

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ ИМПЕРАТОРА АЛЕКСАНДРА I»
(ФГБОУ ВО ПГУПС)

Д. В. Ефанов, Д. В. Пивоваров

СИНТЕЗ ДИСКРЕТНЫХ УСТРОЙСТВ С ОБНАРУЖЕНИЕМ НЕИСПРАВНОСТЕЙ

Учебное пособие

САНКТ-ПЕТЕРБУРГ
2019

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ ИМПЕРАТОРА АЛЕКСАНДРА I»
(ФГБОУ ВО ПГУПС)

Д. В. Ефанов, Д. В. Пивоваров

СИНТЕЗ ДИСКРЕТНЫХ УСТРОЙСТВ С ОБНАРУЖЕНИЕМ НЕИСПРАВНОСТЕЙ

Учебное пособие

САНКТ-ПЕТЕРБУРГ
2019

УДК 681.5+656.25
ББК 32.965
Е90

Рецензенты:

профессор кафедры «Управление и защита информации»
Российского университета транспорта,
доктор технических наук
В. Г. Сидоренко;

доцент, заведующий кафедрой «Автоматика, телемеханика
и связь на железнодорожном транспорте»
Российского университета транспорта, кандидат технических наук
А. А. Антонов

Ефанов Д. В.

Е90 Синтез дискретных устройств с обнаружением неисправностей :
учеб. пособие / Д. В. Ефанов, Д. В. Пивоваров. – СПб. : ФГБОУ ВО ПГУПС,
2019. – 80 с.

ISBN 978-5-7641-1257-2

Настоящее учебное пособие посвящено проблеме синтеза дискретных устройств с обнаружением неисправностей. Для этих целей используются как методы тестового, так и функционального диагностирования. В предлагаемом пособии авторы сконцентрировали свое внимание на проблеме синтеза средств функционального диагностирования. Пособие содержит три основных раздела, освещающих такие вопросы, как математический аппарат описания работы дискретных устройств, методика синтеза дискретных устройств по заданному описанию, анализ работы дискретного устройства с моделированием их «поведения» (в том числе при возникновении неисправностей) в специальных программных средах.

Использование материалов пособия позволяет обучающимся легко перейти от решения задачи «на бумаге» к реализации в виде реально функционирующей схемы. Пособие ориентировано на обучающихся железнодорожных вузов по специальности 23.05.05 «Системы обеспечения движения поездов».

УДК 681.5+656.25
ББК 32.965

ISBN 978-5-7641-1257-2

© Ефанов Д. В., Пивоваров Д. В., 2019
© ФГБОУ ВО ПГУПС, 2019

Введение

При построении надежных и безопасных систем автоматического управления на транспорте и в промышленности широко применяются методы резервирования и технического диагностирования [10, 14]. Эти устройства контролепригодны, неисправности в их элементах идентифицируются либо в процессе нормального функционирования, либо в специально отведенные промежутки времени при подаче тестовых воздействий [9]. Таким образом, свойство идентификации неисправностей, возникающих в блоках и компонентах устройства, закладывается в них еще на этапе проектирования.

Наделение устройства свойством обнаружения неисправностей требует от разработчика внесения в его структуру элементов, избыточных по отношению к минимально необходимым для выполнения им своих основных функций. Это может выражаться как во внесении избыточных элементов в основное устройство, так и в использовании специальных контрольных устройств, обеспечивающих «наблюдение» за работой отдельных компонентов и целых подсистем. Широкое распространение получили самопроверяемые логические схемы, при которых неисправности обнаруживаются в процессе нормального функционирования схем [12, 13]. Такие схемы основаны на использовании избыточного кодирования, базовые принципы которого используются при применении в аппаратных (и программных) реализациях логических устройств [7, 9, 15]. Наделение устройства «способностью» обнаружения неисправностей сопряжено с использованием кодов, ориентированных на обнаружение ошибок (а не на их исправление). Такая особенность связана с тем, что обнаружение ошибок кодами требует меньшей избыточности, чем исправление, а следовательно, аппаратная реализация устройства кодирования/декодирования окажется значительно проще. В реальных системах автоматического управления используются сложные структуры вычислительных комплексов, где, как правило, применяются резервированные блоки, каждый из которых наделяется свойством обнаружения неисправностей. Отказавший в процессе функционирования блок идентифицируется, а результаты вычислений в дальнейших операциях не используются. Такой объект блокируется и либо перезапускается в автоматическом режиме, либо находится в нерабочем режиме до замены на исправный блок.

Настоящее учебное пособие раскрывает особенности синтеза дискретных устройств, снабженных компонентами, позволяющими обнаруживать неисправности в процессе нормального функционирования.

1. Функции алгебры логики

1.1. Функции алгебры логики и устройства автоматики

Основным математическим аппаратом теории дискретных устройств является булева алгебра. Именно использование данного раздела дискретной математики дает возможность описания логики функционирования устройств и систем автоматики и вычислительной техники.

Функцией алгебры логики называется отображение множества входных двоичных наборов $\langle x_1 x_2 \dots x_l \rangle$ на множество значений 0 и 1.

Совокупностью функций алгебры логики описываются любые взаимозависимости между объектами автоматики, в том числе железнодорожной. Рассмотрим, например, следующую задачу.

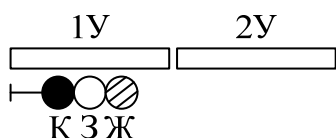


Рис. 1.1. Расположение проходного светофора

Имеется светофор, работающий в составе трехзначной автоблокировки (рис. 1.1), требуется разработать устройство включения огней светофора в зависимости от состояния передидлежащих участков контроля (блок-участков).

Каждое сигнальное показание светофора передается отдельным линзовым комплектом, и в один и тот же момент времени может гореть только одно показание: красный, желтый или зеленый огни светофора. Красный огонь загорается на светофоре в том случае, если передидлежащий участок логически занят (это может быть как наличие на участке подвижной единицы, так и неисправность, приводящая к включению запрещенного показания на светофоре). Желтый огонь на светофоре включается при свободности участка за светофором и занятости второго участка по удалению от светофора. Зеленый огонь включается при двух и более свободных участках перед светофором. Данная логика работы полностью соответствует таблице 1.1.

Таблица 1.1. Описание работы проходного светофора

1У	2У	К	Ж	З
↓	↓	↑	↓	↓
↓	↑	↑	↓	↓
↑	↓	↓	↑	↓
↑	↑	↓	↓	↑

Занятый участок за светофором будем отмечать логическим нулем, свободный – логической единицей. В реальности так и есть, поскольку блок-участок автоблокировки оборудуется нормально замкнутой рельсовой цепью, путевой приемник которой при логической свободности участ-

ка находится под током и выключается при логическом занятии участка. Горение какого-либо огня на светофоре обозначим логической единицей, а негорение – логическим нулем. Тогда таблицу 1.1 можно привести к таблице 1.2, являющейся таблицей истинности трех функций алгебры логики – функций включения красного, желтого и зеленого огней светофора (функций f_K , $f_{Ж}$ и f_3).

Таблица 1.2. Таблица истинности для работы проходного светофора

$x_1 = 1Y$	$x_2 = 2Y$	$f_K = K$	$f_{Ж} = Ж$	$f_3 = З$
0	0	1	0	0
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1

Из табл. 1.2 следует, что

$$f_K = \overline{x_1} \overline{x_2} \vee \overline{x_1} x_2 = \overline{x_1};$$

$$f_{Ж} = x_1 \overline{x_2};$$

$$f_3 = x_1 x_2.$$

Логическая схема, соответствующая приведенным выше функциям алгебры логики, изображена на рис. 1.2.

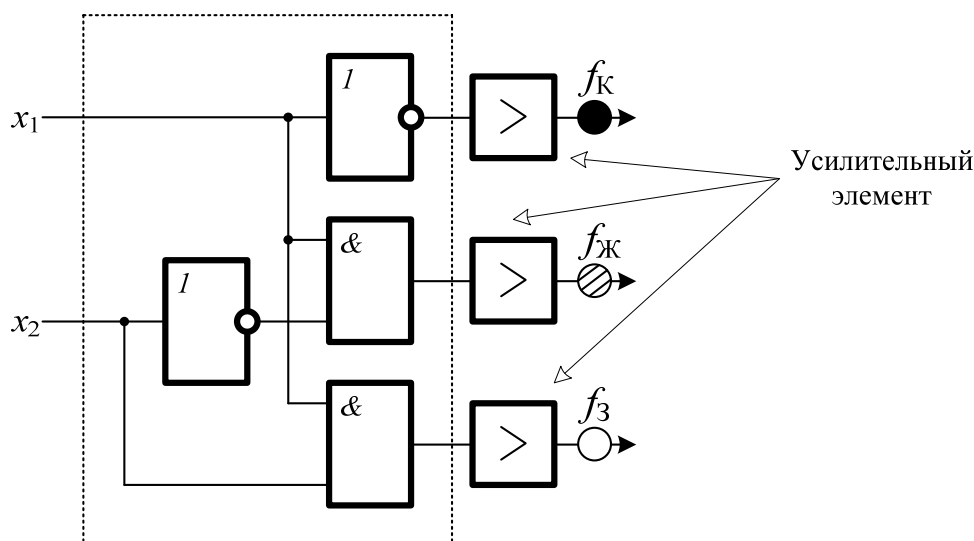


Рис. 1.2. Логическая схема управления проходным светофором (упрощенная)

Следует отметить, что реальная схема проходного светофора гораздо сложнее представленной на рис. 1.2, так как учитывает гораздо больше логических факторов (например, наличие двухнитевых ламп в линзовых комплектах или возможность переноса показания впередилежащего светофора при отказе лампового комплекта), а также сложные условия обеспечения безопасности функционирования. Тем не менее представленный пример показывает значимость булевой алгебры для современных систем автоматики.

1.2. Элементарные функции алгебры логики

Для описания работы устройств автоматики используется конечное множество элементарных функций алгебры логики – функций от одной и двух переменных. Описание этих функций дано в табл. 1.3.

Таблица 1.3. Элементарные функции алгебры логики

Название функции	Обозначение	Таблица истинности															
Функции, не зависящие от переменных																	
Константа нуля	$f = 0$	–															
Константа единицы	$f = 1$	–															
Функции, зависящие от одной переменной																	
Повторение	$f = x$	<table><tr><td>x</td><td>f</td></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	x	f	0	0	1	1									
x	f																
0	0																
1	1																
Инверсия (логическое отрицание, операция «НЕ»)	$f = \overline{x}$	<table><tr><td>x</td><td>f</td></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	x	f	0	1	1	0									
x	f																
0	1																
1	0																
Функции, зависящие от двух переменных																	
Конъюнкция (логическое умножение, функция «И»)	$f = x_1 \& x_2 = x_1 x_2$	<table><tr><td>x_1</td><td>x_2</td><td>f</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x_1	x_2	f	0	0	0	0	1	0	1	0	0	1	1	1
x_1	x_2	f															
0	0	0															
0	1	0															
1	0	0															
1	1	1															
Запрет переменной x_1	$f = x_1 \supset x_2$	<table><tr><td>x_1</td><td>x_2</td><td>f</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x_1	x_2	f	0	0	0	0	1	0	1	0	1	1	1	0
x_1	x_2	f															
0	0	0															
0	1	0															
1	0	1															
1	1	0															

Название функции	Обозначение	Таблица истинности															
Запрет переменной x_2	$f = x_2 \supset x_1$	<table> <tr><th>x_1</th><th>x_2</th><th>f</th></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	x_1	x_2	f	0	0	0	0	1	1	1	0	0	1	1	0
x_1	x_2	f															
0	0	0															
0	1	1															
1	0	0															
1	1	0															
Сложение по модулю два (неравнозначность, исключающее «ИЛИ»)	$f = x_1 \oplus x_2$	<table> <tr><th>x_1</th><th>x_2</th><th>f</th></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	x_1	x_2	f	0	0	0	0	1	1	1	0	1	1	1	0
x_1	x_2	f															
0	0	0															
0	1	1															
1	0	1															
1	1	0															
Функции, не зависящие от переменных																	
Дизъюнкция (логическое сложение, операция «ИЛИ»)	$f = x_1 \vee x_2$	<table> <tr><th>x_1</th><th>x_2</th><th>f</th></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	x_1	x_2	f	0	0	0	0	1	1	1	0	1	1	1	1
x_1	x_2	f															
0	0	0															
0	1	1															
1	0	1															
1	1	1															
Функция Вебба (стрелка Пирса, функция «ИЛИ-НЕ»)	$f = x_1 \downarrow x_2$	<table> <tr><th>x_1</th><th>x_2</th><th>f</th></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	x_1	x_2	f	0	0	1	0	1	0	1	0	0	1	1	0
x_1	x_2	f															
0	0	1															
0	1	0															
1	0	0															
1	1	0															
Эквивалентность (равнозначность)	$f = x_1 \sim x_2$	<table> <tr><th>x_1</th><th>x_2</th><th>f</th></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	x_1	x_2	f	0	0	1	0	1	0	1	0	0	1	1	1
x_1	x_2	f															
0	0	1															
0	1	0															
1	0	0															
1	1	1															
Импликация x_2 в x_1	$f = x_2 \rightarrow x_1$	<table> <tr><th>x_1</th><th>x_2</th><th>f</th></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	x_1	x_2	f	0	0	1	0	1	0	1	0	1	1	1	1
x_1	x_2	f															
0	0	1															
0	1	0															
1	0	1															
1	1	1															

Название функции	Обозначение	Таблица истинности															
Импликация x_1 в x_2	$f = x_1 \rightarrow x_2$	<table border="1"> <tr><td>x_1</td><td>x_2</td><td>f</td></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	x_1	x_2	f	0	0	1	0	1	1	1	0	0	1	1	1
x_1	x_2	f															
0	0	1															
0	1	1															
1	0	0															
1	1	1															
Функция Шеффера (штрих Шеффера, функция «И-НЕ»)	$f = x_1 / x_2$	<table border="1"> <tr><td>x_1</td><td>x_2</td><td>f</td></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	x_1	x_2	f	0	0	0	0	1	1	1	0	1	1	1	1
x_1	x_2	f															
0	0	0															
0	1	1															
1	0	1															
1	1	1															

1.3. Аксиомы и законы алгебры логики

Основные аксиомы и законы алгебры логики перечислены в табл. 1.4. Их объяснение и доказательство приведены в учебнике [13].

Таблица 1.4. Аксиомы и законы алгебры логики

Аксиомы алгебры логики			
$\bar{0} = 1;$ $\bar{1} = 0$	$0 \cdot 0 = 0 \cdot 1 = 1 \cdot 0 = 0;$ $1 \cdot 1 = 1$	$0 \vee 0 = 0;$ $0 \vee 1 = 1 \vee 0 = 1 \vee 1 = 1$	
Законы алгебры логики			
Закон нулевого множества	$0 \cdot x = 0;$ $0 \vee x = x$	Сочетательный закон	$x_1(x_2x_3) = (x_1x_2)x_3;$ $x_1 \vee (x_2 \vee x_3) = (x_1 \vee x_2) \vee x_3$
Закон единичного множества	$1 \cdot x = x;$ $1 \vee x = 1$	Распределительный закон	$x_1(x_2 \vee x_3) = x_1x_2 \vee x_1x_3$
Закон повторения	$x \cdot x = x;$ $x \vee x = x$	Закон поглощения	$x_1 \vee x_1x_2 = x_1;$ $x_1(x_1 \vee x_2) = x_1$
Закон двойного отрицания	$\overline{\overline{x}} = x$	Закон поглощения (склеивания, объединения)	$x_1x_2 \vee x_1\overline{x_2} = x_1;$ $(x_1 \vee x_2)(\overline{x_1} \vee \overline{x_2}) = \overline{x_1x_2}$
Закон логического нуля	$x \cdot \overline{x} = 0$	Правила де Моргана	$\overline{x_1x_2} = \overline{x_1} \vee \overline{x_2};$ $\overline{x_1 \vee x_2} = \overline{x_1} \cdot \overline{x_2}$
Закон логической единицы	$x \vee \overline{x} = 1$	Закон Блейка – Порецкого	$\overline{x_1x_2 \vee x_1x_3} = \overline{x_1x_2} \vee \overline{x_1x_3} \vee \overline{x_2x_3}$
Переместительный закон	$x_1x_2 = x_2x_1;$ $x_1 \vee x_2 = x_2 \vee x_1$	Обобщенный закон упрощения	$x(x \vee \varphi) = x;$ $x \vee \varphi x = x,$ φ – любое логическое выражение

1.4. Конечные автоматы

Логические устройства автоматики и вычислительной техники делятся на два больших класса – устройства, не обладающие памятью (комбинационные схемы), и устройства, обладающие памятью (многотактные схемы, или конечные автоматы). Комбинационные схемы описываются гораздо проще, чем конечные автоматы – с помощью таблиц истинности. Описанием же конечного автомата является таблица переходов или альтернативная форма – граф переходов.

Признаком наличия памяти у устройства автоматики является наличие обратных связей. Комбинационные схемы таковых не имеют.

Конечные автоматы могут иметь асинхронную и синхронную реализации. В асинхронном конечном автомате нет тактовых генераторов, а критические состязания устраняются путем специального кодирования состояний. В синхронной реализации для исключения состязаний элементов памяти используются тактовые генераторы, вырабатывающие специальные синхронизирующие импульсы.

Рассмотрим реализацию синхронного конечного автомата в виде автомата Мили I рода (рис. 1.3).

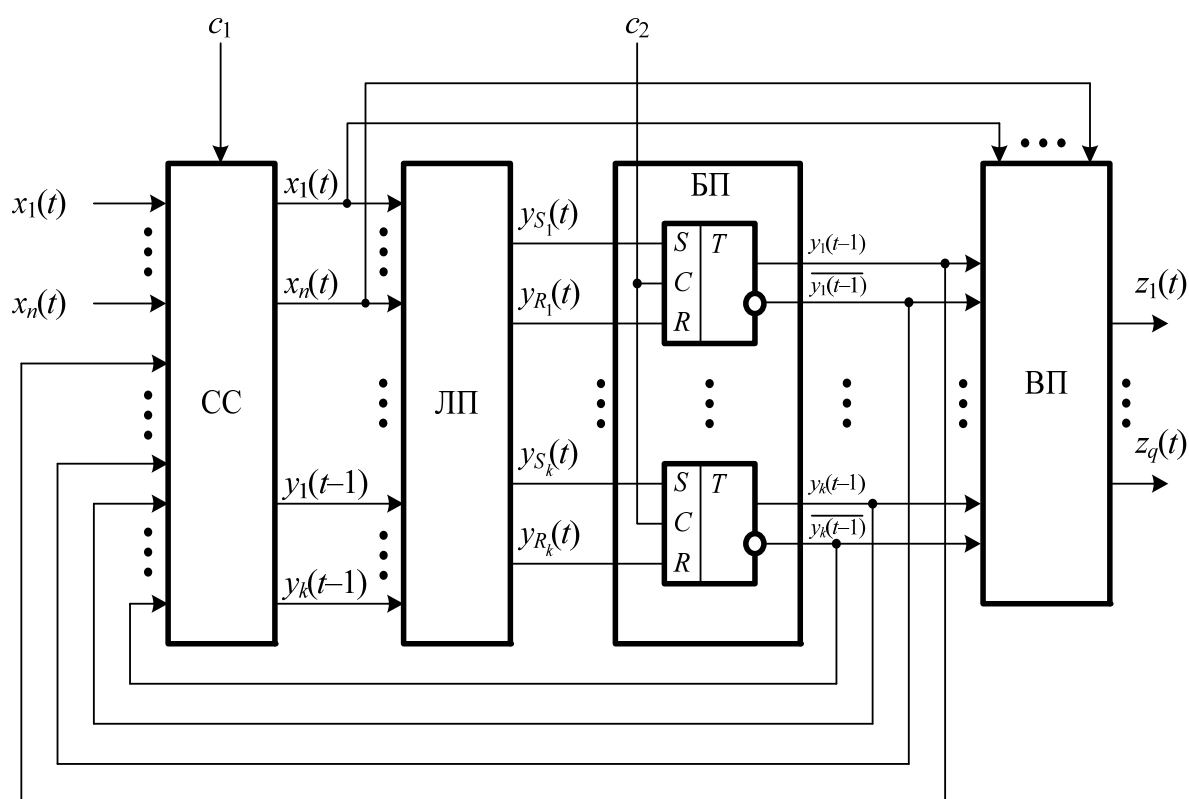


Рис. 1.3. Структурная схема автомата Мили I рода

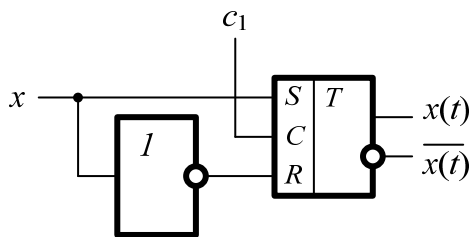


Рис. 1.4. Схема синхронизации входных воздействий

В моменты времени $t = 1, 2, 3, \dots$ по переднему фронту синхроимпульсов c_1 срабатывают схемы синхронизации (СС) и на вход логического преобразователя (ЛП) поступают сигналы $x(t)$ и $y(t-1)$. В качестве схемы синхронизации используется техническое решение, приведенное на рис. 1.4.

Логический преобразователь вычисляет новые значения функций $y_S(t)$ и $y_R(t)$. Затем, когда все переходные процессы в логическом преобразователе закончены, спустя время τ поступает синхроимпульс c_2 и происходит переключение триггеров (рис. 1.5). Формируются новые значения сигналов $y(t-1)$ и $\overline{y(t-1)}$. На вход логического преобразователя эти сигналы поступают только в следующий момент времени, когда придет новый синхроимпульс c_1 . Данные сигналы поступают на входы выходного преобразователя (ВП).

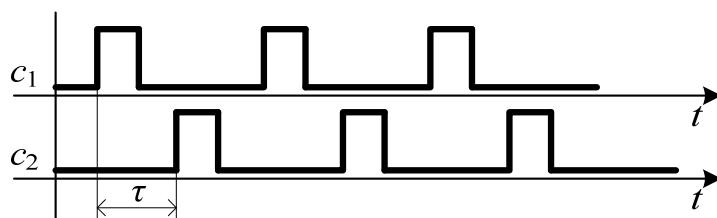


Рис. 1.5. Временная диаграмма работы синхроимпульсов

За время τ все переходные процессы в логическом преобразователе закончатся и все триггеры блока памяти (даже самый медленно действующий) перейдут в новое состояние. Этим исключаются критические состязания элементов памяти. Кроме того, упрощается сам процесс синтеза, поскольку кодирование состояний может быть произвольным и нет необходимости анализировать схему для выявления критических состязаний элементов памяти.

Рассмотрим особенности синтеза конечных автоматов с возможностью обнаружения неисправностей в процессе эксплуатации.

2. Синтез синхронных автоматов с минимизацией количества состояний

2.1. Способы задания конечных автоматов

Конечный автомат задается алгоритмом своей работы. Это может быть либо словесное описание особенностей функционирования, либо формализованное описание в виде:

- специальных вход-выходных временных последовательностей (рис. 2.1);
- таблицы переходов и выходов (табл. 2.1);
- графа переходов (рис. 2.2).

Все эти способы задания конечного автомата равнозначны.

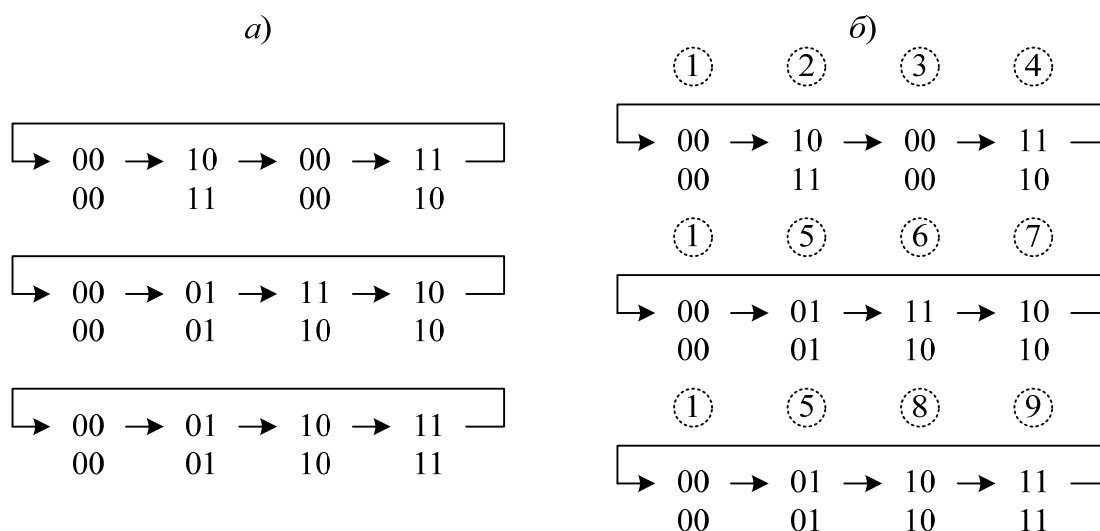


Рис. 2.1. Вход-выходные временные последовательности:
а – исходные; б – пронумерованные

Все способы задания конечных автоматов подразумевают описание всех возможных их состояний и переходов между состояниями. Переходы между состояниями осуществляются путем изменения входных воздействий.

На рис. 2.1 изображены вход-выходные последовательности, задающие работу конечного автомата. Каждая из трех последовательностей содержит информацию о значениях входов x_1 и x_2 автомата (верхнее двузначное число) и значениях выходов z_1 и z_2 (нижнее двузначное число). Каждая группа чисел определяет полное устойчивое состояние конечного автомата – такого состояния, в котором автомат находится сколь угодно долго при изменении входных воздействий, соответствующих допустимому переходу в другое состояние. Стрелками указаны возможные переходы автомата.

Таблица 2.1. Исходная таблица переходов и выходов

S	Столбец			
	I	II	III	IV
	Значения входных переменных			
	x_1x_2			
	00	01	10	11
1	(1), 00	5, 01	2, 11	~
2	3, 00	~	(2), 11	~
3	(3), 00	~	~	4, 10
4	1, 00	~	~	(4), 10
5	~	(5), 01	8, 10	6, 10
6	~	~	7, 10	(6), 10
7	1, 00	~	(7), 10	~
8	~	~	(8), 10	9, 11
9	1, 00	~	~	(9), 11

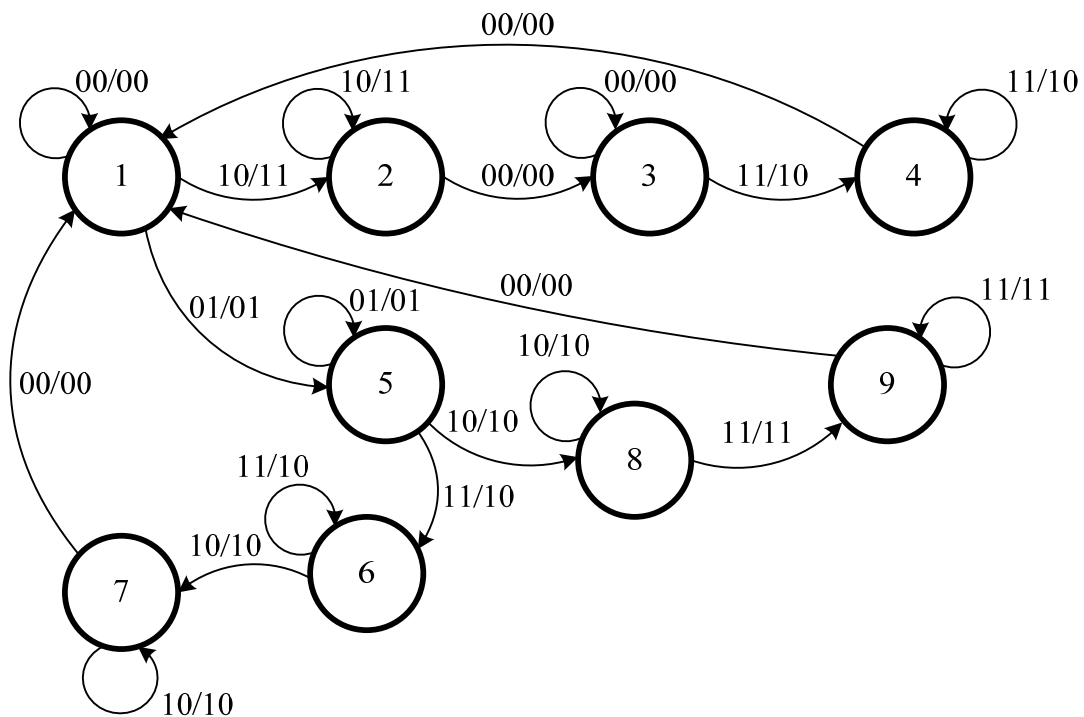


Рис. 2.2. Исходный граф переходов

От вход-выходной последовательности нетрудно перейти к двум другим формам задания конечного автомата – к таблице переходов и графу переходов. Для этого необходимо пронумеровать состояния конечного автомата по следующим правилам:

1. Вводится начальное состояние, соответствующее первому устойчивому состоянию – состоянию № 1.
2. Далее для каждого нового состояния вводится новый номер.
3. Если некоторое новое состояние в новой вход-выходной последовательности совпадает с уже введенным ранее в иной вход-выходной последовательности и все предыдущие состояния совпадают с ранее введенными, то новое состояние не вводится, а рассматриваемое состояние нумеруется уже известным номером.
4. При нумерации в одной последовательности не должно быть повторяющихся номеров, в противном случае при построении автомата будет нарушена последовательность смены состояний и образован цикл переключений, нарушающий заданный алгоритм функционирования.

