Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина
Уральский энергетический институт

В. В. Куцин
К. Е. Нестеров
А. В. Кириллов

АВТОМАТИЗАЦИЯ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ

Учебное пособие

Министерство науки и высшего образования
Российской Федерации
Уральский федеральный университет
имени первого Президента России Б. Н. Ельцина

В. В. Куцин, К. Е. Нестеров, А. В. Кириллов

АВТОМАТИЗАЦИЯ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ

Учебное пособие

Под общей редакцией кандидата технических наук,
доцента А. В. Кириллова

Рекомендовано методическим советом
Уральского федерального университета
для студентов вуза, обучающихся по направлениям подготовки:
13.03.02 «Электроэнергетика и электротехника»,
13.04.02 «Электроэнергетика и электротехника»

Екатеринбург
Издательство Уральского университета
2024

УДК 621.31(075.8)
ББК 31.27-05я73
К95

Рецензенты:
кафедра электроэнергетики и транспорта Института инженерно-педагогического образования ФГАОУ ВО РГППУ (заведующий кафедрой кандидат педагогических наук, доцент *А. О. Прокубовская*); начальник «Научно-инженерного центра» АО «Автоматизированные системы и комплексы» кандидат технических наук, доцент *С. И. Шилин*

Куцин, Валерий Васильевич.

К95 Автоматизация технологических процессов : учебное пособие / В. В. Куцин, К. Е. Нестеров, А. В. Кириллов ; под общей редакцией кандидата технических наук, доцента А. В. Кириллова ; М-во науки и высшего образования РФ. — Екатеринбург : Изд-во Урал. ун-та, 2024. — 140 с. — ISBN 978-5-7996-3865-8. — Текст : непосредственный.

ISBN 978-5-7996-3865-8

Настоящее учебное пособие посвящено вопросам синтеза логических алгоритмов и их технической реализации для систем технологической автоматики. Также рассматриваются методы получения реальной циклограммы управляющего устройства с учетом используемой элементной базы и практическая реализация полученных алгоритмов.

Издание предназначено для студентов, обучающихся по направлениям подготовки 13.03.02 «Электроэнергетика и электротехника» и 13.04.02 «Электроэнергетика и электротехника», а также для персонала, работающего с современными системами автоматизации технологических процессов.

Библиогр.: 15 назв. Табл. 25. Рис. 69.

УДК 621.31(075.8)
ББК 31.27-05я73

ISBN 978-5-7996-3865-8

Оглавление

Введение

Большое число механизмов в металлургии и машиностроении работают в циклическом режиме, обеспечивая выпуск изделия в соответствии с заданной технологией (например, прокатные станы, многофункциональные металлообрабатывающие станки). Технологами задается последовательность работы каждого звена механизма в цикле работы всего механизма в виде циклограммы. Включение и отключение указанных звеньев осуществляется с помощью разного вида электромеханических устройств.

В учебном пособии рассматриваются основные определения и законы алгебры логики; синтез и техническая реализация комбинационных функций; синтез и техническая реализация последовательностных функций на основе циклограмм; получение рабочей циклограммы на основании исходных технологических данных; методы построения систем автоматики при использовании различных типов программируемых контроллеров, и их программирование; вопросы расчета надежности систем автоматики, методы повышения надежности при использовании резервирования. Особое внимание уделяется управляющим устройствам, выполненным на основе программируемых контроллеров различных типов. Заключительная глава пособия посвящена вопросам расчета показателей надежности систем технологической автоматики.

Для лучшего усвоения теоретического материала в конце каждой главы представлен список контрольных вопросов для самостоятельного решения. Также в конце издания приведен список используемой литературы.

Глава 1.
Синтез комбинационных функций

1.1. Основные определения

Существующие системы счисления (двоичные, восьмеричные, десятичные и т. п.) предназначены для решения арифметических и других задач и, за исключением двоичной, не являются удобными и рациональными для применения в вычислительных и логических устройствах. Двоичная система счисления имеет основанием число 2, для записи в этой системе используются две цифры — 0 и 1. Любое число в двоичной системе счисления записывается как комбинация нолей и единиц.

Несмотря на всю свою простоту, недостаток двоичной системы заключается в большой длине двоичных чисел при их записи, что требует большого внимания при вычислениях и увеличивает вероятность ошибок. Это обстоятельство делает систему непрактичной для расчетов вручную, но дает существенные преимущества при использовании ее в цифровых вычислительных машинах [1]. Достоинство двоичной системы счисления, заключающееся в простоте выполнения арифметических операций, дает возможность значительно упростить структуру арифметических и запоминающих устройств.

Действие автоматического устройства можно описать функциональными зависимостями между величинами на его входах и выходах, называемыми также сигналами, которые имеют дискретный характер. Такие устройства называют устройствами релейного действия. Релейные устройства, осуществляющие логические преобразования сигналов, называются логическими устройствами. В них каждая переменная может принимать лишь одно из двух возможных значений, обозначаемых цифрами 0 и 1.

Математическим аппаратом анализа и синтеза логических устройств служит двузначная алгебра логики (булева алгебра), которая изучает

связи между переменными, принимающими только два значения. Этим двум значениям ставятся в соответствие различные взаимоисключающие действия, условия или состояния в логических устройствах. Например: замыкание контакта — размыкание контакта; наличие сигнала — отсутствие сигнала; замкнутая цепь — разомкнутая цепь.

Необходимо подчеркнуть, что цифры 0 и 1 не выражают здесь количественных соотношений и являются не числами, а символами, следовательно, алгебра логических устройств является не алгеброй чисел, а алгеброй состояний. Управляющее логическое устройство обычно содержит следующие основные части [2]:

1. Входные элементы, воспринимающие входные воздействия от аппаратов управления и датчиков и преобразующие их с помощью согласующих элементов в сигналы, обозначаемые цифрами 0 и 1.

2. Промежуточные (логические) элементы, преобразующие входные сигналы в выходные в соответствии с заданной программой работы, причем выходные сигналы также могут принимать одно из двух значений, обозначаемые цифрами 0 и 1.

3. Усилители, повышающие мощность выходных сигналов.

4. Исполнительные элементы, воспринимающие выходные сигналы и выполняющие функции, для которых предназначено устройство. Ими являются контакторы, электромагниты, сигнальные лампы и т. п.

Элементы и сигналы (переменные) обозначаются обычно буквами латинского алфавита. В данном пособии применяются следующие обозначения:

- входные элементы — A, B, C;
- контакты входных элементов:
  - замыкающие — a, b, c;
  - размыкающие — $\bar{a}$, $\bar{b}$, $\bar{c}$;
- входные переменные — a, b, c;
- промежуточные элементы и промежуточные функции — P_1, P_2, P_3;
- промежуточные переменные — p_1, p_2, p_3;
- исполнительные элементы и выходные функции — Z, Y, X;
- контакты исполнительных элементов:
  - замыкающие — z, y, x;
  - размыкающие — $\bar{z}$, $\bar{y}$, $\bar{x}$;
- выходные переменные — z, y, x.

Однако данная система буквенных обозначений не является обязательной. При большом числе переменных используются порядковые номера: входные переменные — a_1, a_2, a_3, ... a_n; выходные переменные — x_1, x_2, x_3, ... x_m; промежуточные переменные — p_1, p_2, p_3, ... p_k.

1.2. Логические функции

Логической переменной называется величина, которая может принимать только два значения — 0 или 1. Логические переменные обозначаются буквами латинского алфавита.

Логической функцией (функцией алгебры логики) называется функция, которая, как и ее аргументы (логические переменные), может принимать только два значения — 0 или 1. Логические функции выражают зависимость выходных переменных от входных переменных. Эти функции в зависимости от числа входных переменных делятся на функции одной переменной, двух переменных и многих переменных.

Различные комбинации значений входных переменных в логических функциях называются наборами. Функция является полностью заданной, если указаны ее значения для всех наборов [3].

Например, три входные переменные, принимая значения 0 или 1, могут дать всего восемь различных сочетаний нулевых и единичных значений, т. е. восемь наборов. Сопоставляя каждому набору значение функции, равное 0 или 1, можно получить табличное задание данной функции (табл. 1.1). Такая таблица называется таблицей истинности, или таблицей соответствия.

Таблица 1.1

Таблица истинности логической функции

a	b	c	$f(a, b, c)$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Для упорядочения записи наборов в таблицах их можно записывать в виде целых положительных чисел в двоичной системе счисления, располагаемых в порядке их возрастания. Обозначим через m_n число различных наборов для n входных переменных.

Для одной входной переменной возможны два набора: 0 и 1, $m_1 = 2$.

Для двух переменных — четыре набора: 00, 01, 10, 11, $m_2 = m_1 \times 2 = 2 \times 2 = 2^2 = 4$.

Для трех переменных — восемь наборов: 000, 001, 010, 011, 100, 101, 110, 111, $m_3 = m_2 \times 2 = 4 \times 2 = 2^3 = 8$.

Для n входных переменных число различных наборов конечно и равно $m_n = 2^n$.

Задавая то или иное значение функции для каждого из 2^n наборов, можно тем самым задавать одну из возможных функций n переменных. Но поскольку каждому набору может соответствовать одно из двух значений (0 или 1) функции независимо от ее значений при других наборах, то число N различных логических функций от n переменных конечно и равно 2^{2^n}, поэтому прямое изучение логических функций с помощью таблиц практически возможно лишь до $n = 3$. Для описания логических функций от большего числа переменных используют принцип суперпозиции, который состоит в подстановке в функцию новых функций вместо аргументов. Например, функция от трех переменных $f(a, b, c)$ может быть преобразована в две функции от двух переменных $f[g(a, b), c]$ и $g(a, b)$. Поэтому в алгебре логики очень важною роль играют логические функции одной и двух переменных, с помощью которых, используя принцип суперпозиции, можно построить все логические функции от любого числа переменных.

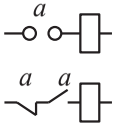
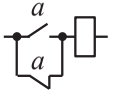
1.2.1. Функции одной переменной

Рассмотрим функции одной переменной $f(a)$. Поскольку входная переменная a может иметь значения 0 или 1, существуют четыре функции одной переменной (табл. 1.2). Функция $f_0 = 0$, называемая нулевой, и функция $f_3 = 1$, называемая единичной, не зависят от значений входной переменной и, следовательно, являются постоянными, или функциями-константами [3].

Функция $f_1(a) = \overline{a}$ называется инверсией, она является одной из основных в алгебре логики. Ее также называют отрицанием, дополнением или просто функцией НЕ. Функция $f_2(a) = a$ называется повторением, поскольку ее значение является повторением значения входной переменной. Ее называют еще тождеством или тождественностью.

Таблица 1.2

Функции одной переменной

Функция	Таблица истинности			Обозначение	Содержание логической функции	Формула	Контактная схема	Условное обозначение
	a	**1**	**0**					
Нулевая	f_0	0	0	0	Функция всегда имеет значение 0	$f_0(a)=0;$ $f_0(a)=a\bar{a}$		
Инверсия (функция НЕ)	f_1	0	1	$\bar{a}$	Функция имеет значение, обратное значению переменной	$f_1(a)=\bar{a}$		
Повторение	f_2	1	0	a	Функция повторяет значение переменной	$f_2(a)=a$		
Единичная	f_3	1	1	1	Функция всегда имеет значение 1	$f_3(a)=1;$ $f_3(a)=a+\bar{a}$		

1.2.2. Функции двух переменных

Функции двух переменных $f(a, b)$ являются основными функциями алгебры логики [4]. Четырем наборам двух входных переменных соответствует 16 возможных логических функций, которые приведены в табл. 1.3. (стр. 13). Можно заметить, что шесть из них встречались среди функций одной переменной. Функции f_0 и f_{15} являются соответственно нулевой и единичной функциями. Функции f_{10} и f_{12} являются функциями повторения и зависят каждая только от одной из двух входных переменных. Функции f_3 и f_5 есть инверсии одной из входных переменных, т. е. являются фактически функциями только одной переменной. Из остающихся десяти функций две (f_2 и f_{13}) не являются единственными, т. к. они отличаются от соответствующих им функций f_4 и f_{11} лишь порядком расположения входных переменных. Для оставшихся теперь восьми оригинальных функций двух переменных ($f_1, f_4, f_6, f_7, f_8, f_9, f_{11}, f_{14}$) можно указать некоторые их свойства, а также встречающиеся в литературе их названия и обозначения, выбрав для дальнейшего изложения те, которые наиболее соответствуют смыслу и содержанию логической функции и являются наиболее употребительными в технических приложениях алгебры логики.

Функция $f_1 = a \downarrow b$ называется стрелкой Пирса. Также ее называют инверсией суммы, функцией ИЛИ-НЕ, функцией НИ...НИ..., функцией Даггера, функцией Вебба. Иногда ее обозначают как $a \bar{\vee} b$. Для этой функции имеют место следующие соотношения:

$$a \downarrow 0 = \bar{a};\ \ a \downarrow 1 = 0;\ \ a \downarrow a = \bar{a};\ \ a \downarrow \bar{a} = 0.$$

Функцию $f_4 = a \leftarrow b$ в технических приложениях называют запретом. Иногда ее называют инверсией импликации. Для нее справедливы следующие соотношения:

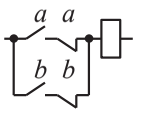
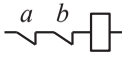
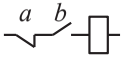
$$a \leftarrow 0 = a;\ \ a \leftarrow 1 = 0;\ \ a \leftarrow a = 0;\ \ a \leftarrow \bar{a} = a;\ \ 1 \leftarrow a = \bar{a}.$$

Функция $f_6 = a \otimes b$ называется неэквивалентностью. Также ее называют исключающим ИЛИ, неравнозначностью, альтернативой, сложением по модулю 2, разноименностью и обозначают посредством символов $\neq$, Δ, ∇. Для этой функции имеют место соотношения:

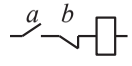
$$a \otimes 0 = a;\ \ a \otimes 1 = \bar{a};\ \ a \otimes a = 0;\ \ a \otimes \bar{a} = 1.$$

Таблица 1.3

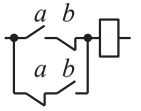
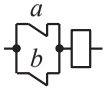
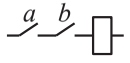
Функции двух переменных

Функция	Таблица истинности					Символическое обозначение	Содержание логической функции	Структурная формула	Контактная схема	Условное обозначение
	a	**1**	**1**	**0**	**0**					
	b	**1**	**0**	**1**	**0**					
Нулевая	f_0	0	0	0	0	0	Функция всегда равна нулю, вне зависимости от значений переменных	$a\bar{a}+b\bar{b}$		
ИЛИ-НЕ (Стрелка Пирса)	f_1	0	0	0	1	$a \downarrow b$	Функция имеет значение 1 тогда и только тогда, когда обе переменные равны 0	$\overline{a+b}=\bar{a}\bar{b}$		
Запрет *a*	f_2	0	0	1	0	$b \leftarrow a$	Функция имеет значение 0, если переменная *a* равна 1, вне зависимости от значения переменной *b*. Функция повторяет значение переменной *b*, если переменная *a* равна 0	$\bar{a}b$		

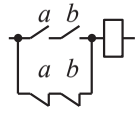
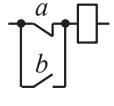
Продолжение табл. 1.3

Функция	Таблица истинности					Символическое обозначение	Содержание логической функции	Структурная формула	Контактная схема	Условное обозначение
	a	**1**	**1**	**0**	**0**					
	b	**1**	**0**	**1**	**0**					
НЕ *a* (инверсия *a*)	f_3	0	0	1	1	$\overline{a}$	Функция имеет значение, обратное значению переменной *a*, и не зависит от значения переменной *b*	$\overline{a}$		
Запрет *b*	f_4	0	1	0	0	$a \leftarrow b$	Функция имеет значение 0, если переменная *b* равна 1, вне зависимости от значения переменной *a*. Функция повторяет значение переменной *a*, если переменная *b* равна 0	$a\overline{b}$		
НЕ *b* (инверсия *b*)	f_5	0	1	0	1	$\overline{b}$	Функция имеет значение, обратное значению переменной *b*, и не зависит от значения переменной *a*	$\overline{b}$		

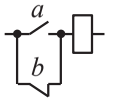
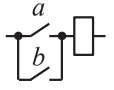
Продолжение табл. 1.3

Функция	Таблица истинности					Символическое обозначение	Содержание логической функции	Структурная формула	Контактная схема	Условное обозначение
	a	**1**	**1**	**0**	**0**					
	b	**1**	**0**	**1**	**0**					
Неэквивалентность (исключающее ИЛИ)	f_6	0	1	1	0	$a \otimes b$	Функция имеет значение 1 тогда и только тогда, когда либо переменная a, либо переменная b имеет значение 1 (но не обе вместе)	$\bar{a}b + a\bar{b}$		
И-НЕ (Штрих Шеффера)	f_7	0	1	1	1	a / b	Функция имеет значение 0 тогда и только тогда, когда обе переменные имеют значение 1	$\overline{ab}$ $\bar{a} + \bar{b}$		
И (конъюнкция)	f_8	1	0	0	0	ab	Функция имеет значение 1 тогда и только тогда, когда и переменная a, и переменная b имеют значение 1	ab		

