, не имеющих шин, связанных с элементами, инвертирующими значения входных переменных. Данное условие может быть выполнено, так как тестеры описываются монотонными функциями алгебры логики (ФАЛ).

Для двухуровневых тестеров в [79] получено следующее условие реализации на ПЛМ (оно обеспечивает возможность подключения соседних шин матрицы типа И (см. рис. 2.11) к различным выходным шинам матрицы типа ИЛИ):

$$|\mu_1 - \mu_2| \leq 1, \quad (2.51)$$

где μ_1, μ_2 — число кодовых слов в подмножествах W_{m1} и W_{m2} соответственно.

Заметим, что метод [15] не дает решения для кода 4С2. Оно может быть получено путем перебора вариантов. Например,

$$\begin{aligned} z_1 &= x_1 x_2 \vee x_1 x_3 \vee x_2 x_4, \\ z_2 &= x_2 x_3 \vee x_1 x_4 \vee x_3 x_4. \end{aligned} \quad (2.52)$$

Недостатками рассмотренных тестеров (обозначим их как СПТ9) являются большая сложность и максимальная длина проверяющего теста, включающего в себя все слова кода nCm . В табл. 2.21 приведены основные характеристики быстроедействующих тестеров, а в табл. 2.22 — параметры реализующих их ПЛМ.

Таблица 2.21

m/n	Тип тестера	Х а р а к т е р и с т и к и						
		L	N_1	r_1	t	h	S	r_3
2/5	СПТ9	30	12	2	10	1	70	1
	СПТ10	38	15	3	10	2	124	2
	СПТ11	35	12	2	10	2	60	1
	СПТ12	32	14	3	7	3	76	2
2/6	СПТ9	—	—	—	—	—	—	—
	СПТ10	63	20	3	15	2	192	2
	СПТ11	44	17	3	12	3	163	2
	СПТ12	48	17	3	12	3	129	2
3/6	СПТ9	80	22	2	20	1	160	1
	СПТ10	80	22	2	20	1	160	1
	СПТ11	48	14	2	13	2	84	1
	СПТ12	36	14	3	13	3	117	2
3/8	СПТ9	224	58	2	56	1	560	1
	СПТ10	264	62	3	56	2	832	2
	СПТ11	230	48	2	45	2	414	1
	СПТ12	170	50	3	37	3	949	2

Указанные недостатки в определенной мере устраняются в методе [68], который позволяет построить в виде трехуровневой схемы вида И — И — ИЛИ или ИЛИ — ИЛИ — И любой m/n -СПТ (в том числе и $1/n$ - и $(n-1)/n$ -СПТ, которые, однако, имеют существенно большую сложность, чем СПТ3 в табл. 2.10). Метод также основан на разбиении множества W_m слов кода nCm на подмножества W_{m1} и W_{m2} , отвечающие условиям

Таблица 2.22

m/n	Характеристики ПЛМ			
	СПТ9	СПТ10	СПТ11	СПТ12
2/5	$M [5, 10, 2]$	$M1 [5, 3, 3],$ $M2 [8, 10, 2]$	$M1 [5, 5, 1],$ $M2 [5, 5, 1]$	$M1 [4, 4, 1],$ $M2 [6, 3, 1],$ $M3 [6, 5, 1]$
2/6	—	$M1 [6, 3, 3],$ $M2 [9, 15, 2]$	$M1 [6, 5, 5],$ $M2 [8, 4, 1],$ $M3 [11, 6, 1]$	$M1 [6, 3, 3],$ $M2 [6, 6, 1],$ $M3 [9, 6, 1]$
3/6	$M [6, 20, 2]$	$M1 [6, 20, 2]$	$M1 [6, 6, 1],$ $M2 [6, 6, 1]$	$M1 [6, 3, 3],$ $M2 [9, 3, 1],$ $M3 [9, 6, 1]$
3/8	$M [8, 56, 2]$	$M1 [8, 4, 4],$ $M2 [12, 56, 2]$	$M1 [8, 23, 1],$ $M2 [8, 23, 1]$	$M1 [8, 11, 11],$ $M2 [19, 14, 1],$ $M3 [19, 23, 1]$

Н1 — Н4. Показано, что для кодов $2mCm$ данные условия выполняются, если используется следующий принцип разбиения. Множество переменных B разделяется на подмножества B_1 и B_2 так, что $n_1 = n_2 = n/2$. В множество включаются все слова, содержащие нечетное число разрядов $x_i \in B_1$, равных 1, а в множество W_{m2} — содержащие четное число таких разрядов. С использованием этого принципа строятся двухуровневые $m/2m$ -СПТ, а для всех остальных кодов он дает двухуровневую схему, не обладающую свойством самопроверки.

Рассмотрим, например, код 5СЗ. Множество B разобьем на подмножества $B_1 = \{x_1, x_2, x_3\}$ и $B_2 = \{x_4, x_5\}$. Тогда

$$W_{m1} = \{11100, 10011, 01011, 00111\},$$

$$W_{m2} = \{11010, 10110, 01110, 11001, 10101, 01101\}.$$

Данному разбиению множества W_m соответствует схема, описываемая функциями

$$\begin{aligned} z_1 &= x_1 x_2 x_3 \vee x_1 x_4 x_5 \vee x_2 x_4 x_5 \vee x_3 x_4 x_5, \\ z_2 &= x_1 x_2 x_4 \vee x_1 x_3 x_4 \vee x_2 x_3 x_4 \vee x_1 x_2 x_5 \vee \\ &\quad \vee x_1 x_3 x_5 \vee x_2 x_3 x_5. \end{aligned} \quad (2.53)$$

В схеме не обнаруживаются неисправности, соответствующие следующим фиксациям переменных: $x_1 = 1$ в конъюнкции $x_1 x_4 x_5$, $x_2 = 1$ в $x_2 x_4 x_5$ и $x_3 = 1$ в $x_3 x_4 x_5$. Для решения задачи обнаружения этих неисправностей двухуровневая схема тестера преобразуется в трехуровневую за счет введения элементов И с числом входов $\mu < m$, участвующих в реализации нескольких

конъюнкций, принадлежащих как z_1 , так и z_2 . Элементы И выбираются таким образом, чтобы они включали в себя входы, имеющие необнаруживаемые неисправности, и чтобы, если неисправность входа не фиксируется по выходу z_1 , она могла быть зафиксирована по выходу z_2 и наоборот. В данном случае требуется выделение трех элементов И, реализующих функции $g_1 = x_1x_4$, $g_2 = x_2x_4$ и $g_3 = x_3x_5$. Тогда система (2.53) заменяется системой функций

$$\begin{aligned} z_1 &= x_1x_2x_3 \vee g_1x_5 \vee g_2x_5 \vee g_3x_4, \\ z_2 &= x_1g_2 \vee g_1x_3 \vee g_2x_3 \vee x_1x_2x_5 \vee x_1g_3 \vee x_2g_3. \end{aligned} \quad (2.54)$$

Таким образом, при синтезе трехуровневых m/n -СПТ необходимо решать задачу выделения элементов И первого уровня. В [68] данная задача решена в алгоритмах синтеза m/n -СПТ для случая, когда $n > 2m$. Тестеры со значениями $n < 2m$ могут быть получены из $(n-m)/n$ -СПТ путем замены в схеме последнего из элементов И на элементы ИЛИ и наоборот.

Алгоритм 2.5 (случай $n = \varphi(2)$):

1. Множество переменных $B = \{x_1, x_2, \dots, x_{2k-1}, x_{2k}\}$ разбивается на два подмножества $B_1 = \{x_1, \dots, x_k\}$ и $B_2 = \{x_{k+1}, \dots, x_{2k}\}$.

2. Формируются множества W_{m_1} и W_{m_2} с использованием указанного выше принципа, и по ним составляются функции z_1 и z_2 , описывающие двухуровневую схему.

3. Для каждого слова ρ_i nCm -кода определяется специальное множество $d(\rho_i)$ с помощью следующей процедуры:

3.1. Составляется список переменных x_i , принимающих в слове ρ_i значение 1.

3.2. Если в списке имеются переменные x_i и x_{i+k} , то множество $\{x_i, x_{i+k}\}$ включается в $d(\rho_i)$; все остальные переменные списка включаются в $d(\rho_i)$ как одноэлементные подмножества.

4. Составляется общий список двухэлементных подмножеств $\{x_i, x_{i+k}\}$, входящих в множества $d(\rho_i)$ ($i \in \{1, \dots, C_n^m\}$). Устанавливаются двухвходовые элементы И, реализующие функции $g = x_i x_{i+k}$, соответствующие каждому из двухэлементных подмножеств.

5. По функциям, полученным в п. 2, реализуется трехуровневая схема тестера с учетом первого уровня схемы, образованного полученными в п. 4 двухвходовыми элементами И. При этом обязательным является использование элементов И первого уровня для реализации конъюнкций, содержащих переменные, фиксация которых в константу не обнаруживается.

Алгоритм 2.6 (случай $n = 2m-1$):

1. Множество $B = \{x_1, \dots, x_{2k}, x_{2k+1}\}$ разбивается на подмножества $B_1 = \{x_1, \dots, x_{k+1}\}$ и $B_2 = \{x_{k+2}, \dots, x_{2k+1}\}$.

2. Выполняются п. 2—5 алгоритма 2.5, причем п. 3.2 формулируется следующим образом: если в списке имеются перемен-

ные x_1 и x_{k+2} , то множество $\{x_1, x_{k+2}\}$ включается в $d(\rho_i)$; если в списке имеются переменные x_i и x_{i+k} ($i \geq 2$), то множество $\{x_i, x_{i+k}\}$ также включается в $d(\rho_i)$; все остальные переменные списка включаются в $d(\rho_i)$ как одноэлементные подмножества.

Алгоритм 2.7 (случай $n = \varphi(1)$ и $2m-3 \geq n \geq m+2$):

1. Выполняется п. 1 алгоритма 2.6.

2. Выполняются п. 2—5 алгоритма 2.5, причем п. 3.2 формулируется следующим образом: если в списке имеются переменные x_i и x_{i+k} ($i \in \{3, 4, \dots, k+1\}$), то множество $\{x_i, x_{i+k}\}$ включается в $d(\rho_i)$; если в списке имеется переменная x_{k+2} , а также одна или обе переменные из множества $\{x_1, x_2\}$, то в зависимости от этого в $d(\rho_i)$ включается одно из множеств $\{x_1, x_{k+2}\}$, $\{x_2, x_{k+2}\}$ или $\{x_1, x_2, x_{k+2}\}$; все остальные переменные списка включаются в $d(\rho_i)$ как одноэлементные подмножества.

Пример 2.10. Получим трехуровневый 3/5-СПТ. Так как в этом случае $n = 2m-1$, то надо применить алгоритм 2.6. Множество $B = \{x_1, \dots, x_5\}$ делим на подмножества $B_1 = \{x_1, x_2, x_3\}$ и $B_2 = \{x_4, x_5\}$. Выше для данного случая получены подмножества W_{m1} и W_{m2} , а также функции z_1 и z_2 (2.53). Находим множества $d(\rho_i)$:

$$\begin{aligned} d(11100) &= \{\{x_1\}, \{x_2\}, \{x_3\}\}; & d(10110) &= \{\{x_1, x_4\}, \{x_3\}\}; \\ d(10011) &= \{\{x_1, x_4\}, \{x_5\}\}; & d(01110) &= \{\{x_2, x_4\}, \{x_3\}\}; \\ d(01011) &= \{\{x_2, x_4\}, \{x_5\}\}; & d(11001) &= \{\{x_1\}, \{x_2\}, \{x_5\}\}; \\ d(00111) &= \{\{x_3, x_5\}, \{x_4\}\}; & d(10101) &= \{\{x_1\}, \{x_3, x_5\}\}; \\ d(11010) &= \{\{x_2, x_4\}, \{x_1\}\}; & d(01101) &= \{\{x_2\}, \{x_3, x_5\}\}. \end{aligned}$$

Составляем общий список двухэлементных подмножеств: $\{x_1, x_4\}, \{x_2, x_4\}, \{x_3, x_5\}$. Им соответствуют элементы И, реализующие функции $g_1 = x_1x_4$, $g_2 = x_2x_4$, $g_3 = x_3x_5$. Тестер реализуется по формулам (2.54).

Для построения трехуровневых тестеров требуется $N_1' = C_n^m + N_1' + 2$ элементов, где $N_1' = n/2$ (при $n = \varphi(2)$); $N_1' = \lfloor n/2 \rfloor$ (при $n = 2m-1$); $N_1' = \lfloor n/2 \rfloor + 1$ (при $2m-3 \geq n \geq m+2$). Длина проверяющего теста $t = C_n^m$. В табл. 2.21 и 2.22 рассмотренные тестеры обозначены как СПТ10. Трехуровневые тестеры реализуются на двух ПЛМ.

В работе [40] предложен второй универсальный метод синтеза m/n -СПТ с максимальным быстродействием. Определим множество V ($V \in W_m$) конъюнкций M_i ранга m , удовлетворяющих следующим условиям:

1) каждая конъюнкция $M_j \in W_{m+1}$ покрывается хотя бы одной конъюнкцией $M_i \in V$;

2) каждая конъюнкция $M_j \in W_{m-1}$ покрывает хотя бы одну конъюнкцию $M_i \in V$;

3) каждая конъюнкция $M_i \in V$ покрывает хотя бы одну такую конъюнкцию $M_j \in W_{m+1}$, которая не покрывается более какой-либо другой конъюнкцией из множества V .

Рассмотрим, например, код 5С2. Для него имеем: $W_m = \{x_1x_2, x_1x_3, x_1x_4, x_1x_5, x_2x_3, x_2x_4, x_2x_5, x_3x_4, x_3x_5, x_4x_5\}$; $W_{m+1} = \{x_1x_2x_3, x_1x_2x_4, x_1x_2x_5, x_1x_3x_4, x_1x_3x_5, x_1x_4x_5, x_2x_3x_4, x_2x_3x_5, x_2x_4x_5, x_3x_4x_5\}$; $W_{m-1} = \{x_1, x_2, x_3, x_4, x_5\}$. Множество V может быть получено с помощью таблиц покрытий (см. табл. 2.4 и 2.5). В рассматриваемом случае имеем

$$V = \{x_1x_2, x_3x_4, x_3x_5, x_4x_5\}.$$

Определим функции z_1 и z_2 , описывающие m/n -СПТ. Функция z_1 находится как дизъюнкция конъюнкций, входящих в V :

$$z_1 = x_1x_2 \vee x_3x_4 \vee x_3x_5 \vee x_4x_5. \quad (2.55)$$

Функция z_2 представляет собой выражение общего конъюнктивного вида и содержит столько дизъюнкций, сколько конъюнкций содержится в z_1 . При этом каждая дизъюнкция z_2 образуется из конъюнкций z_1 путем соединения знаком дизъюнкции всех тех входных переменных тестера, которые не входят в данную конъюнкцию z_1 . Такие конъюнкцию и дизъюнкцию будем называть условно инверсионными. По выражению (2.55) находим:

$$z_2 = (x_3 \vee x_4 \vee x_5) (x_1 \vee x_2 \vee x_5) \times \\ \times (x_1 \vee x_2 \vee x_4) (x_1 \vee x_2 \vee x_3). \quad (2.56)$$

Тестер, описываемый функциями z_1 и z_2 , обладает свойством контроля входного вектора, но не обладает свойством самопроверки. Для обеспечения этого свойства при определении множества V требуется выполнить два дополнительных условия:

4) для каждой конъюнкции $M_j \in W_{m-1}$ имеется хотя бы одна конъюнкция $M_i \in W_m$, которая покрывается конъюнкцией M_j и не входит в множество V ;

5) для каждой конъюнкции $M_j \in W_{m+1}$ имеется хотя бы одна конъюнкция $M_i \in W_m$, которая покрывает конъюнкцию M_j и не входит в множество V .

Необходимость выполнения этих условий приводит к увеличению мощности множества V . Так, для кода 5С2 мощность увеличивается на единицу. Например,

$$V = \{x_1x_2, x_3x_4, x_1x_5, x_2x_3, x_4x_5\}.$$

В этом случае 2/5-СПТ описывается следующими функциями:

$$z_1 = x_1x_2 \vee x_3x_4 \vee x_1x_5 \vee x_2x_3 \vee x_4x_5, \\ z_2 = (x_3 \vee x_4 \vee x_5) (x_1 \vee x_2 \vee x_5) (x_2 \vee x_3 \vee x_4) \wedge \\ \wedge (x_1 \vee x_4 \vee x_5) (x_1 \vee x_2 \vee x_3). \quad (2.57)$$

Множество V с выполнением условий 1—5 определяется по таблицам покрытий с использованием перебора вариантов для тех кодов, для которых существуют двухуровневые модификации тестеров. Существует целенаправленный алгоритм, позво-

ляющий получать множества V с выполнением необходимых условий, но не гарантирующий получение его минимального варианта [40].

Трехуровневая схема тестера может быть построена для любого кода nCm . В этом случае строится множество V^* с выполнением только первых трех из указанных условий. При этом для кодов, у которых $n \geq 2m$, автоматически выполняется условие 4, а для кодов, у которых $n \leq 2m$, — условие 5. Задача вы-

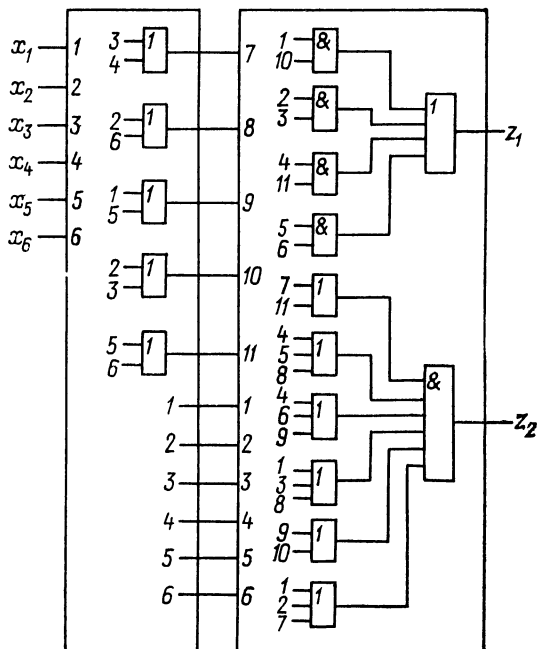


Рис. 2.27

полнения условий 1—3 совпадает с известной задачей построения минимального покрытия [29]. Например, для кода 6C2 получено множество $V^* = \{x_1x_2, x_1x_3, x_2x_3, x_4x_5, x_4x_6, x_5x_6\}$, которому соответствуют следующие функции:

$$\begin{aligned} z_1 &= x_1x_2 \vee x_1x_3 \vee x_2x_3 \vee x_4x_5 \vee x_4x_6 \vee x_5x_6, \\ z_2 &= ([x_3] \vee x_4 \vee x_5 \vee x_6) ([x_2] \vee x_4 \vee x_5 \vee x_6) \times \\ &\times ([x_1] \vee x_4 \vee x_5 \vee x_6) (x_1 \vee x_2 \vee x_3 \vee [x_6]) \times \\ &\times (x_1 \vee x_2 \vee x_3 \vee [x_5]) (x_1 \vee x_2 \vee x_3 \vee [x_4]). \end{aligned} \quad (2.58)$$

Тестер, реализованный по этим функциям, обладает свойством контроля входного вектора, но не обладает свойством самопроверки. Выполнение условия 4 обеспечивает обнаружение неисправностей в схеме, реализующей z_1 . В схеме же, реали-

зующей z_2 , не обнаруживаются (ввиду невыполнения условия 5) неисправности, соответствующие фиксации отдельных букв в 0. Будем называть такие буквы неопределенными (они заключены в квадратные скобки). Для обнаружения указанных неисправностей в схему тестера вводится третий уровень элементов, которые реализуют дизъюнкции ранга $q < n - m$. Такие дизъюнкции, включающие в себя неопределенные переменные, будем называть первичными. На рис. 2.27 представлена трехуровневая схема 2/6-СПТ, реализованная по функциям (2.58). Рассмотрим реализацию дизъюнкций $[x_3] \vee x_4 \vee x_5 \vee x_6$ и $x_1 \vee x_2 \vee x_3 \vee [x_4]$. В схеме двухуровневого тестера они реализуются с помощью отдельных элементов ИЛИ с четырьмя входами. Неисправности типа $K \rightarrow 0$ входов этих элементов, связанных с неопределенными переменными, не обнаруживаются в структуре двухуровневого тестера. Для реализации трехуровневого тестера выделяется общая часть указанных функций — первичная дизъюнкция $x_3 \vee x_4$, которая реализуется отдельным элементом. Этот элемент входит в первый уровень схемы и соединяется со входами элементов второго уровня, реализующих рассматриваемые дизъюнкции. При этом вход x_3 элемента первого уровня проверяется по условиям проверки переменной x_3 в дизъюнкции $x_1 \vee x_2 \vee x_3 \vee [x_4]$, а вход x_4 — по условиям проверки переменной x_4 в дизъюнкции $[x_3] \vee x_4 \vee x_5 \vee x_6$.

Первичные дизъюнкции, реализуемые элементами первого уровня, должны отвечать следующим условиям: а) каждая первичная дизъюнкция должна полностью входить в две или более дизъюнкций функции z_2 ; б) все неопределенные переменные должны входить в выбранные первичные дизъюнкции; в) если неопределенная переменная x_i входит в некоторую первичную дизъюнкцию, то последняя должна составлять часть хотя бы одной такой дизъюнкции, в которой переменная x_i не заключена в квадратные скобки.

В [40] разработаны методы нахождения всех неопределенных букв в функции z_2 и всех возможных первичных дизъюнкций, а также способ представления функции z_2 , при котором обеспечивается обнаружение одиночных неисправностей в трехуровневой структуре тестера. С использованием этих методов сформулирован следующий алгоритм синтеза трехуровневых m/n -СПТ.

Алгоритм 2.8:

1. Для заданного кода nCm определяются множества W_m , W_{m+1} и W_{m-1} .
2. Путем решения задачи минимального покрытия определяется множество V^* .
3. Для множества V^* составляются функции z_1 и z_2 .
4. В функции z_2 находятся и заключаются в квадратные скобки неопределенные переменные.
5. Для функции z_2 составляется множество первичных дизъюнкций.

6. Находится представление функции z_2 , в котором все неопределенные переменные включаются в первичные дизъюнкции (последние помещаются в фигурные скобки).

7. Осуществляется произвольное эквивалентное преобразование функций z_1 и z_2 с целью упрощения схемной реализации тестера. При этом в функции z_2 сохраняются все выделенные первичные дизъюнкции, а функция z_2 преобразуется с учетом возможности совместной реализации обеих функций.

8. Функции z_1 и z_2 реализуются по полученным выражениям, при этом первичные дизъюнкции, включенные в функцию z_2 , реализуются элементами первого уровня схемы.

Структура 2/6-СПТ на рис. 2.27 соответствует следующему представлению функций:

$$\begin{aligned} z_1 &= x_1 \{x_2 \vee x_3\} \vee x_2 x_3 \vee x_4 \{x_5 \vee x_6\} \vee x_5 x_6, \\ z_2 &= (\{x_3 \vee x_4\} \vee \{x_5 \vee x_6\}) (\{x_2 \vee x_6\} \vee x_4 \vee x_5) (\{x_1 \vee x_5\} \vee \\ &\vee x_4 \vee x_6) (x_1 \vee x_3 \vee \{x_2 \vee x_6\}) (\{x_1 \vee x_5\} \vee \{x_2 \vee x_3\}) \wedge \\ &\wedge (x_1 \vee x_2 \vee \{x_3 \vee x_4\}). \end{aligned}$$

В табл. 2.21 и 2.22 рассмотренные тестеры обозначены как СПТ11. Двухуровневые тестеры реализуются на двух ПЛМ, а трехуровневые — на трех ПЛМ. Возможна вторая модификация этих тестеров — СПТ12. Двухуровневые тестеры преобразуются произвольным образом в трехуровневые, при этом уменьшается сложность L , но увеличивается до трех число ПЛМ. В трехуровневых тестерах общее число уровней сохраняется, но функция z_1 реализуется с помощью отдельной двухуровневой схемы без учета выделения первичных дизъюнкций. При этом увеличивается сложность L , но уменьшаются размеры ПЛМ. По сравнению с СПТ9 и СПТ10 рассмотренные тестеры имеют меньшие сложность и длину проверяющего теста, но требуют, как правило, большего числа ПЛМ.

В работах [41, 51] предложены методы, позволяющие строить четырех- и пятиуровневые тестеры. Более экономичные схемы дает метод [41], который предполагает следующую схему преобразования кодов: $nCm \rightarrow t(k/p)C1 \rightarrow pCk \rightarrow 2C1$. При этом в качестве кода pCk выбирается такой, для которого существует двухуровневый k/p -СПТ. Так как для проверки k/p -СПТ требуется $t(k/p)$ слов кода pCk , то они должны быть сформированы на выходе преобразователя $nCm \rightarrow pCk$, что достигается за счет двухкаскадного преобразования по схеме $nCm \rightarrow t(k/p)C1 \rightarrow pCk$. Преобразователь $nCm \rightarrow t(k/p)C1$ строится на основе разложения типа F , поэтому параметры кода pCk определяются с помощью системы (2.40). В табл. 2.23 приведены границы применения основных двухуровневых тестеров, рассчитанные с помощью этой системы. В таблице тестеру 2/4-СПТ первой модификации соответствует рис. 2.1, а второй моди-

Таблица 2.23

k/p -СПТ	$m = 2$	$m \geq 3$
2/4 (1)	$4 \leq n \leq 16$	$2m \leq n \leq 4m$
2/4 (2)	$16 < n \leq 64$	$4m < n \leq 8m$
2/5	$64 < n \leq 1024$	$8m < n \leq 32m$
3/6	$1024 < n \leq 1048576$	$32m < n \leq 1024m$

фикации — рис. 2.5. Из таблицы видно что во всех практически важных случаях могут быть использованы 2/4- или 2/5-СПТ.

Преобразователь $nCm \rightarrow t(k/p)C1$ строится в виде двухуровневой схемы И — ИЛИ, преобразователь $t(k/p)C1 \rightarrow pCk$ — в виде одноуровневой схемы из p элементов ИЛИ, а преобразователь $pCk \rightarrow 2C1$ (k/p -СПТ) — в виде двухуровневой схемы И — ИЛИ. В результате формируется пятиуровневая структура И — ИЛИ — ИЛИ — И — ИЛИ. Второй и третий уровни, реализуемые элементами ИЛИ, могут быть объединены, в результате чего образуется четырехуровневая структура тестера, которая реализуется на двух ПЛМ. Будем обозначать данные тестеры как СПТ13 (пятиуровневые) и СПТ14 (четырёхуровневые).

2.7. Блочная реализация тестеров

Так как на практике применяется большое число разновидностей кода nCm , то возникает задача построения m/n -СПТ с помощью стандартных блоков, которые могут быть реализованы типовыми микросхемами. В качестве таких блоков могут выступать m/n -СПТ с небольшими значениями m и n , а также специальные логические блоки, служащие для объединения m/n -СПТ между собой. Будем называть функционально-полным набором (ФПН) такой набор блоков, с помощью которых может быть реализован (путем соединения блоков между собой) любой m/n -СПТ.

Все множество равновесных кодов разобьем на три класса: $nC1$, $nC(n-1)$ и nCm ($2 \leq m \leq n-2$). Метод блочной реализации $1/n$ -СПТ непосредственно вытекает из каскадного принципа [24], описанного в § 2.2. При этом в ФПН достаточно включать только 1/4- и 1/5-СПТ с минимальной сложностью (см. рис. 2.4 и систему функций (2.24)). Класс кодов $nC(n-1)$ является обратным классу $nC1$ и обладает теми же свойствами. Любой $(n-1)/n$ -СПТ реализуется путем соединения между собой 3/4- и 4/5-СПТ. Структуры последних образуются соответственно из 1/4- и 1/5-СПТ путем замены в них элементов И на элементы ИЛИ и наоборот. Такие схемы называются обрат-

ными. Из каскадного принципа также следует, что для реализации тестеров первых двух классов в ФПН достаточно включить только 1/3-СПТ (см. рис. 2.15) и 2/3-СПТ (обратная схема рис. 2.15). Однако при этом схемы тестеров будут представлять собой последовательное соединение схем с памятью, которое требует специальных мер для обеспечения устойчивой работы.

Рассмотрим метод блочной реализации m/n -СПТ с любыми значениями m и n [57]. ФПН блоков содержит 17 тестеров (1/4-, 2/4-, 3/4-, 1/5-, 2/5-, 3/5-, 4/5-, 1/6-, 2/6-, 3/6-, 4/6-, 5/6-, 2/7-, 3/7-, 4/7-, 5/7- и 4/8-СПТ) и два блока сравнения парафазных сигналов (СС1 и СС2), которые имеют парафазные выходы и обладают свойством самопроверки. Блок СС1 имеет два

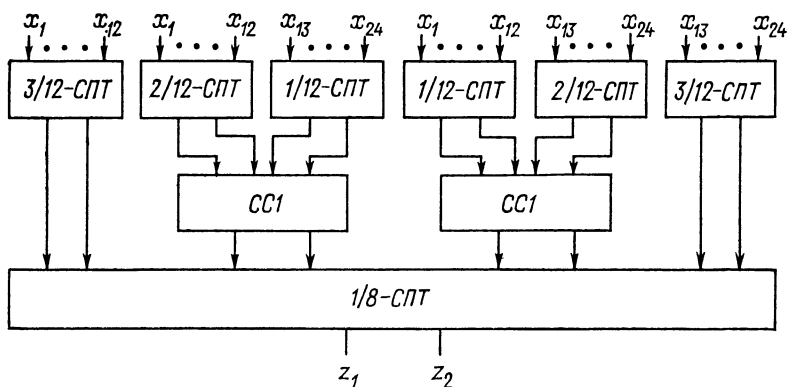


Рис. 2.28

парафазных входа (x_1, \bar{x}_1) и (x_2, \bar{x}_2) и двухуровневую структуру, описываемую функциями $z_1 = x_1 x_2 \vee \bar{x}_1 \bar{x}_2$ и $z_2 = x_1 x_2 \vee \bar{x}_1 \bar{x}_2$. Блок СС2 имеет структуру, обратную структуре блока СС1. Блочная реализация основана на разложении базовой функции тестера (2.33), причем $k = \lfloor n/2 \rfloor$. Идею метода проиллюстрируем на примере построения 3/24-СПТ. Для кода 24СЗ имеем:

$$\begin{aligned} f^3(x_1 \div x_{24}) = & f^3(x_1 \div x_{12}) \vee f^2(x_1 \div x_{12}) f^1(x_{13} \div x_{24}) \vee \\ & \vee f^1(x_1 \div x_{12}) f^2(x_{13} \div x_{24}) \vee f^3(x_{13} \div x_{24}). \end{aligned} \quad (2.59)$$

На рис. 2.28 представлена блочная схема 3/24-СПТ, соответствующая разложению (2.59). Каждый элемент разложения реализуется отдельной самопроверяемой схемой с двумя выходами. Элементы $f^m(x_{i_1}, \dots, x_{i_k})$ реализуются m/k -тестерами, а элементы $f^{m-t}(x_{i_1}, \dots, x_{i_k}) f^t(x_{i_{k+1}}, \dots, x_{i_n})$ — двумя отдельными тестерами $((m-t)/k$ -СПТ и $t/(m-k)$ -СПТ), выходы которых объединены блоком СС1. В результате осуществляется преобразование $nCm \rightarrow 2qC1$, где q равно числу элементов правой части разложения (2.33). Каждый из тестеров, входящих

в полученную блочную структуру, представляется аналогичным образом на основании соответствующего разложения вида (2.33). Процесс представления тестеров их блочными структурами заканчивается тогда, когда все входящие в схему блоки будут входить в ФПН.

При разложении функций вида $f^m(x_{i_1}, \dots, x_{i_{2m}})$ и $f^m(x_{i_1}, \dots, x_{i_{2m}}, x_{i_{2m+1}})$ возникают элементы разложения вида $f^m(x_{i_1}, \dots, x_{i_m})$, которые не реализуются самопроверяемой схемой. В таких случаях осуществляется преобразование формулы разложения с использованием равенства

$$f^m(x_{i_1}, \dots, x_{i_m}) \vee f^{m-1}(x_{i_1}, \dots, x_{i_m}) f^1(x_{i_{m+1}}, \dots, x_{i_{2m(2m+1)}}) = f^m(x_{i_1}, \dots, x_{i_{m+1}}) \vee f^{m-1}(x_{i_1}, \dots, x_{i_m}) f^1(x_{i_{m+2}}, \dots, x_{i_{2m(2m+1)}}).$$

Например, для кода 8С4 имеем:

$$f^4(x_1 \div x_8) = f^4(x_1 \div x_4) \vee f^3(x_1 \div x_4) f^1(x_5 \div x_8) \vee f^2(x_1 \div x_4) \wedge f^2(x_5 \div x_8) \vee f^1(x_1 \div x_4) f^3(x_5 \div x_8) \vee f^4(x_5 \div x_8).$$

Преобразованная формула имеет вид

$$f^4(x_1 \div x_8) = f^4(x_1 \div x_5) \vee f^3(x_1 \div x_4) f^1(x_6 \div x_8) \vee f^2(x_1 \div x_4) \wedge f^2(x_5 \div x_8) \vee f^1(x_1 \div x_3) f^3(x_5 \div x_8) \vee f^4(x_4 \div x_8).$$

Недостатком метода [57] является большое число блоков, входящих в ФПН. Существенное уменьшение числа блоков достигнуто в [42], где блочная структура тестера также строится на основе разложения (2.33), но при этом $k = n-1$. В этом случае имеет место стандартное разложение вида

$$\begin{aligned} f^m(x_1, \dots, x_{n-1}, x_n) &= \\ &= f^m(x_1, \dots, x_{n-1}) \vee \\ &\vee f^{m-1}(x_1, \dots, x_{n-1}) f^1(x_n), \end{aligned} \quad (2.60)$$

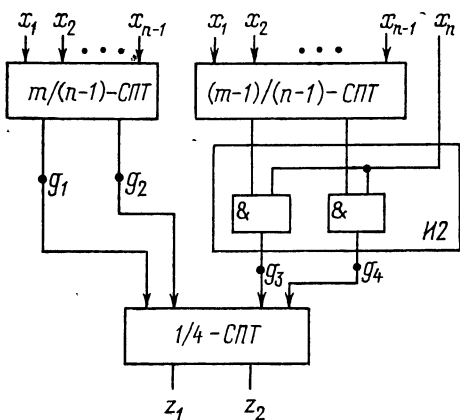


Рис. 2.29

которому соответствует стандартная блочная структура m/n -СПТ, показанная на рис. 2.29. Блок $m/(n-1)$ -СПТ реализует слова кода nCm , соответствующие конъюнкциям, входящим в функцию $f^m(x_1, \dots, x_{n-1})$. Структура, представляющая собой последовательное соединение блоков $(m-1)/(n-1)$ -СПТ и И2, реализует слова, отвечающие конъюнкциям функции $\varphi = f^{m-1}(x_1, \dots, x_{n-1}) f^1(x_n)$. Блок $1/4$ -СПТ осуществляет объеди-

нение всех слов кода nCm . Возможна также вторая модификация стандартной структуры, в которой блок $1/4$ -СПТ заменен двумя элементами ИЛИ, реализующими функции $z_1 = g_1 \vee g_3$ и $z_2 = g_2 \vee g_4$.

В виде стандартной структуры не могут быть представлены $1/n$ -, $(n-1)/n$ - и $2/4$ -СПТ. Остальные тестеры реализуются стандартной структурой, в которую в виде отдельных блоков входят m/n -СПТ с меньшими значениями m и n , чем у исходного тестера. Последние, в свою очередь, также могут быть представлены в виде стандартной структуры и т. д. При таком подходе для реализации любого m/n -СПТ требуются только тестеры с минимальными значениями m и n , которые не могут быть представлены как композиция более простых тестеров. К ним относятся тестеры: $1/4$ -, $1/5$ -, $3/4$ -, $4/5$ - и $2/4$ -СПТ, которые совместно с блоком И2 составляют ФПН.

Сформулируем алгоритм реализации m/n -СПТ.

Алгоритм 2.9:

1. Для заданного m/n -СПТ записывается формула разложения следующего вида: $m/n = m/(n-1) + (m-1)/(n-1)$, описывающая стандартную структуру, представленную на рис. 2.29. Указываются параметры $m(n-1)$ -СПТ и $(m-1)/(n-1)$ -СПТ, входящих в стандартную структуру. Знак «суммы» определяет объединение выходов этих тестеров с помощью стандартных блоков И2 и $1/4$ -СПТ. Например, для $2/5$ -СПТ имеем: $2/5 = 2/4 + 1/4$.

2. В формуле, полученной при выполнении предыдущего пункта, обозначения тестеров с параметрами $m \geq 2$ и $n \geq 5$ заменяются соответствующими формулами разложения. Например, для $2/6$ -СПТ имеем: $2/6 = (2/5 + 1/5) = ((2/4 + 1/4) + 1/5)$.

3. Повторяется процедура п. 2 для вновь полученной формулы и т. д. Процесс повторяется до тех пор, пока формула не будет содержать только обозначения $2/4$, $1/n$ и $(n-1)/n$ ($n \in \{4, 5, 6, \dots\}$). Рассмотрим, например, случай $4/7$ -СПТ: $4/7 = (4/6 + 3/6) = ((4/5 + 3/5) + (3/5 + 2/5)) = ((4/5 + (3/4 + 2/4)) + ((3/4 + 2/4) + (2/4 + 1/4)))$.

4. По тупиковой формуле разложения составляется блочная структура m/n -СПТ. При этом каждая сумма, заключенная в скобки, представляется в виде стандартной структуры рис. 2.29.

5. В блочной структуре m/n -СПТ все тестеры вида $1/n$ -СПТ ($n \geq 6$) представляются в виде композиции $1/4$ - и $1/5$ -СПТ, а тестеры $(n-1)/n$ -СПТ ($n \geq 6$) — в виде композиции $3/4$ - и $4/5$ -СПТ.

В результате применения данного алгоритма образуется структура, содержащая определенное число однотипных блоков, многие из которых могут быть объединены. Объединение блоков дает стандартные структуры тестеров для заданного m . На рис. 2.30 показана диаграмма, содержащая структуру $3/n$ -СПТ при $n \leq 10$. Тестер $3/10$ -СПТ образуется как объединение $3/9$ -

и 2/9-СПТ. Плюс в кружке определяет объединение тестеров с помощью блоков объединения, содержащих И2 и 1/4-СПТ. В свою очередь, 3/9- и 2/9-тестеры образуются объединением соответственно 3/8-, 2/8-СПТ и 2/8-, 1/8-СПТ. Причем на этом этапе представления 3/10-СПТ тестеры 2/8-СПТ могут быть совмещены (что и показано на диаграмме). Аналогичное совмещение осуществляется на всех других этапах представления тестера. Все входящие в структуру 1/*n*-СПТ реализуются с помощью двух каскадных цепей. Первая цепь состоит из 1/4-СПТ и реализует все 1/*n*-тестеры с *n* = φ(2). В начале второй цепи включается 1/5-СПТ, а затем последовательно с ним — 1/4-СПТ. Эта цепь реализует все 1/*n*-тестеры с *n* = φ(1). На рис. 2.30 в кружках указаны все необходимые для построения 3/10-СПТ блоки.

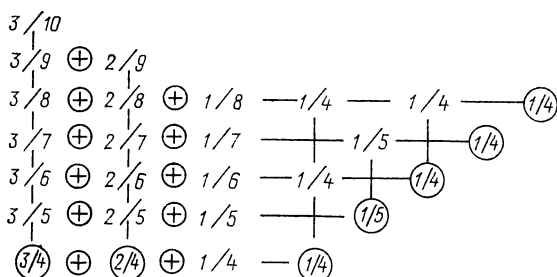


Рис. 2.30

Стандартная блочная реализация *m/n*-СПТ требует: (*n*—*m*—1)(*m*—1)—1 блоков объединения; по одному 2/4- и 1/5-СПТ; *n*—*m*—3 1/4-СПТ; один 3/4-СПТ, если *m* = 2; *m*—3 3/4-СПТ, если *m* ≥ 4; один 4/5-СПТ, если *m* ≥ 4. Для нее *r* = 3*n*—9 при *m* = 2, *r* = 3*n* + *m*—10 при *m* ≥ 3, *t* = 4 + (5α₁¹ + 3α₁²) + 5(α₂¹ + 3α₂²) + 2(*n*—*m*—3) + α₃ + 2*b* + α₄, где α₁¹ = 1 и α₁² = 0 при *n*—*m* ≤ 5; α₁¹ = 0 и α₁² = 1 при *n*—*m* ≥ 6; α₂¹ = α₂² = 0 при *m* ≤ 3; α₂¹ = 1 и α₂² = 0 при 3 < *m* ≤ 5; α₂¹ = 0 и α₂² = 1 при *m* ≥ 6; α₃ = 2 при *n*—*m* ≤ 5; α₃ = 4 при *n*—*m* ≥ 6; *b* = 0 при *m* = 2; *b* = 1 при *m* = 3; *b* = *m*—3 при *m* ≥ 4; α₄ = 0 при *m* = 2; α₄ = 2 при 3 ≤ *m* ≤ 5; α₄ = 4 при *m* ≥ 6.

Уменьшение числа блоков ФПН достигается за счет использования инверторов. Тогда имеем ФПН {1/4-, 2/4-, 1/5-СПТ, И2, НЕ}. Тестеры 3/4- и 4/5-СПТ реализуются с помощью 1/4- и 1/5-СПТ при установлении на их входах инверторов. Минимальное число блоков ФПН равно трем: {1/3-СПТ, НЕ, блок объединения (И2, 1/4-СПТ)}.

2.8. Каталоги *m/n*-тестеров

Универсальные методы синтеза *m/n*-СПТ позволяют решать задачу построения тестера для любого кода. Учет особенностей

конкретного кода или некоторого узкого класса кодов позволяет улучшить характеристики тестеров. Выше были рассмотрены следующие классы кодов: $nC1$, $nC(n-1)$ и $2mCm$.

В [62] для кодов $(2m-1)Cm$ и $(2m+1)Cm$ предложен метод, позволяющий строить простые трехуровневые тестеры. Он является модификацией метода [58], который рассмотрен в § 2.2. Множество переменных B разбивается на $m+1$ подмножеств B_1, B_2, \dots, B_{m+1} . Подмножества B_1, B_2, \dots, B_m содержат по два элемента, а B_{m+1} — один. На основе этого разбиения осуществляется разложение базовой функции вида (2.48). Например, для кода 7C3 имеем: $B = \{x_1 \div x_7\}$, $B_1 = \{x_1, x_2\}$, $B_2 = \{x_3, x_4\}$, $B_3 = \{x_5, x_6\}$, $B_4 = \{x_7\}$,

$$f^3(x_1 \div x_7) = P_{2100} \vee P_{2010} \vee P_{2001} \vee P_{1200} \vee P_{1020} \vee P_{1110} \vee \\ \vee P_{1101} \vee P_{1011} \vee P_{0210} \vee P_{0201} \vee P_{0120} \vee P_{0111} \vee P_{0021}. \quad (2.61)$$

С помощью специального алгоритма из элементов данного разложения составляются функции z_1 и z_2 :

$$z_1 = P_{2010} \vee P_{1200} \vee P_{0120} \vee P_{1011} \vee P_{1101} \vee P_{0111},$$

$$z_2 = P_{2100} \vee P_{0210} \vee P_{1020} \vee P_{2001} \vee P_{0201} \vee P_{0021} \vee P_{1110}.$$

Первый уровень тестера содержит $2(m-1)$ двухвходовых элементов, реализующих для каждого множества $B_j = \{x_{i_1}, x_{i_2}\}$ ($j \in \{1, 2, \dots, m+1\}$) функции $f^1(x_{i_1}, x_{i_2})$ и $f^2(x_{i_1}, x_{i_2})$. Второй уровень схемы тестера реализует элементы разложения (2.61) $P_{i_1 i_2 \dots i_{m+1}}$. Число элементов второго уровня определяется как сумма

$$N_1' = 1 + C_{m-1}^1 C_{m-1}^{m-2 \times 1} + C_{m-1}^2 C_{m-2}^{m-2 \times 2} + C_{m-1}^3 C_{m-3}^{m-2 \times 3} + \dots \\ \dots + C_{m-1}^{\lfloor m/2 \rfloor} C_{m-\lfloor m/2 \rfloor}^{m-2 \times \lfloor m/2 \rfloor},$$

где первое слагаемое дает число элементов с m входами, второе — число элементов с $m-1$ входами, третье — число элементов с $m-2$ входами и т. д.

Третий уровень схемы тестера состоит из двух элементов ИЛИ с общим числом входов $L' = N_1'$. Для проверки тестера необходимо $t = N_1'$ кодовых слов.

Рассмотренные тестеры (обозначим их как СПТ15) проигрывают по основным характеристикам многоуровневым тестерам (например, СПТ6), но существенно выигрывают у трехуровневых тестеров СПТ10 и СПТ12. Например, 3/7-тестер типа СПТ15 имеет по сравнению с СПТ10 сложность L в 2,4 раза и длину теста t в 2,7 раза меньше, а по сравнению с СПТ12 — соответственно в 1,5 и 1,7 раза меньше.

В [71] предложен метод синтеза m/n -СПТ для класса кодов, ограниченного неравенствами: $m \geq 3$, $4m \geq n \geq 2m$. Осуществляется преобразование кодов по схеме $nCm \rightarrow 4C2 \rightarrow 2C1$. Идея построения преобразователя $nCm \rightarrow 4C2$ состоит в следующем. К базовой функции применяется разложение вида (2.33), при-

чем $k = \lfloor n/2 \rfloor$. Определяются числа $a_1 = \lfloor m/2 \rfloor$ и $a_2 = m - a_1$. Например, для кода 8C3 имеем $k = 4$, $a_1 = 1$, $a_2 = 2$,

$$\begin{aligned} f^3(x_1 \div x_8) &= f^3(x_1 \div x_4) \vee f^2(x_1 \div x_4) f^1(x_5 \div x_8) \vee \\ &\vee f^1(x_1 \div x_4) f^2(x_5 \div x_8) \vee f^3(x_5 \div x_8). \end{aligned}$$

К элементам правой части полученного выражения типа f^m повторно применяется разложение (2.33):

$$\begin{aligned} f^3(x_1 \div x_4) &= f^2(x_1, x_2) f^1(x_3, x_4) \vee f^1(x_1, x_2) f^2(x_3, x_4), \\ f^3(x_5 \div x_8) &= f^2(x_5, x_6) f^1(x_7, x_8) \vee f^1(x_5, x_6) f^2(x_7, x_8). \end{aligned}$$

В элементах типа $f^{m-1} f^1$ (кроме элемента $f^{a_1} f^{a_2}$) разложение (2.33) применяется к одной из функций произведения (имеющей больший верхний индекс):

$$\begin{aligned} f^2(x_1 \div x_4) f^1(x_5 \div x_8) &= f^2(x_1, x_2) f^1(x_5 \div x_8) \vee f^1(x_1, x_2) f^1(x_3, x_4) \times \\ &\times f^1(x_5 \div x_8) \vee f^2(x_3, x_4) f^1(x_5 \div x_8). \end{aligned}$$

С помощью специального алгоритма составляются функции $y_1 \div y_4$, описывающие преобразователь $nCm \rightarrow 4C2$, в виде:

$$\begin{aligned} y_1 &= f^{a_1} \vee y_1^*, & y_3 &= y_3^*, \\ y_2 &= f^{a_2} \vee y_2^*, & y_4 &= y_4^*, \end{aligned}$$

где функции $y_1^* \div y_4^*$ образуются из элементов, получаемых на втором этапе разложения:

$$\begin{aligned} y_1 &= f^1(x_1 \div x_4), \\ y_2 &= f^2(x_5 \div x_8) \vee f^1(x_1, x_2) f^2(x_3, x_4), \\ y_3 &= f^1(x_5, x_6) f^2(x_7, x_8) \vee f^2(x_1, x_2) f^1(x_3, x_4) \vee \\ &\vee f^1(x_1, x_2) f^1(x_3, x_4) f^1(x_5 \div x_8), \\ y_4 &= f^2(x_5, x_6) f^1(x_7, x_8) \vee f^2(x_1, x_2) f^1(x_5 \div x_8) \vee f^2(x_3, x_4) f^1(x_5 \div x_8). \end{aligned}$$

Тестер состоит из трех последовательно включенных блоков: блока S , реализованного по тупиковым представлениям функций; блока, реализующего функции $y_1 \div y_4$, и 2/4-СПТ. По своим характеристикам данные тестеры примерно соответствуют СПТ6, обозначим их как СПТ16.