Номера состояний соответствуют тем возможным состояниям, в которые может переходить конечный автомат при своей работе. Переход к табличной форме задания от вход-выходных последовательностей производят следующим образом.

Таблица 2.1 заполняется, исходя из анализа вход-выходной последовательности. Каждая строка таблицы соответствует устойчивому состоянию проектируемого конечного автомата. В столбцах таблицы указаны возможные входные воздействия x_1 и x_2 (для данного примера они разделены на четыре группы: I – 00, II – 01, III – 10, IV – 11). Заполнение таблицы осуществляется путем последовательного анализа каждой вход-выходной последовательности:

1. На пересечении первой строки и первого столбца ставится число, соответствующее номеру данного устойчивого состояния. Это число заключается в скобки, что соответствует полному устойчивому состоянию. Через запятую указываются значения выходов z_1 и z_2 .
2. Анализируется переход из первого состояния со значениями входов/выходов 00/00 к состоянию со значениями входов/выходов 10/11. Находится клетка таблицы, расположенная на пересечении строки первого состояния и столбца, соответствующего значениям входов. В данную клетку заносится номер состояния, в которое должен перейти автомат. Данный номер в скобки не заключается, так как соответствует неустойчивому состоянию перехода в следующее состояние. Через запятую также указываются значения выходов автомата, но уже в новом состоянии.
3. В таблице находится строка, соответствующая номеру внесенного ранее в клетку состояния. В эту строку в том же столбце, который соответствует входным значениям перехода в данное состояние, записывается номер данного состояния. Он заключается в скобки, что указывает на полное устойчивое состояние.

4. Процедура п. 2 повторяется, но уже с учетом следующего перехода.
5. После внесения данных из всех вход-выходных последовательностей все клетки таблицы переходов, которые остались незаполненными, заполняются безразличными состояниями. В данные состояния автомат никогда не попадает.

Задание автомата в виде графа оказывается еще более простым, а сам переход от вход-выходной последовательности реализуется так:

1. Для каждого состояния вход-выходной последовательности вводится узел графа (вершина) с номером данного состояния.
2. Каждому узлу графа соответствует полное устойчивое состояние, что обозначается петлей – дугой исходящей и входящей в данный узел. Над дугой указываются через косую черту значения входов/выходов, соответствующих данному состоянию.
3. Далее узлы соединяются дугами согласно имеющейся во вход-выходной последовательности информации о переходах. Дуга между двумя состояниями изображается в том случае, если во вход-выходной последовательности имеется соответствующий переход. Над дугой записываются значения входов/выходов, которые обеспечивают данный переход в соответствующее состояние.

Исходные таблица переходов и граф переходов приведены в табл. 2.1 и на рис. 2.2.

Поскольку граф переходов не является полным (не все узлы графа попарно связаны дугами между собой), а в таблице переходов имеются безразличные состояния, возможно сокращение числа состояний конечного автомата – его минимизация.

Минимизация числа состояний автомата позволяет уменьшить сложность его технической реализации по сравнению с исходным вариантом за счет сокращения числа необходимых для реализации элементов памяти. К примеру, для реализации автомата по исходным таблице переходов или графу переходов потребуется как минимум четыре элемента памяти (девять состояний кодируются минимум четырьмя переменными).

Рассмотрим процедуру минимизации таблицы переходов по известному алгоритму [11].

2.2. Минимизация таблиц переходов

Для минимизации используется понятие максимальных подмножеств совместимых строк таблицы переходов.

Строки таблицы могут быть объединены с учетом непротиворечивости заполнения соответствующих столбцов. Например, строки № 1 и № 4 исходной таблицы переходов могут быть объединены (табл. 2.2). Правила объединения строк следующие:

1. Если в столбцах записаны одинаковые состояния, но одно из них устойчивое, а одно – неустойчивое, то в результирующей строке, объединяющих оба состояния, записывается устойчивое состояние.
2. Если в столбцах записаны какое-либо полное состояние (устойчивое или неустойчивое) и безразличное состояние, то в результирующей строке записывается номер полного состояния.
3. При объединении двух строк, в столбцах которых записаны безразличные состояния, записывается безразличное состояние.

Строки, у которых в одинаковых столбцах записаны номера различных полных состояний, объединить невозможно.

Объединять можно не только две строки, но и большее их число.

Таблица 2.2. Исходная таблица переходов и выходов

S	Столбец			
	I	II	III	IV
	Значения входных переменных			
	x_1x_2			
	00	01	10	11
1	(1), 00	5, 01	2, 11	~
4	1, 00	~	~	(4), 10
1, 4	(1), 00	5, 01	2, 11	(4), 10

Анализ таблицы 2.1 показывает, что число вариантов объединения строк велико. Наилучший вариант объединения находится путем минимизации.

С этой целью находятся все варианты объединения строк по столбцам таблицы переходов – находятся подмножества строк, в которых в столбце i проставлено состояние j или знак безразличного состояния (подмножества E_i^j):

$$E_I^1 = \{1; 4; 5; 6; 7; 8; 9\};$$

$$E_{III}^7 = \{3; 4; 6; 7; 9\};$$

$$E_I^3 = \{2; 3; 5; 6; 8; 9\};$$

$$E_{IV}^4 = \{1; 2; 3; 4; 7\};$$

$$E_{II}^5 = \{1; 2; 3; 4; 5; 6; 7; 8; 9\};$$

$$E_{IV}^6 = \{1; 2; 5; 6; 7\};$$

$$E_{III}^2 = \{1; 2; 3; 4; 9\};$$

$$E_{IV}^9 = \{1; 2; 7; 8; 9\}.$$

$$E_{III}^8 = \{3; 4; 5; 8; 9\};$$

Следующим шагом минимизации являются получение пересечений подмножеств совместимых по столбцам строк и поиск оптимального объединения. Находятся все множества $E_{I,II,III,IV}^{j_1,j_2,j_3,j_4}$, являющиеся пересечениями всех введенных ранее подмножеств по столбцам таблицы переходов (рис. 2.3). Не пересекаются между собой только подмножества, входящие в один и тот же столбец таблицы переходов.

Число пересечений подмножеств совместимых по столбцам строк равно $2 \cdot 1 \cdot 3 \cdot 3 = 18$. Принцип пересечения подмножеств показан на рис. 2.4. Фактически при пересечении четырех подмножеств требуется найти повторяющиеся в каждом из пересекаемых подмножеств номера состояний.

Результат пересечения следующий:

$$\begin{aligned}
 E^{1,5,2,4} &= \{1;4\} - a; & E^{3,5,2,4} &= \{2;3\} - g; \\
 E^{1,5,2,6} &= \{1\} (-); & E^{3,5,2,6} &= \{2\} (-); \\
 E^{1,5,2,9} &= \{1,9\} (-); & E^{3,5,2,9} &= \{2;9\} - h; \\
 E^{1,5,8,4} &= \{4\} (-); & E^{3,5,8,4} &= \{3\} (-); \\
 E^{1,5,8,6} &= \{5\} - b; & E^{3,5,8,6} &= \{5\} (-); \\
 E^{1,5,8,9} &= \{8;9\} - c; & E^{3,5,8,9} &= \{8;9\} (-); \\
 E^{1,5,7,4} &= \{4;7\} - d; & E^{3,5,7,4} &= \{3\}; \\
 E^{1,5,7,6} &= \{6;7\} - e; & E^{3,5,7,6} &= \{6;7\} (-); \\
 E^{1,5,7,9} &= \{9\}; & E^{3,5,7,9} &= \{7;9\} (-).
 \end{aligned}$$

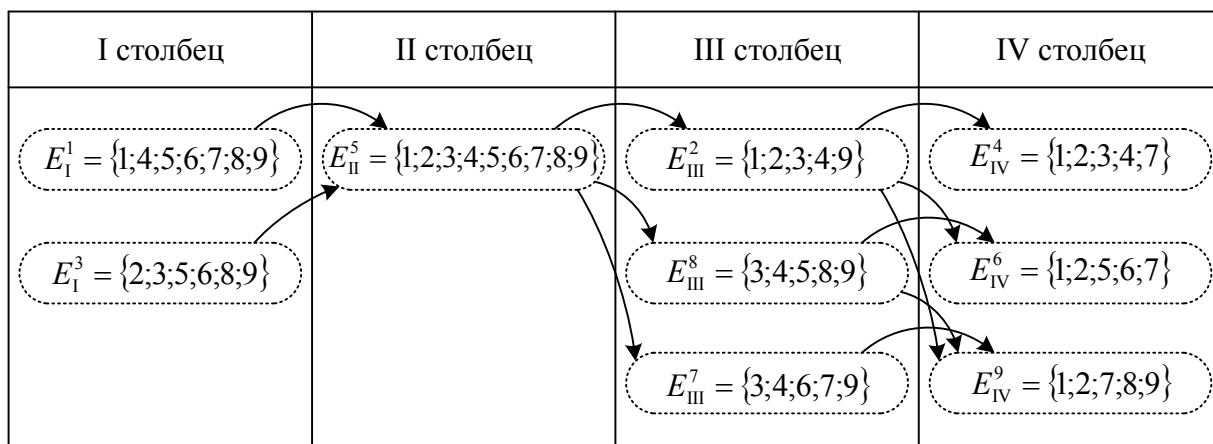


Рис. 2.3. Варианты пересечений подмножеств совместимых по столбцам строк

E_I^1	1			4	5	6	7	8	9
E_{II}^5	1	2	3	4	5	6	7	8	9
E_{III}^2	1	2	3	4					9
E_{IV}^4	1	2	3	4			7		
$E^{1,5,2,4}$	1			4					

Рис. 2.4. Принцип определения пересечений

Из полученных множеств исключаются те, которые полностью входят в другие множества, а оставшиеся множества обозначаются латинскими буквами.

Следующий этап подразумевает составление таблицы покрытий (табл. 2.3). В данной таблице по строкам перечисляются все возможные состояния конечного автомата, а в столбцах – все варианты объединения строк. На пересечении строки и столбца ставится знак покрытия (например, «х»), если данное состояние входит в соответствующее столбцу объединение.

Таблица 2.3. Таблица покрытий

S	a	b	c	d	e	g	h
	$\{1;4\}$	$\{5\}$	$\{8;9\}$	$\{4;7\}$	$\{6;7\}$	$\{2;3\}$	$\{2;9\}$
1	×						
2						×	×
3						×	
4	×			×			
5		×					
6					×		
7				×	×		
8			×				
9			×				×

После составления таблицы покрытия выписывается логическое выражение вида «конъюнкция дизъюнкций букв, позволяющих покрыть каждое из состояний»:

$$Q = a(g \vee h)g(a \vee d)be(d \vee e)c(c \vee h) = abceg.$$

Выражение Q сокращено за счет использования известных законов алгебры логики [13].

Из конечного выражения Q выбирается конъюнкция с минимальным количеством букв (конъюнкция минимальной длины). Она соответствует оптимальному покрытию:

$$W = abceg = \{1; 4\} \{5\} \{8; 9\} \{6; 7\} \{2; 3\}.$$

Удаляя только повторяющиеся номера (оставляя только один из них), получаем окончательный вариант объединения строк таблицы переходов:

$$W' = \{1; 4\} \{5\} \{8; 9\} \{6; 7\} \{2; 3\}.$$

Результат объединения строк показан в табл. 2.4.

Таблица 2.4. Минимизированная таблица переходов и выходов до перенумерации состояний

S	S^*	Группа			
		I	II	III	IV
		Значения входных переменных			
		x_1x_2			
		00	01	10	11
1, 4	s_1	(1), 00	5, 01	2, 11	(4), 10
2, 3	s_2	(3), 00	~	(2), 11	4, 10
5	s_3	~	(5), 01	8, 10	6, 10
6, 7	s_4	1, 00	~	(7), 10	(6), 10
8, 9	s_5	1, 00	~	(8), 10	(9), 11

В табл. 2.4 присутствует пять состояний, однако номера состояний в клетках изменяются от 1 до 9. Требуется перенумеровать состояния. Каждому состоянию, соответствующему каждой объединенной строке, присваивается новый номер, или новое обозначение – s_j (см. столбец S^* в табл. 2.4). Например, все состояния, соответствующие первой строке (состояния 1 и 4 в табл. 2.4), заменяются состоянием s_1 ; все состояния, соответствующие второй строке (состояния 2 и 3 в табл. 2.4), заменяются состоянием s_2 , и т. д.

Конечный вариант минимизации таблицы переходов представлен в табл. 2.5, а соответствующий такой таблице переходов конечный автомат представлен на рис. 2.6.

Следует отметить, что в настоящее время известны системы автоматической минимизации конечных автоматов. К примеру, минимизировать конечные автоматы позволяет известный программный модуль SIS (A System for Sequential Circuit Synthesis), разработанный в университете Калифорнии (Berkeley) в конце 1980-х – начале 1990-х [23, 24].

Таблица 2.5. Минимизированная таблица переходов и выходов после перенумерации состояний

S^*	Группа			
	I	II	III	IV
	Значения входных переменных			
	x_1x_2			
	00	01	10	11
s_1	$(s_1), 00$	$s_3, 01$	$s_2, 11$	$(s_1), 10$
s_2	$(s_2), 00$	\sim	$(s_2), 11$	$s_1, 10$
s_3	\sim	$(s_3), 01$	$s_5, 10$	$s_4, 10$
s_4	$s_1, 00$	\sim	$(s_4), 10$	$(s_4), 10$
s_5	$s_1, 00$	\sim	$(s_5), 10$	$(s_5), 11$

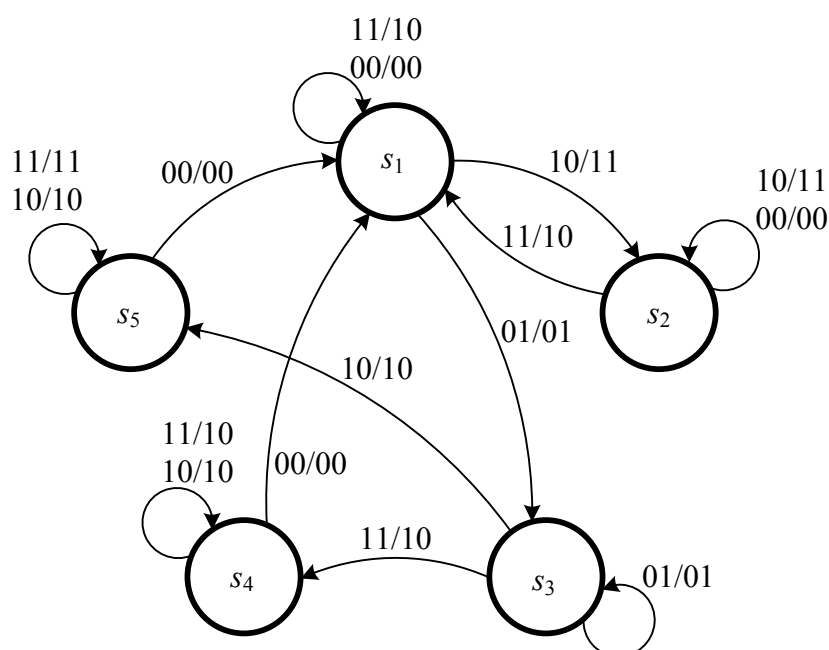


Рис. 2.6. Минимизированный граф переходов

2.3. Синтез избыточных синхронных автоматов

2.3.1. Кодирование таблицы переходов

Конечный автомат синтезируется непосредственно по минимизированной таблице переходов (см. табл. 2.5) или же по минимизированному графу переходов (см. рис. 2.6). Для синтеза избыточного конечного автомата выбирается способ кодирования каждого состояния.

Минимальное число элементов памяти, необходимое для синтеза конечного автомата, определяется по формуле

$$N_{\text{ЭП}} = \lceil \log_2 N_s \rceil,$$

где N_s – число состояний конечного автомата;

запись $\lceil \dots \rceil$ обозначает целое сверху от вычисляемого значения (ближайшее целое, превосходящее данное).

Пользуясь этой формулой, получаем:

$$N_{\text{ЭП}} = \lceil \log_2 5 \rceil = \lceil 2,32 \rceil = 3.$$

Таким образом, для синтеза конечного автомата потребуется три элемента памяти – Y_1 , Y_2 и Y_3 .

Существует большое число вариантов кодирования пяти состояний тремя переменными. Способ кодирования определяет различные характеристики реализуемого дискретного устройства и влияет на сложность технической реализации логического и выходного преобразователей, количество переключений логических элементов (непосредственно влияет на энергопотребление устройства), быстродействие комбинационных составляющих и т. д.

В табл. 2.6 представлена кодированная таблица переходов, где каждое состояние закодировано последовательно возрастающими двоичными числами.

Таблица 2.6. Кодированная таблица переходов

S^*	Группа			
	I	II	III	IV
	Значения входных переменных			
	x_1x_2			
	00	01	10	11
$s_1=000$	(000), 00	010, 01	001, 11	(000), 10
$s_2=001$	(001), 00	~	(001), 11	000, 10
$s_3=010$	~	(010), 01	100, 10	011, 10
$s_4=011$	000, 00	~	(011), 10	(011), 10
$s_5=100$	000, 00	~	(100), 10	(100), 11

По кодированной таблице переходов синтезируется конечный автомат.

Для исключения критических состязаний могут быть выбраны две основные реализации – асинхронная и синхронная.

Асинхронная реализация потребует увеличения количества элементов памяти и использования специального кодирования для того, чтобы одновременно мог переключаться только один элемент памяти при переходе в каждое состояние.

Синхронная реализация потребует синхронизации работы элементов памяти с помощью тактовых генераторов. Это связано с выбором конкретных элементов памяти.

Рассмотрим синхронную реализацию конечного автомата.

2.3.2. Выбор элементов памяти и получение значений функций их включения

Для реализации конечного автомата может быть выбран любой тип элемента памяти (триггера). Основными типами триггеров являются RS -, JK -, E -, R -, S -, D -, T -триггеры [12]. Таблицы 2.7 и 2.8 представляют собой таблицы переходов триггеров различных типов.

Таблица 2.7. Таблица переходов триггеров с двумя управляющими входами

t^n		t^{n+1}				
R^n, K^n	S^n, J^n	Y^{n+1}				
		RS -триггер	JK -триггер	E -триггер	R -триггер	S -триггер
0	0	Y^n	Y^n	Y^n	Y^n	Y^n
0	1	1	1	1	1	1
1	0	0	0	0	0	0
1	1	\sim	$\overline{Y^n}$	Y^n	0	1

Таблица 2.8. Таблица переходов триггеров с одним управляющим входом

t^n	t^{n+1}	
D^n, T^n	Y^{n+1}	
	D -триггер	T -триггер
0	0	Y^n
1	1	$\overline{Y^n}$

Все триггеры характеризуются числом управляющих входов (одно- или двухвходовые триггеры). Принципы обозначения входов триггеров следующие.

RS -, E -, R -, S -триггеры имеют управляющие входы S (set – установить) и R (reset – сбросить).

JK -триггер имеет управляющие входы J (jump – включить) и K (kill – выключить).

Аналогично обозначаются сигналы на управляющих входах D - и T -триггеров, соответственно входах D (delay – задержать) и T (toggle – переключить).

В табл. 2.6 и 2.7 над обозначениями входов триггеров указаны переменные n , обозначающие значение на входе триггера в настоящий момент

времени t^n . В остальных столбцах таблиц указаны состояния выходов триггеров Y^{n+1} в последующий момент времени t^{n+1} .

Рассмотрим реализацию конечного автомата более подробно для двух типов триггеров – RS - и JK -типов.

Сформулируем *правила заполнения* таблицы истинности функций Y_S и Y_R включения RS -триггеров.

1. Пусть в клетке (x', y'_1, y'_2, y'_3) кодированной таблицы (см. табл. 2.6) записан код y''_1, y''_2, y''_3 на пересечении столбца x' и строки y'_1, y'_2, y'_3 . Значение y'_i указывает на состояние триггера в предшествующий момент времени $t-1$, а значение y''_i определяет состояние, в которое триггер должен переключиться в момент времени t .
2. Для определения функций Y_S и Y_R используются такие правила:
 - если $y(t-1) = 0, y(t) = 1$, то $Y = 1, Y_R = 0$, так как триггер должен переключиться из состояния 0 в состояние 1;
 - если $y(t-1) = 0, y(t) = 0$, то $Y_S = 0, Y_R = \sim$, так как триггер был в состоянии 0 и должен сохранить это состояние;
 - если $y(t-1) = 1, y(t) = 0$, то $Y_S = 0, Y_R = 1$, так как триггер должен переключиться из состояния 1 в состояние 0;
 - если $y(t-1) = 1, y(t) = 1$, то $Y_S = \sim, Y_R = 0$, так как триггер был в состоянии 1 и должен сохранить это состояние.

Указанные правила нетрудно свести в единую таблицу (табл. 2.9), которая позволяет наглядно отразить правила вычисления входных значений триггера в зависимости от состояний выходов в предыдущий и в настоящий моменты времени (состояния $Y(t-1)$ и $Y(t)$).

Аналогичную таблице 2.9 таблицу определения входов триггера нетрудно получить из анализа логики работы каждого триггера (см. табл. 2.7 и 2.8). Например, для JK -триггера она имеет вид таблицы 2.10.

Таблица 2.9. Правила определения значений функций переключения RS -триггера

$Y(t-1)$	$Y(t)$	
	0	1
0	0~	10
1	01	~0

Таблица 2.10. Правила определения значений функций переключения JK -триггера

$Y(t-1)$	$Y(t)$	
	0	1
0	0~	1~
1	~1	~0

Выбор триггера для реализации конечного автомата определяется исходя из различных соображений. Например, на выбор может влиять наличие или отсутствие триггеров в элементной базе разработчика или же

различная сложность реализации логического преобразователя с применением различных типов триггеров и т. д.

В табл. 2.11 представлены значения функций включения RS -триггеров и выходов, полученные путем анализа кодированной таблицы переходов (см. табл. 2.6). Данная таблица позволяет перейти к непосредственной реализации комбинационных составляющих конечного автомата – логического и выходного преобразователей.

Таблица 2.11. Значения функций включения RS -триггеров и выходов

$x_1x_2y_1y_2y_3$	Y_{S_1}	Y_{R_1}	Y_{S_2}	Y_{R_2}	Y_{S_3}	Y_{R_3}	z_1	z_2
00000	0	~	0	~	0	~	0	0
00001	0	~	0	~	~	0	0	0
00010	~	~	~	~	~	~	~	~
00011	0	~	~	1	~	1	0	0
00100	0	1	0	~	0	~	0	0
00101	~	~	~	~	~	~	~	~
00110	~	~	~	~	~	~	~	~
00111	~	~	~	~	~	~	~	~
01000	0	~	1	0	0	~	0	1
01001	~	~	~	~	~	~	~	~
01010	0	~	~	0	0	~	0	1
01011	~	~	~	~	~	~	~	~
01100	~	~	~	~	~	~	~	~
01101	~	~	~	~	~	~	~	~
01110	~	~	~	~	~	~	~	~
01111	~	~	~	~	~	~	~	~
10000	0	~	0	~	1	0	1	1
10001	0	~	0	~	~	0	1	1
10010	1	0	0	1	0	~	1	0
10011	0	~	~	0	~	0	1	0
10100	~	0	0	~	0	~	1	0
10101	~	~	~	~	~	~	~	~
10110	~	~	~	~	~	~	~	~
10111	~	~	~	~	~	~	~	~
11000	0	~	0	~	0	~	1	0
11001	0	~	0	~	0	1	1	0
11010	0	~	~	0	1	0	1	0
11011	0	~	~	0	~	0	1	0
11100	~	0	0	~	0	~	1	1
11101	~	~	~	~	~	~	~	~
11110	~	~	~	~	~	~	~	~
11111	~	~	~	~	~	~	~	~

2.3.3. Минимизация функций включения элементов памяти и выходов с использованием карт Карно

Для получения логических выражений, описывающих функции включения элементов памяти и выходов, необходимо проанализировать таблицу истинности (см. табл. 2.8). Минимизируя функции, можно получить формальные выражения, по которым уже синтезируются логический и выходной преобразователи дискретного устройства.

При малом числе входных переменных (как правило, до пяти включительно) удобным способом является ручной способ минимизации, использующий карты Карно [13].

Как известно, карта Карно является одним из способов задания функций алгебры логики. Каждая клетка такой карты соответствует строке таблицы истинности, а клеткам, разделенным общим ребром, ставятся в соответствие соседние конъюнкции, отличающиеся знаком только одной переменной. При этом клетки, соответствующие крайним граням в одной и той же строке, но по разные стороны таблицы, также являются соседними. Более того, для карт Карно от большого числа переменных важным является и расположение контура относительно центральных осей – контуры должны быть симметричны. Указанные особенности карт Карно лежат в основе визуального метода минимизации функций алгебры логики.

Правила минимизации функций алгебры логики по картам Карно следующие [13]:

1. Необходимо в соответствии с таблицей истинности проставить значения функций в клетки карты Карно.
2. Если при минимизации получают дизъюнктивную нормальную форму функции, то целью является объединение в прямоугольные контуры всех клеток, в которых проставлены единицы.
3. Число клеток в таком контуре является степенью числа «два» – 2^k ($k = 0, 1, \dots, n$, где n – число входных переменных), за счет чего возможно исключение k букв в записи конъюнкции, соответствующей каждому контуру.
4. Наличие безразличных состояний позволяет упрощать запись конъюнкции за счет включения в контур, содержащий клетку с единицей, также и попарно соседних клеток с записанными безразличными состояниями.
5. Контуры, расположенные симметрично относительно центральных осей разбиения карты Карно, соответствуют одной конъюнкции.
6. Выбирается минимальное число контуров, содержащих максимальное число клеток. Контуры, все клетки которых принадлежат другим контурам, являются лишними.
7. Возможно наложение контуров друг на друга.

8. Каждому контуру ставится в соответствие конъюнкция, составленная из переменных, значения которых неизменны во всех клетках контура (ранг указанной конъюнкции равен $n - k$).
9. Конъюнкции каждого контура объединяются знаками дизъюнкции.

Располагая нулевые, единичные и безразличные состояния каждой из функций включения элементов памяти и выходов в соответствующих картах Карно, получаем логические выражения, показанные на рис. 2.7 и 2.8.

Сами конечные формулы, описывающие выходы логического и выходного преобразователей, имеют вид:

$$Y_{S_1} = \overline{x_2} y_2 \overline{y_3};$$

$$Y_{R_1} = \overline{x_1};$$

$$Y_{S_2} = \overline{x_1} x_2;$$

$$Y_{R_2} = \overline{x_2} y_3 \vee \overline{x_1} y_3;$$

$$Y_{S_3} = x_1 x_2 y_2 \vee \overline{x_1} \overline{x_2} \overline{y_1} y_2;$$

$$Y_{R_3} = x_2 \overline{y_2} \vee \overline{x_1} y_2;$$

$$z_1 = x_1;$$

$$z_2 = \overline{x_1} x_2 \vee x_2 y_1 \vee \overline{x_1} x_2 \overline{y_1} y_2.$$

По указанным логическим выражениям синтезируются логический и выходной преобразователи.

Отметим, что недостатком карт Карно, наряду с ручным анализом, является если не невозможность, то сложность минимизации систем функций алгебры логики. Минимизация систем функций позволяет сокращать не только каждую из функций, но и в целом конечное устройство за счет выделения одинаковых частей выражений, содержащихся в каждой из функций системы.

2.3.4. Автоматизированная система минимизации функций алгебры логики

В настоящее время многие алгоритмы минимизации как отдельных функций и систем функций, так и самих конечных автоматов по различным критериям (например, по числу состояний) автоматизированы. Более того, автоматизированы и процедуры по синтезу дискретных устройств – пользователю достаточно задать конечный автомат в виде графа переходов, а реализацию его в выбранном базисе даст специальная система автоматизированного проектирования. Так, например, решает задачу проектирования на программируемых логических интегральных схемах известная программа Quartus [18].