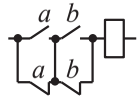
Продолжение табл. 1.3

Функция	Таблица истинности					Символическое обозначение	Содержание логической функции	Структурная формула	Контактная схема	Условное обозначение
	a	**1**	**1**	**0**	**0**					
	b	**1**	**0**	**1**	**0**					
Эквивалентность (равнозначность)	f_9	1	0	0	1	$a \sim b$	Функция имеет значение 1 тогда и только тогда, когда обе переменные имеют одинаковое значение, и значение 0, когда переменные имеют разные значения	$ab + \bar{a}\bar{b}$		
Повторение *b*	f_{10}	1	0	1	0	b	Функция повторяет значение переменной *b* независимо от значения переменной *a*	b		
Импликация *b*	f_{11}	1	0	1	1	$a \to b$	Функция имеет значение 0 тогда и только тогда, когда переменная *a* имеет значение 1, а переменная *b* имеет значение 0	$\bar{a} + b$		

Продолжение табл. 1.3

Функция	Таблица истинности					Символическое обозначение	Содержание логической функции	Структурная формула	Контактная схема	Условное обозначение
	a	**1**	**1**	**0**	**0**					
	b	**1**	**0**	**1**	**0**					
Повторение *a*	f_{12}	1	1	0	0	a	Функция повторяет значение переменной *a* независимо от значения переменной *b*	a		
Импликация *a*	f_{13}	1	1	0	1	$b \to a$	Функция имеет значение 0 тогда и только тогда, когда переменная *b* имеет значение 1, а переменная *a* имеет значение 0	$a + \bar{b}$		
ИЛИ (дизъюнкция)	f_{14}	1	1	1	0	$a + b$	Функция равна 0 тогда и только тогда, когда обе переменные равны 0. Функция равна 1, когда или переменная *a* или переменная *b*, или обе вместе равны 1	$a + b$		

Окончание табл. 1.3

Функция	**Таблица истинности**					**Символическое обозначение**	**Содержание логической функции**	**Структурная формула**	**Контактная схема**	**Условное обозначение**
	a	**1**	**1**	**0**	**0**					
	b	**1**	**0**	**1**	**0**					
Единичная	f_{15}	1	1	1	1	1	Функция всегда равна единице, вне зависимости от значений переменных	$(a+\bar{a})(b+\bar{b})$		

Функция $f_7 = a / b$ называется штрихом Шеффера, а также инверсией произведения, функцией И-НЕ, несовместностью. Для нее справедливы следующие соотношения:

$$a / 0 = 1;\ \ a / 1 = \overline{a};\ \ a / a = \overline{a};\ \ a / \overline{a} = 1.$$

Функция $f_8 = a \cdot b$ называется конъюнкцией. Также ее называют произведением, логическим умножением, функцией И, пересечением и обозначают с помощью символов $\wedge$, $\cap$, &. Для этой функции справедливы следующие соотношения:

$$a \cdot 0 = 0;\ \ a \cdot 1 = a;\ \ a \cdot a = a;\ \ a \cdot \overline{a} = 0.$$

Функция $f_9 = a \sim b$ называется эквивалентностью, или равнозначностью. Для ее обозначения используют также символы $\equiv$, $\rightleftarrows$. Для нее имеют место соотношения:

$$a \sim 0 = \overline{a};\ \ a \sim 1 = a;\ \ a \sim a = 1;\ \ a \sim \overline{a} = 0.$$

Функция $f_{11} = a \to b$ называется импликацией (иногда включением). Для нее справедливы соотношения:

$$a \to 0 = \overline{a};\ \ a \to 1 = 1;\ \ a \to a = 1;\ \ a \to \overline{a} = \overline{a};\ \ 0 \to a = 1;\ \ 1 \to a = a.$$

Функция $f_{14} = a + b$ называется дизъюнкцией, а также суммой, логическим сложением, функцией ИЛИ, объединением. Для ее обозначения применяют также символы $\vee$, $\cup$. Для дизъюнкции справедливы следующие равенства:

$$a + 0 = a;\ \ a + 1 = 1;\ \ a + a = a;\ \ a + \overline{a} = 1.$$

Анализируя табл. 1.3, можно выделить самостоятельные функции, с помощью которых реализовывают реальную систему управления [5]:

- $f_1 = \overline{a + b} = \overline{a}\,\overline{b}$ — функция ИЛИ-НЕ;
- $f_2 = a\overline{b}$, $f_4 = \overline{a}b$ — запрет;
- $f_6 = a\overline{b} + \overline{a}b$ — неэквивалентность (неравнозначность);
- $f_7 = \overline{ab} = \overline{a} + \overline{b}$ — функция И-НЕ;
- $f_8 = ab$ — конъюнкция — функция И;
- $f_9 = ab + \overline{a}\,\overline{b}$ — эквивалентность (равнозначность);

– $f_{11} = \overline{a} + b$, $f_{13} = a + \overline{b}$ — импликация;
– $f_{14} = a + b$ — дизъюнкция — функция ИЛИ.

К этому списку необходимо добавить инверсию (функцию НЕ) — $f = \overline{a}$, рассмотренную в предыдущей таблице (1.2).

1.2.3. Аксиомы алгебры логики

Аксиомы — постулаты, принимаемые за истину без доказательств [3]. В алгебре логики используются следующие аксиомы:

1. Существуют такие 0 и 1, что

$$\overline{0} = 1;$$
$$\overline{1} = 0.$$

2. Переменная может принимать лишь одно из двух возможных значений:

$$a = 0, \text{ если } a \neq 1;$$
$$a = 1, \text{ если } a \neq 0.$$

3.
$$0 \cdot 0 = 0;$$
$$1 + 1 = 1.$$

4.
$$1 \cdot 1 = 1;$$
$$0 + 0 = 0.$$

5.
$$1 \cdot 0 = 0;$$
$$0 + 1 = 1 + 0 = 1.$$

Каждая из представленных аксиом состоит из двух частей, что соответствует правилу инверсии, которое заключается в том, что любая аксиома может быть преобразована в другую аксиому одновременной заменой цифры 0 на цифру 1 и операции конъюнкции на дизъюнкцию и наоборот [6].

На основе этих аксиом выводятся все теоремы, выражающие основные законы алгебры логики [3].

1.2.4. Законы алгебры логики

1. Законы нулевого множества:

$$0 + a = a;$$
$$0 \cdot a = 0;$$
$$0 \cdot a \cdot b \cdot c \cdot \ldots \cdot w = 0,$$

т. е. конъюнкция любого числа переменных обращается в нуль, если какая-либо одна переменная имеет значение 0, независимо от значений других переменных.

2. Законы универсального множества:

$$1 \cdot a = a;$$
$$1 + a = 1;$$
$$1 + a + b + c + \ldots + w = 1,$$

т. е. дизъюнкция любого числа переменных обращается в единицу, если хотя бы одна из ее переменных имеет значение 1, независимо от значений других переменных.

3. Законы идемпотентности (повторения, тавтологии):

$$aa = a;$$
$$aa \ldots a = a^n = a;$$
$$a + a = a$$
$$a + a + \ldots + a = na = a.$$

4. Закон двойной инверсии:

$$\bar{\bar{a}} = a,$$

т. е. двойную инверсию можно убрать.

5. Законы дополнительности:
логическое противоречие:

$$a\bar{a} = 0,$$

т. е. конъюнкция любой переменной и ее инверсии есть 0;
закон исключения третьего:

$$a + \bar{a} = 1,$$

т. е. дизъюнкция любой переменной и ее инверсии есть 1.

6. Коммутативные (переместительные) законы:

$$ab = ba;$$
$$a + b = b + a,$$

т. е. результаты выполнения операций конъюнкции и дизъюнкции не зависят от того, в каком порядке следуют переменные.

7. Ассоциативные (сочетательные) законы:

$$a(bc) = (ab)c = abc;$$
$$a + (b + c) = (a + b) + c = a + b + c,$$

т. е. для записи конъюнкции или дизъюнкции скобки можно опустить.

8. Дистрибутивные (распределительные) законы:
конъюнкции относительно дизъюнкции:

$$a(b + c) = ab + ac;$$

дизъюнкции относительно конъюнкции:

$$a + bc = (a + b)(a + c).$$

9. Законы поглощения:

$$a(a + b) = a;$$
$$a(a + b)(a + c)\ldots(a + w) = a;$$
$$a + ab = a;$$
$$a + ab + ac + \ldots aw = a.$$
$$a(\bar{a} + b) = ab;$$
$$a + \bar{a}b = a + b.$$

10. Закон склеивания (распространения):

$$ab + a\bar{b} = a;$$
$$(a + b)(a + \bar{b}) = a.$$

11. Закон обобщенного склеивания:

$$ab + \bar{a}c + bc = ab + \bar{a}c;$$
$$(a + b)(\bar{a} + c)(b + c) = (a + b)(\bar{a} + c);$$
$$(a + b)(\bar{a} + c) = ac + \bar{a}b.$$

12. Законы де Моргана (законы инверсии, законы двойственности): для двух переменных:

$$\overline{ab} = \overline{a} + \overline{b},$$

т. е. инверсия конъюнкции есть дизъюнкция инверсий;

$$\overline{a + b} = \overline{a}\,\overline{b},$$

т. е. инверсия дизъюнкции есть конъюнкция инверсий;
для n переменных:

$$\overline{abc\ldots w} = \overline{a} + \overline{b} + \overline{c} + \ldots + \overline{w};$$
$$\overline{a + b + c + \ldots + w} = \overline{a}\,\overline{b}\,\overline{c}\ldots\overline{w};$$

обобщение законов де Моргана, предложенное Шенноном,

$$\overline{f\left(a,b,c,\ldots,w,\cdot,+\right)} = f\left(\overline{a},\overline{b},\overline{c},\ldots,\overline{w},+,\cdot\right),$$

т. е. инверсия любой функции получается заменой каждой переменной ее инверсией и одновременно взаимной заменой символов конъюнкции и дизъюнкции.

Справедливость любого закона алгебры логики можно доказать разными методами. Законы 1–5 доказываются путем прямой подстановки вместо двоичной переменной a значений 0 или 1, что непосредственно приводит к принятым аксиомам.

Формальный (аналитический) метод доказательства законов состоит в том, что справедливость каждого доказывается на основе аксиом и ранее доказанных законов. Доказательство заключается в приведении обеих частей выражения к одному виду с помощью последовательных преобразований. Приведенные ниже доказательства некоторых законов можно рассматривать также как примеры равносильных преобразований функций.

Закон поглощения. Для левой части на основании распределительного закона можно записать

$$a\left(a + b\right) = aa + ab.$$

Затем на основании закона повторения:

$$aa + ab = a + ab;$$

на основании закона универсального множества:

$$a + ab = a \cdot 1 + ab;$$

на основании распределительного закона:

$$a \cdot 1 + ab = a(1 + b);$$

на основании закона универсального множества:

$$a(1 + b) = a \cdot 1 = a.$$

Следовательно, $a(a + b) = a$.

Ряд законов доказывается методом перебора всех возможных значений, который состоит в том, что составляются всевозможные комбинации (наборы) значений переменных, и для них проверяется справедливость закона. Доказательство ведется обычно в табличной форме путем построения и сравнения таблиц истинности. Для доказательства законов достаточно показать тождественность выражений, образующих левую и правую стороны доказываемого соотношения при всех наборах переменных, принимающих значения 0 или 1.

Закон де Моргана. Для доказательства построена совмещенная таблица истинности (табл. 1.4).

Таблица 1.4

Доказательство закона де Моргана

a	b	$\overline{ab}$	$\bar{a} + \bar{b}$	$\overline{a + b}$	$\bar{a}\bar{b}$
0	0	1	1	1	1
0	1	1	1	0	0
1	0	1	1	0	0
1	1	0	0	0	0

Совпадение значений левой и правой частей соотношения при одинаковых наборах переменных доказывает справедливость этих законов.

При построении бесконтактных систем управления необходим набор логических элементов, соответствующих всему ряду выделенных при анализе таблиц 1.1 и 1.2 функций (их 9). Однако достаточно иметь три элемента — И, ИЛИ, НЕ, чтобы реализовать систему управления любой сложности.

Например, функция эквивалентность $f_9 = ab + \bar{a}\bar{b}$ включает в себя две инверсии ($\bar{a}$, $\bar{b}$), две конъюнкции (ab, $\bar{a}\bar{b}$) и одну дизъюнкцию ($ab + \bar{a}\bar{b}$).

Набор элементов, реализующих функции И, ИЛИ, НЕ, является избыточным базисом. Если использовать закон де Моргана, то от избыточного базиса можно перейти к двум вариантам нормального базиса — И, НЕ и ИЛИ, НЕ. Наконец, можно любую систему реализовать, используя только элементы И-НЕ или только элементы ИЛИ-НЕ. Рассмотрим на примере функции эквивалентности всевозможные варианты ее реализации (рис. 1.1–1.5). Для преобразования функции используются законы де Моргана и двойной инверсии. Двойную инверсию можно добавить над преобразуемой функцией.

1. Реализация функции «Эквивалентность» с использованием элементов И, ИЛИ, НЕ:

$$x = ab + \bar{a}\bar{b}$$

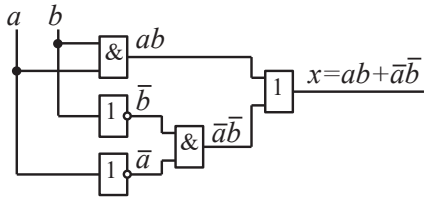


Рис. 1.1. Функция «Эквивалентность» на базе элементов И, ИЛИ, НЕ

2. Реализация функции «Эквивалентность» с использованием элементов И, НЕ:

$$x = \overline{\overline{ab + \bar{a}\bar{b}}} = \overline{\overline{ab}\,\overline{\bar{a}\bar{b}}}$$

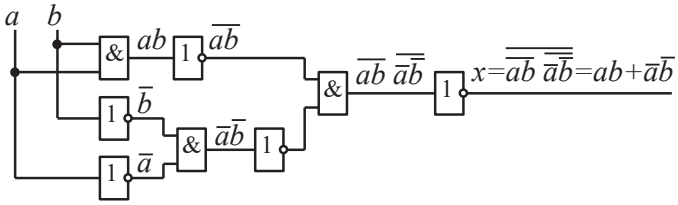


Рис. 1.2. Функция «Эквивалентность» на базе элементов И, НЕ

3. Реализация функции «Эквивалентность» с использованием элементов ИЛИ, НЕ:

$$x = ab + \bar{a}\bar{b} = \overline{\overline{ab}} + \overline{\overline{\bar{a}\bar{b}}} = \overline{\overline{a} + \overline{b}} + \overline{\overline{a}} + \overline{\overline{b}} = \overline{\overline{a} + \overline{b}} + \overline{a + b}$$

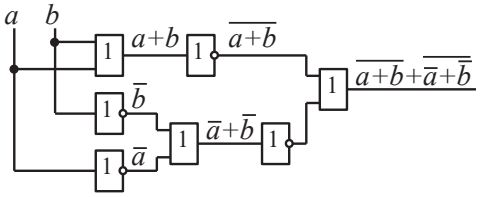


Рис. 1.3. Функция «Эквивалентность»
на базе элементов ИЛИ, НЕ

4. Реализация функции «Эквивалентность» с использованием только элементов И-НЕ:

$$x = \overline{\overline{ab + \bar{a}\bar{b}}} = \overline{\overline{ab}\,\overline{\bar{a}\bar{b}}}$$

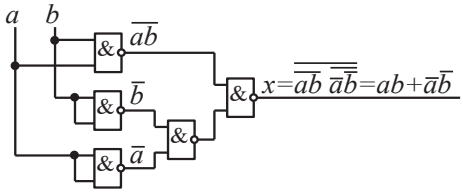


Рис. 1.4. Функция «Эквивалентность»
на базе элементов И-НЕ

5. Реализация функции «Эквивалентность» с использованием элементов ИЛИ-НЕ:

$$x = ab + \bar{a}\bar{b} = \overline{\overline{ab}} + \overline{\overline{\bar{a}\bar{b}}} = \overline{\overline{a} + \overline{b}} + \overline{\overline{a}} + \overline{\overline{b}} = \overline{\overline{\overline{a} + \overline{b}} + \overline{a + b}}$$

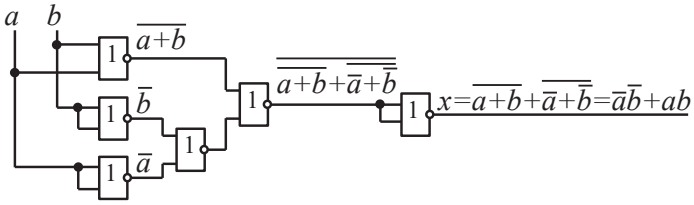


Рис. 1.5. Функция «Эквивалентность»
на базе элементов ИЛИ-НЕ

Сравнивая все варианты реализации алгоритма, следует отметить, что наименьшее число используемых элементов требуют варианты 1 (И, ИЛИ, НЕ) и 4 (И-НЕ). Преимущества варианта 1 — простота проектирования и наладки, кроме того, упрощается контроль и диагностика устройства при эксплуатации. Остальные варианты на практике реализуются в тех случаях, когда в имеющейся элементной базе отсутствуют элементы И или ИЛИ.

1.3. Нормальные и совершенные нормальные формы логических функций

1.3.1. Элементарные конъюнкция и дизъюнкция

Элементарной конъюнкцией (дизъюнкцией) называется конъюнкция (дизъюнкция) любого числа различных независимых переменных, входящих в данную конъюнкцию (дизъюнкцию) с инверсией или без нее не более одного раза [3]. Например, выражения

$$abc,\ \ \bar{a}b\bar{c},\ \ \bar{a}\,\bar{c},\ \ \bar{a},\ \ b,\ \ c\cdot 1$$

являются элементарными конъюнкциями, а выражения

$$\left(a+\bar{b}\right)c,\ \ ab\bar{b}c,\ \ \overline{ac}$$

не являются элементарными конъюнкциями. Аналогично выражения

$$a+b+c,\ \ \bar{a}+\bar{c},\ \ \bar{a}+b+c,\ \ \bar{a},\ \ b,\ \ d+0$$

являются элементарными дизъюнкциями, а выражения

$$ab+c,\ \ a+c+\bar{a},\ \ a+b+a,\ \ \overline{a+\bar{b}+c}$$

не являются элементарными дизъюнкциями.