Анализ конкретных кодов позволяет получать структуры m/n -СПТ с улучшенными характеристиками. Рассмотрим, например, синтез $2/n$ -СПТ. Для случая $6 \leq n \leq 8$ осуществляется преобразование $nC2 \rightarrow 4C2 \rightarrow 2C1$. К базовой функции применяется разложение (2.33) ($k = \lfloor n/2 \rfloor$):

$$\begin{aligned} f^2(x_1, \dots, x_k, \dots, x_n) &= f^2(x_1, \dots, x_k) \vee \\ &\vee f^1(x_1, \dots, x_k) f^1(x_{k+1}, \dots, x_n) \vee f^2(x_{k+1}, \dots, x_n). \end{aligned}$$

Повторное преобразование применяется к функциям $f^2(x_1, \dots, x_k)$ и $f^2(x_{k+1}, \dots, x_n)$, причем $l = \lfloor k/2 \rfloor$ и $p = k +$

$+] (n-k)/2 [$. Функции $y_1 \div y_4$, описывающие преобразователь $nC2 \rightarrow 4C2$, имеют вид:

$$\begin{aligned} y_1 &= f^1(x_1, \dots, x_k), \quad y_2 = f^1(x_{k+1}, \dots, x_n), \\ y_3 &= f^1(x_1, \dots, x_l) f^1(x_{l+1}, \dots, x_k) \vee f^1(x_{k+1}, \dots, x_p) f^1(x_{p+1}, \dots, x_n), \\ y_4 &= f^2(x_1, \dots, x_l) \vee f^2(x_{l+1}, \dots, x_k) \vee f^2(x_{k+1}, \dots, x_p) \vee \\ &\quad \vee f^2(x_{p+1}, \dots, x_n). \end{aligned} \quad (2.62)$$

Код $5C2$ представляет собой исключение. Преобразователь $5C2 \rightarrow 4C2$ описывается функциями:

$$\begin{aligned} y_1 &= f^1(x_1, x_2), \quad y_2 = f^2(x_3, x_4), \\ y_3 &= f^1(x_1, x_2) f^1(x_5) \vee f^1(x_3, x_4) f^1(x_5), \\ y_4 &= f^2(x_1, x_2) \vee f^2(x_3, x_4). \end{aligned} \quad (2.63)$$

Для случая $n \geq 9$ осуществляется преобразование $nC2 \rightarrow (d+2)C2 \rightarrow 2C1$, где $d = \lfloor n/4 \rfloor$. В этом случае множество переменных B разбивается на d подмножеств: $B_1 = \{x_1, \dots, x_{k_1}\}$, $B_2 = \{x_{k_1+1}, \dots, x_{k_2}\}$, $B_3 = \{x_{k_2+1}, \dots, x_{k_3}\}$, ..., $B_{d-1} = \{x_{k_{d-2}+1}, \dots, x_{k_{d-1}}\}$, $B_d = \{x_{k_{d-1}+1}, \dots, x_n\}$ таким образом, чтобы каждое подмножество содержало три или четыре элемента. Затем каждое из этих подмножеств, в свою очередь, разбивается на два подмножества, содержащих один или два элемента.

В соответствии с полученными разбиениями составляются функции, описывающие преобразователь $nC2 \rightarrow (d+2)C2$:

$$\begin{aligned} y_1 &= f^1(x_1, \dots, x_{k_1}), \\ y_2 &= f^1(x_{k_1+1}, \dots, x_{k_2}), \\ &\quad \vdots \\ y_{d-1} &= f^1(x_{k_{d-2}+1}, \dots, x_{k_{d-1}}), \\ y_d &= f^1(x_{k_{d-1}+1}, \dots, x_n), \\ y_{d+1} &= f^1(x_1, x_2) f^1(x_3, x_{k_1}) \vee f^1(x_{k_1+1}, x_{k_1+2}) \wedge \\ &\quad \wedge f^1(x_{k_1+3}, x_{k_2}) \vee f^1(x_{k_2+1}, x_{k_2+2}) f^1(x_{k_2+3}, x_{k_3}) \vee \\ &\quad \vee \dots \vee f^1(x_{k_{d-1}+1}, x_{k_{d-1}+2}) f^1(x_{k_{d-1}+3}, x_n), \\ y_{d+2} &= f^2(x_1, x_2) \vee f^2(x_3, x_{k_1}) \vee f^2(x_{k_1+1}, x_{k_1+2}) \vee \\ &\quad \vee f^2(x_{k_1+3}, x_{k_2}) \vee \dots \vee f^2(x_{k_{d-1}+1}, x_{k_{d-1}+2}) \vee \\ &\quad \vee f^2(x_{k_{d-1}+3}, x_n). \end{aligned} \quad (2.64)$$

Пример 2.11. Рассмотрим код $13C2$. Для него $d = \lfloor 13/4 \rfloor = 4$. Множество $B = \{x_1 \div x_{13}\}$ делим на $d = 4$ подмножеств: $B_1 = \{x_1 \div x_4\}$, $B_2 = \{x_5 \div x_7\}$, $B_3 = \{x_8 \div x_{10}\}$, $B_4 = \{x_{11} \div x_{13}\}$. Каждое из них разбиваем на два подмножества:

$B_{11} = \{x_1, x_2\}$, $B_{12} = \{x_3, x_4\}$, $B_{21} = \{x_5, x_6\}$, $B_{22} = \{x_7\}$, $B_{31} = \{x_8, x_9\}$; $B_{32} = \{x_{10}\}$, $B_{41} = \{x_{11}, x_{12}\}$, $B_{42} = \{x_{13}\}$. С помощью системы (2.64) находим функции, описывающие преобразователь $13C2 \rightarrow 6C2$:

$$\begin{aligned} y_1 &= f^1(x_1 \div x_4), & y_3 &= f^1(x_8 \div x_{10}), \\ y_2 &= f^2(x_5 \div x_7), & y_4 &= f^1(x_{11} \div x_{13}), \\ y_5 &= f^1(x_1, x_2)f^1(x_3, x_4) \vee f^1(x_5, x_6)f^1(x_7) \vee \\ &\vee f^1(x_8, x_9)f^1(x_{10}) \vee f^1(x_{11}, x_{12})f^1(x_{13}), \\ y_6 &= f^2(x_1, x_2) \vee f^2(x_3, x_4) \vee f^2(x_5, x_6) \vee f^2(x_8, x_9) \vee f^2(x_{11}, x_{13}). \end{aligned}$$

Тестеры данного типа (СПТ17) представляются в виде последовательного соединения преобразователя $nC2 \rightarrow (d+2)C2$ и $2/(d+2)$ -СПТ. В качестве последнего может быть использован любой $2/n$ -СПТ, не требующий для своей проверки всего множества кодовых слов. Преобразователь $nC2 \rightarrow (d+2)C2$ реализуется в виде трехуровневой схемы с характеристиками: $L = 5(n-d)$, $N_1 = 2(n-d+1)$. Если в качестве $2/(d+2)$ -СПТ использовать СПТ15, то для произвольного n $2/n$ -тестер строится на основе стандартной схемы преобразования:

$$nC2 \rightarrow n^1C2 \rightarrow n^2C2 \rightarrow n^3C2 \rightarrow \dots \rightarrow 4C2 \rightarrow 2C1,$$

где $n^1 =]n/4[+2$, $n^2 =]n^1/4[+2$, $n^3 =]n^2/4[+2$ и т. д. Это дает возможность разработать для $2/n$ -СПТ ФПН, состоящий только из одного блока, реализующего один этап указанного преобразования.

Наиболее полно можно учесть особенности кода при рассмотрении конкретного m/n -тестера. Очевидно, что не существует такого m/n -СПТ, у которого все рассмотренные выше десять характеристик являются минимальными. Тем более что характеристики тестеров находятся между собой в определенной зависимости. Увеличение быстродействия тестера связано с увеличением его сложности. Уменьшение длины проверяющего теста t может быть достигнуто при уменьшении сложности тестера, что ведет к уменьшению быстродействия. Достижение минимального коэффициента симметрии k в большинстве случаев связано с увеличением сложности тестеров. При построении тестеров на ПЛМ не наблюдается строгой зависимости между числом матриц h и их площадью S . Однако, как правило, уменьшение S может быть достигнуто за счет увеличения h .

Для конкретного m/n -СПТ целесообразно строить каталог тестеров, в который включать только те из них, которые обладают наилучшими характеристиками или наилучшими сочетаниями определенных характеристик. При этом все характеристики разделяются на две группы: основные (L , N_1 , N_2 , r_1 , r_2 , t , k) и характеристики ПЛМ (L , S , r_3 , t , k), сравнение по которым производится отдельно.

Процесс составления каталога проиллюстрируем на примере 2/6-СПТ. Для него имеют место следующие минимальные значения характеристик: $r_1 = 3$, $t = 6$, $k = 1,14$, $h = r_3 = 2$. Число слов кода 6С2 $b = 15$. В табл. 2.24 представлены характеристики 2/6-СПТ, реализованных в виде рассмотренных выше тестеров.

Таблица 2.24

Тип тестера	Х а р а к т е р и с т и к и									
	L	N_1	N_2	r_1	r_2	t	k	h	S	r_3
СПТ4	113	37	76	6	10	15	1,14	—	—	—
СПТ5	42	19	23	5	7	9	2,0	—	—	—
СПТ6	36	18	18	6	6	6	1,5	—	—	—
СПТ7	37	18	19	5	6	6	1,14	—	—	—
СПТ8	36	18	18	6	6	6	2,75	—	—	—
СПТ9	63	20	43	3	6	15	1,5	2	192	2
СПТ11	44	17	27	3	6	12	1,5	3	163	2
СПТ12	48	17	31	3	6	12	1,5	3	129	2
СПТ13	65	29	36	5	7	15	1,14	4	202	3
СПТ14	78	27	51	4	6	15	1,14	2	186	2
СПТ17	32	16	16	5	5	6	4,0	—	—	—
СПТ18	51	19	32	3	6	13	1,14	3	184	2
СПТ19	55	19	33	3	6	13	1,14	3	146	2

В тестерах Т5—Т8, реализованных в соответствии с преобразованием $6C2 \rightarrow 4C1 \rightarrow 2C1$, в качестве 1/4-СПТ использованы тестеры типа СПТ2 (см. рис. 2.4). Проведем анализ таблицы. Тестеры Т4 и Т5 проигрывают по всем характеристикам тестеру Т7 и, следовательно, не включаются в каталог. Тестер Т6 имеет наименьшую сложность L , N_1 и N_2 при наилучшем значении $k = 1,5$ (все тестеры со значениями $k \leq 1,5$ сложнее, чем Т6). Тестер Т7 имеет наименьшую сложность L , N_1 и N_2 при минимальном значении k . Тестер Т8 проигрывает по всем характеристикам Т6. Тестер Т9 проигрывает по основным характеристикам Т11, а по характеристикам ПЛМ — Т14. Тестер Т11 имеет наименьшую сложность L , N_1 и N_2 при минимальном быстродействии ($r_1 = 3$). Тестер Т12 имеет наименьшее значе-

ние характеристики S . Тестер T13 проигрывает по основным характеристикам T7, а по характеристикам ПЛМ — T12. Тестер T14 имеет наименьшее значение характеристики S при минимальном значении h . Тестер T17 имеет наименьшие значения характеристик L , N_1 , N_2 и r_2 .

В результате проведенного анализа можно составить каталог 2/6-тестеров, представляющий собой множество $\{T6, T7, T11, T12, T14, T17\}$. Можно сделать также следующие выводы. Получены модификации 2/6-СПТ с минимальными значениями характеристик r_1 , t , k , h и r_3 . Минимальные значения характеристик сложности L , N_1 и N_2 неизвестны. Однако можно предположить, что значения $L = 32$ и $N_1 = N_2 = 16$, достигнутые в тестере T17, являются минимальными или близки к ним. Дальнейшее улучшение характеристик 2/6-СПТ связано с поиском таких структур тестеров, которые обладают лучшими сочетаниями характеристик, чем известные тестеры. Такие тестеры могут быть получены как за счет разработки новых методов синтеза m/n -СПТ, так и за счет изменения структур известных тестеров. Например, недостатком трехуровневых тестеров T9, T11 и T12 является то, что они имеют значение $k = 1,5 > k_{\min} = 1,14$. Тестеры T11 и T12, реализуемые методом [40], строятся исходя из условия получения минимальной сложности (см. рис. 2.27). За счет увеличения сложности можно добиться уменьшения коэффициента k .

Для тестеров T11 и T12

$$k = \frac{b(V)}{C_n^m - b(V)},$$

где $b(V)$ — число элементов множества V . Из данной формулы следует, что для построения тестера с минимальным значением k необходимо построить множество V' , содержащее $\frac{1}{2}C_n^m$ элементов (если C_n^m — четное число), либо $\frac{1}{2}(C_n^m - 1)$ элементов (если C_n^m — нечетное число). Учитывая это, можно модифицировать структуры T11 и T12 (см. рис. 2.27 и систему (2.58)) за счет добавления к множеству V , на основе которого они построены, еще одной конъюнкции. Например, $V' = \{x_1x_2, x_1x_3, x_2x_3, x_4x_5, x_4x_6, x_5x_6, x_2x_5\}$. По данному множеству находим

$$z_1 = x_1x_2 \vee x_1x_3 \vee x_2x_3 \vee x_4x_5 \vee x_4x_6 \vee x_5x_6 \vee x_2x_5, \quad (2.65)$$

$$\begin{aligned} z_2 = & (x_3 \vee x_4 \vee x_5 \vee x_6) (x_2 \vee x_4 \vee x_5 \vee x_6) \times \\ & \times (x_1 \vee x_4 \vee x_5 \vee x_6) (x_1 \vee x_2 \vee x_3 \vee x_6) \times \\ & \times (x_1 \vee x_2 \vee x_3 \vee x_5) (x_1 \vee x_2 \vee x_3 \vee x_4) \wedge \\ & \wedge (x_1 \vee x_3 \vee x_4 \vee x_6). \end{aligned} \quad (2.66)$$

В результате эквивалентных преобразований получаем следующие функции:

$$\begin{aligned} y_1 &= x_3 \vee x_4, \quad y_2 = x_2 \vee x_6, \quad y_3 = x_1 \vee x_5, \\ y_4 &= x_2 \vee x_3, \quad y_5 = x_5 \vee x_6, \end{aligned} \quad (2.67)$$

$$\left. \begin{aligned} z_1 &= x_1 y_4 \vee x_2 y_2 \vee x_4 y_5 \vee x_5 x_6 \vee x_2 x_5, \\ z_2 &= (y_1 \vee y_5) (y_2 \vee x_4 \vee x_5) (y_3 \vee x_4 \vee x_6) (x_1 \vee x_2 \vee y_2) \wedge \\ &\quad \wedge (y_3 \vee y_4) (x_1 \vee x_2 \vee y_1) (x_1 \vee x_6 \vee y_1). \end{aligned} \right\} \quad (2.68)$$

По выражениям (2.67) и (2.68) реализуется тестер Т18, а по выражениям (2.65) и (2.68) — тестер Т19 (см. табл. 2.24). Тестер Т18 имеет наименьшую сложность (L , N_1 и N_2), а Т19 — наименьшее значение S при минимальных значениях характеристик k и r_1 . Оба тестера включаются в каталог, в результате чего каталог 2/6-СПТ представляет собой следующее множество:

$$\{\text{T6, T7, T11, T12, T14, T17, T18, T19}\}.$$

Приведем описание тестеров, входящих в каталог:

СПТ6: $7 = 1+2$, $8 = 1 \times 2$, $9 = 3+4$, $10 = 3 \times 4$, $11 = 5+6$, $12 = 5 \times 6$, $13 = 7+9$, $14 = 11 \times 13$, $15 = 7 \times 9$, $16 = 8+10$, $17 = 12+14$, $18 = 15+16$, $z_1 = (15+17) (12+18)$, $z_2 = (14+18) \times (16+17)$;

СПТ7: $7 = 1+2$, $8 = 1 \times 2$, $9 = 3+4$, $10 = 3 \times 4$, $11 = 5+6$, $12 = 5 \times 6$, $13 = 8+10+12$, $14 = 7 \times 9$, $15 = 7 \times 11$, $16 = 9 \times 11$, $17 = 13+16$, $18 = 14+15$, $z_1 = (14+17) (16+18)$, $z_2 = (13+18) \times (15+17)$;

тестеры СПТ11, СПТ12, СПТ18, СПТ19 описаны выше;

СПТ14: $7 = 1 \times 3$, $8 = 1 \times 4$, $9 = 2 \times 3$, $10 = 2 \times 4$, $11 = 1 \times 5$, $12 = 1 \times 6$, $13 = 2 \times 5$, $14 = 2 \times 6$, $15 = 3 \times 5$, $16 = 3 \times 6$, $17 = 4 \times 5$, $18 = 4 \times 6$, $19 = 1 \times 2$, $20 = 3 \times 4$, $21 = 5 \times 6$, $22 = 7+8+9+10+11+12+13+14$, $23 = 7+8+9+10+15+16+17+18$, $24 = 11+12+15+16+19+20+21$, $25 = 13+14+17+18+19+20+21$, $z_1 = 22 \times 23 + 22 \times 24 + 23 \times 25$, $z_2 = 22 \times 25 + 23 \times 24 + 24 \times 25$;

СПТ17: $7 = 1+2$, $8 = 1 \times 2$, $9 = 4+5$, $10 = 4 \times 5$, $11 = 3+7$, $12 = 6+9$, $13 = 3 \times 7 + 6 \times 9$, $14 = 8+10$, $z_1 = (11+13) (12+14)$, $z_2 = 11 \times 13 + 12 \times 14$.

Рассмотрим каталоги для основных кодов. Для любого 1/ n -СПТ каталог состоит из двух тестеров (см. табл. 2.10): СПТ2 (имеет наилучшие основные характеристики) и СПТ3 (имеет наилучшие характеристики ПЛМ). Каталог 2/4-СПТ также состоит из двух тестеров, показанных на рис. 2.1 и 2.5. В табл. 2.25 приведен каталог 2/5-СПТ. Тестер СПТ20 получен из СПТ9 в результате эквивалентных преобразований, при этом за счет уменьшения быстродействия достигнуто уменьшение сложности. Описание тестеров:

СПТ9: $6 = 1 \times 5$, $7 = 2 \times 4$, $8 = 1 \times 2$, $9 = 3 \times 4$, $10 = 3 \times 5$, $11 = 2 \times 5$, $12 = 1 \times 4$, $13 = 4 \times 5$, $14 = 1 \times 3$, $15 = 2 \times 3$, $z_1 = 6 + 7 + 8 + 9 + 10$, $z_2 = 11 + 12 + 13 + 14 + 15$;

Таблица 2.25

Тип тестера	Х а р а к т е р и с т и к и									
	L	N_1	N_2	r_1	r_2	t	k	h	S	r_3
СПТ9	30	12	18	2	5	10	1,0	1	70	1
СПТ11	35	12	23	2	6	10	1,0	2	60	1
СПТ12	29	12	17	3	5	9	1,0	4	65	2
СПТ17	28	14	14	5	5	5	1,5	—	—	—
СПТ20	26	12	14	3	4	10	1,0	2	98	2

СПТ11: $6 = 1 \times 2$, $7 = 3 \times 4$, $8 = 1 \times 5$, $9 = 2 \times 3$, $10 = 4 \times 5$,
 $11 = 3 + 4 + 5$, $12 = 1 + 2 + 5$, $13 = 2 + 3 + 4$, $14 = 1 + 4 + 5$, $15 =$
 $= 1 + 2 + 3$, $z_1 = 6 + 7 + 8 + 9 + 10$, $z_2 = 11 \times 12 \times 13 \times 14 \times 15$;

СПТ12: $6 = 2 + 5$, $7 = 2 + 4$, $8 = 2 \times 5$, $9 = 2 \times 4$, $10 = 1 \times 6$,
 $11 = 3 \times 7$, $12 = 4 \times 5$, $13 = 3 + 4 + 8$, $14 = 1 + 5 + 9$, $15 = 1 + 2 + 3$,
 $z_1 = 10 + 11 + 12$, $z_2 = 13 \times 14 \times 15$;

СПТ17: $6 = 1 + 2$, $7 = 1 \times 2$, $8 = 3 + 4$, $9 = 3 \times 4$, $10 = 5 \times 6 + 9$,
 $11 = 5 \times 8 + 7$, $z_1 = (6 + 10)(8 + 11)$, $z_2 = 6 \times 10 + 8 \times 11$;

СПТ20: $6 = 2 + 5$, $7 = 2 + 3$, $8 = 2 + 4$, $9 = 3 + 4$, $10 = 1 \times 6$,
 $11 = 4 \times 7$, $12 = 3 \times 5$, $13 = 5 \times 8$, $14 = 1 \times 9$, $15 = 2 \times 3$, $z_1 =$
 $= 10 + 11 + 12$, $z_2 = 13 + 14 + 15$.

В табл. 2.26 приведен каталог 3/6-СПТ. Тестер СПТ21 получен из СПТ6 путем очевидных эквивалентных преобразований с целью уменьшения сложности. Описание тестеров:

СПТ2 описывается формулами (2.15) и (2.16);

СПТ3 показан на рис. 2.8;

Таблица 2.26

Тип тестера	Х а р а к т е р и с т и к и									
	L	N_1	N_2	r_1	r_2	t	k	h	S	r_3
СПТ2	36	14	22	3	6	8	1,0	4	76	2
СПТ3	30	14	16	3	4	6	2,33	4	96	2
СПТ9	80	22	58	2	6	20	1,0	1	160	1
СПТ11	48	14	34	2	5	13	2,33	2	84	1
СПТ21	30	14	16	4	5	6	1,0	—	—	—

СПТ9: $7 = 1 \times 2 \times 3$, $8 = 1 \times 4 \times 5$, $9 = 1 \times 4 \times 6$, $10 = 1 \times 5 \times 6$,
 $11 = 2 \times 4 \times 5$, $12 = 2 \times 4 \times 6$, $13 = 2 \times 5 \times 6$, $14 = 3 \times 4 \times 5$, $15 =$
 $= 3 \times 4 \times 6$, $16 = 3 \times 5 \times 6$, $17 = 1 \times 2 \times 4$, $18 = 1 \times 2 \times 5$, $19 =$

$= 1 \times 2 \times 6$, $20 = 1 \times 3 \times 4$, $21 = 1 \times 3 \times 5$, $22 = 1 \times 3 \times 6$, $23 = 2 \times 3 \times 4$, $24 = 2 \times 3 \times 5$, $25 = 2 \times 3 \times 6$, $26 = 4 \times 5 \times 6$, $z_1 = 7 + 8 + 9 + 10 + 11 + 12 + 13 + 14 + 15 + 16$, $z_2 = 17 + 18 + 19 + 20 + 21 + 22 + 23 + 24 + 25 + 26$;

СПТ10: $7 = 1 \times 2 \times 3$, $8 = 1 \times 2 \times 4$, $9 = 3 \times 4 \times 5$, $10 = 3 \times 4 \times 6$, $11 = 1 \times 5 \times 6$, $12 = 2 \times 5 \times 6$, $13 = 4 + 5 + 6$, $14 = 3 + 5 + 6$, $15 = 1 + 2 + 6$, $16 = 1 + 2 + 5$, $17 = 2 + 3 + 4$, $18 = 1 + 3 + 4$, $z_1 = 7 + 8 + 9 + 10 + 11 + 12$, $z_2 = 13 \times 14 \times 15 \times 16 \times 17 \times 18$;

СПТ21: $7 = 1 + 2$, $8 = 1 \times 2$, $9 = 4 + 5$, $10 = 4 \times 5$, $11 = 3 \times 7 + 6 \times 10 + 8$, $12 = 3 \times 8 + 6 \times 9 + 10$, $13 = 6 + 9$, $14 = 3 + 8$, $15 = 3 + 7$, $z_1 = 11 \times 13$, $z_2 = 12 \times 15$.

ГЛАВА ТРЕТЬЯ

СХЕМЫ КОНТРОЛЯ И ДЕШИФРАЦИИ КОДОВ

3.1. Самопроверяемые схемы контроля кодов с повторением

Для кода с повторением nPk известен [52] простой метод реализации СПТ. Осуществляется преобразование кода nPk в парафазный код $n\bar{P}k$ путем инвертирования всех контрольных разрядов кода. Для построения $n\bar{P}k$ -СПТ используется специ-

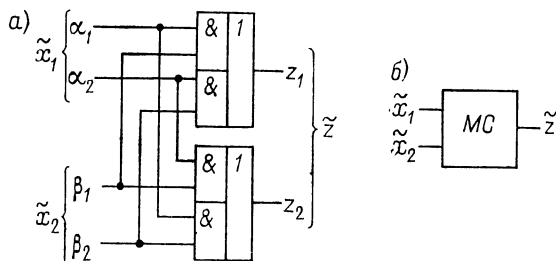


Рис. 3.1

альный модуль сравнения MC (рис. 3.1, а), представляющий собой схему, которая производит проверку на парафазность сигналов в каждой из двух пар входов: $\tilde{x}_1(\alpha_1 \neq \alpha_2)$ и $\tilde{x}_2(\beta_1 \neq \beta_2)$. Выход \tilde{z} исправной схемы MC также является парафазным ($z_1 \neq z_2$). При нарушении парафазности входных сигналов и возникновении одиночных константных неисправностей на выходах MC устанавливаются одинаковые сигналы. Для проверки схемы MC необходимы четыре входных набора $\alpha_1\alpha_2\beta_1\beta_2$:

0101, 0110, 1001, 1010. На рис. 3.1, б показано условное обозначение модуля сравнения.

Контроль парафазности любого числа сигналов $\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_k$ осуществляется с помощью схемы, представляющей собой последовательное или древовидное соединение MC . При этом требуется $k-1$ модуль. На рис. 3.2 показаны структуры $10\overline{P}5$ -СПТ. В таких структурах обеспечивается [45] поступление на входы каждого MC четырех наборов проверяющего теста при подаче всех 2^n слов парафазного кода.

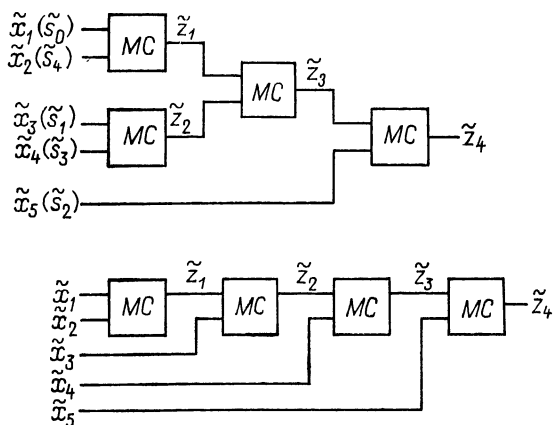


Рис. 3.2

Определим минимальное значение характеристики t для $n\overline{P}k$ -СПТ. Для парафазного сигнала $\tilde{x} = \alpha_1\alpha_2$ введем обозначения: если $\alpha_1\alpha_2 = 01$, то $\tilde{x} = 1$; если $\alpha_1\alpha_2 = 10$, то $\tilde{x} = 0$. Тогда работа MC описывается табл. 3.1, отвечающей таблице истинности логического элемента «сложение по модулю 2» (элемент $M2$). По этой причине с точки зрения обнаружения одиночных неисправностей последовательное соединение MC эквивалентно одноканальной линейной схеме, состоящей из элементов $M2$. Это связано с тем, что в обеих схемах при построении одиночного теста необходимо выполнить только одно условие, состоящее в требовании подачи на входы каждого элемента необходимых четырех наборов. В [43] показано, что минимальный одиночный проверяющий тест линейной схемы при любом числе

Таблица 3.1

\tilde{x}_1	\tilde{x}_2	\tilde{z}
0	0	0
0	1	1
1	0	1
1	1	0

переменных содержит $t = 4$ входных набора. Для построения теста составляется таблица, содержащая n столбцов и четыре строки. Столбцы соответствуют входным переменным $\tilde{x}_1, \dots, \tilde{x}_n$, а строки — наборам входных переменных. Введем обозначения: i — номер столбца; γ_i^q — число (0 или 1), записанное в клетке на пересечении i -го столбца и q -й строки ($q \in \{1, 2, 3, 4\}$). Таблица заполняется по столбцам при помощи следующего алгоритма.

Алгоритм 3.1:

1. Полагаем $i = 1$.
2. Заполняем $\gamma_i^1 = 0, \gamma_i^2 = 0, \gamma_i^3 = 1, \gamma_i^4 = 1$.
3. Полагаем $i = i + 1$.
4. Если $i < n + 1$, то переходим к п. 5. Если $i = n + 1$, переходим к п. 8.
5. Подсчитываем суммы

$$S_i^1 = \left(\sum_{j=1}^{i-1} \gamma_j^1 \right)_{\text{mod } 2}, \quad S_i^2 = \left(\sum_{j=1}^{i-1} \gamma_j^2 \right)_{\text{mod } 2},$$

$$S_i^3 = \left(\sum_{j=1}^{i-1} \gamma_j^3 \right)_{\text{mod } 2}, \quad S_i^4 = \left(\sum_{j=1}^{i-1} \gamma_j^4 \right)_{\text{mod } 2}.$$

6. Если $S_i^q = S_i^p = 0, S_i^r = S_i^t = 1$ ($q, p, r, t \in \{1, 2, 3, 4\}$), то $\gamma_i^q = 0, \gamma_i^p = 1, \gamma_i^r = 0, \gamma_i^t = 1$.
7. Переходим к п. 3.
8. Конец.

В табл. 3.2 приведен пример построения теста для схем рис. 3.2 (в таблицу включены также столбцы \tilde{z}_i , соответствующие промежуточным сигналам схемы).

Таблица 3.2

\tilde{x}_1	\tilde{x}_2	\tilde{x}_3	\tilde{x}_4	\tilde{x}_5	\tilde{z}_1	\tilde{z}_2	\tilde{z}_3	\tilde{z}_4
0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	1	1	1
1	0	1	1	1	1	0	1	0
1	1	1	1	1	0	1	0	1

Недостатком СПТ, реализованных на МС, является их невысокое быстродействие. Быстродействующие двухуровневые тестеры строятся следующим образом [45]. Обозначим через x_i информационную и через x_i' соответствующую ей контрольную переменную парафазного кода ($x_i' = \overline{x_i}$). Каждому слову парафазного кода соответствует конъюнкция M_i ранга n , состоя-

щая из трех переменных, которые принимают значение 1 в этом слове. Обозначим через A_1 множество конъюнкций M_i , содержащих четное число информационных переменных, а через A_2 — нечетное их число. Функции z_1 и z_2 , описывающие $n\bar{P}k$ -СПТ, определяются по следующему правилу: функция z_1 есть дизъюнкция конъюнкций $M_i \in A_1$, а функция z_2 — дизъюнкция конъюнкций $M_i \in A_2$. Например, для кода $6\bar{P}3$ имеем:

$$z_1 = x_1x_2x_3' \vee x_1x_3x_2' \vee x_2x_3x_1' \vee x_1'x_2'x_3',$$

$$z_2 = x_1x_2x_3 \vee x_1x_2'x_3' \vee x_2x_1'x_3' \vee x_3x_1'x_2'.$$

Такие тестеры реализуются на одной ПЛМ.

3.2. Контроль кодов с суммированием

Код с суммированием nSk как разделимый код может контролироваться на основе сравнения информационной части кодового слова с его контрольной частью. В этом случае для кода nSk строится (n, k) -тестер в соответствии со структурной схемой, показанной на рис. 3.3. Тестер содержит блоки A и D . На входы x_1, x_2, \dots, x_k блока A подаются переменные, соответствующие информационным разрядам кода (см. табл. 1.14). Блок A имеет r выходов ($r = n - k$) и осуществляет преобразование вектора, соответствующего информационной части кодового слова, в вектор, соответствующий контрольной части этого слова.

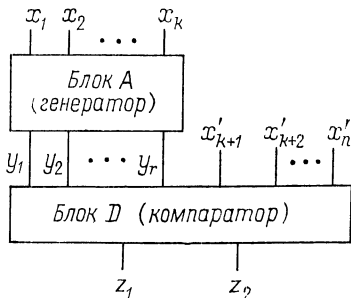


Рис. 3.3

Блок D имеет $2r$ входов и два выхода z_1 и z_2 . На входы $x'_{k+1}, x'_{k+2}, \dots, x'_n$ подаются переменные, соответствующие контрольным разрядам кода. Блок D осуществляет сравнение вектора, сформированного на выходах блока A , с вектором, образуемым r контрольными разрядами кодового слова. Если указанные векторы совпадают, то на выходах z_1 и z_2 присутствуют сигналы (0, 1) или (1, 0), в противном случае — сигналы (0, 0) или (1, 1). В дальнейшем блок A будем называть генератором контрольных разрядов (или просто генератором), а блок D — компаратором.

Рассмотрим способы реализации генератора. На его выходах y_1, y_2, \dots, y_r может быть реализовано либо непосредственно контрольное слово, либо соответствующее ему вспомогательное слово (см. табл. 1.14) с учетом дальнейшего инвертирования каждого его символа. Более целесообразно формировать вспомогательное слово, так как для подсчета числа единиц в контрольном слове можно использовать непосредственно комбина-

ционные схемы сумматоров и полусумматоров [50, 66]. Полный сумматор MS имеет три входа x_1, x_2, x_3 и два выхода S (сумма) и C (перенос), на которых реализуются функции

$$S = x_1(x_2x_3 \vee \bar{x}_2\bar{x}_3) \vee \bar{x}_1(\bar{x}_2x_3 \vee x_2\bar{x}_3),$$

$$C = x_1x_2 \vee x_1x_3 \vee x_2x_3.$$

Полусумматор HS имеет два входа и те же выходы: $S = x_1\bar{x}_2 \vee \bar{x}_1x_2$, $C = x_1x_2$. Сумматор MS позволяет осуществлять логическое сложение разрядов трех, а сумматор HS — двух двоичных чисел.

Код nSk называется полным, если выполняется условие $k = 2^{n-k} - 1$. В полном коде все 2^{n-k} r -разрядных ($r = n - k$) двоичных слов появляются в контрольной части кодовых слов. Если указанные условия не выполняются, то код nSk называется неполным. Это условие является достаточно жестким, и поэтому большинство кодов nSk относится к неполным кодам.

Рассмотрим эффективный метод реализации самопроверяемых генераторов на основе использования модулей MS и HS [66]. Для полных кодов структура генератора определяется с помощью следующего алгоритма.

Алгоритм 3.2:

1. Множество входных информационных переменных $B = \{x_1, \dots, x_k\}$ ($k = 2^r - 1$) разбивается на три подмножества: $B_1 = \{x_1, \dots, x_t\}$ ($t = 2^{r-1} - 1$), $B_2 = \{x_{t+1}, \dots, x_{k-1}\}$ и $B_3 = \{x_k\}$. Если $B = \{x_1, x_2, x_3\}$, то генератор для (5,3)-СПТ представляет собой модуль MS .

2. Множествам B_1 и B_2 присваиваются два одинаковых блока A_1 и A_2 , которые представляют собой генераторы для кода $(t+p)St$, где $p = \lceil \log_2(t+1) \rceil$ определяет число выходов блока. Обозначим через ω_1 и ω_2 p -разрядные двоичные векторы, образуемые на выходах соответственно блоков A_1 и A_2 , а через ω_3 — одноразрядный двоичный вектор на входе x_k .

3. С помощью p модулей MS составляется схема сложения двоичных чисел ω_1, ω_2 и ω_3 .

4. Если $t = 3$, то в качестве блоков A_1 и A_2 устанавливаются модули MS . Если $t > 3$, то блоки A_1 и A_2 реализуются путем повторения п. 1 — 4 данного алгоритма.

Пример 3.1. Рассмотрим код 19S15 ($r = 4$). Множество $B = \{x_1 \div x_{15}\}$ разбиваем на три подмножества: $B_1 = \{x_1 \div x_7\}$, $B_2 = \{x_8 \div x_{14}\}$ и $B_3 = \{x_{15}\}$. На рис. 3. 4 показана структура (19, 15)-СПТ. Так как $t = 2^{4-1} - 1 = 7$ и $p = \lceil \log_2(7+1) \rceil = 3$, то блоки A_1 и A_2 имеют по три выхода. На выходах $y_1' - y_3'$ формируется двоичное число ω_1 , определяющее количество единичных разрядов среди переменных x_1, \dots, x_t , а на выходах $y_1'' - y_3''$ — двоичное число ω_2 , определяющее количество таких же разрядов среди переменных x_{t+1}, \dots, x_{k-1} . На входе x_{15} формируется двоичное одноразрядное число ω_3 . Для сложения двоич-

ных чисел ω_1 , ω_2 и ω_3 устанавливаем три модуля MS , выходы которых связываются с выходами генератора $y_1 - y_4$. Так как $t > 3$, то множество B_1 также разбиваем на три подмножества: $B_{11} = \{x_1 \div x_3\}$, $B_{12} = \{x_4 \div x_6\}$ и $B_{13} = \{x_7\}$. На основе этого разбиения составляем структуру блока A_1 в виде генератора для $(10, 7)$ -СПТ. Блок A_2 идентичен блоку A_1 . Ввиду того что на втором этапе разбиения $t = 3$, процесс формирования генератора структуры $(19, 15)$ -СПТ заканчивается.

Для построения генератора (n, k) -СПТ по алгоритму 3.2 требуется $\sum_{j=1}^{t-1} 2^{j-1}(r-j)$ модулей MS . Число модулей, включенных последовательно, равно $2r-3$. Для проверки генератора

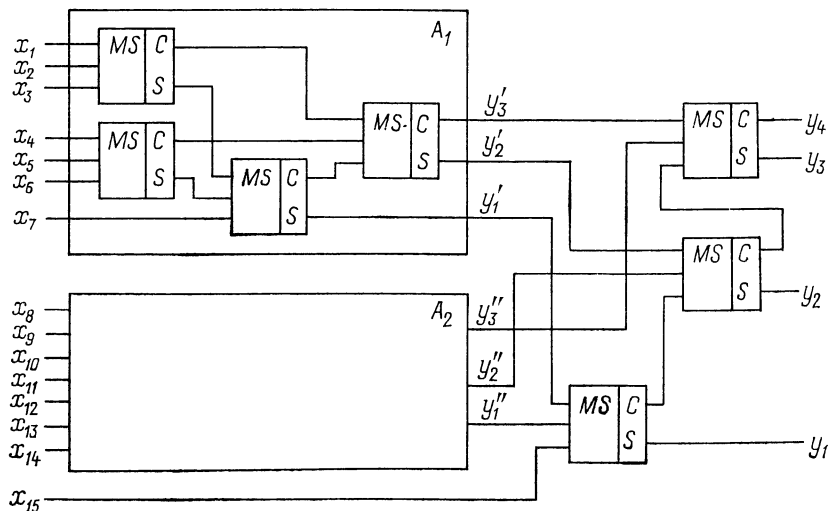


Рис. 3.4

достаточно восьми входных наборов, на которых обнаруживаются все одиночные неисправности входов и выходов модулей MS , а также любые сочетания одиночных неисправностей внутри модулей (при этом на входах каждого модуля возможно обеспечить наличие всех восьми входных наборов). Характеристики генератора зависят от принятой схемы реализации модуля MS .

Для неполных кодов nSk структура генератора реализуется по следующему алгоритму.

Алгоритм 3.3:

1. Множество информационных переменных $B = \{x_1, \dots, x_k\}$ разбивается на q ($q \leq r-1$) подмножеств $B_1 = \{x_1, \dots, x_{t_1}\}$, $B_2 = \{x_{t_1+1}, \dots, x_{t_2}\}$, \dots , $B_{q-1} = \{x_{t_{q-2}+1}, \dots, x_{t_{q-1}}\}$, $B_q = \{x_{t_{q-1}+1}, \dots, x_k\}$ таким образом, чтобы в каждом множестве B_i ($i \in \{1, 2, \dots, q-1\}$) содержалось $2^{r-i}-1$ элементов. Множество B_q вклю-

чает в себя $b_q = k - \sum_{i=1}^{q-1} b_i$ элементов (b_i — число переменных в множестве B_i), причем $b_q \in \{0, 1, 2\}$.

2. Каждому множеству B_i ($i \in \{1, 2, \dots, q-1\}$) сопоставляется блок A_i , построенный с помощью алгоритма 3.2. Обозначим через ω_i двоичный вектор, образуемый на выходах блока A_i , а через ω' и ω'' — одноразрядные двоичные векторы, соответствующие переменным $x \in B_q$.

3. С помощью модулей MS и HS составляется схема сложения двоичных чисел $\omega_1, \omega_2, \dots, \omega_{q-1}, \omega'$ и ω'' .

Пример 3.2. Рассмотрим код 16S12 ($r = 4$). Множество $B = \{x_1 \div x_{12}\}$ разбивается на три подмножества: $B_1 = \{x_1 \div x_7\}$, $B_2 = \{x_8, x_9, x_{10}\}$ и $B_3 = \{x_{11}, x_{12}\}$. Для множества B_1 с помощью алгоритма 3.1 строим блок A_1 (совпадает с блоком A_1 на рис. 3.4). Так как множество B_2 содержит только три переменные, то в качестве блока A_2 устанавливается модуль MS . На выходах y_1', y_2', y_3' блока A_1 формируется трехразрядный двоичный вектор ω_1 , на выходах $y_1'' = S$ и $y_2'' = C$ блока A_2 — двухразрядный вектор ω_2 , а на входах x_{11} и x_{12} — одноразрядные векторы ω' и ω'' . Для этого требуется три модуля MS и два модуля HS . Модуль MS_1 (имеющий выходы S_1 и C_1) суммирует разряды y_1', y_1'' и x_1 , модуль HS_2 — разряды x_{12} и S_1 , модуль MS_3 — разряды y_2', y_2'' и C_1 , модуль HS_4 — разряды C_2 и S_3 , модуль MS_5 — разряды y_3', C_3 и C_4 . Выходы генератора y_1, y_2, y_3 и y_4 соединяются соответственно с выходами модулей S_2, S_4, S_5 и C_5 .

Для проверки генератора, реализованного по алгоритму 3.3, требуется 8 ($r-1$) входных наборов. Необходимое число модулей MS определяется количеством этих модулей, входящих в блоки A_1, A_2, \dots, A_{q-1} и в схему суммирования двоичных чисел, формируемых на выходах этих блоков и на входах генератора x_{k-1}, x_k . Необходимое число модулей HS определяется только модулями, входящими в схему суммирования. Например, при $k = 14$ в структуру генератора входят два генератора для $k = 7$ (требуется по 4 модуля MS на каждый из них) и схема суммирования двух трехразрядных чисел (требуется один модуль HS и два модуля MS).

Второй способ реализации генератора состоит в следующем [26]. Введем обозначения: S_0, S_1, \dots, S_{r-1} — функции, описывающие выходы генератора, при этом функции S_0 и S_{r-1} описывают соответственно выходы, отвечающие младшему x_n^* и старшему x_{k+1}^* разрядам вспомогательного слова (см. табл. 1.14).

Анализ таблиц, задающих код nSk (например, табл. 1.14), показывает, что функции вида S_i ($i \in \{0, 1, \dots, r-1\}$) могут быть вычислены по следующей формуле:

$$S_i = f^{m_1} \overline{f^{m_1+2^i}} \vee f^{m_2} \overline{f^{m_2+2^i}} \vee \dots \vee f^a \overline{f^b}, \quad (3.1)$$

Пример 3.3. Получим функции, описывающие блок A_1^3 для (9, 6)-СПТ. Определяем множество V ($B = \{x_1 \div x_6\}$). На первом этапе разбиения получаем: $B_1^1 = \{x_1 \div x_4\}$, $B_2^1 = \{x_5, x_6\}$. На втором этапе разбиваем множество B_1^1 : $B_1^2 = \{x_1, x_2\}$, $B_2^2 = \{x_3, x_4\}$. На основании первого этапа разбиения вычисляем основные представления функций $f^1 \div f^6$:

$$\begin{aligned} f^1(x_1 \div x_6) &= f^1(x_1 \div x_4) \vee f^1(x_5, x_6), \\ f^2(x_1 \div x_6) &= f^2(x_1 \div x_4) \vee f^1(x_1 \div x_4) f^1(x_5, x_6) \vee f^2(x_5, x_6), \\ f^3(x_1 \div x_6) &= f^3(x_1 \div x_4) \vee f^2(x_1 \div x_4) f^1(x_5, x_6) \vee f^1(x_1 \div x_4) f^2(x_5, x_6), \\ f^4(x_1 \div x_6) &= f^4(x_1 \div x_4) \vee f^3(x_1 \div x_4) f^1(x_5, x_6) \vee f^2(x_1 \div x_4) f^2(x_5, x_6), \\ f^5(x_1 \div x_6) &= f^4(x_1 \div x_4) f^1(x_5, x_6) \vee f^3(x_1 \div x_4) f^2(x_5, x_6), \\ f^6(x_1 \div x_6) &= f^4(x_1 \div x_4) f^2(x_5, x_6). \end{aligned} \quad (3.5)$$

На основании второго этапа разбиения вычисляем функции, входящие в правые части вышеприведенных равенств:

$$\begin{aligned} f^1(x_1 \div x_4) &= f^1(x_1, x_2) \vee f^1(x_3, x_4), \\ f^2(x_1 \div x_4) &= f^2(x_1, x_2) \vee f^1(x_1, x_2) f^1(x_3, x_4) \vee f^2(x_3, x_4), \\ f^3(x_1 \div x_4) &= f^2(x_1, x_2) f^1(x_3, x_4) \vee f^1(x_1, x_2) f^2(x_3, x_4), \\ f^4(x_1 \div x_4) &= f^2(x_1, x_2) f^2(x_3, x_4). \end{aligned} \quad (3.6)$$

Генератор реализуется в виде последовательного соединения трех подсхем. Первая подсхема (один уровень элементов) реализует элементарные функции $f^1(x_1, x_2)$, $f^2(x_1, x_2)$, $f^1(x_3, x_4)$, $f^2(x_3, x_4)$, $f^1(x_5, x_6)$ и $f^2(x_5, x_6)$. Вторая подсхема (два уровня элементов) реализует систему функций (3.6), а третья (два уровня элементов) — систему функций (3.5).