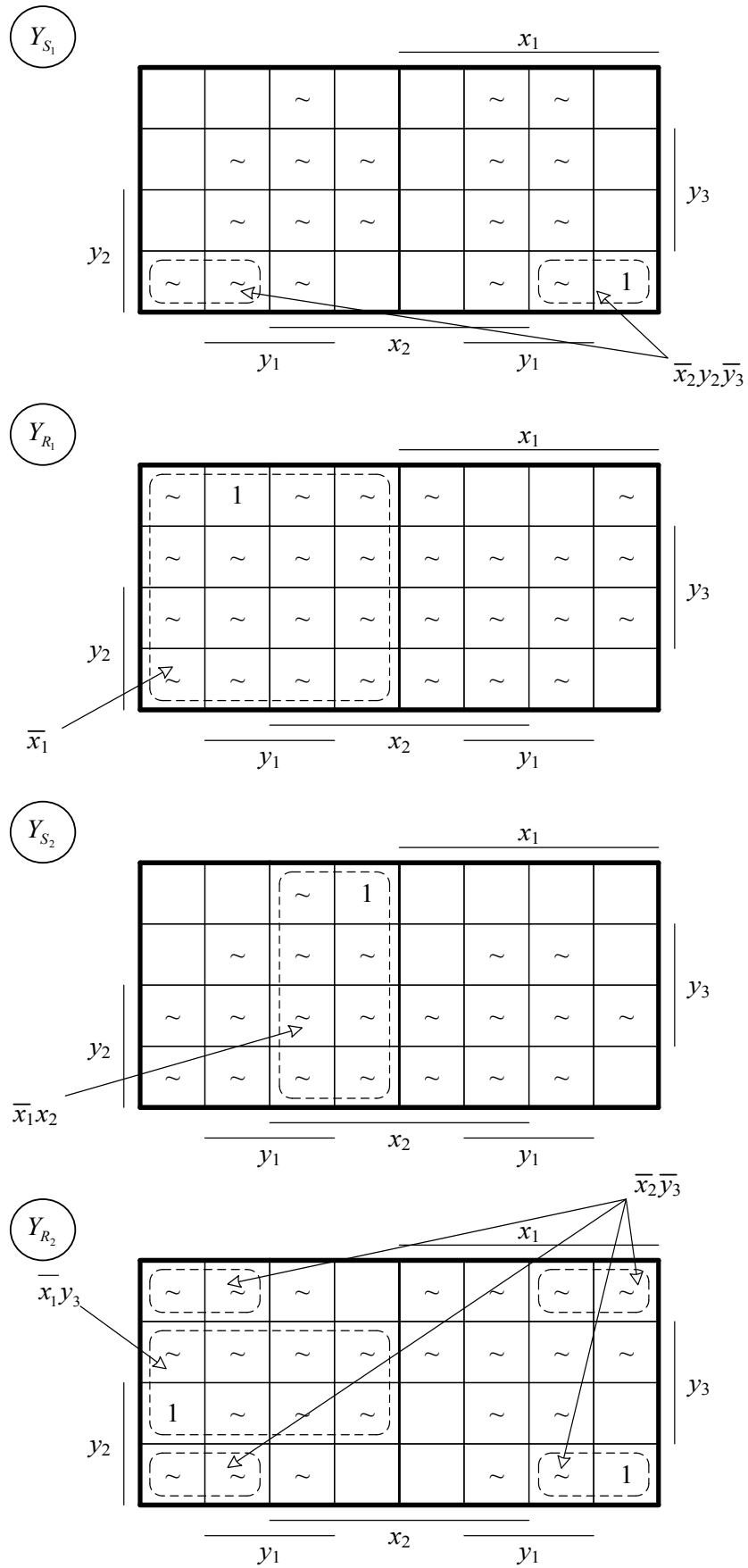


Рис. 2.7. Минимизация функций Y_{S_1} , Y_{R_1} , Y_{S_2} , Y_{R_2}

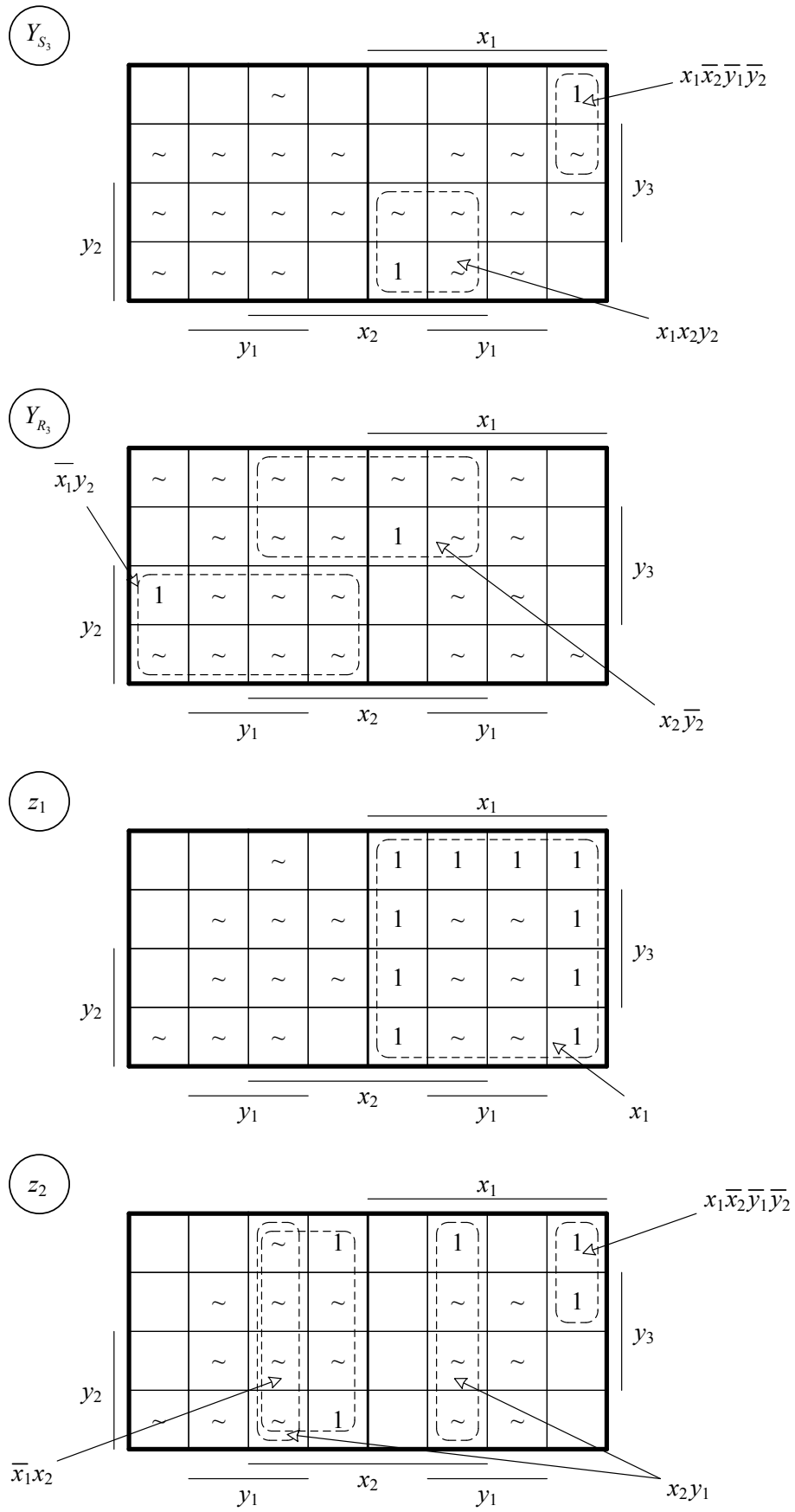


Рис. 2.8. Минимизация функций Y_{S_3} , Y_{R_3} , z_1 , z_2

Рассмотрим простую консольную программу открытого доступа SIS [23, 24]. Окно данной программы с перечнем всех команд представлено на рис. 2.9. SIS позволяет выполнять широкий спектр задач, связанных с логическим проектированием и тестированием. Однако на страницах данного учебного пособия мы рассмотрим только одну ее функцию – возможность автоматизировать процесс минимизации функций и систем функций алгебры логики.

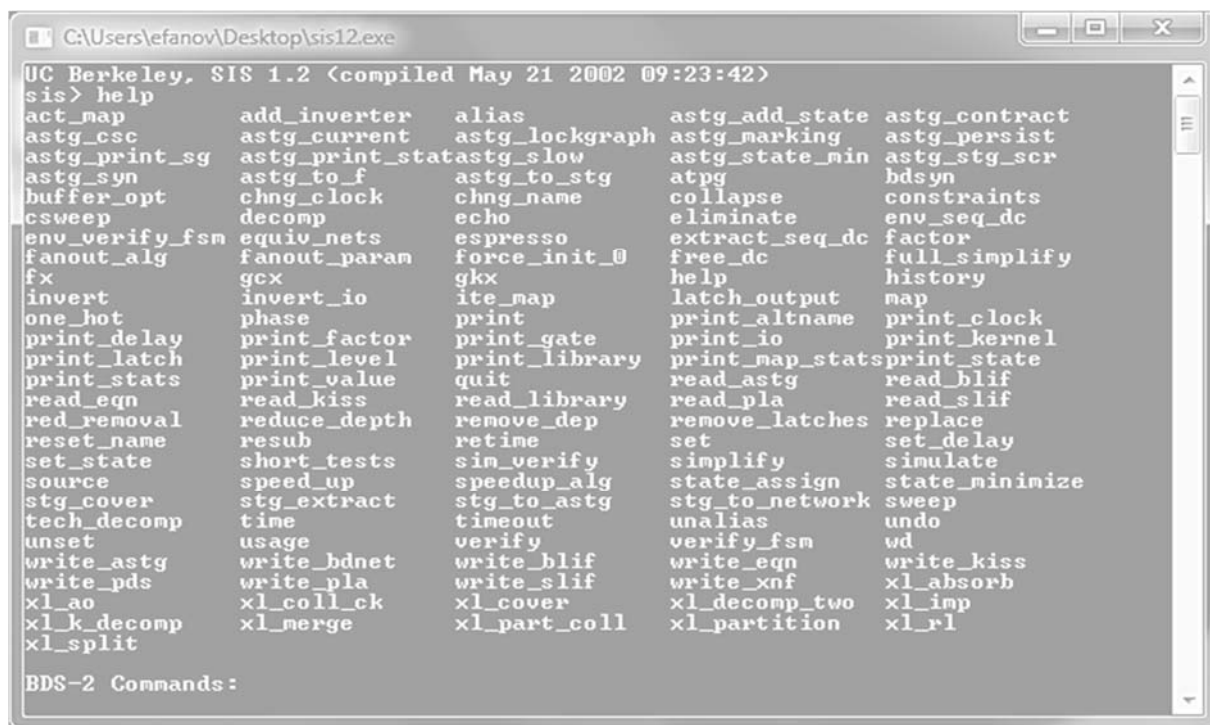


Рис. 2.9. Команды SIS

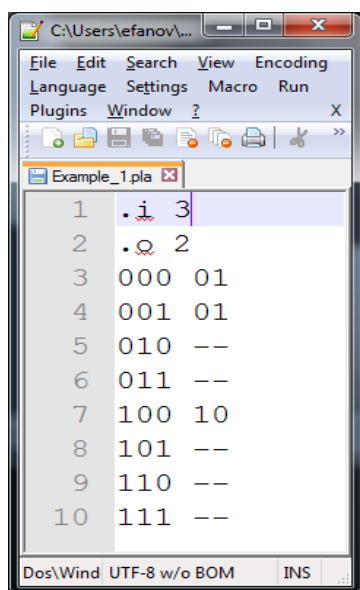


Рис. 2.10. Файл в формате *.pla

Поскольку функции включения элементов памяти, значения которых получают в процессе анализа работы исходного дискретного устройства, в конечном итоге задаются в виде таблицы истинности, при минимизации функций следует использовать табличную форму их задания. Такая форма для логического устройства задается в файле с расширением *.pla. Например, на рис. 2.10 изображено окно приложения «notepad++», в котором раскрыт пример схемы, заданной в формате *.pla. Схема имеет три входа и два выхода, что обозначается в первой и второй строках файла, соответственно: запись «.i 3» указывает на то, что устройство имеет три входа, а запись «.o 2» – то, что оно имеет два выхода. В последующих строках файла перечисляются

все входные комбинации от трех переменных и через пробел в каждой строке указываются соответствующие значения выходов устройства на каждом входном наборе. Символы «0» и «1» обозначают значения логического нуля и логической единицы, а символом «–» обозначается безразличное состояние на данном входном наборе (мы же привыкли к обозначению «~»).

Чтобы использовать для анализа данной схемы программу SIS, требуется загрузить файл в формате *.pla. Для этого в строке записывается команда «read_pla», а затем, через пробел, точное название файла схемы с указанием расширения: «example_1.pla» (при этом строчная это буква или прописная – не имеет значения). Загрузка файла возможна только из того каталога, в котором расположен сам файл приложения sis12.exe.

При корректной записи названия или команды в приложении файл будет прочитан, о чем свидетельствует переход окна команды приложения на новую, пустую строку. При наличии ошибки в команде будет выведена запись «unknown command» с указанием неверного написания команды; при наличии ошибки в названии файла – название файла и запись «no such file or directory».

Для минимизации частично определенных функций используется команда «full_simplify». Чтобы просмотреть полученный результат, необходимо воспользоваться командой «print».

Для записи результата минимизации и получения значения формул, описывающих функции, удобно использовать команду «write_eqn», позволяющую выгружать результат минимизации в дизъюнктивной нормальной форме. Например, на рис. 2.11 показаны перечень команд и их последовательность при решении задачи минимизации, а на рис. 2.12 изображен результат минимизации функций примера, заданного на рис. 2.10. Само дискретное устройство с обозначениями входов и выходов представлено на рис. 2.13.



Рис. 2.11. Файл в формате *.eqn с минимизированными функциями

В первой строке с надписью «INORDER = v0 v1 v2;» приводятся новые наименования входов устройства, а во второй, с надписью «OUTORDER = v3.0 v3.1;», – новые названия выходов. В последующих строках перечисляются логические выражения, описывающие выходы схемы. Логические

операции основного базиса ($\{И; ИЛИ; НЕ\}$) описываются специальными знаками «*», «+» и «!» – соответственно конъюнкция, дизъюнкция и инверсия.

Таким образом, запись « $v3.0 = !v3.1$ » трактуется как $v3.0 = \overline{v3.1}$, а запись « $v3.1 = !v0$ » – как $v3.1 = \overline{v0}$. По аналогии с общепринятыми обозначениями получаем: $z_2 = x_1$, $z_1 = z_2$.

```

1 INORDER = v0 v1 v2;
2 OUTORDER = v3.0 v3.1;
3 v3.0 = !v3.1;
4 v3.1 = !v0;
5
6 Don't care:
7 INORDER = v0 v1 v2;
8 OUTORDER = v3.0 v3.1;
9 v3.0 = !v0*v1*!v2 + v0*v1*!v2 + v0*!v1*v2 + !v0*v1*v2 + v0*v1*v2;
10 v3.1 = !v0*v1*!v2 + v0*v1*!v2 + v0*!v1*v2 + !v0*v1*v2 + v0*v1*v2;

```

Рис. 2.12. Файл в формате *.eqn с минимизированными функциями

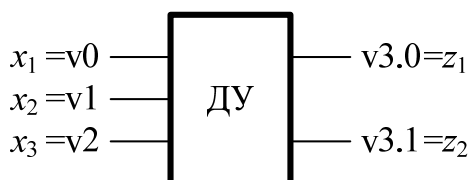


Рис. 2.13. Дискретное устройство

Следует отметить, что аналогичный результат получается и при использовании для минимизации карт Карно (рис. 2.14).

Через строку в файле формата *.eqn (см. рис. 2.12) записаны данные о том, на каких входных наборах каждая функция принимает неопределенные значения.

Таким образом, вместо ручного метода минимизации функций алгебры логики может быть использован автоматизированный метод с применением SIS. Далее в описании процедур по синтезу дискретных устройств с обнаружением неисправностей минимизация будет производиться именно в автоматизированном режиме. При синтезе же дискретного устройства без свойства обнаружения неисправностей рекомендуется использовать карты Карно (для закрепления материала).

Минимизируя с помощью SIS функции включения элементов памяти и выходов, получаем:

$$Y_{S_1} = \overline{y_3} Y_{R_2};$$

$$Y_{R_1} = \overline{x_1};$$

$$Y_{S_2} = \overline{x_1} x_2;$$

$$Y_{R_2} = \overline{x_2} y_2 \overline{y_3} \vee Y_{R_3};$$

$$Y_{S_3} = x_1 \overline{x_2} \overline{y_1} y_2 \vee x_2 y_2 \overline{Y_{S_2}};$$

$$Y_{R_3} = \overline{x_1} y_2 \overline{Y_{S_2}} \vee x_2 \overline{y_2} y_3;$$

$$z_1 = x_1;$$

$$z_2 = \overline{x_1} x_2 \vee x_2 y_1 \vee x_1 \overline{x_2} y_1 y_2.$$

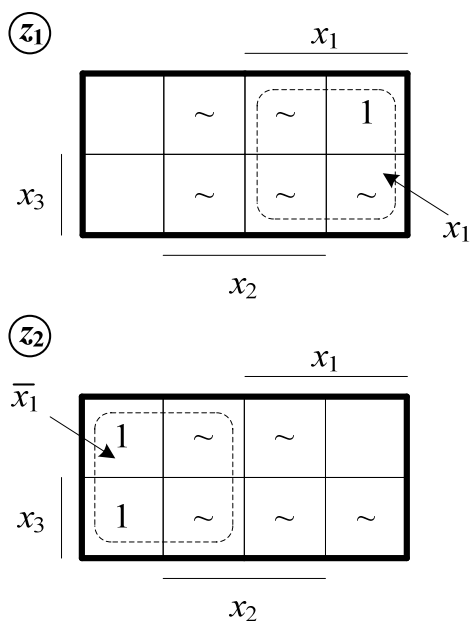


Рис. 2.14. Минимизация функций с применением карт Карно

На рис. 2.15 и 2.16 изображены реализованные по представленным функциям логический и выходной преобразователи.

2.3.5. Схема синхронного конечного автомата

Схема синхронного конечного автомата приведена на рис. 2.17.

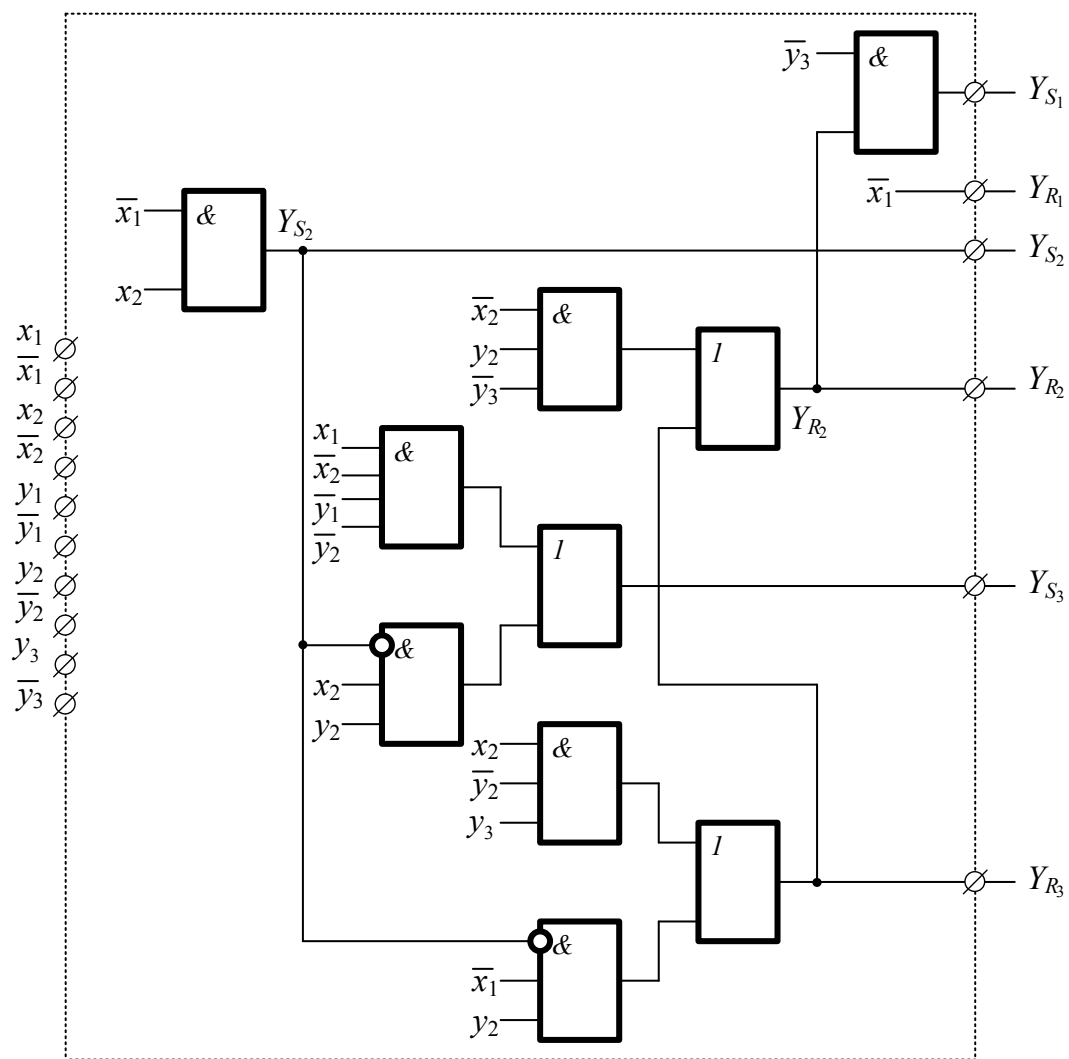


Рис. 2.15. Схема логического преобразователя

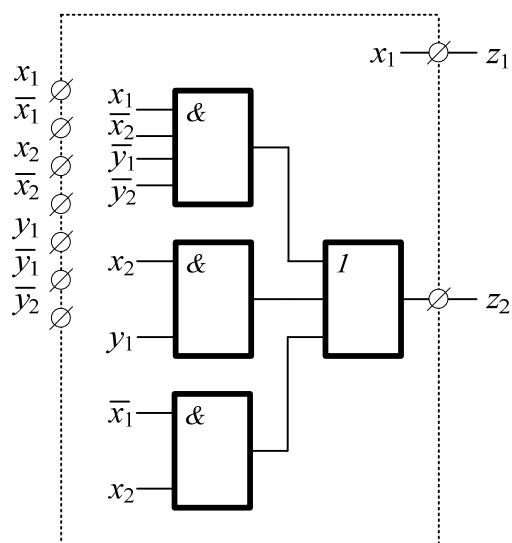


Рис. 2.16. Схема выходного преобразователя

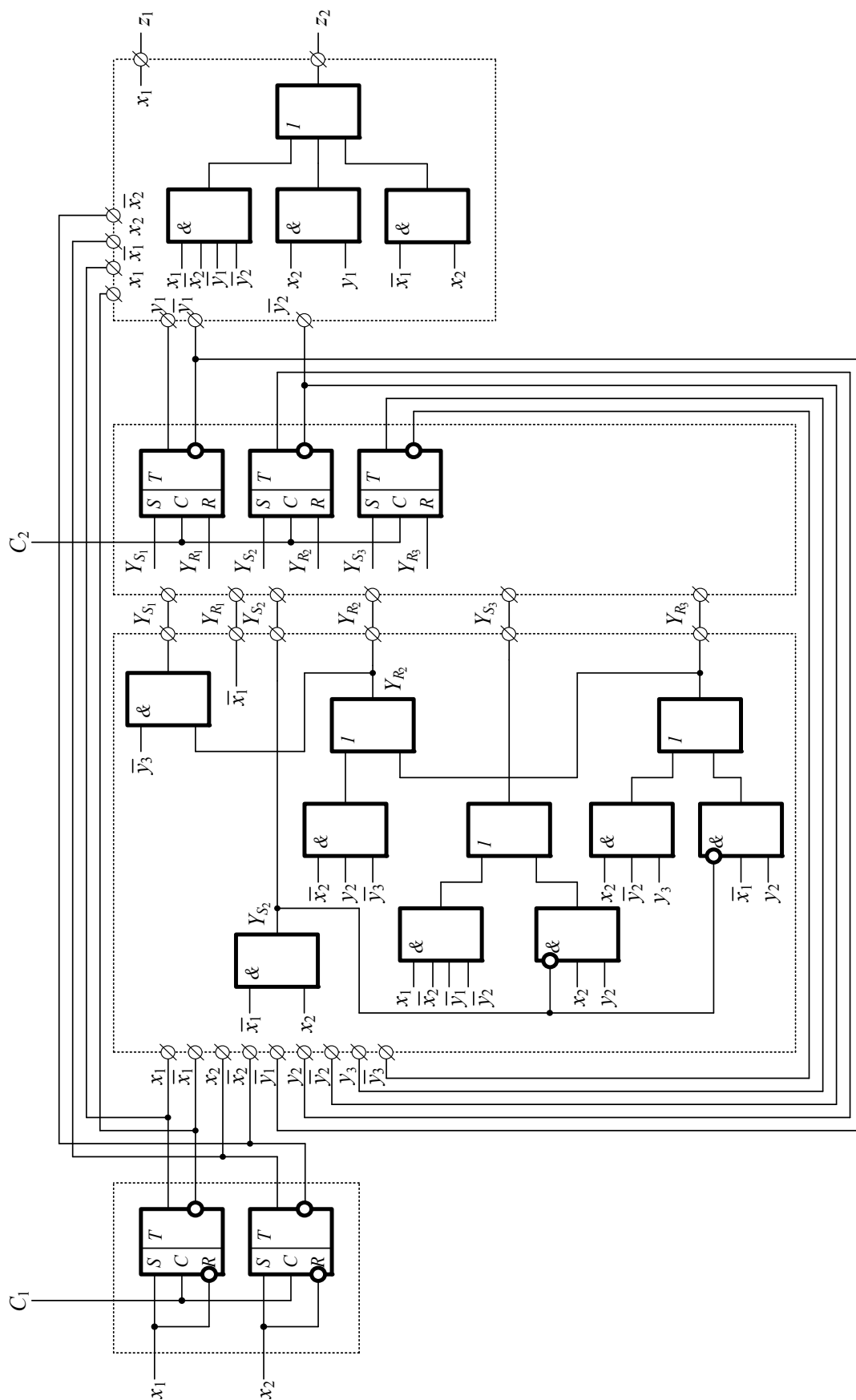


Рис. 2.17. Структурная схема конечного автомата без избыточности

2.4. Синтез синхронных автоматов с обнаружением неисправностей

2.4.1. Принципы обнаружения неисправностей в дискретных устройствах

Для обнаружения неисправностей в логических устройствах требуется использование признака наличия неисправностей. Например, при использовании в качестве рабочих комбинаций значений «логический ноль» и «логическая единица» идентифицировать ошибку ($0 \rightarrow 1$ или $1 \rightarrow 0$) оказывается невозможным, поскольку не ясно, истинно наблюдаемое значение или нет. Для обнаружения ошибок требуется расширение множества используемых комбинаций и дополнение их защитными комбинациями, появление которых будет свидетельствовать о наличии ошибки. В табл. 2.12 показан вариант интерпретации сигналов, где выделены как рабочие, так и защитные комбинации.

Таблица 2.12. Кодирование сигналов для обнаружения ошибок

Тип комбинаций	Традиционный вариант	Кодирование с избыточностью
Рабочие комбинации	0	01
	1	10
Защитные комбинации	3	00
	3	11

Рабочие комбинации 01 и 10 отличаются в обоих разрядах (кодовое расстояние между ними равно двум). Любое одиночное искажение какого-либо разряда рабочей комбинации переводит ее в защитную комбинацию 00 или 11. При этом по наблюдению сформированного сигнала невозможно определить, какая именно рабочая комбинация присутствовала до появления искажения (рис. 2.18).

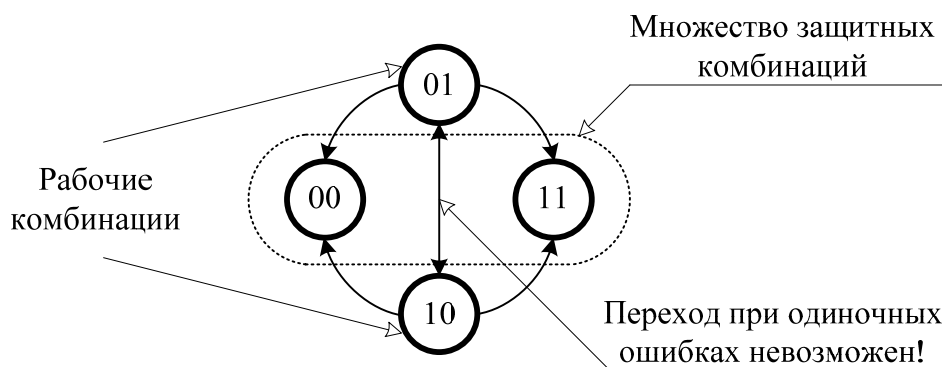


Рис. 2.18. Искращения рабочих комбинаций

Рабочие комбинации представляют собой так называемые парафазные комбинации (комбинации равновесного кода «1 из 2»). Часто говорят, что сигналы закодированы парафазным кодом (two-rail code). Защитные же комбинации соответствуют непарафазным сигналам.

Для наделения проектируемого устройства свойством обнаружения ошибок с большей кратностью требуется большая избыточность. Однако, как правило, устройства строятся с учетом более вероятных одиночных искажений, чем менее вероятных – многократных искажений [2, 8].

С точки зрения решения задачи обнаружения неисправностей важным свойством проектируемого устройства автоматики является самопроверяемость.

Устройство называется *полностью самопроверяемым*, иначе говоря, имеет полностью самопроверяемую структуру в том случае, если оно является самотестируемым и защищенным от внутренних неисправностей.

Под *самотестируемым* устройством понимается устройство, для которого при любой неисправности из заданного класса существует хотя бы один входной набор, на котором искажается значение хотя бы одного его выхода.