1.3.2. Дизъюнктивная и конъюнктивная нормальные формы

Дизъюнкция любого числа элементарных конъюнкций называется дизъюнктивной нормальной формой (ДНФ). Например,

$$a+bc+\bar{a}bc+\bar{a}\bar{b}c.$$

Конъюнкция любого числа элементарных дизъюнкций называется конъюнктивной нормальной формой (КНФ). Например,

$$a\left(a+b\right)\left(\overline{b}+c\right)\left(\overline{a}+b+\overline{c}\right).$$

Для каждой функции может существовать несколько дизъюнктивных и конъюнктивных нормальных форм (являющихся равносильными друг другу). Инверсия любой функции, записанной в дизъюнктивной (конъюнктивной) нормальной форме, может быть записана в конъюнктивной (дизъюнктивной) нормальной форме путем замены знаков «+» на «·» и «·» на «+» и инвертирования каждой переменной. Например, инверсия функции

$$f = a + b\overline{c} + \overline{a}\overline{b}c$$

имеет вид:

$$\overline{f} = \overline{a}\left(\overline{b}+c\right)\left(a+b+\overline{c}\right).$$

Логическую функцию, заданную любым аналитическим выражением, можно непосредственно преобразовать к нормальной дизъюнктивной (или конъюнктивной) форме [3]. Для этого необходимо:

– выразить все операции через операции конъюнкции, дизъюнкции и инверсии;

– избавиться от инверсии над целыми выражениями, перейдя к форме, в которой имеются инверсии только отдельных переменных;

– раскрыть скобки, применяя дистрибутивный закон;

– привести конъюнкции (дизъюнкции) к элементарным.

Например,

$$f = a\overline{\left(a+c\right)\left(\overline{c}+\overline{a}\right)}\left(\overline{b}+\overline{c}\right)+a\overline{c} = a\overline{\left(a\overline{c}+a\overline{a}+c\overline{c}+\overline{a}c\right)}\left(\overline{b}+\overline{c}\right)+a\overline{c} =$$

$$= a\overline{\left(a\overline{c}+\overline{a}c\right)}\left(\overline{b}+\overline{c}\right)+a\overline{c} = a\left(\overline{a}+c\right)\left(a+\overline{c}\right)\left(\overline{b}+\overline{c}\right)+a\overline{c} =$$

$$= \left(a\overline{a}a+a\overline{a}\,\overline{c}+aca+ac\overline{c}\right)\left(\overline{b}+\overline{c}\right)+a\overline{c} = a\overline{b}c+ac\overline{c}+a\overline{c} = a\overline{b}c+a\overline{c}.$$

1.3.3. Конституенты единицы и нуля

Если считать в общем случае единицу функцией n переменных, то ее можно разложить на 2^n конституентов (составляющих), т. е. на 2^n дизъюнктивно связанных элементарных конъюнкций n-го ранга,

т. к. количество различных наборов n двоичных переменных равно 2^n. Например, при $n = 2$

$$1 = ab + a\bar{b} + \bar{a}b + \bar{a}\bar{b}.$$

Конституентом единицы для данного числа переменных называется всякая конъюнкция всех переменных, взятых по одной с инверсией или без нее [3]. Например, для трех переменных a, b, c конституентами единицы будут конъюнкции abc, $\bar{a}bc$, $\bar{a}\bar{b}\bar{c}$ и так далее.

Конституент единицы для некоторого набора строится следующим образом: каждая переменная входит в него без инверсии, если значение этой переменной в данном наборе равно единице, или с инверсией, если значение этой переменной в данном наборе равно нулю. Например, набору 00111 соответствует конституент единицы $\bar{a}\bar{b}cde$. Отсюда следует: каждый конституент единицы, как функция переменных, равняется единице только на одном соответствующем ему наборе. На всех остальных наборах данный конституент единицы равен нулю. Действительно, если взять другой набор 00011, то для него значение вышеприведенного конституента единицы $\bar{a}\bar{b}cde$ равно нулю.

Если считать в общем случае нуль функцией n переменных, то его можно разложить на 2^n конституентов (составляющих), т. е. на 2^n конъюнктивно связанных элементарных дизъюнкций n-го ранга, т. к. количество различных наборов n двоичных переменных равно 2^n. Например, при $n = 2$

$$0 = (a + b)(a + \bar{b})(\bar{a} + b)(\bar{a} + \bar{b}).$$

Конституентом нуля для данного числа переменных называется всякая дизъюнкция всех переменных, взятых по одной с инверсией или без нее [3]. Например, для трех переменных a, b, c конституентами нуля будут дизъюнкции $(a + b + c)$, $(\bar{a} + b + c)$, $(\bar{a} + \bar{b} + c)$ и так далее.

Конституент нуля для некоторого набора значений строится следующим образом: каждая переменная в него входит без инверсии, если значение этой переменной в данном наборе равно нулю, или с инверсией, если значение этой переменной в данном наборе равно единице. Например, набору 00111 соответствует конституент нуля $a + b + \bar{c} + \bar{d} + \bar{e}$. Отсюда следует: каждый конституент нуля, как функция n переменных, равняется нулю только на одном соответствующем ему наборе.

На всех остальных наборах данный конституент нуля равен единице. Действительно, для другого набора 00011 значение вышеприведенного конституента нуля $a+b+\overline{c}+\overline{d}+\overline{e}$ равно единице.

1.3.4. Совершенные дизъюнктивная и конъюнктивная нормальные формы

Существует один вид дизъюнктивной нормальной формы и один вид конъюнктивной нормальной формы, в котором функция может быть записана только единственным образом. Они называются совершенными нормальными формами. В совершенной дизъюнктивной (конъюнктивной) нормальной форме:

– каждая элементарная конъюнкция (дизъюнкция) включает все переменные (с инверсиями или без них);

– нет одинаковых конъюнкций (дизъюнкций).

Совершенной дизъюнктивной нормальной формой (СДНФ) данной логической функции называется дизъюнкция конституентов единицы тех наборов значений переменных, где данная функция равна единице. Например, из таблицы истинности (табл. 1.5) для функции $f=a+b$ следует ее запись в СДНФ:

$$f=\overline{a}b+a\overline{b}+ab.$$

Таблица 1.5

Таблица истинности для функции $f=a+b$

a	b	f
0	0	0
0	1	1
1	0	1
1	1	1

Выше было показано, что любую функцию можно привести к ДНФ (или к КНФ). От любой ДНФ можно перейти к СДНФ функции при помощи равносильных преобразований. Такой переход называется развертыванием. Для этого необходимо:

– ввести недостающие переменные в каждую конъюнкцию умножением ее на равносильность вида $a+\overline{a}=1$, где a — недостающая переменная;

– раскрыть скобки, применяя коммутативный закон ($ab = ba$);

– избавиться от повторяющихся конъюнкций на основании закона идемпотентности ($a + a = a$).

Для примера, который представлен в п. 1.3.2, была получена ДНФ в виде:

$$f = a\bar{b}c + a\bar{c}.$$

Следуя изложенному правилу перехода к СДНФ, получаем:

$$f = a\bar{b}c + a\bar{c}\left(b + \bar{b}\right) = a\bar{b}c + ab\bar{c} + a\bar{b}\bar{c}.$$

Совершенной конъюнктивной нормальной формой (СКНФ) данной логической функции называется конъюнкция конституентов нуля тех наборов, где данная функция равна нулю.

Например, из таблицы истинности (табл. 1.6) для функции $f = ab$ следует ее запись в СКНФ:

$$f = \left(a + b\right)\left(a + \bar{b}\right)\left(\bar{a} + b\right).$$

Таблица 1.6

Таблица истинности для функции $f = ab$

a	***b***	***f***
0	0	0
0	1	0
1	0	0
1	1	1

Переход от КНФ к СКНФ осуществляется по аналогии с переходом от ДНФ к СДНФ. Для этого необходимо:

– ввести недостающие переменные в каждую дизъюнкцию, используя закон противоречия $a\bar{a} = 0$ (a — недостающая переменная);

– выполнить преобразования, применяя второй закон дистрибутивности $a + bc = \left(a + b\right)\left(a + c\right)$ и коммутативный закон $a + b = b + a$;

– избавиться от повторяющихся дизъюнкций на основании закона идемпотентности $aa = a$.

Например, развертывание КНФ вида

$$f = \left(a + b\right)\left(\bar{b} + c\right)\left(\bar{a} + \bar{c}\right)$$

в СКНФ осуществляется следующим образом:

$$f = (a + b + c\bar{c})(a\bar{a} + \bar{b} + c)(\bar{a} + b\bar{b} + \bar{c}) =$$
$$= (a + b + c)(a + b + \bar{c})(a + \bar{b} + c)(\bar{a} + \bar{b} + c)(\bar{a} + b + \bar{c})(\bar{a} + \bar{b} + \bar{c}).$$

Совершенные нормальные формы обладают следующими особенностями:

– если при каком-то наборе функция равна единице, то в СДНФ только один из ее членов принимает единичное значение;

– если функция для данного набора равна нулю, то в СДНФ ни один из членов не принимает единичного значения;

– если для данного набора функция равна нулю, то в СКНФ только один из членов принимает нулевое значение;

– если для данного набора функция равна единице, то в СКНФ ни один из членов не принимает нулевого значения.

Отсюда следует: СДНФ имеет столько членов, сколько единиц в таблице истинности; СКНФ имеет столько членов, сколько нулей в таблице истинности.

Необходимо отметить, что сумма числа членов СДНФ и числа членов СКНФ равна 2^n, где n — число переменных данной функции. Очевидно, что если функция тождественно-истинная, т. е. в ее таблице истинности стоят только одни единицы, то число членов СДНФ будет равно 2^n, а число членов СКНФ будет равно нулю. Если же функция тождественно-ложная, то число членов СДНФ будет равно нулю, а число членов СКНФ будет равно 2^n.

Имея в виду правила получения конституентов СДНФ и СКНФ, можно заметить, что дизъюнкции СКНФ могут быть получены инвертированием конъюнкций, не вошедших в СДНФ, т. е. конъюнкций, на которых функция принимает нулевое значение. Поэтому для того, чтобы получить СКНФ из СДНФ, необходимо:

– найти не вошедшие в СДНФ конъюнкции;

– составить их дизъюнкцию;

– взять инверсию от этой дизъюнкции.

Например, имея СДНФ вида $f = ab + a\bar{b}$, для получения СКНФ необходимо:

– найти не вошедшие в СДНФ конъюнкции: $\bar{a}b$, $\bar{a}\bar{b}$;

– составить их дизъюнкцию: $\bar{a}b + \bar{a}\bar{b}$;

– взять инверсию от этой дизъюнкции: $\overline{\bar{a}b + \bar{a}\bar{b}} = \overline{\bar{a}b}\,\overline{\bar{a}\bar{b}} = (a + \bar{b})(a + b)$.

Полученное выражение и есть СКНФ исходной функции.

Аналогичным образом можно перейти от СКНФ к СДНФ.

1.4. Применение карт Карно для изображения и преобразования логических функций

1.4.1. Понятие о карте Карно

Рассмотрим логическую функцию двух входных переменных $f(a,b)$.

Две входные логические переменные *a* и *b* могут образовать всего четыре возможных набора. Каждому набору можно поставить в соответствие клетку карты Карно [1, 3, 4]. Квадратная скобка с именем входной переменной обозначает область карты (столбцы или строки), в которой данная переменная равна единице. В оставшейся части карты эта переменная равна нулю. Для пояснения того, что написано выше, на рис. 1.6 показана карта Карно, заполненная значениями входных переменных. Обычно эти значения не пишутся, а подразумеваются, вместо них карта Карно заполняется значениями самой функции (0 или 1) для соответствующего набора входных переменных.

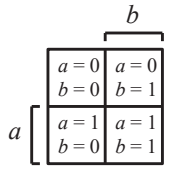


Рис. 1.6. Карта Карно для функции двух переменных

Следует отметить, что каждая из входных переменных делит карту на две равные части, в одной из которых значение этой переменной равно 1, а в другой — 0.

1.4.2. Составление карт Карно по таблицам истинности

В таблицах истинности, с помощью которых задаются логические функции, число строк равно числу всех возможных наборов (2^n). Поэтому таблицы истинности для функций более чем двух переменных становятся громоздкими. Изображение логических функций посредством карт Карно является более компактным, т. к. каждому набору в ней соответствует клетка, а не строка.

Карта Карно может составляться непосредственно по таблице истинности. Для этого строится карта с числом клеток, равным числу строк таблицы. По внешним сторонам карты определенным образом располагаются входные переменные. Для каждого набора (строки) таблицы отыскивается соответствующий набор (клетка) карты. В эту клетку проставляется значение функции для данного набора. Например,

для функции трех переменных $f(a,b,c)$, заданной таблицей истинности (табл. 1.7), карта Карно имеет вид, показанный на рис. 1.7.

Таблица 1.7

Функция $f(a, b, c)$

a	*b*	*c*	*f*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

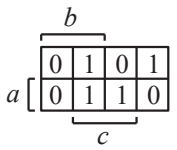


Рис. 1.7. Карта Карно для функции $f(a, b, c)$

Число клеток карты Карно определяется величиной 2^n, где n равно числу входных переменных. Отсюда следует, что добавление каждой новой переменной удваивает число клеток, т. е. увеличивает карту вдвое. При этом, как будет показано далее, новые переменные должны располагаться так, чтобы иметь общую площадь со всеми прежними переменными.

1.4.3. Составление карт Карно по алгебраическим выражениям

Карта Карно для логической функции, заданной алгебраическим выражением, может быть составлена в следующем порядке:

– по числу переменных, входящих в выражение заданной функции, строится карта Карно и располагаются переменные;

– заданное алгебраическое выражение приводится к СДНФ;

– в карте Карно для каждого конституента единицы СДНФ находится соответствующая клетка (с таким же набором переменных), в которую записывается 1, в остальные клетки карты записываются 0.

В качестве примера рассмотрим логическую функцию, заданную алгебраическим выражением $f = \bar{a}c + bc + \bar{a}b$. Эта функция является

функций трех переменных, и ее карта должна иметь $2^3 = 8$ клеток. Приведем функцию к СДНФ:

$$f = \bar{a}\left(b + \bar{b}\right)c + \left(a + \bar{a}\right)bc + \bar{a}b\left(c + \bar{c}\right) =$$

$$= \bar{a}bc + \bar{a}\bar{b}c + abc + \bar{a}bc + \bar{a}bc + \bar{a}b\bar{c} = \bar{a}bc + \bar{a}\bar{b}c + abc + \bar{a}b\bar{c}.$$

Функция имеет значения, равные 1, в клетках карты, соответствующих конституентам СДНФ, т. е. карта Карно для заданной функции имеет вид, представленный на рис. 1.8.

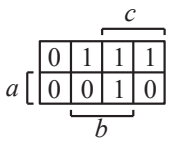


Рис. 1.8. Карта Карно для функции $f = \bar{a}c + bc + \bar{a}b$

Можно воспользоваться приведением исходного выражения к СКНФ. В этом случае в клетки карты, соответствующие конституентам нуля, записывается 0, в остальные — 1.

Основное применение карты Карно находят при решении обратной задачи, т. е. для определения алгебраических выражений функций по картам, полученным в результате синтеза логических устройств.

1.4.4. Свойства карты Карно

1. Наборы значений переменных для соседних клеток карты Карно отличаются значением только одной переменной. При переходе из одной клетки в соседнюю изменяется значение только одной переменной от своего прямого значения к его инверсии или обратно.

Рассмотрим карту для четырех переменных (рис. 1.9). Значения функции 0 или 1 в клетках карты пока во внимание не принимаются. Для клеток, выделенных в карте, выписаны соответствующие им наборы.

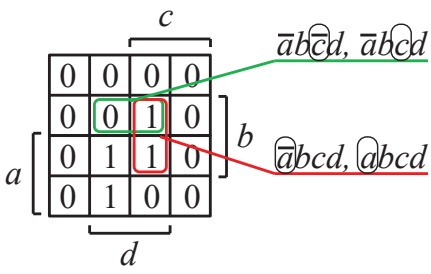


Рис. 1.9. Карта Карно для функции четырех переменных

Видно, что выражения для соседних клеток отличаются друг от друга значением только одной переменной.

2. Соседними между собой являются также крайние левые клетки с крайними правыми и крайние нижние клетки с крайними верхними (как если бы карты были свернуты в цилиндры по вертикали и горизонтали).

3. Рассмотрим карту для пяти переменных (рис. 1.10), не принимая пока во внимание значения функции в клетках карты. Выпишем наборы значений переменных для выделенных в карте клеток.

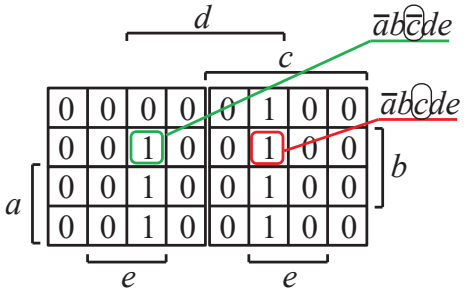


Рис. 1.10. Карта Карно
для функции пяти переменных

Наборы этих клеток отличаются значением только одной переменной. Можно сделать общее заключение: наборы клеток, симметричных относительно вертикальной двойной линии отличаются значением одной переменной.