Сложность блока A_1^3 вычисляется по формуле, которая непосредственно следует из формулы (2.42):

$$\begin{aligned} L(A_1^3) &= 4h_2 + 8h_3 + \sum_{t \in \{4, 6, 8, 10, \dots\}} h_t(a_1 + a_2 + \dots + \\ &+ a_{t/2-1} + a_{t/2} + a_{t/2+1} + a_{t/2+2} + \dots + a_{t-1} + a_t) + \\ &+ \sum_{t \in \{5, 7, 9, 11, \dots\}} h_t(b_1 + b_2 + \dots + b_{t/2-1/2} + b_{t/2+1/2} + \\ &+ b_{t/2+3/2} + b_{t/2+5/2} + \dots + b_{t-1} + b_t), \end{aligned} \quad (3.7)$$

где h_l ($l \in \{2, 3, \dots\}$) — число подмножеств переменных с l элементами, входящих в базовое множество V ; $a_1 = a_t = b_1 = b_t = 2$, $a_i = a_{i-1} + 3$ ($i \in \{2, 3, \dots, t/2-1\}$); $a_{t/2} = a_{t/2-1} + 3$, $a_{t/2+1} = a_{t/2} + 1$ и $a_{t/2+2} = a_{t/2+1} - 3$, если $t/2 = \varphi(2)$; $a_{t/2} = a_{t/2-1} + 2$, $a_{t/2+1} = a_{t/2}$ и $a_{t/2+2} = a_{t/2+1} - 1$, если $t/2 = \varphi(1)$; $a_j = a_{j-1} - 3$ ($j \in \{t/2+3, t/2+4, \dots, t-1\}$); $b_i = b_{i-1} + 3$ ($i \in$

$\in \{2, 3, \dots, t/2 - 1/2\}$), $b_{t/2+1/2} = b_{t/2-1/2} + 2$, $b_{t/2+3/2} = b_{t/2+1/2} - 1$; $b_j = b_{j-1} - 3$ ($j \in \{t/2 + 5/2, t/2 + 7/2, \dots, t-1\}$).

Длина проверяющего теста вычисляется по формуле

$$t_k(A_1^3) = t'_k + 1, \quad (3.8)$$

где

$$t'_k = k^2/4 + k, \text{ если } k = \varphi(2) \text{ и } k/2 = \varphi(2);$$

$$t'_k = (k^2 - 4)/4 + k, \text{ если } k = \varphi(2) \text{ и } k/2 = \varphi(1);$$

$$t'_k = (k^2 - 1)/4 + k, \text{ если } k = \varphi(1).$$

Общее число уровней блока A_1^3

$$r(A_3) = 2 \lceil \log_2 k \rceil - 1. \quad (3.9)$$

Используя формулы (3.7) — (3.9), рассчитаем характеристики рассмотренного в примере 3.3 блока A_1^3 для (9, 6)-СПТ. Базовое множество V включает в себя подмножества, содержащие 2, 4 и 6 элементов, причем $h_2 = 3$, $h_4 = 1$ и $h_6 = 1$. Поэтому имеем $L(A_1^3) = 4h_2 + h_4(a_1 + a_2 + a_3 + a_4) + h_6(a_1 + a_2 + a_3 + a_4 + a_5 + a_6) = 4 \times 3 + 1 \times (2 + 5 + 6 + 2) + 1 \times (2 + 5 + 7 + 7 + 6 + 2) = 56$. Так как $k = 6 = \varphi(2)$ и $k/2 = 3 = \varphi(1)$, то $t'_k = (6^2 - 4)/4 + 6 = 14$, $t_k(A_1^3) = 14 + 1 = 15$. Кроме того, $r(A_3) = 2 \lceil \log_2 6 \rceil - 1 = 5$.

На рис. 3.5 представлена схема блока A_2 для (9, 6)-СПТ.

Сложность блока A_2 может быть вычислена либо непосредственно по системе функций (3.1), описывающих блок, либо с помощью следующих формул:

$$L_k(A_2) = \sum_{i=0}^{r-1} L_k(S_i) + \{k/2\}_-,$$

$$L_k(S_i) = 2g - \{g/2\}_- + 1, \quad (3.10)$$

$$g = \{g'/2^i\}_+, \quad g' = k - 2^{i+1} + 1,$$

где $\{a\}_-$ и $\{a\}_+$ — обозначения соответственно ближайшего меньшего и ближайшего большего к a целого числа, при этом, если $a < 0$, то $\{a\}_- = \{a\}_+ = 0$; если $a = 1/2$, то $\{a\}_- = \{a\}_+ = 1$, если a — целое число, то $\{a\}_- = \{a\}_+ = a$.

Из формулы (3.1) и рис. 3.5 следует, что схема блока A_2 представляет собой трехуровневую схему с инверторами на входах. В проверяющий тест схемы блока A_2 входит набор, содержащий нули во всех k разрядах ($f^1 = f^2 = \dots = f^k = 0$) и k

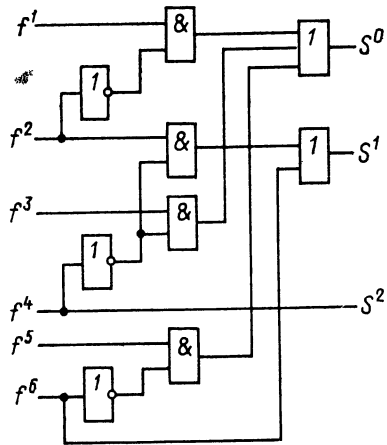


Рис. 3.5

наборов (следовательно, $t = k + 1$) следующего вида (последовательность разрядов $f^1 f^2 f^3 \dots f^k$):

100...000
 110...000
 111...000

 111...100
 111...110
 111...111

Данное множество наборов генерируется на выходах блоков A_1 всех трех типов при подаче на их входы проверяющего теста. Поэтому проверка блока A_1 гарантирует и проверку блока A_2 .

Блок A имеет три разновидности в зависимости от того, какой из типов блока A_1 используется в его структуре. Блок A^1 , содержащий подблок A_1^1 , имеет следующие характеристики:

$$L(A^1) = L(A_1^1) + L_k(A_2), \quad r(A^1) = 5, \quad t(A^1) = t(A_1^1).$$

Блок A^2 имеет характеристики:

$$L(A^2) = 2k(k-1) + L_k(A_2), \quad r(A^2) = 2k, \quad t(A^2) = 2k.$$

Блок A^3 имеет характеристики:

$$L(A^3) = L(A_1^3) + L_k(A_2), \quad r(A^3) = 2] \log_2 k [+2, \quad t(A^3) = t_k(A_1^3).$$

Рассмотрим способы реализации компаратора. Так как компаратор (см. рис. 3.3) осуществляет сравнение двух одинаковых двоичных векторов, образованных контрольными разрядами $x'_{k+1}, x'_{k+2}, \dots, x'_n$ кода nSk и выходными сигналами y_1, y_2, \dots, y_r генератора, то он фактически выполняет функции схемы контроля кода с повторением. Ввиду того что на выходе описанных генераторов реализуется вспомогательное слово, в качестве компараторов используются тестеры для парафазного кода, рассмотренные в § 3.1.

Определим структуру компаратора, которую целесообразно использовать в (n, k) -СПТ в сочетании с рассмотренными генераторами. Компаратор реализуется в виде пирамидальной схемы из модулей сравнения МС (см. рис. 3.1, а). Так как входы компаратора связаны только с выходами генератора, то необходимо выбрать структуру компаратора таким образом, чтобы множество выходных векторов генератора, образующихся на словах проверяющего его теста, составляло проверяющий тест компаратора. В этом плане следует выделить такие (n, k) -СПТ, у которых $k = 2^{n-k-1}$. В них невозможно осуществить полную проверку компаратора, так как код nSk в этом случае содержит только одно вспомогательное слово, в котором старший разряд

равен 1. Например, код 7C4 содержит следующие вспомогательные слова $x_5'x_6'x_7'$: 000, 001, 010, 011, 100. Компаратор (7, 4)-СПТ состоит из двух последовательно соединенных модулей сравнения $MC1$ и $MC2$. На выходах генератора (7, 4)-СПТ в соответствии с (3.1) реализуются функции S_0 , S_1 и S_2 , соответствующие переменным x_7' , x_6' и x_5' . На входы $MC1$ подаются парафазные сигналы $\tilde{S}_0(S_0 = \bar{x}_7')$ и $\tilde{S}_1(S_1 = x_6')$, при этом обеспечивается поступление всех четырех указанных для модуля наборов проверяющего теста (во вспомогательных словах присутствуют все четыре возможных комбинации разрядов x_6' и x_7'). На входы $MC2$ подаются парафазный сигнал с выхода $MC1$ и парафазный сигнал $\tilde{S}_3(S_3 \neq \bar{x}_5')$. Так как единичное значение переменной x_5' встречается только в одном вспомогательном слове, то невозможно в этом случае обеспечить поступление на входы $MC2$ всех четырех наборов проверяющего теста. Поэтому для указанного класса кодов тестер не может быть реализован в виде структуры рис. 3.3.

Для всех (n, k) -СПТ, у которых $k \neq 2^{n-k-1}$, структуры, соответствующие рис. 3.3, существуют. Для случая, когда $r = n - k = \varphi(2)$, компараторы реализуются в виде правильной пирамиды, нижний ряд которой состоит из $(r-1)/2$ модулей МС. При этом на объединяемые по выходам пары МС подаются следующие парафазные сигналы: $(\tilde{S}_0, \tilde{S}_{r-1}), (\tilde{S}_1, \tilde{S}_{r-2}), \dots, (\tilde{S}_{r/2-2}, \tilde{S}_{r/2+1}), (\tilde{S}_{r/2-1}, \tilde{S}_{r/2})$. Для случая, когда $r = n - k = \varphi(1)$, компаратор состоит из двух подсхем. Первая подсхема представляет собой правильную пирамиду с нижним рядом из $(r-1)/2$ модулей МС. В этом случае на объединяемые по выходам пары МС подаются следующие сигналы: $(\tilde{S}_0, \tilde{S}_{r-1}), (\tilde{S}_1, \tilde{S}_{r-2}), \dots, (\tilde{S}_{(r-1)/2-2}, \tilde{S}_{(r-1)/2+2}), (\tilde{S}_{(r-1)/2-1}, \tilde{S}_{(r-1)/2+1})$. Вторая подсхема состоит из одного модуля МС, один вход которого соединяется с выходом первой подсхемы, а на второй вход подается парафазный сигнал $\tilde{S}_{(r-1)/2}$. В описанных структурах обеспечивается подача на входы всех МС четырех наборов проверяющего теста, так как на выходах генераторов при поступлении на их входы проверяющих тестов формируются двоичные векторы, соответствующие десятичным числам от 0 до n .

Наиболее неблагоприятным является случай, когда $k = 2^{n-k-1} - 1$. К нему относится, например, (22, 17)-СПТ. Компаратор для данного тестера совпадает со структурой, показанной на рис. 3.2, а (входные парафазные сигналы указаны в скобках). В табл. 3.3 приведены проверяющий тест и значения сигналов на его наборах во внутренних точках схемы компаратора. Из таблицы видно, что на входы всех МС поступают четыре набора проверяющего теста. Для обеспечения полной проверки

Таблица 3.3

\tilde{S}_4	\tilde{S}_3	\tilde{S}_2	\tilde{S}_1	\tilde{S}_0	\tilde{z}_1	\tilde{z}_2	\tilde{z}_3
0	0	0	0	0	1	1	1
0	0	0	0	1	0	1	0
0	0	0	1	0	1	0	0
0	0	0	1	1	0	0	1
0	0	1	0	0	1	1	1
0	0	1	0	1	0	1	0
0	0	1	1	0	1	0	0
0	0	1	1	1	0	0	1
0	1	0	0	0	1	0	0
0	1	0	0	1	0	0	1
0	1	0	1	0	1	1	0
0	1	0	1	1	0	1	1
0	1	1	0	0	1	0	0
0	1	1	0	1	0	0	1
0	1	1	1	0	1	1	1
0	1	1	1	1	0	1	0
1	0	0	0	0	1	1	1
1	0	0	0	1	0	1	0

МС нижнего уровня необходимо выполнить только одно условие: на входы одного МС надо подать переменные \tilde{S}_0 и \tilde{S}_{r-1} , соответствующие младшему и старшему разрядам кодовых слов, образующихся на выходах генератора. При этом, если переменная \tilde{S}_{r-1} (переменная \tilde{S}_4 в табл. 3.3) принимает значение 1 только в двух наборах проверяющего теста (случай $k = 2^{n-k-1} + 1$), то на этих наборах переменная \tilde{S}_0 принимает противоположные значения. Этим обеспечивается поступление на входы МС четырех различных входных наборов. Для всех остальных

ных МС нижнего уровня поступление этих наборов обеспечивается при любом сочетании остальных входных переменных $\tilde{S}_1, \dots, \tilde{S}_{r-2}$. Поступление тестов на входы МС других уровней обеспечивается тем сочетанием входных переменных, которое указано выше.

Описанные компараторы содержат по $n-k-1$ МС. Число МС, включенных последовательно в схеме компаратора, равно $\lceil \log_2 (n-k) \rceil$. Учитывая, что $L(МС) = 12$ и $r(МС) = 2$ (см. рис. 3.1), имеем следующие формулы для расчета характеристик компаратора:

$$L(D) = 12(n-k-1), \quad r(B) = 2 \lceil \log_2 (n-k) \rceil. \quad (3.11)$$

Таблица 3.4

k	СПТ1			СПТ2			СПТ3			СПТ4		
	L	t	r_1	L	t	r_1	L	t	r_1	L	t	r_1
3	41	8	7	32	8	7	29	6	8	29	6	8
5	92	21	15	140	32	9	75	10	14	73	12	12
6	121	21	15	288	64	9	100	12	16	96	15	12
7	140	12	17	608	128	9	125	14	18	118	20	14
9	201	29	22	2866	512	9	205	18	22	190	30	14
10	230	29	24	6197	1024	9	245	20	24	225	35	14
11	249	29	24	13365	2048	9	236	22	26	260	42	14
12	288	29	28	28731	4096	9	337	24	28	304	49	14
14	336	29	25	131134	16384	9	442	28	32	392	62	14

В соответствии с рассмотренными структурами генератора можно выделить четыре типа (n, k) -СПТ, представленные в табл. 3.4. Тестеры СПТ1 включают в себя генераторы, реализованные с помощью модулей MS и HS , СПТ2 — генераторы на основе блока A_1^1 , СПТ3 — генераторы на основе блока A_1^2 и СПТ4 — генераторы на основе блока A_1^3 . Компараторы у всех тестеров одинаковы. Тестеры СПТ2 имеют наибольшее быстродействие, но и существенную сложность. Тестеры СПТ4 также имеют достаточно хорошее быстродействие и значительно меньшую сложность. Тестеры СПТ3 имеют наилучшую характеристику t , а также наименьшую сложность при $k \leq 10$, а СПТ1 — наименьшую сложность для значений $k \geq 11$.

Общим недостатком СПТ1 — СПТ4 является их невысокое быстродействие и то, что они не могут быть реализованы с помощью ПЛМ. Быстродействующие тестеры могут быть реализованы на основе принципа преобразования кодов. В [49] предложена следующая схема преобразования кодов $nSk \rightarrow 2^k C1 \rightarrow 2C1$, которая дает тестеры с большой сложностью. Тестер включает в себя одноуровневый преобразователь $nSk \rightarrow 2^k C1$, состоящий из 2^k элементов И. Каждый элемент И реализует конъюнкцию, состоящую из переменных, принимающих значение 1 в соответствующем одном из 2^k слов кода nSk . Последовательно с этим преобразователем включается $2/2^k$ -СПТ. Минимальное число уровней последнего равно трем. Поэтому любой (n, k) -СПТ может быть реализован с помощью четырехуровневой схемы со сложностью

$$L = k + \sum_{i=1}^k C_k^i (i + p) + L(2/2^k\text{-СПТ}), \quad (3.12)$$

где p — число единиц в $(n-k)$ -разрядном слове, полученном из двоичного представления числа i путем замены в нем 1 на 0 и наоборот. Для проверки тестера требуются все слова кода nSk .

В [67] предложен более эффективный метод синтеза быстродействующих тестеров. Обозначим через W множество всех слов nSk -кода. Каждому слову $\rho_i \in W$ соответствует конъюнкция

$$M_i = x_{i_1} x_{i_2} \dots x_{i_t} x'_{i_1} x'_{i_2} \dots x'_{i_p} (x_{i_j} \in \omega_1, j \in \{1, 2, \dots, t\}),$$

ω_1 — множество информационных разрядов, $x'_{i_l} \in \omega_2, l \in \{1, 2, \dots, p\}$, ω_2 — множество контрольных разрядов), включающая в себя те и только те переменные x_i и x'_i , которые соответствуют разрядам, имеющим значение 1 в кодовом слове ρ_i . Между словами ρ_i и конъюнкциями M_i существует взаимно однозначное соответствие, что позволяет множество всех конъюнкций M_i также обозначить через W . Например, для заданного табл. 1.14 5S3-кода имеем

$$W = \{x_4' x_5', x_3 x_4', x_2 x_4', x_2 x_3 x_5', x_1 x_4', x_1 x_3 x_5', x_1 x_2 x_5', x_1 x_2 x_3\}.$$

Для каждого слова $\rho \in W$ получим множества двоичных векторов $W_0(\rho)$ и $W_1(\rho)$. В множество $W_0(\rho)$ входят все векторы, которые образуются фиксацией одного единичного разряда слова ρ в нуль, а в множество $W_1(\rho)$ — векторы, образуемые фиксацией одного нулевого разряда в единицу. Обозначим через W_0 и W_1 множество векторов (а также соответствующих им конъюнкций), получаемых как объединение соответственно всех множеств $W_0(\rho)$ и $W_1(\rho)$. Для 5C3-кода имеем:

$$W_0 = \{x_1, x_2, x_3, x_4', x_5', x_1 x_2, x_1 x_3, x_1 x_5', x_2 x_3, x_2 x_5', x_3 x_5'\},$$

$$W_1 = \{x_1 x_2 x_4', x_1 x_3 x_4', x_1 x_4' x_5', x_2 x_3 x_4', x_2 x_4' x_5', x_3 x_4' x_5', x_1 x_2 x_3 x_4', x_1 x_2 x_3 x_5', x_1 x_2 x_4' x_5', x_1 x_3 x_4' x_5', x_2 x_3 x_4' x_5'\}.$$

Для любого кода nSk строится преобразователь $nSk \rightarrow (k+1)C1$, который описывается функциями y_1, y_2, \dots, y_{k+1} , удовлетворяющими следующим условиям (по аналогии с условиями $H1^* - H4^*$):

$H1'$. Функции y_1, \dots, y_{k+1} должны представлять собой дизъюнкцию конъюнкций $M_i \in W$.

$H2'$. Каждая конъюнкция $M_i \in W$ должна входить в одну и только в одну из функций y_1, \dots, y_{k+1} .

$H3'$. Каждая конъюнкция $M_i \in W_1$ должна покрываться хотя бы одной конъюнкцией, включенной в функцию y_i , и хотя бы одной конъюнкцией, включенной в функцию y_j ($i \neq j, j \in \{1, 2, \dots, k+1\}$).

$H4'$. Каждая конъюнкция $M_i \in W_0$ должна покрывать хотя бы одну конъюнкцию, включенную в функцию y_i , и хотя бы одну конъюнкцию, включенную в функцию y_j ($i \neq j$).

Например, для кода $5S3$ данным условиям удовлетворяет система (описывает преобразователь $5S3 \rightarrow 4C1$):

$$\begin{aligned} y_1 &= x_1x_4' \vee x_2x_3x_5', & y_3 &= x_3x_4' \vee x_1x_2x_5', \\ y_2 &= x_2x_4' \vee x_1x_3x_5', & y_4 &= x_1x_2x_3 \vee x_4'x_5'. \end{aligned} \quad (3.13)$$

Наиболее просто условия $H1' - H4'$ выполняются для полных кодов. Рассмотрим множество слов W . Обозначим через Q_p ($p \in \{0, 1, 2, \dots, k\}$) множество слов nSk -кода, у которых равны единице p информационных разрядов, через T_d ($d \in \{0, 1, \dots, r\}$, $r = n - k$) — множество слов nSk -кода, у которых равны единице d контрольных разрядов, а через ρ_{pd} — вектор, у которого равны единице p информационных и d контрольных разрядов (M_{pd} — соответствующая ему конъюнкция). Если $\rho_{pd} \in W$, то $\rho_{pd} \in Q_p$ и $\rho_{pd} \in T_d$.

Таблица 3 5

Q_p	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8'	x_9'	x_{10}'
Q_0	0	0	0	0	0	0	0	1	1	1
Q_1	1	0	0	0	0	0	0	1	1	0
Q_2	1	1	0	0	0	0	0	1	0	1
Q_3	1	1	1	0	0	0	0	1	0	0
Q_4	1	1	1	1	0	0	0	0	1	1
Q_5	1	1	1	1	1	0	0	0	1	0
Q_6	1	1	1	1	1	1	0	0	0	1
Q_7	1	1	1	1	1	1	1	0	0	0

Преобразуем множество Q_2 в множество

$$Q_2^* = \{x_1x_2, x_1x_3, x_1x_4, x_1x_5, \dots, x_6x_7\},$$

которое можно рассматривать как множество конъюнкций, соответствующее коду 7C2. Для получения функций f_1, f_2, \dots, f_k воспользуемся методом [39], описанным в § 2.5. Он позволяет для кода 7C2 получить следующие три функции, удовлетворяющие условиям Н1* — Н4*:

$$\begin{aligned} f_1' &= x_1x_5 \vee x_1x_6 \vee x_1x_7 \vee x_2x_5 \vee x_2x_6 \vee x_2x_7 \vee x_3x_5 \vee \\ &\quad \vee x_3x_6 \vee x_3x_7 \vee x_4x_5 \vee x_4x_6 \vee x_4x_7, \\ f_2' &= x_1x_3 \vee x_1x_4 \vee x_2x_3 \vee x_2x_4 \vee x_5x_7 \vee x_6x_7, \\ f_3' &= x_1x_2 \vee x_3x_4 \vee x_5x_6. \end{aligned}$$

Так как в рассматриваемом случае $k = 7$, то необходимо получить семь функций f_i , удовлетворяющих указанным условиям. Очевидно, что это можно сделать за счет произвольного разложения функций $f_1' \div f_3'$. Например,

$$f_1 = f_1', \quad f_2 = x_1x_3 \vee x_1x_4 \vee x_2x_3 \vee x_2x_4, \quad f_3 = x_5x_7, \\ f_4 = x_6x_7, \quad f_5 = x_1x_2, \quad f_6 = x_3x_4, \quad f_7 = x_5x_6.$$

Полученные функции преобразуются в функции $f_1^2 \div f_7^2$ путем приписывания справа к каждой конъюнкции произведения контрольных переменных $x_8'x_{10}'$.

Функции y_1, y_2, \dots, y_{k+1} , описывающие для полного кода преобразователь $nSk \rightarrow (k+1)C1$, определяются с помощью следующего алгоритма.

Алгоритм 3.4:

1. Для заданного nSk -кода составляется ТПС.
2. По ТПС находятся множества $Q_0, Q_1, Q_2, \dots, Q_p, \dots, Q_k$.
3. Для каждого множества Q_p вычисляются функции $f_1^p, f_2^p, \dots, f_k^p$.
4. Вычисляются функции y_1, y_2, \dots, y_{k+1} по следующим формулам:

$$\begin{aligned}
y_1 &= f_1^1 \vee f_1^2 \vee f_1^3 \vee \dots \vee f_1^{k-1}, \\
y_2 &= f_2^1 \vee f_2^2 \vee f_2^3 \vee \dots \vee f_2^{k-1}, \\
&\vdots \\
y_k &= f_k^1 \vee f_k^2 \vee f_k^3 \vee \dots \vee f_k^{k-1}, \\
y_{k+1} &= x_1 x_2 \dots x_k \vee x'_{k+1} x'_{k+2} \dots x'_n.
\end{aligned} \tag{3.15}$$

Система (3.13) для кода 5С3 соответствует данным формулам. Для кода 10S7 имеем:

$$\begin{aligned} y_1 &= f_1^1 \vee f_1^2 \vee \dots \vee f_1^6, \\ y_2 &= f_2^1 \vee f_2^2 \vee \dots \vee f_2^6, \\ &\vdots \end{aligned}$$

$$y_7 = f_7^1 \vee f_7^2 \vee \dots \vee f_7^6,$$

$$y_8 = x_1 x_2 x_3 x_4 x_5 x_6 x_7 \vee x_8' x_9' x_{10}'.$$

Неполные nSk -коды содержат в себе (в отличие от полных кодов) не все 2^{n-k} контрольных векторов, а только часть из них. Это приводит к тому, что для неполного кода имеют место случаи невыполнения условия Н4'. Для исключения этих случаев осуществляется частичное преобразование контрольных векторов кода за счет следующего изменения ТПС. Рассматривается каждый разряд, равный 1, контрольной части каждого кодового вектора в ТПС. Если хотя бы в одном векторе, расположенном в ТПС ниже данного вектора, в рассматриваемом разряде стоит 0, то единичное значение разряда сохраняется, а в противном случае заменяется на 0. В последней строке ТПС во всех контрольных разрядах проставляется 0.

Таблица 3 6

Q_p	x_1	x_2	x_3	x_4	x_5	x_6'	x_7'	x_8'	x_7'
Q_0	0	0	0	0	0	1	1	1	1
Q_1	1	0	0	0	0	1	1	0	1
Q_2	1	1	0	0	0	1	0	1	0
Q_3	1	1	1	0	0	1	0	0	0
Q_4	1	1	1	1	0	0	1	1	0
Q_5	1	1	1	1	1	0	1	0	0

Рассмотрим неполный код 8S5. Для него ТПС представлена в табл. 3.6. Преобразованию подлежит только разряд x_7' , новый вариант которого показан в крайнем правом столбце таблицы. По преобразованной ТПС составляем множества Q_p :

$$Q_0 = \{x_6 x_7 x_8\},$$

$$Q_1 = \{x_1 x_6' x_7', x_2 x_6' x_7', \dots, x_5 x_6' x_7'\},$$

$$Q_2 = \{x_1 x_2 x_6' x_8', x_1 x_3 x_6' x_8', \dots, x_4 x_5 x_6' x_8'\},$$

$$Q_3 = \{x_1 x_2 x_3 x_6', x_1 x_2 x_4 x_6', \dots, x_3 x_4 x_5 x_6'\},$$

$$Q_4 = \{x_1 x_2 x_3 x_4 x_8', x_1 x_2 x_3 x_5 x_8', \dots, x_2 x_3 x_4 x_5 x_8'\},$$

$$Q_5 = \{x_1 x_2 x_3 x_4 x_5\}.$$

Функции $y_1 \div y_6$, описывающие преобразователь $8S5 \rightarrow 6C1$, вычисляются с помощью выполнения п. 3 и 4 алгоритма 3.4.

Рассмотренные преобразователи $nSk \rightarrow (k+1)C1$ реализуются в виде двухуровневой схемы вида И — ИЛИ с $k+1$ выходом, сложность которой определяется по формуле

$$L = \sum_{p \in \{1, 2, \dots, k\}} C_p^k \omega(Q_p) + n + 2^k - k, \quad (3.16)$$

где $\omega(Q_p)$ — число единиц, расположенных в строке Q_p ТПС (или преобразованной ТПС для неполных кодов).

В структуре (n, k) -СПТ последовательно с преобразователем $nSk \rightarrow qC1$ включается любой из $1/q$ -СПТ ($q = k+1$), описанных в § 2.4. Целесообразно использовать $1/q$ -СПТ2 и $1/q$ -СПТ3 (см. табл. 2.10). Обозначим как СПТ5 тестер, состоящий из преобразователя $nSk \rightarrow qC1$ и $1/q$ -СПТ1. Например, $(5, 3)$ -СПТ5 состоит из преобразователя $5S3 \rightarrow 4C1$, описываемого системой (3.13), и $1/4$ -СПТ1, показанного на рис. 2.4. Данный тестер содержит пять уровней. Однако число уровней СПТ5 всегда может быть уменьшено на единицу. Это связано с тем, что в $1/q$ -СПТ1 (независимо от значения числа q) первые два уровня схемы состоят только из элементов ИЛИ. Поэтому возможно объединение последнего уровня схемы преобразователя $nSk \rightarrow qC1$ (состоящего также из элементов ИЛИ) с двумя первыми уровнями $1/q$ -СПТ1. Сформулируем алгоритм синтеза таких (n, k) -СПТ (тип СПТ6).

Алгоритм 3.5:

1. Для заданного кода nSk с помощью алгоритма 3.4 вычисляются функции y_1, y_2, \dots, y_q ($q = k+1$), описывающие преобразователь $nSk \rightarrow qC1$.

2. С помощью метода [37] вычисляются функции z_1 и z_2 , описывающие $1/q$ -СПТ. При этом входными переменными $1/q$ -СПТ являются переменные y_1, y_2, \dots, y_q , так как выходы преобразователя $nSk \rightarrow qC1$ непосредственно соединяется с $1/q$ -СПТ.

3. Конъюнкции M_i , входящие в качестве элементов в функции y_1, y_2, \dots, y_q , обозначаются буквами g_j ($j \in \{1, 2, \dots, 2^k\}$). Индексы j присваиваются произвольным образом.

4. В системе y_1, y_2, \dots, y_q осуществляется подстановка функций g_j вместо соответствующих им конъюнкций M_j .

5. В функциях z_1 и z_2 , полученных в п. 2, переменные заменяются функциями, полученными в п. 4.

6. Тестер строится в виде последовательного соединения двух блоков. Первый блок реализует систему конъюнкций M_i , определенных в п. 3 (один уровень элементов), а второй — функции z_1 и z_2 , полученные в п. 5 (три уровня элементов).

Пример 3.4. Получим функции, описывающие $(5, 3)$ -СПТ6. В данном случае $q = k+1 = 3+1 = 4$. Преобразователь $5S3 \rightarrow 4C1$ описывается функциями $y_1 \div y_4$ из системы (3.13). Структура $1/4$ -СПТ (см. рис. 2.4) с учетом поступления на ее входы сигналов с выхода преобразователя $5S3 \rightarrow 4C1$ описывается функциями:

$$z_1 = [(y_1 \vee y_4) \vee y_2] [(y_2 \vee y_3) \vee y_4], \quad (3.17)$$

$$z_2 = [(y_2 \vee y_3) \vee y_1] [(y_1 \vee y_4) \vee y_3].$$

В этой системе одинаковые выражения, заключенные в круглые скобки, реализуются одним и тем же логическим элементом. В систему (3.13) входит восемь конъюнкций M_i . Обозначим: $g_1 = x_1x_4'$, $g_2 = x_2x_4'$, $g_3 = x_3x_4'$, $g_4 = x_1x_2x_5'$, $g_5 = x_1x_3x_5'$, $g_6 = x_2x_3x_5'$, $g_7 = x_1x_2x_3$, $g_8 = x_4'x_5'$. Осуществляем подстановку введенных функций в систему (3.13):

$$\begin{aligned} y_1 &= g_1 \vee g_6, & y_3 &= g_3 \vee g_4, \\ y_2 &= g_2 \vee g_5, & y_4 &= g_7 \vee g_8. \end{aligned} \quad (3.17a)$$

В функциях z_1 и z_2 системы (3.17) заменяем переменные y_i соответствующими функциями из системы (3.17a)

$$\begin{aligned} z_1 &= [(g_1 \vee g_6 \vee g_7 \vee g_8) \vee g_2 \vee g_5] \times \\ &\quad \times [(g_2 \vee g_3 \vee g_4 \vee g_5) \vee g_7 \vee g_8], \\ z_2 &= [(g_2 \vee g_3 \vee g_4 \vee g_5) \vee g_1 \vee g_6] \times \\ &\quad \times [(g_1 \vee g_6 \vee g_7 \vee g_8) \vee g_3 \vee g_4]. \end{aligned}$$

Полученные функции соответствуют четырехуровневой структуре тестера. Первый уровень состоит из восьми элементов И (реализует конъюнкции $g_1 \div g_8$), второй — из двух элементов ИЛИ (реализует дизъюнкции $g_1 \vee g_6 \vee g_7 \vee g_8$ и $g_2 \vee g_3 \vee g_4 \vee g_5$), третий — из четырех элементов ИЛИ (реализует функции, заключенные в квадратные скобки) и четвертый — из двух элементов И (реализует функции z_1 и z_2).

Число уровней СПТ5 и СПТ6 с ростом значений n и k увеличивается, так как растет число уровней $1/q$ -СПТ2. Замена последних на трехуровневые $1/q$ -СПТ3 (см. табл. 2.10) позволяет получить (n, k) -СПТ со стандартными значениями числа уровней. Последовательное соединение двухуровневого преобразователя $nSk \rightarrow qC1$ и трехуровневого $1/q$ -СПТ3 дает тестер с пятью уровнями (тип СПТ7). Тестеры СПТ7 преобразуются в четырехуровневые тестеры СПТ8 так же, как тестеры СПТ5 в тестеры СПТ6.

Таблица 3.7

k	СПТ5		СПТ6		СПТ7		СПТ8	
	L	r_1	L	r_1	L	r_1	L	r_1
3	44	5	44	4	48	5	48	4
4	97	5	103	4	100	5	106	4
5	184	5	209	4	190	5	212	4
6	390	6	425	5	410	5	547	4
7	804	6	911	5	828	5	1128	4

В табл. 3.7 для ряда значений k представлены характеристики рассмотренных тестеров. Их сравнение с характеристиками СПТ1 и СПТ4 (см. табл. 3.4) показывает, что увеличение быстродействия связано с существенным увеличением сложности. Кроме того, данные тестеры требуют для своей проверки все слова nSk -кода. Тестеры СПТ8 имеют стандартную структуру вида И—ИЛИ—И—ИЛИ и могут быть реализованы на двух ПЛМ. В табл. 3.8 приведены характеристики матриц.

Таблица 3.8

k				
3	4	5	6	7
$M1 [5, 8, 4]$ $M2 [4, 4, 2]$	$M1 [7, 16, 4]$ $M2 [4, 5, 2]$	$M1 [8, 32, 4]$ $M2 [4, 5, 2]$	$M1 [9, 64, 7]$ $M2 [7, 7, 2]$	$M1 [10, 128, 7]$ $M2 [7, 8, 2]$

В [72] предложены оригинальные способы построения тестеров для двух классов nSk -кода. Идея реализации (n, k) -СПТ для полных кодов состоит в следующем. Код nSk рассматривается как совокупность кодов $(k+1)Cm$, где $m \in \{1, 2, \dots, k\}$. При этом в множество переменных $(k+1)Cm$ -кода входят все информационные переменные и контрольная переменная x_n' . Рассмотрим, например, 10S7-код, заданный табл. 3.5. Будем

Таблица 3.9

x_1	x_2	x_3	x_4	x_5'	x_6'	x_7'
0	0	0	0	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	1	1	1	0	0
1	0	0	0	0	1	1
1	0	0	1	0	1	0
1	0	1	1	0	0	1
1	1	1	1	0	0	0

считать, что множество Q_p задает совокупность кодовых слов, определяемых значениями переменных $x_1, x_2, \dots, x_7, x_{10}'$. Тогда объединение множества Q_0 и Q_1 задает код 8C1, объединение Q_2 и Q_3 — код 8C3, объединение Q_4 и Q_5 — код 8C5 и объединение Q_6 и Q_7 — код 8C7. Из табл. 3.5 видно, что каждому коду

$(k+1)Cm$ однозначно соответствует определенная комбинация контрольных разрядов (с учетом исключения разряда x_{10}'). Так, коду 8C1 соответствует комбинация $x_8'x_9' = 11$, коду 8C3 — комбинация $x_8'x_9' = 10$ и т. д. На основе этого соответствия возможно построение (n, k) -СПТ за счет реализации совокупности $m/(k+1)$ -СПТ. Так, на рис. 3.6 представлена схема $(5, 3)$ -СПТ. Код 5S3 рассматривается как совокупность кодов 4C1 и 4C3, задаваемых на разрядах x_1, x_2, x_3, x_5' (см. табл. 1.14). Код 4C1 образуется объединением множеств Q_0 и Q_1 , код 4C3 — множеств Q_2 и Q_3 . Коду 4C1 соответствует значение контрольной переменной $x_4' = 1$, а коду 4C3 — переменной $x_4' = 0$. На основе

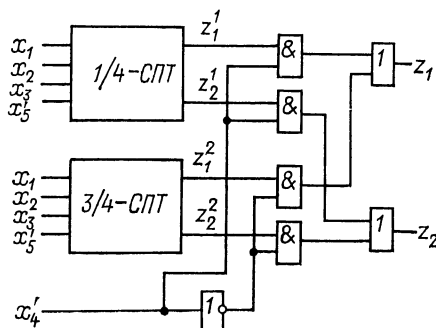


Рис. 3.6

данного подхода реализуются быстродействующие тестеры с минимальным числом уровней $r = 5$, которые, однако, по своим характеристикам уступают (n, k) -СПТ6

Для кодов nSk , у которых $k = 2^{n-k-1}$, известен подход [72], позволяющий реализовать тестеры с помощью модулей MC , MS и HS . Рассмотрим его на примере кода 7S4. В табл. 3.9 для этого кода построена ТПС специального вида, предусматривающая пре-

образование кода. Верхняя часть таблицы содержит векторы со значениями старшего разряда информационного слова $x_1 = 0$, а нижняя часть — со значениями $x_1 = 1$. Контрольные слова в верхней части таблицы образуются по правилу построения кода nSk , при этом во всех словах старший разряд $x_5' = 1$. В нижней части таблицы контрольные слова формируются следующим образом. Старшему разряду x_5' во всех словах присваивается значение 0, а слова, образуемые остальными контрольными разрядами (x_6' и x_7'), формируются по правилу построения кода nSk , но относительно информационных слов, которые получаются из исходных за счет исключения старшего разряда x_1 . В этом случае, как видно из таблицы, старшие разряды информационных и контрольных слов (x_1 и x_5') образуют парафазный код, а остальные разряды — код $(n-2)S(k-1)$ (в данном случае код 5S3). Поэтому (n, k) -СПТ реализуется следующим образом. Устанавливается СПТ для кода $(n-2) \times S(k-1)$. Информационными и контрольными разрядами данного кода являются все разряды исходного кода за исключением указанных старших разрядов. Парафазный выход СПТ подается на один из парафазных входов модуля MC (см. рис. 3.1). На второй парафазный вход MC подаются переменные, соответствующие старшим разрядам информационного и контроль-

ного слов (x_1 и x_5'). Выход МС является выходом тестера.

Данные тестеры выигрывают по сложности у тестеров вида СПТ5—СПТ8, но проигрывают им в быстродействии. Так, (7, 4)-СПТ состоит из (5, 3)-СПТ и МС. Так как в качестве (5, 3)-СПТ может быть использован любой из восьми типов рассмотренных выше тестеров, то (7, 4)-СПТ имеет восемь модификаций. При использовании (5, 3)-СПТ3 имеем характеристики: $L = 41$, $r = 10$, а при использовании (5, 3)-СПТ6 $L = 56$, $r = 6$.

3.3. Тестеры для кодов с проверкой на нечетность (четность)

При построении контрольных схем для кодов с проверкой на нечетность $nH1$ (четность — $nH2$) используются линейные схемы, состоящие из модулей $M2$, реализующих функцию «сложение по модулю 2». Как правило, используются двух-входовые модули $M2$. При этом возможны два подхода. Первый подход связан с рассмотрением одиночных константных неисправностей входов и выхода модуля (в этом случае для проверки модуля требуется три входных набора). Для заданного подхода вопросы построения СПТ рассмотрены в [1, 13, 53, 70].

Тестер для кода $nH1$ строится следующим образом. Множество переменных кода разбивается произвольным образом на два непересекающихся подмножества. Переменные, входящие в одно подмножество, объединяются схемой свертки по модулю 2. На рис. 3.7 приведен тестер для кода $6H1$.

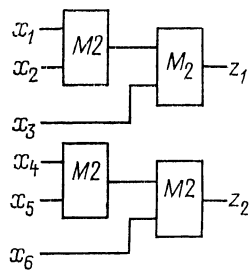


Рис. 3.7

Тестеры для кодов $nH2$ строятся аналогичным образом с установкой дополнительного инвертора на одном из выходов z_1 или z_2 . Если множество W слов заданного кода включает в себя все слова с нечетным (четным) числом единиц ($\mu = 2^{n-1}$), то проверка схемы тестера обеспечивается при произвольном распределении переменных по ее входам. Если $\mu < 2^{n-1}$, то для обеспечения самопроверки должны быть выполнены определенные условия [1]. В [13] показано, что эти условия выполняются, если $\mu \geq 2^{n-2} + 1$. Для противоположного случая [1, 13] предложены алгоритмы, позволяющие обеспечивать самопроверяемость тестеров.

Второй подход к синтезу тестеров для кодов $nH1$ и $nH2$ предусматривает полную проверку внутренней структуры модулей $M2$. Это требует подачи на входы каждого модуля четырех входных наборов. Из результатов работы [43] вытекает следующий метод построения СПТ. Множество переменных кода B разбивается на два подмножества: $B_1 = \{x_1, \dots, x_k\}$ и $B_2 =$

$= \{x_{k+1}, \dots, x_n\}$, причем $k = \lceil n/2 \rceil$. Составляется одноканальная схема из модулей $M2$ на k входов, которые соединяются с переменными из множества B_1 , и аналогичная схема на $n-k$ входов, которые соединяются с переменными из множества B_2 . Выходы одноканальных схем являются выходами тестера. Тестер проверяется четырьмя входными наборами, которые строятся следующим образом. С помощью алгоритма 3.1 находятся по четыре проверяющих набора обеих одноканальных схем. При этом два набора из четырех имеют нечетное число единиц, а два — четное. Формируются четыре набора проверяющего теста СПТ. Каждый из наборов теста первой одноканальной схемы объединяется с одним из наборов теста второй одноканальной схемы так, чтобы число единиц в образуемом наборе было нечетным. Схема СПТ, показанная на рис. 3.7, отвечает описанному методу. В табл. 3.10 приведен проверяющий тест для этой схемы.

Таблица 3.10

x_1	x_2	x_3	x_4	x_5	x_6
0	0	0	0	1	0
0	1	0	0	0	0
1	0	1	1	1	1
1	1	1	1	0	1

Недостатком СПТ, реализованных на модулях $M2$, является их низкое быстродействие. Рассмотрим метод синтеза быстродействующих тестеров, который, однако, связан с существенным увеличением их сложности. В табл. 3.11 представлена ТПС для кода 6H1. Информационными являются переменные $x_1 \div x_5$, x_6 —

Таблица 3.11

W	x_1	x_2	x_3	x_4	x_5	x_6	nCm
W_0	0	0	0	0	0	1	6C1
W_1	0	0	0	0	1	0	
W_2	0	0	0	1	1	1	6C2
W_3	0	0	1	1	1	0	
W_4	0	1	1	1	1	1	6C5
W_5	1	1	1	1	1	0	

контрольная переменная. В строке таблицы указывается один представитель множества кодовых слов W_i , содержащих i единичных информационных разрядов. Из таблицы видно, что код $6H1$ может быть представлен как совокупность трех равновесных кодов: $6C1$ (образуется объединением множеств W_0 и W_1), $6C2$ (W_2 и W_3) и $6C5$ (W_4 и W_5). Данное обстоятельство позволяет реализовать СПТ для кода $6H1$ в виде структуры, приведенной на рис. 3.8. Нетрудно показать, что она обладает свойством самопроверки относительно одиночных неисправностей.

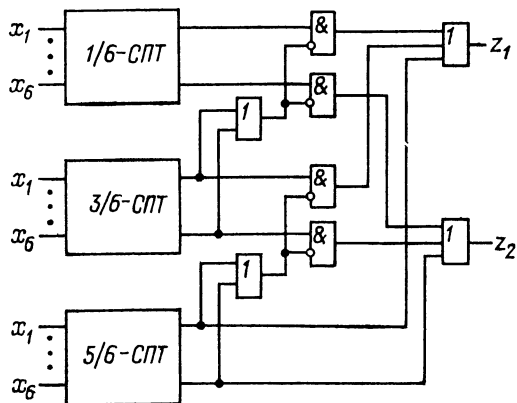


Рис. 3.8

Структура типа рис. 3.8 существует для любого кода $nH1$, если $n = \varphi(2)$. Характеристики СПТ зависят от способа реализации m/n -тестеров и определяются по следующим формулам:

$$L_n^1 = \sum_{\substack{i=1, \\ i=\varphi(1)}}^{n-1} L(i/n\text{-СПТ}) + 4n - 6, \quad (3.18)$$

$$t_n^1 = \sum_{\substack{i=1, \\ i=\varphi(1)}}^{n-1} t(i/n\text{-СПТ}), \quad (3.19)$$

$$r_n^1 = \max [r(1/n\text{-СПТ}), r(3/n\text{-СПТ}), \dots, r((n-1)/n\text{-СПТ})] + 3. \quad (3.20)$$

Так как любой m/n -СПТ может быть реализован в виде трехуровневой схемы, то $r_n^1_{\max} = 6$.

Для кода $nH2$ при $n = \varphi(2)$ рассмотренная структура СПТ не является самопроверяемой, так как множества W_0 и W_n в этом случае образуют коды $nC0$ и nCn , для которых СПТ не существуют. Поэтому тестер для кода $nH2$ целесообразно получать из тестера для кода $nH1$ при инвертировании контрольной переменной. Для кода $nH1$ ($n = \varphi(1)$) самопроверяе-

мая структура может быть получена, если исключить из кода вектор $111\dots 11$, а для кода $nH2$ ($n = \varphi(1)$) — если исключить вектор $000\dots 00$.

3.4. Самопроверяемые дешифраторы кодов

Дешифратор (ДШ) является одним из основных элементов дискретных систем и представляет собой комбинационное устройство, имеющее n входов и k выходов. На входы x_1, x_2, \dots, x_n подаются слова рассматриваемого кода. На выходах y_1, y_2, \dots, y_k реализуются кодовые слова. ДШ, не обладающие свойством обнаружения неисправностей, строятся непосредственно по соответствующим им логическим формулам и имеют следующие свойства:

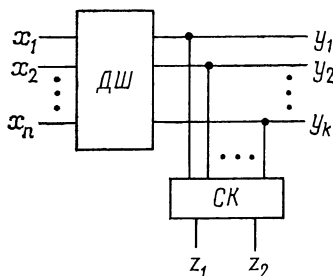


Рис. 3.9

Д1) дешифрации — при поступлении на вход ДШ кодового слова на соответствующем ему выходе появляется сигнал 1;

Д2) контроля входного вектора на выходе ДШ — при поступлении на вход ДШ некодового вектора на всех выходах $y_1—y_k$ формируется сигнал 0.

Для контроля исправности ДШ снабжается контрольной схемой,

которая подключается к выходам $y_1—y_k$ (рис. 3.9). При дешифрации избыточного кода, когда число кодовых слов равно 2^n , используется три способа контроля. При первом способе в качестве схемы контроля (СК) используется $1/k$ -СПТ ввиду того, что на выходах $y_1—y_k$ ДШ формируется вектор кода $kC1$. Второй способ заключается в дублировании ДШ с применением тестеров для кодов с повторением. Третий способ описан в [53]. Он предусматривает построение ДШ в виде пирамидальной схемы с n уровнями. На первом уровне реализуются инверсии входных переменных, на втором — конъюнкции второго ранга, а на третьем — конъюнкции третьего ранга и т. д. На выходах ДШ реализуются конъюнкции ранга n . Схема контроля образуется путем объединения через один элемент ИЛИ всех выходов ДШ, на которых реализуются конъюнкции с четным числом переменных с инверсией, а через второй элемент ИЛИ — конъюнкции с нечетным числом таких переменных.

Например, ДШ для избыточного кода при $n = 3$ описывается следующей системой функций: $4 = \bar{1}$, $5 = \bar{2}$, $6 = \bar{3}$, $7 = 1 \times 2$, $8 = 1 \times 5$, $9 = 2 \times 4$, $10 = 4 \times 5$, $y_1 = 6 \times 10$, $y_2 = 3 \times 10$, $y_3 = 6 \times 9$, $y_4 = 3 \times 9$, $y_5 = 6 \times 8$, $y_6 = 3 \times 8$, $y_7 = 6 \times 7$, $y_8 = 3 \times 7$, $z_1 = y_1 \vee y_4 \vee y_6 \vee y_7$, $z_2 = y_2 \vee y_3 \vee y_5 \vee y_6$.