Устройство называется *защищенным* от внутренних неисправностей в том случае, если при возникновении любой неисправности из заданного класса на его выходах вычисляются правильные значения сигналов либо неисправность проявляется в виде защитной комбинации этих сигналов.

Под *защитной комбинацией сигналов* понимается запрещенное выходное кодовое слово.

Теории синтеза самопроверяемых дискретных устройств и систем посвящены многие научные публикации, в том числе известные монографии [12, 15, 19].

Рассмотрим задачу синтеза дискретного устройства, наделенного свойством обнаружения неисправностей.

2.4.2. Структурные схемы организации контроля автоматов

Для того чтобы наделить конечный автомат свойством обнаружения неисправностей, необходимо внести в его структуру избыточность по определенным правилам. Эту избыточность можно вносить, например, на этапе кодирования состояний минимизированной таблицы переходов или минимизированного графа переходов (см. табл. 2.3 и рис. 2.6). Поскольку количество состояний устройства равно пяти, то для его построения потребуется три элемента памяти. Если закодировать состояния именно тремя переменными, будет реализовано свойство выполнения заданного алгоритма функционирования, однако при этом не будут учтены возможные возникновения неисправностей в ходе работы устройства.

Реализовать дискретное устройство с обнаружением неисправностей можно за счет использования избыточного кодирования состояний, например, кодом с контролем четности (кодом паритета) или равновесным кодом [1, 22]. На рис. 2.19 изображена организационная структура схемы контроля по признаку четности кодовой комбинации, а на рис. 2.20 – организационная структура схемы контроля по равновесному коду (в данном случае удобным оказывается применение кода «2 из 4», имеющего шесть разрешенных кодовых комбинаций). В обоих случаях число внутренних переменных, таким образом, увеличивается на единицу. Соответственно изменяется число функций включения элементов памяти и их вид по сравнению с кодированием без избытка.

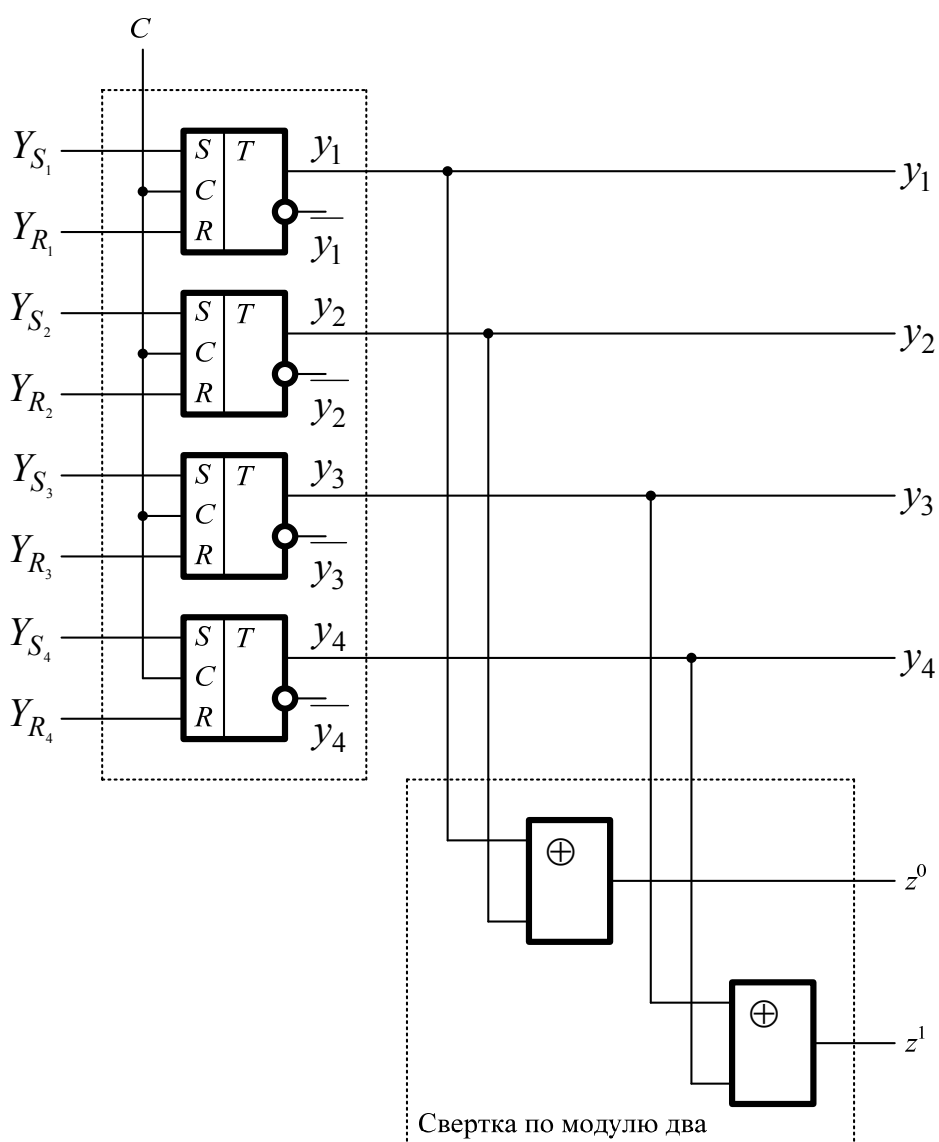


Рис. 2.19. Организационная структура схемы контроля на основе кода с контролем четности

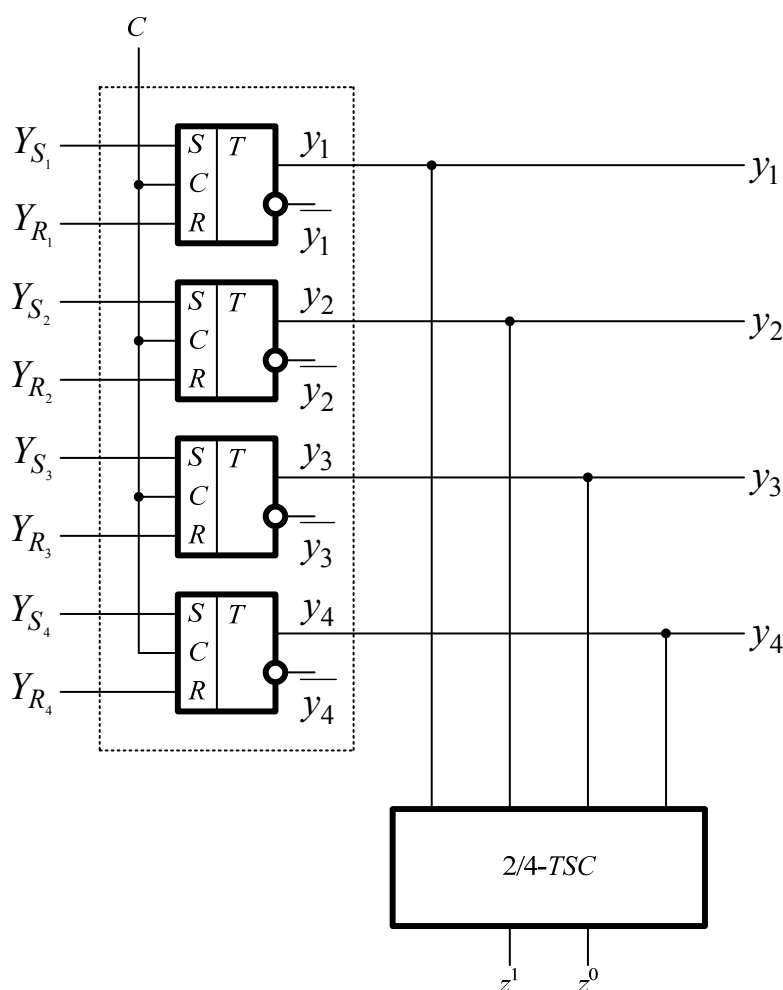


Рис. 2.20. Организационная структура схемы контроля на основе равновесного кода

При построении схемы контроля по коду паритета необходимо контролировать четность формируемой на выходе блока памяти кодовой комбинации $\langle y_1 y_2 y_3 y_4 \rangle$. Это делается путем выделения двух групп выходов (по два выхода в каждой) и установки двух схем свертки по модулю два значений каждой группы выходов. Два выхода полученной схемы представляют собой два контрольных выхода — z^0 и z^1 — синтезируемого устройства. На них при нормальной работе автомата (без отказов) формируется парафазный сигнал $\langle 01 \rangle$ или $\langle 10 \rangle$.

При наличии одиночных неисправностей элементов памяти (в том числе вызванных отказами схем их включения) будет искажаться только одно значение в векторе $\langle y_1 y_2 y_3 y_4 \rangle$. При этом нарушится четность комбинации. Парафазность сигналов z^0 и z^1 будет нарушена, что будет являться сигналом ошибки.

Выделение двух групп при таком способе контроля необходимо для того, чтобы обнаруживались еще и неисправности самой схемы контроля:

при одиночных неисправностях схемы контроля также нарушится парафазность на ее выходах.

Использование равновесного кода при организации контроля блока памяти является альтернативным вариантом по отношению к использованию для этих целей кода паритета. В этом случае каждое состояние конечного автомата кодируется заранее выбранным равновесным кодом. Выбор конкретного кода определяется несколькими факторами, в том числе необходимым для кодирования всех состояний числом разрешенных кодовых комбинаций и сложностью тестера выбранного кода. В рассматриваемом примере для кодирования пяти состояний выбран равновесный код «2 из 4» (как отмечалось выше, такой код имеет шесть разрешенных кодовых слов). В целях контроля принадлежности формируемого на выходах блока памяти кодового вектора $\langle y_1 y_2 y_3 y_4 \rangle$ для кода «2 из 4» устанавливается самопроверяемый тестер $2/4-TSC$, предназначение которого – фиксация неисправностей – при их наличии вырабатывается непарафазный контрольный сигнал. Наиболее простая структура тестера кода «2 из 4» изображена на рис. 2.21 [12].

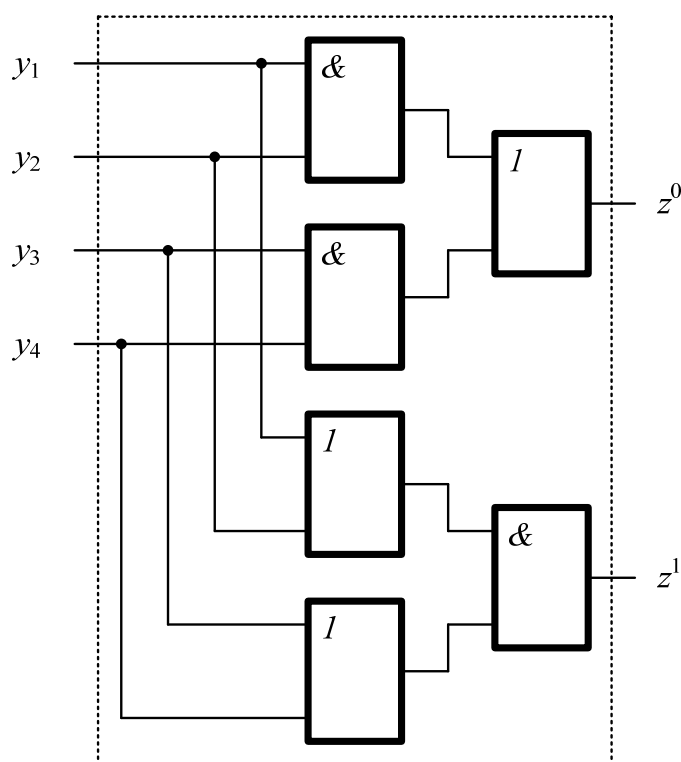


Рис. 2.21. Наиболее простая структурная схема $2/4-TSC$

Использование равновесных кодов при синтезе дискретных устройств с обнаружением неисправностей – весьма распространенный способ реализации дискретных устройств [22]. Применение равновесных кодов эффективно в том числе и потому, что они обнаруживают любые однона-

правленные (монотонные) искажения в кодовых векторах – такие, при которых искажаются либо только нулевые, либо только единичные разряды. На этом свойстве равновесных кодов основаны как методы построения дискретных устройств с обнаружением неисправностей, так и методы построения контролепригодных дискретных устройств. Альтернативой использования равновесных кодов является использование классических кодов с суммированием (кодов Бергера) и их некоторых модификаций [3–6].

2.4.3. Синтез конечных автоматов с обнаружением неисправностей

В табл. 2.13 и 2.14 приведены закодированные состояния и переходы между ними при использовании кода паритета и равновесного кода «2 из 4» соответственно.

Таблица 2.13. Таблица переходов
при кодировании состояний кодом паритета

S^*	Группа			
	I	II	III	IV
	Значения входных переменных			
	x_1x_2			
	00	01	10	11
$s_1=0001$	(0001), 00	0100, 01	0010, 11	(0001), 10
$s_2=0010$	(0010), 00	~	(0010), 11	0001, 10
$s_3=0100$	~	(0100), 01	1011, 10	1000, 10
$s_4=1000$	0001, 00	~	(1000), 10	(1000), 10
$s_5=1011$	0001, 00	~	(1011), 10	(1011), 11

Таблица 2.14. Таблица переходов
при кодировании состояний равновесным кодом

S^*	Группа			
	I	II	III	IV
	Значения входных переменных			
	x_1x_2			
	00	01	10	11
$s_1=0011$	(0011), 00	0110, 01	0101, 11	(0011), 10
$s_2=0101$	(0101), 00	~	(0101), 11	0011, 10
$s_3=0110$	~	(0110), 01	1010, 10	1001, 10
$s_4=1001$	0011, 00	~	(1001), 10	(1001), 10
$s_5=1010$	0011, 00	~	(1010), 10	(1010), 11

Для обеих таблиц (2.13 и 2.14) доопределение аналогично. В качестве примера приведем расширенную кодированную таблицу при использовании для контроля кода паритета (табл. 2.15).

Таблица 2.15. Расширенная таблица переходов при кодировании состояний кодом паритета

S^*	Группа			
	I	II	III	IV
	Значения входных переменных			
	x_1x_2			
	00	01	10	11
$s_1=0001$	(0001), 00	0100, 01	0010, 11	(0001), 10
$s_2=0010$	(0010), 00	~	(0010), 11	0001, 10
$s_3=0100$	~	(0100), 01	1011, 10	1000, 10
$s_4=1000$	0001, 00	~	(1000), 10	(1000), 10
$s_5=1011$	0001, 00	~	(1011), 10	(1011), 11
0000	(0000), ~	(0000), ~	(0000), ~	(0000), ~
0011	(0011), ~	(0011), ~	(0011), ~	(0011), ~
0101	(0101), ~	(0101), ~	(0101), ~	(0101), ~
0110	(0110), ~	(0110), ~	(0110), ~	(0110), ~
0111	(0111), ~	(0111), ~	(0111), ~	(0111), ~
1001	(1001), ~	(1001), ~	(1001), ~	(1001), ~
1010	(1010), ~	(1010), ~	(1010), ~	(1010), ~
1100	(1100), ~	(1100), ~	(1100), ~	(1100), ~
1101	(1101), ~	(1101), ~	(1101), ~	(1101), ~
1110	(1110), ~	(1110), ~	(1110), ~	(1110), ~
1111	(1111), ~	(1111), ~	(1111), ~	(1111), ~

Следующим шагом синтеза автомата с обнаружением неисправностей является получение функций включения элементов памяти (по аналогии с представленными ранее вычислениями). Такая таблица, напрямую полученная из таблицы 2.15 при использовании в качестве элементов памяти RS -триггеров, представлена ниже (табл. 2.16). В ней в каждой клетке записаны восемь значений, соответствующих функциям включения элементов памяти: $Y_{S_1}, Y_{R_1}, Y_{S_2}, Y_{R_2}, Y_{S_3}, Y_{R_3}, Y_{S_4}, Y_{R_4}$.

Описав полученные функции в формате *.pla и применив процедуру их минимизации в программе SIS, получаем результат, представленный на рис. 2.22. При переходе к использованным нами обозначениям ($v_0 = x_1, v_1 = x_2, v_2 = y_1, v_3 = y_2, v_4 = y_3, v_5 = y_4$) получаем следующие выражения для автомата, состояния которого закодированы кодом паритета:

$$v6.0 = Y_{S_1} = Y_{R_2};$$

$$v6.1 = Y_{R_1} = \overline{y_2} Y_{S_4} \vee Y_{R_3};$$

$$v6.2 = Y_{S_2} = x_2 Y_{R_4};$$

$$v6.3 = Y_{R_2} = x_1 \overline{y_1} \overline{y_2} \overline{y_3} \overline{y_4};$$

$$v6.4 = Y_{S_3} = \overline{x_2} Y_{R_4} \vee Y_{R_2} Y_{S_4};$$

$$v6.5 = Y_{R_3} = \overline{x_1} \overline{y_1} \overline{y_2} \overline{y_3} \overline{y_4} \vee \overline{y_3} Y_{S_4};$$

$$v6.6 = Y_{S_4} = \overline{x_1} \overline{y_1} \overline{y_2} \overline{y_3} \overline{y_4} \vee x_2 \overline{y_1} \overline{y_2} \overline{y_3} \overline{y_4} \vee \overline{x_2} Y_{R_2} = \overline{y_2} \overline{y_4} (\overline{x_1} \overline{y_1} \overline{y_3} \vee x_2 \overline{y_1} \overline{y_3}) \vee \overline{x_2} Y_{R_2};$$

$$v6.7 = Y_{R_4} = \overline{x_1} \overline{x_2} \overline{y_1} \overline{y_2} \overline{y_3} \overline{y_4} \vee \overline{x_1} \overline{x_2} \overline{y_1} \overline{y_2} \overline{y_3} \overline{y_4} = \overline{y_1} \overline{y_2} \overline{y_3} \overline{y_4} (x_1 \overline{x_2} \vee \overline{x_1} x_2).$$

Таблица 2.16. Таблица функций включения элементов памяти при использовании кода паритета

S*	Группа			
	I	II	III	IV
	Значения входных переменных			
	x1x2			
	00	01	10	11
s1=0001	0~ 0~ 0~ ~0	0~ 10 0~ 01	0~ 0~ 10 01	0~ 0~ 0~ ~0
s2=0010	0~ 0~ ~0 0~	~	0~ 0~ ~0 0~	0~ 0~ 01 10
s3=0100	~	0~ ~0 0~ 0~	10 01 10 10	10 01 0~ 0~
s4=1000	01 0~ 0~ 10	~	~0 0~ 0~ 0~	~0 0~ 0~ 0~
s5=1011	01 0~ 01 ~0	~	~0 0~ ~0 ~0	~0 0~ ~0 ~0
0000	0~ 0~ 0~ 0~	0~ 0~ 0~ 0~	0~ 0~ 0~ 0~	0~ 0~ 0~ 0~
0011	0~ 0~ ~0 ~0	0~ 0~ ~0 ~0	0~ 0~ ~0 ~0	0~ 0~ ~0 ~0
0101	0~ ~0 0~ ~0	0~ ~0 0~ ~0	0~ ~0 0~ ~0	0~ ~0 0~ ~0
0110	0~ ~0 ~0 0~	0~ ~0 ~0 0~	0~ ~0 ~0 0~	0~ ~0 ~0 0~
0111	0~ ~0 ~0 ~0	0~ ~0 ~0 ~0	0~ ~0 ~0 ~0	0~ ~0 ~0 ~0
1001	~0 0~ 0~ ~0	~0 0~ 0~ ~0	~0 0~ 0~ ~0	~0 0~ 0~ ~0
1010	~0 0~ ~0 0~	~0 0~ ~0 0~	~0 0~ ~0 0~	~0 0~ ~0 0~
1100	~0 ~0 0~ 0~	~0 ~0 0~ 0~	~0 ~0 0~ 0~	~0 ~0 0~ 0~
1101	~0 ~0 0~ ~0	~0 ~0 0~ ~0	~0 ~0 0~ ~0	~0 ~0 0~ ~0
1110	~0 ~0 ~0 0~	~0 ~0 ~0 0~	~0 ~0 ~0 0~	~0 ~0 ~0 0~
1111	~0 ~0 ~0 ~0	~0 ~0 ~0 ~0	~0 ~0 ~0 ~0	~0 ~0 ~0 ~0

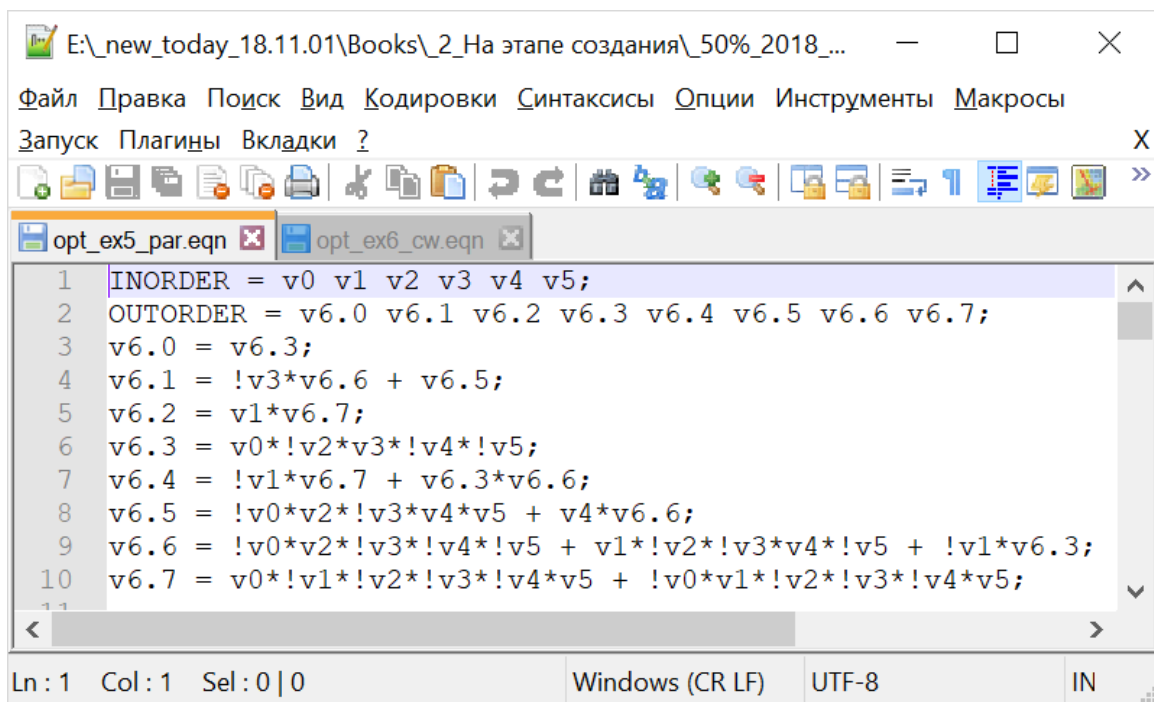


Рис. 2.22. Файл в формате *.eqn с минимизированными функциями включения элементов памяти при контроле автомата на основе кода паритета

На рис. 2.23 изображен логический преобразователь конечного автомата с обнаружением неисправностей при его контроле на основе кода паритета.

Аналогично выполняется процедура получения функций включения элементов памяти автомата при кодировании его состояний равновесным кодом. Получается сначала табл. 2.15, затем 2.16 и выполняется минимизация в SIS (данные шаги читатель может проделать по аналогии самостоятельно). Результат минимизации представлен на рис. 2.24.

При переходе к использованным обозначениям получаем систему функций включения элементов памяти автомата, закодированного равновесным кодом «2 из 4»:

$$v6.0 = Y_{S_1} = \overline{y_4} Y_{R_3};$$

$$v6.1 = Y_{R_1} = \overline{y_2} Y_{S_4} \vee Y_{S_3};$$

$$v6.2 = Y_{S_2} = y_4 Y_{R_3} \vee Y_{R_4};$$

$$v6.3 = Y_{R_2} = x_1 \overline{y_1} y_2 y_3 \overline{y_4} \vee Y_{S_3};$$

$$v6.4 = Y_{S_3} = \overline{x_1} y_1 \overline{y_2} \overline{y_3} y_4 \vee x_2 \overline{y_1} y_2 \overline{y_3} y_4 = \overline{y_3} y_4 (\overline{x_1} y_1 \overline{y_2} \vee x_2 \overline{y_1} y_2);$$

$$v6.5 = Y_{R_3} = x_1 \overline{x_2} \overline{y_1} \overline{y_2} y_3 y_4 \vee \overline{y_2} Y_{S_4} = \overline{y_2} (x_1 \overline{x_2} \overline{y_1} y_3 y_4 \vee Y_{S_4});$$

$$v6.6 = Y_{S_4} = x_1 x_2 \overline{y_1} y_2 y_3 \overline{y_4} \vee \overline{x_1} y_1 \overline{y_2} y_3 \overline{y_4} = y_3 \overline{y_4} (x_1 x_2 \overline{y_1} y_2 \vee \overline{x_1} y_1 \overline{y_2});$$

$$v6.7 = Y_{R_4} = \overline{x_1} x_2 \overline{y_1} \overline{y_2} y_3 y_4.$$

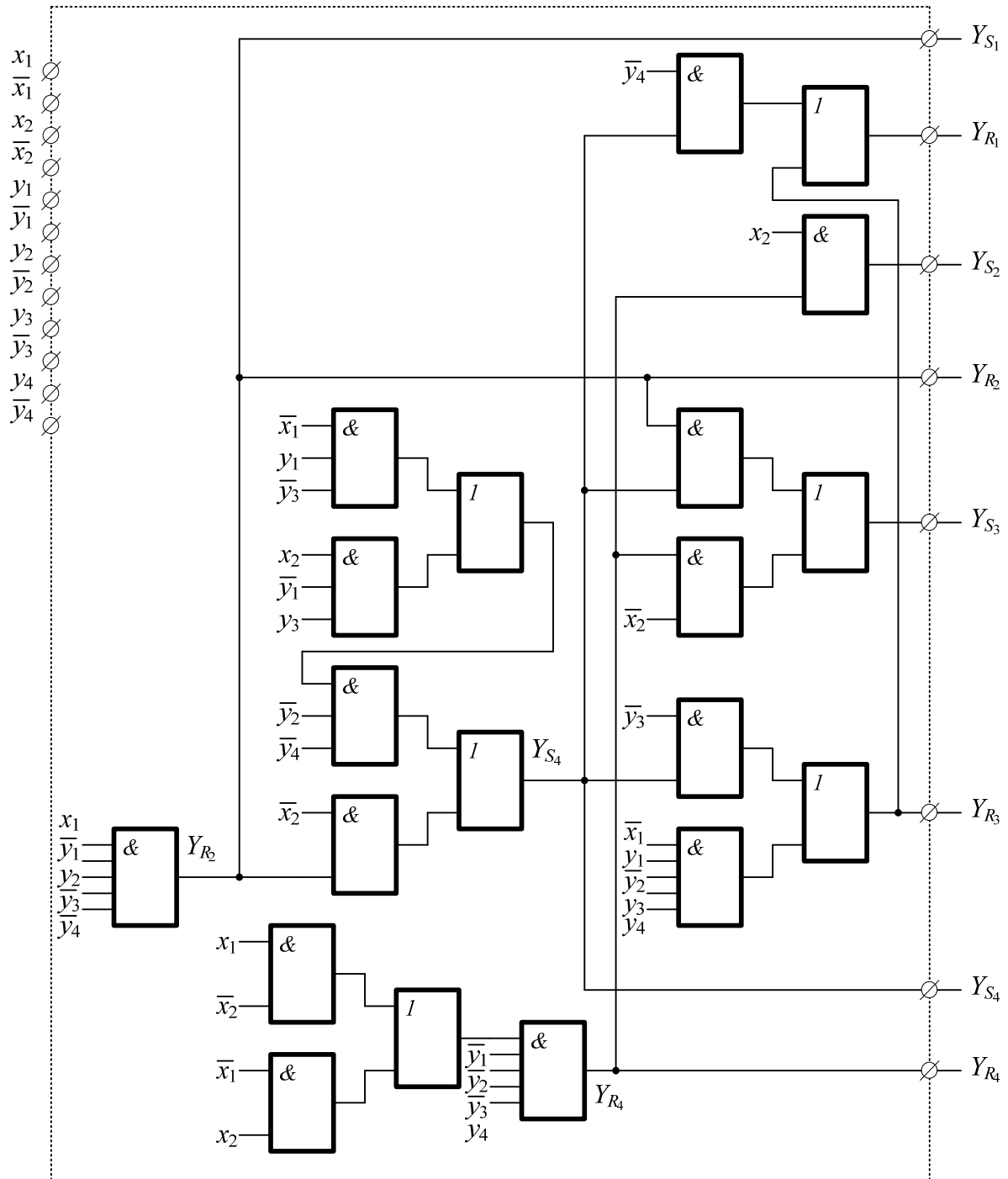


Рис. 2.23. Схема логического преобразователя конечного автомата с обнаружением неисправностей на основе кода паритета

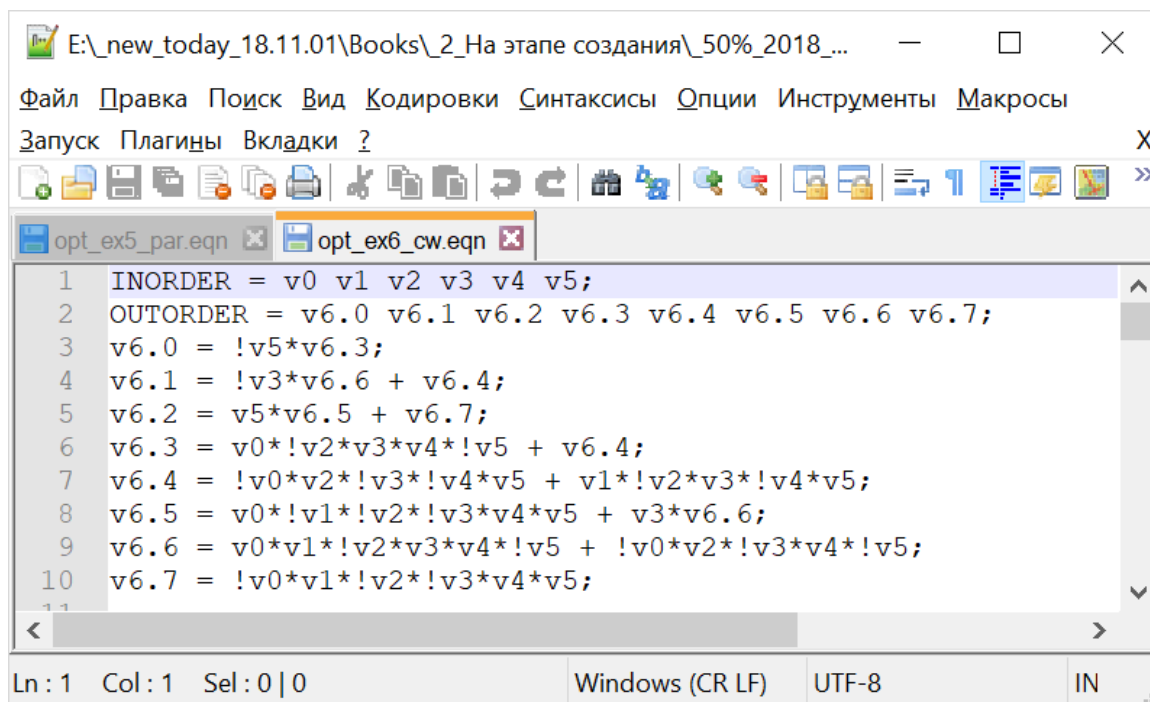


Рис. 2.24. Файл в формате *.eqn с минимизированными функциями включения элементов памяти при контроле автомата на основе кода «2 из 4»

На рис. 2.25 изображен логический преобразователь конечного автомата с обнаружением неисправностей при его контроле на основе равновесного кода «2 из 4». Если оценивать сложность схем по числу входов логических элементов, то можно отметить более простую реализацию схемы на рис. 2.23 (40 входов) в сравнении со схемой на рис. 2.25 (47 входов).