Таким образом, все клетки, отличающиеся значением только одной переменной, являются соседними, несмотря на то, что иногда они расположены не рядом. Это свойство карты является очень важным для определения минимальных алгебраических выражений функции.

1.4.5. Определение СДНФ и СКНФ логических функций по картам Карно

СДНФ логической функции, изображенной в виде карты Карно, определяется следующим образом:

– для каждой клетки, в которой функция имеет значение 1, записывается конъюнкция всех входных переменных (прямых или инверсных), соответствующих этой клетке;

– составляется дизъюнкция этих конъюнкций, которая и представляет собой СДНФ данной функции.

СКНФ логической функции определяется по карте Карно в следующем порядке:

– для каждой клетки, в которой функция имеет значение 0, записывается дизъюнкция инверсий входных переменных, определяющих данную клетку;

– составляется конъюнкция этих дизъюнкций, представляющая собой СКНФ данной функции.

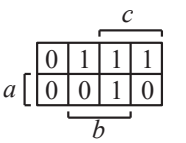


Рис. 1.11. Карта Карно для функции двух переменных

Например, дана логическая функция, представленная в виде карты Карно (рис. 1.11). Необходимо определить ее совершенные формы.

В этом случае

СДНФ:

$$f = \bar{a}b\bar{c} + \bar{a}bc + \bar{a}\bar{b}c + abc.$$

СКНФ:

$$f = (a + b + c)(\bar{a} + b + c)(\bar{a} + \bar{b} + c)(\bar{a} + b + \bar{c}).$$

Следуя приведенным выше правилам, можно определить СДНФ и СКНФ любой логической функции, изображенной в виде карты Карно.

1.4.6. Определение по карте Карно алгебраических выражений логических функций

Для некоторой логической функции, представленной посредством карты Карно, можно записать несколько алгебраических выражений различной сложности в дизъюнктивной или конъюнктивной формах. Приведем правила, которыми следует при этом руководствоваться:

1. Все единицы (при записи функции в дизъюнктивной форме) или все нули (при записи функции в конъюнктивной форме) должны быть заключены в прямоугольные контуры. Единичные контуры могут объединять несколько единиц, но не должны содержать внутри себя нули. Нулевые контуры могут объединять несколько нулей, но не должны содержать внутри себя единиц. Одноименные контуры могут накладываться друг на друга, т. е. одна и та же единица (или нуль) может входить в несколько единичных (нулевых) контуров.

2. Площадь любого контура должна быть симметричной относительно границ переменных, пересекаемых данным контуром.

Другими словами, число клеток в контуре должно быть равно 2^n, где $n = 0, 1, 2, ...$, т. е. число клеток выражается числами 1, 2, 4, 8, 16, ...

3. Во избежание получения лишних контуров, все клетки которых уже вошли в другие контуры, построение следует начинать с тех единиц или нулей, которые могут войти в один единственный контур.

4. В контуры можно объединять только соседние клетки, содержащие единицы или нули. Соседними считаются клетки, отличающиеся значением только одной входной переменной. Соседние клетки могут быть расположены не рядом, поэтому контуры могут иметь видимый разрыв.

5. Каждой единичной клетке соответствует конъюнкция входных переменных, определяющих данную клетку. Каждой нулевой клетке соответствует дизъюнкция инверсий входных переменных, определяющих данную клетку.

6. В контуре, объединяющем две клетки, одна из переменных меняет свое значение, поэтому выражение для контура из двух клеток не зависит от этой переменной, а представляется всеми остальными переменными. Это правило относится и к контурам, охватывающим число клеток более двух, и имеет такую формулировку: выражения, соответствующие контурам, не содержат тех переменных, чьи границы пересекаются площадью, ограниченной данным контуром. Возможна и другая формулировка этого правила: выражение, соответствующее контуру, не содержит те переменные, которые меняют свое значение в клетках контура.

7. Выражение логической функции может быть записано по соответствующей ей карте Карно в дизъюнктивной и конъюнктивной формах. Дизъюнктивная форма составляется в виде дизъюнкции конъюнкций, соответствующих единичным контурам, выделенным на карте для определения функции; конъюнктивная — в виде конъюнкции дизъюнкций, соответствующих нулевым контурам.

8. Для контуров, охватывающих различное количество клеток, получаются выражения различной сложности. Поэтому для данной логической функции можно записать по ее карте Карно несколько отличающихся по сложности алгебраических выражений. Наиболее сложное выражение соответствует случаю, когда каждый контур состоит только из одной клетки. Это выражение представляет собой СДНФ или СКНФ данной функции. С увеличением размеров контуров алгебраическое выражение упрощается. Самое простое выражение функции получается при образовании наибольших контуров. На этом свойстве основывается метод минимизации логических функций с помощью карт Карно.

В качестве примера найдем алгебраические выражения логической функции по соответствующей ей карте Карно, приведенной на рис. 1.12.

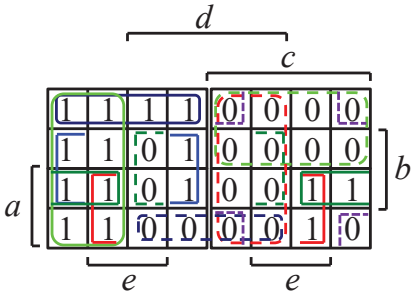


Рис. 1.12. Карта Карно
для функции пяти переменных

На данной карте каждая из клеток имеет непосредственные границы четырех переменных a, b, c, d, а граница пятой переменной e показана двойной линией. Сначала найдем выражение функции по ее единичным значениям. Руководствуясь правилом 3, находим в карте те единицы, которые можно включить только в один контур. Таких единиц четыре, они записываются согласно правилу 5 в виде конъюнкций:

$$\bar{a}\bar{b}\bar{c}de,\ \ ab\bar{c}d\bar{e},\ \ \ abcd\bar{e},\ \ a\bar{b}cde.$$

Учитывая правила 1, 2, 4, 8, заключаем найденные единицы в обязательные контуры возможно большего размера, которым согласно правилу 6, соответствуют следующие выражения:

$$\bar{a}\bar{b}\bar{c},\ \ b\bar{c}\,\bar{e},\ \ ab\bar{d},\ \ a\bar{d}e.$$

После этого остаются лишь две единицы $\bar{a}b\bar{c}\bar{d}e$, $a\bar{b}\bar{c}\bar{d}\bar{e}$, которые можно заключить в один общий контур. Этому контуру соответствует выражение $\bar{c}\bar{d}$. Применяя правило 7, можно записать для заданной логической функции алгебраическое выражение в дизъюнктивной форме:

$$f = \bar{a}\bar{b}\bar{c} + b\bar{c}\,\bar{e} + ab\bar{d} + a\bar{d}e + \bar{c}\bar{d}.$$

Переходя к определению выражения функции по нулевым значениям, получаем три обязательных контура:

$$\left(\bar{a}+b+\bar{d}\right),\ \ \left(\bar{b}+\bar{d}+\bar{e}\right),\ \ \left(b+\bar{c}+e\right).$$

Остальные нули могут быть заключены в два наибольших контура $(a+\bar{c})$ и $(\bar{c}+\bar{d})$. Алгебраическое выражение функции в конъюнктивной форме, записанное по нулевым контурам, имеет вид:

$$f=(a+\bar{c})(\bar{c}+\bar{d})(\bar{a}+b+\bar{d})(\bar{b}+\bar{d}+\bar{e})(b+\bar{c}+e).$$

Следует заметить, что на карте можно образовать еще другие единичные или нулевые контуры, однако они являются лишними, т. к. заключенные в них клетки уже вошли в обязательные контуры, без которых обойтись нельзя.

1.5. Минимизация логических функций

1.5.1. Общие понятия и сведения о минимизации

Одна из задач, которую приходится решать при построении логических устройств по заданной (или полученной в процессе синтеза) логической функции, состоит в следующем. Известны (либо заданы):

– реализуемая логическая функция (в виде аналитического выражения, СДНФ, таблицы истинности, карты Карно или другим способом);

– система элементарных логических функций и соответствующий ей стандартный набор типовых логических элементов;

– сравнительная сложность реализации каждой из элементарных функций системы, которая характеризуется условным коэффициентом (так называемой «ценой»), связанным с каждой из стандартных логических элементов;

– критерий минимизации (экономичность, быстродействие, минимум элементов какого-либо типа, количество операций, входящих в формулу, и др.).

Требуется найти алгебраическое выражение заданной функции и соответствующую ему схему, удовлетворяющую заданному критерию минимизации. Эту проблему называют проблемой минимизации [7]. Обычно ее решение связано с проблемой реализации различных схем и устройств с помощью минимального количества выбранных типовых элементов. Наиболее разработаны и употребительны методы минимизации совершенных нормальных форм функций. Поэтому для минимизации функции с помощью этих методов нужно предварительно переходить к СДНФ (СКНФ). Целесообразность представления функции в той

или иной системе зависит от того, какие элементарные логические функции реализуют отдельные логические элементы, входящие в выбранный набор. Набор функциональных типов логических элементов выбирается исходя из технических условий проектирования схемы логического устройства. В этот набор могут входить электромеханические реле, бесконтактные логические элементы, программируемые контроллеры.

Обычно при минимизации формул в системе И, ИЛИ, НЕ принимается, что, во-первых, функции конъюнкции и дизъюнкции одинаково сложны при реализации и, во-вторых, инверсия над переменными не влияет на усложнение реализации и может не учитываться. Однако зачастую на практике следует принимать во внимание отклонения от этих допущений, особенно для бесконтактных логических элементов [7].

Методы минимизации логических функций, известные в настоящее время, разработаны применительно к задаче получения формул минимальной длины. Критерием минимизации здесь является количество букв, входящих в алгебраическое выражение заданной функции. Формуле минимальной длины всегда соответствует релейно-контактная схема с минимальным числом контактов. Для схем из бесконтактных логических элементов это соответствие может не выполняться, т.е. минимальная длина формулы не является достаточным условием получения схемы минимальной сложности. Например, преобразование формулы логической функции посредством выноса переменных за скобки с уменьшением длины формулы всегда позволяет построить более простую релейно-контакторную схему, для бесконтактной же схемы такое преобразование часто приводит к ее усложнению.

Применение методов минимизации, разработанных для релейно-контакторных схем, при синтезе схем из бесконтактных логических элементов не позволяет учесть ряд специфических особенностей последних (например, реализацию любой элементарной логической функции одним элементом; наличие нескольких входов и одного выхода элемента; способность передавать сигналы только в одном направлении; необходимость фазировки сигналов и др.). Поэтому применительно к бесконтактным логическим схемам можно говорить не о минимальных, а о минимизированных в определенной степени схемах.

1.5.2. Метод непосредственного упрощения

Вначале отметим, что конституенты, имеющие одинаковое число букв и отличающиеся друг от друга значением лишь одной входящей в них переменной, называются соседними [3]. Непосредственное

упрощение исходной логической функции, заданной в СДНФ, выполняется в следующем порядке.

Для каждой из возможных пар соседних конституентов СДНФ применяется операция полного склеивания. При этом из них исключается по одной переменной. Затем выполняется приведение подобных членов. Этот процесс повторяется до тех пор, пока в полученном выражении не будет больше конъюнкций, отличающихся друг от друга значением одной переменной. Полученная таким образом форма называется сокращенной нормальной формой. Конъюнкции, входящие в сокращенную форму, называются простыми импликантами. Каждой логической функции соответствует лишь одна сокращенная форма.

Применяя к сокращенной нормальной форме операцию обобщенного склеивания, исключают из нее лишние конъюнкции (импликанты). Полученная в результате последовательного ряда таких преобразований форма, не допускающая дальнейших склеиваний, называется тупиковой дизъюнктивной формой (ТДФ) логической функции. Непосредственное упрощение исходной логической функции, заданной в СКНФ, дает тупиковую конъюнктивную форму логической функции (ТКФ). Сравнивая ТДФ и ТКФ, выбираем минимальную форму (МФ), которая затем реализуется.

Критерием выбора минимальной формы функции для релейно-контакторной схемы является минимальное число букв, для бесконтактных схем с использованием логических элементов — минимальное суммарное число элементов И и ИЛИ [7].

Таким образом, схема минимизации имеет следующий вид:

$$\begin{array}{l} \text{СДНФ} \xrightarrow{\text{ЗПС}} \text{СкДНФ} \xrightarrow{\text{ЗОС}} \text{ТДФ} \searrow \\ \text{СКНФ} \xrightarrow{\text{ЗПС}} \text{СкКНФ} \xrightarrow{\text{ЗОС}} \text{ТКФ} \nearrow \end{array} \quad \text{МФ}$$

где ЗПС — закон полного склеивания, СкДНФ (КНФ) — сокращенная дизъюнктивная (конъюнктивная) форма, ЗОС — закон обобщенного склеивания.

Пример 1. Найти минимальную форму функции, заданной СДНФ вида

$$f = a\bar{b}c + \bar{a}\bar{b}c + \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + ab\bar{c} + abc.$$

Применяя операцию полного склеивания к сочетаниям каждого из конституентов со всеми соседними и приводя подобные члены, получаем сокращенную нормальную форму:

$$f = \bar{a}\bar{b} + \bar{b}c + ac + ab + b\bar{c} + \bar{a}\,\bar{c}.$$

Применение операции обобщенного склеивания к импликантам можно осуществить в нескольких вариантах. Каждому из них соответствует одна из следующих тупиковых форм:

$$f_1 = ac + \bar{b}c + b\bar{c} + \bar{a}\,\bar{c};$$
$$f_2 = ac + b\bar{c} + \bar{a}\,\bar{c};$$
$$f_3 = \bar{b}c + ab + \bar{a}\,\bar{c}.$$

Очевидно, что рассматриваемой функции соответствуют две минимальные нормальные формы f_2 и f_3.

Пример 2. Минимизировать функцию $f(a,b,c)$, заданную следующей СДНФ:

$$f = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + a\bar{b}c + abc.$$

Имеем три пары соседних конъюнкций:

$$\bar{a}\bar{b}\bar{c},\ \bar{a}\bar{b}c;\ a\bar{b}c,\ abc;\ \bar{a}\bar{b}c,\ a\bar{b}c.$$

Применяя к ним операции полного склеивания, получаем простые импликанты: $\bar{a}\bar{b}$, ac , $\bar{b}c$. Их дизъюнкция даст сокращенную нормальную форму:

$$f = \bar{a}\bar{b} + ac + \bar{b}c.$$

Применяя к ней операцию обобщенного склеивания, получаем тупиковую форму, являющуюся в данном случае минимальной:

$$f = \bar{a}\bar{b} + ac.$$

Для реализации закона обобщенного склеивания на практике используется импликантная матрица Квайна [3]. Построение ее рассмотрим на примере ниже. Необходимо минимизировать функцию, заданную в виде следующей СДНФ:

$$f = \bar{a}\bar{b}\bar{c} + a\bar{b}\bar{c} + ab\bar{c} + abc.$$

Находим пары соседних конституентов, т. е. исходное выражение можно представить в виде

$$f = \bar{a}\bar{b}\bar{c} + a\bar{b}\bar{c} + a\bar{b}\bar{c} + ab\bar{c} + ab\bar{c} + abc.$$

Применив закон полного склеивания, получим:

$$f = \bar{b}\bar{c} + a\bar{c} + ab.$$

Полученное выражение есть сокращенная форма ДНФ.

Теперь строим импликантную матрицу Квайна, располагая по вертикали простые импликанты из сокращенной ДНФ, а по горизонтали — конституенты единицы заданной функции (табл. 1.8). Для каждой импликанты отыскиваются конституенты, частью которых она является, что отмечается знаком «+» в соответствующей клетке матрицы. Из матрицы непосредственно определяются импликанты для образования тупиковых ДНФ функции. Среди тупиковых ДНФ определяются минимальные ДНФ. Импликанты, соответствующие столбцам с одним знаком «+», обязательно войдут в тупиковую или минимальную ДНФ. Поэтому в данном примере сразу можно записать выражение $f = \bar{b}\bar{c} + ab$, являющееся тупиковой и минимальной ДНФ.

Таблица 1.8

Матрица Квайна

Простые импликанты	**Конституенты единицы**			
	$\bar{a}\bar{b}\bar{c}$	$a\bar{b}\bar{c}$	$ab\bar{c}$	abc
$\bar{b}\bar{c}$	+	+		
$a\bar{c}$		+	+	
ab			+	+

Рассмотренный метод Квайна позволяет сравнительно легко определить минимальную форму логической функции небольшого числа переменных. Однако применение метода Квайна для более сложных функций обнаруживает одно существенное неудобство. Оно связано с необходимостью исчерпывающего попарного сравнения всех членов исходной СДНФ функции. Число таких сравнений растет с увеличением числа членов СДНФ, что влечет возможность ошибки.

Рассмотрим процедуру минимизации методом непосредственного упрощения на примере синтеза компаратора, реализующего функцию $x = A \geq B$. На практике это сравнение десятичных чисел, которые в вычислительном устройстве преобразованы в двоичный код. Любое число от 0 до 9 можно представить четырехразрядным двоичным числом. Например, число 7 будет представлено комбинацией 0111, где веса разрядов, начиная со старшего — 2^3, 2^2, 2^1, 2^0. Сравнение чисел A и B

будет производиться поразрядно, поэтому общее число переменных $n = 8$, число наборов — 256. Упростим задачу, чтобы можно было пройти весь путь решения в ручном режиме: сравним двухразрядные числа a_1, a_2, b_1, b_2. В табл. 1.9 представлены все наборы и значения функции для каждого набора.