В описанных дешифраторах по выходу СК обнаруживаются все одиночные неисправности, кроме неисправностей входов. Полный контроль ДШ возможен только в дешифраторах для кодов с обнаружением ошибок nRp [44]. В этом случае свойство Д2 находится в противоречии с требованием обнаружения неисправностей. Это связано с тем, что код nRp имеет избыточность, наблюдаемую и в схемах дешифрации. При этом элементы дешифратора содержат избыточные входы, неисправности которых не приводят к нарушению правильной работы ДШ при поступлении кодовых слов (сохраняется свойство дешифрации), но искажают работу ДШ при подаче некодовых слов (теряется свойство контроля входного вектора на выходе ДШ), в результате чего при поступлении некодового слова на одном из выхо-

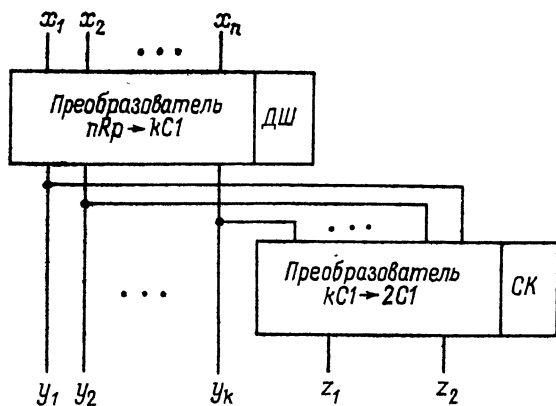


Рис. 3.10

дов $y_1 - y_k$ формируется сигнал 1. Для устранения указанного противоречия дешифратор для кодов с обнаружением ошибок строится в виде структуры, показанной на рис. 3.10. Такой ДШ обладает свойством дешифрации, а также еще четырьмя дополнительными свойствами:

Д3) контроля входного вектора на выходе СК — выходы СК z_1 и z_2 принимают значения (1, 0) или (0, 1), если на входе ДШ присутствует вектор nRp -кода, и значения (0, 0) или (1, 1) в противном случае;

Д4) самопроверки СК — для любой одиночной неисправности СК существует вектор nRp -кода, на котором выходы z_1 и z_2 принимают значения (0, 0) или (1, 1);

Д5) обнаружения неисправностей ДШ — для любой одиночной неисправности ДШ существует вектор nRp -кода, на котором выходы z_1 и z_2 принимают значения (0, 0) или (1, 1);

Д6) защищенности ДШ от неисправностей — любая одиночная неисправность ДШ на любом векторе nRp -кода либо не

приводит к искажению значений выходов дешифратора, либо в противном случае вызывает также появление на выходах z_1 и z_2 СК значений (0, 0) или (1, 1).

В структуре рис. 3.10 каждый из преобразователей обладает свойствами контроля входного вектора и самопроверки. Дешифратор заменен на преобразователь $nRp \rightarrow kC1$, который обладает также свойством защищенности от неисправностей (при этом допускается при поступлении на вход преобразователя некодового слова формирование сигналов 1 сразу на нескольких выходах). Такая замена допустима, так как поступление на вход ДШ некодового слова фиксируется на выходе СК. С другой стороны, эта замена позволяет решить задачу обнаружения неисправностей ДШ, так как преобразователь $nRp \rightarrow kC1$ не содержит избыточных элементов.

Рассмотрим положения метода, который позволяет строить для любого кода преобразователь $nRp \rightarrow kC1$, обладающий свойствами контроля входного вектора, самопроверки и защищенности от неисправностей.

1. Преобразователь $nRp \rightarrow kC1$ описывается функциями y_1, y_2, \dots, y_k , каждая из которых определяется на множестве слов кода nRp и слов, не входящих в этот код.

2. Каждая функция y_i ($i \in \{1, 2, \dots, k\}$) задается следующим образом: она равна 1 на i -м слове кода (т. е. на слове, реализуемом на i -м выходе дешифратора), равна 0 на всех остальных кодовых словах и принимает неопределенное значение на всех некодовых словах.

3. Функции y_i определяются путем совместной минимизации системы функций y_1, y_2, \dots, y_k с учетом использования некодовых слов. При этом каждая функция y_i рассматривается как не полностью определенная функция. Находятся минимальные ДНФ функций y_i при выполнении следующего условия: если некоторое некодовое слово использовано для упрощения функции y_i , то оно должно быть использовано и для упрощения функции y_j ($i \neq j, i, j \in \{1, 2, \dots, k\}$). В результате минимизации каждая функция будет представлять собой конъюнкцию ранга $r < n$.

4. Преобразователь $nRp \rightarrow kC1$ реализуется путем установок k элементов И. На выходе каждого из них реализуется одна из функций y_i . Если код nRp является несравнимым, то функции y_i не содержат инверсных переменных. В этом случае никаких ограничений на структуру преобразователя $nRp \rightarrow kC1$ не накладывается. Если код nRp является сравнимым, то функции y_i содержат инверсные переменные. В этом случае на структуру преобразователя накладывается следующее ограничение: для инверсии входных переменных на каждый вход устанавливается либо только один инвертор, либо несколько включенных параллельно инверторов с учетом увеличения их нагрузочной способности.

Пример 3.5. Рассмотрим код $4H1$. Так как он имеет 8 кодовых слов, то строится преобразователь $4H1 \rightarrow 8C1$. В табл. 3.12 представлены кодовые ($a_1 \div a_8$) и некодовые слова ($b_1 \div b_8$). Каждую функцию y_i получаем на основе объединения слова a_i

Таблица 3.12

a	x_1	x_2	x_3	x_4	b	x_1	x_2	x_3	x_4
a_1	0	0	0	1	b_1	0	0	0	0
a_2	0	0	1	0	b_2	0	0	1	1
a_3	0	1	0	0	b_3	0	1	0	1
a_4	0	1	1	1	b_4	0	1	1	0
a_5	1	0	0	0	b_5	1	0	0	1
a_6	1	0	1	1	b_6	1	0	1	0
a_7	1	1	0	1	b_7	1	1	0	0
a_8	1	1	1	0	b_8	1	1	1	1

и одного из слов b_j . В результате получаем следующую систему функций (в скобках указаны объединяемые слова):

$$y_1 = \bar{x}_1 \bar{x}_2 x_4 (a_1, b_2); \quad y_5 = x_1 \bar{x}_2 \bar{x}_3 (a_5, b_5);$$

$$y_2 = \bar{x}_1 \bar{x}_2 x_3 (a_2, b_2); \quad y_6 = x_1 \bar{x}_2 x_4 (a_6, b_5);$$

$$y_3 = \bar{x}_1 x_2 \bar{x}_3 (a_3, b_3); \quad y_7 = x_1 x_2 x_4 (a_7, b_8);$$

$$y_4 = \bar{x}_1 x_2 x_4 (a_4, b_3); \quad y_8 = x_1 x_2 x_3 (a_8, b_8).$$

По данной системе функций с учетом ограничения, указанного в п. 4, реализуется преобразователь $4H1 \rightarrow 8C1$ (содержит 8 трехходовых элементов И и три элемента НЕ), который включается в структуру ДШ с обнаружением отказов (в соответствии с рис. 3.10) совместно с $1/8$ -СПТ.

Дешифратор, реализованный рассмотренным методом, обладает свойством дешифрации, так как каждая функция y_i равна 1 только на одном из кодовых слов (см. п. 2). При поступлении на вход ДШ (совпадает со входом преобразователя $nRp \rightarrow kC1$) кодовых слов на выходах z_1 и z_2 формируются значения (1, 0) или (0, 1), при поступлении некодовых слов, использованных для упрощения функций y_i , — значения (1, 1), а при поступлении остальных некодовых слов — значения (0, 0).

Неисправности преобразователя $kC1 \rightarrow 2C1$ обнаруживаются путем установления одинаковых значений сигналов z_1 и z_2 .

Одиночные неисправности преобразователя $nRp \rightarrow kC1$ интерпретируются как фиксация в 0 или 1 букв и конъюнкций в системе функций $y_1 \div y_k$. Можно выделить следующие классы неисправностей: фиксацию в 0(1) отдельной функции y_i ; фиксацию в 1 отдельной буквы в конъюнкции, соответствующей функции y_i ; фиксацию в 0(1) одной и той же буквы в двух и более функциях y_i .

Для каждого указанного случая неисправностей при подаче одного из слов кода nRp на выходе дешифратора (совпадает с выходом преобразователя $nRp \rightarrow kC1$) формируется вектор с весом $w \neq 1$ (чем обеспечивается свойство защищенности ДШ от неисправностей), а на выходах z_1 и z_2 устанавливаются одинаковые сигналы (чем обеспечивается свойство обнаружения неисправностей ДШ). Тест проверки ДШ и СК включает в себя полное множество слов кода nRp .

Рассмотренные дешифраторы обозначим как ДШ1. Возможна вторая модификация ДШ2, которая отличается от ДШ1 тем, что преобразователь $kC1 \rightarrow 2C1$ (см. рис. 3.10) реализуется в виде последовательного соединения специального преобразователя $kC1 \Rightarrow qC1$ и преобразователя $qC1 \rightarrow 2C1$. При построении преобразователя $kC1 \Rightarrow qC1$ используются свойства кода, для которого строится дешифратор. При этом реализуется преобразователь $nRp \rightarrow qC1$, состоящий из последовательно включенных преобразователей $nRp \rightarrow kC1$ и $kC1 \Rightarrow qC1$. Он должен обладать свойствами контроля входного вектора и самопроверки, которые обеспечиваются при выполнении следующих условий построения преобразователя $kC1 \Rightarrow qC1$:

1. Преобразователь $kC1 \Rightarrow qC1$ описывается функциями f_1, f_2, \dots, f_q .

2. Каждая функция f_j ($j \in \{1, 2, \dots, q\}$) представляется в виде: $f_j = y_{i_1} \vee y_{i_2} \vee \dots \vee y_{i_p}$, где $i_1, i_2, \dots, i_p \in \{1, 2, \dots, k\}$.

3. Каждая функция y_t ($t \in \{1, 2, \dots, k\}$) из системы, описывающей преобразователь $nRp \rightarrow kC1$, должна входить в одну и только в одну из функций f_1, f_2, \dots, f_q .

4. Каждое некодовое слово, использованное для минимизации функций y_1, y_2, \dots, y_k , должно покрываться хотя бы одной функцией вида y_t , включенной в функцию f_i , и хотя бы одной функцией вида y_t , включенной в f_j . ($i \neq j$; $i, j \in \{1, 2, \dots, q\}$).

5. Каждая функция y_t^* , полученная из функции y_t ($t \in \{1, 2, \dots, k\}$) путем фиксации в 1 одной из ее букв x_l ($x_l = x_l$ или \bar{x}_l , $l \in \{1, 2, \dots, n\}$), должна покрывать хотя бы одно кодовое слово, реализуемое функцией y_r ($r \in \{1, 2, \dots, k\}$), включенной в f_i , и хотя бы одно кодовое слово, реализуемое функцией y_s ($s \in \{1, 2, \dots, k\}$), включенной в функцию f_j ($i \neq j$, $i, j \in \{1, 2, \dots, q\}$).

Выполнение данных условий обеспечивается с помощью таблиц покрытий, аналогичных таблицам, описанным в § 2.2.

Так, для кода $4H1$, рассмотренного в примере 3.5, получаем систему функций, описывающую преобразователь $8C1 \Rightarrow 4C1$:

$$\begin{aligned} f_1 &= y_1 \vee y_7, & f_3 &= y_4 \vee y_6, \\ f_2 &= y_2 \vee y_5, & f_4 &= y_3 \vee y_8. \end{aligned}$$

ДШ2 имеет меньшую сложность чем ДШ1. В данном случае в структуру ДШ1 входит $1/8$ -СПТ с $L = 36$ (см. табл. 2.10), а в структуру ДШ2 — преобразователь $8C1 \Rightarrow 4C1$ с $L = 8$ и $1/4$ -СПТ с $L = 16$.

Рассмотрим особенности реализации ДШ для определенных кодов с обнаружением ошибок. Для кода $nH1$ возможно построение нескольких вариантов преобразователя $nRp \rightarrow kC1$ в зависимости от принятого способа упрощения функций $y_1 \div y_k$. Все варианты являются равноценными с точки зрения характеристик схемы. Поэтому можно использовать следующий стандартный способ определения функций y_i . Выписывается 2^{n-1} двоичных векторов длины $n-1$ избыточного кода в порядке возрастания двоичных чисел. К каждому вектору добавляется контрольный разряд. Слова полученного кода обозначаются как a_i с индексами i ($i = 1, 2, 3, \dots$) в порядке возрастания. Из каждого слова a_i с нечетным индексом образуется конъюнкция y_i , реализующая это слово путем исключения переменной x_{n-1} . Из каждого слова a_i с четным индексом образуется конъюнкция y_i путем исключения переменной x_n . Сложность ДШ1 определяется по формуле

$$L(\text{ДШ1}) = (n-1)2^{n-1} + n - 1 + L(1/2^{n-1} - \text{СПТ}).$$

При построении ДШ2 для любого кода $nH1$ может быть получен преобразователь $kC1 \Rightarrow 3C1$. При этом функция f_1 образуется как дизъюнкция всех функций y_i с нечетными индексами, а функции вида f_3 и f_2 составляются из функций y_i с четными индексами. С помощью таблиц покрытий получены следующие функции f_2 и f_3 для трех первых кодов $nH1$. Для $n = 3$ $f_2 = y_2$, $f_3 = y_4$. Для $n = 4$ $f_2 = y_2 \vee y_8$, $f_3 = y_4 \vee y_6$. Для $n = 5$ $f_2 = y_2 \vee y_8 \vee y_{12} \vee y_{14}$, $f_3 = y_4 \vee y_6 \vee y_{10} \vee y_{16}$.

Для $n \geq 6$ функции f_2 и f_3 находятся на основании следующих соображений. Функции f_2 и f_3 для кода с $n+1$ разрядами (обозначим их f_2^{n+1} и f_3^{n+1}) могут быть получены из функций f_2 и f_3 для кода с n разрядами (f_2^n и f_3^n) с помощью приводимых ниже формул. Пусть

$$\begin{aligned} f_2^n &= y_{i_1} \vee y_{i_2} \vee \dots \vee y_{i_r}, \\ f_3^n &= y_{j_1} \vee y_{j_2} \vee \dots \vee y_{j_p}. \end{aligned}$$

Тогда

$$\begin{aligned} f_2^{n+1} &= y_{i_1} \vee y_{i_2} \vee \dots \vee y_{i_r} \vee y_{j_1+2^n} \vee \dots \vee y_{j_p+2^n}, \\ f_3^{n+1} &= y_{j_1} \vee y_{j_2} \vee \dots \vee y_{j_p} \vee y_{i_1+2^n} \vee \dots \vee y_{i_r+2^n}. \end{aligned}$$

Так как в структуре ДШ2 целесообразно использовать не $1/3$ -СПТ, а $1/4$ -СПТ, то функция f_1 заменяется двумя следующими функциями:

$$f_1 = y_1 \vee y_3 \vee y_5 \vee \dots \vee y_{2^{n-2}-1},$$

$$f_2 = y_{2^{n-2}+1} \vee y_{2^{n-2}+3} \vee \dots \vee y_{2^{n-1}-1}.$$

В этом случае сложность ДШ2 подсчитывается по формуле

$$L(\text{ДШ2}) = n(2^{n-1} + 1) + 15.$$

Для кода nCm ДШ1 имеет стандартную структуру, состоящую из преобразователей $nCm \rightarrow kC1$ и $kC1 \rightarrow 2C1$ ($k = C_n^m$), методы построения которых описаны в гл. 2. Сложность ДШ1 определяется по формуле

$$L(\text{ДШ1}) = mC_n^m + L(1/C_n^m\text{-СПТ}).$$

Структура ДШ2 определяется параметрами кода nCm . Для кодов, у которых существуют двухуровневые m/n -СПТ (см. § 2.6), ДШ2 строится следующим образом. Устанавливается преобразователь $nCm \rightarrow kC1$ ($k = C_n^m$), выходы которого являются выходами дешифратора. В качестве СК устанавливаются два элемента ИЛИ, которые соединяются с выходами преобразователя $nCm \rightarrow kC1$ в соответствии с функциями, описывающими двухуровневый m/n -СПТ. Сложность ДШ2 определяется по формуле

$$L(\text{ДШ2}) = (m+1)C_n^m.$$

Для кодов nCm , у которых не существует двухуровневых тестеров, в качестве СК устанавливаются q элементов ИЛИ, выходы которых соединяются со входами $1/q$ -СПТ. Число q определяется с помощью системы (2.40). Функции f_1, f_2, \dots, f_q , реализуемые на выходах элементов ИЛИ, определяются методом [39] вычисления функций, описывающих преобразователь $nCm \rightarrow qC1$. В этом случае

$$L(\text{ДШ2}) = (m+1)C_n^m + L(1/q\text{-СПТ}).$$

Таблица 3.13

Код	4H1	10H1	5C2	8C4	5S3
$L(\text{ДШ1})$	63	7025	64	586	56
$L(\text{ДШ2})$	51	5145	30	350	44

Для кода с суммированием nSr (здесь r — число информационных разрядов) функции y_1, y_2, \dots, y_k ($k = 2^r$), описывающие преобразователь $nSr \rightarrow kC1$, определяются из кодовых слов a_i как конъюнкции тех переменных, которые имеют в этом слове прямые значения. При построении ДШ2 необходимо использовать таблицы покрытия.

В табл. 3.13 приведена сложность дешифраторов для ряда кодов.

САМОПРОВЕРЯЕМЫЕ ТРИГГЕРНЫЕ УСТРОЙСТВА

4.1. Свойства парафазных схем

Важнейшим элементом любого дискретного устройства является триггер. Триггеры выполняют разнообразные функции. Они являются элементами памяти, элементами задержки, делителями частоты, счетчиками по модулю и т. п.

Классификация триггеров содержит много их разновидностей [18]. Прежде всего они делятся по логическим условиям работы, которые задаются таблицами переходов. Основными являются RS -, JK -, T -, D -, E -, R -, S -триггеры. По способу управления различают асинхронные и синхронные триггеры. Асинхронные схемы имеют только информационные (логические) входы и переключаются сразу же после изменения сигналов на этих входах. Синхронные триггеры имеют еще синхронизирующие входы. Сигналы синхронизации определяют моменты переключения триггерных схем в зависимости от состояния информационных входов. Здесь выделяют три группы триггеров: со статическим, динамическим управлением и двухступенчатые. При статическом управлении переключение происходит при условии достижения синхроимпульсом определенного уровня. Динамическое управление связано с переключением по фронту синхроимпульсов. Двухступенчатые схемы содержат два триггера, которые переключаются последовательно (для чего организуется две последовательности синхроимпульсов).

В самопроверяемых ДУ могут использоваться обычные и самопроверяемые триггеры. Если применяются обычные триггеры, то контроль их работы осуществляется методами, описанными в первой главе. Если самопроверяемые триггеры применяются в качестве элементов памяти, то задача синтеза самопроверяемых ДУ упрощается и они приобретают новые и полезные свойства. Конечно, самопроверяемые триггеры сложнее обычных, но при изготовлении их в виде интегральных микросхем это не имеет существенного значения.

В данной главе приводятся метод синтеза и схемы основных самопроверяемых триггеров. Так как они выполняются в виде парафазных устройств, то рассмотрим сначала свойства парафазных комбинационных схем.

В парафазной логике для представления двоичной переменной x выделяются две фазы (две линии): единичная фаза x^1 и нулевая фаза x^0 . Тогда логическая единица кодируется как $x^1x^0 = 10$, а логический нуль — как $x^1x^0 = 01$. Коды 10 и 01 являются рабочими, а коды 00 и 11 — ошибочными (защит-

ными). В табл. 4.1, 4.2 и 4.3 приведены таблицы истинности соответственно для функций И, ИЛИ и НЕ в парафазной логике. Из них следуют логические функции, которые описывают

Таблица 4.1

x_1	x_2	f
$x_1^1 x_1^0$	$x_2^1 x_2^0$	$f^1 f^0$
0 1	0 1	0 1
0 1	1 0	0 1
1 0	0 1	0 1
1 0	1 0	1 0

Таблица 4.2

x_1	x_2	f
$x_1^1 x_1^0$	$x_2^1 x_2^0$	$f^1 f^0$
0 1	0 1	0 1
0 1	1 0	1 0
1 0	0 1	1 0
1 0	1 0	1 0

схемы элемента И ($f^1 = x_1^1 x_2^1$, $f^0 = x_1^0 \vee x_2^0$), элемента ИЛИ ($f^1 = x_1^1 \vee x_2^1$, $f^0 = x_1^0 x_2^0$) и элемента НЕ ($f^1 = x^0$, $f^0 = x^1$). На рис. 4.1, а, б и в приведены схемы соответственно парафазных

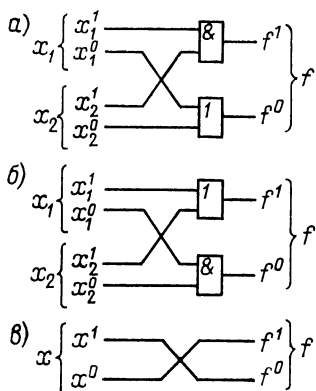


Рис. 4.1

элементов И, ИЛИ и НЕ, построенные по данным формулам. Функция НЕ реализуется за счет перекоммутации фаз (рис. 4.1, в) без использования какой-либо аппаратуры. Таким образом, любая парафазная комбинационная схема состоит из двух независимых подсхем, каждая из которых реализует монотонную функцию.

В табл. 4.4 приведена таблица неисправностей линий схемы элемента И. Из нее следует, что все одиночные неисправности приводят хотя бы на одном наборе проверяющего теста {0110, 1001, 1010} к нарушению парафазности на выходе и только

к этому. Это важно, поскольку нет замены сигналов типа 01→10 или 10→01 на выходе элемента. Значение выходного сигнала является либо правильным, либо защитным. То же самое происходит, очевидно, и при нарушении парафазности на входе элемента и активизации пути через него. Аналогичными свойствами обладают элементы ИЛИ и НЕ.

Таким образом, если построить однофазную избыточную реализацию булевой функции и затем получить ее парафазную реализацию заменой элементов И, ИЛИ, НЕ их парафазными схемами, то получим

Таблица 4.3

x	f
$x^1 x^0$	$f^1 f^0$
0 1	1 0
1 0	0 1

T -триггер имеет один парафазный вход T^1T^0 , парафазный выход Q^1Q^0 (рис. 4.3) и работает в соответствии с таблицей переходов, приведенной в табл. 4.5. Применяв для кодирования внутренних состояний код 4C2, получим кодированную таблицу переходов (табл. 4.6). По ней с помощью формулы (3.23) из [34] находим функции, описывающие схему парафазного T -триггера (рис. 4.4):

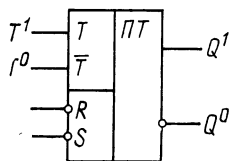


Рис. 4.3

$$\begin{aligned} y_1 &= T^0 y_1 \vee T^1 y_3, \\ y_2 &= T^0 y_2 \vee T^1 y_4, \\ y_3 &= T^0 y_2 \vee T^1 y_3, \\ y_4 &= T^0 y_1 \vee T^1 y_4, \\ Q^1 &= y_1, \quad Q^0 = y_2. \end{aligned} \quad (4.1)$$

Схема состоит из четырех бистабильных ячеек (элементы 1, 2; 3, 4; 5, 6; 7, 8) и имеет два вида обратных связей; обратные связи внутри бистабильных ячеек и обратные связи между ячей-

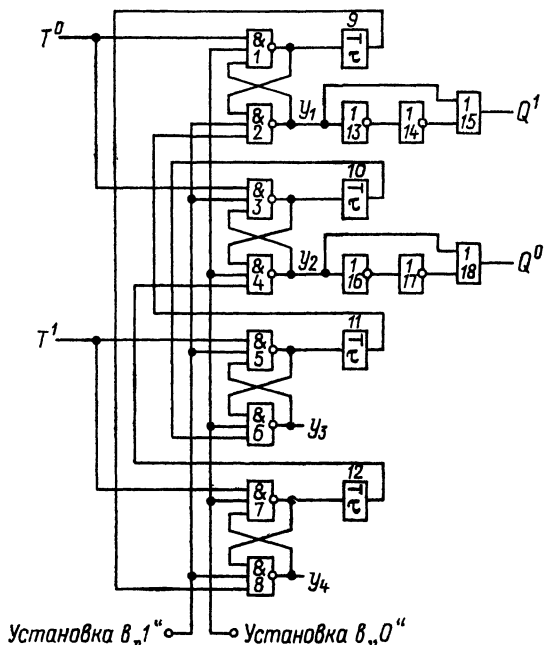


Рис. 4.4

ками, проходящие через линии задержки τ . Особенностью схемы является отсутствие выходного преобразователя. Элементы 13, 14, 15 и 16, 17, 18 образуют несимметричные линии задержки, которые исключают непарафазность сигналов Q^1 и Q^0 . При кратковременной подаче логического сигнала нуля на вход

Таблица 4.5

s	T^1T^0	
	0 1	1 0
1	(1), 01	2
2	3	(2), 10
3	(3), 10	4
4	1	(4), 01

Таблица 4.6

$y_1y_2y_3y_4$	T^1T^0	
	0 1	1 0
0 1 1 0	(0110), 01	1 0 1 0
1 0 1 0	1 0 0 1	(1010), 10
1 0 0 1	(1001), 10	0 1 0 1
0 1 0 1	0 1 1 0	(0101), 01

«Установка в 0» и наличии сигналов $T^1T^0 = 01$ схема приходит в устойчивое начальное состояние 0110 (состояние «0» триггера — см. табл. 4.6). При подаче нулевого сигнала на вход «Уста-

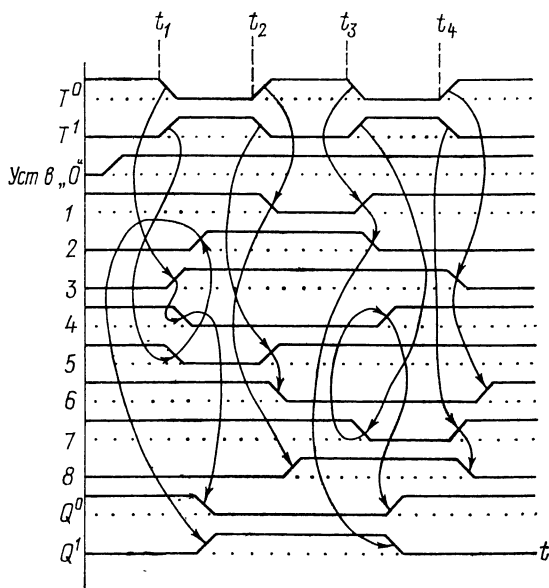


Рис. 4.5

новка 1» и наличии сигналов $T^1T^0 = 01$ схема приходит в устойчивое состояние 1001 (состояние «1» триггера). Полный цикл работы триггера происходит на входной последовательности сигналов T^1T^0 вида 01, 10, 01, 10, 01. При этом схема триггера последовательно проходит все свои состояния: 0110→1010→→1001→0101→0110. Данный процесс переключений отражен на временной диаграмме (рис. 4.5), где стрелками указаны причинно-следственные связи. В моменты времени t_1, t_2, t_3, t_4 происходит изменение входных сигналов. При построении вре-

менной диаграммы принято, что время задержки $\tau = 2\tau_3$, где τ_3 — время задержки сигнала одним элементом.

Основным ценным свойством парафазного T -триггера является его блокировка в защитном состоянии при возникновении одиночных отказов и при нарушении парафазности входного сигнала. Это свойство определяется тем, что для кодирования состояний использован равновесный код и схема построена в соответствии с принципами 1-реализации (см. § 1.3). Обнаружение всех одиночных отказов происходит на входной последовательности 01, 10, 01, 10. Например, на рис. 4.6 показана вре-

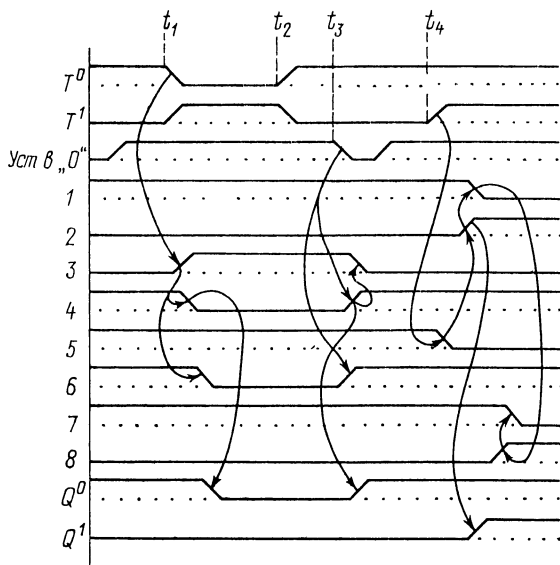


Рис. 4.6

менная диаграмма работы схемы при наличии в ней неисправности «константа 1» на выходе элемента 5. Если триггер находится в состоянии 0110, то при изменении входного сигнала 01→10 (см. момент времени t_1 на рис. 4.6) и при изменении сигнала 0→1 на входе T^1 не происходит изменения сигнала 1→0 на выходе элемента 5 (ср. рис. 4.5 и 4.6 в моменты времени t_1) и изменения сигнала 0→1 на выходе элемента 2. Зато происходит ложное изменение сигнала 1→0 на выходе элемента 6. Поэтому триггер из состояния 0110 кратковременно попадает в состояние 0010, а затем переходит в состояние 0000. Из системы функций (4.1) следует, что если все переменные y равны нулю, то они и остаются равными нулю навсегда. В результате (см. рис. 4.6) на выходах элементов 2, 4, 6 и 8 устанавливается логический сигнал 0, T -триггер переходит в защитное состояние 0000 и блокируется в нем ($Q^1 Q^0 = 00$). Последнее означает, что при

смене входного сигнала $10 \rightarrow 01$ (см. момент времени t_2 на рис. 4.6) и при всех последующих изменениях сигналов $T^1 T^0$ схема остается в состоянии 0000. Перевод схемы в начальное состояние 0110 осуществляется подачей сигнала 0 на вход установки (см. момент времени t_3 на рис. 4.6). Аналогичным образом работает неисправная схема и при отказах других элементов. При этом в зависимости от места и вида отказа происходит блокировка схемы либо в состоянии 0000, либо в состоянии 1111. Соответственно на выходе триггера фиксируются непарафазные сигналы 00 или 11.

Блокировка схемы в защитном состоянии происходит и при нарушении парафазности входного сигнала. Из системы функций (4.1) следует, что, если $T^1 = T^0 = 0$, то триггер в том же такте работы переходит в состояние 0000, а если $T^1 = T^0 = 1$, — переходит в состояние 1111. Процесс переключения бистабильных ячеек в последнем случае отражен на рис. 4.6 в момент времени t_4 . Следовательно, искажение входной информации фиксируется на выходе триггера путем нарушения парафазности непосредственно в такте возникновения этого искажения.

Описанные свойства парафазного T -триггера говорят о его «сверхчувствительности» по отношению к отказам и временным рассогласованиям логических сигналов. При этом, однако, усложняется проблема устойчивости схемы к состязаниям внутренних сигналов и временным сдвигам входных парафазных сигналов.

При переключении бистабильных ячеек критические состязания могут возникать на входах элементов И — НЕ. Например, в процессе переключения, показанного на рис. 4.5, в момент времени t_1 , возникают состязания на входах элемента 6, когда происходит одновременное изменение сигнала $1 \rightarrow 0$ на выходе элемента 5 и $0 \rightarrow 1$ на выходе элемента 3. Так как при этом сигнал 1 на выходе элемента 6 должен сохраниться, то для исключения таких состязаний необходимо, чтобы изменение сигнала $0 \rightarrow 1$, поступающее по цепи обратной связи между бистабильными ячейками, было задержано относительно изменения сигнала $1 \rightarrow 0$, поступающего по цепи обратной связи внутри бистабильных ячеек, на некоторое время $\tau_{л.з}$. Эту задачу выполняют линии задержки (элементы 9—12 на рис. 4.4). При построении временных диаграмм на рис. 4.5 и 4.6 предполагалось, что линии задержки состоят из двух элементов И — НЕ и поэтому $\tau_{л.з} = 2\tau_z$. Длительность $\tau_{л.з}$ зависит также от установленного допустимого относительного сдвига входных парафазных сигналов $\tau_{вх}$.

Определим сдвиг фронтов логических сигналов на входах элемента 6 (на выходах элементов 5 и 10) при условии абсолютной синхронизации сигналов T^1 и T^0 . Обозначим через $t(i \rightarrow f)$ задержку логического сигнала между линиями i и f (номер линии совпадает с номером элемента, к выходу кото-

рого эта линия подключена). Тогда для процесса переключения триггера $0110 \rightarrow 1010$ (см. момент времени t_1 на рис. 4.5) имеем: $t(T^0 \rightarrow 10) - t(T^1 \rightarrow 5) = t(T^0 \rightarrow 3) + \tau_{л.з} - t(T^1 \rightarrow 5) = \tau_э + \tau_{л.з} - \tau_э = \tau_{л.з}$. Пусть теперь сигнал T^1 отстает от сигнала T^0 на время $\tau_{вх}$. В этом случае $t(T^0 \rightarrow 10) - t(T^1 \rightarrow 5) = \tau_{л.з} - \tau_{вх}$. Поскольку условием отсутствия критических состязаний является соотношение $\tau_{л.з} - \tau_{вх} > 0$, то для устойчивой работы схемы требуется, чтобы выполнялось неравенство $\tau_{л.з} > \tau_{вх}$. Это же соотношение обеспечивает устойчивую работу схемы в том случае, когда сигнал T^0 отстает от сигнала T^1 на время $\tau_{вх}$.

Таким образом, увеличивая $\tau_{л.з}$, можно решить проблему устойчивости схемы к возможному сдвигу входных сигналов. Однако при этом ухудшаются другие характеристики триггера:

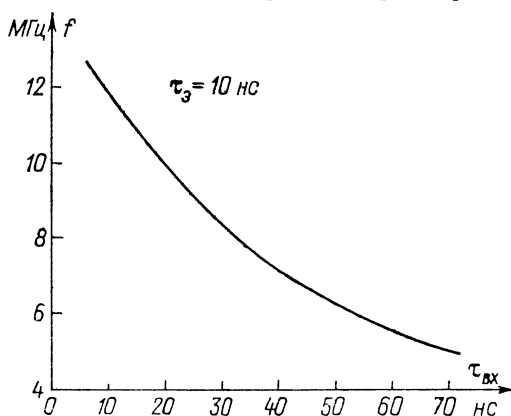


Рис. 4.7

увеличивается сдвиг выходных парафазных сигналов $\tau_{вых}$, который вносится схемой, и увеличивается время переключения триггера. Как следует из рис. 4.5, при переключении триггера типа $0110 \rightarrow 1010$ (момент времени t_1) сигнал y_2 на выходе элемента 4 опережает сигнал y_1 на выходе элемента 2 на время $\tau_{л.з}$. В самом деле, $\tau_{вых} = t(T^1 \rightarrow 2) - t(T^0 \rightarrow 4) = t(T^1 \rightarrow 5) + t(5 \rightarrow 2) - t(T^0 \rightarrow 3) - t(3 \rightarrow 4) = \tau_э + (\tau_{л.з} + \tau_э) - \tau_э - \tau_э = \tau_{л.з}$. При переключении триггера типа $1001 \rightarrow 0101$ (момент времени t_3), наоборот, сигнал y_1 опережает сигнал y_2 на время $\tau_{л.з}$. Чтобы исключить эту непарафазность сигналов на выходах триггера Q^1 и Q^0 , используются несимметричные линии задержки на элементах 13—18. Они дают задержку сигнала при его изменении вида $1 \rightarrow 0$ на время $3\tau_э$, а при изменении вида $0 \rightarrow 1$ — на время $\tau_э$. Время задержки, осуществляемое на элементах 13, 14 и 16, 17, должно быть равным $\tau_{л.з}$ (в данном случае $2\tau_э$). В такой схеме, если на вход триггера поступает парафазный сигнал с временным сдвигом $\tau_{вх}$, этот же сдвиг наблюдается и на выходах Q^1 , Q^0 .

Время переключения t_n триггера измеряется от момента начала изменения сигналов T^1 и T^0 до момента окончания переходных процессов. Оно равно при всех типах переключений в схеме $3\tau_3 + \tau_{л.з}$. Например, при переключении типа 0110 \rightarrow 1010 (момент времени t_1 на рис. 4.5) имеем $t_n = t(T^1 \rightarrow 2) = t(T^1 \rightarrow 5) + t(5 \rightarrow 2) + t(2 \rightarrow 15) = \tau_3 + \tau_{л.з} + \tau_3 + \tau_3 = 3\tau_3 + \tau_{л.з}$. Время задержки распространения входного сигнала, вносимое схемой триггера, $t_3 = t_n$.

Минимальная длительность входного управляющего импульса $t_n = t_n$, а разрешающее время триггера $t_p \geq t_n + t_n = 6\tau_3 + \tau_{л.з}$. Тогда максимальная частота переключения триггера

$$f_{\max} = \frac{1}{t_p} = \frac{1}{6\tau_3 + 2\tau_{л.з}}.$$

Из сказанного следует, что частотно-временные характеристики T -триггера зависят от максимального допустимого временного сдвига парафазных входных сигналов, так как $\tau_{л.з} > \tau_{вх}$. На рис. 4.7 показана зависимость $f_{\max}(\tau_{вх})$ при $\tau_3 = 10$ нс. Это является отличительным свойством полностью самопроверяемых триггерных устройств, тогда как у обычных триггеров эта характеристика отсутствует.

4.3. Универсальный метод синтеза самопроверяемых триггерных устройств

Сформулируем перечень требований к ПСП-схеме любого триггерного устройства, который вытекает из анализа свойств

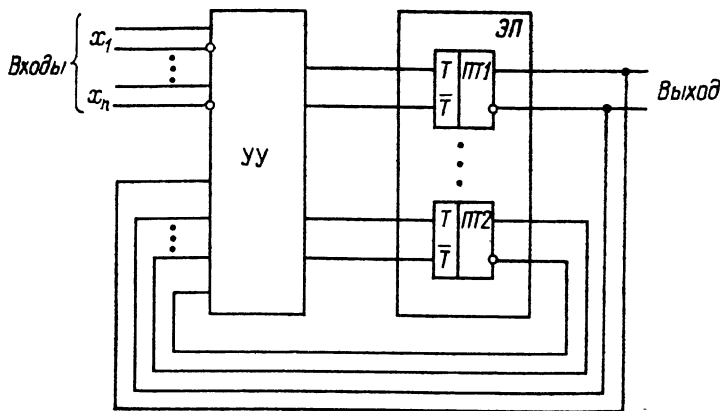


Рис. 4.8

T -триггера, выполненного в предыдущем параграфе. Схема ПСП-триггера должна:

- 1) выполнять логику работы триггера данного типа;

- 2) при возникновении одиночных отказов блокироваться в защитном состоянии;
- 3) при нарушении парафазности любого входного сигнала блокироваться в защитном состоянии;
- 4) обеспечивать вывод триггера из защитного состояния только по цепи установки;
- 5) быть устойчивой к состязаниям между внутренними сигналами;
- 6) обладать по возможности максимальным быстродействием и рабочей частотой (для данной элементной базы);

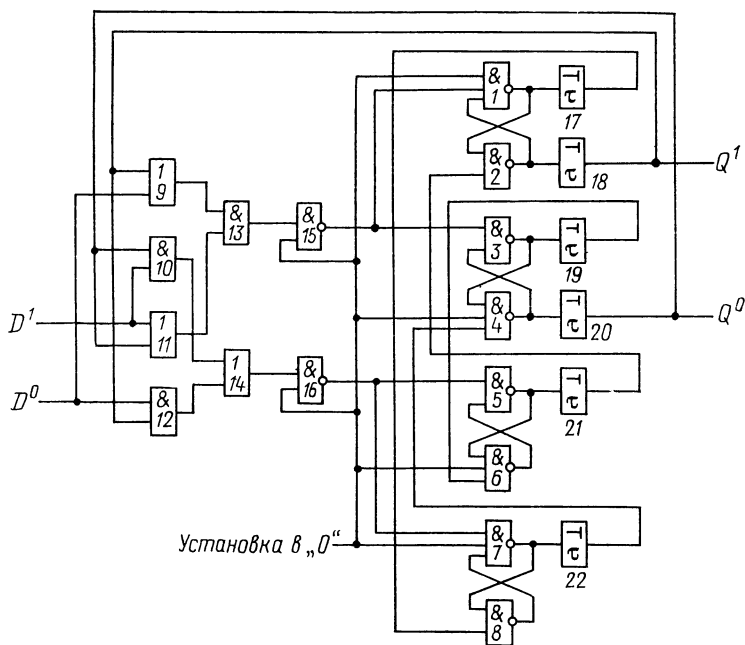


Рис. 4.9

- 7) быть устойчивой к временному сдвигу парафазных входных сигналов до определенного (заданного) порога;
- 8) давать минимальный возможный временной сдвиг парафазных выходных сигналов.

Из перечисленных требований видно, что при реализации схемы ПСП-триггера приходится решать противоречивые задачи. Это связано с тем, что свойство самопроверяемости, определяя высокую чувствительность триггеров к отказам элементов и искажениям внешних и внутренних сигналов, вступает в противоречие с устойчивостью их работы, так как в реальных условиях трудно добиться строгой парафазности входных сигналов. Поэтому в состав структуры ПСП-триггеров включаются опре-

деленным образом линии задержки, как это показано на примере T -триггера.

Для синтеза ПСП-триггеров любого типа будем использовать композиционную модель, отражающую структуру триггера как дискретного автомата (рис. 4.8). Она состоит из устройства управления ($УУ$) и элементов памяти ($ЭП$). В качестве последних используются парафазные T -триггеры. Устройство управления есть парафазная комбинационная схема, реализующая функции включения T -триггеров. Эти функции получаются в результате синтеза данного триггера по его таблице переходов.

Самопроверяемость схемы (рис. 4.8) обеспечивается тем, что для ее построения используются ПСП-триггеры и ПСП-схема управления. Согласно утверждению 4.1 вся схема триггера будет самопроверяемой, если на вход устройства управления поступает проверяющий тест. Это последнее условие и надо выполнить при синтезе ПСП-триггера.

4.4. Асинхронный D -триггер

Осуществим синтез описанным методом схемы парафазного D -триггера, работа которого задается таблицей переходов (табл. 4.7). Если использовать для построения схемы в качестве элемента памяти T -триггер, то согласно правилам, описанным в § 1.1, получим функцию его включения $T = D\bar{Q} \vee \bar{D}Q$ или в парафазном виде:

$$T^1 = D^1Q^0 \vee D^0Q^1, \quad (4.2)$$

$$T^0 = (D^1 \vee Q^0) (D^0 \vee Q^1).$$

Таблица 4.7

Q	D	
	0	1
0	(0)	1
1	0	(1)

Таблица 4.8

$y_1y_2y_3y_4$	D^1D^0		Q^1Q^0
	0 1	1 0	
0 1 1 0	(0110)	1 0 1 0	0 1
1 0 1 0	~	1 0 0 1	1 0
1 0 0 1	0 1 0 1	(1 0 0 1)	1 0
0 1 0 1	0 1 1 0	~	0 1

Формула (4.2) описывает устройство управления (см. рис. 4.8). На рис. 4.9 представлена схема парафазного D -триггера. Она состоит из схемы T -триггера (элементы 1—8), схемы управления (элементы 9—16) и линий задержки (элементы 17—22). Линии задержки 18 и 20 образуют третий вид обратной связи —

глобальные обратные связи между T -триггером и устройством управления. Они являются несимметричными (см. элементы 13—18 на рис. 4.4). Элементы 15 и 16 на рис. 4.9 необходимы для того, чтобы при установке схемы в нулевое состояние зафиксировать требуемые значения сигналов на входах T -триггера. В табл. 4.8 приведена кодированная таблица переходов, отражающая процессы переключений в схеме. Состояние $y_1y_2y_3y_4 = 0110$ есть нулевое состояние D -триггера. Изменение парафазного входного сигнала D^1D^0 вида $01 \rightarrow 10$ переключает

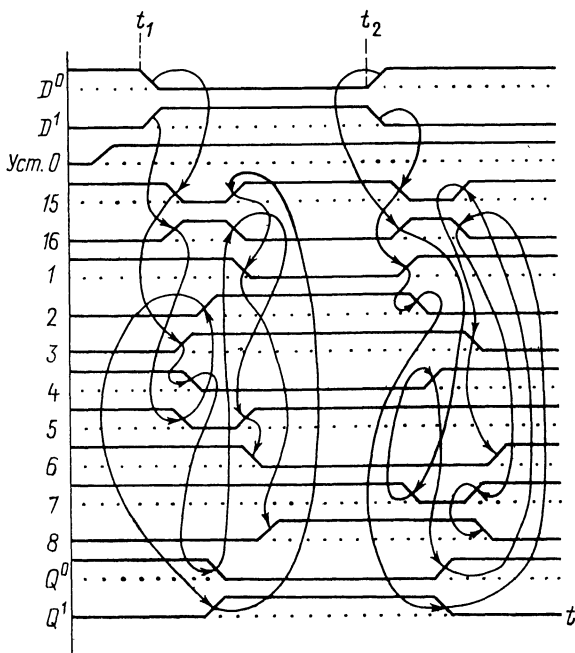


Рис. 4.10

схему сначала в состояние 1010, а затем, после вступления в действие глобальных обратных связей, — в устойчивое состояние 1001, которое является состоянием «1» D -триггера (см. табл. 4.8). Парафазный выход схемы Q^1Q^0 принимает значение 10. Данный процесс отражен на временной диаграмме (рис. 4.10) в момент времени t_1 . При смене входных сигналов вида $10 \rightarrow 01$ схема сначала переключается в состояние 0101, а затем в устойчивое состояние 0110 (см. табл. 4.8 и момент времени t_2 на рис. 4.10) и выходы Q^1Q^0 принимают значения 01. Таким образом, на выход триггера транслируется парафазный сигнал, который появляется на его входе. В связи с этим D -триггер можно назвать самопроверяемым повторителем парафазного сигнала.

Из временной диаграммы (рис. 4.10) видно, что при переключении D -триггера в новое состояние T -триггер переключается дважды за счет действия глобальных обратных связей. Для исчерпывающего тестирования всей схемы достаточно обеспечить двухкратное изменение входного вектора D^1D^0 : 01, 10, 01. При этом T -триггер, переключившись четыре раза, получает на своих входах (выходах элементов 15 и 16) полный тест. Для тестирования комбинационной управляющей схемы достаточно на ее входах $D^1D^0Q^1Q^0$ обеспечить четыре набора 0101, 1001, 1010, 1001, что также выполняется при двухкратном изменении входного сигнала (см. рис. 4.10). Таким образом, схема (рис. 4.10) является самопроверяемой.