Схемы на рис. 2.23 и рис. 2.25 не имеют в своих структурах ни одной инверсии (все инверсии реализованы на входах каждой из схем). Эта особенность реализации устройства позволяет при любой неисправности элементов его внутренней структуры вызывать только монотонную ошибку на выходах [6]. Поскольку только для второй схемы использован код с обнаружением любых монотонных ошибок, в ней будут обнаруживаться любые одиночные неисправности, тогда как в первой схеме существует некоторая вероятность необнаружения ошибки (часть логических элементов внутренней структуры связана четным количеством путей с выходами самого логического преобразователя).

С использованием SIS получают также логические выражения, описывающие функции z_1 и z_2 – функции выходного преобразователя (рис. 2.26 и 2.27).

Для случая кодирования состояний кодом паритета имеем

$$z_1 = x_1;$$

$$z_2 = \overline{x_1 x_2 y_1 y_2} \vee x_2 y_1 y_4 \vee \overline{x_1 y_1}.$$

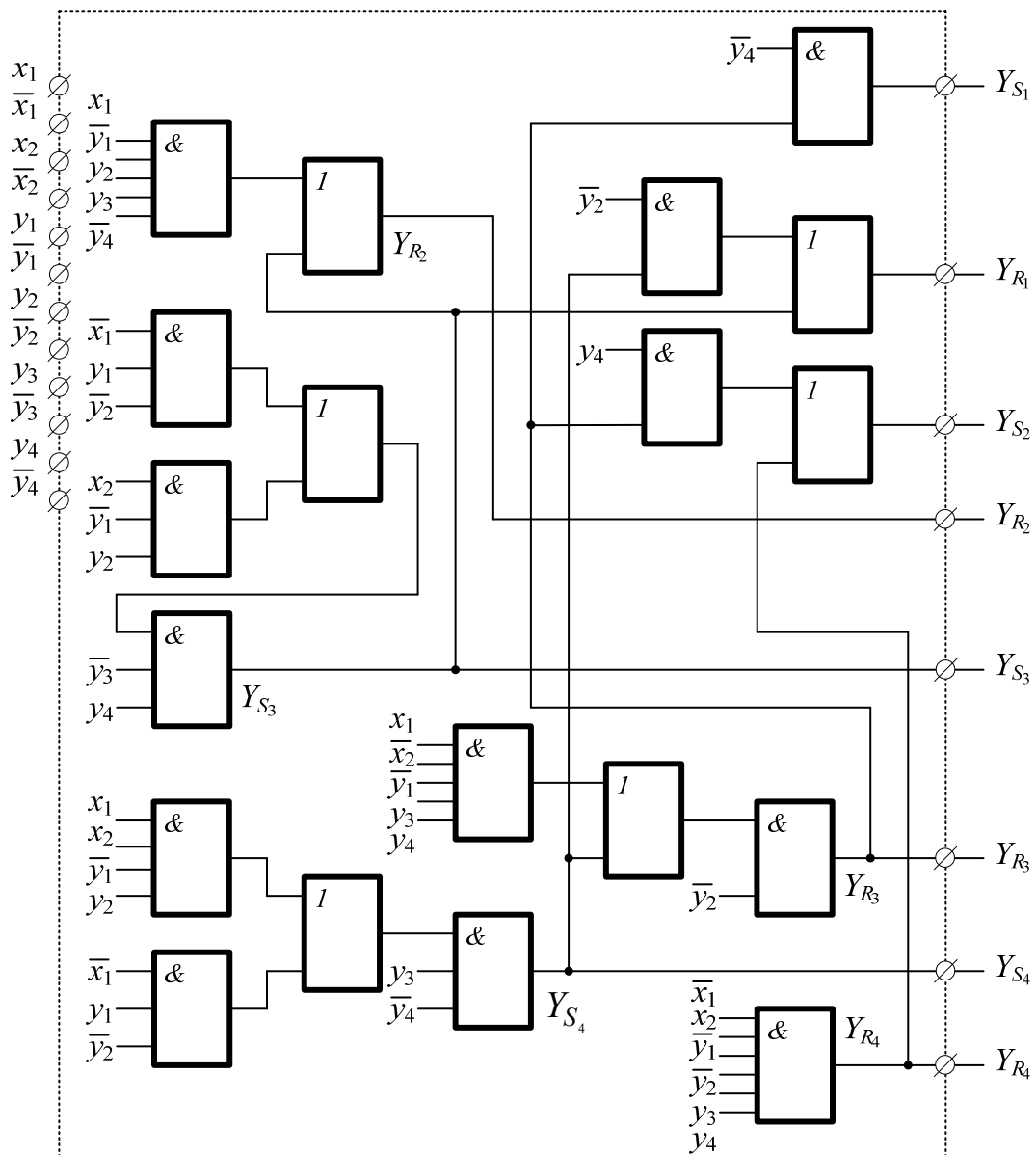


Рис. 2.25. Схема логического преобразователя конечного автомата с обнаружением неисправностей на основе кода «2 из 4»

Техническая реализация выходного преобразователя при кодировании состояний кодом паритета показана на рис. 2.28, а.

При использовании равновесного кода «2 из 4» функции выходов принимают вид

$$z_1 = x_1;$$

$$z_2 = \overline{x_1} \overline{x_2} \overline{y_1} y_4 \vee x_2 y_1 y_4 \vee x_1 y_1.$$

Техническая реализация выходного преобразователя при кодировании состояний кодом «2 из 4» показана на рис. 2.28, б.

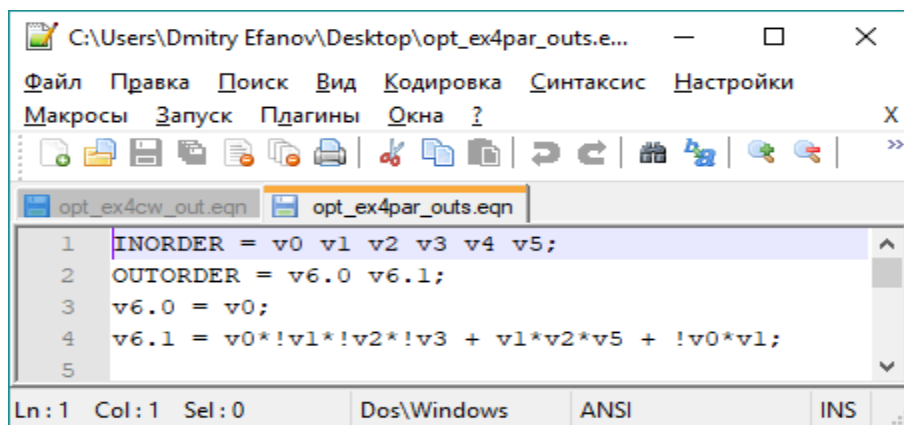


Рис. 2.26. Минимизированные функции выходов при кодировании состояний кодом паритета

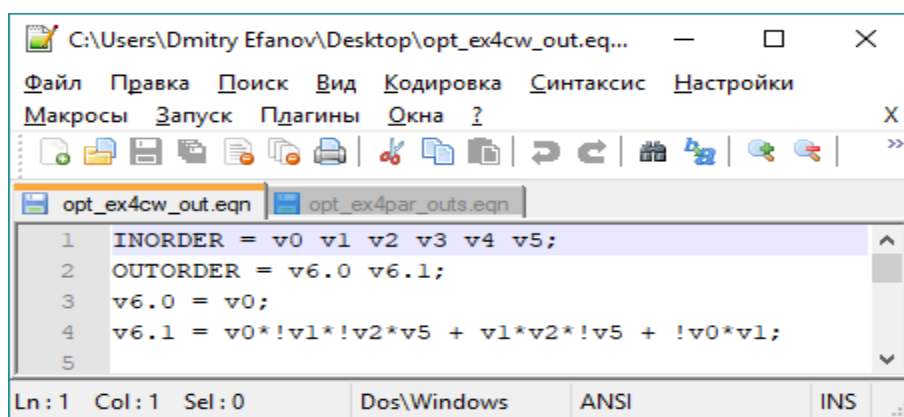


Рис. 2.27. Минимизированные функции выходов при кодировании состояний кодом «2 из 4»

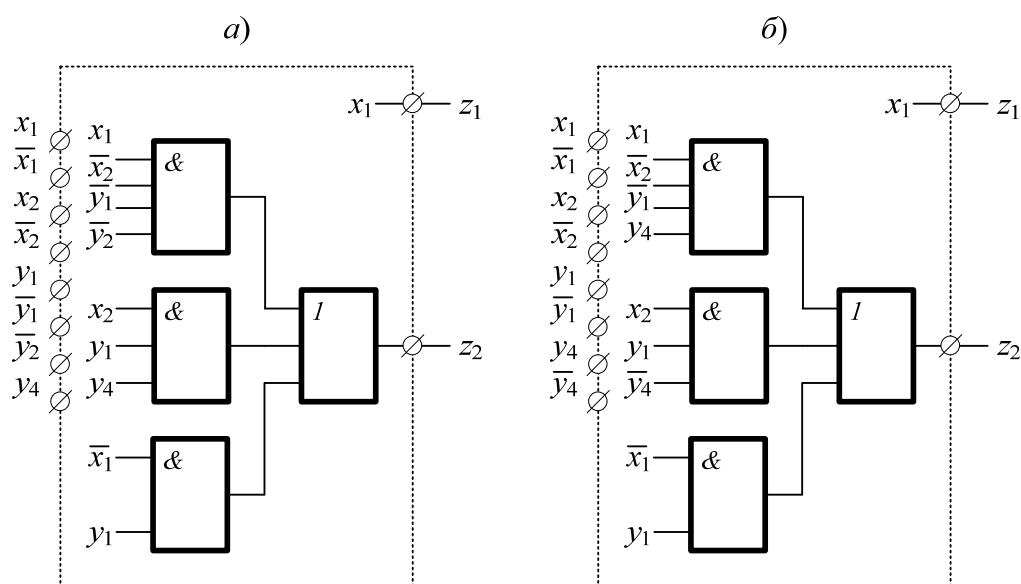


Рис. 2.28. Выходной преобразователь устройства:
а – при кодировании его состояний кодом паритета;
б – при кодировании его состояний кодом «2 из 4»

2.4.4. Синтез схем встроенного контроля

Реализовать конечный автомат с обнаружением неисправностей можно и другим способом – без внесения изменений в структуру автомата по сравнению с исходным вариантом, не фиксирующим неисправности (см. рис. 2.17). Эту задачу можно решить путем синтеза схем встроенного контроля.

Наиболее простой вариант организации схемы контроля – контроль вычислений на основе кода паритета [10]. Блок $F(x)$ является логическим преобразователем конечного автомата (см. рис. 2.17). С целью организации контроля неисправностей косвенно, по результатам вычислений, логический преобразователь снабжается схемой контроля в составе генератора кода паритета и блока контрольной логики. Блок контрольной логики $G(x)$ имеет всего один выход и вычисляет функцию паритета:

$$g = f_1 \oplus f_2 \oplus \dots \oplus f_m.$$

Таким образом, схема контроля по паритету (рис. 2.29) основана на использовании наименее избыточного кода – кода с контролем на четность (нечетность), или кода паритета ($P(m,1)$ -кода). Данный код имеет всего один контрольный разряд, принимающий единичное значение в том случае, если число единиц в информационном векторе нечетно, и значение нуля – в том случае, если число единиц в информационном векторе четно.

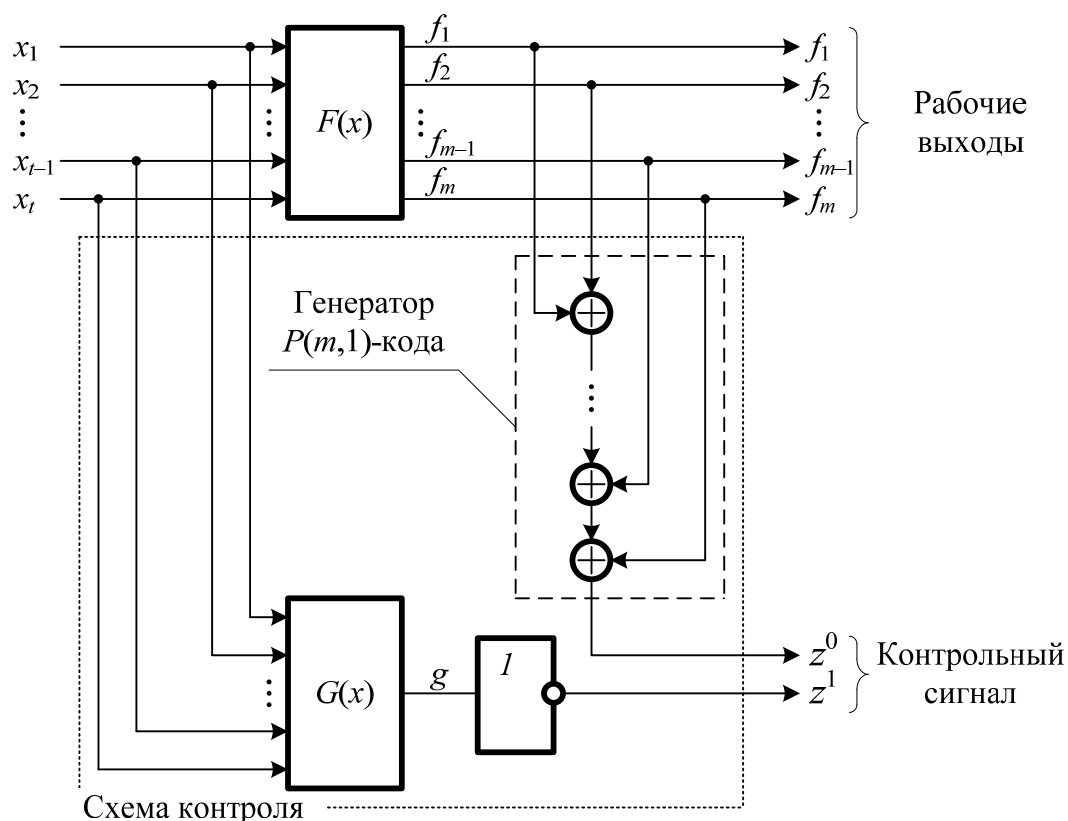


Рис. 2.29. Структурная схема системы контроля по паритету

Синтез схемы контроля по паритету состоит в следующем. Для выходов логического преобразователя (Y_{S_1} , Y_{R_1} , Y_{S_2} , Y_{R_2} , Y_{S_3} , Y_{R_3}) вычисляется функция паритета. Составляется таблица истинности, в которой рассчитываются значения функций включения элементов памяти на исходных двоичных наборах $\langle x_1 x_2 y_1 y_2 y_3 \rangle$. Это аналог табл. 2.11, в котором однозначно доопределены безразличные значения (они доопределяются на основании минимизации в соответствии с рис. 2.7 и рис. 2.8 или по полученным формулам, описывающим функции включения элементов памяти). Получается таблица 2.17.

Таблица 2.17. Значения функций включения RS-триггеров и выходов

$x_1 x_2 y_1 y_2 y_3$	Y_{S_1}	Y_{R_1}	Y_{S_2}	Y_{R_2}	Y_{S_3}	Y_{R_3}	g
00000	0	1	0	0	0	0	1
00001	0	1	0	0	0	0	1
00010	1	1	0	1	0	1	0
00011	0	1	0	1	0	1	1
00100	0	1	0	0	0	0	1
00101	0	1	0	0	0	0	1
00110	1	1	0	1	0	1	0
00111	0	1	0	1	0	1	1
01000	0	1	1	0	0	0	0
01001	0	1	1	1	0	1	0
01010	0	1	1	0	0	0	0
01011	0	1	1	0	0	0	0
01100	0	1	1	0	0	0	0
01101	0	1	1	1	0	1	0
01110	0	1	1	0	0	0	0
01111	0	1	1	0	0	0	0
10000	0	0	0	0	1	0	1
10001	0	0	0	0	1	0	1
10010	1	0	0	1	0	0	0
10011	0	0	0	0	0	0	0
10100	0	0	0	0	0	0	0
10101	0	0	0	0	0	0	0
10110	1	0	0	1	0	0	0
10111	0	0	0	0	0	0	0
11000	0	0	0	0	0	0	0
11001	0	0	0	1	0	1	0
11010	0	0	0	0	1	0	1

$x_1x_2y_1y_2y_3$	Y_{S_1}	Y_{R_1}	Y_{S_2}	Y_{R_2}	Y_{S_3}	Y_{R_3}	g
11011	0	0	0	0	1	0	1
11100	0	0	0	0	0	0	0
11101	0	0	0	1	0	1	0
11110	0	0	0	0	1	0	1
11111	0	0	0	0	1	0	1

Функция g минимизируется. По полученному выражению синтезируется блок $G(x)$. Остальные элементы в схеме контроля являются стандартными. Минимизируя функцию g любым из описанных выше методов, получаем:

$$g = \overline{x_1} \overline{x_2} y_3 \vee \overline{x_1} x_2 \overline{y_2} \vee x_2 \overline{y_1} \overline{y_2} \vee x_1 x_2 y_2.$$

Иные способы синтеза схем контроля описаны в [10]. В том числе подробно описаны способы применения помехозащитных блочных равномерных кодов для решения задачи организации диагностического обеспечения комбинационных составляющих конечных автоматов.

3. Анализ работы автомата в среде моделирования Multisim

3.1. Работа в среде моделирования Multisim

Для симуляции работы логической схемы можно использовать программный комплекс Multisim. Данное программное обеспечение позволяет симулировать работу различных электрических и логических схем.

Вначале разберем интерфейс. На рис. 3.1 представлено начальное окно Multisim. В центре экрана расположено поле, в котором находятся элементы схемы. При помощи колеса мыши можно осуществлять масштабирование данного поля. Также, вращением колеса мыши при зажатой клавише Ctrl, осуществляется прокрутка по вертикали, а при одновременно зажатых клавишах Ctrl и Shift – прокрутка по горизонтали. В центральной верхней части окна расположены кнопки запуска симуляции, постановки симуляции на паузу и остановки симуляции. Запуск и постановку на паузу также можно осуществлять по нажатиям на клавиши F5 и F6 соответственно. В левой верхней части окна находится панель с различными элементами схем. В работе рекомендуется использовать источники питания (Source) и микросхемы ТТЛ (TTL).

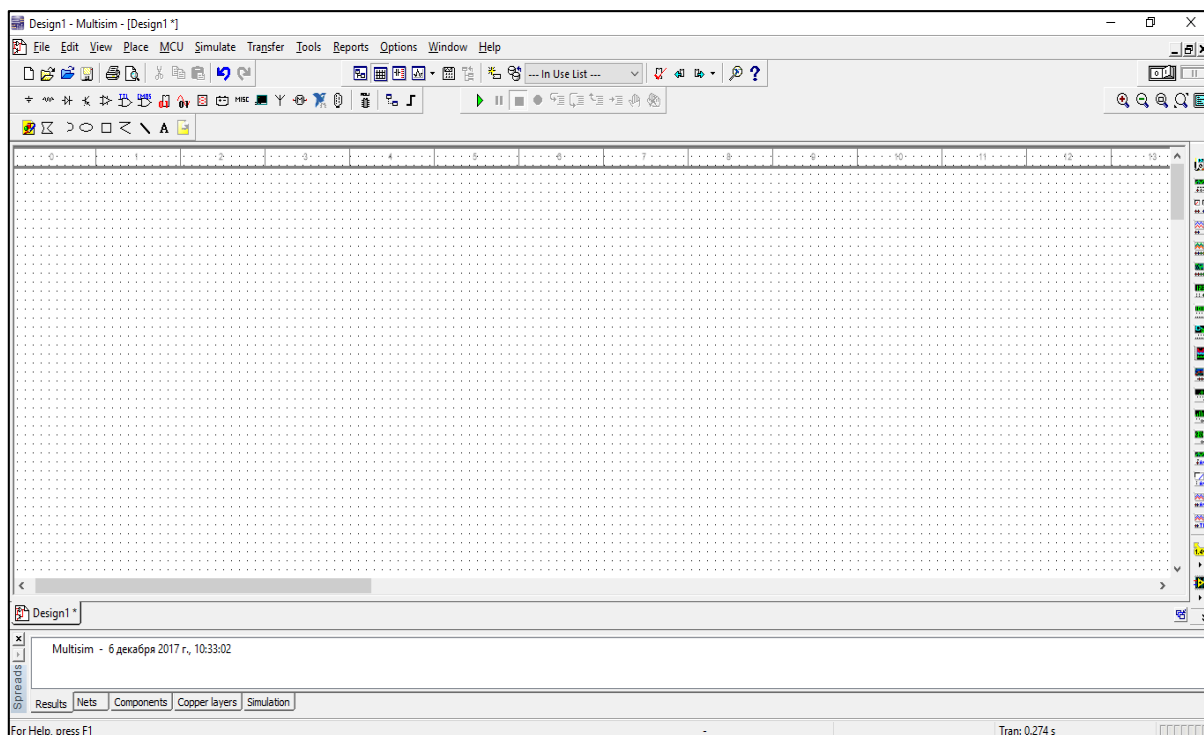


Рис. 3.1. Окно Multisim

Окно на рис. 3.1 можно настроить. Для этого надо щелкнуть правой кнопкой мыши на свободном пространстве в поле расположения объектов схем и в появившемся списке выбрать *Properties*. Откроется окно с несколькими вкладками. Во вкладке *Sheet visibility* можно настроить отображение различных надписей, таких как названия элементов, значения, присвоенные им, и т. д. Для удобства восприятия схемы рекомендуется убирать все надписи, ненужные при построении моделируемой схемы. Во вкладке *Colors* можно поменять цвета поля, элементов схемы, расположенной на нем, и надписей. Цвет в данном случае не несет большой информации, поэтому рекомендуется использовать предустановленную цветовую схему *White & black*. Возможно использование цветовых схем с темным фоном. Во вкладке *Workspace* можно настроить размеры поля, на котором располагается схема. Поскольку до построения схемы сложно спрогнозировать размеры занимаемого ею пространства, рекомендуется выбирать максимально возможный размер поля.

В начале построения схемы следует разместить на поле источники сигналов. Для этого необходимо нажать кнопку *Place Sources*. Откроется окно с различными источниками сигналов и питания (рис. 3.2). Простейшие источники логических сигналов находятся во вкладке *DIGITAL_SOURCES*. Во вкладке *POWER_SOURCES* находятся источники электрических сигналов и заземлители, которые могут быть использованы, в том числе, как источники логических сигналов.

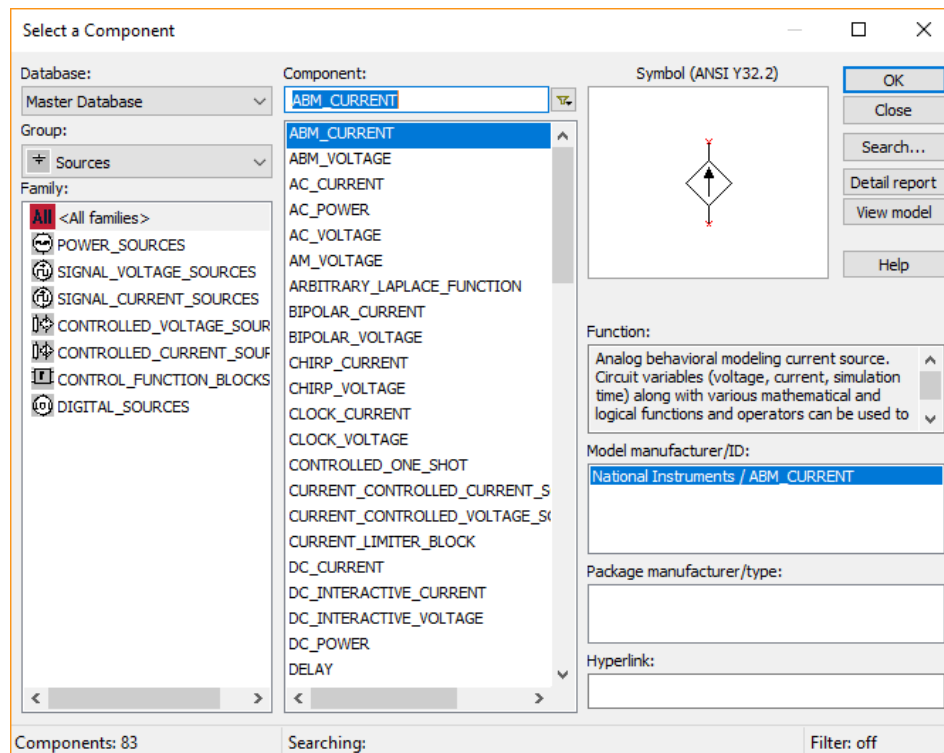


Рис. 3.2. Окно источников сигналов

Во вкладке `DIGITAL_SOURCES` находятся три вида источников логических сигналов (рис. 3.3). На рисунке `U1` – это генератор тактовых импульсов (`DIGITAL_CLOCK`). Под ним указана частота генерируемых импульсов. При двойном щелчке по генератору открывается окно настроек данного элемента с несколькими вкладками. Наибольший интерес представляют вкладки `Label` и `Value`. Во вкладке `Label`, в строке `RefDes`, можно поменять название данного элемента. Во вкладке `Value` можно задать частоту импульсов в строке `Frequency` и их сдвиг по фазе в строке `Delay time`. `U2` – это источник логической константы (`DIGITAL_CONSTANT`). Под ним написано, какой логический сигнал генерирует данный элемент. При двойном нажатии на генератор логической константы открывается окно параметров, аналогичное окну параметров генератора тактовых импульсов с одним отличием, а именно: во вкладке `Value` выбирается генерируемый логический сигнал. `U3` – это источник переменного логического сигнала (`INTERACTIVE_DIGITAL_CONSTANT`). В центре этого источника написано, какой сигнал в данный момент генерируется. Сигнал можно поменять, щелкнув один раз по генератору или нажав привязанную к нему клавишу, которая указана под ним. При двойном нажатии также открывается окно параметров, аналогичное окну параметров предыдущих двух генераторов, за исключением вкладки `Value`. В этой вкладке можно выбрать, к какой клавише привязать данный источник сигнала.

Кроме источников, необходимы также индикаторы сигналов. Они расположены во вкладке Place indicator. Для проверки наличия логического сигнала достаточно обычных лампочек, расположенных во вкладке PROBE (рис. 3.4). При двойном нажатии открывается окно параметров данного элемента, в котором можно поменять его название.