Таблица 1.9

Функция $x = A \geq B$

2^1	2^0	2^1	2^0	$x = A \geq B$
a_1	a_2	b_1	b_2	
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Из таблицы получим СДНФ:

$$x = \overline{a_1}\,\overline{a_2}\,\overline{b_1}\,\overline{b_2} + \overline{a_1}a_2\overline{b_1}\,\overline{b_2} + \overline{a_1}a_2\overline{b_1}b_2 + a_1\overline{a_2}\,\overline{b_1}\,\overline{b_2} +$$
$$+ a_1\overline{a_2}\,\overline{b_1}b_2 + a_1\overline{a_2}b_1\overline{b_2} + a_1a_2\overline{b_1}\,\overline{b_2} + a_1a_2\overline{b_1}b_2 + a_1a_2b_1\overline{b_2} + a_1a_2b_1b_2.$$

СКНФ:

$$x = \left(a_1 + a_2 + b_1 + \overline{b_2}\right)\left(a_1 + a_2 + \overline{b_1} + b_2\right)\left(a_1 + a_2 + \overline{b_1} + \overline{b_2}\right)\times$$
$$\times\left(a_1 + \overline{a_2} + \overline{b_1} + b_2\right)\left(a_1 + \overline{a_2} + \overline{b_1} + \overline{b_2}\right)\left(\overline{a_1} + a_2 + \overline{b_1} + \overline{b_2}\right).$$

Теперь необходимо найти все возможные пары соседних конституентов, реализовать закон полного склеивания (поглощения), затем продолжать эту процедуру с импликантами, пока это возможно. После этого — закон обобщенного склеивания (матрица Квайна). Как видно, работа достаточно большая, требует внимания и чревата ошибками. Читателю предлагается пройти этот путь самостоятельно и оценить затраты времени. Поэтому практический метод минимизации основан на использовании карт Карно.

1.5.3. Метод минимизации с помощью карт Карно

Для получения по карте Карно минимального выражения логической функции следует руководствоваться единственным правилом (кроме общих правил, рассмотренных ранее): единицы или нули должны объединяться минимальным числом наибольших контуров [3].

Иногда для некоторых наборов значение функции строго не определено, и ее карта Карно содержит условные члены, т. е. такие клетки, в которых значение функции можно считать равным единице или нулю. Такие члены на карте принято обозначать знаком ~, символизирующим возможность любого из значений — 0 или 1. Знак тильда в клетках карты может и не проставляться. Пустая клетка соответствует условному значению функции.

Наличие условных значений функции в клетках карты Карно позволяет включать эти клетки в контуры с единицами или нулями (по усмотрению проектировщика), что способствует получению минимальных алгебраических выражений для данной логической функции.

Пример 1. Минимизировать функцию, представленную в виде карты Карно на рис. 1.13.

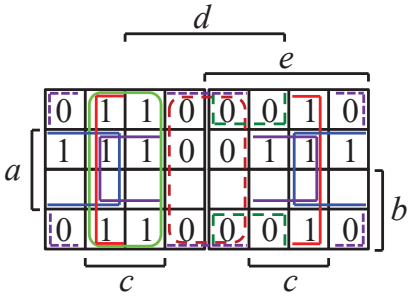


Рис. 1.13. Карта Карно с пустыми клетками

Следуя правилам определения алгебраических выражений по карте Карно, руководствуясь правилом минимизации и учитывая наличие условных значений функции, можно записать следующие выражения для данной логической функции:

по единичным контурам:

$$x = ac + a\bar{d} + c\bar{d} + c\bar{e};$$

по нулевым контурам:

$$x = (a + c)(c + \bar{d})(a + \bar{d} + \bar{e}).$$

Нетрудно убедиться, что полученные выражения равносильны, перемножив выражения в скобках и проведя необходимые упрощения.

В пункте 1.5.2 была поставлена задача синтеза компаратора на основании таблицы истинности (табл. 1.9). По таблице истинности заполняем карту Карно.

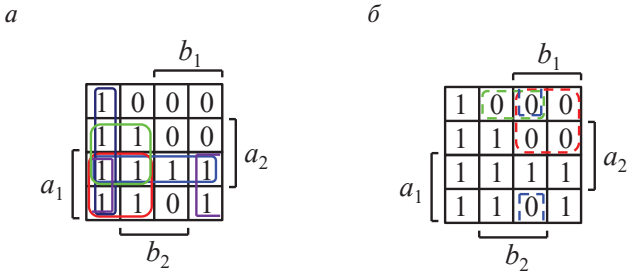


Рис. 1.14. Карта Карно с единичными (*а*) и нулевыми (*б*) контурами

По единичным контурам можно составить следующее выражение:

$$x = \bar{b}_1\bar{b}_2 + a_1a_2 + a_1\bar{b}_2 + a_2\bar{b}_1 + a_1\bar{b}_1.$$

По нулевым —

$$x = (a_1 + \bar{b}_1)(a_1 + a_2 + \bar{b}_2)(a_2 + \bar{b}_1 + \bar{b}_2).$$

При реализации первого варианта потребуется пять элементов И и один элемент ИЛИ, для второго варианта — три элемента ИЛИ и один элемент И, т. е. второй вариант при реализации будет проще.

Рассмотрим более сложную задачу — синтез преобразователя прямого двоичного кода (ПДК) в дополнительный код (ДК). Применяемые в системах автоматизированного управления коды будут рассмотрены

ниже. Большинство кодов имеют четыре разряда. Дополнительный код образуется, если прибавить единицу к младшему разряду обратного кода, который получается инвертированием разрядов прямого кода.

Прямой код					**Обратный код**					**Дополнительный код**			
0	0	0	0	→	1	1	1	1	→	0	0	0	0
0	0	0	1	→	1	1	1	0	→	1	1	1	1
0	0	1	0	→	1	1	0	1	→	1	1	1	0

Рис. 1.15. Преобразователь кода

Согласно таблице истинности (табл. 1.10), составляем карты Карно для каждой выходной переменной.

Таблица 1.10

Таблица истинности для преобразователя кодов

Число	**Прямой код**				**Дополнительный код**			
	2^3	2^2	2^1	2^0				
	a_1	a_2	a_3	a_4	x_1	x_2	x_3	x_4
0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1
2	0	0	1	0	1	1	1	0
3	0	0	1	1	1	1	0	1
4	0	1	0	0	1	1	0	0
5	0	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	0
7	0	1	1	1	1	0	0	1
8	1	0	0	0	1	0	0	0
9	1	0	0	1	0	1	1	1

По изображенным на рис. 1.16 картам Карно можно получить алгебраические выражения для выходных переменных:

$$x_1 = a_2 + a_3 + \overline{a_1} a_4 + a_1 \overline{a_4};$$

$$x_2 = \overline{a}_2 a_3 + \overline{a}_2 a_4 + a_2 \overline{a}_3 \overline{a}_4;$$

$$x_3 = \overline{a}_3 a_4 + a_3 \overline{a}_4;$$

$$x_1 = a_4.$$

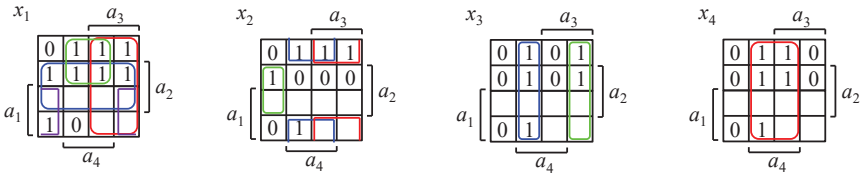


Рис. 1.16. Карты Карно, построенные по табл. 1.10

Реализуем преобразователь ПДК-ДК с использованием элементов избыточного базиса И, ИЛИ, НЕ в виде схемы, показанной на рис. 1.17.

В вычислительной технике для реализации подобного рода задач наряду с элементами, реализующими функции И, ИЛИ, НЕ, выпускаются элементы, реализующие функцию Неэквивалентность (исключающая ИЛИ, XOR), $f = a\overline{b} + \overline{a}b$, условное обозначение $a \otimes b$. Ее использование позволяет существенно упростить схему.

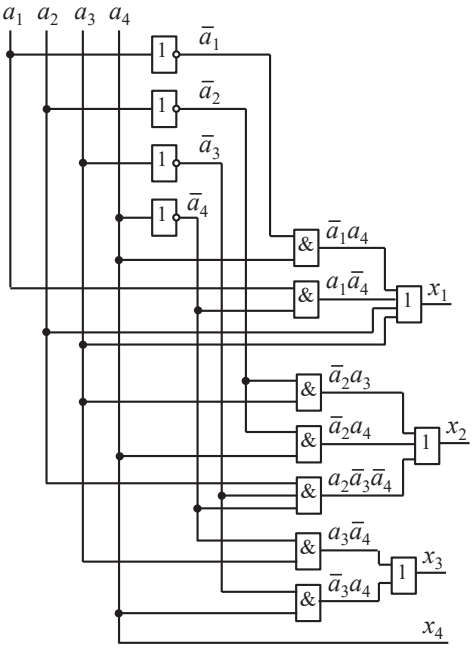


Рис. 1.17. Схема преобразователя ПДК-ДК
на логических элементах И, ИЛИ, НЕ

Преобразуем полученные для $x_1 \ldots x_4$ формулы:

$$x_1 = a_2 + a_3 + a_1 \otimes a_4;$$

$$x_2 = \overline{a_2}a_3 + \overline{a_2}a_4 + a_2\overline{a_3}\,\overline{a_4} = \overline{a_2}\left(a_3 + a_4\right) + a_2\left(\overline{a_3 + a_4}\right) = a_2 \otimes \left(a_3 + a_4\right);$$

$$x_3 = \overline{a_3}a_4 + a_3\overline{a_4} = a_3 \otimes a_4;$$

$$x_1 = a_4 = a_4\overline{0} + \overline{a_4}0 = a_4 \otimes 0.$$

В результате мы видим, насколько упростилась схема (рис. 1.18).

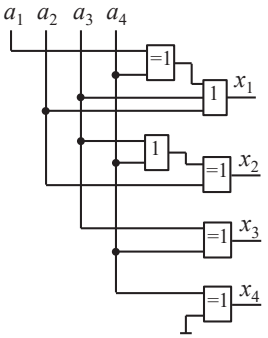


Рис. 1.18. Схема преобразователя ПДК-ДК на основе элементов OR и XOR

Еще один пример — простейшая школьная задача о резервуаре и двух трубах, но рассматривается она с точки зрения автоматизации. Насос наполняет бак до допустимого уровня и отключается. По мере расходования воды ее уровень падает до минимально допустимого, при котором насос автоматически снова включается и наполняет бак до максимальной отметки, после чего — отключается и так далее.

Уровень воды (верхний и нижний) контролируется двумя датчиками — d_1 и d_2 (рис. 1.19). Таблица истинности (табл. 1.11) составлена на основе анализов процесса заполнения и опорожнения бака. При пустом баке $d_1 = d_2 = 0$ и насос должен работать (H = 1). При подъеме уровня воды до первого датчика ($d_1 = 1$) насос продолжает работать. При $d_1 = d_2 = 1$ насос отключается, после чего уровень падает, $d_1 = 1$, $d_2 = 0$, насос не работает. Включение насоса произойдет при $d_1 = 0$.

Таблица 1.11

Алгоритм работы насоса

№ набора	d_1	d_2	H
1	0	0	1
2	1	0	1
3	1	1	0
4	1	0	0
5	0	1	

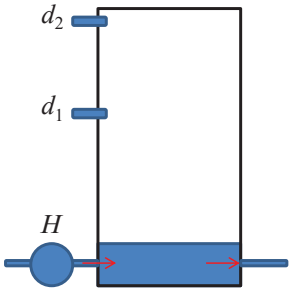


Рис. 1.19. Схема резервуара с водой

Анализ заполненной таблицы истинности показывает, что совпадающим наборам № 2 и № 4 соответствуют разные значения функции. Набор № 2 появляется после набора 0, 0, а набор № 4 — после набора 1, 1, т. е. выходной сигнал зависит не только от комбинации входных переменных, но и от их предшествующей комбинации, а это возможно при использовании элементов «Память». Отметим, что комбинация 0, 1 (пятый набор) при нормальной работе схемы появиться не может. Если такое происходит, то это говорит о нарушении работы датчика d_1 или d_2. При появлении этого набора необходимо отключить насос и включить аварийную сигнализацию.

1.5.4. Логические функции и реализующие их схемы

Логические функции подразделяются на два класса: комбинационные и последовательностные [3]. Комбинационными называются логические функции, значения которых определяются только комбинациями значений входных переменных вне зависимости от последовательности появления этих комбинаций. Последовательностными называются логические функции, значения которых зависят не только от комбинации входных переменных, но и от последовательности появления этих комбинаций, т. е. такие функции содержат элемент «Память». Комбинационные и последовательностные функции реализуются соответственно однотактными и многотактными схемами релейного действия. Многотактные схемы, или схемы с обратными связями, являются основными схемами промышленной автоматики. Синтез последовательностных функций может быть реализован по словесному описанию, на основе циклограмм, с использованием графов и ряда других методов.

Условия работы однотактных схем могут быть заданы таблицами истинности, СДНФ (СКНФ) функции, картами Карно. Условия работы многотактных схем, записанные в виде такой таблицы, будут содержать противоречивые строки и называться нереализуемыми. Нереализуемые условия работы схемы можно перевести в реализуемые, вводя дополнительные переменные с таким расчетом, чтобы различным комбинациям значений выходных переменных соответствовали различные комбинации значений входных и дополнительных входных переменных. Дополнительные входные переменные носят название промежуточных переменных. Они могут быть получены за счет образования обратных связей, посредством которых сигналы с выходов схемы или от промежуточных элементов подаются на входы соответствующих элементов схемы. Если использование имеющихся в схеме промежуточных и выходных сигналов не позволяет сделать ее реализуемой, следует вводить дополнительные промежуточные элементы, с выходов которых в схему будут вводиться необходимые промежуточные переменные. В качестве промежуточных переменных в первую очередь используются выходные переменные.

Контрольные вопросы к главе 1

1. Что есть логическая функция?
2. Какая логическая функция считается полностью заданной?
3. В чем заключается сущность комбинационной логической функции?
4. Каким образом может быть описана логическая функция?
5. Какие аксиомы алгебры логики вам известны?
6. Перечислите основные законы алгебры логики.
7. Какая форма логической функции считается дизъюнктивной?
8. Дайте объяснение термину «конъюнктивная форма функции».
9. Для чего используются карты Карно?
10. Перечислите способы минимизации логических функций.

Глава 2.

Синтез управляющих логических устройств на основе циклограмм

При проектировании устройств управления механизмами, работающими циклически, метод записи условий их работы посредством циклограмм имеет преимущество наглядности и поэтому широко применяется.

2.1. Понятия и определения

Циклограмма — это графическое изображение последовательности работы отдельных элементов управляющего логического устройства во времени (рис. 2.1). Работа элементов дискретного действия в логическом устройстве характеризуется появлением и исчезновением сигналов в определенной последовательности [3].

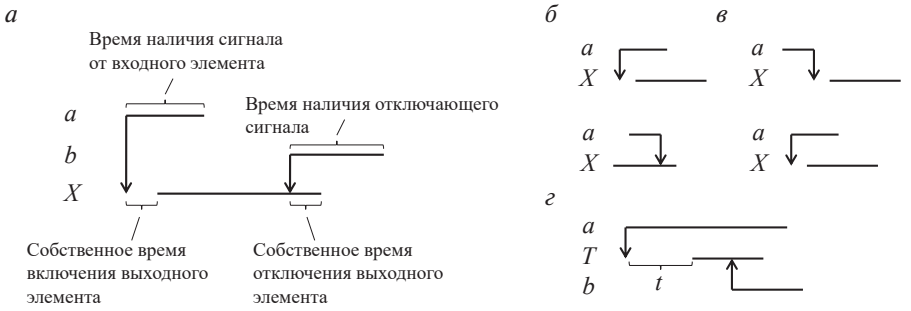


Рис. 2.1. Примеры циклограмм

Наличие сигнала изображается на циклограмме отрезком горизонтальной прямой. Толстой линией обозначаются сигналы входных и выходных элементов, тонкой — дополнительных промежуточных элементов, пунктиром — условное включение элемента. Условное включение элемента соответствует условному состоянию, т. е. включенное или выключенное состояние этого элемента в данный период не оказывает влияния на состояние выходных элементов устройства.

Слева от отрезка, отражающего работу элемента, на границе циклограммы проставляется обозначение соответствующего сигнала. Последовательность работы элементов определяется положением концов отрезков, изображающих их работу, относительно левой границы циклограммы (рис. 2.1 *а*). На циклограмме отражается любое изменение состояния элементов и указывается собственное время их срабатывания. Воздействие одного элемента на другой изображается на циклограмме стрелкой, указывающей направление воздействия. Варианты воздействия элемента *а* на элемент *X* показаны на рис. 2.1 *б* (при включении) и на рис. 2.1 *в* (при отключении).

В циклограмме время не оценивается количественно, поэтому она выполняется без масштаба. Отмечается лишь факт наличия или отсутствия сигнала. При наличии специального элемента задержки его сигнал на циклограмме обозначается буквой *T*, а время, по истечении которого он появляется или исчезает, — буквой *t* (рис. 2.1 *г*).