Частотно-временные характеристики D -триггера отличаются от характеристик T -триггера вследствие наличия схемы управления (элементы 9—16 на рис. 4.9). Время переключения равно $11\tau_3 + 2\tau_{л.з}$. Например, при переключении типа $0 \rightarrow 1$ (см. момент времени t_1 на рис. 4.10) $t_n = t(D^1 \rightarrow 8) = t(D^1 \rightarrow 16) + t(16 \rightarrow 5) + t(5 \rightarrow 2) + t(2 \rightarrow 18) + t(18 \rightarrow 15) + t(15 \rightarrow 1) + t(1 \rightarrow 8) = 3\tau_3 + \tau_3 + (\tau_{л.з} + \tau_3) + \tau_3 + 3\tau_3 + \tau_3 + (\tau_{л.з} + \tau_3) = 11\tau_3 + 2\tau_{л.з}$. Время задержки выходного сигнала относительно входного, вносимое схемой D -триггера, меньше времени переключения и равно $t_3 = t(D^1 \rightarrow 18) = 6\tau_3 + \tau_{л.з}$. Максимальная частота переключения

$$f_{\max} = \frac{1}{22\tau_3 + 4\tau_{л.з}}.$$

4.5. Триггеры с двумя входами

Основным видом триггера с двумя входами является JK -триггер, работа которого задается табл. 4.9. Строки 1 и 4 таблицы соответствуют нулевому состоянию триггера, а строки 2 и 3 — единичному состоянию. Используем для построения схемы два парафазных T -триггера. Кодированная таблица переходов представлена в табл. 4.10. Из нее получаем функции включения элементов памяти:

Таблица 4.9

s	JK			
	0 0	0 1	1 0	1 1
1	(1), 0	(1), 0	2	3
2	(2), 1	1	(2), 1	4
3	2	1	2	(3), 1
4	1	1	2	(4), 0

Таблица 4.10

$Y_1 Y_2$	JK			
	0 0	0 1	1 0	1 1
0 1	0 1	0 1	1 0	1 1
1 0	1 0	0 1	1 0	0 0
1 1	1 0	0 1	1 0	1 1
0 0	0 1	0 1	1 0	0 0

$$T_1 = KY_1(\bar{J} \vee \bar{Y}_2) \vee J\bar{Y}_1(\bar{K} \vee Y_2),$$

$$T_2 = \bar{K}Y_2(J \vee Y_1) \vee \bar{J}\bar{Y}_2(K \vee \bar{Y}_1)$$

или в парафазном виде:

$$T_1^1 = K^1Y_1^1(J^0 \vee Y_2^0) \vee J^1Y_1^0(K^0 \vee Y_2^1),$$

$$T_1^0 = (K^0 \vee Y_1^0 \vee J^1Y_2^1)(J^0 \vee Y_1^1 \vee K^1Y_2^0),$$

$$T_2^1 = K^0Y_2^1(J^1 \vee Y_1^1) \vee J^0Y_2^0(K^1 \vee Y_1^0),$$

$$T_2^0 = (K^1 \vee Y_2^0 \vee J^0Y_1^0)(J^1 \vee Y_2^1 \vee K^0Y_1^1).$$

(4.3)

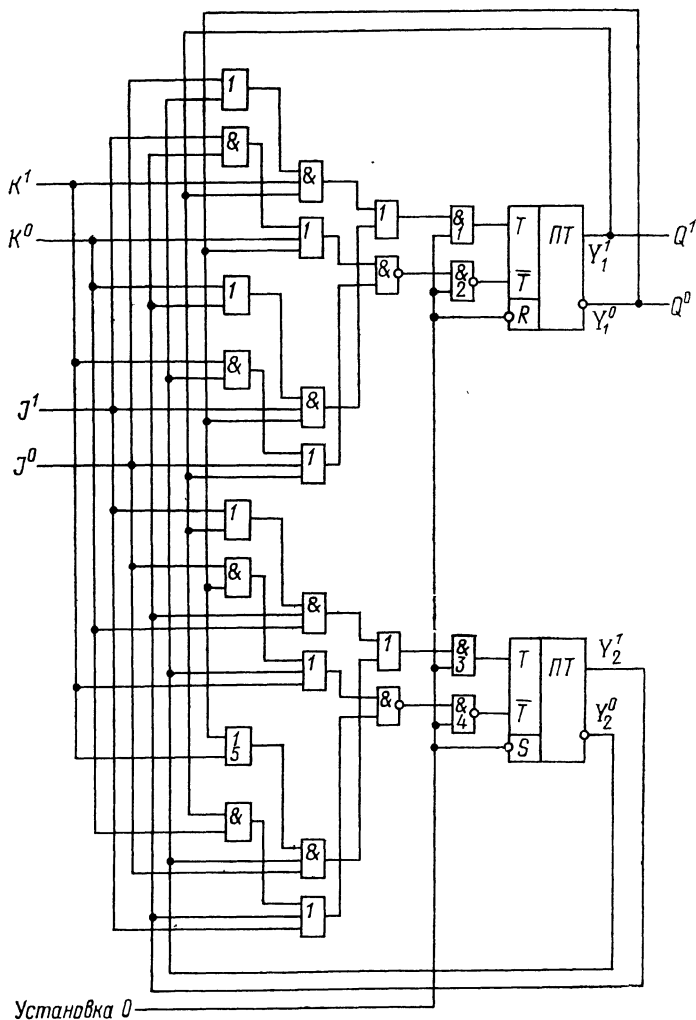


Рис. 4.11

Схема парафазного JK -триггера (рис. 4.11) содержит два полностью самопроверяемых T -триггера с несимметричными линиями задержки (см. рис. 4.4), схему управления, построенную по системе формул (4.3), и элементы установки 1—4. Критические состязания между элементами памяти исключены за счет противогоночного кодирования (см. табл. 4.10). На рис. 4.12 временная диаграмма иллюстрирует процесс переключения триггера из состояния 1 в состояние 2 при изменении значений входов JK с 00 на 10.

Для исчерпывающего тестирования схемы JK -триггера достаточно, чтобы в процессе работы схема приходила в каждую клетку таблицы переходов. Например, тестовой является такая последовательность сигналов на входах JK : 00, 10, 11, 00, 11, 01, 11, 10, 11, 10, 01, 11, 00, 11, 01. При отказах элементов схемы типа $0 \rightarrow 1$ или $1 \rightarrow 0$ происходит блокировка обоих T -триггеров в состояниях, когда на их выходах устанавливаются одинаковые сигналы 11 или 00. Рассмотрим, например, отказ типа «константа 1» выхода элемента ИЛИ, отмеченного на рис. 4.11 как элемент 5. В этом случае в системе (4.3) искажается функция

$$T_2^1 = K^0 Y_2^1 (J^1 \vee Y_1^1) \vee J^0 Y_2^0. \tag{4.4}$$

Одна из ситуаций, при которой данный отказ обнаруживается, состоит в следующем. Пусть на вход триггера поступают сигналы $JK = 00$ и он приходит в состояние 2 ($Y_1 Y_2 = 10$) и в этот момент происходит отказ. Тогда имеем значение функций

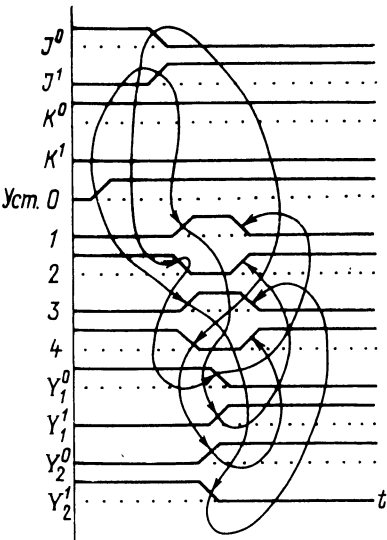


Рис. 4.12

Таблица 4.11

Q	RS			
	00	01	10	11
0	(0)	1	(0)	~
1	(1)	(1)	0	~

Таблица 4.12

Q	RS			
	00	01	10	11
0	(0)	1	(0)	(0)
1	(1)	(1)	0	(1)

(согласно (4.3) и (4.4)): $T_1^1 = 0$, $T_1^0 = 1$, $T_2^1 = 1$, $T_2^0 = 1$. Следовательно, нарушается парафазность на входе T -триггера Y_2 (на выходах элементов 3 и 4), и он блокируется в состоянии,

когда $Y_2^0 = Y_2^1 = 1$. Триггер Y_1 работает правильно, и выходы Q^1Q^0 имеют истинные значения. Если теперь на входы JK поступают сигналы 11, то согласно (4.3) получаем: $T_1^0 = 1$, $T_1^1 = 1$. Поэтому происходит блокировка и триггера Y_1 , а выхо-

Таблица 4.13

Q	RS			
	00	01	10	11
0	(0)	1	(0)	(0)
1	(1)	(1)	0	0

Таблица 4.14

Q	RS			
	00	01	10	11
0	(0)	1	(0)	1
1	(1)	(1)	0	(1)

ды Q^1 и Q^0 принимают значение 1. Из состояния блокировки JK -триггер переводится в начальное состояние ($Y_1Y_2 = 01$) подачей сигнала 0 на вход «Установка в 0».

Таблица 4.15

Тип триггера	Асинхронные	Синхронные
D	$T^1 = D^1Q^0 \vee D^0Q^1$, $T^0 = (D^1 \vee Q^0)(D^0 \vee Q^1)$	$T^1 = D^1C^1Q^0 \vee D^0C^1Q^1$, $T^0 = (D^0 \vee C^0 \vee Q^1)(D^1 \vee C^0 \vee Q^0)$
RS	$T^1 = S^1Q^0 \vee R^1Q^1$, $T^0 = (S^0 \vee Q^1)(R^0 \vee Q^0)$	$T^1 = S^1C^1Q^0 \vee R^1C^1Q^1$, $T^0 = (S^0 \vee C^0 \vee Q^1)(R^0 \vee C^0 \vee Q^0)$
E	$T^1 = R^0S^1Q^0 \vee R^1S^0Q^1$, $T^0 = (R^1 \vee S^0 \vee Q^1)(R^0 \vee S^1 \vee Q^0)$	$T^1 = R^0S^1Q^0C^1 \vee R^1S^0Q^1C^1$, $T^0 = (R^1 \vee S^0 \vee Q^1 \vee C^0)(R^0 \vee S^1 \vee Q^0 \vee C^0)$
R	$T^1 = R^0S^1Q^0 \vee R^1Q^1$, $T^0 = (R^1 \vee S^0 \vee Q^1)(R^0 \vee Q^0)$	$T^1 = R^0S^1Q^0C^1 \vee R^1Q^1C^1$, $T^0 = (R^1 \vee S^0 \vee Q^1 \vee C^0)(R^0 \vee Q^0 \vee C^0)$
S	$T^1 = R^1S^0Q^1 \vee S^1Q^0$, $T^0 = (R^0 \vee S^1 \vee Q^0)(S^0 \vee Q^1)$	$T^1 = R^1S^0Q^1C^1 \vee S^1Q^0C^1$, $T^0 = (R^0 \vee S^1 \vee Q^0 \vee C^0)(S^0 \vee Q^1 \vee C^0)$
JK	$T_1^1 = K^1Y_1^1(J^0 \vee Y_2^0) \vee$ $\vee J^1Y_1^0(K^0 \vee Y_2^1)$, $T_1^0 = (K^0 \vee Y_1^0 \vee J^1Y_2^1) \wedge$ $\wedge (J^0 \vee Y_1^1 \vee K^1Y_2^0)$, $T_2^1 = K^0Y_2^1(J^1 \vee Y_1^1) \vee$ $\vee J^0Y_2^0(K^1 \vee Y_1^0)$, $T_2^0 = (K^1 \vee Y_2^0 \vee J^0Y_1^0) \wedge$ $\wedge (J^1 \vee Y_2^1 \vee K^0Y_1^1)$	$T_1^1 = K_1Y_1^1(J^0 \vee Y_2^0)C^1 \vee$ $\vee J^1Y_1^0(K^0 \vee Y_2^1)C^1$, $T_1^0 = (K^0 \vee Y_1^0 \vee J^1Y_2^1 \vee C^0) \wedge$ $\wedge (J^0 \vee Y_1^1 \vee K^1Y_2^0 \vee C^0)$, $T_2^1 = K^0Y_2^1(J^1 \vee Y_1^1)C^1 \vee$ $\vee J^0Y_2^0(K^1 \vee Y_1^0)C^1$, $T_2^0 = (K^1 \vee Y_2^0 \vee J^0Y_1^0 \vee C^0) \wedge$ $\wedge (J^1 \vee Y_2^1 \vee K^0Y_1^1 \vee C^0)$

Аналогичным образом осуществляется синтез и других полностью самопроверяемых триггерных схем, среди которых основными являются RS -триггер (табл. 4.11), E -триггер (табл. 4.12), R -триггер (табл. 4.13) и S -триггер (табл. 4.14). В табл. 4.15 представлены структурные формулы полной системы ПСП-триггеров, управляемых уровнем сигнала (статических). Все схемы, кроме JK -триггера, содержат в своем составе один парафазный T -триггер. Здесь также приведены формулы для синхронных триггеров, имеющих парафазный синхронный вход C^1C^0 . Если $C^1C^0 = 01$, то, как следует из формул табл. 4.15, всегда $T^1T^0 = 01$ и воздействия на триггер по информационным входам не происходит. В связи с этим на триггер не влияет и кратковременное нарушение парафазности информационных сигналов. Поэтому применение синхронных ПСП-триггеров эффективно решает проблему синхронизации парафазных сигналов.

4.6. Триггеры с динамическим управлением

Часто при проектировании дискретных систем используются триггеры с динамическим управлением записью информации [18]. Такие триггеры «прозрачны» для входного сигнала лишь короткое время в момент перепада синхронизирующего импульса на его фронте и (или) спаде. В другие моменты времени триггер независимо от уровня синхрои́мпульса не воспринимает информационных сигналов. Логика работы динамических триггеров сложнее логики работы статических, и поэтому их схемы также сложнее.

Таблица 4.16

Состояние	CD				Y_1Y_2
	00	01	10	11	
1	(1), 0	(1), 0	4	3	0 1
2	(2), 1	(2), 1	4	3	1 0
3	2	2	(3), 1	(3), 1	1 1
4	1	1	(4), 0	(4), 0	0 0

Рассмотрим для примера таблицу переходов полностью самопроверяемого D -триггера с динамическим управлением по фронту синхрои́мпульса C (табл. 4.16). Строки 1 и 4 таблицы соответствуют нулевому состоянию триггера, а строки 2 и 3 — единичному состоянию. Если $C = 0$ и триггер находится в состоянии 0 (строка 1), то переход в состояние 1 осуществляется при изменении входов типа $00 \rightarrow 11$ или $01 \rightarrow 11$. Если $C = 1$ и

Таблица 4.17

Тип триггера	Функциональное описание
D	$T_1^1 = C^1 (D^1 Y_1^0 Y_2^1 \vee D^0 Y_1^1 Y_2^0),$ $T_1^0 = C^0 \vee (D^0 \vee Y_1^1 \vee Y_2^0) (D^1 \vee Y_1^0 \vee Y_2^1),$ $T_2^1 = C^0 (Y_1^0 Y_2^0 \vee Y_1^1 Y_2^1) \vee C^1 (D^1 Y_1^1 Y_2^0 \vee D^0 Y_1^0 Y_2^1),$ $T_2^0 = (C^1 \vee (Y_1^1 \vee Y_2^1) (Y_1^0 \vee Y_2^0)) (C^0 \vee (D^0 \vee Y_1^0 \vee Y_2^1) (D^1 \vee Y_1^1 \vee Y_2^0))$
T	$T_1^1 = C^1 T^1 (Y_1^0 Y_2^1 \vee Y_1^1 Y_2^0),$ $T_1^0 = C^0 \vee T^0 \vee (Y_1^1 \vee Y_2^0) (Y_1^0 \vee Y_2^1),$ $T_2^1 = C^0 (Y_1^0 Y_2^0 \vee Y_1^1 Y_2^1) \vee C^1 T^0 (Y_1^0 Y_2^1 \vee Y_1^1 Y_2^0),$ $T_2^0 = (C^1 \vee (Y_1^1 \vee Y_2^1) (Y_1^0 \vee Y_2^0)) (C^0 \vee T^1 \vee (Y_1^1 \vee Y_2^0) (Y_1^0 \vee Y_2^1))$
JK	$T_1^1 = C^1 (J^1 K^0 Y_1^0 Y_2^1 \vee J^0 K^1 Y_1^0 Y_2^1),$ $T_1^0 = C^0 \vee (J^0 \vee K^1 \vee Y_1^1 \vee Y_2^0) (J^1 \vee K^0 \vee Y_1^1 \vee Y_2^0),$ $T_2^1 = C^0 (Y_1^0 Y_2^0 \vee Y_1^1 Y_2^1) \vee C^1 (J^0 Y_1^0 Y_2^1 \vee K^0 Y_1^1 Y_2^0),$ $T_2^0 = (C^1 \vee (Y_1^1 \vee Y_2^1) (Y_1^0 \vee Y_2^0)) (C^0 \vee (J^1 \vee Y_1^1 \vee Y_2^0) (K^1 \vee Y_1^0 \vee Y_2^1))$
E	$T_1^1 = C^1 (J^1 K^0 Y_1^0 Y_2^1 \vee J^0 K^1 Y_1^0 Y_2^1),$ $T_1^0 = C^0 \vee (J^0 \vee K^1 \vee Y_1^1 \vee Y_2^0) (J^1 \vee K^0 \vee Y_1^1 \vee Y_2^0),$ $T_2^1 = C^0 (Y_1^0 Y_2^0 \vee Y_1^1 Y_2^1) \vee C^1 (Y_1^0 Y_2^1 (J^0 \vee K^1) \vee Y_1^1 Y_2^0 (J^1 \vee K^0)),$ $T_2^0 = (C^1 \vee (Y_1^1 \vee Y_2^1) (Y_1^0 \vee Y_2^0)) (C^0 \vee (Y_1^1 \vee Y_2^0 \vee J^1 K^0) (Y_1^0 \vee Y_2^1 \vee J^0 K^1))$
R	$T_1^1 = C^1 (K^1 Y_1^1 (Y_2^0 \vee J^1) \vee J^1 K^0 Y_1^0 Y_2^1),$ $T_1^0 = C^0 \vee (K^0 \vee Y_1^0 \vee Y_2^1 J^0) (J^0 \vee K^1 \vee Y_1^1 \vee Y_2^0),$ $T_2^1 = C^0 (Y_1^0 Y_2^0 \vee Y_1^1 Y_2^1) \vee C^1 (Y_1^0 Y_2^1 (J^0 \vee K^1) \vee K^0 Y_1^1 Y_2^0 \vee J^0 K^1 Y_2^1),$ $T_2^0 = (C^1 \vee (Y_1^1 \vee Y_2^1) (Y_1^0 \vee Y_2^0)) (C^0 \vee (Y_1^1 \vee Y_2^0 \vee J^1 K^0) \wedge (K^1 \vee Y_1^0 \vee Y_2^1) (J^1 \vee K^0 \vee Y_2^0))$
S	$T_1^1 = C^1 (J^1 Y_1^0 (Y_2^1 \vee K^1) \vee J^0 K^1 Y_1^1 Y_2^0),$ $T_1^0 = C^0 \vee (J^0 \vee Y_1^1 \vee Y_2^0 K^0) (J^1 \vee K^0 \vee Y_1^0 \vee Y_2^1),$ $T_2^1 = C^0 (Y_1^0 Y_2^0 \vee Y_1^1 Y_2^1) \vee C^1 (Y_1^1 Y_2^0 (J^1 \vee K^0) \vee J^0 Y_1^0 Y_2^1 \vee J^1 K^1 Y_2^0),$ $T_2^0 = (C^1 \vee (Y_1^1 \vee Y_2^1) (Y_1^0 \vee Y_2^0)) (C^0 \vee (Y_1^0 \vee Y_2^1 \vee J^0 K^1) \wedge (J^1 \vee Y_1^1 \vee Y_2^0) (J^0 \vee K^0 \vee Y_2^1))$

триггер находится в состоянии 0 (строка 4), то при всех изменениях входов это состояние сохраняется. Если $C = 0$ и триггер находится в состоянии 1 (строка 2), то переход в состояние 0 происходит при изменении входов типа $00 \rightarrow 10$ или $01 \rightarrow 10$.

Таблица 4.18

Состоя- ние	CT			
	00	01	10	11
1	(1), 0	(1), 0	4	3
2	(2), 1	(2), 1	3	4
3	2	2	(3), 1	(3), 1
4	1	1	(4), 0	(4), 0

Если $C = 1$ и триггер находится в состоянии 1 (строка 3), то при всех изменениях входов это состояние сохраняется. Таким образом, для построения данного типа триггера необходимы два

Таблица 4.19

Состоя- ние	CJK							
	000	001	010	011	100	101	110	111
1	(1), 0	(1), 0	(1), 0	(1), 0	4	4	3	3
2	(2), 1	(2), 1	(2), 1	(2), 1	3	4	3	4
3	2	2	2	2	(3), 1	(3), 1	(3), 1	(3), 1
4	1	1	1	1	(4), 0	(4), 0	(4), 0	(4), 0

Таблица 4.20

Состоя- ние	Тип триггера									
	E, R, S							E	R	S
	CRS									
	000	001	010	011	100	101	110	111	111	111
1	(1), 0	(1), 0	(1), 0	(1), 0	4	3	4	4	4	3
2	(2), 1	(2), 1	(2), 1	(2), 1	3	3	4	3	4	3
3	2	2	2	2	(3), 1	(3), 1	(3), 1	(3), 1	(3), 1	(3), 1
4	1	1	1	1	(4), 0	(4), 0	(4), 0	(4), 0	(4), 0	(4), 0

элемента памяти (два парафазных T -триггера). В правом крайнем столбце табл. 4.16 указан вариант противофазного кодирования, а в табл. 4.17 приведены формулы, описывающие схему управления D -триггера в парафазном виде.

В табл. 4.18—4.20 приведены таблицы переходов ПСП-триггеров с динамическим управлением по фронту синхроимпульса типов T , JK , E , R и S . Вариант кодирования, указанный в табл. 4.16, является противофазным для всех триггеров, и для этого варианта в табл. 4.17 дано функциональное описание схем.

4.7. Двухступенчатые триггеры

Двухступенчатые триггеры, или триггеры MS -типа, содержат два элемента памяти, которые переключаются последовательно во времени. Первый элемент (ведущий) воспринимает информацию, поступающую по информационным входам, и переключается в первый момент времени. Во второй момент времени переключается ведомый элемент памяти, который служит для

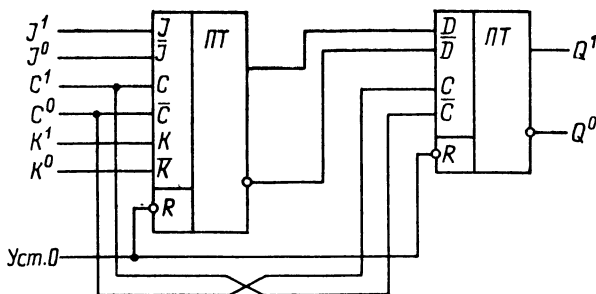


Рис. 4 13

запоминания и хранения информации. Такое переключение обеспечивается с помощью двухфазной синхронизации, запрещающих связей или инверторов синхронизирующего сигнала [18]. Двухступенчатые триггеры используются, если устойчивая работа цифровых схем требует задержек в переключении триггеров, например, при исключении критических состязаний элементов памяти [32].

В парафазной реализации наиболее простую структуру имеют ПСП-триггеры MS -типа, использующие инверсию синхронизирующего сигнала. При этом возможны два способа их построения. При первом способе ведущая ступень реализует заданный алгоритм работы триггера, а ведомая выполняется в виде полностью самопроверяемого синхронного D -триггера (см. рис. 4.9). Для управления ведомой ступенью используется инверти-

рованный синхросигнал, получаемый путем перекоммутации его единичной и нулевой фаз. На рис. 4.13 приведена схема полностью самопроверяемого двухступенчатого JK -триггера, построенного данным способом. Первая ступень его — это полностью самопроверяемый синхронный JK -триггер. Если $C^1C^0 = 01$, то триггер закрыт для приема информации. При поступлении тактового импульса, когда $C^1C^0 = 10$, первая ступень воспринимает информацию по входам J^1J^0 и K^1K^0 . В это время D -триггер не реагирует на изменения сигналов по входам D^1D^0 , так как на его входах C^1C^0 присутствуют сигналы 01. После окончания

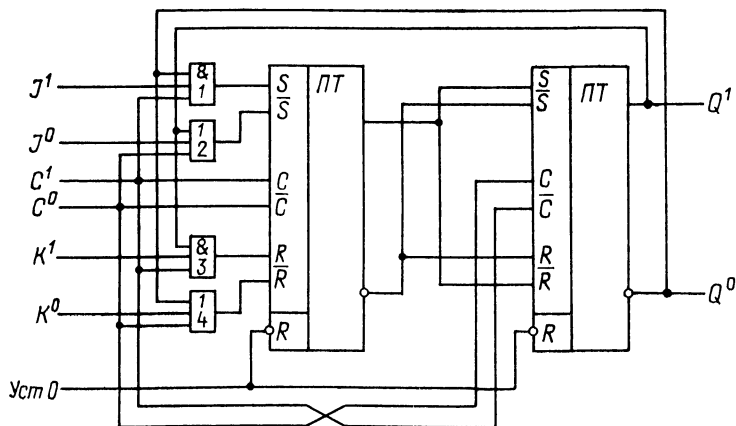


Рис. 4.14

тактового импульса, когда $C^1C^0 = 01$, информация переносится из первой во вторую ступень и на выходах D -триггера Q и \bar{Q} появляются новые значения сигналов.

Свойство самопроверяемости данной схемы определяется самопроверяемостью обеих ступеней. Блокировка в защитном состоянии JK -триггера приводит к немедленной блокировке и D -триггера, поскольку происходит нарушение парафазности на входах D, \bar{D} . Быстродействие двухступенчатого триггера зависит от быстродействия обеих ступеней. При этом длительность тактового импульса должна быть больше времени переключения первой ступени.

Второй способ построения ПСП-триггера MS -типа состоит в том, что логика его работы реализуется с помощью обеих ступеней. В этом случае существуют обратные связи между второй и первой ступенями. На рис. 4.14 приведена такая схема JK -триггера на базе полностью самопроверяемых синхронных RS -триггеров. Если схема находится в нулевом состоянии ($Q^1Q^0 = 01$), то открыта для приема информации парафазная схема И (элементы 1 и 2). Тогда, если $J^1J^0 = 10$, то при поступ-

лении тактового импульса ($C^1C^0 = 10$) на входы S, \bar{S} первой ступени подаются сигналы 10 и первый RS -триггер переключается в состояние 1. После окончания тактового импульса переключается в состояние 1 и второй RS -триггер. При этом открывается для приема парафазная схема И на элементах 3 и 4 и возникает возможность воздействия на первую ступень сигналов на входах K^1, K^0 . Обратные связи между второй и первой ступенями нужны, собственно, для того, чтобы обеспечить переключение триггера в случае, если $JK = 11$. При этом, если $Q^1Q^0 = 01$, то работает схема И на элементах 1 и 2 и триггер переключается в состояние 1, а если $Q^1Q^0 = 10$, то работает схема И на элементах 3 и 4 и триггер переключается в состояние 0. Легко убедиться, что на входы элементов 1 — 4 в процессе работы триггера поступает проверяющий тест. Поэтому данная схема является ПСП.

4.8. Синтез самопроверяемых ДУ на самопроверяемых триггерах

Разработка полного набора полностью самопроверяемых триггерных устройств позволяет по-новому поставить проблему синтеза самопроверяемых ДУ. Описанные в первой главе методы синтеза решают эту задачу с использованием обычных эле-

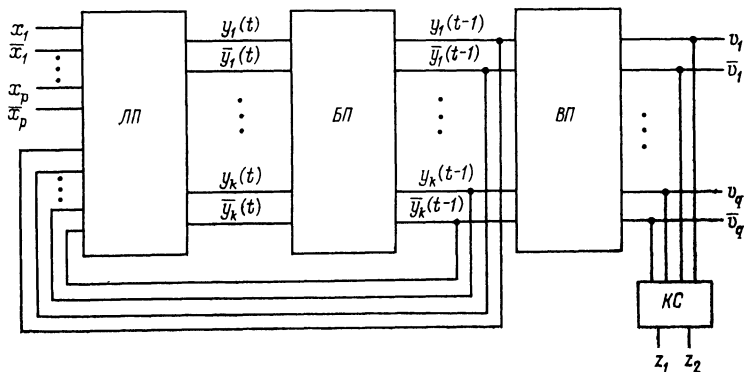


Рис. 4.15

ментов памяти. Однако ясно, что если сам элемент памяти является самопроверяемым, то и процедуры построения ПСП-схем будут более простыми.

Общая структура полностью самопроверяемых дискретных устройств на самопроверяемых триггерах показана на рис. 4.15. Особенностью ее является то, что все внешние и внутренние сигналы схемы представлены в парафазном виде. Пусть в качестве элемента памяти используется полностью самопроверяе-

мый синхронный *D*-триггер с динамическим управлением. Важным является свойство триггера блокироваться в защитном состоянии, при котором $Q = \overline{Q}$, в случае возникновения одиночных отказов и нарушения парафазности входов *D* и *C*. Это свойство позволяет строить самопроверяемые синхронные схемы без избыточного кодирования состояний кодом с обнаружением ошибок.

При синтезе используются следующие правила:

1. Состояния ДУ кодируются безызыбыточным кодом.
2. Логический и выходной преобразователи строятся в виде парафазных ПСП-схем.
3. В качестве элементов памяти применяются синхронные парафазные ПСП-триггеры.
4. Контрольная схема подключается к выходам блока *ВП* и является $\frac{q}{2}/q$ -СПТ.

Осуществим синтез синхронного ДУ, заданного табл. 4.21. Для безызыбыточного кодирования состояний, приведенного в табл. 4.22, получаем функции включения элементов памяти и выходные функции:

Таблица 4.21

<i>s</i>	<i>x</i>	
	0	1
1	1,01	3,01
2	1,01	3,10
3	4,11	2,00
4	3,10	1,00

$$\begin{aligned}
 y_1 &= \overline{x}y_1 \vee x\overline{y}_1, \\
 y_2 &= \overline{x}y_1\overline{y}_2 \vee x\overline{y}_1y_2, \\
 v_1 &= \overline{x}y_1 \vee x\overline{y}_1\overline{y}_2, \\
 v_2 &= \overline{x} (\overline{y}_1 \vee \overline{y}_2) \vee x\overline{y}_1\overline{y}_2.
 \end{aligned}
 \tag{4.5}$$

При построении схемы по формулам (4.5) блоки *ЛП* и *ВП* реализуются в парафазном виде (рис. 4.16). Покажем, что данная схема является ПСП. Для этого надо провести анализ трех типов отказов: в блоках *ЛП*, *БП* и *ВП*. Пусть произошел

Таблица 4.22

<i>s</i>	<i>y</i> ₁ <i>y</i> ₂	<i>x</i>	
		0	1
1	0 0	00, 01	10, 01
2	0 1	00, 01	10, 10
3	1 0	11, 11	01, 00
4	1 1	10, 10	00, 00

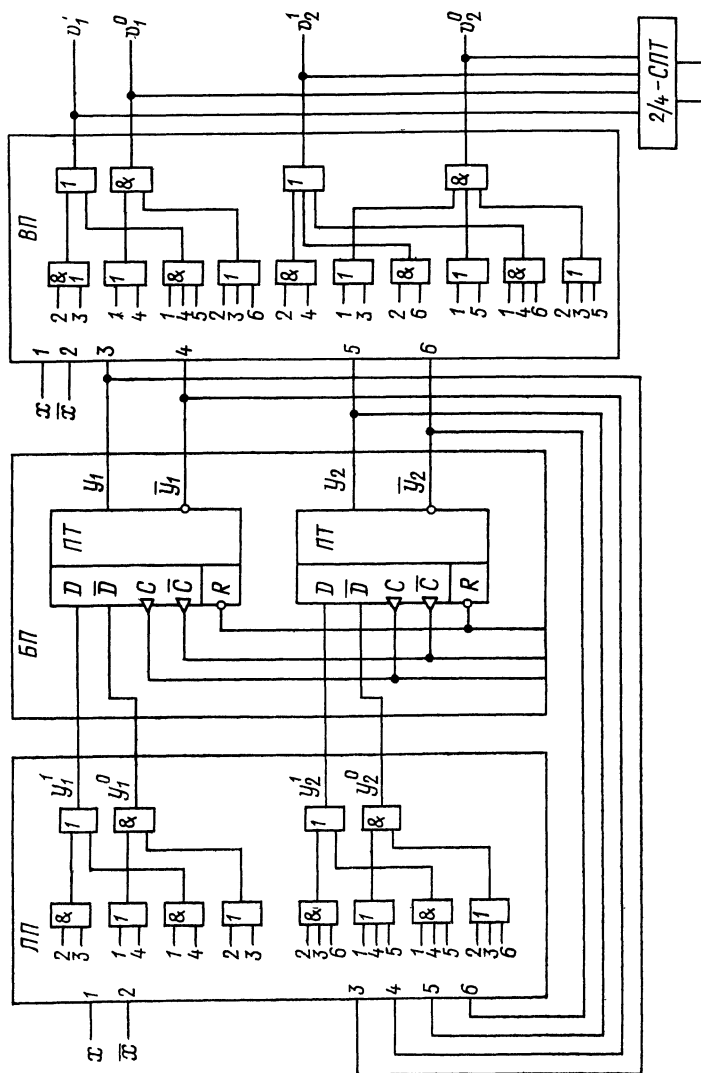


Рис. 4.16

отказ в блоке *ВП*. Поскольку кодирование состояний ДУ является безызбыточным, то функции y и v безызбыточны. Это значит, что для любой неисправности *ВП* найдется такое состояние схемы и значение входов, при которых неисправность проявляется нарушением парафазности на одном из выходов блока *ВП*, что фиксируется контрольной схемой. Например, отказ «константа 1» входа z у элемента *ВП*, отмеченного на рис. 4.16 как элемент 1, проявится, когда схема будет находиться в состоянии 01 при значении $x = 0$. При этом для исправной схемы $v_1^1 = 0$, $v_1^0 = 1$, $v_2^1 = 1$, $v_2^0 = 0$, а для неисправной $v_1^1 = 1$, $v_1^0 = 1$, $v_2^1 = 1$, $v_2^0 = 0$.

Отказы триггеров и их блокировка в защитном состоянии приводят к нарушению парафазности переменной y на входах блока *ВП*. Это вызывает непарафазность выходов *ВП* и обнаруживается контрольной схемой. Отказы элементов *ЛП* приводят к нарушению парафазности входов D и \bar{D} триггеров, что сопровождается блокировкой триггеров и также фиксируется контрольной схемой.

Итак, имеет место

Утверждение 4.2. Парафазное ДУ при использовании ПСП-триггеров и безызбыточного кодирования состояний является полностью самопроверяемым.

Данное утверждение справедливо при применении ПСП-триггера любого типа. Следует отметить, что парафазное ДУ при возникновении любого отказа после нескольких тактов работы приходит в состояние, когда все его триггеры будут заблокированы в защитном состоянии, а на всех выходах блока *ВП* появятся непарафазные сигналы. По этой причине такие ДУ можно рекомендовать для построения схем с высокими требованиями по безопасности.

ГЛАВА ПЯТАЯ

САМОПРОВЕРЯЕМЫЕ ТИПОВЫЕ ЦИФРОВЫЕ УСТРОЙСТВА

5.1. Двоичные счетчики и регистры

В данной главе рассмотрим основные типовые цифровые устройства, построенные с использованием ПСП-триггеров, дешифраторов и контрольных схем, а также их применение для организации ПСП контроля дискретных систем.

Исправность схемы легко контролируется на парафазных выходах последнего триггера. Если возникает неисправность в каком-либо из триггеров, он блокируется в защитном состоя-



Последнее обстоятельство определяет основной недостаток счетчиков с трактом последовательного переноса. Задержка распространения сигнала в них растет пропорционально числу разрядов. От этого недостатка свободны счетчики с параллельным переносом. На рис. 5.2 показана схема полностью самопроверяемого синхронного счетчика с параллельным переносом, собранная на синхронных T -триггерах с динамическим управлением по фронту синхроимпульса. Сигнал на переключение x^1x^0 поступает на входы C, \bar{C} всех триггеров одновременно, и происходит одновременное переключение тех из них, на входы T, \bar{T} которых поданы переменные 1 и 0. Это будет, если все триггеры младших разрядов находятся в состоянии 1, что проверяется с помощью парафазных схем И (элементы 1, 2 и 3, 4). Элементы 3 и 4 формируют парафазный сигнал переноса CR ,

когда все триггеры находятся в состоянии 1. Контрольные выходы подключаются к выходам триггера старшего разряда, который блокируется в защитном состоянии при возникновении неисправности в нем самом или в любом из триггеров младших разрядов. Элементы, образующие парафазные схемы И, тестируются в процессе работы счетчика. Например, на входы элемента 1 поступает тест 00, 01, 10, 11 в процессе прохождения счет-

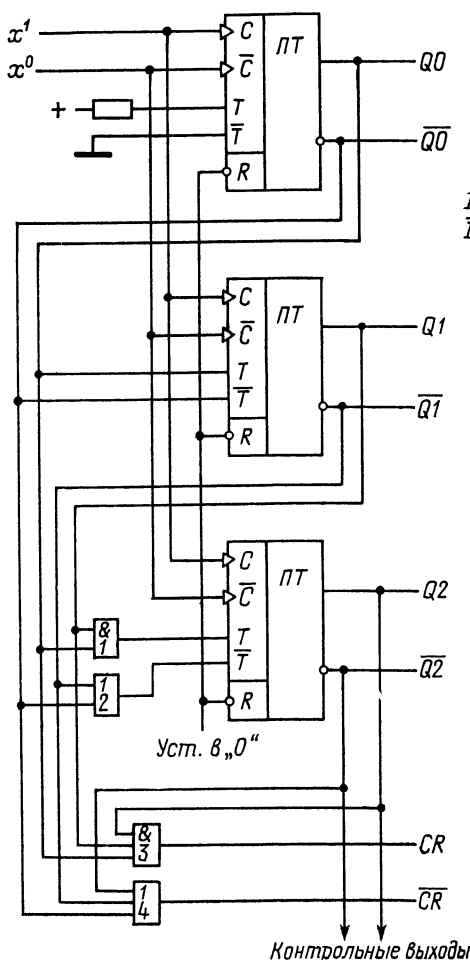


Рис. 5.2

руются в процессе работы счетчика. Например, на входы элемента 1 поступает тест 00, 01, 10, 11 в процессе прохождения счет-

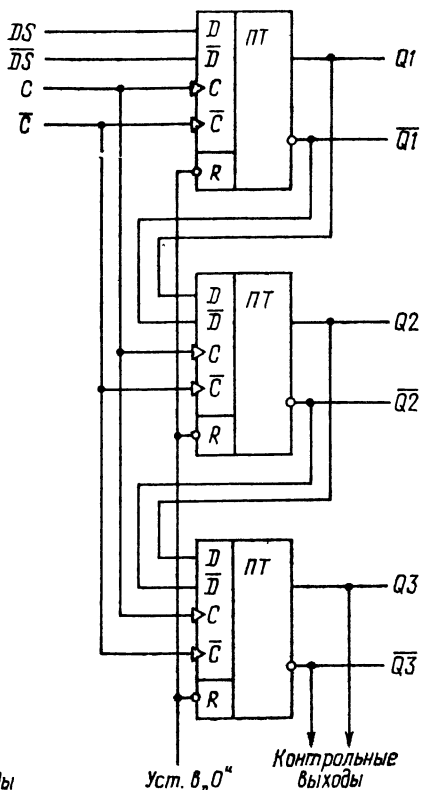


Рис. 5.3

чика через состояния (Q_0, Q_1, Q_2) : 000, 100, 010, 110. При отказе элемента 1 на входах T, \bar{T} триггера Q_2 происходит нарушение парафазности (элемент 2 работает правильно) и триггер Q_2 блокируется в защитном состоянии. Отказы элементов 3 и 4 проявляются на выходах CR, \overline{CR} и фиксируются последующей счетной схемой.

Аналогичные принципы могут быть использованы при построении ПСП-схем счетчиков других видов, а также при построении двоичных регистров. Сдвиговый регистр строится на синхронных D -триггерах (рис. 5.3). На входы DS , \overline{DS} поступают данные в последовательном коде. По мере поступления импульсов сдвига на входы C , \overline{C} происходит запись данных в триггеры регистра и выдача их в параллельном коде на выходах Q . При возникновении неисправности в каком-либо триггере регистра он блокируется в защитном состоянии и при поступлении импульсов сдвига в защитное состояние переходят все последующие триггеры вплоть до последнего, выходы которого являются одновременно и контрольными выходами.

5.2. Распределители

Распределители осуществляют параллельно-последовательное распределение импульсов, поступающих на вход устройства в последовательном виде. В общем случае распределитель

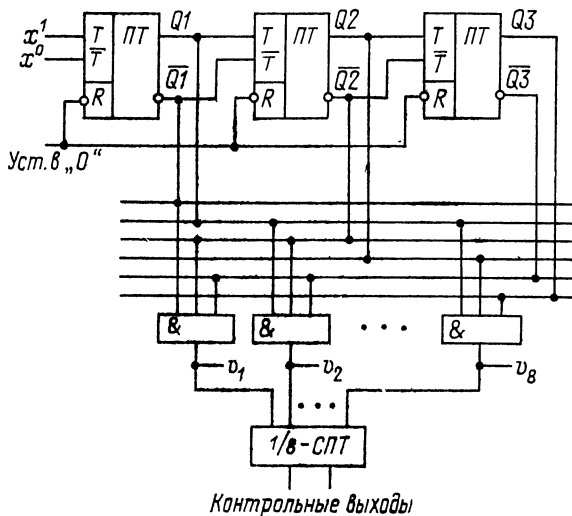


Рис. 5.4

состоит из счетчика и дешифратора. Счетчик считает импульсы, а дешифратор определяет, какое число записано в счетчик и подключает соответствующий выход. В ПСП-распределителях надо применять самопроверяемые счетчики и самопроверяемые дешифраторы.

Широко распространена на практике схема распределителя, состоящая из двоичного счетчика и дешифратора, построенного на конъюнкторах. Ее ПСП-вариант приведен на рис. 5.4. Конт-

рольные выходы формируются с помощью тестера 1/8-СПТ, который фиксирует наличие только одной логической единицы на выходах распределителя $v_1 \div v_8$. Отказы элементов И приводят к нарушению веса 1 на входе тестера. То же происходит и при блокировке триггеров в защитном состоянии. Например, блокировка триггера $Q3$ в состоянии, когда $Q3 = \overline{Q3} = 0$, устанавливает на всех выходах v логический нуль.

Таблица 5.1

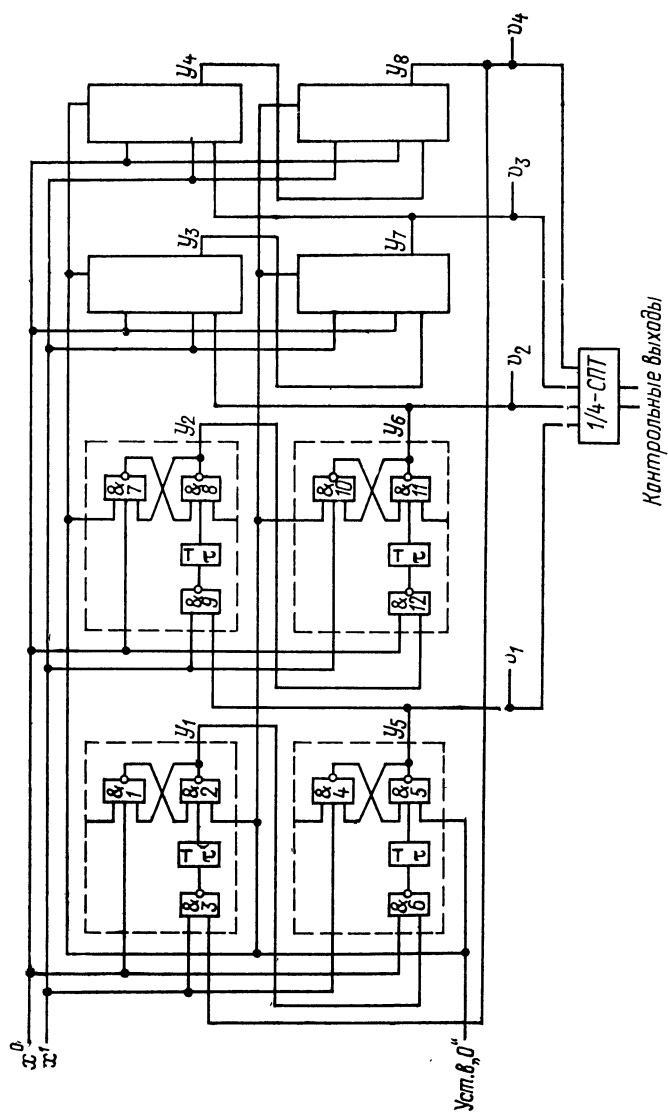
s	x^1x^0	
	01	10
1	(1), 1000	2, 1000
2	3, 0100	(2), 1000
3	(3), 0100	4, 0100
4	5, 0010	(4), 0100
5	(5), 0010	6, 0010
6	7, 0001	(6), 0010
7	(7), 0001	8, 0001
8	1, 1000	(8), 0001

Таблица 5.2

s	$y_1y_2y_3y_4y_5y_6y_7y_8$	x^1x^0	
		01	10
1	1 0 0 0 1 0 0 0	1000 1000	0100 1000
2	0 1 0 0 1 0 0 0	0100 0100	0100 1000
3	0 1 0 0 0 1 0 0	0100 0100	0010 0100
4	0 0 1 0 0 1 0 0	0010 0010	0010 0100
5	0 0 1 0 0 0 1 0	0010 0010	0001 0010
6	0 0 0 1 0 0 1 0	0001 0001	0001 0010
7	0 0 0 1 0 0 0 1	0001 0001	1000 0001
8	1 0 0 0 0 0 0 1	1000 1000	1000 0001

Другой разновидностью распределителя является схема, работающая в унитарном коде. В табл. 5.1 приведена таблица переходов такого распределителя на четыре позиции. Его переход в новую позицию происходит по отрицательному фронту парафазного входного импульса, т. е. при изменении сигналов x^1x^0 вида $10 \rightarrow 01$. Синтез схемы осуществим методом кодирования состояний по столбцам таблицы переходов с использованием 1-реализации (см. первую главу). В каждом столбце таблицы переходов имеется четыре устойчивых состояния, которые кодируются кодом 4C1 (табл. 5.2). При этом переменные y_5, y_6, y_7 и y_8 повторяют значения выходных переменных v_1, v_2, v_3 и v_4 , что позволяет исключить из структуры схемы выходной преобразователь. С помощью формулы (3.25) в [34] получаем функции включения элементов памяти:

$$\begin{aligned}
 y_1 &= x_1y_1 \vee x_2y_8, & y_5 &= x_1y_1 \vee x_2y_5, \\
 y_2 &= x_1y_2 \vee x_2y_5, & y_6 &= x_1y_2 \vee x_2y_6, \\
 y_3 &= x_1y_3 \vee x_2y_6, & y_7 &= x_1y_3 \vee x_2y_7, \\
 y_4 &= x_1y_4 \vee x_2y_7, & y_8 &= x_1y_4 \vee x_2y_8, \\
 v_1 &= y_5, & v_2 &= y_6, & v_3 &= y_7, & v_4 &= y_8.
 \end{aligned}
 \tag{5.1}$$



Контрольные выходы

Рис. 5.5

Система (5.1) имеет регулярную структуру и строится из стандартных модулей, реализующих функции y_i (рис. 5.5). Из таких модулей может быть собрана схема распределителя для любого числа позиций n . При этом требуется $2n$ модулей. Линии задержки, включенные в состав модулей, пре-

Таблица 5.3

s	$Q_3 Q_2 Q_1$
0	0 0 0
1	0 0 1
2	0 1 1
3	1 1 1
4	1 1 0
5	1 0 0
0	0 0 0

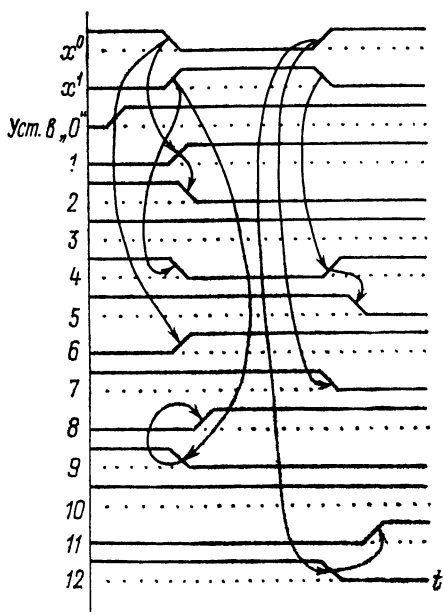


Рис. 5.6

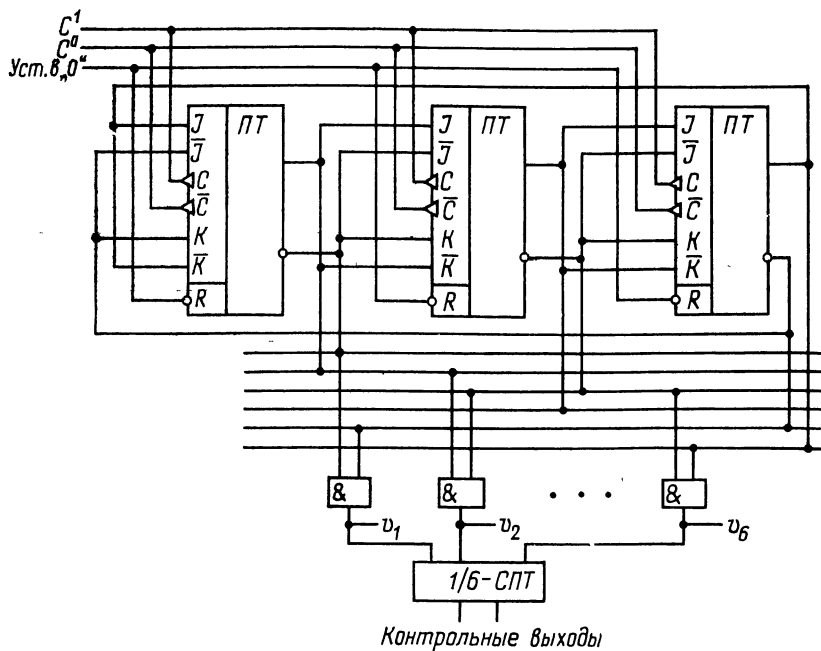


Рис. 5.7

дотвращают состязания, возникающие на входах элементов 2, 5, 8, 11, ... при одновременном изменении фронтов сигналов x^0 и x^1 . На рис. 5.6 приведена временная диаграмма, отражающая работу распределителя при поступлении на входы x^1 , x^0 первого парафазного тактового импульса.