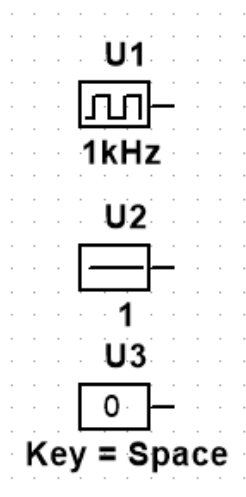


Рис. 3.3. Источники логических сигналов

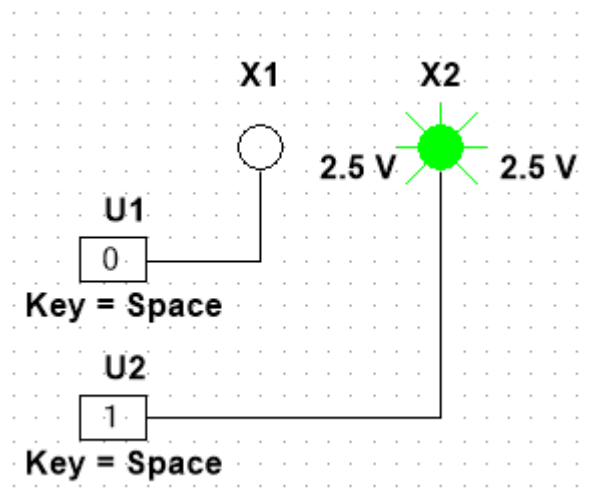


Рис. 3.4. Горящая и негорящая зеленые лампочки

Для моделирования различных логических схем необходимы различные логические элементы. Их можно найти, нажав на кнопку TTL. Откроется окно (рис. 3.5), в котором будут располагаться логические элементы, различные по реализуемым функциям и по семействам. При построении схем рекомендуется пользоваться элементами одного семейства, так как элементы различных семейств могут иметь различные электрические параметры. В данном случае удобно использовать элементы семейства 74LS. Для поиска необходимого элемента следует нажать на кнопку Search и в открывшемся окне, в строке Function, ввести реализуемую функцию элемента, а в строке Family выбрать необходимое семейство элементов. Ввод функции производится на английском языке. Наименования некоторых логических элементов на английском языке приведены в табл. 3.1. После ввода необходимых параметров искомых элементов следует нажать кнопку Search – появится окно со списком элементов, удовлетворяющих условиям поиска (рис. 3.6). Поиск осуществляется по названию функции, поэтому, кроме искомых элементов, в данном списке могут быть и другие. Например, при поиске элемента ИЛИ (OR) выводятся также элементы ИЛИ-НЕ (NOR).

Окно со списком элементов, удовлетворяющих условиям поиска, содержит несколько разделов. В данном случае нас интересуют три из них – это раздел Components, раздел Function и раздел Symbol. Первый раздел содержит список найденных элементов. В разделе Function записаны rea-

лизуемые функции выбранного из списка элемента. Данный раздел может содержать также информацию о количестве входов и выходов или о количестве элементов на одном кристалле. Например, запись QUAD 2-INPUT NOR означает, что один кристалл содержит четыре элемента ИЛИ-НЕ, и каждый из них с двумя входами. В разделе Symbol отображается обозначение на схеме выбранного элемента.

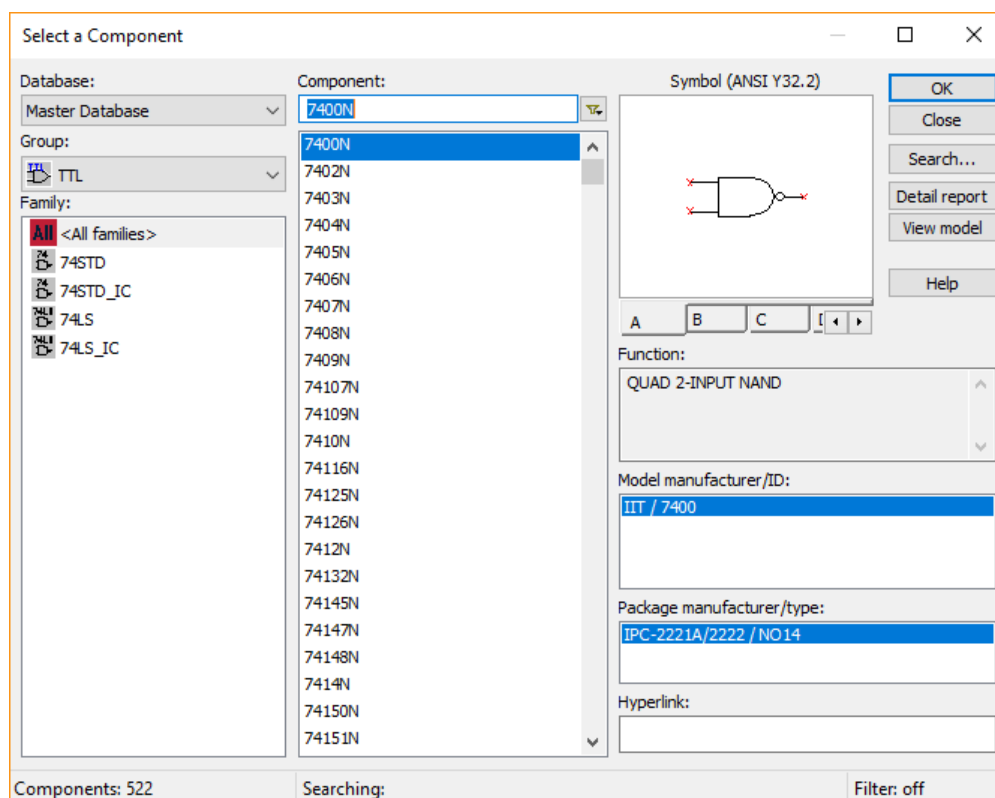


Рис. 3.5. Окно логических элементов TTL

Таблица 3.1. Названия логических элементов на английском языке

Название элемента на русском языке	Название элемента на английском языке
И	And
ИЛИ	Or
И-НЕ	Nand
ИЛИ-НЕ	Nor
НЕ	Inverter
Мультиплексор	Mux
Кодер	Coder
Декодер	Decoder

После выбора необходимого элемента нужно нажать кнопку ОК. Некоторые схемы могут содержать несколько элементов. В этом случае, после нажатия ОК, высвечивается окно выбора элемента (рис. 3.7). В данном окне отображается название микросхем и названия элементов в них. Неактивные

кнопки означают, что данный элемент уже используется на схеме, а активные кнопки означают, что данный элемент свободен и может быть использован.

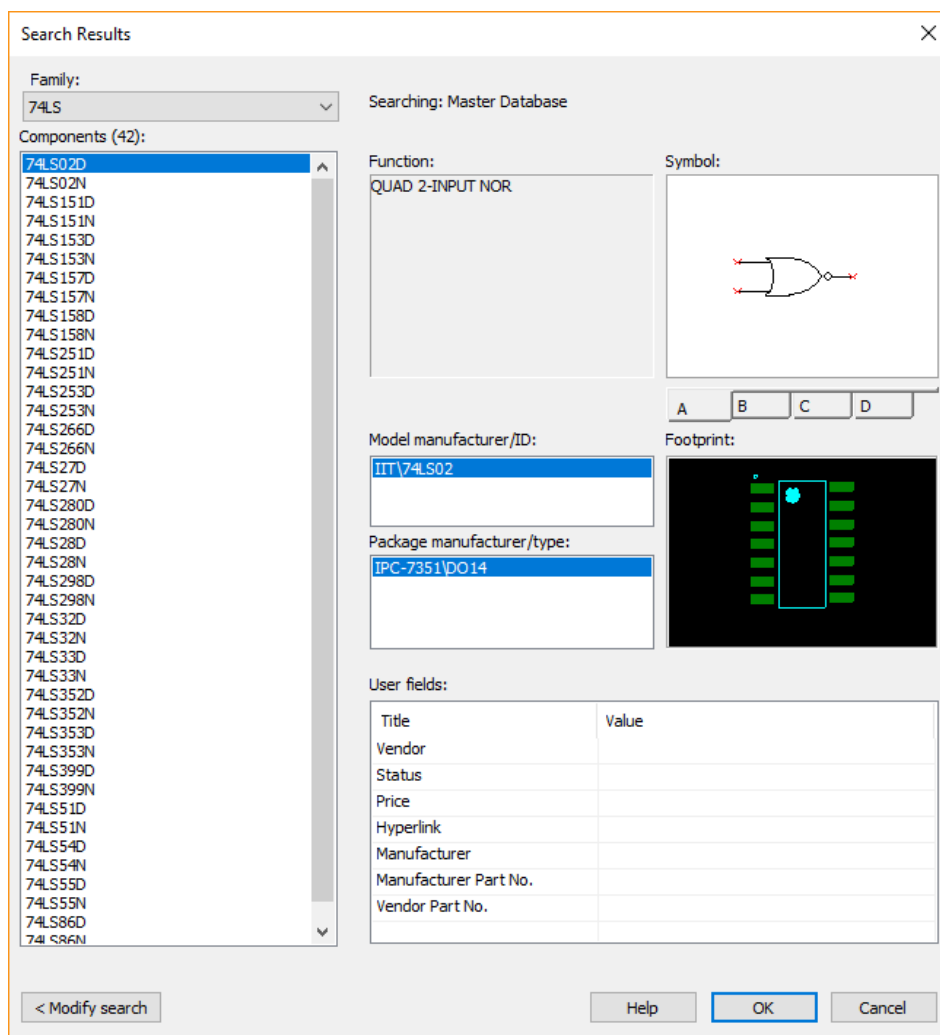


Рис. 3.6. Окно со списком элементов, удовлетворяющих условиям поиска

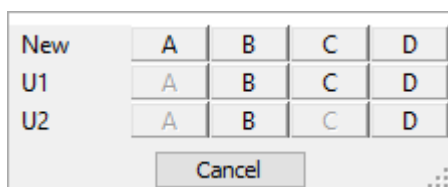


Рис. 3.7. Окно выбора элемента

это обозначение элемента на схеме. Смысл подписей может отличаться в зависимости от используемого элемента. Например, подпись под ключом соответствует «горячей» клавише для этого ключа. Следует отметить, что подписи под всеми логическими элементами соответствуют марке микросхемы. Необходимо обратить внимание, что оставление входов элементов

После выбора элемента необходимо поместить его на схему. Размещенный на схеме элемент по умолчанию будет сопровожден верхней и нижней надписью (рис. 3.8). Подпись соответствует марке микросхемы данного элемента, а надпись —

без сигнала может привести к ошибкам в схеме. Поэтому на любые неиспользуемые входы следует подавать сигналы с источников логических сигналов нуля или единицы в зависимости от принципа работы элемента. Например, на неиспользуемый вход элемента AND следует подавать логическую единицу, так как если подавать ноль, на выходе этого элемента будет всегда ноль.

Большинство сложных логических элементов, таких как триггеры или мультиплексоры, кроме рабочих входов, имеют еще и входы разрешения, сброса в ноль или установки в единицу.

Разберем работу некоторых сложных логических элементов. Начнем с мультиплексора 74LS151D (рис. 3.9). *Мультиплексоры* – это устройства, имеющие две группы входов – информационные и управляющие. Принцип работы мультиплексора заключается в следующем: на выход устройства передается информация с того информационного входа, номер которого в двоичной системе счисления подается на управляющие входы.

Мультиплексор 74LS151D имеет восемь информационных входов, три управляющих, вход разрешения и два выхода – прямой и инверсный. Данный мультиплексор работает в том случае, если на вход разрешения подается сигнал логического нуля. Мультиплексор можно также использовать в качестве какого-либо различного логического элемента. Для этого управляющие входы используются как входы логического элемента, а на информационные входы подаются постоянные сигналы нулей или единиц в зависимости от логики функционирования моделируемого логического элемента.

Для моделирования схем с памятью могут понадобиться триггеры. Рассмотрим JK-триггер 74LS107D (рис. 3.10), поскольку его можно использовать и как RS-, и как D-, и как T-триггер. Входы J и K являются информационными входами данного триггера. Вход CLK является входом синхронизации. Триггер будет менять свое состояние только тогда, когда на данном входе изменится значение с 1 на 0. Обычно на этот вход подают тактовый сигнал. Вход \sim CLR является входом сброса. При подаче на данный вход сигнала логического нуля триггер сбрасывается в начальное состояние (при котором на прямом выходе сигнал 0, а на инверсном – 1) независимо от того, какие сигналы будут приходить на другие входы. Кроме того, чтобы триггер работал, на вход \sim CLR необходимо подавать

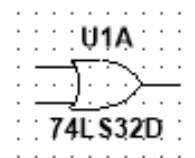


Рис. 3.8. Вид элемента на схеме

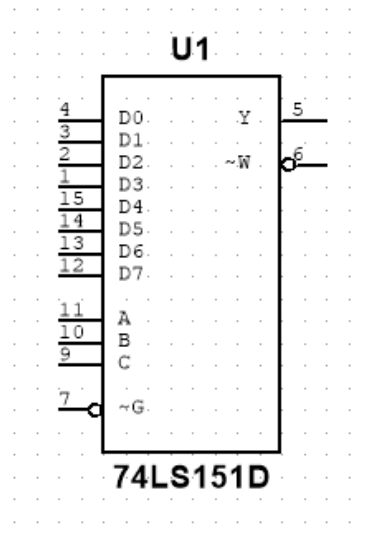


Рис. 3.9. Мультиплексор 74LS151D

сигнал логической единицы. Триггер имеет два выхода: прямой Q и инверсный $\sim Q$.

Рассмотрим кодер 74LS147D (рис. 3.11). Данный кодер имеет четыре выхода и девять входов. Все входы и выходы являются инверсными. Данный кодер работает следующим образом: на выходах появляется номер того входа, на котором появляется сигнал логического нуля в двоичной системе счисления, с инверсией всех разрядов. На данный кодер следует подавать сигнал нуля только на один из входов.

Рассмотрим декодер 74LS138D (рис. 3.12). Данный декодер имеет три информационных входа, три входа разрешения, два из которых инверсные, и восемь инверсных выходов. Принцип работы данного устройства следующий: в случае, если на не инверсном входе разрешения будет сигнал логической единицы, а на инверсных входах разрешения сигналы логического нуля, то сигнал логической единицы будет на том выходе, номер которого в двоичном виде приходит на информационные входы. Если использовать неинверсный вход разрешения как информационный, а информационные входы как управляющие, то данный декодер можно использовать в качестве демультиплексора.

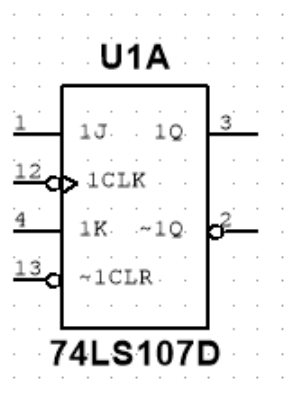


Рис. 3.10. JK-триггер 74LS107D

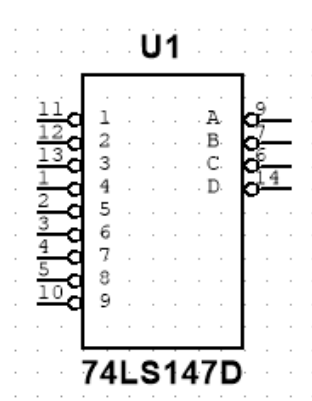


Рис. 3.11. Кодер 74LS147D

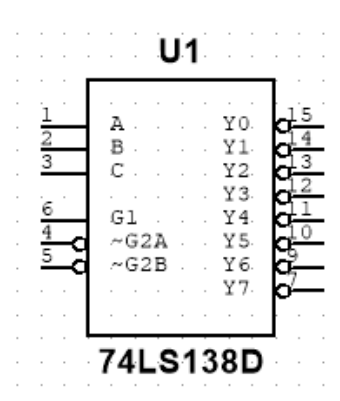


Рис. 3.12. Декодер 74LS138D

Некоторые подписи и надписи можно убрать или добавить. Для этого следует в окне схемы щелкнуть правой кнопкой мыши и в появившемся списке выбрать строку Properties. В открывшемся окне, во вкладке Sheet visibility, можно убирать или добавлять различные надписи. В этом же окне, в других вкладках, можно выбрать цветовую гамму окна со схемой из уже заготовленных или создать свою, а также размер и ориентацию листа, толщину проводов и шин и шрифт надписей.

Различные элементы на схеме соединяются проводами. По умолчанию провода имеют красный цвет и не имеют имен. Цвет провода и его

наименование можно изменить. Для этого следует выделить провод, щелкнуть по нему правой кнопкой мыши и выбрать строку во вкладке Properties. В открывшемся окне, Net name, можно поменять как цвет, так и имя провода.

При большом количестве проводов удобно использовать шины. В одну шину может входить множество проводов. Для размещения шины на схеме необходимо на панели инструментов в верхней части окна нажать кнопку Bus. При подключении провода к шине высвечивается окно со списком проводов в шине (рис. 3.13). Изначально данный список пуст. В строке Bus line вводится имя провода в шине, к которому необходимо подключить входящий в нее провод. Если в списке провода с введенным в эту строку названием нет, то создается новый провод с таким названием. Входящие и исходящие из шины провода с одинаковыми названиями отождествляются с одним проводом, так как по ним проходит одинаковый сигнал. Таким образом, чтобы избежать нагромождения проводов, можно использовать шины.

Справа на панели инструментов находятся два осциллографа. Первый (oscilloscope) – двухканальный, а второй (four channel oscilloscope) – четырехканальный. К входам A, B, C, D данного осциллографа подключаются провода, сигналы в которых необходимо анализировать. Для вывода на экран окна осциллографа (рис. 3.14) необходимо дважды щелкнуть по нему левой кнопкой мыши.

Большую часть окна занимает экран осциллографа. В нижней части окна находятся поля настройки цены деления шкалы времени, указаны позиции осциллограмм относительно оси Y, цена деления оси Y и положение осциллограмм относительно оси X. Цвет осциллограммы на экране будет аналогичен цвету провода, подключенного к соответствующему каналу. Нажав кнопку Reverse, можно поменять цвет фона с черного на белый.

Полезным на практике для симуляции работы схем может оказаться генератор кодовых слов (Word generator). Он находится в правой панели инструментов. Его задача – формирование заранее заданной последовательности кодовых слов. Для управления генератором необходимо открыть окно (рис. 3.15) двойным щелчком левой кнопки мыши по самому

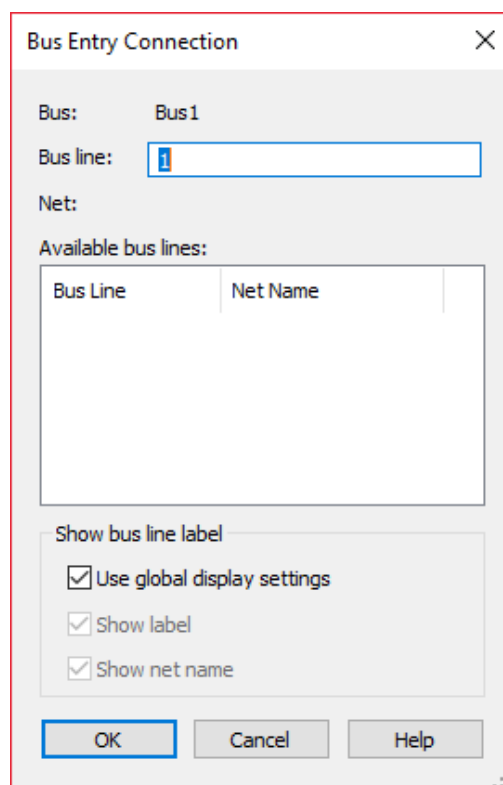


Рис. 3.13. Окно со списком проводов в шине

генератору. В правой части окна изображена последовательность кодовых слов. Она может задаваться в двоичной, десятичной, шестнадцатеричной системах счисления или в коде ASCII. Выбор системы счисления осуществляется в области левее изображения последовательности кодов. При щелчке левой кнопкой мыши по квадрату левее изображения какого-либо кодового слова открывается список, при помощи которого можно задавать начальное слово последовательности (Set initial position), конечное слово (Set final position), точку остановки (Set breakpoint) или поместить курсор на данную позицию (Set Cursor). В левой части окна находятся кнопки управления генератором. Нажатием кнопки Cycle запускается непрерывная работа схемы. В этом режиме остановка схемы происходит только при достижении генератором точки остановки.

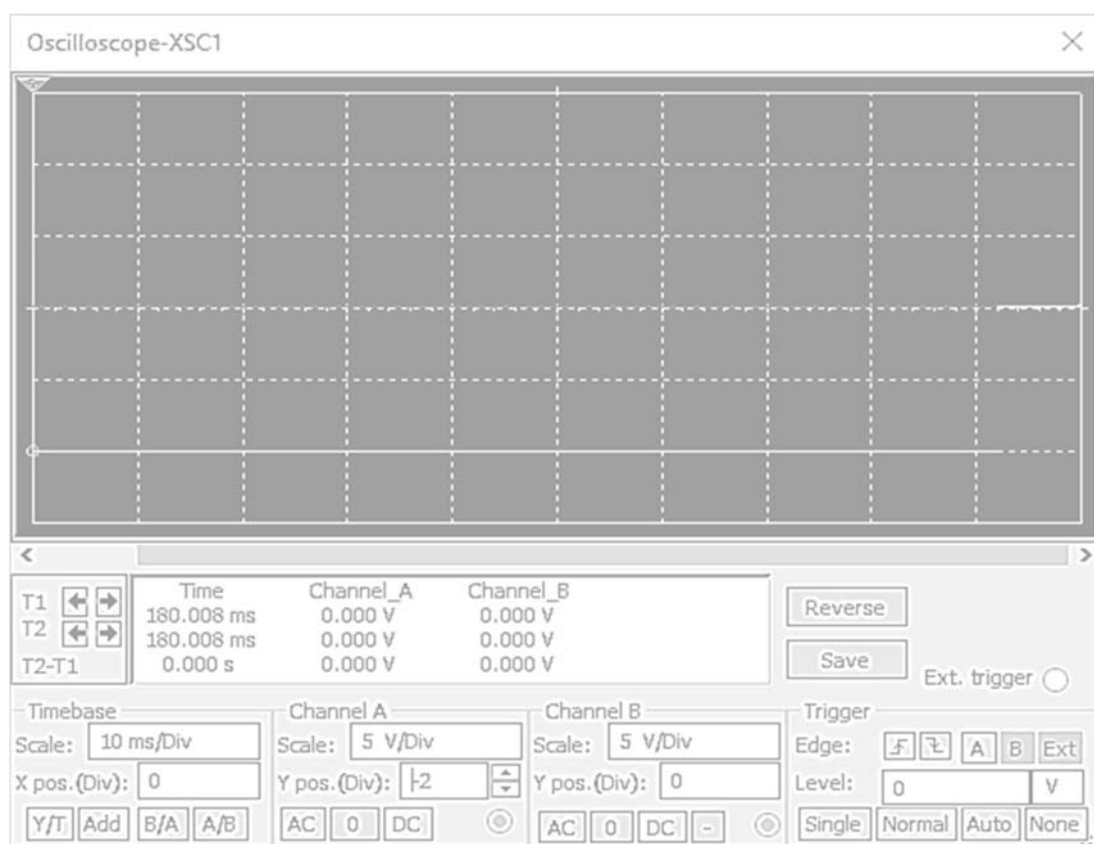


Рис. 3.14. Окно осциллографа

При нажатии кнопки Burst схема запускается и останавливается при достижении генератором финального слова последовательности или точки остановки. При нажатии кнопки Step схема запускается и останавливается при достижении генератором следующего кодового слова. Кнопкой Reset осуществляется перевод генератора в начальное состояние, при котором он формирует начальное кодовое слово. При нажатии кнопки Set открывается окно настройки генератора. В нем можно настроить напряжения логиче-

ской единицы и нуля, сохранить текущую кодовую последовательность, загрузить ранее сохраненную, отчистить кодовую последовательности и задать количество слов последовательности. Нажатием кнопки Internal генератор переводится в режим работы за счет внутреннего тактового генератора. Кнопкой External генератор переводится на режим работы от внешнего тактового генератора, подключаемого к входу Т. Ниже кнопки External задается частота внутреннего тактового генератора.

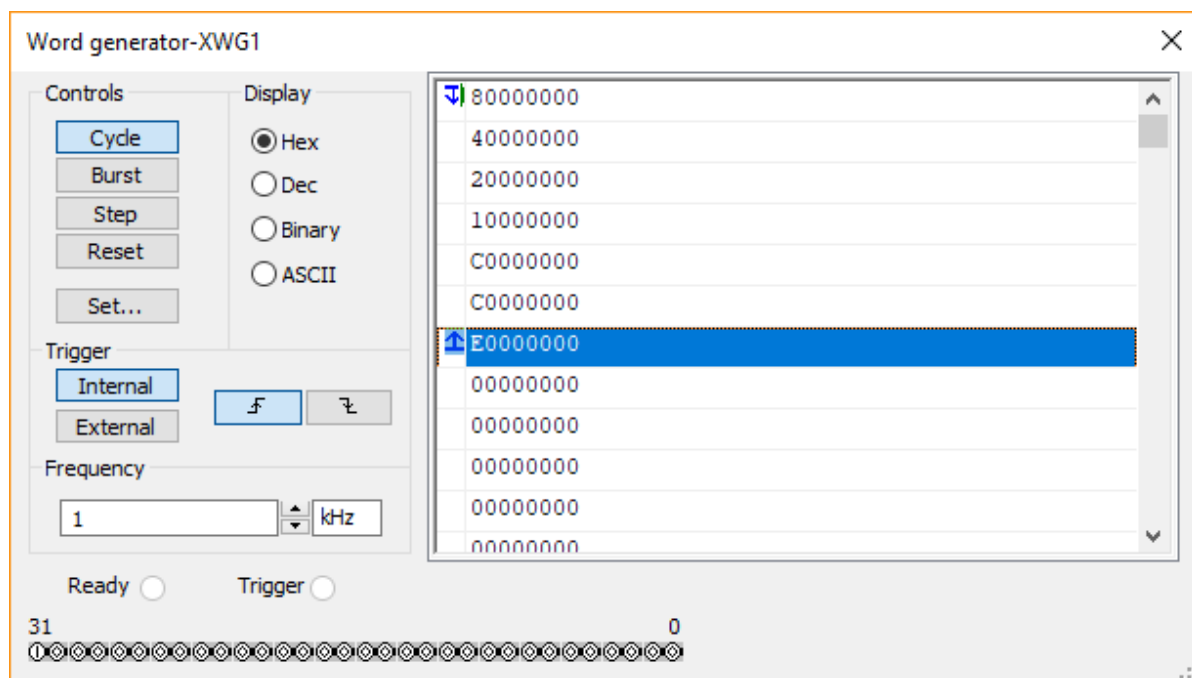


Рис. 3.15. Генератор кодовых слов

3.2. Синтез синхронного автомата, снабженного встроенной схемой контроля

Пользуясь Multisim, реализуем схему конечного автомата, полученную в результате синтеза (см. рис. 2.17). Изображенная в среде Multisim схема приведена на рис. 3.16.

Моделирование исходной последовательности входных воздействий (см. рис. 2.1) позволяет подтвердить корректность проведенной процедуры синтеза.

Снабдим логический преобразователь схемой контроля по паритету (рис. 3.17).

Аналогично синтезируются и схемы автоматов, состояния которых закодированы каким-либо блочным помехозащитным кодом.