Тактами называются периоды, в течение которых в схеме не изменяется состояние ни одного из входных, промежуточных или выходных сигналов. Каждое изменение состояния одного или одновременно нескольких элементов является началом нового такта.

Периодом включения элемента называется непрерывный ряд тактов, в течение которого этот элемент находится во включенном состоянии. Периодом отключения элемента называется непрерывный ряд тактов, в течение которого этот элемент находится в отключенном состоянии. Период включения элемента обозначается чертой. В периоде отключения элемента черта на циклограмме отсутствует (рис. 2.2).

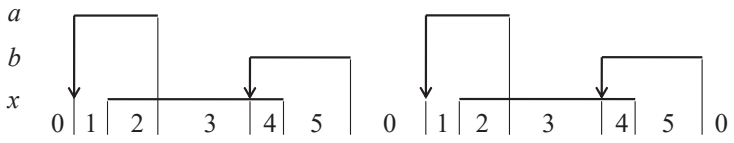


Рис. 2.2. Изображение на циклограмме характерных тактов и периодов:
1 — включающий такт; 4 — отключающий такт; такты 2, 3, 4 — период включения; такты 1, 2, 3 — включающий период; такты 4, 5, 0 — отключающий период; такты 5, 0, 1 — период отключения

Включающим тактом называется такт, предшествующий периоду включения данного элемента. Отключающим тактом называется такт, предшествующий периоду отключения данного элемента.

Включающий период состоит из включающего такта и периода включения без отключающего такта. Отключающий период состоит из отключающего такта и периода отключения без включающего такта (понятие отключающего периода вводится при наличии нескольких периодов включения). Приведенные определения иллюстрируются циклограммой на рис. 2.2, по которой включающий период для переменной x — такты 1, 2, 3 (от включающей стрелки до отключающей), отключающий период — такты 4, 5, 0 (от отключающей стрелки до следующей включающей).

2.2. Составление алгебраических выражений по циклограммам

Алгебраические выражения или структурные формулы, описывающие управляющее логическое устройство, составляются для каждого выходного и промежуточного элементов, затем составляется общее выражение для всего логического устройства. Общая структурная формула анализируется с точки зрения возможности минимизации с целью упрощения схемы устройства. Для минимизации алгебраических выражений используются алгебраические равносильные преобразования.

Изменение состояния какого-либо выходного или промежуточного элемента, т. е. его включение и отключение, обусловлено изменением состояния других, воздействующих на него элементов. Условия, вызывающие изменение состояния элемента, возникают во включающем и отключающем тактах и называются условиями срабатывания во включающем такте и несрабатывания в отключающем такте (включающий сигнал и отключающий сигнал) [7].

Условие срабатывания обозначается f'. В условие срабатывания входит сигнал элемента, изменяющего свое состояние во включающем такте. Если в этом такте изменяют свое состояние несколько элементов, то в условие срабатывания достаточно ввести сигнал одного (любого из одновременно срабатывающих элементов) основного элемента, от которого на циклограмме отходит стрелка в направлении рассматриваемого выходного или промежуточного элемента. В условие

срабатывания записывается сигнал этого элемента в том состоянии, которое он имеет во включающем такте.

Условия несрабатывания элемента возникают в отключающем такте. В эти условия входит сигнал элемента, изменяющего свое состояние в этом такте. При наличии нескольких элементов, изменяющих свое состояние в отключающем такте, следует ввести в условие несрабатывания сигнал любого из основных элементов, от которого на циклограмме отходит стрелка в направлении рассматриваемого выходного или промежуточного элемента. В условие несрабатывания сигнал основного элемента записывается в том состоянии, в котором он находится в отключающем такте, со знаком инверсии. Условие несрабатывания обозначается $\overline{f''}$ и приписывается к предыдущему условию срабатывания со знаком логического умножения [3]. Таким образом, структурная формула для одного периода включения какого-либо исполнительного или промежуточного элемента имеет вид:

$$X = f'\overline{f''}.$$

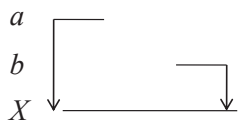


Рис. 2.3. Пример циклограммы

На рис. 2.3 показаны в качестве примера условия срабатывания элемента X под воздействием элемента a и условия его несрабатывания под воздействием элемента b.

Во включающем такте появляется сигнал a. Он вызывает срабатывание элемента X и появление одноименного сигнала X. Условие срабатывания для этого случая запишется так:

$$f' = a.$$

Условия несрабатывания возникает в отключающем такте и записывается следующим образом:

$$\overline{f''} = \overline{\overline{b}}.$$

Один знак инверсии проставлен, потому что предусмотрен формулой условий несрабатывания, а второй, — потому что в отключающем такте сигнал b исчезает ($b = 0$).

Структурная формула для периода включения элемента X:

$$X = f'\overline{f''} = ab.$$

Релейно-контакторная схема, соответствующая полученному выражению, изображена на рис. 2.4. Анализ этой схемы показывает, что она неработоспособна, т. к. входной элемент *a* не обеспечивает необходимых условий включения выходного элемента *X* и поэтому следует произвести проверку схемы и структурной формулы и ввести дополнительные элементы.

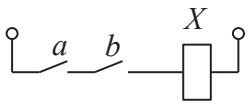


Рис. 2.4. Схема, реализующая условия циклограммы на рис 2.3

При наличии нескольких периодов включения и отключения какого-либо выходного или промежуточного элемента условия срабатывания и несрабатывания составляются для каждого периода. Для первого периода составляется выражение в виде $f_1' \overline{f_1''}$, для второго периода — $f_2' \overline{f_2''}$, и для *n*-го периода — $f_n' \overline{f_n''}$.

Общая структурная формула для входного или промежуточного элемента *X* записывается так:

$$X = f_1' \overline{f_1''} + f_2' \overline{f_2''} + \ldots + f_n' \overline{f_n''},$$

т. е. в виде логической суммы логических произведений условий срабатывания и условий несрабатывания для всех периодов включения.

Введение только основных элементов в условия срабатывания и несрабатывания иногда бывает недостаточным для обеспечения работы устройства по данной циклограмме. В этих случаях приходится вводить промежуточные элементы. Для определения необходимости введения промежуточных элементов и их числа необходимо производить три проверки реализуемости циклограммы. Все три проверки проводятся в отдельности для каждого включающего периода.

Первая проверка заключается в анализе того, существуют ли записанные ранее условия срабатывания (f') в течение всего включающего периода. Если функция f' неизменна в этом периоде, то эти условия срабатывания для данного периода включения являются достаточными. Если же функция f' вторично изменяет свое значение в течение включающего периода, то необходимо в схему ввести дополнительный элемент (P') таким образом, чтобы он изменял свое состояние до изменения состояния f' и оставался в этом состоянии далее в течение всего анализируемого включающего периода. Таким образом, если на циклограмме мысленно совместить линии действия сигналов f' и P', то это будет непрерывная прямая в течение всего включающего периода рассматриваемого элемента. Желательно в качестве про-

межуточного элемента при возможности использовать выходной элемент, применяя самоблокировку, или другие элементы циклограммы, линии действия сигналов которых удовлетворяют условиям выбора P'. Сигнал дополнительного элемента (P') или самоблокировки приписывается к условию несрабатывания $\overline{f''}$ в виде логического произведения $P'\overline{f''}$, а произведение в свою очередь приписывается к первоначальному условию срабатывания (f') в виде суммы, причем его состояние принимается обратным тому, которое он имеет во включающем такте. Эта проверка производится для каждого периода включения данного элемента.

Введение самоблокировки в результате первой проверки в первом периоде включения не требует дополнительных рассуждений. Применение самоблокировки в последующих периодах включения следует увязывать с предыдущими. Если в первом периоде включения была введена самоблокировка, то для использования ее в последующих периодах необходимо учитывать имеющиеся комбинации сигналов в сочетании с сигналом, используемым для самоблокировки. При наличии хотя бы одной из этих комбинаций в течение всех последующих периодов рассматриваемого элемента использование самоблокировки возможно в случае, если сигнал или сигналы, соответствующие условиям несрабатывания для данного включающего периода, не встречались в предыдущих включающих периодах или встречались, но также в качестве условий несрабатывания. Тогда в виде произведения они приписываются к используемой комбинации сигналов с самоблокировкой во всех структурных формулах включающих периодов рассматриваемого элемента. Ниже будет рассмотрен пример, иллюстрирующий наряду с другими и это положение.

Для n-го периода включения при введении дополнительного элемента P' структурная формула элемента X запишется так:

$$X_n = f_n' + p'\overline{f_n''},$$

или при самоблокировке

$$X_n = f_n' + x\overline{f_n''}.$$

Возможна и такая запись:

$$X_n = \left(f_n' + x\right)\overline{f_n''}.$$

Функциональная равносильность этих записей при определенной последовательности появления значений входных сигналов $\begin{pmatrix} 1 & 0 & 1 & 0 \\ a & a & b & b \end{pmatrix}$ очевидна из приведенных на рис. 2.5 релейно-контакторных схем.

При срабатывании элемента *a* получает питание элемент *X*, самоблокируется и отключается лишь при срабатывании элемента *b*. В схемах автоматики часто требуется, чтобы после срабатывания отключающего элемента схему нельзя было вновь запустить в работу до тех пор, пока он не придет в исходное состояние. Например, если нажата кнопка «Стоп», отключающая какой-либо аппарат, то одновременное нажатие кнопки «Пуск» не должно вызывать даже кратковременного его включения. С этой точки зрения целесообразней использовать структурную формулу $X = (a + x)\bar{b}$.

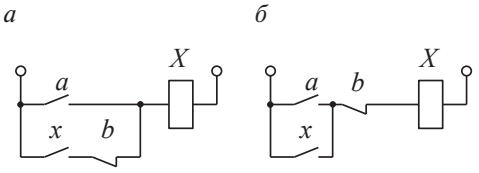


Рис. 2.5. Функционально равносильные варианты релейно-контакторных схем с самоблокировкой:
$а$ — $X = a + x\bar{b}$; $б$ — $X = (a + x)\bar{b}$

Вторая проверка предназначена для выяснения, существуют ли записанные ранее условия несрабатывания ($\overline{f''}$) в течение всего включающего периода. Если функция $\overline{f''}$ неизменна в этом периоде, то эти условия несрабатывания для данного периода включения являются достаточными. Если же функция $\overline{f''}$ вторично изменяет свое состояние в течение включающего периода, то необходимо ввести в схему промежуточный элемент (P'), изменяющий свое состояние в периоде включения рассматриваемого элемента, но после того, как изменит свое значение функция $\overline{f''}$. Если функция $\overline{f''}$ изменяет свое состояние несколько раз, то промежуточный элемент (P') должен изменять свое состояние после последнего изменения состояния функции $\overline{f''}$.

В качестве промежуточного элемента желательно использовать при возможности другие выходные или уже введенные в циклограмму промежуточные элементы, изменяющие свое состояние так, как необходимо для обеспечения условий несрабатывания рассматриваемого

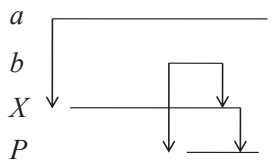


Рис. 2.6. Пример циклограммы, в которой срабатывает вторая проверка

элемента. Условия несрабатывания в этом случае записываются в виде инверсии произведения двух сигналов f'' и p'', т. е. $\overline{f''p''}$. Значения сигналов f'' и p'' берутся такими, которые они имеют в отключающем такте.

Для приведенной на рис. 2.6 циклограммы первая проверка не требует самоблокировки, а по результатам второй проверки вводится промежуточная переменная P:

$$X = a\overset{= 2\text{-я проверка}}{b} \rightarrow a\overline{\overline{b}p} = a\left(b + \overline{p}\right).$$

Включающим сигналом для P служит b, отключить P можно сигналом X в момент его отключения. На практике отключить P можно позднее, если появляется возможность для использования P в качестве промежуточной переменной для других выходных переменных:

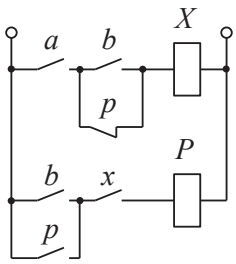


Рис. 2.7. Схема, реализующая алгоритм со второй проверкой

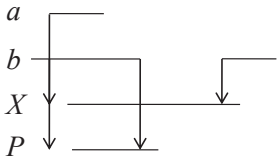
$$P = b\overset{= 1\text{-я проверка}}{x} \rightarrow \left(b + p\right)x.$$

Таким образом, итоговый алгоритм имеет следующий вид:

$$X = a\left(b + \overline{p}\right);\ \ P = \left(b + p\right)x,$$

а схема, реализующая его, представлена на рис. 2.7.

На рис. 2.8 приведена циклограмма, для которой первая и вторая проверки дают положительный результат, т. е. требуются самоблокировка и введение промежуточной переменной P. В этом случае P может включаться входным сигналом a.



$$X = a\overline{b}\overset{1\,пр.}{\rightarrow} a + x\overline{b}\overset{2\,пр.}{\rightarrow} a + x\left(\overline{b\overline{p}}\right);$$

$$X = a + x\left(\overline{b} + p\right) = a + x\overline{b} + xp;$$

$$P = a\overset{= 1\,пр.}{b} \rightarrow a + bp.$$

Рис. 2.8. Циклограмма, в которой срабатывают первая и вторая проверки

Формула схемы:

$$X = a + x\overline{b} + xp;\ \ P = a + bp.$$

Реализуем схему с использованием элементов И, ИЛИ, НЕ (рис. 2.9).

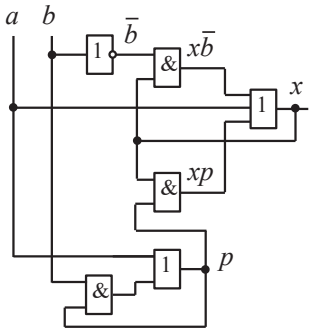


Рис. 2.9. Схема, реализующая циклограмму, показанную на рис. 2.8

Третья проверка предназначена для контроля того, чтобы после отключения выходного элемента не создавались вновь условия для его повторного (неправильного) включения. С этой целью следует проверить каждую комбинацию сигналов в выражениях $f'\overline{f''}$, $f'+p'\overline{f''}$, $f'\overline{f''p''}$ или $f'+p'\overline{f''p''}$ после того, как будут раскрыты все скобки и исключены общие знаки инверсии, т. е. получено выражение функции в дизъюнктивной нормальной форме, причем выражение вида $X=f'\overline{f''}$ получается, если первоначально принятые условия срабатывания и несрабатывания удовлетворяют всему включающему периоду данного элемента; выражение $X=f'+p'\overline{f''}$ получается, если в условия срабатывания вводится дополнительный элемент; выражение $X=f'\overline{p''f''}$ получается, если в условия несрабатывания вводится дополнительный элемент; выражение $X=f'+p'\overline{p''f''}$ получается, если дополнительные элементы вводятся в условия срабатывания и несрабатывания.

Каждое слагаемое в ДНФ функции X в данном периоде включения после организации двух проверок дает определенную комбинацию сигналов и их инверсий. Ни одна из этих комбинаций не должна встречаться в отключающих периодах элемента. Если в отключающем периоде появляется одна из этих комбинаций, то в эту комбинацию должен быть введен еще хотя бы один дополнительный промежуточный элемент P'''. Состояния этого элемента должны быть различны при данной комбинации во включающем и отключающем периодах.

В качестве промежуточного элемента также желательно использовать уже введенные в циклограмму промежуточные или выходные элементы, если они удовлетворяют условиям применения этого промежуточного элемента. Сигнал вводимого промежуточного элемента приписывается в виде произведения к тому слагаемому, которое дает комбинацию, встречающуюся во включающем и в отключающем периодах, состояние этого элемента принимается таким, какое он имеет при данной комбинации во включающем периоде.

Для наиболее сложной структурной формулы, которая получается если и первая, и вторая проверки дают положительный результат, по результатам третьей проверки каждой из включающих комбинаций может потребоваться введение p''', то есть

$$X = f' p_1''' + p' \overline{p}'' p_2''' + p' \overline{f}'' p_3''' .$$

Практическая реализация третьей проверки будет рассмотрена ниже на конкретных примерах.

После того как проведены по три проверки для каждого периода включения данного элемента, полученную структурную формулу:

$$X = X_1 + X_2 + \ldots + X_n = f_1' \overline{f_1''} + f_2' \overline{f_2''} + \ldots + f_n' \overline{f_n''},$$

можно упрощать, используя алгебраические равносильные и другие преобразования с целью получения рациональной структурной схемы. Аналогично получаются структурные формулы для остальных выходных элементов. Для промежуточных элементов структурные формулы составляются так же, как и для выходных.

Общая структурная формула всего логического устройства составляется в виде суммы произведений общей формулы каждого выходного или промежуточного элемента на сигнал этого элемента:

$$F = \left(X_1 + X_2 + \ldots + X_n\right) X + \left(Y_1 + Y_2 + \ldots + Y_m\right) Y + \left(P_1 + P_2 + \ldots + P_k\right) P.$$

Общая структурная формула при возможности преобразовывается с целью упрощения общей схемы логического устройства. Основным преобразованием этой формулы, как правило, является вынесение общих множителей за скобки, что при реализации алгебраических выражений дает возможность выделять в схеме узлы, общие для нескольких выходов, и вычерчивать их один раз, не повторяя для каждого выхода в отдельности.

Если в условия срабатывания входит несколько основных сигналов (например, элемент X включается при появлении сигналов a_1, или a_2, или a_3), то в структурную формулу условия срабатывания записываются в виде логической суммы $f' = a_1 + a_2 + a_2$.