Для построения кольцевых распределителей [30] удобно использовать триггеры с двумя входами. На рис. 5.7 показана ПСП-схема кольцевого распределителя на шесть позиций, построенная на базе счетчика Джонсона с использованием парафазных JK -триггеров. При поступлении первых трех импульсов на тактовый вход C^1C^0 счетчик работает на сложение в унитарном коде (табл. 5.3), а при поступлении следующих трех импульсов — на вычитание. Выходы распределителя организуются по формулам:

$$\begin{aligned}v_1 &= \bar{Q}_1\bar{Q}_3, \quad v_2 = Q_1\bar{Q}_2, \quad v_3 = Q_2\bar{Q}_3, \\v_4 &= Q_1Q_3, \quad v_5 = \bar{Q}_1Q_2, \quad v_6 = \bar{Q}_2Q_3.\end{aligned}$$

5.3. Генераторы и схемы синхронизации

На базе ПСП-триггеров могут быть построены различные генераторные схемы, обладающие свойством самопроверяемости. На рис. 5.8 приведена схема полностью самопроверяемого триггера-генератора, построенного с использованием D -триггера. Она описывается уравнениями:

$$\begin{aligned}D^1 &= S^1Q^0 \vee R^0Q^1, \\D^0 &= (S^0 \vee Q^1)(R^1 \vee Q^0).\end{aligned}$$

Таблица 5.4

Q	RS			
	00	01	10	11
0	(0)	1	(0)	1
1	(1)	(1)	0	0

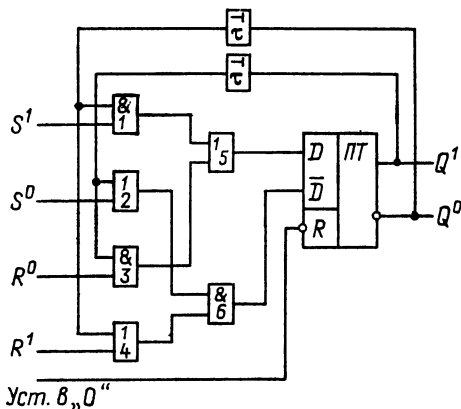


Рис. 5.8

Из таблицы переходов (табл. 5.4) видно, что схема работает как RS -триггер за исключением случая, когда $R = S = 1$. Этот случай для RS -триггера является запрещенным; схема (рис. 5.8) при таком состоянии парафазных входов переходит в генераторный режим работы (см. временную диаграмму на рис. 5.9).

При возникновении одиночных неисправностей и нарушении парафазности входных сигналов генерация прекращается и схема блокируется в защитном состоянии. Так как время переключе-

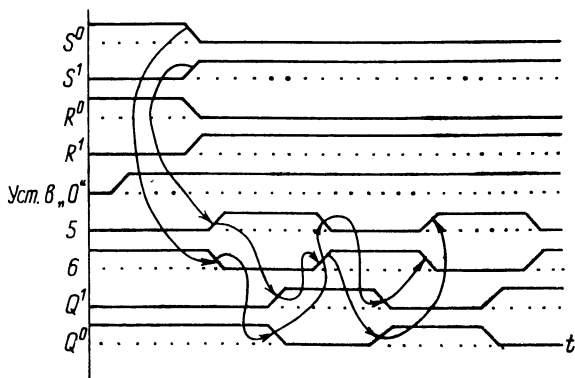


Рис. 5.9

ния D -триггера равно $11\tau_3 + 2\tau_{л.з.}$, а задержка сигнала $6\tau_3 + \tau_{л.з.}$ (см. § 4.4), то задержка в цепи обратной связи генератора должна быть $\tau_{о.с.} \geq t_n - t_3 = 5\tau_3 + \tau_{л.з.}$. В противном случае сигналы на входах D -триггера изменятся раньше, чем закончатся переходные процессы внутри его схемы. Время переключения схемы генератора равно $t_n = 13\tau_3 + \tau_{л.з.}$, а частота работы

$$f = \frac{1}{26\tau_3 + 2\tau_{л.з.}}$$

В синхронных схемах генераторы импульсов работают совместно с различными служебными схемами, образующими систему синхронизации. Последняя служит для согласования во времени всех процессов в цифровой схеме. В самопроверяемых схемах целесообразно, чтобы и сама система синхронизации обладала этим же свойством. На рис. 5.10 изображена самопроверяемая схема двухфазной синхронизации, состоящая из ПСП тактового генератора ($TГ$), парафазного T -триггера, линий

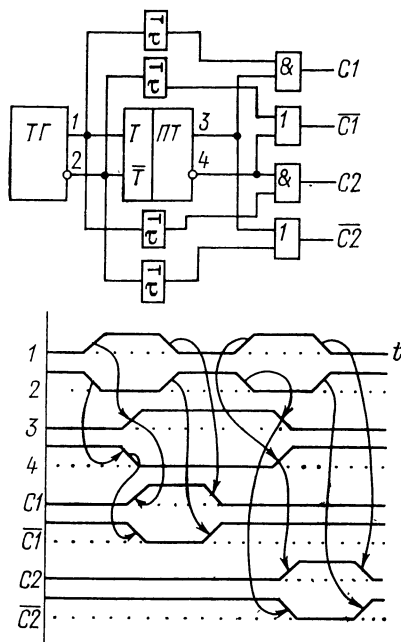


Рис. 5.10

задержек и элементов И и ИЛИ. На выходах $C1$, $\overline{C1}$ и $C2$, $\overline{C2}$ формируются последовательности парафазных синхроимпульсов (см. временную диаграмму на рис. 5.10). Линии задержки служат для исключения состязаний на входах элементов И и ИЛИ. Задержка τ должна быть не менее времени переключения T -триггера. Самопроверяемость схемы обеспечивается самопроверяемостью $TГ$ и $ПТ$, а также тем, что на входы элементов И и ИЛИ в процессе динамической работы поступает проверяющий тест. При возникновении отказов нарушается парафазность сигналов $C1$ и $C2$, что приводит к блокировке в защитном состоянии ПСП-триггеров цифровой схемы, на которые подаются эти сигналы.

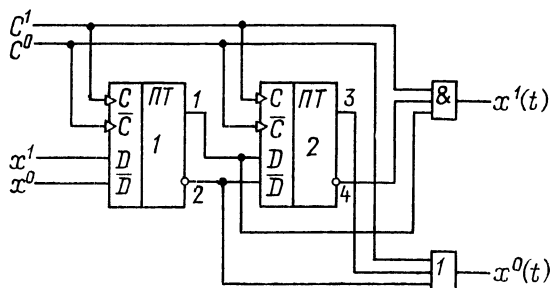


Рис. 5.11

Для синхронизации входных сигналов используются специальные устройства — синхронизаторы [30]. На рис. 5.11 приведена ПСП-схема синхронизатора парафазного сигнала x . Она состоит из двух полностью самопроверяемых D -триггеров и выходной парафазной схемы И. Входной сигнал x является асинхронным, т. е. может изменяться в любой момент времени и иметь произвольную длительность. Сигнал $x(t)$ является синхронным, т. е. его изменения совпадают по времени с изменением синхроимпульса C и он имеет длительность этого импульса. Нетрудно проверить, что при однократном изменении сигнала x на входы элементов И и ИЛИ поступает проверяющий тест. Поэтому схема синхронизатора является самопроверяемой.

5.4. Контрольная парафазная логика

В том случае, если дискретная система состоит из ПСП-блоков A_1, A_2, \dots, A_n (рис. 5.12), которые имеют самопроверяемые схемы внутреннего контроля ($ССВК$) и два контрольных парафазных выхода, возникает проблема построения общей схемы контроля ($СК$) с парафазным выходом z . Задача $СК$ состоит в контроле парафазности выходов всех $ССВК$ по заданным условиям. При этом удобно ввести в рассмотрение специальную

контрольную парафазную логику (КПЛ), которая отличается от обычной парафазной логики представлением логических сигналов 0 и 1. В КПЛ логический сигнал 1 представляется значениями двух фаз парафазного сигнала 01 или 10, а логический сигнал 0 — значениями 00 или 11. Таким образом, в КПЛ логический сигнал 1 соответствует сигналу исправности контролируемого ПСП-блока, а сигнал 0 является сигналом отказа.

На рис. 5.13 приведены условное обозначение и схема двухвходового логического элемента И в КПЛ. Он имеет парафазные входы и выход. Их фазы нельзя интерпретировать как нулевую и единичную, и поэтому они различаются числом штрихов. Особенностью работы такого элемента по сравнению с работой элемента И в обычной логике является то, что в исправном

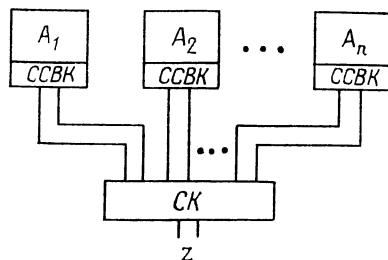


Рис. 5.12

(рабочем) состоянии контролируемой системы на его входы поступают только сигналы $x = y = 1$, т. е. четыре входных набора: $x'y''y'' = \{0101, 0110, 1001, 1010\}$. Поэтому логический элемент И в КПЛ будем называть самопроверяемым, если каждая его одиночная неисправность проявляется на выходе z на тесте, состоящем из наборов $\{0101, 0110, 1001, 1010\}$.

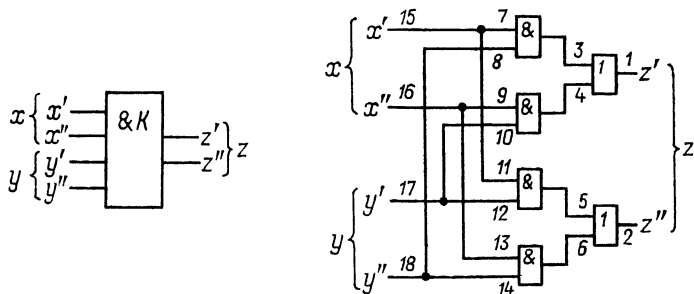


Рис. 5.13

Функции z' и z'' , описывающие схему самопроверяемого элемента И на n входов, составляются по следующим правилам. Каждая функция является дизъюнкцией 2^{n-1} конъюнкций ранга n . Функция z' (z'') содержит все конъюнкции с нечетным (четным) числом переменных с одним штрихом. Например, для двухвходового элемента имеем систему функций (см. рис. 5.13):

$$\begin{aligned} z' &= x'y'' \vee x''y', \\ z'' &= x'y' \vee x''y'', \end{aligned} \quad (5.2)$$

а для трехвходового элемента — систему

$$\begin{aligned} z' &= x'y''v'' \vee x''y'v'' \vee x''y''v' \vee x'y'v', \\ z'' &= x''y''v'' \vee x'y'v'' \vee x'y''v' \vee x''y'v'. \end{aligned} \quad (5.3)$$

Покажем, что элемент, построенный по указанным формулам, является самопроверяемым элементом И. Прежде всего требуется показать, что на всех наборах, состоящих из одних единиц, функции z' и z'' не равны. При этом возможны два случая, когда конъюнкции, соответствующие такому набору, содержат нечетное или четное число переменных с одним штрихом. В первом случае $z' = 1$, $z'' = 0$, а во втором $z' = 0$, $z'' = 1$. На всех остальных входных наборах функции z' и z'' должны быть равны. Здесь возможны также два случая. В первом случае входной набор содержит хотя бы одну переменную, у которой $x_i' = x_i'' = 0$. Поскольку каждая конъюнкция в функциях z' и z'' содержит либо x_i' , либо x_i'' , то $z' = z'' = 0$. Во втором случае входной набор не содержит ни одной переменной, у которой $x_i' = x_i'' = 0$, и содержит хотя бы одну переменную, у которой $x_i' = x_i'' = 1$. Пусть это будет набор $\gamma = \tilde{x}_1' \tilde{x}_1'' \dots \tilde{x}_{i-1}' \tilde{x}_{i-1}'' \times \times 1 \tilde{x}_{i+1}' \tilde{x}_{i+1}'' \dots \tilde{x}_n' \tilde{x}_n''$. Существуют, по крайней мере, два набора α и β , у которых для всех переменных $x_i' \neq x_i''$, и такие, что $\gamma > \alpha$ и $\gamma > \beta$. Например, это наборы $\alpha = 01 \dots 010101 \dots 01$ и $\beta = 01 \dots 011001 \dots 01$. Пусть $z'(\alpha) = 0$ и $z''(\alpha) = 1$. Тогда $z'(\beta) = 1$ и $z''(\beta) = 0$. Отсюда, учитывая, что функции z' и z'' являются монотонными, следует $z'(\gamma) = z''(\gamma) = 1$. Теперь остается показать, что множество наборов α , у которых для всех i $x_i' \neq x_i''$, образует проверяющий тест, обнаруживающий все одиночные неисправности в схемах, реализующих функции z' и z'' . Множество наборов α разбивается на две группы наборов, которые содержат нечетное и четное число переменных $x_i' = 1$. Например, для системы функций (5.3) эти группы образуются множествами $A = \{010110, 011001, 100101, 101010\}$ и $B = \{010101, 011010, 100110, 101001\}$. Множество A содержит все минимально истинные наборы функции z' [56], т. е. такие наборы α , что для любого $\gamma > \alpha$ имеет место $z(\gamma) = 1$. Множество B содержит все максимально ложные наборы функции z' , т. е. такие наборы α , что для любого $\gamma < \alpha$ имеет место $z(\gamma) = 0$. Тест, содержащий все минимально истинные и максимально ложные наборы, обнаруживает все одиночные неисправности монотонной схемы [56]. Итак, схема z' проверяется наборами из множеств A и B . Аналогичный результат имеет место и для схемы z'' , для которой наборы из A есть максимально ложные наборы, а наборы из B — минимально истинные.

В табл. 5.5 и 5.6 поясняется работа схемы двухвходового элемента И, который наиболее часто используется на практике.

В табл. 5.6 столбцы соответствуют входным наборам теста, а строки — номерам линий схемы на рис. 5.13. На пересечении строки и столбца проставлен 1 или 0 в том случае, если отказ данной линии типа «константа 1» или «константа 0» проявляется на данном входном наборе. Это проявление заключается

Таблица 5.5

xy	$\neg x'x''$	$y'y''$	$z'z''$	z
0 0	0 0	0 0	0 0	0
	0 0	1 1	0 0	
	1 1	0 0	0 0	
	1 1	1 1	1 1	
0 1	0 0	0 1	0 0	0
	0 0	1 0	0 0	
	1 1	0 1	1 1	
	1 1	1 0	1 1	
1 0	0 1	0 0	0 0	0
	0 1	1 1	1 1	
	1 0	0 0	0 0	
	1 0	1 1	1 1	
1 1	0 1	0 1	0 1	1
	0 1	1 0	1 0	
	1 0	0 1	1 0	
	1 0	1 0	0 1	

Таблица 5.6

Номер линии	Входной набор			
	0101	0110	1001	1010
1	1	0	0	1
2	0	1	1	0
3	1	—	0	1
4	1	0	—	1
5	—	1	1	0
6	0	1	1	—
7	1	—	0	—
8	—	—	0	1
9	—	0	—	1
10	1	0	—	—
11	—	1	—	0
12	—	—	1	0
13	0	—	1	—
14	0	1	—	—
15	1	1	0	0
16	0	0	1	1
17	1	0	1	0
18	0	1	0	1

в возникновении сигнала 0 (в КПЛ) на выходе z . Нетрудно также показать (построением аналогичной таблицы), что для двухвходовой схемы И тестами являются также четверки наборов, соответствующие значениям $xy = 01$ и 10 , но не является тестом четверка наборов для $xy = 00$.

Многовходовую схему И можно построить также путем каскадного соединения двухвходовых схем. На рис. 5.14 приведена ПСП-схема контроля системы из четырех блоков при условии, что система работоспособна, если исправны все блоки.

Для того чтобы реализовать в КПЛ функцию отрицания, необходимо инвертировать одну из фаз сигнала. Очевидно, что такой элемент НЕ является самопроверяемым. Таким образом, в КПЛ имеется функционально полная система самопроверяемых элементов {И, НЕ}. Однако она не может быть использована для построения любой ПСП-схемы контроля (по произвольно логической формуле) из-за того, что происходит нарушение свойства самопроверяемости. Покажем это на примере реализации функции ИЛИ.

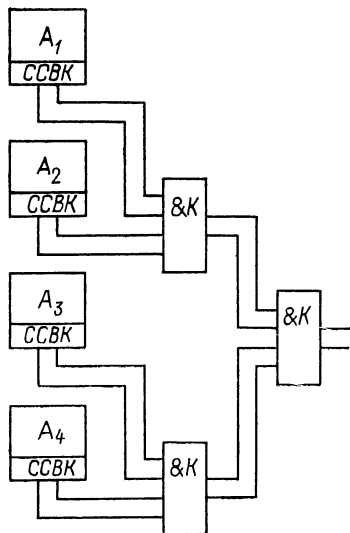


Рис. 5.14

по формуле $x \vee y = \overline{\overline{x} \overline{y}}$ можно построить схему элемента ИЛИ. В логике КПЛ этот элемент выполняет функцию ИЛИ, но не является самопроверяемым. В самом деле, при поступлении на его входы сигналов $xy = 11$, т. е. множества наборов {0101, 0110, 1001, 1010}, из-за наличия входных инверторов на входах элемента И образуется мно-

жество {0000, 0011, 1100, 1111}, которое не является проверяющим тестом для элемента И.

Двухвходовый элемент ИЛИ используется в КПЛ для контроля исправности хотя бы одного блока из двух в дискретной системе. Следовательно, рабочими значениями сигналов на

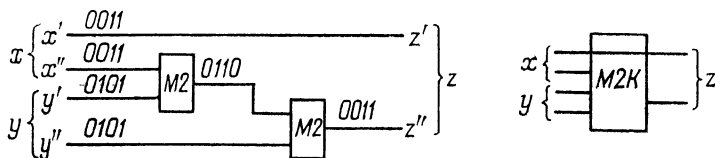


Рис. 5.15

входах его являются значения $xy = 01, 10, 11$. Поэтому логический элемент ИЛИ в КПЛ будем называть самопроверяемым, если каждая его одиночная неисправность проявляется на выходе на каждом из трех тестов {0101, 0110, 1001, 1010}, {0001, 0010, 1101, 1110}, {0100, 0111, 1000, 1011}. Таким образом, за-

дача построения самопроверяемого элемента ИЛИ является более сложной по сравнению с задачей построения элемента И.

Самопроверяемый элемент ИЛИ может быть построен на базе ПСП-элементов И и «исключающее ИЛИ». Схема и условное обозначение последнего показаны на рис. 5.15 [59]. На

Таблица 5.7

xy	$x'x''$	$y'y''$	$z'z''$	z
0 0	0 0	0 0	0 0	0
	0 0	1 1	0 0	
	1 1	0 0	1 1	
	1 1	1 1	1 1	
0 1	0 0	0 1	0 1	1
	0 0	1 0	0 1	
	1 1	0 1	1 0	
	1 1	1 0	1 0	
1 0	0 1	0 0	0 1	1
	0 1	1 1	0 1	
	1 0	0 0	1 0	
	1 0	1 1	1 1	
1 1	0 1	0 1	0 0	0
	0 1	1 0	0 0	
	1 0	0 1	1 1	
	1 0	1 0	1 1	

Таблица 5.8

xy	$x'x''$	$y'y''$	$z'z''$	z
0 0	0 0	0 0	0 0	0
	0 0	1 1	1 1	
	1 1	0 0	0 0	
	1 1	1 1	1 1	
0 1	0 0	0 1	0 1	1
	0 0	1 0	1 0	
	1 1	0 1	0 1	
	1 1	1 0	1 0	
1 0	0 1	0 0	0 1	1
	0 1	1 1	1 0	
	1 0	0 0	0 1	
	1 0	1 1	1 0	
1 1	0 1	0 1	1 0	1
	0 1	1 0	0 1	
	1 0	0 1	1 0	
	1 0	1 0	0 1	

выходе z в КПЛ реализуется функция $z = x \oplus y$ (табл. 5.7). Как известно [43], одиночный проверяющий тест линейной схемы содержит четыре набора, которые обеспечивают на всех входах элементов $M2$ тривиальный тест. На рис. 5.15 показано, что четыре набора, соответствующие коду $xy = 00$, удовлетворяют этому требованию. Легко также убедиться, что проверяющими тестами являются и другие четверки наборов, соответствующие значениям $xy = 01, 10, 11$. Следовательно, данная схема является

самопроверяемой относительно всех четырех входных наборов xy . Это качество позволяет использовать элемент «исключающее ИЛИ» для построения более сложных ПСП-схем контроля.

На рис. 5.16 приведена ПСП-схема контроля четности парафазных сигналов в КПЛ ($z = \overline{a \oplus b \oplus c \oplus d}$). Пусть, например, $abcd = 1001$ и $z = 1$. Тогда на входах элемента 1 присутствуют сигналы $ab = 10$ (имеется в виду поступление четверки наборов 1000, 0100, 1011, 0111), на входах элемента 2 — сигналы 00, на входах элемента 3 — сигналы 11. Пусть при этом произошел,

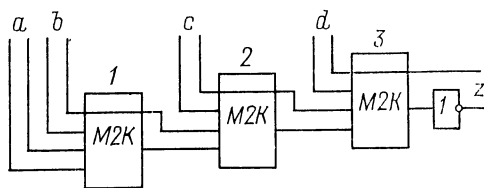


Рис. 5.16

например, отказ элемента 2. Так как четверка наборов на его входах, соответствующая сигналам 00, образует проверяющий тест, то на выходе элемента 2 вместо сигнала 1 появляется сигнал 0. В результате этого сигнал z становится рав-

ным 0, чем и фиксируется данный отказ.

Таким образом, в ПСП-схемах контроля элементы «исключающие ИЛИ» могут соединяться последовательно друг с другом. Учитывая это, двухвходовую ПСП-схему ИЛИ в контрольной парафазной логике строят по известной формуле $x \vee y = xy \oplus y \oplus x$ [29]. На рис. 5.17 изображена схема, построенная по данной формуле, и ее условное обозначение [59]. При этом

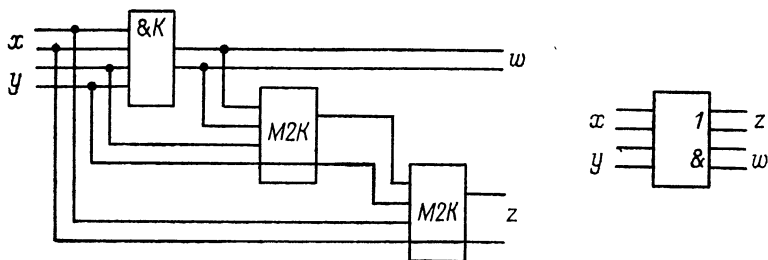


Рис. 5.17

данный элемент имеет еще один дополнительный выход w , на котором реализуется конъюнкция $w = xy$. Работа схемы поясняется табл. 5.8. Пусть при значениях сигналов $xy = 01$ (имеется в виду поступление четверки наборов 0001, 0010, 1101, 1110) и $z = 1$ в схеме (рис. 5.17) происходит отказ типа «константа 1» линии 3 (см. рис. 5.13) у элемента И. На вход элемента И подается четверка наборов, которая является, как указано выше, проверяющим тестом. Поэтому данный отказ обнаруживается на наборе 0001 тем, что на выходе элемента И в схеме (рис.

5.17) появляется сигнал 0. В результате на выходе первого элемента $M2$ появляется сигнал 0, так же как и на выходе z . Этим фиксируется данный отказ. Итак, учитывая то, что для элемента И тестами являются сигналы 01, 10, 11, а для элемента «исключающее ИЛИ» — сигналы 00, 01, 10, 11, схема элемента ИЛИ является самопроверяемой. Для него тестами являются сигналы 01, 10, 11, но сигналы 00 таковыми не являются. По этой причине многовходовую ПСП-схему ИЛИ для контроля исправности хотя бы одного из n блоков нельзя получить путем последовательного соединения схем (рис. 5.17).

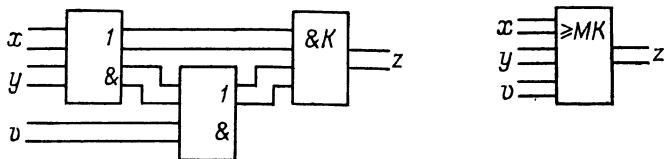


Рис. 5.18

Важную роль при построении схем контроля играет мажоритарный элемент «2 из 3», который реализует в КПЛ мажоритарную функцию $z = xy \vee xv \vee yv$. Он используется для контроля исправности двух блоков из трех. На рис. 5.18 приведена ПСП-схема и условное обозначение мажоритарного элемента, построенная на ПСП-элементах И и ИЛИ. Каскадное включение подобных схем позволяет контролировать исправность $n-1$ блока из n блоков. На рис. 5.19 показана схема контроля ис-

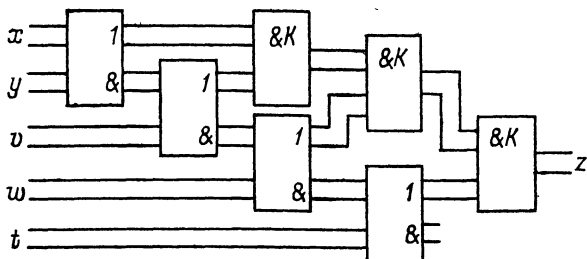


Рис. 5.19

правности 4 блоков из 5, реализующая функцию $z = xyvw \vee \vee xyvt \vee xywt \vee xvwt \vee yvwt$.

С помощью описанных в данном разделе ПСП-элементов И, ИЛИ, НЕ, «исключающее ИЛИ», «2 из 3», работающих в КПЛ, можно строить разнообразные схемы контроля самопроверяемых дискретных систем.

5.5. Компараторы и связь с объектами

При построении схем контроля дискретных систем возникает проблема «последнего сторожа». Она заключается в том, что всегда существует последний контрольный или управляющий элемент, по состоянию которого следует судить об исправности всей системы. Но кто будет контролировать работу этого элемента? Естественный ответ на этот вопрос состоит в том, что последний элемент должен сам себя контролировать, т. е. быть самопроверяемым. В качестве последнего элемента обычно используются компараторы (схемы сравнения), которые можно разделить на три группы: компараторы контроля, управления и включения.

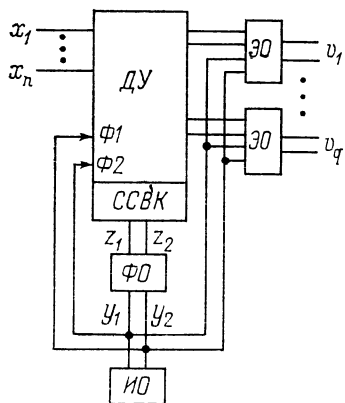


Рис. 5.20

Компараторы контроля служат для сравнения сигналов на выходах самопроверяемых контрольных схем и фиксации факта отказа дискретного устройства. Поэтому их еще называют фиксаторами отказа (ΦO) [60, 69]. Рис. 5.20 поясняет принцип использования ΦO . Дискретное устройство имеет рабочие входы x_1, \dots, x_n , рабочие выходы v_1, \dots, v_q и контрольные выходы z_1 и z_2 , которые формируются ССВК. Нарушение парафазности выходов z_1 и z_2 регистрируется ПСП-схемой

ΦO , играющей роль «последнего сторожа». При нарушении парафазности входов ΦO блокируется в защитном состоянии, при котором нарушается парафазность его выходов и которое не зависит от последующего изменения входов. ΦO включает индикатор отказа системы ($ИО$). После восстановления исправности ДУ фиксатор ошибки переводится по цепи установки в рабочее состояние. В качестве ΦO может быть использован любой ПСП-триггер.

При возникновении отказа и блокировке ΦO возможны три стратегии в поведении ДУ:

- 1) отключение всей системы;
- 2) повторный запуск фиксатора ошибки;
- 3) отключение рабочих выходов.

В первом случае организуется самопроверяемая обратная связь, когда выходы ΦO подключаются к специальным тактовым входам $\Phi 1$ и $\Phi 2$ дискретного устройства (см. рис. 5.20). При нарушении парафазности на входах $\Phi 1$, $\Phi 2$ отключается тактовое питание системы и ДУ переводится в защитное состояние. Для этого может быть использован самопроверяемый тактовый генератор (см. § 5.3), у которого при нарушении пара-

фазности управляющих входов прекращается генерация тактовых импульсов.

Другой способ отключения системы состоит в коммутации цепей питания на контактах специального контрольного реле K , обмотка которого подключается к выходам ΦO через самопроверяемую схему включения реле ($ССВР$). Пример такой схемы приведен на рис. 5.21. Она является компаратором сигналов y_1 и y_2 . Если эти сигналы парафазны, то реле K включено либо по цепи: $+U_k - VT1 - VD1 - K - VD4 - VT4 - \text{земля}$, либо по цепи: $+U_k - VT2 - VD2 - K - VD3 - VT3 - \text{земля}$. Если $y_1 = y_2$, то выключены одновременно либо транзисторы $VT1$ и $VT2$, либо транзисторы $VT3$ и $VT4$. Поэтому обе указанные цепи не существуют и реле K выключено. Все наиболее вероятные

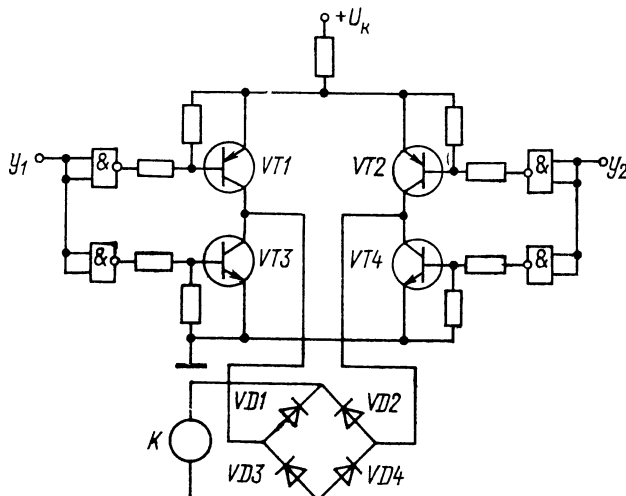
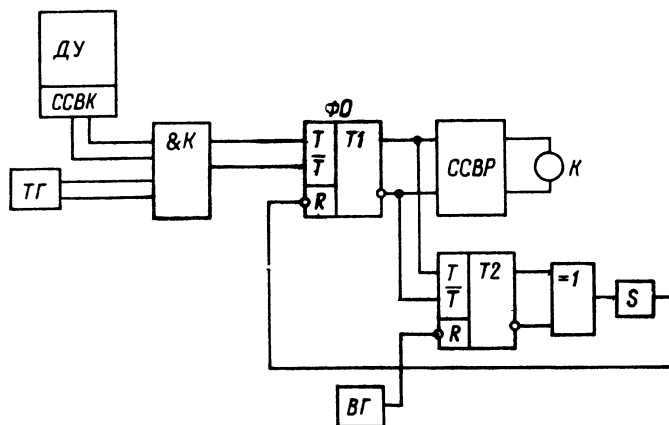


Рис. 5.21

отказы схемы (обрывы и короткие замыкания диодов и транзисторов) приводят к тому, что реле K выключается. Например, при коротком замыкании транзистора $VT2$ обмотка реле K шунтируется по цепи: $+U_k - VT2 - VT4 - \text{земля}$, а при коротком замыкании диода $VD1$ — по цепи: $+U_k - VT2 - VD2 - VD1 - VT3 - \text{земля}$.

Схема (рис. 5.21) является самопроверяемой, но при $y_1 = y_2$ имеет одиночные отказы, приводящие к включению реле (короткое замыкание транзисторов). В этом смысле более надежными являются $ССВР$ с импульсным характером работы. Пример такой схемы приведен на рис. 5.22 [3]. На ее входы y_1, y_2 поступают импульсные парафазные сигналы (переменное напряжение прямоугольной формы), которые управляют ключевыми схемами на транзисторах $VT1, VT2, VT3$. Транзисторы $VT1$ и $VT3$ поочередно подключают источники положительного

процессов заряда конденсаторов и накопления энергии схема выпрямителя с умножением формирует напряжение, достаточное для срабатывания реле K . Если прекращается поступление импульсных парафазных сигналов или происходит отказ любого



элемента схемы, на обмотку реле K будет подаваться напряжение, недостаточное для удержания якоря.

180

в контрольной парафазной логике (см. рис. 5.13). На другой вход элемента И поступают парафазные импульсы от тактового генератора. Как видно из табл. 5.5, если на одном входе элемента И имеется статический парафазный сигнал, то при непрерывной смене парафазного сигнала на другом входе на выходе элемента формируется импульсный парафазный сигнал.

Рис. 5.23 иллюстрирует также вторую стратегию поведения ДУ при отказе — повторный запуск фиксатора ошибки. Эта стратегия применяется для повышения устойчивости дискретной системы относительно сбоев. Если в результате случайного сбоя на выходе *ССВК* нарушается парафазность сигналов, то блокируются триггеры *T1* и *T2*. На выходах *T2* устанавливаются

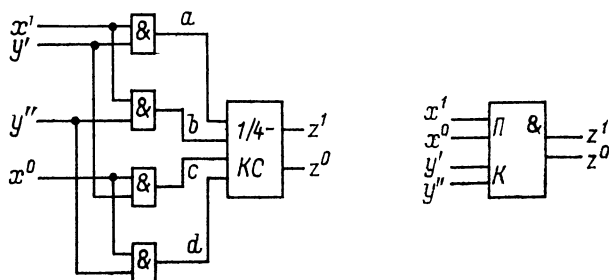


Рис. 5.24

одинаковые сигналы, и происходит запуск одновибратора *S*, который через некоторое время τ вырабатывает сигнал на установку фиксатора ошибок (*ФО*) в рабочее состояние. Время τ меньше, чем время отпущения якоря реле *K*, поэтому в момент установки *ФО* питание системы еще не отключено. Если действие сбоя было кратковременным и парафазность сигналов на выходе *ССВК* восстановилась, то после окончания сигнала установки триггер *T1* остается в рабочем состоянии, продолжает работу *ССВР* и реле *K* снова получает питание. Нормальное функционирование ДУ продолжается. В противном случае после окончания сигнала установки триггер *T1* снова блокируется, реле *K* отпускает якорь и выключает питание системы. Установка триггера *T2* в рабочее состояние происходит уже после завершения описанных процессов, что обеспечивается специальным генератором восстановления (*ВГ*) с низкой частотой работы.

Третья стратегия поведения ДУ при отказах состоит в отключении рабочих выходов. При этом парафазные выходные сигналы ДУ (см. рис. 5.20) транслируются через специальные самопроверяемые элементы отключения (*ЭО*). На рис. 5.24 показана схема *ЭО* и ее условное обозначение. Она имеет парафазный информационный вход *x*, который связан с ДУ (см. рис. 5.20) и работает в обычной парафазной логике, а также

парафазный контрольный вход y , связанный с ΦO и работающий в контрольной парафазной логике. Выход z работает в обычной парафазной логике и повторяет значение x , если на входе y имеется парафазный сигнал (сигнал 1 в КПЛ). Таким образом, при возникновении отказа и блокировке ΦO в защитном состоянии ($y = 0$) ЭО отключает парафазные выходы DY от нагрузки. Элемент содержит самопроверяемую контрольную схему «1 из 4», которая описывается формулами [34]:

$$z_1 = ((a \vee c) \vee (b \vee c)) (a \vee d) (b \vee d),$$

$$z_2 = (a \vee c) (b \vee c) \vee (a \vee d) (b \vee d).$$

Таблица 5.9

xy	x^1x^0	$y'y''$	$abcd$	z^1z^0	z
0 1	0 1	0 1	0 0 0 1	0 1	0
	0 1	1 0	0 0 1 0	0 1	
1 1	1 0	0 1	0 1 0 0	1 0	1
	1 0	1 0	1 0 0 0	1 0	

Таблица 5.10

x^1x^0	$y'y''$	$abcd$	z^1z^0
0 0	0 0	0 0 0 0	0 0
0 0	0 1	0 0 0 0	0 0
0 0	1 0	0 0 0 0	0 0
0 0	1 1	0 0 0 0	0 0
0 1	0 0	0 0 0 0	0 0
0 1	1 1	0 0 1 1	1 1
1 0	0 0	0 0 0 0	0 0
1 0	1 1	1 1 0 0	1 1
1 1	0 0	0 0 0 0	0 0
1 1	0 1	0 1 0 1	1 1
1 1	1 0	1 0 1 0	1 1
1 1	1 1	1 1 1 1	1 1

В табл. 5.9 отражена работа ЭО при рабочих значениях сигналов x и y , когда исправны ДУ, ССВК и ФО. При этом на линиях a, b, c, d присутствует только один сигнал 1 и выход z повторяет вход x . В табл. 5.10 показана работа элемента для всех остальных значений сигналов x^1, x^0, y^1, y^0 , которые возникают при отказах ДУ, ССВК и ФО. При этом выход z всегда имеет защитное значение. Теперь покажем, что данная схема является самопроверяемой. Четыре рабочих набора (табл. 5.9) обеспечивают поступление на входы элементов И полного теста 00, 01, 10, 11. Поэтому все отказы входов элементов И нарушают вес кода «1 из 4» на линиях a, b, c, d хотя бы на одном из рабочих наборов. Это фиксируется нарушением парафазности

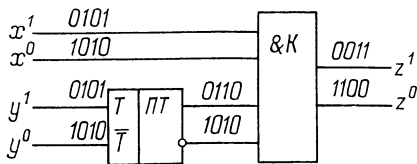
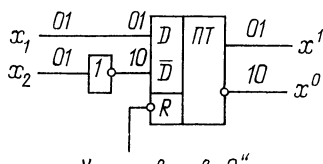


Рис. 5.25



Установка в „0“

Рис. 5.26

на выходе 1/4-КС, и поэтому схема ЭО является самопроверяемой.

Отметим теперь, что все три указанные стратегии могут быть использованы одновременно, что обеспечит высокий уровень безопасности, т. е. высокую вероятность отсутствия неправильного воздействия на объекты управления.

В резервированных системах возникает задача сравнения парафазности двух логических совпадающих парафазных сигналов. Если для этого использовать элемент И (см. рис. 5.13), то на его входы будут поступать только два набора 0101 и 1010, которые не составляют проверяющий тест. Решение задачи дает схема самопроверяемого компаратора, приведенного на рис. 5.25. Поскольку Т-триггер работает с частотой в два раза меньшей, чем частота изменения сигнала y , парафазные сигналы на выходах элемента И «перемешиваются» и образуют проверяющий тест.

Вторую группу компараторов составляют компараторы управления. Они решают более сложную задачу, чем компараторы контроля, так как помимо сравнения одинаковых однофазных или парафазных сигналов должны еще транслировать логическое значение сигнала на выход. Схемы И такую задачу не решают, поскольку на их входы не поступает проверяющий тест. Этим объясняется тот факт, что такие компараторы имеют более сложные схемы. На рис. 5.26 и 5.27 приведены схемы самопроверяемых компараторов соответственно для однофазных и парафазных сигналов. В схеме на рис. 5.26 выходной

парафазный сигнал x повторяет логическое значение двух одинаковых входных однофазных сигналов x_1 и x_2 . Если $x_1 \neq x_2$, то происходит блокировка D -триггера и выход x принимает защитное значение. Для тестирования схемы достаточна двукратная смена сигналов на входе.

В схеме на рис. 5.27 выходной сигнал z повторяет логическое значение двух совпадающих друг с другом входных парафазных переменных x и y . Парафазность на выходе нарушается, если $x \neq y$ или если нарушается парафазность хотя бы одного входного сигнала. Как показано на рис. 5.27, на последовательности из четырех рабочих входных наборов 0101, 1010, 0101, 1010 обеспечивается проверяющий тест на входах всех элементов. Логика работы схемы состоит в следующем. Элементы 1 и 2

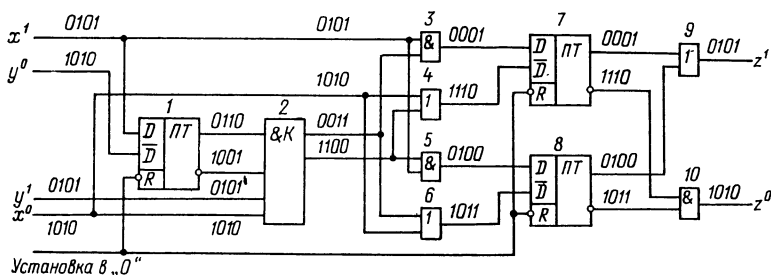


Рис. 5.27

осуществляют «перемешивание» входных наборов, чем обеспечивается поступление проверяющих тестов на входы элементов 3, 4, 5 и 6. Элементы 3, 4, 5, 6, 9, 10 выполняют сравнение (конъюнкцию) парафазных сигналов x и y . Триггеры 7 и 8 фиксируют нарушение парафазности на выходах элементов 3, 4 и 5, 6. Они необходимы потому, что отказы входов этих элементов не транслируются на выходы z^1 и z^0 .

Компараторы включения служат для связи дискретных систем с управляемыми объектами. Они не только осуществляют сравнение и трансляцию логических сигналов, но и включают какие-либо исполнительные устройства (приводы, двигатели, реле, лампы, индикаторы и т. п.). Чаще всего на выход дискретных систем включаются электромагнитные реле, контакты которых коммутируют цепи управляемых объектов. Основное требование, предъявляемое к самопроверяемой схеме включения, состоит в том, чтобы не происходило ложного включения объекта при возникновении неисправностей в схеме. В практических схемах это достигается за счет использования двух основных принципов: динамического характера работы и гальванической развязки входных и выходных сигналов [46]. Для реализации второго принципа используются схемы с индуктивными, емкостными и оптронными связями,

На рис. 5.28 приведена схема, поясняющая принцип работы схемы с индуктивной связью. При поступлении импульсных парафазных сигналов на входы y_1 и y_2 поочередно открываются транзисторные ключи и протекают импульсные токи в первичной обмотке импульсного трансформатора (ИТ). В результате трансформации энергии во вторичную обмотку ИТ происходит заряд конденсатора и включение реле K . Все наиболее вероятные повреждения в схеме приводят к выключению реле. Схема на рис. 5.22 является примером компаратора с емкостной связью.

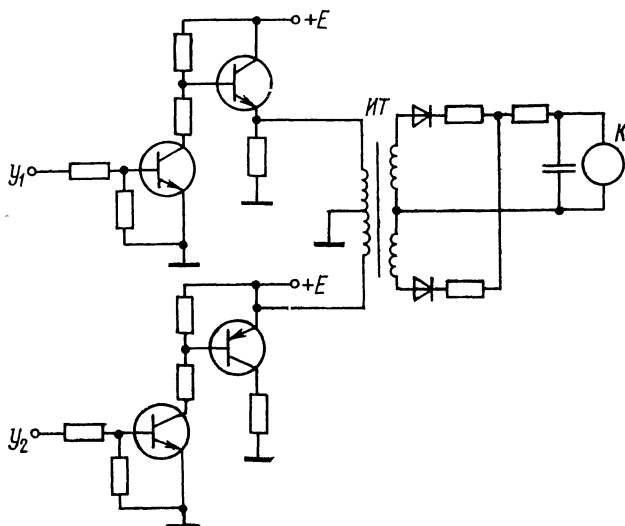


Рис. 5.28

Принцип использования оптронных связей иллюстрирует схема мажоритарного элемента на рис. 5.29 [4]. Реле K включается при синхронном поступлении последовательностей импульсов хотя бы на два входа из трех y_1 , y_2 , y_3 . При этом во время импульсов происходит заряд конденсаторов $C1$, $C2$, $C3$, во время интервалов они разряжаются на светодиоды оптопар $VO1$ и $VO2$ через резисторы $R1$ и $R2$. Напряжение, действующее на светодиоды, равно сумме напряжений на конденсаторе и источнике питания. В результате этого переключаются фототранзисторы оптопар $VO1$, $VO2$ и на входе схемы выпрямителя с преобразованием полярности (конденсаторы $C4$, $C5$ и диоды $VD6$, $VD7$) формируются импульсные сигналы положительной полярности. На конденсаторе $C5$ происходит накопление энергии, необходимое для включения реле K , причем полярность напряжения меняется на противоположную. Одна обмотка поляризованного реле K включена параллельно конденсатору $C5$, а другая подключена к источнику опорного напряжения $+E$.

Реле притягивает якорь при совпадении магнитных потоков в обеих обмотках после поступления нескольких импульсов на вход мажоритарного элемента.

Если импульсные сигналы присутствуют только на одном входе y из трех, напряжение, действующее на светодиоды, недостаточно для переключения оптронов и реле K не включается. То же самое происходит при повреждениях элементов, образующих входные цепи оптронов. Повреждения элементов вы-

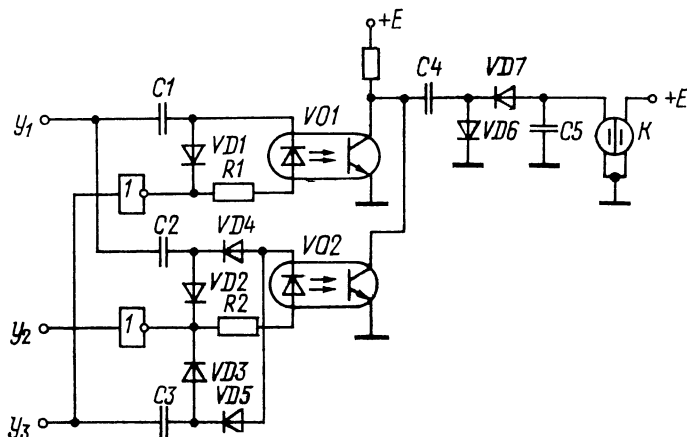


Рис. 5.29

прямителя с преобразованием полярности приводят к тому, что на обмотку реле поступает напряжение, полярность которого противоположна рабочей. При этом магнитные потоки в обмотках направлены встречно и реле K не включается.