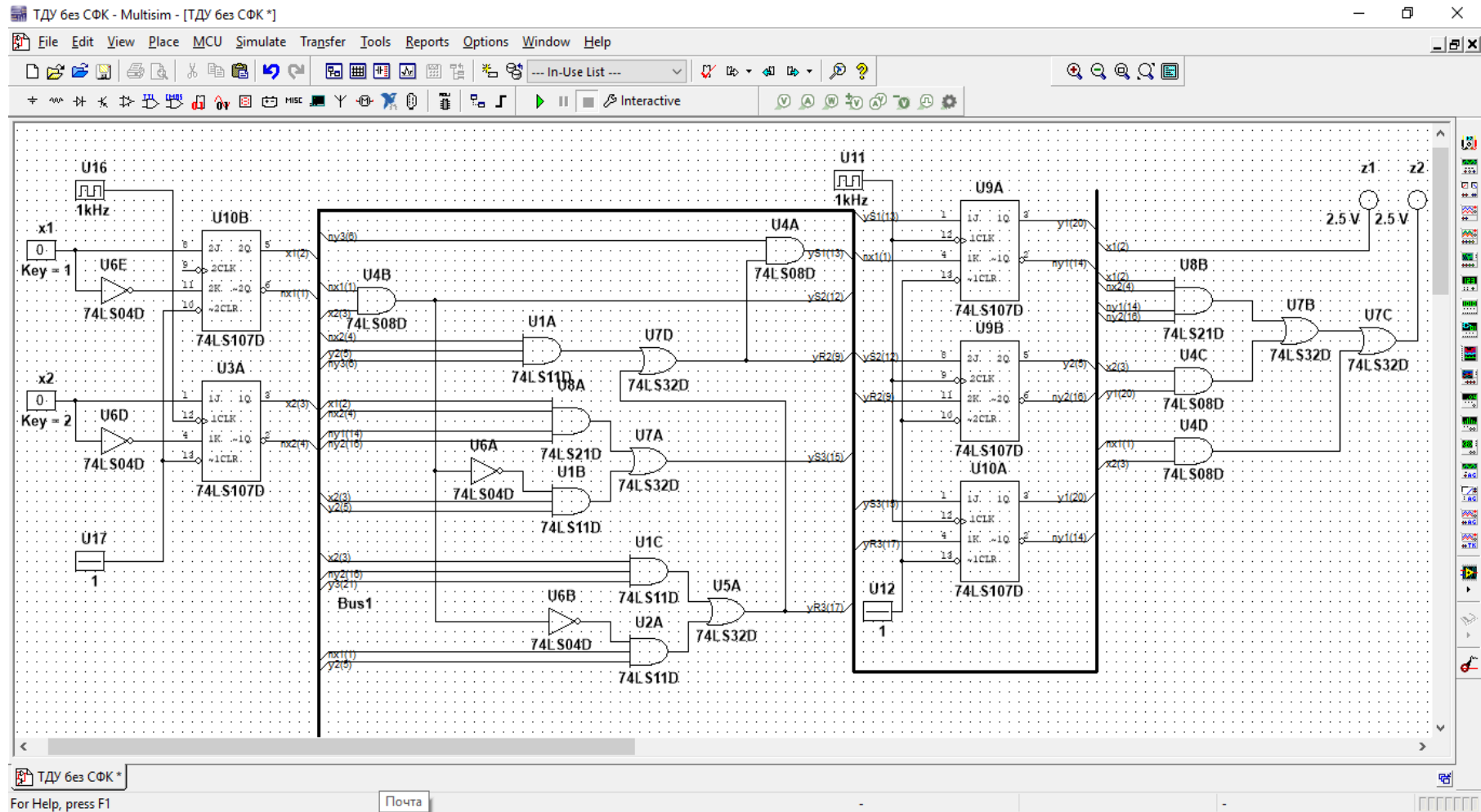


Рис. 3.16. Изображенная в Multisim схема конечного автомата

3.3. Моделирование работы автомата при неисправностях элементов

3.3.1. Конечный автомат как объект диагностирования

Как отмечалось ранее, конечный автомат состоит из ряда функциональных блоков (рис. 3.18). Схемы синхронизации (СС) и блок памяти (БП) представляют собой схемы, обладающие памятью, тогда как логический и выходной преобразователи (ЛП и ВП) являются комбинационными схемами.

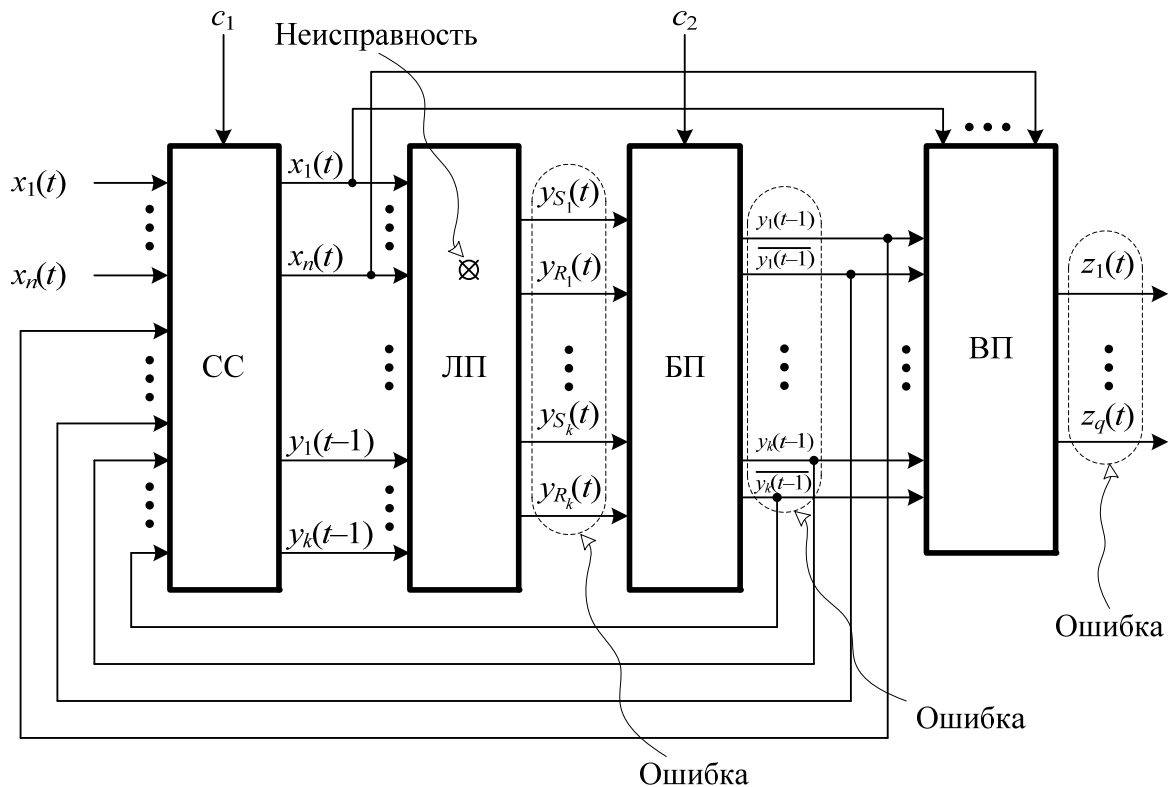


Рис. 3.18. Распространение ошибок в конечном автомате

Если при тестировании комбинационной схемы для какой-либо единичной константной неисправности достаточно только одного тестового набора (такого набора, подача которого на входы обеспечит проявление неисправности на соответствующей линии схемы, а также трансляцию ошибки на выходы схемы [10]), то процедура тестирования конечного автомата значительно сложнее.

Рассмотрим неисправность, возникающую в структуре логического преобразователя (см. рис. 3.18). Необходимо, чтобы неисправность проявилась на выходах логического преобразователя. Для этого сама схема конечного автомата должна перейти в определенное состояние, предшествующее проявлению отказа в виде ошибки (задача установки состояния). Затем нужно, чтобы ошибка, вызванная неисправностью логического пре-

образователя, распространилась через блок памяти и выходной преобразователь на выходы конечного автомата (задача трансляции ошибки). В этом случае по несовпадению выходных значений с ожидаемыми значениями на соответствующем такте работы конечного автомата можно зафиксировать наличие неисправности. При этом результат технического диагностирования будет выведен в виде «исправен/неисправен», а конкретное место возникновения дефекта указано не будет. Другими словами, будет решена задача проверки исправности конечного автомата (проведена процедура подачи последовательности входных воздействий на конечный автомат в виде проверяющего теста).

Процедуру распространения неисправности в конечном автомате удобно моделировать с использованием графа переходов. Для этого рассматривается входная последовательность сигналов (например, по минимизированному графу переходов, изображенному на рис. 2.6) и отмечается последовательность переходов между состояниями графа переходов. При определенных условиях, при наличии неисправности во внутренней структуре конечного автомата (если, конечно, схема не избыточна), последовательность смены его состояний нарушится или же появятся некорректные выходные значения.

3.3.2. Моделирование работы конечного автомата

Процесс моделирования реального объекта непосредственно по графу переходов оказывается затруднительным, так как требуется перерасчет функций в таблице переходов при внесении конкретной неисправности в технический объект, а затем получение нового графа переходов. Гораздо проще моделировать работу конечного автомата с неисправностями путем использования модели дискретного устройства, синтезированной, например, в Multisim, и исходных вход-выходных временных последовательностей. При их нарушении будет зафиксирована неисправность.

Рассмотрим процесс моделирования работы автомата с неисправностью на примере анализа работы конечного автомата со схемой обнаружения неисправностей по коду паритета. Для этого внесем неисправность в логический преобразователь и зафиксируем установочную и трансляционную последовательности для нее. Кроме того, определим, фиксируется ли данная неисправность схемой контроля.

Для внесения неисправности какого-либо элемента оборвем провод, идущий с выхода этого элемента. Подсоединим ко входу элемента, на который шел сигнал, источник переменного логического сигнала. Сигнал логической единицы или нуля на выходе этого источника будет моделировать соответствующую неисправность элемента, вместо которого присоединен. Затем будем подавать последовательности входных сигналов, заданные на рис. 2.1 или 2.2, и фиксировать непарафазный сигнал на кон-

трольных выходах. Если после подачи всех последовательностей непарафазный сигнал так и не появился, то неисправность считается необнаруженной. В табл. 3.2 приведены неисправности некоторых элементов логического преобразователя и информация об их обнаружении. Названия элементов соответствуют названиям на рис. 3.17. Если ошибка обнаружена, то указывается состояние исходного графа переходов, на котором эта ошибка была обнаружена.

Таблица 3.2. Названия логических элементов на английском языке

Название элемента на схеме	Вид ошибки	Факт обнаружения на выходах схемы контроля	Последовательность, в конце которой ошибка проявляется в автомате
U4B	Константа 0	Обнаружена	1-5
	Константа 1	Обнаружена	1
U1A	Константа 0	Не обнаружена	1-5-8
	Константа 1	Обнаружена	1-2
U7D	Константа 0	Не обнаружена	1-5-6-7-1
	Константа 1	Обнаружена	1-2
U1C	Константа 0	Не обнаружена	1-2-3-4
	Константа 1	Обнаружена	1-2
U5A	Константа 0	Не обнаружена	1-2-3-4
	Константа 1	Обнаружена	1
U2A	Константа 0	Не обнаружена	1-5-6-7-1
	Константа 1	Обнаружена	1
U6B	Константа 0	Не обнаружена	1-5-6-7-1
	Константа 1	Обнаружена	1-5

Заключение

Развитие техники и технологий первой половины XXI в. открывает широкие перспективы для реализации систем автоматического управления на основе микроэлектронной и микропроцессорной техники, в том числе в области железнодорожной автоматики. С усложнением материальной базы и появлением новой техники усложняются и подходы к обеспечению правильности функционирования блоков и компонентов разрабатываемых систем автоматического управления. Основные же методы обеспечения надежности и безопасности ориентированы на внесение аппаратной и программной избыточности в соответствующие компоненты систем управления.

Знания в области синтеза дискретных устройств с обнаружением отказов позволяют будущему инженеру, работающему в области автоматики и вычислительной техники, иметь представление о методах внесения избыточности в структуры устройств для наделения их свойством контролепригодности и идентификации неисправностей в процессе эксплуатации. Выбор способа контроля технического состояния имеет фундаментальное значение и определяет все ключевые характеристики конечного дискретного устройства, включая те из них, которые непосредственно влияют на стоимость его разработки и эксплуатации.

Представленные в учебном пособии способы синтеза дискретных устройств с обнаружением неисправностей универсальны и не ограничивают читателя только элементной базой в виде набора микросхемных реализаций логических элементов. Сведения, полученные читателем на страницах данной книги, могут быть использованы для применения программируемых логических интегральных схем для реализации дискретных устройств. Такие схемы получили в настоящее время широкое распространение во всех областях науки и техники, в том числе в транспортной отрасли [16, 17, 20, 21].

Библиографический список

1. Аксенова Г. П. Необходимые и достаточные условия построения полностью проверяемых схем свертки по модулю 2 / Г. П. Аксенова // Автоматика и телемеханика. – 1979. – № 9. – С. 126–135.
2. Гавзов Д. В. Методы обеспечения безопасности дискретных систем / Д. В. Гавзов, Вал. В. Сапожников, Вл. В. Сапожников // Автоматика и телемеханика. – 1994. – № 8. – С. 3–50.
3. Ефанов Д. В. О свойствах кода с суммированием в схемах функционального контроля / Д. В. Ефанов, Вал. В. Сапожников, Вл. В. Сапожников // Автоматика и телемеханика. – 2010. – № 6. – С. 155–162.
4. Ефанов Д. В. Применение модульных кодов с суммированием для построения систем функционального контроля комбинационных логических схем / Д. В. Ефанов, Вал. В. Сапожников, Вл. В. Сапожников // Автоматика и телемеханика. – 2015. – № 10. – С. 152–169.
5. Ефанов Д. В. Синтез самопроверяемых комбинационных устройств на основе выделения специальных групп выходов / Д. В. Ефанов, Вал. В. Сапожников, Вл. В. Сапожников // Автоматика и телемеханика. – 2018. – № 9. – С. 79–94.
6. Ефанов Д. В. Условия обнаружения неисправности логического элемента в комбинационном устройстве при функциональном контроле на основе кода Бергера / Д. В. Ефанов, Вал. В. Сапожников, Вл. В. Сапожников // Автоматика и телемеханика. – 2017. – № 5. – С. 152–165.
7. Ефанов Д. В. Функциональный контроль и мониторинг устройств железнодорожной автоматики и телемеханики : монография / Д. В. Ефанов. – СПб. : ФГБОУ ВО ПГУПС, 2016. – 171 с.
8. Сапожников Вал. В. Методы построения безопасных микроэлектронных систем железнодорожной автоматики / Вал. В. Сапожников, Вл. В. Сапожников, Х. А. Христов, Д. В. Гавзов ; под ред. Вл. В. Сапожникова. – М. : Транспорт, 1995. – 272 с.
9. Сапожников Вал. В. Надежность систем железнодорожной автоматики, телемеханики и связи : учеб. пособие / Вал. В. Сапожников, Вл. В. Сапожников, Д. В. Ефанов, В. И. Шаманов ; под ред. Вл. В. Сапожникова. – М. : ФГБУ ДПО «Учебно-методический центр по образованию на железнодорожном транспорте», 2017. – 318 с.
10. Сапожников Вал. В. Основы технической диагностики : учебник / Вал. В. Сапожников, Вл. В. Сапожников, Д. В. Ефанов // Издание второе, перераб. и доп. ; под ред. Вал. В. Сапожникова. – М. : ФГБУ ДПО «Учебно-методический центр по образованию на железнодорожном транспорте», 2019. – 424 с.
11. Сапожников Вал. В. Получение функций включения элементов памяти конечного автомата при кодировании состояний по столбцам таблицы переходов / Вал. В. Сапожников, Вл. В. Сапожников // Проблемы передачи информации. – 1973. – Т. 9. – № 4. – С. 90–91.
12. Сапожников Вал. В. Самопроверяемые дискретные устройства / Вал. В. Сапожников, Вл. В. Сапожников. – СПб. : Энергоатомиздат, 1992. – 224 с.
13. Сапожников Вал. В. Теория дискретных устройств железнодорожной автоматики, телемеханики и связи : учебник / Вал. В. Сапожников, Вл. В. Сапожников, Д. В. Ефанов ; под ред. Вал. В. Сапожникова. – М. : ФГБОУ «Учебно-методический центр по образованию на железнодорожном транспорте», 2016. – 339 с.
14. Сапожников Вл. В. Микропроцессорные системы централизации : учебник для техникумов и колледжей железнодорожного транспорта / Вл. В. Сапожников, В. А. Кононов, С. А. Куренков, А. А. Лыков, О. А. Наседкин, А. Б. Никитин, А. А. Про-

кофьев, М. С. Трясов ; под ред. Вл. В. Сапожникова. – М. : ГОУ «Учебно-методический центр по образованию на железнодорожном транспорте», 2008. – 398 с.

15. Согомонян Е. С. Самопроверяемые устройства и отказоустойчивые системы / Е. С. Согомонян, Е. В. Слабаков. – М. : Радио и связь, 1989. – 207 с.

16. Хаханов В. И. Проектирование и тестирование цифровых систем на кристаллах / В. И. Хаханов, Е. И. Литвинова, О. А. Гузь. – Харьков : ХНУРЭ, 2009. – 484 с.

17. Dobias R. FPGA Based Design of the Railway's Interlocking Equipments / R. Dobias, H. Kubatova // Euromicro Symposium on Digital System Design (DSD 2004), 31 Aug. – 3 Sept. 2004, Rennes, France. – Pp. 467–473.

18. Fang Y. Y. Design and simulation of DDS based on Quartus II // Y. Y. Fang; X. J. Chen // IEEE International Conference on Computer Science and Automation Engineering (CSAE), Shanghai, China, 10–12 June 2011. Vol. 2. – N 2. – Pp. 357–360.

19. Goessel M. Error Detection Circuits / M. Goessel, S. Graf. – L. : McGraw-Hill, 1994. – 261 p.

20. Kacou M. A. Error Rate Estimation of a Design Implemented in an FPGA based on the Operating Conditions / M. A. Kacou, F. Ghaffari, O. Romain, B. Condamin // Proceedings of 15th IEEE East-West Design & Test Symposium (EWDTS'2017), Novi Sad, Serbia, September 29 – October 2, 2017. – Pp. 459–465.

21. Kharchenko V. Green IT Engineering: Concepts, Models, Complex Systems Architectures / V. Kharchenko, Yu. Kondratenko, J. Kacprzyk // Springer Book series «Studies in Systems, Decision and Control». – 2017. – Vol. 74. – 305 p.

22. Matrosova A. Yu. Self-Checking Synchronous FSM Network Design with Low Overhead / A. Yu. Matrosova, I. Levin, S. A. Ostanin // VLSI Design. – 2000. – Vol. 11. – Issue 1. – Pp. 47–58.

23. Sentovich E. M. Sequential circuit design using synthesis and optimization / E. M. Sentovich, K. J. Singh, C. Moon, H. Savoj, R. K. Brayton, A. Sangiovanni-Vincentelli // Proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD'92), 11–14 October 1992, Cambridge, MA, USA. – Pp. 328–333.

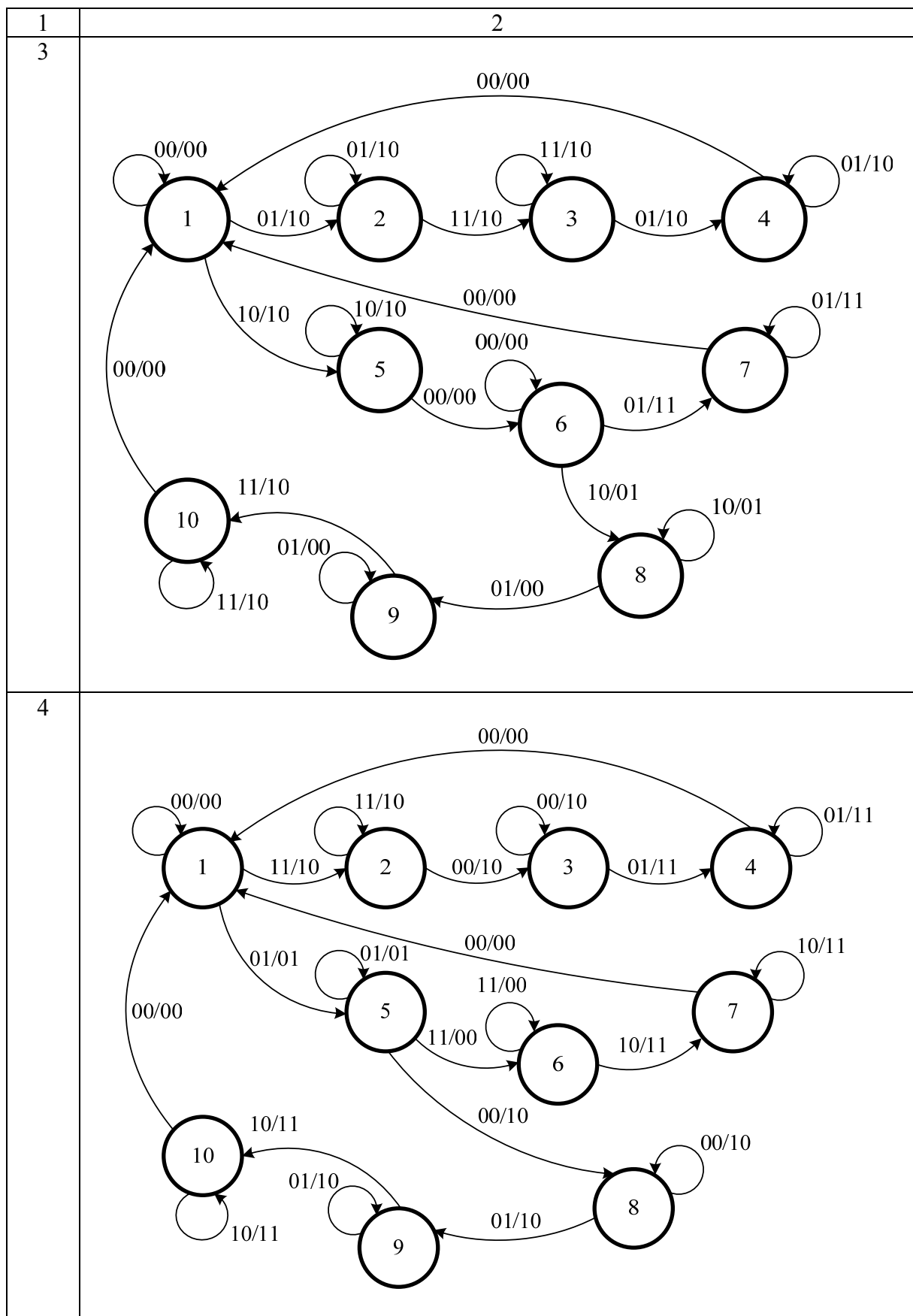
24. Sentovich E. M. SIS: A System for Sequential Circuit Synthesis / E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, A. Sangiovanni-Vincentelli // Electronics Research Laboratory, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 4 May 1992. – 45 p.

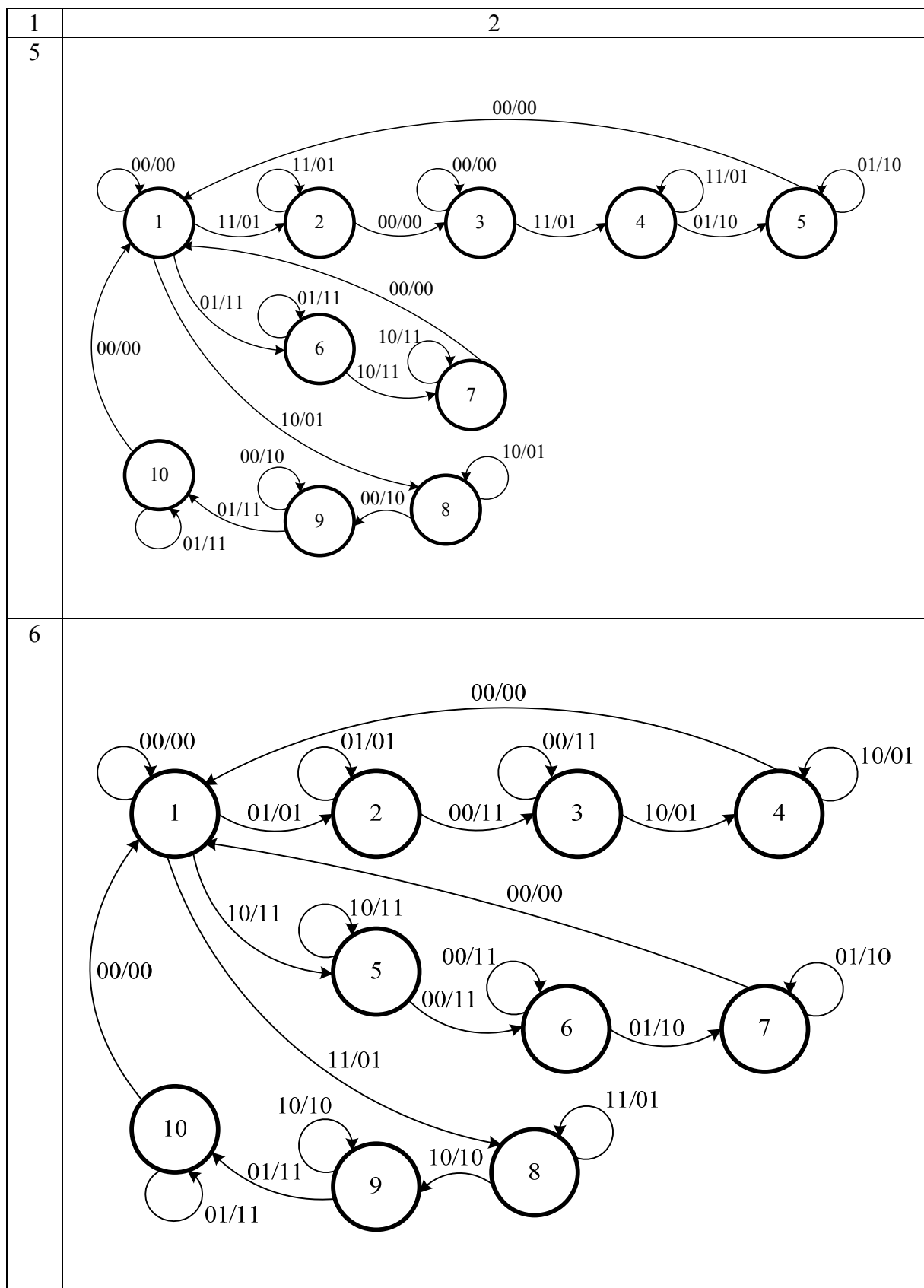
ПРИЛОЖЕНИЕ А. ВАРИАНТЫ ЗАДАНИЙ

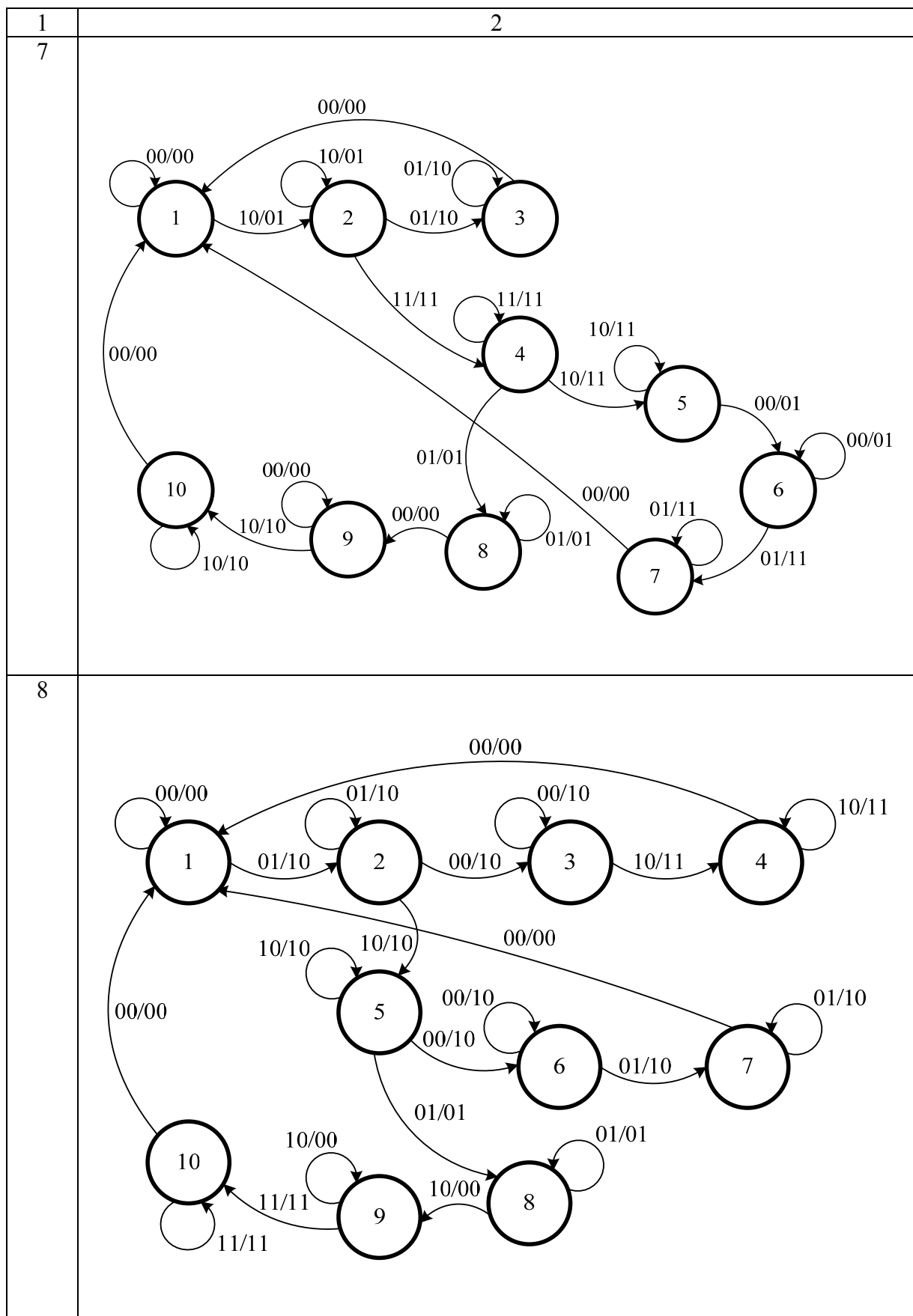
I. Варианты заданий небольшой сложности: в качестве исходных данных дается граф переходов конечного автомата.