Если в условия несрабатывания входит несколько основных сигналов (например, исполнительный элемент X отключается, если будут сигналы или b_1, или b_2, или b_3), то в структурной формуле условия несрабатывания записываются в виде инверсии логической суммы $\overline{f''} = \overline{b_1 + b_2 + b_2} = \overline{b_1} \cdot \overline{b_2} \cdot \overline{b_3}$. Функции f' и f'' могут иметь и более сложные

выражения, которые составляются на основании словесных формулировок условий срабатывания и несрабатывания.

В представленной на рис. 2.10 циклограмме выходная переменная X в течение времени цикла $t_{ц}$ имеет два периода включения: $X = X_1 + X_2$.

$$X_1 = a\,b \overset{=1\,пр.}{\rightarrow} a + xb;$$

$$X_2 = \bar{c}\bar{d}.$$

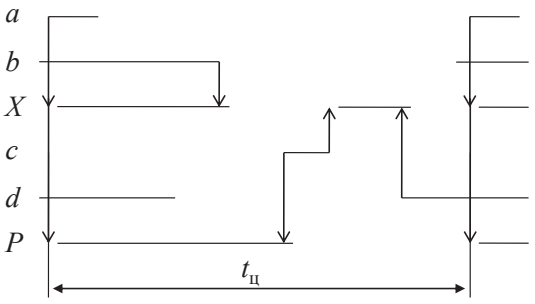


Рис. 2.10. Циклограмма, содержащая два периода включения выходной переменной

Для X_2 первая и вторая проверки дают отрицательный результат.

$$X = a + xb + \bar{c}\bar{d}.$$

После отключения сигнала d и до включения сигнала c выражение $\bar{c}\bar{d}$ равно единице, т. е. в отключающем периоде произойдет включение выходной переменной X. Для предотвращения ненужного, а часто и аварийного включения X, вводим промежуточную переменную:

$$P = a\bar{c} \overset{1\,пр.}{\rightarrow} a + p\bar{c},$$

на инверсное значение которой умножаем комбинацию $\bar{c}\bar{d}$:

$$X = a + xb + \bar{c}\bar{d} \overset{3\,пр.}{\rightarrow} a + xb + \bar{c}\bar{d}\bar{p}.$$

Как уже отмечалось ранее, в циклограмме время не оценивается количественно, на рисунках обычно такты представляются одинаковой длины; реальная временная продолжительность различных тактов может сильно отличаться. Например, длительность включающих и отключающих тактов определяется временем срабатывания аппаратуры

(от микросекунд до десятых долей секунду), а время включенного состояния элемента может составлять минуты и даже часы (в зависимости от характера технологического процесса).

Составим структурные формулы для выходных и промежуточных функций по заданной на рис. 2.11 циклограмме.

X имеет два периода включения, тогда $X = X_1 + X_2$. X_1 включается сигналом d, отключается сигналом $\overline{a}$:

$$X_1 = d\,a \overset{=1\,пр.}{\rightarrow} d + x\,\overline{a}.$$

Отключающий сигнал a меняет свое состояние во включающем периоде (вторая проверка дает положительный результат), поэтому надо ввести промежуточную переменную P_1, включив ее, например, сигналом a. Поскольку время включения P_1 (включающий такт для P_1) много меньше времени действия сигнала a, приходится вводить дополнительный такт, разделив такт 4 на два такта (пунктирная линия) и ввести новую нумерацию тактов (цифры во втором ряду). Поскольку вводимую переменную P_1 можно попытаться использовать в дальнейшем для решения других задач, вопрос ее отключения решим позднее.

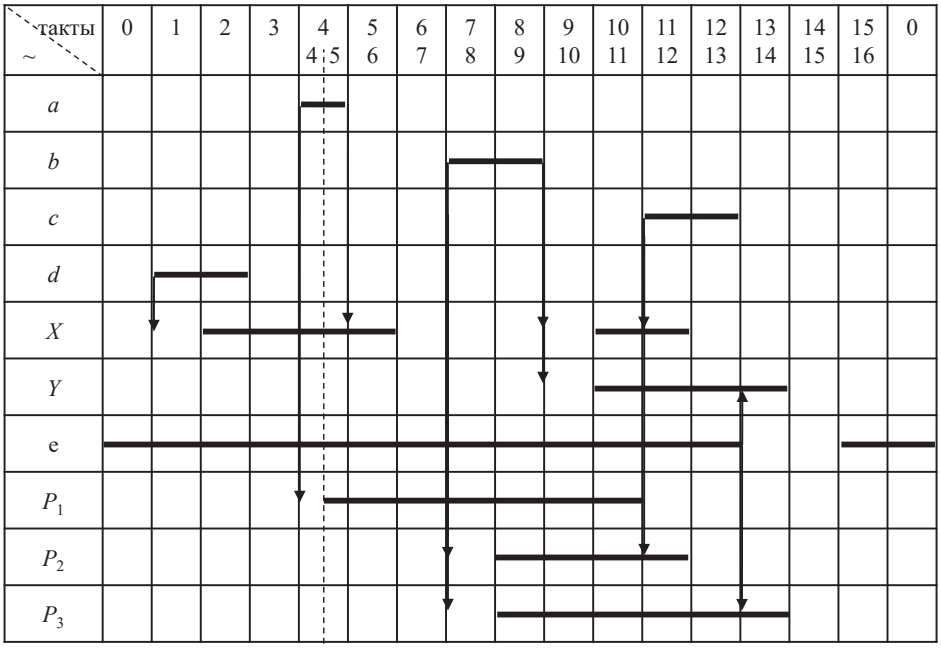


Рис. 2.11. Циклограмма, описывающая работу двух выходных элементов

Итак, после второй проверки получаем:

$$X_1 = d + x\overline{\bar{a}p_1} = d + x\left(a + \overline{p_1}\right) = d + xa + x\,\overline{p_1}.$$

$X_2 = \bar{b}\bar{c}$, первая и вторая проверки не требуют введения промежуточных переменных.

$X = d + xa + x\,\overline{p_1} + \bar{b}\bar{c}$, третья проверка показывает, что слагаемые d и xa не встречаются в отключающем периоде.

$x\,\overline{p_1}$, если P_1 отключить в шестом такте, будет равняться 1 в тактах 7–11 и далее. Поскольку в тактах 10 и 11 $X = 1$ ($\bar{b}\bar{c} = 1$), $x\,\overline{p_1}$ станет равной единице, и X не отключится до конца цикла, что недопустимо. Поэтому P_1 нужно отключать сигналом c одновременно с отключением X:

$$P_1 = a + p_1\bar{c}.$$

Слагаемое $\bar{b}\bar{c}$ будет равно единице в тактах 13–0–6, что нужно исключить, введя промежуточную переменную P_2:

$$P_2 = b + p_2\bar{c}.$$

Окончательно, $X = d + xa + x\,\overline{p_1} + \bar{b}\bar{c}p_2$.

Структурная формула для Y: $Y = \bar{b}\overline{\overline{e}} = \bar{b}e$.

Первая и вторая проверки не требуют использования промежуточных переменных, а третья проверка дает положительный результат. Использование уже имеющихся P_1 и P_2 не решает задачу, поэтому вводим P_3:

$$P_3 = b\overset{=1\,пр.}{e} \rightarrow b + p_3e;$$

$$Y = \bar{b}ep_3.$$

В итоге окончательные выражения имеют следующий вид:

$$X = d + xa + x\,\overline{p_1} + \bar{b}\bar{c}p_2;$$

$$Y = \bar{b}ep_3;$$

$$P_1 = a + p_1\bar{c};$$

$$P_2 = b + p_2\bar{c};$$

$$P_3 = b + p_3e.$$

На рис. 2.12 приведена релейно-контакторная схема, построенная в соответствии с полученными структурными формулами.

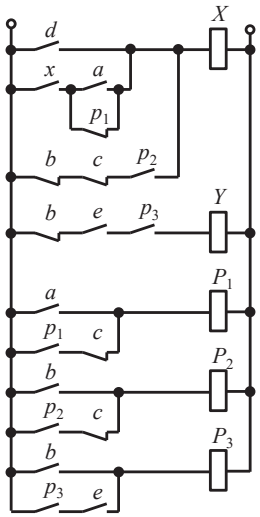


Рис. 2.12. Релейно-контакторная схема для выходных и промежуточных элементов

При реализации схемы с использованием бесконтактных логических элементов потребуется 7 элементов И и 4 элемента ИЛИ.

2.3. Особенности схемной реализации положительного результата второй и третьей проверок

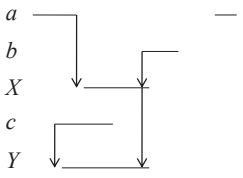


Рис. 2.13. Циклограмма, позволяющая перейти от третьей проверки к первой

Как уже отмечалось выше, прежде чем вводить по результатам проверки промежуточные переменные, нужно использовать имеющиеся входные и выходные переменные, если они удовлетворяют условиям применения этих промежуточных элементов. Довольно часто при положительном результате третьей проверки удается избежать введения промежуточной переменной за счет перевода начального алгоритма ($f'\overline{f''}$) в русло первой проверки. Рассмотрим это на примере рис. 2.13.

$X = \bar{a}\bar{b}$. На основании первой и второй проверок не требуется вводить промежуточные переменные. Третья проверка показывает, что в начальный момент времени, когда все входные сигналы равны нулю, произойдет включение элемента *X*, что недопустимо.

Меняем условие включения, вместо включающего сигнала $\bar{a}$ используем произведение $\bar{a}c$. Тогда $X = \bar{a}c \cdot \bar{b} \overset{1\,пр.}{\rightarrow} \bar{a}c + x\bar{b}$. В результате в начальный момент времени *X* не включится.

Рассмотрим еще один пример решения подобного рода проблем (рис. 2.14).

$$X = a\,\bar{b} \overset{=1\,пр.}{\rightarrow} a + xb \overset{2\,пр.}{\rightarrow} a + x\overline{\bar{b}z} = a + xb + x\bar{z};$$

$$Y = \bar{b}\bar{c}.$$

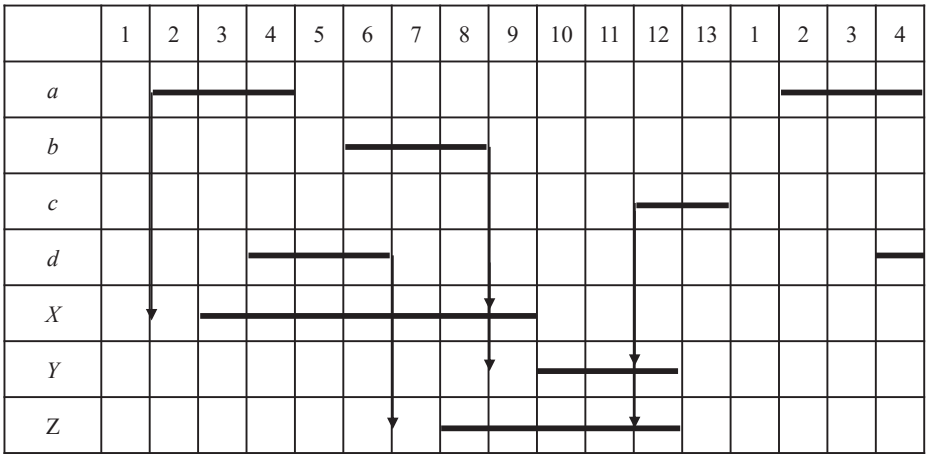


Рис. 2.14. Пример циклограммы, в которой можно перейти от третьей проверки к первой

Для элемента *Y* третья проверка дает положительный результат, *Y* включится в 1–5 тактах. Переводим алгоритм в русло первой проверки:

$$Y = \bar{b}x \cdot \bar{c} \overset{1\,пр.}{\rightarrow} \bar{b}x + y\bar{c}.$$

Теперь третья проверка конъюнкции $\bar{b}x$ дает ненужное включение *Y* в тактах 3–5. Исключаем это, используя переменную *Z*:

$$Y = \bar{b}xz + y\bar{c}.$$

Наконец, $Z = \bar{d}\bar{c} \overset{3\,пр.}{\rightarrow} \bar{d}b \cdot \bar{c} \overset{1\,пр.}{\rightarrow} \bar{d}b + z\bar{c}$.

2.4. Использование элементов «Память» при проектировании бесконтактных схем

Для бесконтактных схем удобнее пользоваться выражением, в котором участвуют формулы «Памяти» [6]. На рис. 2.15 представлено условное обозначение логического элемента «Память» или *RS*-триггера (*S* – включающий вход, *R* — отключающий вход), а табл. 2.1 поясняет его работу в течение технологического цикла.

Таблица 2.1

Таблица состояний *RS*-триггера

S	0	1	0	0	1
R	0	0	0	1	1
X	0	1	1	0	?

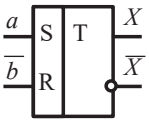


Рис. 2.15. Условное изображение *RS*-триггера

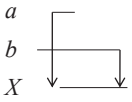


Рис. 2.16. Циклограмма, при реализации которой удобно применить триггер

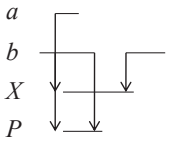


Рис. 2.17. Другой пример циклограммы, реализуемой при помощи триггера

Обратите внимание, что при комбинации входных сигналов $S = R = 0$ выходной сигнал X может быть равным как 0, так и 1. Это обусловлено тем, что при возникновении данной комбинации входных сигналов *RS*-триггер находится в состоянии хранения информации. То есть, если на предыдущем шаге записи информации выходной сигнал X был равен 0, то это состояние сохраняется и на текущем шаге. Данная ситуация аналогична и для $X = 1$.

Элемент «Память» применяется, когда первая проверка дает положительный результат. В этом случае на вход S подается включающий сигнал в том виде, в каком он существует во включающем такте, а на вход R — отключающий сигнал в том виде, в каком он существует в отключающем такте (рис. 2.16).

Первая и вторая проверки циклограммы, показанной на рис. 2.17, дают положительный результат.

$$X = a\bar{b} \overset{1\,пр.}{\rightarrow} a + x\bar{b} \overset{2\,пр.}{\rightarrow} a + x\overline{b\bar{p}} = a + x\bar{b} + xp,$$

$$P = a\,b \overset{=1\,пр.}{\rightarrow} a + pb.$$

Схема, представленная на рис. 2.18, реализует полученный алгоритм. При проектировании систем автоматики следует не допускать появления единичных сигналов одновременно на обоих входах, т. к. поведение *RS*-триггера в большинстве случаев оказывается неопределенным.

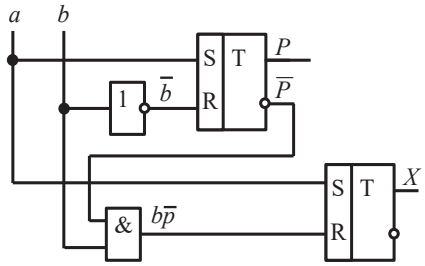


Рис. 2.18. Реализация циклограммы, изображенной на рис. 2.17

Вернемся к циклограмме, заданной рис. 2.11. Полученные при ее анализе структурные формулы выходных и промежуточных элементов реализуем схемами с использованием элемента «Память» (рис. 2.19).

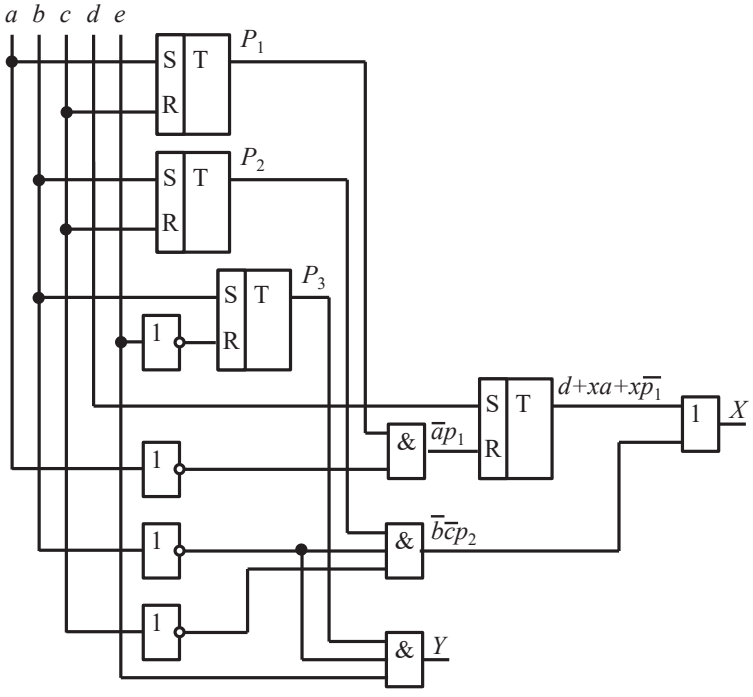


Рис. 2.19. Реализация циклограммы, заданной рис. 2.11, на основе триггеров

$$X = d + x\overline{\overline{a}p_1} + \overline{b}\overline{c}p_2;$$
$$Y = \overline{b}ep_3;$$
$$P_1 = a + p_1\overline{c};$$
$$P_2 = b + p_2\overline{c};$$
$$P_3 = b + p_3e.$$

Использование элементной базы избыточного базиса (И, ИЛИ, НЕ) при реализации этой схемы требует 11 элементов (7 элементов И и 4 элемента ИЛИ); использование элементов «Память» уменьшает количество элементов до 8 («Память» — 4, И — 3, ИЛИ — 1).