5.6. Организация контроля избыточных самопроверяемых систем

Рассмотрим способы применения описанных выше полностью самопроверяемых контрольных устройств для организации контроля наиболее часто используемых на практике избыточных дублированных и троированных структур дискретных систем.

На рис. 5.30 показана последовательная дублированная структура. Она состоит из двух одинаковых, синхронно работающих блоков $ДУ1$ и $ДУ2$. Сигналы на одноименных выходах $ДУ$ сравниваются между собой с помощью элементов $И$. Для контроля исправности элементов $И$ есть два способа. Первый способ состоит в сравнении значений сигналов на выходах элементов $И$ с сигналами на выходах одного (или обоих) $ДУ$ (рис. 5.30). Это сравнение осуществляется с помощью полностью самопроверяемой контрольной схемы $И$ в контрольной парафаз-

ной логике. В случае отказа выходного элемента И сигнал на его выходе не совпадает на одном из тактов работы схемы с сигналом на соответствующем выходе ДУ. При этом на выходе контрольного элемента И нарушается парафазность, срабатывает фиксатор ошибки ΦO и система включается за счет действия самопроверяемой обратной связи.

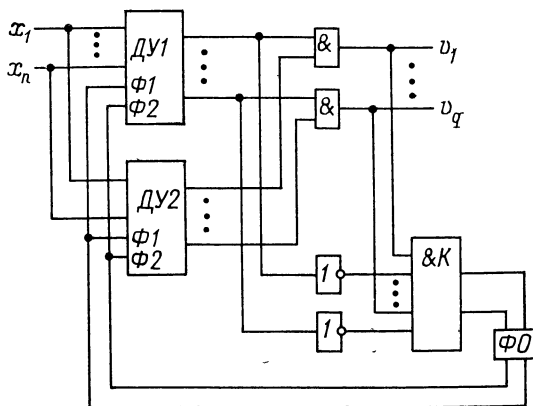


Рис. 5.30

Второй способ состоит в использовании в качестве элементов И самопроверяемых компараторов. На рис. 5.31 и 5.32 показаны такие структуры соответственно для однофазных и парафазных сигналов, в которых использованы ПСП-компараторы, описанные в предыдущем параграфе (на рис. 5.32 через ССТ (схема сравнения триггерная) обозначена схема рис. 5.27). В рассмотренных структурах на рис. 5.30—5.32 сами ДУ не являются ПСП. На рис. 5.33 приведен вариант дублирования структуры с ПСП-блоками, обеспечивающий высокую достоверность работы дискретной системы. В этой структуре ΦO срабатывает, если произойдет отказ ДУ1, или ДУ2, или выходного элемента И.

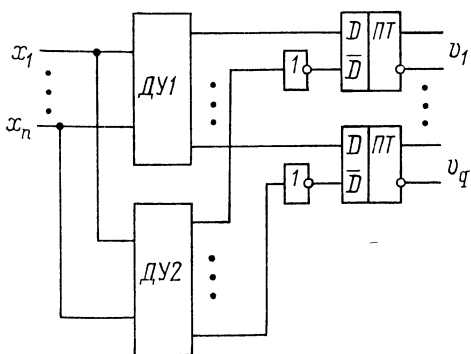


Рис. 5.31

На рис. 5.34 показана параллельная дублированная система [45]. При отказе одного из ДУ, например ДУ1, это фиксирует ССВК1 и на входах элемента М2 устанавливаются одинаковые

логические сигналы. На выходе $M2$ появляется сигнал 0 и триггер $T1$ переключается в состояние 1. На входах элементов И, связанных с инверсным выходом триггера, появляется сигнал 0 и выходы $ДУ1$ отключаются от выходов системы v_1, \dots, v_q . Сигналы v_1, \dots, v_q определяются теперь только сигналами на выходах $ДУ2$. Недостатком данной схемы является то, что в ней не контролируется работа триггеров, элементов $M2$, И и ИЛИ. Например, при отказе триггера $T1$ типа «ложная фиксация в состоянии 0» неисправность $ДУ1$ не будет зафиксирована и выходные сигналы v_1, \dots, v_q будут определяться неправильными выходными сигналами $ДУ1$.

Таблица 5.11

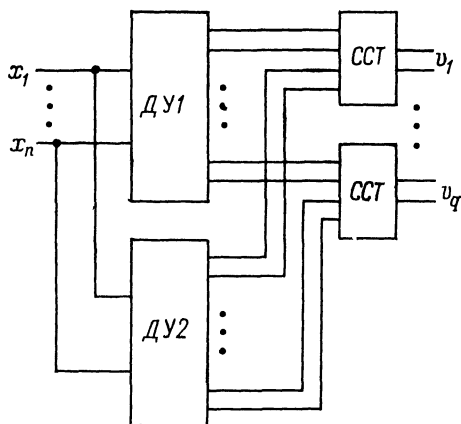


Рис. 5.32

Решение данной проблемы можно получить, используя полностью самопроверяемые контрольные устройства в параллельной дублированной системе с парафазными выходами (рис. 5.35). В ней работа $ДУ$ контролируется с помощью $ССВК$ и $ФО$. Выходы $ФО$ связаны с элементами отключения 1—4 (см. также рис. 5.24). Парафазные сигналы на выходах элементов 1—4 при исправном $ДУ$ повторяют выходные сигналы $ДУ$ и подаются на элементы ИЛИ (элементы 5, 6), которые формируют выходные сигналы системы v_1, \dots, v_q . Элементы ИЛИ работают в обычной парафазной логике, однако отличаются от элементов, описанных в § 4.1. Это отличие заключается в том, что при нарушении парафазности одного из входов сигнал на выходе элемента ИЛИ повторяет парафазный сигнал на другом входе, а при нарушении парафазности обоих входов $z^1z^0 = 00$ (см.

x^1x^0	y^1y^0	z^1z^0
00	00	00
00	01	01
00	10	10
00	11	00
01	00	01
01	01	01
01	10	11
01	11	01
10	00	10
10	01	11
10	10	10
10	11	10
11	00	00
11	01	01
11	10	10
11	11	10

табл. 5.11). Схема такого элемента описывается формулами:

$$z^1 = x^1 \bar{x}^0 \vee y^1 \bar{y}^0, \quad z^0 = \bar{x}^1 x^0 \vee \bar{y}^1 y^0.$$

Так как элементы ИЛИ не являются полностью самопроверяемыми, то для контроля их работы используются элементы И

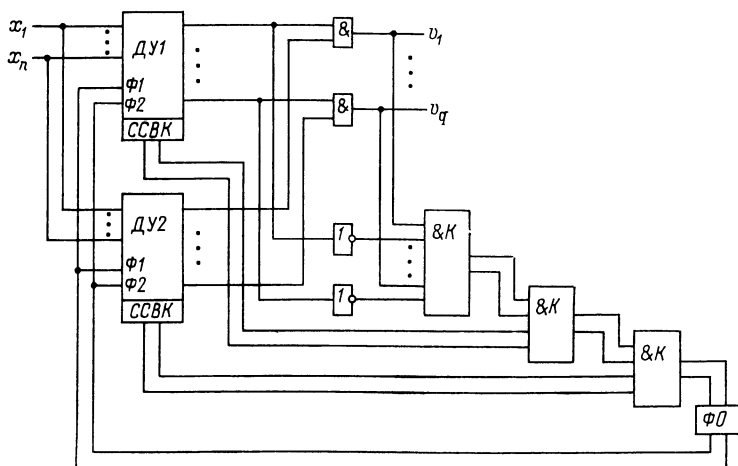


Рис. 5.33

(элементы 7 и 8 на рис. 5.35) и ИЛИ (элемент 9), работающие в контрольной парафазной логике, а также ФОЗ. Элементы И сравнивают прямые значения парафазных выходных сигналов

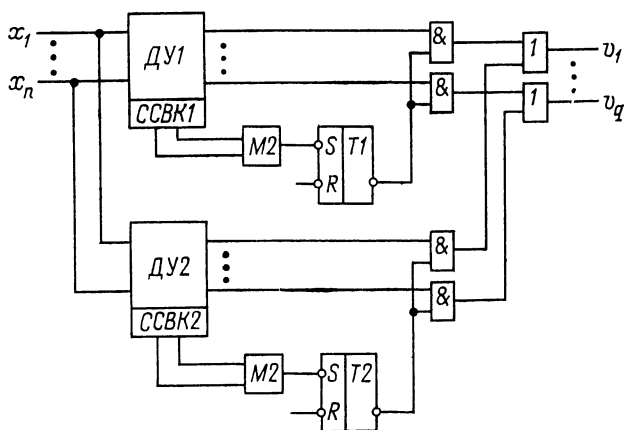


Рис. 5.34

v_1, \dots, v_q с инверсными значениями соответствующих сигналов на выходах элементов отключения.

Рассмотрим работу схемы рис. 5.35 при возможных типах отказов (их четыре). Пусть, например, произошел отказ *ДУ1* (или *ССВК1*, или *ФО1*). При этом *ФО1* блокируется в защитном состоянии, нарушается парафазность сигналов на входах *K* и выходах элементов 1 и 2. Теперь работа схем ИЛИ (элементы 5 и 6) определяется значениями сигналов на выходах элементов 3 и 4. Следовательно, сигналы на выходах системы v_1, \dots, v_q зависят от работы *ДУ2*, а *ДУ1* отключается. Кроме того, нарушается соответствие между сигналами v_1, \dots, v_q и сигналами на выходах элементов 1 и 2, что вызывает наруше-

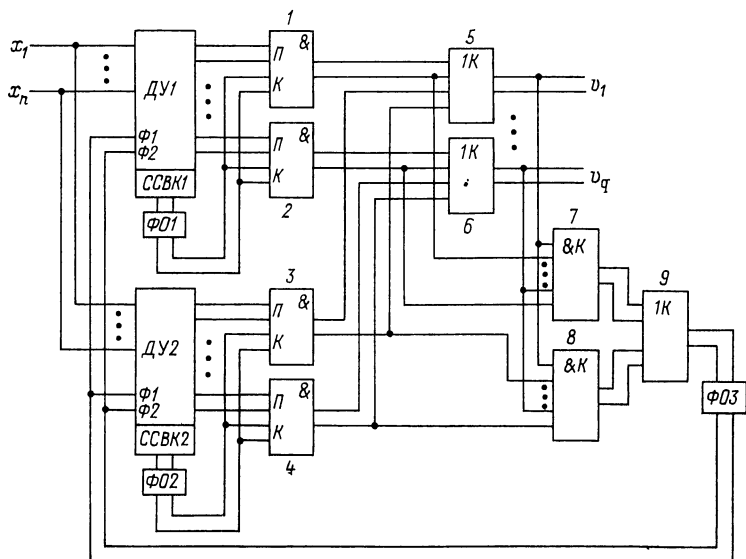


Рис. 5.35

ние парафазности на выходе элемента 7, но не приводит к нарушению парафазности на выходе элемента 9 и блокировке *ФО3*. Рассмотрим второй тип отказа — отказ элементов отключения 1—4. Пусть, например, произошел отказ элемента 1. Это приводит к нарушению парафазности на его выходе и может привести к нарушению парафазности на выходе элемента 7. Однако правильная работа системы продолжается. К третьему типу отказов относятся отказы выходных элементов ИЛИ 5 и 6, что фиксируется одновременно элементами 7 и 8. На выходе элемента 9 нарушается парафазность, и *ФО3* блокируется в защитном состоянии. В результате этого вступает в действие самопроверяемая обратная связь и оба *ДУ* отключаются. Таким образом, отказы элементов 5 и 6 в системе не маскируются, но обнаруживаются. То же происходит и при отказах элементов 7, 8, 9 и *ФО3* (четвертый тип отказа).

На рис. 5.36 показана троированная структура с ПСП-схемой контроля (СК). Она содержит три одинаковых синхронно работающих блока ДУ1, ДУ2 и ДУ3. Их одноименные выходы (для примера на рис. 5.36 каждый блок имеет три выхода) подключаются к мажоритарным элементам «2 из 3». Схема контроля приведена на рис. 5.37. Контроль работы каждого блока осуществляется сравнением инвертированных сигналов на выходах мажоритарного элемента с сигналами на соответствующих выходах ДУ. Для этого применяются схемы И, работающие

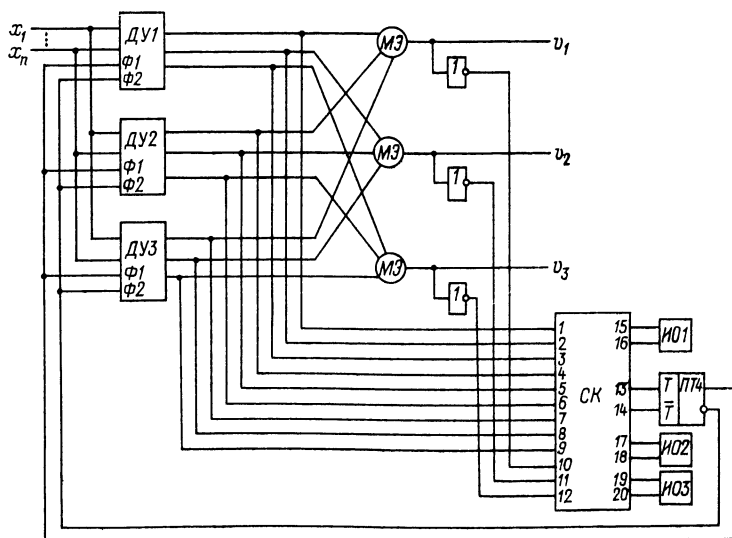


Рис. 5.36

в контрольной парафазной логике (рис. 5.37): элементы 1, 2, 3 и триггер ПТ1 — для контроля ДУ1; элементы 4, 5, 6 и триггер ПТ2 — для контроля ДУ2; элементы 7, 8 и триггер ПТ3 — для контроля ДУ3. К выходам 15, 16, 17, 18 и 19, 20 подключаются индикаторы отказов соответствующих ДУ. Элементы 9, 10, 11 образуют полностью самопроверяемую мажоритарную схему в контрольной парафазной логике (см. рис. 5.18), на входы которой поступают парафазные сигналы контроля исправности трех блоков. Таким образом, парафазность на выходах 13, 14 сохраняется, если исправны хотя бы два блока из трех. При отказе одного ДУ (например, ДУ1) нарушается парафазность на выходе элемента 3, триггер ПТ1 блокируется в защитном состоянии и включается индикатор отказа ИО1. Но парафазность на выходах 13, 14 сохраняется и троированная система продолжает функционировать правильно.

Если теперь откажет еще один, второй блок, то нарушается парафазность на выходах 17, 18 или 19, 20. Это приводит к на-

рушению парафазности на выходах 13, 14 и блокировке в защитном состоянии триггера ПТ4 (см. рис. 5.36). В результате действия самопроверяемой обратной связи все ДУ отключаются. То же происходит и при отказе хотя бы одного мажоритарного элемента, когда парафазность нарушается на всех выходах схемы контроля.

Рассмотрим более подробно работу схем И, контролирующих исправность ДУ. Трудность, которая возникает при сравнении работы одинаковых блоков, состоит в том, что на входах контрольных схем может не обеспечиваться проверяющий тест. Чтобы решить эту проблему, в схему (рис. 5.37) включены элементы 3 и 6. Пусть, например, на выходах ДУ появляется пол-

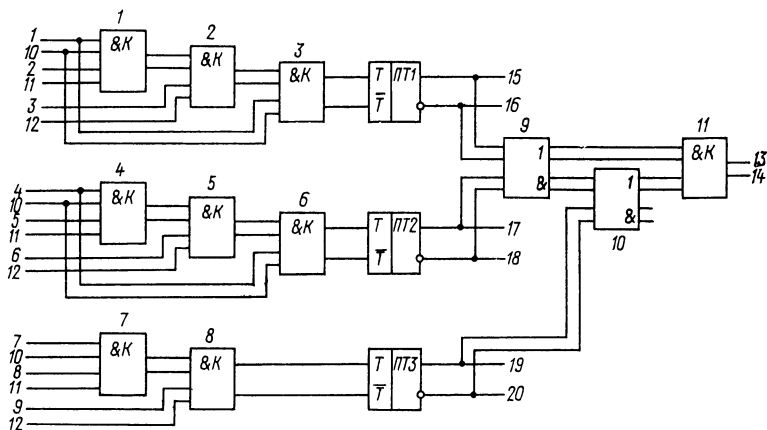


Рис. 5.37

ная последовательность слов 000, 001, 010, 011, 100, 101, 110, 111. Тогда на парафазных входах схем И (1, 10, 2, 11, 3, 12), (4, 10, 5, 11, 6, 12) и (7, 10, 8, 11, 9, 12) появляются одинаковые последовательности сигналов: 010101, 010110, 011001, 011010, 100101, 100110, 101001, 101010. На выходах элементов 2, 5 и 8 появляются также одинаковые последовательности сигналов: 01, 10, 10, 01, 10, 01, 01, 10. Если выходы элементов 2 и 5 подключить непосредственно ко входам триггеров ПТ1 и ПТ2, то все три триггера будут работать одинаково и на их выходах сформируются сигналы: 01, 10, 10, 10, 01, 01, 01, 10. Это не обеспечивает проверяющего теста на входах элементов 9, 10 и 11 мажоритарной схемы. Включение дополнительных элементов И (элементов 3 и 6), входы которых подсоединены к разным внешним входам, позволяет «сместить» сигналы. В данном случае на выходе элемента 3 имеем последовательность 01, 10, 10, 01, 01, 10, 10, 01, а на выходе элемента 6 — последовательность 01, 10, 01, 10, 10, 01, 10, 01. Соответственно на выходах триггеров ПТ1 и ПТ2 имеем последовательности 01, 10, 10, 10, 10, 01, 01, 01 и

01, 10, 10, 01, 01, 0, 1, 10, 10. Это обеспечивает поступление на входы элементов 9, 10 и 11 проверяющих тестов. Соответственно на входах этих элементов имеем последовательности: 0101, 1010, 1010, 1001, 1001, 0101, 0110, 0110; 0101, 0110, 0110, 1010, 1001, 0101, 1001, 1010 и 1010, 0101, 0101, 1001, 1010, 1010, 0110, 0101.

Рассмотрим теперь случай, когда на выходах ДУ не формируются все возможные выходные наборы, т. е. когда выходы не являются независимыми. Пусть, например, ДУ является распределителем. Поэтому его выходы либо отключены, либо только на одном из выходов присутствует сигнал 1. Пусть на выходах ДУ появляется циклическая последовательность сигналов 000, 100, 010, 001. Тогда на входы элементов И в схеме (рис. 5.37) поступает последовательность 010101, 100101, 011001,

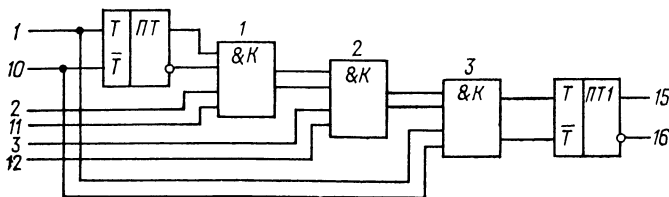


Рис. 5.38

010110, которая не составляет проверяющего теста. В этом случае надо «смешать» входные сигналы, для чего используется самопроверяемый Т-триггер. На рис. 5.38 показан фрагмент схемы контроля для ДУ1. Проверяющий тест на все элементы схемы И поступит после появления на выходах ДУ последовательности: 000, 100, 010, 001, 000, 100, 010, 001. При этом на входах элементов 1, 2, 3 появляются соответственно такие последовательности сигналов: 0101, 1001, 1010, 1001, 1001, 0101, 0110, 0101; 0101, 1001, 0110, 1010, 1001, 0101, 1010, 0110 и 0101, 1010, 1001, 0101, 1001, 0110, 0101, 1001.

ГЛАВА ШЕСТАЯ

САМОПРОВЕРЯЕМЫЕ ПРОГРАММНЫЕ РЕАЛИЗАЦИИ ДИСКРЕТНЫХ УСТРОЙСТВ

6.1. Принципы построения самопроверяемых управляющих программ в микропроцессорных системах

Существует два способа построения дискретных устройств на современных интегральных схемах:

- 1) построение систем управления с «жесткой логикой» на

основе интегральных схем малой, средней и большой степени интеграции (аппаратная реализация);

2) построение систем управления «с гибкой логикой» с применением микропроцессоров и микро-ЭВМ (программная реализация).

Аппаратная реализация была основным способом построения *ДУ* до середины 70-х годов, когда начали внедряться первые микропроцессорные системы. Они выявили основной недостаток аппаратных средств — отсутствие универсальности. Аппаратные *ДУ* — это обычно узко специализированные устройства, у которых отсутствует возможность внесения существенных изменений в алгоритм работы без коренной переделки схемы устройства. Такую возможность предоставляют программные реализации *ДУ*. Функционирование устройства в этом случае определяется программой, хранимой в памяти микропроцессорной системы. Это позволяет путем изменения программы при неизменных технических средствах получить совершенно различные режимы работы устройства и тем самым решать разные логические задачи.

Различают две группы методов программной реализации *ДУ*: компиляционные и интерпретирующие [21, 31]. При компиляционном методе для каждого *ДУ* строится своя программа, при выполнении которой в микро-ЭВМ необходимо ввести набор входных переменных и слово начального состояния (для последовательных *ДУ*). При интерпретирующем методе строится универсальная программа, которая настраивается на реализацию конкретного *ДУ* путем ввода в микро-ЭВМ массива исходных данных. Выбор конкретной программной реализации определяется требованиями к разрабатываемой системе управления с точки зрения ее функциональных возможностей, объема памяти, быстродействия и защищенности от ошибок.

Широкое использование микропроцессорной техники для построения систем управления требует повышения надежности систем, что возможно на основе сочетания методов повышения надежности аппаратуры и программного обеспечения (ПО). Если же для построения системы используются коммерческие микропроцессоры, не имеющие встроенных средств контроля, центр тяжести по обеспечению надежности смещается в область ПО. В данной главе рассматриваются обнаруживающие свойства управляющих программ, реализующих самопроверяемые *ДУ*.

Под надежностью программы понимается ее свойство правильно выполнять заданный алгоритм функционирования в течение определенного промежутка времени. Ошибки программ можно разделить на два класса. Это, во-первых, ошибки, допущенные программистом при написании программы, и во-вторых, ошибки, возникающие при выполнении программы в результате сбоев и отказов аппаратных средств. Последствие

этих ошибок состоит в искажении результатов вычислений и неправильном формировании управляющих сигналов. Однако, если ошибки первого класса могут быть обнаружены при отладке и тестировании программы, то ошибки второго класса обнаруживаются только в процессе вычислений.

Борьба с ошибками первого класса предполагает создание безошибочных программ. При этом используются специальные методы проектирования ПО, автоматизации синтеза программ, доказательства их правильности и тестирования [22, 23]. Для обнаружения ошибок второго класса возможны два подхода. Первый состоит в том, что эта задача возлагается на аппаратные самопроверяемые средства встроенного контроля. При этом избыточность вносится в аппаратуру и возможно построение самопроверяемых микропроцессоров [64]. Во втором случае избыточность вносится в программное обеспечение. Наиболее широко используются методы повторного счета, n -вариантного программирования, методы контрольных точек и контрольных сумм. К этому направлению относятся методы построения самопроверяемых программ [27, 67].

В наиболее общем виде самопроверяемую программу можно определить следующим образом [67].

Определение 6.1. Программа называется самопроверяемой, если и только если для любого входа, удовлетворяющего входной спецификации, программа выдает правильный выходной сигнал или вырабатывает предупреждающее сообщение, информирующее пользователя, что выходной сигнал может быть неправильным.

Для обеспечения свойства самопроверяемости в [67] выбираются точки программы, для которых генерируются «утверждения», характеризующие определенное состояние программы, доказываемая корректность «утверждений» и производится их размещение в выбранных точках программы. Считается, что выходные результаты вычислены правильно для некоторых входных данных, если выполняются все «утверждения» в программе.

Мы рассмотрим эту задачу в более узкой постановке, как задачу построения самопроверяемых управляющих программ в микропроцессорных системах. При этом будем использовать идею обеспечения самопроверяемости аппаратных средств. Она состоит в следующем. Любой алгоритм управления может быть реализован аппаратным или программным путем. Для заданного алгоритма управления найдем описание ДУ (например, систему булевых функций), которое выполняет этот алгоритм, и реализуем это функциональное описание программным путем. При этом будем строить не обычное ДУ, а избыточное, обладающее свойством самопроверяемости. Тогда избыточность программы, которая обеспечит ее способность к обнаружению ошибок, определяется избыточностью функционального описания

ДУ. Данный подход можно отразить диаграммой, показанной на рис. 6.1. Возникает вопрос: в какой степени свойство самопроверяемости ДУ переносится на программную реализацию его функционального описания. Ответ на этот вопрос содержится в последующих разделах данной главы.

Дадим теперь определение самопроверяемой управляющей программы в микропроцессорной системе. Оно отличается от

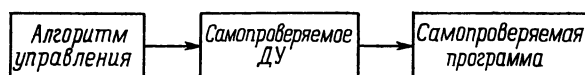


Рис. 6.1

определения самопроверяемой программы произвольного вида (определение 6.1) тем, что указывается конкретный класс одиночных дефектов аппаратных средств, которые должны обнаруживаться. Под одиночной ошибкой программы будем понимать искажение одного бита в слове кода программы или в слове данных. Такая ошибка может возникнуть при дефекте ПЗУ или в процессе выполнения программы в результате сбоя либо неисправности технических средств.

Несмотря на то, что речь идет о дефектах аппаратуры, мы будем говорить об ошибках программы, так как эти дефекты сводятся к искажению объектных кодов программы. Кроме того, считается, что сама программа написана правильно (без ошибок программиста).

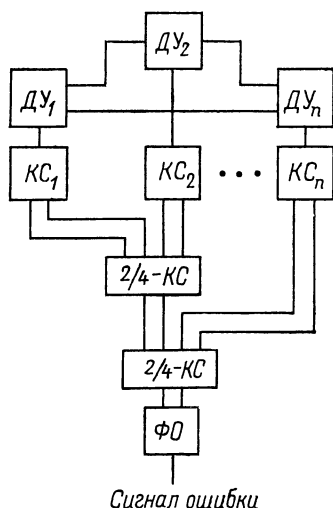


Рис. 6.2

Определение 6.2. Программа называется защищенной от ошибок, если при возникновении любой ошибки из заданного класса на любой рабочей последовательности входных данных выходные данные вычисляются правильно или являются защитными.

Например, при программной реализации самопроверяемых тестеров m/n -СПТ рабочие входные и выход-

ные данные есть соответственно слова кода nSt и значения 01 или 10 выходных функций z_1 и z_2 . Защитными выходными данными являются значения 00 и 11 функций z_1 и z_2 .

Определение 6.3. Программа называется самотестируемой, если для каждой ошибки из заданного класса существует хотя бы одна рабочая последовательность входных данных, на которой вычисляется хотя бы одно защитное значение выходных данных.

Для программы СПТ свойство самотестируемости требует, чтобы для каждой одиночной ошибки существовало слово кода nSt , на котором вычисляются значения 00 или 11 функций z_1 и z_2 .

Определение 6.4. Программа называется самопроверяемой, если она защищена от ошибок и является самотестируемой.

Опыт применения самопроверяемых программ позволяет сформулировать следующие основные принципы построения самопроверяемых программных систем управления.

1. Разделение всей системы на блоки, выполняющие определенные функции (рис. 6.2).

2. Построение каждого блока системы в виде самопроверяемого ДУ, снабженного собственной контрольной схемой.

3. Объединение всех КС в общую контрольную схему путем каскадного соединения самопроверяемых КС. Нарушение парафазности на выходах конечной КС регистрирует фиксатор ошибок (ФО).

4. Программная реализация системы в виде последовательно соединенных самопроверяемых подпрограмм, реализующих функции ДУ и КС (рис. 6.3).

5. Организация циклического повторения программ.

6. Вывод на индикацию состояния выходов контрольных схем подпрограмм ДУ для организации оперативного поиска неисправностей.

Данный подход требует выбора метода программной реализации комбинационных ДУ и ДУ с памятью. Для комбинационных ДУ рассмотрим эту проблему на примере реализации самопроверяемых тестеров для равновесных кодов.

6.2. Программные реализации самопроверяемых тестеров

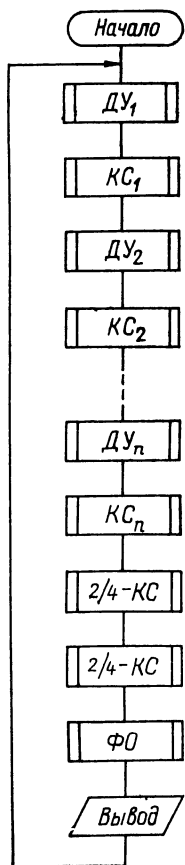


Рис. 6.3

Самопроверяемые тестеры (СПТ) описываются двумя булевыми функциями z_1 и z_2 (см. главу 2). При обработке входных векторов, принадлежащих равновесному коду, и при отсутствии ошибок сигналы z_1 и z_2 принимают значения 01 или 10, т. е. являются парафазными. При нарушении одного из этих условий парафазность на выходе нарушается, выходные сигналы z_1 и z_2 принимают значения 00 или 11, что рассматривается как защитный результат.

Рассмотрим результаты экспериментальных исследований программных реализаций функций z_1 и z_2 , полученных пятью методами: непосредственного вычисления функций, бинарных программ, векторных программ, отображения входного набора, адресных переходов [21, 31]. Первые три метода относятся к компилирующим, а последние два — к интерпретирующим методам. Программы написаны на языке ассемблер. Исследование проводилось на двухпроцессорном комплексе из двух микро-ЭВМ на базе микропроцессора К580. Одна из микро-ЭВМ является ведущей, другая — ведомой. Ведомая микро-ЭВМ используется для выполнения испытываемой программы, а ведущая осуществляет обработку результатов. Программное обеспечение комплекса позволяет определять влияние ошибок, внесенных в байт программы, на результат ее выполнения и исследовать методы повышения защищенности программ от ошибок. Если в результате внесения ошибки происходит заикливание или останов внутри программы, то момент окончания выполнения испытываемой программы определяется с помощью сторожевого таймера. Суть эксперимента заключается в том, что в программе рассеивается p ошибок (если исследуются одиночные ошибки, то $p = 1$) и осуществляется исполнение программы — расчет функций z_1 и z_2 для всех рабочих слов кода nCm .

Результат выполнения программы СПТ анализируется на существование: а) правильного парафазного результата; б) защитного результата (непарафазного или сигнала сторожевого таймера); в) неправильного парафазного результата (когда значения функций z_1 и z_2 инверсны их правильным значениям).

Исследовались программы наиболее распространенных на практике тестеров для $n = 4, 5$ и 6 . Чаще всего в дискретных системах используется тестер 2/4-СПТ. В табл. 6.1 представлены результаты испытаний программ тестера 2/4-СПТ, который задается формулами:

$$z_1 = (x_1 \vee x_2) (x_3 \vee x_4),$$

$$z_2 = x_1 x_2 \vee x_3 x_4.$$

Так как микропроцессор К580 является 8-разрядным, в программах для представления единицы информации отведен один разряд в байте, т. е. логический 0 представлен как $(00)_{16}$, а логическая 1 — как $(01)_{16}$ в шестнадцатичном представлении. Общее число испытаний определяется возможным числом искажений каждого бита в каждом байте программы на каждом из шести слов кода «2 из 4». Например, для столбца 1 в табл. 6.1 имеем $29 \times 8 \times 6 = 1392$.

Степень защищенности программы от одиночной ошибки оценивается числом защитных результатов (должен быть близок к 100 %), а степень самотестируемости — числом обнаруживаемых ошибок (должен быть близок к 0). Ошибка, внесенная в байт программы, является необнаруженной, если пара-

Таблица 6.1

Параметр программы	Метод программной реализации				
	непосредственного вычисления	бинарных программ	векторных программ	отображения входного набора	адресных переходов
Длина программы в байтах	29	72	72	60	21
Общее число испытаний	1392 100 %	3456 100 %	3456 100 %	2880 100 %	1008 100 %
Число правильных парафазных результатов	389 27,95 %	2320 67,13 %	2699 78,10 %	1650 57,29 %	720 71,43 %
Число неправильных парафазных результатов	43 3,09 %	38 1,10 %	33 0,95 %	56 1,95 %	0 0 %
Число защитных результатов	960 68,96 %	1098 31,77 %	724 20,95 %	1174 40,76 %	288 28,57 %
Общее число ошибок	232	576	576	480	168
Число необнаруженных ошибок	6 2,5 %	98 17,01 %	254 44,51	139 28,96 %	80 47,62 %

фазность функций z_1 и z_2 не нарушается ни на одном из шести слов кода «2 из 4».

Из табл. 6.1 следует, что наилучшими свойствами обладает программа, построенная по методу непосредственного вычисления булевых функций. Она имеет наибольшее число защитных результатов (68,96 %) и наименьшее число необнаруживаемых ошибок (2,5 %). И это не случайно. Метод непосредственного вычисления — единственный из указанных пяти методов, который работает со структурой формул для вычисления функций z_1 и z_2 . Для остальных методов эта структура безразлична. Например, метод адресных переходов [31] работает не с формулой, а с таблицей истинности. Таким образом, можно сказать, что все методы, кроме метода непосредственного вычисления, дают «естественные» программные реализации тестеров, в которые не вносятся никакие специальные средства обеспечения самопроверяемости, если не считать парафазного выхода. В то же время метод непосредственного вычисления такое средство имеет. Свойство самопроверяемости схемы тестера полностью определяется структурой формул z_1 и z_2 (изменение структуры формул нарушает это свойство). Поэтому наблюдаемый высокий процент защитных результатов для метода непосредственного вычисления является следствием защищенности системы функций z_1 и z_2 от одиночных и кратных однопавленных отказов букв формул.

Итак, эксперименты показывают, что свойство самопроверяемости системы функций z_1 и z_2 распространяется и на ее

программную реализацию методом непосредственного вычисления. Однако при этом существует определенное количество ошибок, которые не обнаруживаются (2,5 % для 2/4-СПТ). Это объясняется тем, что одиночные ошибки программы не всегда эквивалентны по своим последствиям отказам букв формул z_1 и z_2 , но существенно разнообразнее по своему влиянию на конечный результат вычисления. Это влияние иногда определить достаточно сложно. Например, если в результате ошибки изменяется байтность команды, это приводит к нару-



Рис. 6.4

шению программы. Одиночная ошибка может привести к искажению адресной части, операнда или кода операции команды. Первые два вида искажения приводят к использованию в программе неверных данных, а третий вид — к выполнению другой команды или появлению кода несуществующей команды. Поэтому все множество последствий ошибок можно разделить на две группы: искажение данных и искажение программы. На рис. 6.4 представлены случаи последствий ошибок в программах СПТ.

Анализ причин нарушения свойства самопроверяемости по-

Таблица 6.2

Параметр массива	Метод адрес- ных переходов	Метод отобра- жения входного набора
Длина, байт	16	16
Общее число ошибок	128	128
Общее число испытаний	768 100 %	768 100 %
Число правильных парафазных результатов	720 93,75 %	682 88,80 %
Число неправильных парафазных результатов	0 0 %	0 0 %
Число защитных результатов	48 6,25 %	86 11,20 %
Число необнаруженных ошибок	80 62,50 %	88 68,75 %

казывает, что необнаружение ошибок в большинстве случаев объясняется отсутствием обнаруживающей способности выбранного кода для представления логических сигналов $(00)_{16}$ и $(01)_{16}$ (кодвое расстояние между ними равно 1). При таком кодировании сигналов могут не обнаруживаться проявления ошибок вида 1.1, 1.3, 1.5, 1.6, 2.1, 2.2, 2.3 на рис. 6.4. Кроме того, в интерпретирующих программах имеется массив констант, используемых при их выполнении, который имеет низкую степень защищенности от одиночной ошибки. В табл. 6.2 приведены данные исследования влияния одиночной ошибки в массиве констант.

Рассмотрим способы повышения защищенности программ непосредственного вычисления булевых функций на примере вычисления функций z_1 и z_2 тестера 2/4-СПТ:

$$z_1 = (x_1 \vee x_2) (x_3 \vee x_4), \quad (6.1)$$

$$z_2 = x_1 x_2 \vee x_3 x_4.$$

Запись программы на языке ассемблер приведена в табл. 6.3. Входные переменные x_1, x_2, x_3, x_4 находятся соответственно в ячейках памяти M1, M2, M3, M4. Адрес ячейки M4 имеет имя BASE.

Применим избыточное кодирование входной и выходной информации программы, используя с этой целью все разряды одного байта. Это позволяет обеспечить в максимальном случае кодвое расстояние между двоичными представлениями логи-

Таблица 6.3

Номер команд	Мнемокод команды	Операнды	Комментарий
1	LXI	H, BASE	Адресация M4
2	MOV	A, M	$x_4 \rightarrow A$
3	MOV	C, M	$x_4 \rightarrow C$
4	DCX	H	Адресация M3
5	MOV	B, M	$x_3 \rightarrow B$
6	DCX	H	Адресация M2
7	MOV	D, M	$x_2 \rightarrow D$
8	DCX	H	Адресация M1
9	MOV	E, M	$x_1 \rightarrow E$
10	ANA	B	$x_3 x_4$
11	MOV	M, A	$x_3 x_4 \rightarrow M1$
12	INX	H	Адресация M2
13	MOV	A, C	$x_4 \rightarrow A$
14	ORA	B	$x_3 \vee x_4$
15	MOV	M, A	$x_3 \vee x_4 \rightarrow M2$
16	MOV	A, D	$x_2 \rightarrow A$
17	ANA	E	$x_1 x_2$
18	MOV	C, A	$x_1 x_2 \rightarrow C$
19	MOV	A, D	$x_2 \rightarrow A$
20	ORA	E	$x_1 \vee x_2$
21	ANA	M	$(x_1 \vee x_2) (x_3 \vee x_4) = z_1$
22	OUT	port B ₁	Вывод z_1
23	MOV	A, C	$x_1 x_2 \rightarrow A$
24	DCX	H	Адресация M1
25	ORA	M	$x_1 x_2 \vee x_3 x_4 = z_2$
26	OUT	port B ₂	Вывод z_2
27	HLT	—	—

Таблица 6.4

Параметр программы	КОД2	КОД3	КОД4	КОД5	КОД6
Форма представления входных данных	1 = FF, 0 = 00	1 = AA, 0 = 55	1 = AK, 0 = 5K	1 = AK, 0 = 5K	1 = 8K, 0 = 4K
Форма представления выходных данных	1 = FF, 0 = 00	1 = FF, 0 = 00	1 = FK, 0 = 0K	1 = FK, 0 = 0K	1 = CK, 0 = 0K
Общее число испытаний	1392 100 %	1392 100 %	1392 100 %	1392 100 %	1392 100 %
Число правильных паразных результатов	355 25,50 %	167 12,00 %	40 2,87 %	20 1,44 %	18 1,29 %
Число неправильных паразных результатов	36 2,59 %	5 0,36 %	0 0 %	0 0 %	0 0 %
Число защитных результатов	1001 71,91 %	1220 87,64 %	1352 97,13 %	1372 98,56 %	1374 98,71 %
Общее число ошибок	232	232	232	232	232
Число необнаруженных ошибок	4 1,72 %	2 0,86 %	3 1,29 %	1 0,43 %	1 0,43 %

ческих сигналов 0 и 1, равное восьми. В табл. 6.4 приведены результаты испытаний программы 2/4-СПТ для различных случаев кодирования.

При кодировании логических сигналов 0 и 1 как 00 и FF улучшается защищенность программы (ср. первые столбцы табл. 6.1 и 6.4), так как исключается возможность преобразования логического 0 в логическую 1 и наоборот в результате искажения одного бита. Однако остаются необнаруженными следующие проявления ошибок: 1.5, 1.6, 2.1, 2.2 и 2.3 (см. рис. 6.4).

Кодирование входных сигналов 55 и AA (КОД3) обладает тем же кодовым расстоянием, что и КОД2, но дает существенно лучшие результаты (ср. столбцы КОД2 и КОД3 в табл. 6.4). Это объясняется тем, что формы представления входных и выходных данных теперь различаются. Например, для кодового слова $x_1x_2x_3x_4 = 0011$ ($x_1 = 55$, $x_2 = 55$, $x_3 = AA$, $x_4 = AA$) имеем (см. формулу (6.1)):

$$\begin{aligned}
 z_1 &= (x_1 \vee x_2) (x_3 \vee x_4) = (55 \vee 55) (AA \vee AA) = \\
 &= (01010101 \vee 01010101) (10101010 \vee 10101010) = \\
 &= 01010101 \times 10101010 = 00000000 = (00)_{16}, \\
 z_2 &= x_1x_2 \vee x_3x_4 = 55 \wedge 55 \vee AA \wedge AA =
 \end{aligned}$$

$$\begin{aligned}
 &= (01010101 \wedge 01010101) \vee (10101010 \wedge 10101010) = \\
 &= 01010101 \vee 10101010 = 11111111 = (FF)_{16}.
 \end{aligned}$$

Напомним, что в микропроцессорах осуществляются поразрядные логические операции двоичных слов. Указанное качество кодирования полностью исключает использование входных данных как конечного результата (искажение 2.1 на рис. 6.4) и частично исключает искажения типа 2.2 и 2.3.

Искажения типа 1.5, 1.6, 2.2 и 2.3 возникают в большинстве случаев из-за замены входных переменных друг на друга. Поясним это на примере ошибки в команде № 16 MOV A, D (см. табл. 6.3) — на кодовом слове $x_1x_2x_3x_4 = 0011$. Правильная программа при кодировании переменных кодом КОДЗ в результате приведенных выше вычислений дает результат: $z_1 = (00)_{16}$, $z_2 = (FF)_{16}$. Это правильный парафазный результат. Команда № 16 имеет код 01111010. Пусть произошло искажение младшего разряда этого кода и возникает код команды 01111011, которая имеет смысл MOV A, E. В результате выполнения искаженной команды № 16 в аккумулятор вместо переменной x_2 , которая хранится в регистре D, засылается переменная x_1 , которая хранится в регистре E, и программа ошибочно вычисляет функции $z_1' = (x_1 \vee x_1) (x_3 \vee x_4)$, $z_2' = x_1x_2 \vee x_3x_4$. Но поскольку на наборе 0011 переменные x_1 и x_2 равны, указанная ошибка не искажает результата вычисления функций z_1 и z_2 и не обнаруживается.

Для борьбы с этим явлением надо использовать различающее кодирование входных переменных, которое будем называть ассоциативным кодированием (по аналогии с термином «ассоциативная память»). Применим разделимый (i, k) -код, где i — число информационных разрядов, k — число ассоциативных разрядов. В дальнейшем принято $i = k = 4$. Информационные разряды (первый полубайт) содержат логическое значение переменной, а ассоциативные разряды (второй полубайт) кодируют индекс входной или выходной переменной. В табл. 6.4 в столбцах КОД4, КОД5, КОД6 приведены результаты испытания программы для трех вариантов ассоциативного кодирования.

Обозначим через K_j шестнадцатиричные значения ассоциативных полубайтов в коде КОД j . Например, запись $K_4(x_1) = 3$ означает, что в коде КОД4 значения ассоциативных разрядов у входной переменной x_1 равны 0011. В табл. 6.4 имеем: $K_4(x_1) = 3$, $K_4(x_2) = 5$, $K_4(x_3) = 9$, $K_4(x_4) = C$, $K_4(z_1) = 5$, $K_4(z_2) = 9$, $K_5(x_1) = 5$, $K_5(x_2) = 9$, $K_5(x_3) = 6$, $K_5(x_4) = A$, $K_5(z_1) = C$, $K_5(z_2) = 3$, $K_6(x_1) = 5$, $K_6(x_2) = 9$, $K_6(x_3) = 6$, $K_6(x_4) = A$, $K_6(z_1) = C$, $K_6(z_2) = 3$.

Покажем на примере кода КОД5 его возможности по обнаружению ошибок. Поскольку всегда выполняются одни и те же действия над ассоциативными частями байтов в соответствии с формулами (6.1), у промежуточных и конечных результатов

вычислений значения ассоциативных разрядов будут одинаковыми независимо от значений переменных x_1, x_2, x_3 и x_4 . Для кода КОД5 имеем: $K_5(x_1x_2) = 0101 \wedge 1001 = 0001$, $K_5(x_1 \vee x_2) = 0101 \vee 1001 = 1101$, $K_5(x_3x_4) = 0110 \wedge 1010 = 0010$, $K_5(x_3 \vee x_4) = 0110 \vee 1010 = 1110$, $K_5(z_1) = K_5(x_1 \vee x_2) \wedge K_5(x_3 \vee x_4) = 1101 \wedge 1110 = 1100$, $K_5(z_2) = K_5(x_1x_2) \vee K_5(x_3x_4) = 0001 \vee 0010 = 0011$. Таким образом, ассоциативные полубайты входных, промежуточных и выходных переменных (на всех этапах вычислений) различны. Это обеспечивает обнаружение ошибок типа 1.5, 1.6, 2.1, 2.2 и 2.3 по рис. 6.4.

Рассмотрим работу программы 2/4-СПТ на входном слове $x_1x_2x_3x_4 = 0011$ для кода КОД5 при отсутствии и наличии указанной выше ошибки в команде № 16. При отсутствии ошибки $z_1 = (x_1 \vee x_2)(x_3 \vee x_4) = (01010101 \vee 01011001) \wedge (10100110 \vee 10101010) = 01011101 \wedge 10101110 = 00001100$, $z_2 = x_1x_2 \vee x_3x_4 = 01010101 \wedge 01011001 \vee 10100110 \wedge 10101010 = 01010001 \vee 10100010 = 11110011$. Имеем правильный парафазный результат, так как значения ассоциативных разрядов 1100 и 0011 у переменных z_1 и z_2 совпадают с их истинными значениями. При наличии ошибки $z_1' = (x_1 \vee x_1)(x_3 \vee x_4) = (01010101 \vee 01010101)(10100110 \vee 10101010) = 01010101 \wedge 10101110 = 00000100$, $z_2' = x_1x_1 \vee x_3x_4 = 01010101 \wedge 01010101 \vee 10100110 \wedge 10101010 = 01010101 \vee 10100010 = 11110111$. Так как значения ассоциативных разрядов 0100 и 0111 отличаются от истинных значений 1100 и 0011, то ошибка обнаруживается.

Экспериментальные исследования программ показывают, что представление входной и выходной информации в ассоциативном коде позволяет обнаружить все виды проявлений одиночных ошибок, указанных на рис. 6.4. При этом достигается почти стопроцентная самопроверяемость программ (количество обнаруживаемых ошибок не превышает одного процента). В качестве еще одного примера в табл. 6.5 приведены результаты испытания программы 2/6-СПТ, которая имеет 0,54 % необнаруживаемых ошибок. Самопроверяемость программ СПТ в определенном смысле оказывается выше, чем аналогичное свойство у аппаратной реализации СПТ, поскольку большинство одиночных ошибок обнаруживается на всех словах кода n Ст, что не может быть достигнуто в аппаратной реализации. Ошибки, которые не обнаруживаются (например, в программе 2/4-СПТ это одна ошибка, а в программе 3/6-СПТ — три ошибки), не нарушают процесса вычислений и не приводят к последствиям, указанным на рис. 6.4.