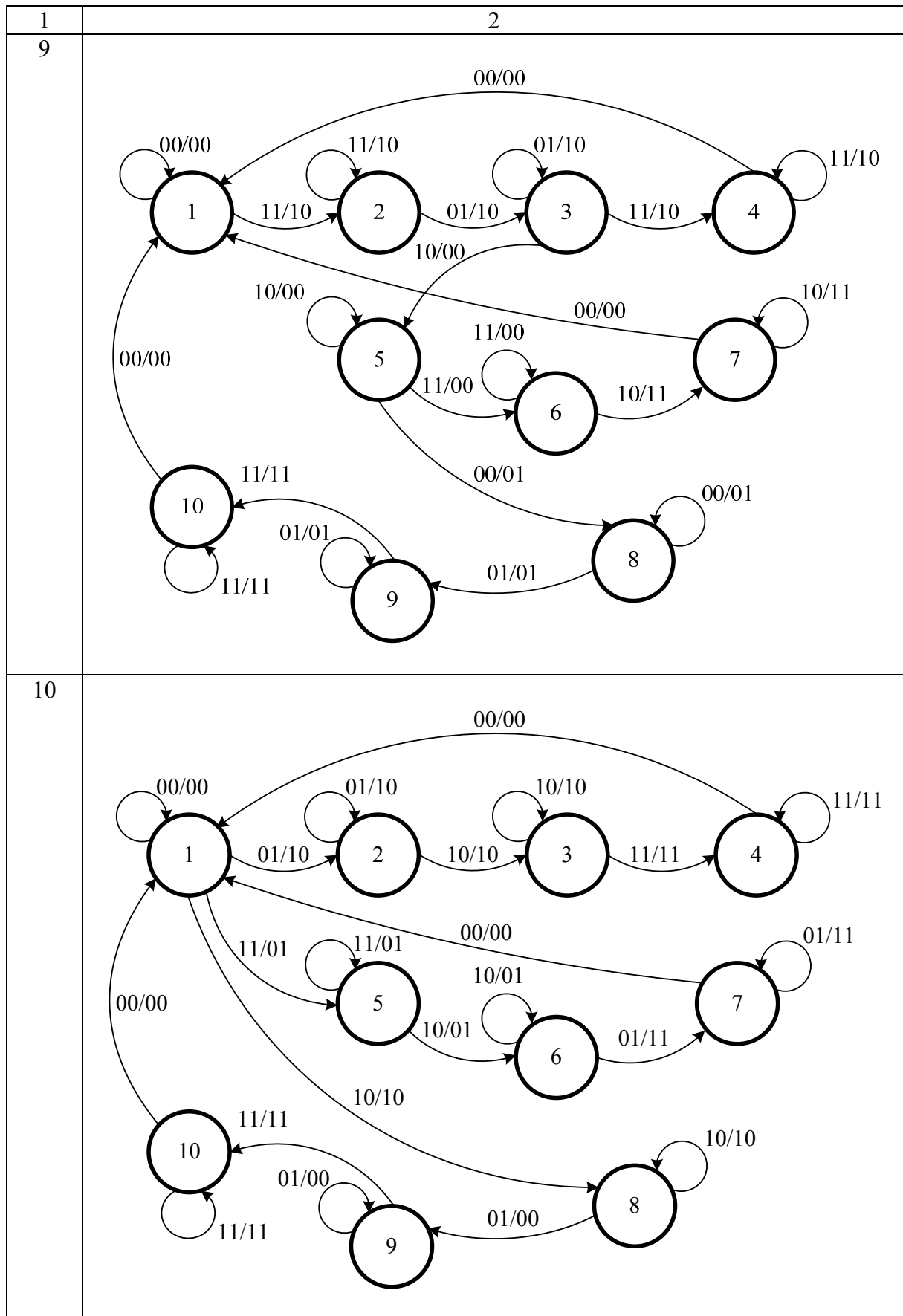
Таблица П.А1

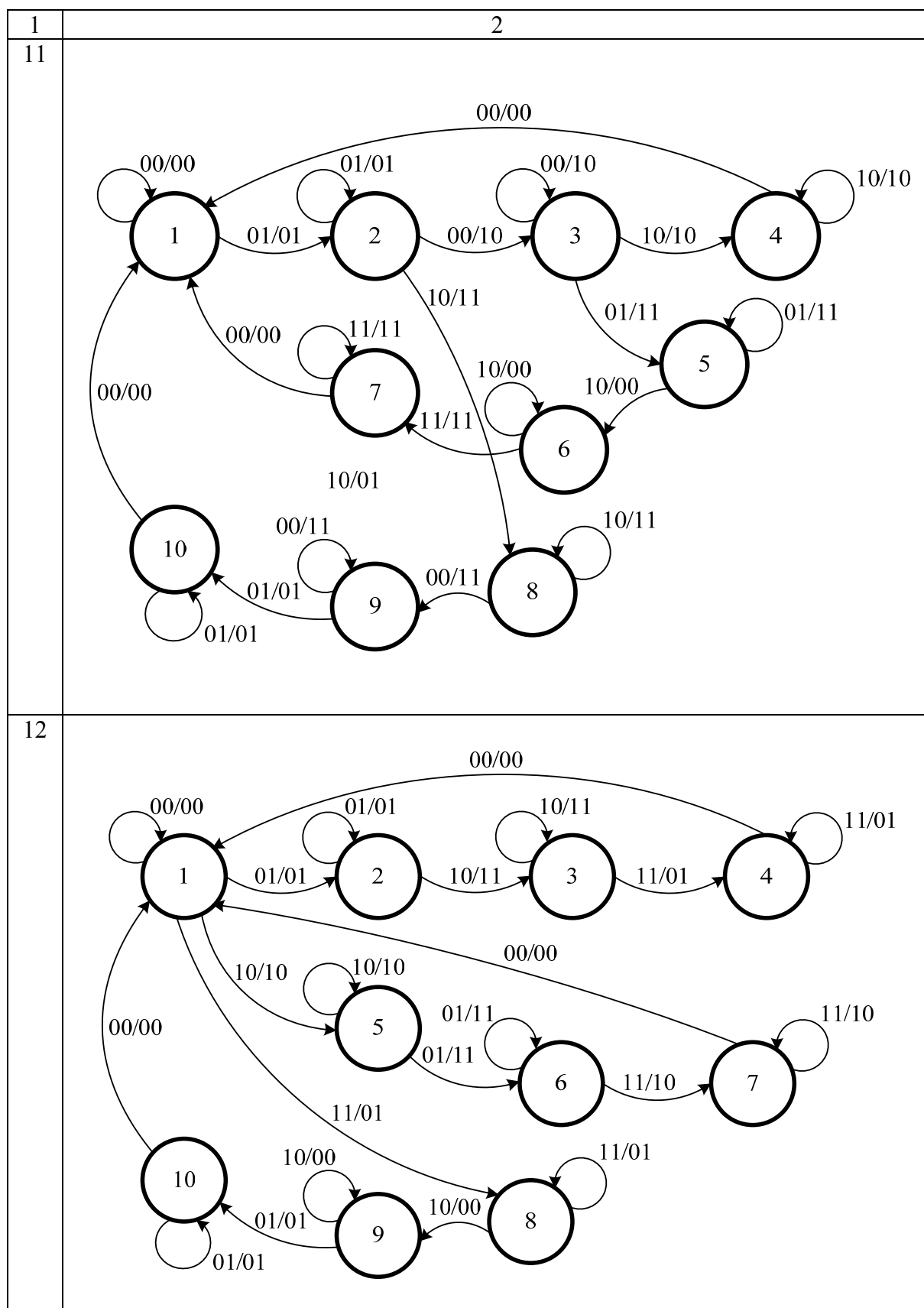
Вариант	Граф переходов
1	2
1	
2	

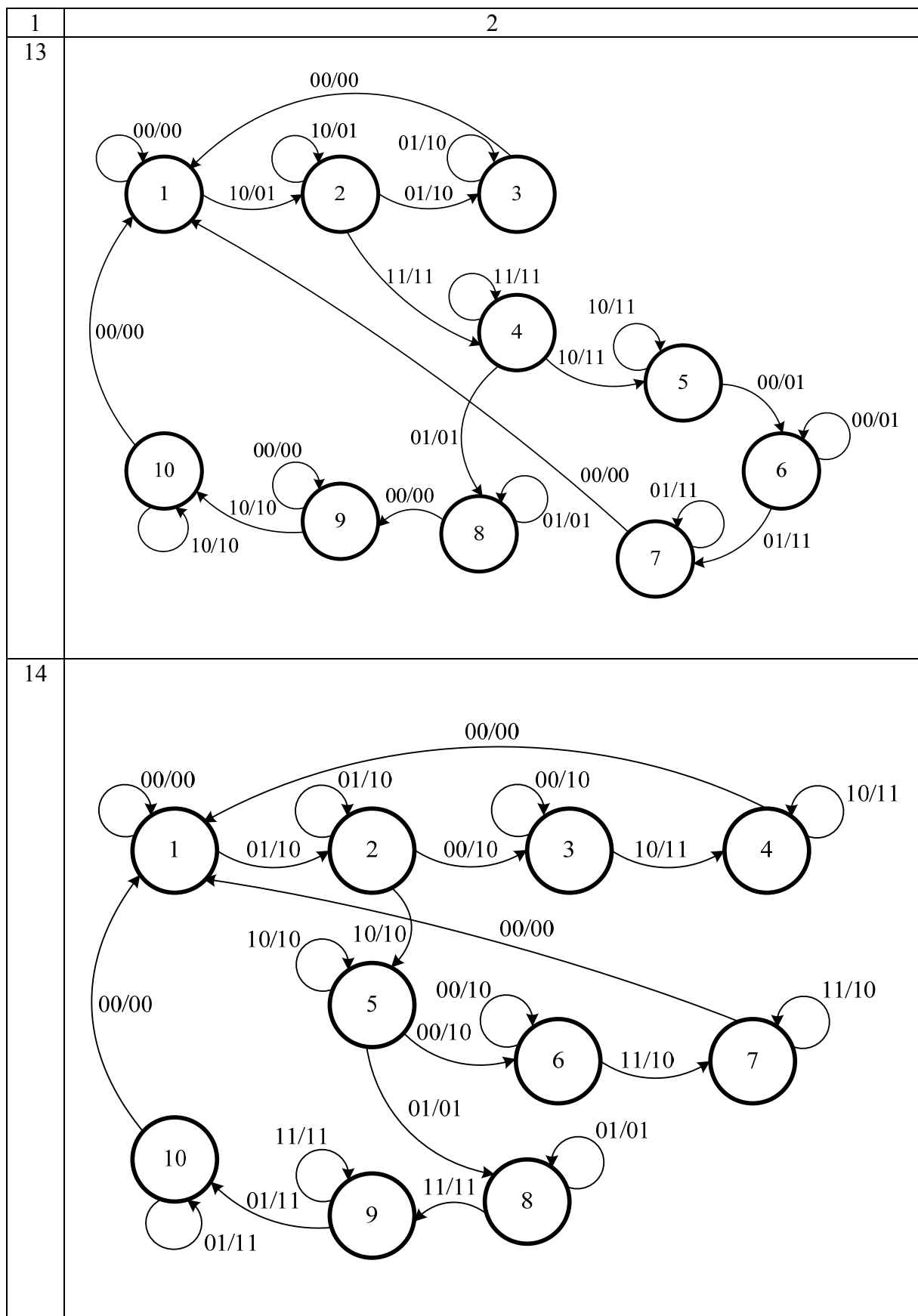


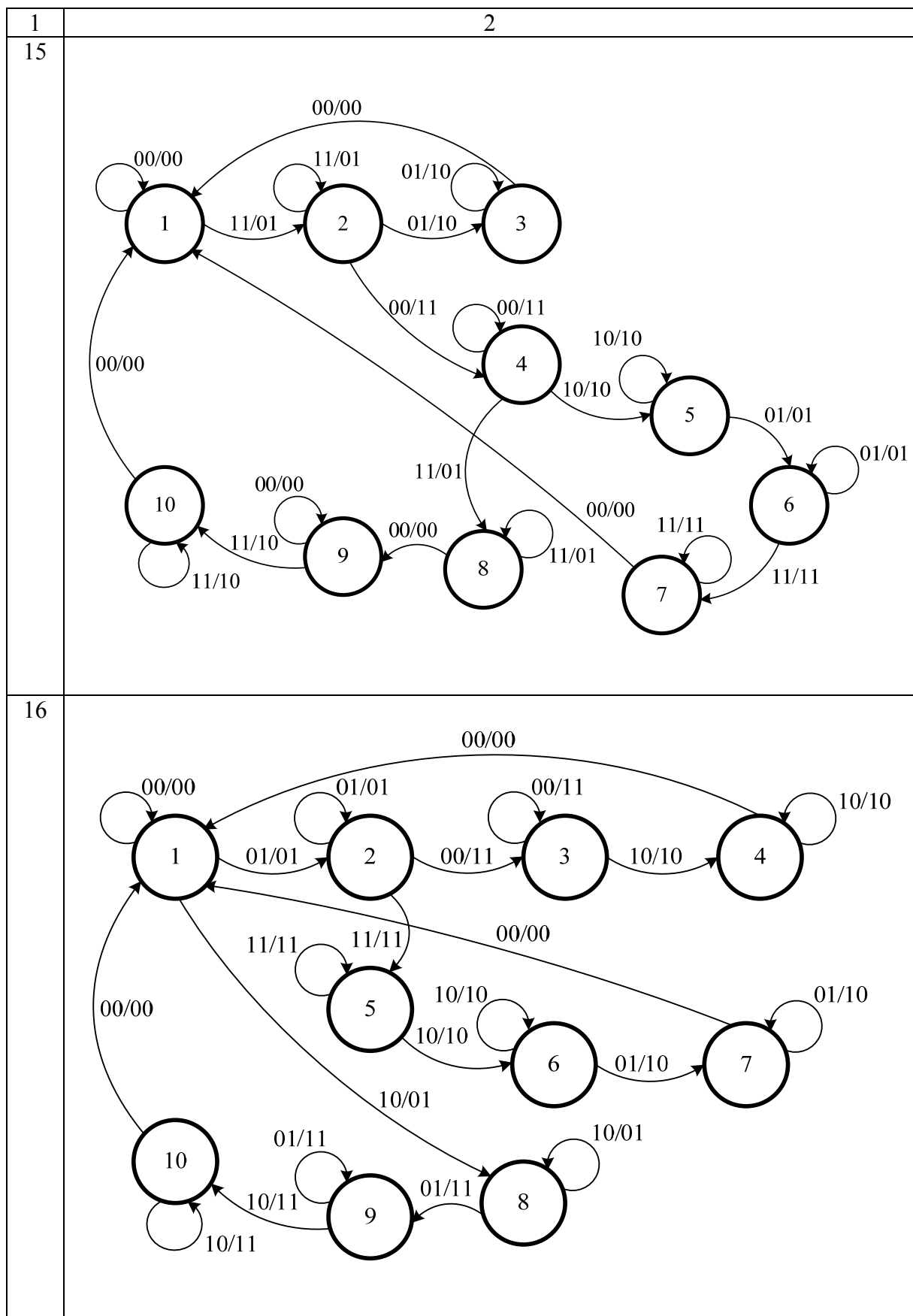


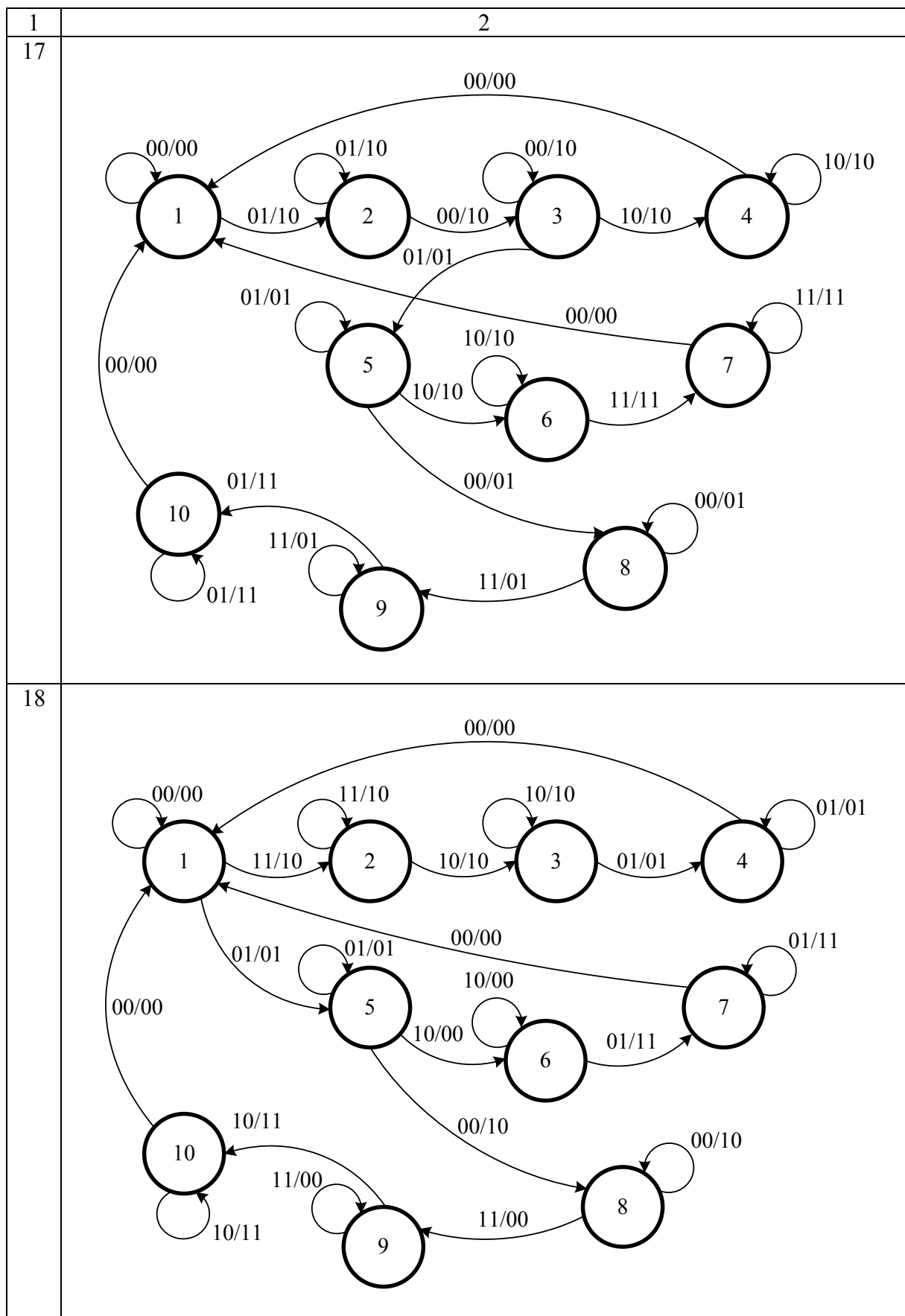


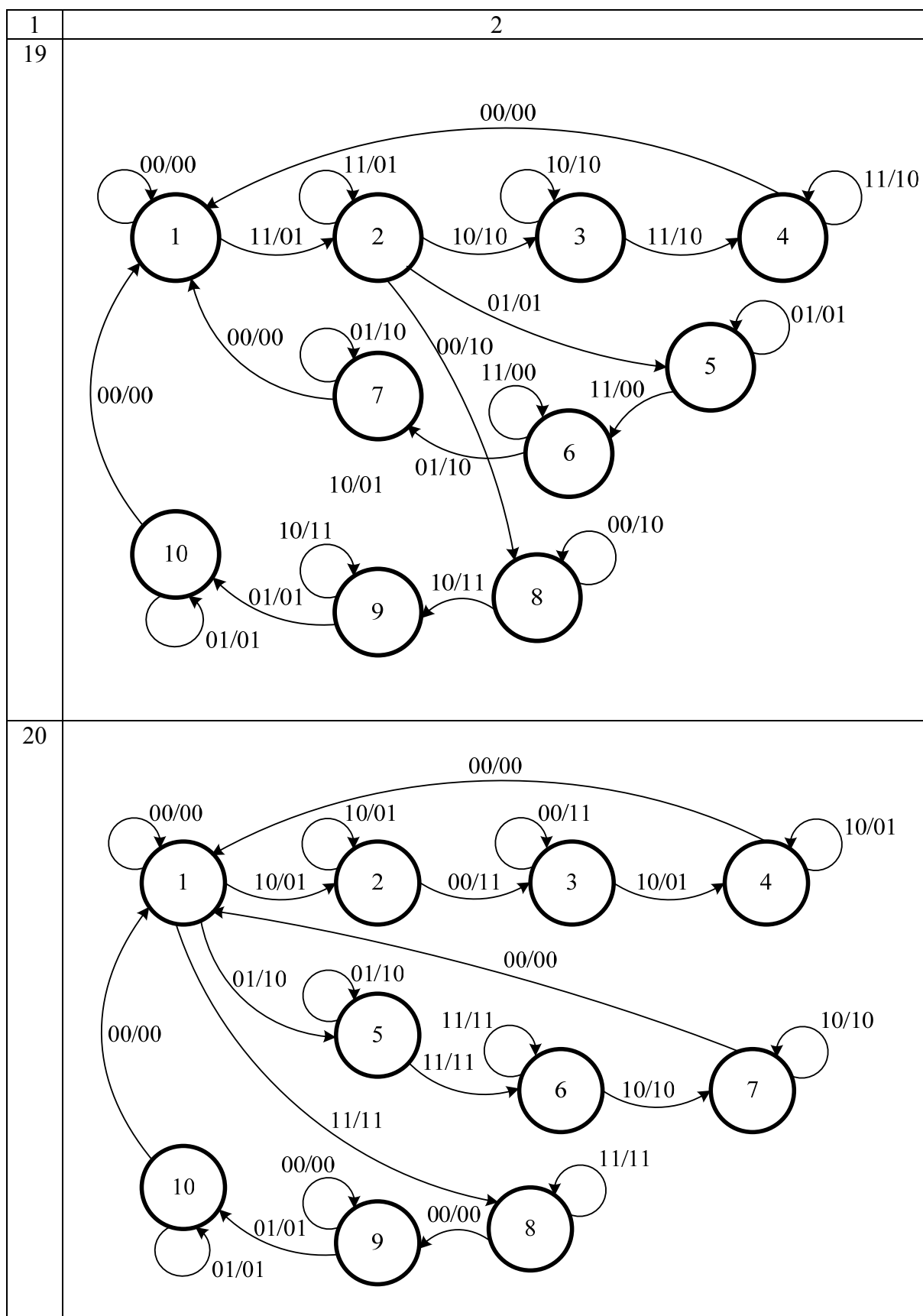












II. *Варианты усложненных заданий: работа устройства задается словесным описанием.*

Таблица П.А2

Вариант	Словесное описание условий функционирования
1	Устройство имеет два входа – x_1 , x_2 , управляемые кнопками, и два выхода – z_1 , z_2 , на которых установлены индикаторы (к примеру, лампы). Одновременное задание значений на обоих входах исключено. Алгоритм работы: лампа z_1 включается при каждом нечетном нажатии кнопки x_1 , лампа z_2 включается при каждом четном нажатии кнопки x_2 .
2	Устройство имеет два входа – x_1 , x_2 , управляемые кнопками, и два выхода – z_1 , z_2 , на которых установлены индикаторы (к примеру, лампы). Одновременное задание значений на обоих входах исключено. Алгоритм работы: лампа z_1 включается при каждом третьем нажатии кнопки x_1 , лампа z_2 включается при каждом четном нажатии кнопки x_2 .
3	Устройство имеет два входа – x_1 , x_2 , управляемые кнопками, и два выхода – z_1 , z_2 , на которых установлены индикаторы (к примеру, лампы). Одновременное задание значений на обоих входах исключено. Алгоритм работы: лампа z_1 включается при каждом нечетном нажатии кнопки x_1 , лампа z_2 включается при каждом третьем нажатии кнопки x_2 .
4	Устройство имеет два входа – x_1 , x_2 , управляемые кнопками, и два выхода – z_1 , z_2 , на которых установлены индикаторы (к примеру, лампы). Одновременное задание значений на обоих входах исключено. Алгоритм работы: лампа z_1 включается при каждом четном нажатии кнопки x_1 , лампа z_2 включается при каждом нечетном нажатии кнопки x_1 ; при каждом четном нажатии кнопки x_2 включаются обе лампы.
5	Устройство имеет два входа – x_1 , x_2 , управляемые кнопками, и два выхода – z_1 , z_2 , на которых установлены индикаторы (к примеру, лампы). Одновременное задание значений на обоих входах исключено. Алгоритм работы: лампа z_1 включается при каждом нечетном нажатии кнопки x_1 , лампа z_2 включается при каждом третьем нажатии кнопки x_1 ; при каждом четном нажатии кнопки x_2 обе лампы гаснут.

ПРИЛОЖЕНИЕ Б. СОСТАВ КУРСОВОЙ РАБОТЫ

Титульный лист

Задание

Введение

1. Описание исходных данных.
2. Построение исходной таблицы переходов и графа переходов.
3. Минимизация таблицы переходов.
4. Этапы синтеза избыточного конечного автомата.
 - 4.1. Выбор способа кодирования состояний автомата.
 - 4.2. Составление кодированной таблицы переходов.
 - 4.3. Выбор типа элементов памяти для реализации конечного автомата.
 - 4.4. Вычисление значений функций включения элементов памяти и выходов.
 - 4.5. Минимизация функций включения элементов памяти и выходов с помощью карт Карно.
 - 4.6. Минимизация функций включения элементов памяти и выходов с помощью SIS.
 - 4.7. Синтез схемы конечного автомата в выбранном элементном базисе.
5. Этапы синтеза автомата со схемой встроенного контроля.
 - 5.1. Расчет контрольной функции при контроле по коду паритета.
 - 5.2. Минимизация контрольной функции с помощью SIS.
 - 5.3. Синтез схемы конечного автомата со встроенной схемой контроля.
6. Этапы синтеза автомата с обнаружением одиночных неисправностей на выходах блока памяти.
 - 6.1. Выбор способа кодирования состояний конечного автомата с необходимым признаком для контроля.
 - 6.2. Составление кодированной таблицы переходов с доопределением неиспользуемых состояний.
 - 6.3. Выбор типа элементов памяти для реализации конечного автомата.
 - 6.4. Вычисление значений функций включения элементов памяти и выходов.
 - 6.5. Минимизация функций включения элементов памяти и выходов с помощью SIS.
 - 6.6. Синтез схемы конечного автомата с обнаружением неисправностей в выбранном элементном базисе.
7. Реализация конечного автомата с обнаружением одиночных неисправностей в Multisim.
 - 7.1. Реализация схемы конечного автомата.
 - 7.2. Результаты моделирования штатной работы (без неисправностей).
 - 7.3. Реализация схемы конечного автомата со встроенной схемой контроля.
 - 7.4. Результаты моделирования одиночных неисправностей выходов элементов логического преобразователя.

Заключение

Список использованных источников

Приложения

Оглавление

Введение.....	3
1. Функции алгебры логики.....	4
1.1. Функции алгебры логики и устройства автоматики	4
1.2. Элементарные функции алгебры логики.....	6
1.3. Аксиомы и законы алгебры логики	8
1.4. Конечные автоматы	9
2. Синтез синхронных автоматов с минимизацией количества состояний.....	11
2.1. Способы задания конечных автоматов.....	11
2.2. Минимизация таблиц переходов.....	14
2.3. Синтез неизбыточных синхронных автоматов	19
2.3.1. Кодирование таблицы переходов	19
2.3.2. Выбор элементов памяти и получение значений функций их включения	21
2.3.3. Минимизация функций включения элементов памяти и выходов с использованием карт Карно	24
2.3.4. Автоматизированная система минимизации функций алгебры логики ...	25
2.3.5. Схема синхронного конечного автомата	31
2.4. Синтез синхронных автоматов с обнаружением неисправностей.....	34
2.4.1. Принципы обнаружения неисправностей в дискретных устройствах.....	34
2.4.2. Структурные схемы организации контроля автоматов.....	35
2.4.3. Синтез конечных автоматов с обнаружением неисправностей.....	39
2.4.4. Синтез схем встроенного контроля.....	47
3. Анализ работы автомата в среде моделирования Multisim	49
3.1. Работа в среде моделирования Multisim.....	49
3.2. Синтез синхронного автомата, снабженного встроенной схемой контроля ...	59
3.3. Моделирование работы автомата при неисправностях элементов.....	62
3.3.1. Конечный автомат как объект диагностирования	62
3.3.2. Моделирование работы конечного автомата	63
Заключение	65
Библиографический список.....	66
<i>Приложение А. Варианты заданий</i>	<i>68</i>
<i>Приложение Б. Состав курсовой работы.....</i>	<i>79</i>

Учебное издание

ЕФАНОВ Дмитрий Викторович,
ПИВОВАРОВ Дмитрий Вячеславович

СИНТЕЗ ДИСКРЕТНЫХ УСТРОЙСТВ С ОБНАРУЖЕНИЕМ НЕИСПРАВНОСТЕЙ

Учебное пособие

Редактор и корректор *И. А. Шабранская*
Компьютерная верстка *М. С. Савастеевой*

План 2018 г., № 1

Подписано в печать с оригинал-макета 19.04.2019.
Формат 60×84¹/₁₆. Бумага для множ. апп. Печать ризография.
Усл. печ. л. 5,125. Тираж 200 экз.

Заказ

ФГБОУ ВО ПГУПС. 190031, СПб., Московский пр., 9.
Типография ФГБОУ ВО ПГУПС. 190031, СПб., Московский пр., 9.