Ошибка в программе 2/4-СПТ (табл. 6.4), которая не обнаруживается, есть ошибка в команде № 3. При этом происходит замена команды MOV C, M (код 01001110) на команду MOV C, A (код 01001111). Ошибка не обнаруживается, так как в момент выполнения команды MOV C, M в аккумуляторе и в ячейке памяти M находится один и тот же операнд. Ошибки подобного

Таблица 6.5

Параметр программы	КОД1	КОД2	КОД3	КОД4	КОД5	КОД6
Форма представления входных данных	1 = 01, 0 = 00	1 = FF, 0 = 00	1 = AA, 0 = 55	1 = FK, 0 = 0K	1 = AK, 0 = 5K	1 = 8K, 0 = 4K
Форма представления выходных данных	1 = 01, 0 = 00	1 = FF, 0 = 00	1 = AA, 0 = 55	1 = FK, 0 = 0K	1 = AK, 0 = 5K	1 = 8K, 0 = 4K
Общее число испытаний	11200 100 %	11200 100 %	11200 100 %	11200 100 %	11200 100 %	11200 100 %
Число правильных паразных результатов	5269 47,04 %	4648 41,50 %	3115 27,81 %	1643 14,67 %	1212 10,82 %	1365 12,19 %
Число неправильных паразных результатов	150 1,34 %	119 1,06 %	8 0,07 %	0 0 %	0 0 %	0 0 %
Число защитных результатов	5781 51,62 %	6433 57,44 %	8077 72,12 %	9557 85,33 %	9988 89,19 %	9835 87,81 %
Общее число ошибок	560	560	560	560	560	560
Число необнаруженных ошибок	12 2,14 %	4 0,71 %	5 0,89 %	3 0,54 %	3 0,54 %	4 0,71 %

типа исключаются двумя способами: изменением последовательности команд или предварительной загрузкой некоторых регистров константами, отличными от констант, используемых в программе. В данном случае достаточно заменить команды 2 и 3 в табл. 6.3 на последовательность двух команд MOV C, M и MOV A, C.

Из результатов данного параграфа следует процедура построения самопроверяемых программ вычислений булевых функций для микропроцессорных систем. Записывается программа на языке ассемблер, реализующая заданные функции методом непосредственного вычисления. Входные и выходные переменные кодируются ассоциативным кодом. Программа отлаживается и тестируется. Затем производится испытание программы и необнаруживаемые ошибки (их число невелико) устраняются указанными выше способами.

6.3. Модульное построение самопроверяемых программ

Рассмотрим теперь подход к построению самопроверяемых программ, который не требует их испытаний. Идея подхода состоит в том, что программа строится из типовых модулей. Самопроверяемость модулей доказана заранее экспериментальным путем. Для исключения размножения одиночной ошибки

при нарушении байтности команды и перехода ошибки из модуля в модуль используется специальная группа команд локализации ошибки (ГКЛО). *

Основными требованиями к модулям являются универсальность, завершенность и изолированность. Для программной реализации любого ДУ достаточно иметь три программных модуля, реализующих функции И, ИЛИ, НЕ. В состав модуля входят команды, выполняющие элементарные операции ANA r, ORA r, CMA) и команды пересылки операндов и результата (MOV A, r; MOV r, A). Основным требованием к командам модуля является отсутствие нарушения байтности команды в результате искажения одного байта. Для адресации операндов используется метод непосредственной адресации, что определяется «разбросанностью» операндов в ОЗУ.

Таблица 6 6

Команда	Регистры общего назначения r					
	B	C	D	E	H	L
MOV r, A						*
MOV A, r			*		*	*
ANA r		*	*		*	
ORA r		*	*		*	

В табл. 6.6 приведены результаты исследования основных команд модулей. В ней отмечены команды, изменяющие свою байтность при искажении одного бита. По результатам таблицы можно выделить команды для использования их в модулях. Это логические операции ANA B, ANA E, ORA B, ORA E и команды пересылок MOV B, A; MOV E, A; MOV A, B; MOV A, E. Регистр C будем использовать как регистр-индикатор в ГКЛО.

В табл. 6.7 приведен типовой модуль, реализующий функцию $x_i \vee x_j$. Операнды x_i и x_j находятся в ячейках памяти M_i и M_j , а результат операции посылается в ячейку памяти M_k . Аналогичным образом строятся модули для функций И и НЕ. Среди команд модуля (табл. 6.7) только команды LDA и STA изменяют свою байтность при одиночной ошибке. Для обнаружения и локализации таких трансформаций команд используется группа команд ГКЛО, которые работают с регистром-индикатором C (в данном случае это команды DCR C).

К командам ГКЛО предъявляются следующие требования:

1) при отсутствии ошибок в программе ГКЛО не должна нарушать правильного функционирования (т. е. должна быть как бы пустой командой);

* Метод разработан совместно с канд. техн. наук А. В. Харитоновым.

Таблица 6.7

Номер команды	Мнемокод команды	Операнды	Комментарий
1	LDA	M_i	$x_i \rightarrow A$
2	DCR	C	—
3	DCR	C	—
4	MOV	B, A	$x_i \rightarrow B$
5	LDA	M_j	$x_j \rightarrow A$
6	DCR	C	—
7	DCR	C	—
8	ORA	B	$x_i \vee x_j$
9	STA	M_k	$x_i \vee x_j \rightarrow M_k$
10	DCR	C	—
11	DCR	C	—

2) при наличии ошибки в программе ГКЛО должна: а) локализовать ошибку, т. е. исключить трансформацию следующей команды и переход ошибки в следующий модуль; б) создавать условия для обнаружения ошибки — передать управление программе обработки ошибки ERR или привести к защитному результату программы;

3) одиночная ошибка в одной из команд ГКЛО не должна приводить к увеличению байтности ГКЛО, т. е. суммарная байтность трансформированных команд ГКЛО не должна превышать суммарную байтность исходной группы команд ГКЛО.

Чтобы выполнить первое требование, будем использовать в качестве ГКЛО команды INR г или DCR г. Так как эти команды располагаются в определенных точках программы модулей (см. табл. 6.7), то известно число их использования. Это число заносится в регистр г в начале выполнения программы с помощью команды MVI. Если в конце исполнения программы значение регистра г не равно эталонному, то в программе были искажения, изменившие ее структуру. Для локализации ошибки (требование 2а) после команд LDA и STA используются две команды ГКЛО DCR, что исключает искажение следующей команды в наихудшем случае, когда команда LDA или STA трансформируется в результате ошибки в трехбайтную команду.

При возникновении ошибок в модульной программе возмож-

ны три случая. Если при искажении исходной команды ORA, ANA, LDA, STA не нарушается ее байтность, то искажается процесс вычисления и вследствие самопроверяемости функционального описания булевой функции программа выдает защитный результат. Если в результате трансформации исходной команды LDA, STA образуется двух- или трехбайтная команда, использующая одну или две команды ГКЛО и не являющаяся командой передачи управления, то результат вычисления может оказаться незащитным. При этом, так как содержимое регистрандикатора г после выполнения программы не будет равно эталонному, то коррекция результата вычисления командой XRA г приведет к защитному результату. В третьем случае в результате трансформации команд LDA и STA образуется команда передачи управления JMP, CALL. Для выполнения требования 26 выделяется область памяти ОЗУ, заполняемая кодами операции RSTn. Программа обслуживания RSTn осуществляет переход к программе обработки ошибки ERR. Размеры области памяти определяются кодами операций команд ГКЛО. Если код операции первой команды КОП1, а второй — КОП2, то выделяется область ОЗУ с адресами (КОП1)00 — (КОП2)FF. Чтобы выполнить третье требование, в качестве команд ГКЛО надо выбирать команды, искажение которых в одном бите не увеличивает байтности команды. Например, команда DCRC при искажении разрядов трансформируется

Таблица 6.8

Параметр программы	КОД 1	КОД 2	КОД 3	КОД 4	КОД 5	КОД 6
Форма представления входных переменных	1=01, 0=00	1=FF, 0=00	1=AA, 0=55	1=AK, 0=5K	1=AK, 0=5K	1=8K, 0=4K
Форма представления выходных переменных	1=01, 0=00	1=FF, 0=00	1=FF, 0=00	1=FK, 0=0K	1=FK, 0=0K	1=CK, 0=0K
Общее число испытания	3408	3408	3408	3408	3408	3408
Число правильных паразных результатов	938 27,52 %	625 18,34 %	238 6,98 %	55 1,61 %	21 0,62 %	32 0,94 %
Число неправильных паразных результатов	7 0,20 %	0 0 %	0 0 %	0 0 %	0 0 %	0 0 %
Число защитных результатов	2463 72,28 %	2783 81,66 %	3170 93,02 %	3353 98,39 %	3387 99,38 %	3376 99,06 %
Общее число ошибок	568	568	568	568	568	568
Число необнаруженных ошибок	11 0,53 %	0 0 %	0 0 %	0 0 %	0 0 %	0 0 %

в команды INRC; RRC; DADB; DCRB; DCRE; DCRL; MOV C, L; ADCL.

В табл. 6.8 и 6.9 приведены результаты испытаний модульных программ 2/4- и 3/6-СПТ соответственно. Они имеют высокую защищенность. Достоинством модульного подхода является

Таблица 6.9

Параметр программы	КОД-1	КОД 2	КОД 3	КОД 4	КОД 5	КОД 6
Форма представления входных переменных	1=01, 0=00	1=FF, 0=00	1=AA, 0=55	1=FK, 0=0K	1=AK, 0=5K	1=8K, 0=4K
Форма представления выходных переменных	1=01, 0=00	1=FF, 0=00	1=AA, 0=55	1=FK, 0=0K	1=AK, 0=5K	1=8K, 0=4K
Общее число испытаний	28000 100 %	28000 100 %	28000 100 %	28000 100 %	28000 100 %	28000 100 %
Число правильных паразных результатов	11222 40,08 %	9678 34,56 %	6038 21,56 %	4654 16,62 %	2896 10,34 %	1415 5,05 %
Число неправильных паразных результатов	40 0,14 %	0 0 %	0 0 %	0 0 %	0 0 %	0 0 %
Число защитных результатов	16738 59,78 %	18322 65,44 %	21962 78,44 %	22226 79,38 %	25104 89,66 %	26585 94,95 %
Общее число ошибок	1400	1400	1400	1400	1400	1400
Число необнаруженных ошибок	11 0,79 %	2 0,14 %	0 0 %	0 0 %	0 0 %	0 0 %

ся также то, что он позволяет легко автоматизировать сам процесс написания программы вычисления булевой функции на базе самопроверяемых модулей.

6.4. Интерпретирующая программа вычисления булевых функций

Метод непосредственного вычисления, который исследовался в предыдущих параграфах, является компиляционным. Это значит, что для каждой булевой функции должна составляться собственная программа. Например, если в системе программ в ее различных контрольных точках требуется вычислять различные СПТ, необходимо составлять и отлаживать различные программы. Но во многих случаях удобнее иметь единую интерпретирующую программу СПТ, к которой можно было бы обращаться как к подпрограмме. Интерпретирующие методы, исследованные ранее, не учитывают структуру логических фор-

мул и дают низкую защищенность от ошибок. В данном разделе рассмотрим программу вычисления булевых функций, которая является интерпретирующей и в то же время вычисляет функцию непосредственно по ее логической формуле, что обеспечивает высокую защищенность от ошибок.

Будем использовать при вычислениях представление булевой функции в виде обратной бесскобочной записи формул [10], предложенной польским математиком А. Лукасевичем. В ней операторные символы всегда следуют за символами переменных своих операндов. Это делает скобки излишними. Операции выполняются последовательно, в порядке просмотра символов. Для временного хранения входных, промежуточных и выходных переменных используется стек.

Приведем, например, обратную бесскобочную запись формул, описывающих тестер 1/6-СПТ:

$$\begin{aligned} z_1 &= (x_1 \vee x_2 \vee x_5) (x_1 \vee x_3 \vee x_6) \vee (x_3 \vee x_4 \vee x_5) (x_2 \vee x_4 \vee x_6), \\ z_2 &= (x_1 \vee x_2 \vee x_3 \vee x_5 \vee x_6) (x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6). \end{aligned} \quad (6.2)$$

Одна и та же функция может иметь несколько разновидностей записи. Так, ассоциативную цепочку $x_1 \vee x_2 \vee x_3$ можно записать двумя способами: $x_1 x_2 x_3 \vee \vee$ и $x_1 x_2 \vee x_3 \vee$. В последнем случае требуется меньшая глубина стека. Поэтому операторные символы надо располагать сразу за своими операндами. Тогда система (6.2) записывается следующим образом:

$$\begin{aligned} z_1 &= x_1 x_2 \vee x_5 \vee x_1 x_3 \vee x_6 \vee \wedge x_3 x_4 \vee x_5 \vee x_2 x_4 \vee x_6 \wedge \vee, \\ z_2 &= x_1 x_2 \vee x_3 \vee x_5 \vee x_6 \vee x_2 x_3 \vee x_4 \vee x_5 \vee x_6 \vee \wedge. \end{aligned}$$

Данная запись является несвязной, так как отдельные функции в системе вычисляются независимо. Однако во многих случаях в разных формулах встречаются повторяющиеся фрагменты. Несвязная программная реализация в этом случае будет избыточна, что приведет к увеличению массива описания системы функций и времени вычисления. Например, особенностью функционального описания СПТ является наличие в функциях z_1 и z_2 одинаковых фрагментов. В системе (6.2) такими фрагментами являются выражения $x_1 \vee x_2 \vee x_5$, $x_1 \vee x_3 \vee x_6$, $x_3 \vee x_4 \vee x_5$, $x_2 \vee x_4 \vee x_6$ (см. каталог m/n -СПТ в работе [34]). Свойство самопроверяемости при аппаратной реализации тестера требует, чтобы эти выражения в схеме тестера (на элементах И, ИЛИ, НЕ) были реализованы на одном элементе (в данном случае на элементах ИЛИ). Другими словами, логические схемы, вычисляющие функции z_1 и z_2 , должны иметь определенную связную реализацию.

Перенесем это свойство на программную реализацию системы булевых функций. При этом одинаковые фрагменты формул в системе вычисляются один раз, а при вычислении последую-

ших формул фрагменты заменяются их значением, вычисленным ранее. Назовем промежуточной переменной результат вычисления некоторого фрагмента. В общем случае порядок вычисления и подстановки промежуточных переменных может

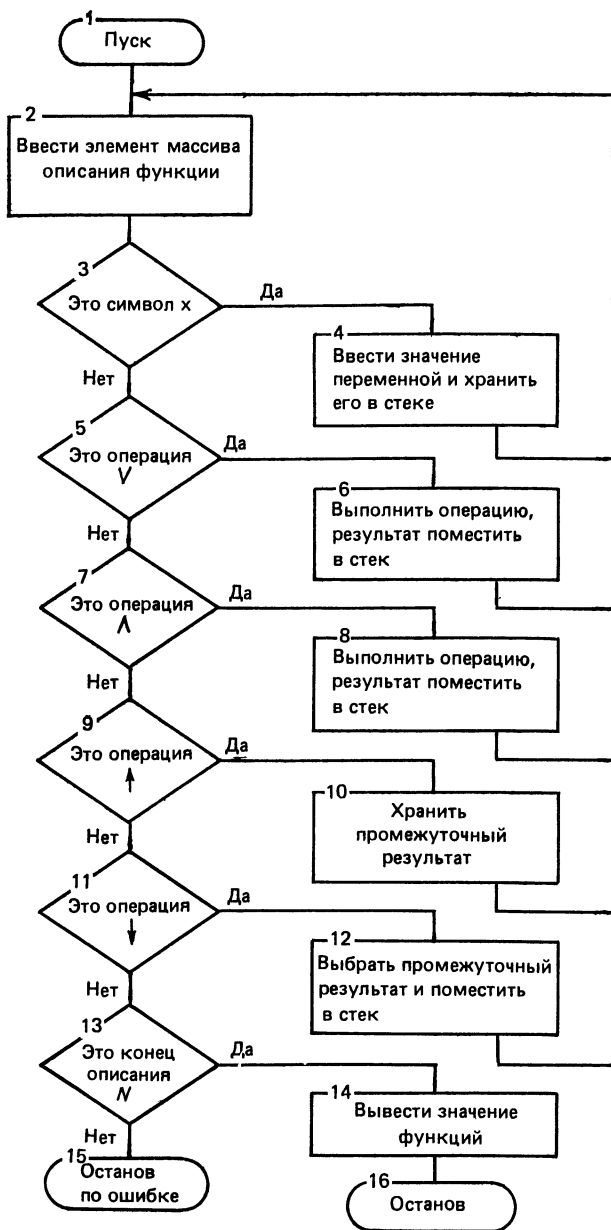


Рис. 6.5

быть произвольным, поэтому часто не удается организовать хранение промежуточных переменных в стеке. Для их хранения необходимо организовать отдельный массив с произвольной адресной выборкой. При этом в обратную бесскобочную запись формул дополнительно вводятся служебные символы $\uparrow Y_j$ и $\downarrow Y_j$, означающие соответственно запись и чтение промежуточной переменной. При этом массив описания системы (6.2) имеет вид:

$$\begin{aligned} x_1 x_2 \vee x_5 \vee \uparrow Y_1 x_1 x_3 \vee x_6 \vee \uparrow Y_2 \wedge x_3 x_4 \vee x_5 \vee \uparrow Y_3 \\ x_2 x_4 \vee x_6 \vee \uparrow Y_4 \wedge \vee \downarrow Y_1 \downarrow Y_2 \vee \downarrow Y_3 \downarrow Y_4 \vee \wedge N. \end{aligned} \quad (6.3)$$

Здесь N есть символ окончания массива.

Подобные записи могут быть легко составлены для любого СПТ из каталога [34], в котором указываются функции для вычисления промежуточных переменных.

Интерпретирующая программная реализация состоит из двух частей: массива описания конкретной системы булевых функций (6.3) и универсальной программы, которая читает символы из массива описания в порядке их расположения, идентифицирует их и сразу выполняет действия, определяемые ими (рис. 6.5). Поэтому задача обеспечения самопроверяемости складывается из обеспечения самопроверяемости программы и определения соответствующих требований к структуре и кодированию массивов описания булевых функций и данных. Массив описания (6.3) содержит четыре группы символов: 1) символы входных переменных; 2) операторные символы и символ окончания массива; 3) символы записи промежуточных переменных; 4) символы чтения промежуточных переменных. Для их различия используется идентифицирующий признак, который располагается в трех старших разрядах восьмиразрядного слова массива описания. С целью защищенности от одиночной ошибки коды признаков должны иметь кодовое расстояние не менее двух. В остальных разрядах слова массива кодируются: а) для входных переменных — относительный адрес переменной x_i в массиве входных переменных; б) для промежуточных переменных — относительный адрес переменной Y_j в массиве промежуточных переменных; в) для операторных символов — операции И, ИЛИ, НЕ, символ N . Массив входных переменных кодируется ассоциативным кодом.

Исходными данными для программы (рис. 6.5) являются начальные адреса массивов описания булевых функций, входных и промежуточных переменных. Программа состоит из блока 2 ввода символов из массива описания, идентифицирующих блоков 3, 5, 7, 9, 11, 13 и блоков выполнения операций и пересылок 4, 6, 8, 10, 12. После безошибочного вычисления системы функций в верхушке стека находятся значения функций в порядке, обратном последовательности их вычисления. Блок 14 выводит

Область внесения ошибок	Про		
	1 = 01, 0 = 00	1 = AA, 0 = 55	
Форма представления входных переменных			
Общее число испытаний	4944	4944	
Общее число ошибок	824	824	
Число правильных парафазных результатов	411 8,31 %	238 4,81 %	
Число неправильных парафазных результатов	53 1,07 %	7 0,015 %	
Число защитных результатов	4480 90,62 %	4699 95,17 %	
Число необнаруженных ошибок	26 3,16 %	28 3,39 %	

эти значения из стека для последующего анализа полученного результата. В результате ошибки либо в массиве описания, либо в блоке одного из идентификаторов может случиться, что считанный символ не будет идентифицирован ни одним из блоков 3, 5, 7, 9, 11, 13; тогда блок 15 осуществляет переход к программе обработки ошибки и выдает сигнал об ошибке. Причем в случае одиночной ошибки в массиве описания гарантируется ее обнаружение сразу при анализе искаженного символа, так как идентифицирующие признаки и массивы переменных кодируются кодом с обнаружением одиночных ошибок.

В табл. 6.10 представлены результаты исследований интерпретирующей программы и массива описания функций 1/6-СПТ. Из нее следует их полная защищенность от одиночных ошибок (число неправильных парафазных результатов равно нулю). Свойство самотестируемости программы не обеспечивается полностью, но число необнаруженных ошибок невелико (3.39 %). Их существование связано в основном с ошибочной трансформацией команд в команды, эквивалентные относительно данной программы, или в команды, не определенные в системе команд микропроцессора. Например, одна из ошибок трансформирует код (СЗ)₁₆ команды JMP addr в код (СВ)₁₆, не используемый в системе команд микропроцессора К580. Но по этому коду микропроцессор также выполняет команду JMP, и ошибка не обнаруживается. Свойство самотестируемости массива описания обеспечивается полностью и не зависит от кодирования переменных.

Таблица 6.10

грамма		Массив описания функций			
$1 = 1K,$ $0 = 0K$	$1 = 8K,$ $0 = 4K$	$1 = 0I,$ $0 = 00$	$1 = AA,$ $0 = 55$	$1 = 1K,$ $0 = 0K$	$1 = 8K,$ $0 = 4K$
4944	4944	1632	1632	1632	1632
824	824	272	272	272	272
193 3,91 %	177 3,58 %	422 25,86 %	346 21,26 %	11 0,67 %	5 0,31 %
0 0 %	0 0 %	0 0 %	0 0 %	0 0 %	0 0 %
4751 96,09 %	4767 96,42 %	1360 83,14 %	1286 78,74 %	1621 99,33 %	1627 99,69 %
29 3,52 %	28 3,39 %	0 0 %	0 0 %	0 0 %	0 0 %

Описанным способом может быть реализована любая система булевых функций с высокой степенью защищенности от одиночных ошибок. Свойство самопроверяемости обеспечивается самопроверяемостью системы функций относительно искажений ее формул, связностью вычисления этих формул и ассоциа-

Таблица 6.11

Область внесения ошибок	Программа		Массив описания функций	
Форма представления входных переменных	$1 = 1K,$ $0 = 0K$	$1 = 8K,$ $0 = 4K$	$1 = 1K,$ $0 = 0K$	$1 = 8K,$ $0 = 4K$
Общее число испытаний	160650	160650	53550	53550
Общее число ошибок	26775	26775	8925	8925
Число правильных парафазных результатов	659 0,41 %	633 0,39 %	48 0,09 %	103 0,19 %
Число неправильных парафазных результатов	0 0 %	0 0 %	0 0 %	0 0 %
Число защитных результатов	159991 99,59 %	160017 99,61 %	53502 99,91 %	53447 99,81 %
Число необнаруженных ошибок	87 0,325 %	83 0,310 %	0 0 %	1 0,011 %

тивным кодированием переменных. Как показали эксперименты, эти мероприятия дают также высокую защищенность программных реализаций и от кратных ошибок в слове программы и массиве данных.

В табл. 6.11 приведены результаты испытаний интерпретирующей программы 1/6-СПТ при внесении кратных ошибок. При этом число возможных искажений восьмиразрядного слова равно 255.

Известно [25], что в логических схемах одиночные проверяющие тесты обладают высокой обнаруживающей способностью относительно кратных неисправностей. Аналогичное свойство имеет место и для программных реализаций логических схем. Методы, обеспечивающие обнаружение одиночных ошибок в байте программы, позволяют обнаружить и подавляющее большинство кратных ошибок в байте.

6.5. Программные реализации дискретных устройств с памятью

Метод непосредственного вычисления может быть использован и при реализации программным путем дискретных устройств с памятью, которые в этом случае задаются с помощью системы булевых функций. В табл. 6.12 приведены результаты испытаний программы ДУ, заданного таблицей переходов (табл. 6.13). Кодирование его состояний выполнено кодом с постоянным весом «2 из 4», а вычисление функций осуществлено с использованием 1-реализации (см. гл. 1):

Таблица 6.12

Вариант программы	1	2	3	4
Длина программы, байт	195	195	202	202
Общее число испытаний	35880 100 %	35880 100 %	37168 100 %	37168 100 %
Число правильных результатов	22129 61,68 %	22029 61,38 %	23417 63 %	22888 61,57 %
Число неправильных незащитных результатов	626 1,74 %	587 1,66 %	47 0,13 %	47 0,13 %
Число защитных результатов	13125 36,58 %	13254 36,96 %	13704 36,87 %	14233 38,32 %
Общее число ошибок	1560	1560	1616	1616
Число необнаруженных ошибок	81 5,19 %	30 1,92 %	29 1,79 %	6 0,37 %

$$\begin{aligned}
y_1 &= a_1(y_1y_2 \vee y_1y_3) \vee a_2 \vee a_3(y_2y_4 \vee y_3y_4 \vee y_1y_2), \\
y_2 &= a_1(y_1y_4 \vee y_2y_4) \vee a_2(y_1y_2 \vee y_1y_3) \vee a_3, \\
y_3 &= a_1(y_2y_3 \vee y_3y_4 \vee y_1y_2 \vee y_1y_3) \vee a_3(y_1y_4 \vee y_2y_3 \vee y_1y_3), \\
y_4 &= (a_1 \vee a_2)(y_1y_4 \vee y_2y_4 \vee y_2y_3 \vee y_3y_4).
\end{aligned}
\tag{6.4}$$

Так как испытуемая программа реализует алгоритм ДУ с памятью, то для ее тестирования в процессе эксперимента формируется последовательность входных данных, определяемая путем подачи всех рабочих входных наборов a_j для каж-

Таблица 6.13

s	a		
	a_1	a_2	a_3
1	2	1	3
2	2	1	5
3	4	1	3
4	4	1	5
5	6	5	5
6	6	5	3

Таблица 6.14

$y_1y_2y_3y_4$	a		
	a_1	a_2	a_3
1 0 0 1	0101	1001	0110
0 1 0 1	0101	1001	1100
0 1 1 0	0011	1001	0110
0 0 1 1	0011	1001	1100
1 1 0 0	1010	1100	1100
1 0 1 0	1010	1100	0110

дого состояния ДУ. Для табл. 6.14 длина тестовой последовательности составляет 23 набора, поэтому общее число испытаний равно $195 \times 23 \times 8 = 35880$. Защитным результатом испытания считается тот, когда после вычисления функций y_1, y_2, y_3, y_4 вес кода внутреннего состояния ДУ оказывается не равным двум.

В табл. 6.12 приведены результаты испытаний четырех вариантов программ вычисления функций (6.4) методом непосредственного вычисления. В первом варианте программы не осуществлено никаких специальных мероприятий по повышению ее защищенности от ошибок, и защищенность программы обеспечивает механизм защиты булевых функций системы (6.4) от искажения ее переменных, присущий самопроверяемому ДУ. Наличие неправильных незащищенных результатов (1,74 %) и необнаруживаемых ошибок (5,19 %) объясняется двумя основными причинами. Первая аналогична причине нарушения защищенности в программах комбинационных ДУ. Она состоит в том, что ошибки программы по своим последствиям разнообразнее отказов аппаратных средств. А вторая причина связана со спецификой программ реализации ДУ с памятью, в которых производится последовательное вычисление логических функций и

возникает необходимость хранения предыдущих значений внутренних переменных y .

При разработке мероприятий по повышению защищенности необходимо придерживаться следующих принципов: 1) программа не должна содержать пассивные элементы (команды, группу команд), искажение которых не влияет на результат вычислений; 2) введение вспомогательных программных процедур с целью улучшения одного свойства программы не должно приводить к ухудшению другого. Первый принцип достигается путем устранения избыточности в описании самопроверяемого ДУ еще на этапе структурного синтеза и использования связной реализации логических функций. Второй принцип требует такого взаимного расположения программных процедур (структуры программы), чтобы их искажение, частичное или полное невыполнение приводило к расчету защитного результата.

В табл. 6.12 приведены результаты последовательного применения различных мероприятий по улучшению свойств самопроверяемости программной 1-реализации ДУ. Во втором варианте программы в отличие от исходного для кодирования логических сигналов 0 и 1 используются коды 00 и FF. Увеличение кодового расстояния в значительной степени снизило количество обнаруживаемых ошибок (1,92 %), так как программа стала более критична к значению исходных данных и промежуточных результатов. Основная масса неправильных незащитных результатов в первых двух вариантах программы связана с искажением процедуры перемещения массива значений внутренних переменных. Для их хранения необходимо иметь две области ОЗУ: первоначальную Y^t и обнуляемую Y^{t+1} . При вычислении новых значений переменных y их прежние значения выбираются из первоначальной области, а новые помещаются в обнуляемую область. После окончания вычислений массив новых значений y переносится в первоначальную область. Любая ошибка, которая приводит к невыполнению этой процедуры, эквивалентна тому, что сохраняется прежнее состояние ДУ. Это не изменяет веса кода внутреннего состояния и, следовательно, не обнаруживается. Для борьбы с этим явлением в третьем варианте программы перед переносом массива y производится также обнуление первоначальной области. В этом случае при нарушении процедуры переноса расчет новых значений функций (6.4) производится для нулевых значений переменных. Благодаря такому приему достигается также вычисление защитного результата в случае, если не выполняется расчет полного набора новых значений переменных y . Это позволило в данной программе исключить 550 неправильных незащитных результатов (см. табл. 6.13).

Наличие обнаруживаемых ошибок в программе третьего варианта объясняется тем, что существование двух массивов

переменных Y^t и Y^{t+1} позволяет микропроцессору в результате искажения адреса команды LDA производить эквивалентный выбор переменных y из массива Y^{t+1} вместо массива Y^t . В четвертом варианте программы это устраняется путем размещения массивов Y по адресам, взаимная трансформация которых невозможна при одиночном искажении байта, что исключило 23 необнаруживаемые ошибки. В результате указанных мероприятий достигается высокая степень самопроверяемости программной реализации ДУ. Она имеет 0,13 % незащитных результатов и 0,37 % необнаруживаемых ошибок.

Незащитные результаты, которые сохранились в программе, возникают вследствие искажения адреса команды STA (команды запоминания конечного результата). В результате этого искажения происходит взаимная замена переменных y_i и y_j в кодовом векторе состояния ДУ. Исключение таких ошибок достигается путем размещения результатов по адресам, не допускающим взаимной трансформации, и путем применения ассоциативного кодирования для внутренних переменных ДУ. Необнаруживаемые ошибки, которые сохранились в программе, связаны с искажением числа переменных в счетчике, по содержимому которого определяется завершение программы переноса массива. Искажение этого числа в большую сторону не влияет на результат вычисления состояний ДУ.

На основе приведенных результатов можно сформулировать требования к программам непосредственного вычисления самопроверяемых ДУ с памятью:

- 1) ДУ синтезируется методом 1-реализации;
- 2) с помощью алгоритма 3.1 из [34] должны быть исключены избыточные буквы из системы функций y ;
- 3) программа вычисления функций y строится как связанная реализация;
- 4) перед переносом значений переменных y из одного массива в другой производится обнуление последнего;
- 5) результаты вычислений располагаются в ячейках памяти, не допускающих взаимной трансформации адресов при одиночных ошибках.

СПИСОК ЛИТЕРАТУРЫ

1. Аксенова Г. П. Необходимые и достаточные условия построения полностью проверяемых схем свертки по модулю 2 // Автоматика и телемеханика. 1979. № 9. С. 126—135.
2. А. с. 921048 СССР. Парафазное триггерное устройство со счетным входом / В. В. Сапожников, Вл. В. Сапожников, В. Г. Трохов. Бюллетень изобретений. 1982. № 14.
3. А. с. 1017570 СССР. Устройство для включения исполнительного реле железнодорожной автоматики / О. К. Дрейман, Д. В. Гавзов, А. А. Бодров. Бюллетень изобретений. 1983. № 18.
4. А. с. 1588615 СССР. Мажоритарное устройство для управления включением исполнительного реле железнодорожной автоматики и телемеханики // Д. В. Гавзов, М. В. Илюхин, Е. Г. Сосновская. Бюллетень изобретений. 1990. № 32.
5. Бергер Д. О кодах, обнаруживающих ошибки в асимметричных каналах // Теория кодирования. Москва, 1964. С. 105—115.
6. Бимуканов М. К., Сапожников В. В., Сапожников Вл. В. Синтез быстродействующих тестеров для кодов с суммированием // Проблемы передачи информации. — 1989. Т. 25. № 2. С. 105—112.
7. Будинский Я. Логические цепи в цифровой технике: Пер. с чешск. М.: Связь, 1977.
8. Букреев И. Н., Мансуров Б. М., Горячев В. И. Микроэлектронные схемы цифровых устройств. М.: Советское радио, 1975.
9. Визирев И. С. Полностью самопроверяемые контрольные схемы с минимальным множеством тестов // Автоматика и вычислительная техника. 1982. № 1. С. 43—49.
10. Вирт Н. Алгоритмы + структуры данных = программы. М.: Мир, 1985.
11. Глушков В. М. Синтез цифровых автоматов. М.: Физматгиз, 1962.
12. Гольдман Р. С., Чипулис В. П. Техническая диагностика цифровых устройств. М.: Энергия, 1976.
13. Горожин А. Д., Крайнов К. М. Построение полностью самопроверяемых комбинационных устройств с использованием полиномиальных форм // Автоматика и телемеханика. 1979. № 12. С. 159—166.
14. Горожин А. Д. Метод синтеза управляющих автоматов с обнаружением неисправностей // Автоматика и вычислительная техника. 1973. № 6. С. 1—4.
15. Дербунович Л. В., Нешвеев В. В. Проектирование полностью самопроверяемых схем контроля на программируемых логических матрицах // Автоматика и телемеханика. 1986. № 4. С. 149—156.
16. Диаз М. Синтез самопроверяющихся последовательностных машин с применением кодов « k из n » и полностью самопроверяющихся контрольных устройств // Дискретные системы: Труды междунар. симп. Рига, 1974. Т. 2. С. 97—108.
17. Закревский А. Д. Логический синтез каскадных схем. М.: Наука, 1981.

18. Зельдин Е. А. Триггеры. М.: Энергоатомиздат, 1983.
19. Злотник Б. М. О построении циклических кодов с постоянным весом // Третья Всесоюзн. конф. по теории передач и кодирования информации: Тез. докл. Ташкент, 1967, С. 3—12.
20. Лазарев В. Г., Пийль Е. И. Синтез управляющих автоматов. М.: Энергоатомиздат, 1989.
21. Лазарев В. Г., Пийль Е. И., Турута Е. Н. Построение программируемых управляющих устройств. М.: Энергоатомиздат, 1984.
22. Липаев В. В. Надежность программного обеспечения АСУ. М.: Энергоиздат, 1981.
23. Липаев В. В. Тестирование программ. М.: Радио и связь, 1986.
24. Мазнев В. И. О синтезе самотестируемых $1/p$ -тестеров // Автоматика и телемеханика. 1978. № 9. С. 142—145.
25. Мелкадзе Т. Г., Сапожников В. В., Сапожников Вл. В. Свойства кратных неисправностей в контактных логических схемах // Автоматика и телемеханика. 1981. № 11. С. 139—146.
26. Мельников А. Г., Сапожников В. В., Сапожников Вл. В. Синтез самопроверяющихся тестеров для кодов с суммированием // Проблемы передачи информации. 1986. Т. 22. № 2. С. 85—97.
27. Обнаружение ошибок в программных реализациях самопроверяемых тестеров в микропроцессорных системах / В. В. Сапожников, Вл. В. Сапожников, А. В. Харитонов, В. М. Чухонин // Автоматика и телемеханика. 1989. № 12. С. 129—140.
28. Пархоменко П. П., Согомонян Е. С. Основы технической диагностики. М.: Энергоатомиздат, 1981.
29. Поспелов Д. А. Логические методы анализа и синтеза схем М.: Энергия, 1974.
30. Потемкин И. С. Функциональные узлы цифровой автоматики. М.: Энергоатомиздат, 1988.
31. Пупырев Е. И. Перестраиваемые автоматы и микропроцессорные системы. М.: Наука, 1984.
32. Сапожников В. В., Кравцов Ю. А., Сапожников Вл. В. Дискретные устройства железнодорожной автоматики, телемеханики и связи. М.: Транспорт, 1988.
33. Сапожников В. В., Сапожников Вл. В. Методы синтеза надежных автоматов. Л.: Энергия, 1980.
34. Сапожников В. В., Сапожников Вл. В. Дискретные автоматы с обнаружением отказов. Л.: Энергоатомиздат, 1984.
35. Сапожников В. В., Сапожников Вл. В., Цегловски Л. Синтез конечных автоматов с обнаружением отказов на T -триггерах // Автоматика и вычислительная техника. 1979. № 1. С. 25—26.
36. Сапожников В. В., Сапожников Вл. В., Шпак А. Ф. О характеристиках самопроверяющихся тестеров для равновесных кодов // Электронное моделирование. 1989. № 5. С. 39—44.
37. Сапожников В. В., Рабара В. Универсальный алгоритм синтеза $1/n$ -тестеров // Проблемы передачи информации. 1982. Т. 18. № 3. С. 62—73.
38. Сапожников В. В. Синтез структурно-совмещенных тестеров самоконтролирующихся автоматов // Известия АН СССР. Техническая кибернетика. 1982. № 4. С. 98—102.
39. Сапожников В. В., Сапожников Вл. В. Универсальный алгоритм синтеза самопроверяющихся тестеров для кодов с постоянным весом // Проблемы передачи информации. 1984. Т. 20. № 2. С. 65—76.
40. Сапожников В. В., Сапожников Вл. В., Цегловски Л. Синтез самопроверяющихся m/n -тестеров с максимальным быстродействием // Автоматика и телемеханика. 1988. № 10. С. 139—154.
41. Сапожников В. В., Сапожников Вл. В. Синтез быстродействующих тестеров для кодов с постоянным весом // Проблемы передачи информации. 1988. Т. 24. № 4. С. 84—92.
42. Сапожников В. В., Сапожников Вл. В., Шпак А. Ф. Блочная реализация самопроверяющихся тестеров для равновесных кодов // Электрон-

ное моделирование. 1990. № 1. С. 66—71.

43. Сапожников В. В. Контроль линейных комбинационных схем // Кибернетика. 1979. № 3. С. 44—47.

44. Сапожников В. В., Сапожников Вл. В., Харитонов А. В. Синтез дешифраторов с обнаружением неисправностей // Известия вузов. Приборостроение. 1989. Т. 32. № 5. С. 30—34

45. Согомонян Е. С., Слабаков Е. В. Самопроверяемые устройства и отказоустойчивые системы. М.: Радио и связь, 1989

46. Христов Х. Электронизация на осигурителната техника. София: Техника, 1984.

47. Шербаков Н. С., Подкопаев Б. П. Структурная теория аппаратного контроля цифровых автоматов. М.: Машиностроение, 1982.

48. Anderson D. A., Metze G. Design of Totally Self-Checking Check Circuits of M -out-of- N Codes // IEEE Trans. Computer. 1973. V. 22. № 3. P. 263—269.

49. Ashjaee J. M., Reddy S. M. Totally Self-Checking Checker for a Class of Separable Codes // Proc. 12th Annual Allerton Conf. on Circuit and System Theory. Illinois. 1974. P. 238—243.

50. Ashjaee J. M., Reddy S. M. On Totally Self-Checking Checkers for Separable Codes // IEEE Trans. Computer. 1977. V. 26. № 8. P. 737—744.

51. Bose B., Lin D. J. PLA Implementation of k -out-of- n Code TSC Checker // IEEE Trans. Computer. 1984. V. 33. № 6. P. 583—588.

52. Carter W. C., Schneider P. R. Design of Dynamically Checked Computers // JFIP Congress. Edinburgh, Scotland. 1968. P. 878—883.

53. Carter W. C., Duke K. A. A Simple Self-Testing Decoder Checking Circuit // IEEE Trans. Computer. 1971. V. 20. № 11. P. 1413—1414.

54. Chuang H. Y. H., Das S. Design of Fail-Safe Sequential Machines Using Separable Codes // IEEE Trans. Computer. 1978. V. 27. № 3. P. 249—252..

55. David R. A. Totally Self-Checking 1-out-of-3 Checker // IEEE Trans Computer. 1978. V. 27. № 6. P. 570—572.

56. Dandapani R. Derivation of Minimal Tests Sets for Monotonic Logic Circuits // IEEE Trans. Computer. 1973. V. 22. № 7. P. 657—661.

57. Elstathiou C., Halatsis C. Modular Realization of Totally Self-Checking Checkers for M -out-of- N Codes // Proc. FTCS-13, 13th Annual Int. Symp. Fault-Tolerant Computing. Milano, Italy. 1983. P. 154—161.

58. Gaitanis N., Halatsis C. A New Design Method for m -out-of- n TSC Checkers // IEEE Trans. Computer. 1983. V. 32. № 3. P. 273—283.

59. Gaitanis N. The Design of Totally Self-Checking TMR Fault-Tolerant Systems // IEEE Trans. Computer. 1988. V. 37. № 11. P. 1450—1454.

60. Gaitanis N. A Totally Self-Checking Error Indicator // IEEE Trans. Computer. 1985. V. 34. № 8. P. 758—761.

61. Golan P. Design of Totally Self-Checking Checker for 1-out-of-3 Code // IEEE Trans. Computer. 1984. V. 33. № 6. P. 285.

62. Halatsis C., Gaitanis N., Sigala M. A Fast and Efficient Totally Self-Checking Checkers for m -out-of- $(2m \pm 1)$ Codes // IEEE Trans. Computer. 1983. V. 32. № 5. P. 507—511.

63. Liu C. N. A State Variable Assignment Method for Asynchronous Sequential Switching Circuits // Journal of the Association for Computing Machinery. 1963. V. C-10. № 2. P. 209—216.

64. Maki G. Design Automation and Fault-Tolerant Computing a Self Checking Microprocessor Design // IEEE Trans. Computer. 1978. V. 27. № 1. P. 15—27.

65. Marouf M. A., Friedman A. D. Efficient Design of Self-Checking Checker for any m -out-of- n Code // IEEE Trans. Computer. 1978. V. 27. № 6. P. 482—490.

66. Marouf M. A., Friedman A. D. Design of Self-Checking Checkers for Berger Codes // Proc. 8th Annual Intern. Conf. on Fault-Tolerant Computing. Toulouse. 1978. P. 179—183.

67. Mili A. Self-Stabilizing Programs: The Fault-Tolerant Capability of

Self-Checking Programs // IEEE Trans. Computer. 1982. V. 31. № 7 P. 685—689.

68. Nanya T., Tohma Y. A 3-level Realization for Totally Self-Checking Checkers for M -out-of- N Codes // Proc. FTCS-13, 13th Annual Int. Symp. Fault-Tolerant Computing. Milano, Italy. 1983. P. 173—176.

69. Nanya T., Kawamura T. On Error Indication for Totally Self-Checking Systems // IEEE Trans. Computer. 1987. V. 36. № 11. P. 1389—1392.

70. Niraj K. J., Abraham J. A. The Design of Totally Self-Checking Embedded Checkers // Digest 14th Ann. Int. Conf. on Fault-Tolerant Computing. Florida, USA. 1984. P. 265—270.

71. Piestrak S. Design Method of Totally Self-Checking Checkers for M -out-of- N Codes // Proc. FTCS-13, 13th Annual Int. Symp. Fault-Tolerant Computing. Milano, Italy. 1983. P. 162—168.

72. Piestrak S. J. Design of Fast Self-Testing Checkers for a Class of Berger Codes // Digest 16th Ann. Int. Conf. on Fault-Tolerant Computing. Michigan, USA. 1985. P. 418—423.

73. Pradhan D. K. Asynchronous State Assignments with Unateness Properties and Fault-Secure Design // IEEE Trans. Computer. 1978. V. 27 № 5. P. 396—404.

74. Rabara V., Rabarova A. Three-Level TSC Checkers for 1-out-of- N Code // Proc. Intern. Conf. "Fault-Tolerant Systems and Diagnostics". Katowice, 1985. P. 223—229.

75. Reddy S. M. A Note on Self-Checking Checkers // IEEE Trans. Computer. 1974. V. 23. № 10. P. 1100—1102.

76. Reddy S. M., Wilson J. R. Easily Testable Cellular Realization for the (Exactly P)-out-of- n and (P or More)-out-of- n Logic Functions // IEEE Trans. Computer. 1974. V. 23. № 1. P. 98—100.

77. Smith J. E. The Design Self-Checking Check Circuits for a Class of Unordered Codes // J. Design Automation and Fault-Tolerant Computing. 1977. V. 1. № 4. P. 321—342.

78. Tohma Y., Ohyama Y., Sakai R. Realization of Fail-Safe Sequential Machines by Using a k -out-of- N Code // IEEE Trans. Computer. 1974. V. C-20. № 11. P. 1270—1275.

79. Wang S. L., Avizienis A. The Design of Totally Self-Checking Circuits Using Programmable Logic Arrays // Proc. FTCS-9, 9th Annual Int. Conf., Fault-Tolerant Computer. Madison, Wisconsin. 1979. P. 173—180.

ОГЛАВЛЕНИЕ

Предисловие	3
Глава первая. ПРИНЦИПЫ ПОСТРОЕНИЯ САМОПРОВЕРЯЕМЫХ ДИСКРЕТНЫХ УСТРОЙСТВ	5
1.1. Способы задания дискретных устройств	—
1.2. Свойства самопроверяемых дискретных устройств	14
1.3. Принципы построения самопроверяемых схем	19
Глава вторая. КОНТРОЛЬНЫЕ СХЕМЫ В САМОПРОВЕРЯЕМЫХ УСТРОЙСТВАХ	24
2.1. Свойства и характеристики самопроверяемых тестеров	—
2.2. Тестеры для равновесных кодов	31
2.3. Синтез тестеров для кодов $2mCm$	39
2.4. Тестеры для кодов $nC1$	43
2.4.1. Универсальные методы синтеза $1/n$ -СПТ	44
2.4.2. $1/3$ -тестеры	54
2.5. Универсальные методы синтеза тестеров для кодов « m из n »	58
2.6. Синтез быстродействующих m/n -тестеров	80
2.7. Блочная реализация тестеров	91
2.8. Каталоги m/n -тестеров	95
Глава третья. СХЕМЫ КОНТРОЛЯ И ДЕШИФРАЦИИ КОДОВ	104
3.1. Самопроверяемые схемы контроля кодов с повторением	—
3.2. Контроль кодов с суммированием	107
3.3. Тестеры для кодов с проверкой на нечетность (четность)	127
3.4. Самопроверяемые дешифраторы кодов	130
Глава четвертая. САМОПРОВЕРЯЕМЫЕ ТРИГГЕРНЫЕ УСТРОИ- СТВА	137
4.1. Свойства парафазных схем	—
4.2. Асинхронный T -триггер	139
4.3. Универсальный метод синтеза самопроверяемых триггерных устройств	145
4.4. Асинхронный D -триггер	147
4.5. Триггеры с двумя входами	149
4.6. Триггеры с динамическим управлением	153
4.7. Двухступенчатые триггеры	156
4.8. Синтез самопроверяемых ДУ на самопроверяемых триггерах	158
Глава пятая. САМОПРОВЕРЯЕМЫЕ ТИПОВЫЕ ЦИФРОВЫЕ УСТРОЙСТВА	161
5.1. Двоичные счетчики и регистры	—
5.2. Распределители	164
5.3. Генераторы и схемы синхронизации	168
5.4. Контрольная парафазная логика	170
5.5. Компараторы и связь с объектами	178
5.6. Организация контроля избыточных самопроверяемых систем	186
Глава шестая. САМОПРОВЕРЯЕМЫЕ ПРОГРАММНЫЕ РЕАЛИЗАЦИИ ДИСКРЕТНЫХ УСТРОЙСТВ	193
6.1. Принципы построения самопроверяемых управляющих программ в микропроцессорных системах	—
6.2. Программные реализации самопроверяемых тестеров	197
6.3. Модульное построение самопроверяемых программ	206
6.4. Интерпретирующая программа вычисления булевых функций	210
6.5. Программные реализации дискретных устройств с памятью	216
Список литературы	220