2.5. Синтез управляющего алгоритма на основе циклограмм, имеющих неоднозначное решение

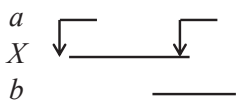


Рис. 2.20. Циклограмма, в которой совпадают условия срабатывания и несрабатывания

Рассмотрим несколько примеров, позволяющих выработать рекомендации для решения нетипичных задач.

Попытаемся получить структурную формулу выходного элемента X (рис. 2.20), по традиционной схеме: исходный алгоритм → первая проверка → вторая проверка → третья проверка.

$$X = a\overline{a} \overset{1\,пр.}{\rightarrow} a + x\overline{a}.$$

Выходная переменная не отключится в отключающем такте. Очевидно, необходимо разделить включающий и отключающий сигналы a с помощью промежуточной переменной, в данном случае с помощью переменной b, используя при этом элемент «Память».

Решение: X: $S = a\overline{b}$, $R = ab$.

Для циклограммы (рис. 2.21) структурная формула X соответствует двум периодам включения.

$$X = X_1 + X_2;$$
$$X_1 = a\overline{b} \overset{1\,пр.}{\rightarrow} a + x\overline{b};$$

$$X_2 = b\bar{a} \xrightarrow{1\,пр.} b + x\bar{a};$$

$$X = a + x\bar{b} + b + x\bar{a}.$$

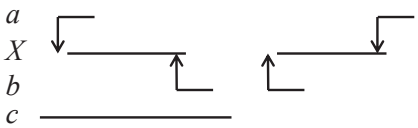


Рис. 2.21. Циклограмма, в которой чередуются включающий и отключающий сигналы

Включившись, X остается включенным до конца цикла и далее. Используя в качестве промежуточной переменной c, можно решить задачу с использованием элемента «Память».

$$X\colon\ S = ac + b\bar{c}\ ,\ R = bc + a\bar{c}\ .$$

Если отсутствует переменная, позволяющая решить задачу, ее необходимо создать таким образом, чтобы исключить возникающие неопределенности. При этом чаще всего необходимо использовать элемент «Память».

Выходной элемент X, работающий по циклограмме, показанной на рис. 2.22, имеет два периода включения.

$$X = X_1 + X_2;$$

$$X_1 = a\bar{\bar{b}};$$

$$X_2 = a\bar{b};$$

$$X = ab + a\bar{b}.$$

Получается, что от сигнала b переменная X не зависит. Для решения циклограммы вводим промежуточную переменную P, используя при этом элемент «Память»:

$$P\colon\ S = \bar{a}\bar{b}\ ,\ R = ab\ ,\ X = ab\bar{p} + a\bar{b}p\ .$$

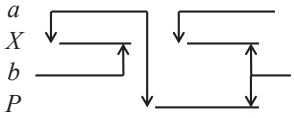


Рис. 2.22. Циклограмма с двумя периодами включения выходного элемента

Рассмотрим еще один популярный пример: открывание — закрывание дверей (шлагбаума) одной кнопкой (рис. 2.23).

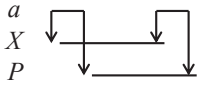


Рис. 2.23. Упрощенная циклограмма работы шлагбаума

Вводим промежуточную переменную P. Структурная формула схемы имеет вид: X: $S = a\overline{p}$, $R = ap$; P: $S = \overline{a}x$, $R = \overline{a}\,\overline{x}$. Схема устройства представлена на рис. 2.24.

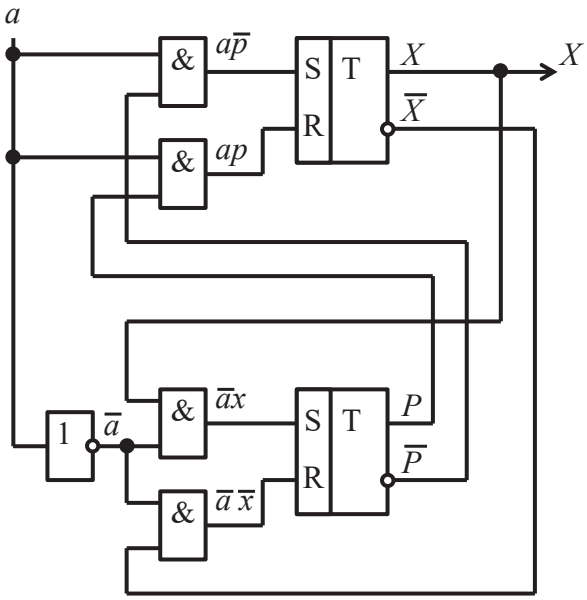


Рис. 2.24. Упрощенная схема системы управления шлагбаумом

Рассмотрим еще один пример, когда для решения задачи требуется введение промежуточных переменных. Две выходные переменные X_1 и X_2 (рис. 2.25) включаются соответственно при нечетных и четных импульсах входной переменной a.

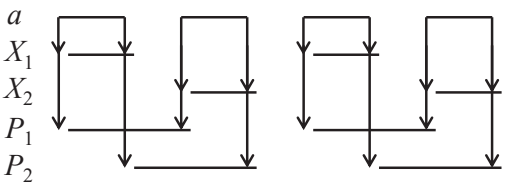


Рис. 2.25. Циклограмма работы делителя частоты

Структурные формулы промежуточных переменных:

$$P_1\colon\ S = a\overline{p_2},\ R = ap_2;\ P_2\colon\ S = \overline{a}p_1,\ R = \overline{a}\,\overline{p_1}.$$

Выходные переменные: $X_1 = ap_1$, $X_2 = ap_2$.

2.6. Получение полных формул системы управления

Полученные по циклограмме структурные формулы выходных переменных не содержат элементов защит и блокировок. Для каждой конкретной системы управления при проектировании выбираются необходимые защиты и блокировки, которые диктуются особенностями технологического процесса и применяемого оборудования, в первую очередь, электрического.

Рассмотрим решение этой задачи на примере получения полной схемы для реверсивного механизма с асинхронным короткозамкнутым двигателем с питанием от тиристорного преобразователя напряжения (ТПН). Силовая часть привода традиционная, рассматривать ее не будем. Управление механизмом осуществляется с помощью кнопок В — вперед, Н — назад, С — стоп (входные сигналы циклограммы, рис. 2.26). X_1 — движение вперед, X_2 — движение назад.

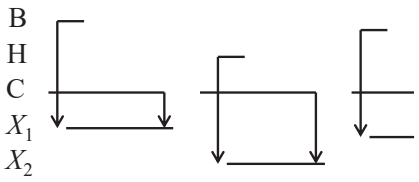


Рис. 2.26. Циклограмма работы пускателя

$$X_1 = \mathrm{BC} \overset{=1\,\text{пр.}}{\rightarrow} (\mathrm{B} + X_1)\mathrm{C};$$

$$X_2 = \mathrm{HC} \overset{=1\,\text{пр.}}{\rightarrow} (\mathrm{H} + X_2)\mathrm{C}.$$

Для исключения одновременного включения X_1 и X_2 необходимо ввести взаимную блокировку.

$$X_1 = (\mathrm{B} + X_1)\mathrm{C}\overline{X_2};$$

$$X_2 = (\mathrm{H} + X_2)\mathrm{C}\overline{X_1}.$$

В формулы необходимо ввести сигналы датчиков максимальной (М) и тепловой (Т) защиты двигателя. При срабатывании любой защиты двигатель должен отключиться.

$$X_1 = (\mathrm{B} + X_1)\mathrm{C}\overline{X_2}\overline{\mathrm{M}}\overline{\mathrm{T}};$$

$$X_2 = (\text{Н} + X_2)\text{С}\overline{X_1}\overline{\text{М}}\overline{\text{Т}}.$$

При использовании элементов «Память» включающие сигналы будут иметь вид:

$$X_1{:}\ S = \text{В}\overline{X_2}\ ;\ X_2{:}\ S = \text{Н}\overline{X_1}\ .$$

Для отключения X_1 и X_2 используется общий сигнал:

$$R = \overline{\text{С}\overline{\text{М}}\overline{\text{Т}}} = \overline{\text{С}} + \text{М} + \text{Т}.$$

Схема, реализующая представленные алгоритмы, показана на рис. 2.27.

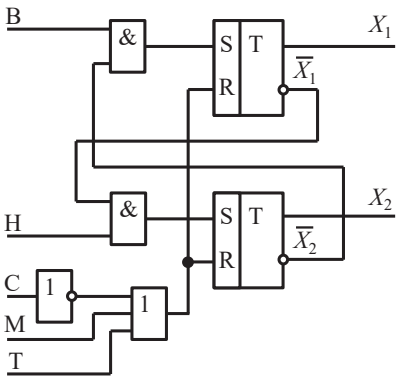


Рис. 2.27. Схема, реализующая работу пускателя с учетом блокировок

Таким образом, при наличии сигнала с одного из датчиков или с обоих одновременно, двигатель будет отключен.

2.7. Составление структурных формул по релейно-контакторной схеме

Необходимость получения структурных формул систем технологической автоматики связано чаще всего с модернизацией действующих релейно-контакторных схем (РКС) при переходе на бесконтактные системы для получения высокой точности и быстродействия.

В результате анализа РКС производится подразделение всех действующих сигналов на входные, выходные и промежуточные [5]. Каждому сигналу присваивается буквенное обозначение согласно принятому в начале курса правилу. Затем производится запись алгебраических выражений, соответствующих цепям выходных и промежуточных переменных РКС.

РКС в большинстве случаев имеют последовательно-параллельную структуру функциональных узлов (схемы класса П). Алгебраические выражения для схем класса П записываются в нормальных формах (ДНФ и КНФ). При наличии узлов с мостиковыми структурами соединения контактов (схемы класса Н) для получения алгебраического выражения сигнала, идущего к определенному элементу, необходимо записывать структурные формулы для всех возможных цепей включения этого элемента.

Таким образом, рекомендуется следующий порядок составления алгебраических выражений:

– составить алгебраические выражения для выходных сигналов;

– составить алгебраические выражения для промежуточных сигналов без обратных связей;

– составить алгебраические выражения для промежуточных сигналов с обратными связями;

– в выражениях выходных сигналов и промежуточных сигналов с обратными связями заменить значения встречающихся промежуточных сигналов без обратных связей их выражениями через входные сигналы;

– упростить полученные выражения, если это окажется возможным, на основе законов алгебры логики;

– составить логическую схему управления из элементов И, ИЛИ, НЕ, реализующую полученные выражения.

2.7.1. Построение бесконтактных схем по релейно-контакторным схемам класса П

При получении структурной формулы по релейно-контакторной схеме с последовательно-параллельным соединением контактов вспомним, как связаны элементы РКС и элементы структурных формул:

– нормально разомкнутому контакту соответствует прямое значение переменной;

– нормально замкнутому контакту соответствует инверсное значение переменной;

– последовательному соединению контактов соответствует конъюнкция переменных;

– параллельному соединению контактов соответствует дизъюнкция переменных.

Последовательность действий при получении структурной формулы покажем на примере схемы, изображенной на рис. 2.28.

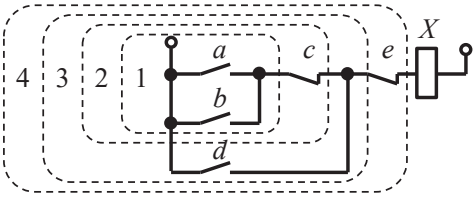


Рис. 2.28. Пример схемы для получения по ней структурной формулы

Начинаем запись формулы с верхней левой точки схемы:

– первый контур: $a+b$;

– второй контур: $(a+b)\overline{c}$;

– третий контур: $(a+b)\overline{c}+d$;

– последний (четвертый) контур: $\left[(a+b)\overline{c}+d\right]\overline{e}=X$.

Рассмотрим еще один пример получения структурной формулы по РКС (рис. 2.29).

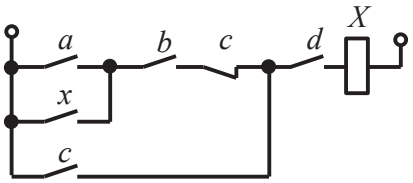


Рис. 2.29. Второй пример схемы для получения по ней структурной формулы

По схеме можно получить выражение $X=\left[(a+x)b\overline{c}+c\right]d$.

Практика показывает, что довольно часто РКС содержат избыточные элементы. Поэтому прежде чем приступить к реализации полученной формулы, нужно ее минимизировать, используя законы алгебры логики. В нашем случае выражение в квадратных скобках упрощается (закон № 9):

$$(a+x)b\overline{c}+c=(a+x)b+c.$$

В итоге окончательная формула $X = \left[\left(a + x \right) b + c \right] d$ может быть реализована с использованием избыточного базиса (И, ИЛИ, НЕ), а также с использованием элемента «Память». Читателю предлагается решить эту задачу самостоятельно.

2.7.2. Построение бесконтактных схем по РКС класса Н

В релейно-контакторных схемах промышленных установок встречаются цепи с мостиковыми соединениями контактных цепей. Эти схемы называются схемами класса Н (непараллельно-последовательная структура). Применение мостиковых соединений приводит к существенному сокращению числа контактов. На рис. 2.30 представлена простейшая мостиковая схема класса Н. В этих схемах отдельные параллельные цепи соединяются между собой в промежуточных точках между контактами при помощи мостовых соединений, в цепях которых могут быть включены контакты или катушки аппаратов, обмотки машин и пр.

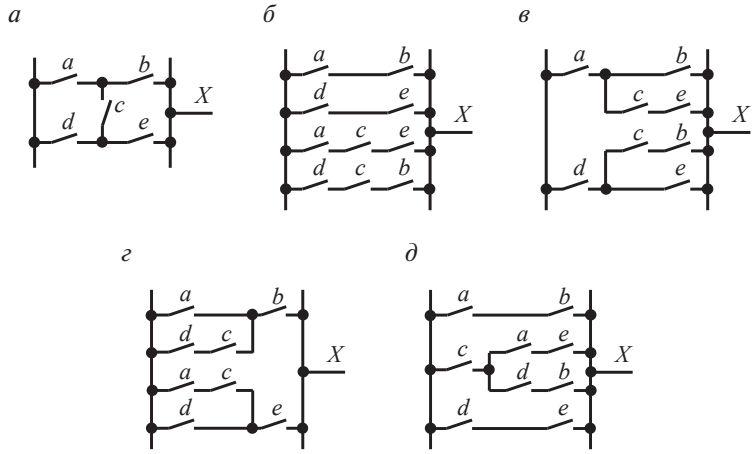


Рис. 2.30. Примеры мостиковых схем

Из рассмотрения рис. 2.30 (*а*) видно, что в схемах класса Н каждый из начальных структурных элементов (в данном случае *a* и *d*) соединен последовательно с каждым из конечных структурных элементов (в данном случае *b* и *e*). Структурные элементы, включенные в мостовое соединение (в данном случае *c*), входят в несколько различных цепей, которые могут образоваться в схеме между ее начальными и конеч-

ными полюсами. Поэтому схему можно описать несколькими структурными формулами:

$$X = ab + de + ace + dcb;$$
$$X = a\left(b + ce\right) + d\left(e + cb\right);$$
$$X = b\left(a + dc\right) + e\left(d + ac\right);$$
$$X = ab + de + c\left(ae + db\right).$$

Все структурные формулы могут быть получены из первой путем равносильных преобразований. Каждой из формул соответствует определенная схема на рис. 2.30 (*б*)–(*д*). Эти схемы равносильны по действию схеме, показанной на рис. 2.30 (*а*), но существенно сложнее ее. Число контактов в мостиковой схеме значительно меньше, чем в равносильных ей последовательно-параллельных структурах. На рис. 2.31 приведены бесконтактные схемы, соответствующие контактным схемам на рис. 2.30 (*б*) и 2.30 (*в*). Вариант бесконтактной схемы на рис. 2.31 (*а*) содержит меньше элементов.

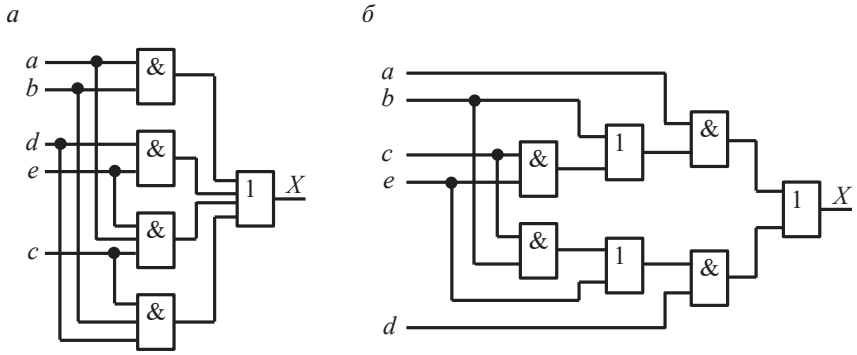


Рис. 2.31. Бесконтактные схемы,
эквивалентные схемам *б* и *в* на рис. 2.30

Таким образом, при построении бесконтактной схемы по релейно-контактной схеме, содержащей узлы с мостиковыми соединениями контактов (схема класса Н), необходимо привести эти схемы к эквивалентным схемам последовательно-параллельной структуры (схемы класса П). Ниже рассматриваются методы решения указанной задачи.

Методы выделения цепей, проходящих через начальные или конечные элементы. На рис. 2.32 (*а*) представлена исходная релейно-контактная мостиковая схема. Разложение может быть проведено как по начальным, так и по конечным элементам схемы. В качестве начальных

элементов могут быть приняты, например, элементы g и X, а в качестве конечных — элементы a и b.

Для получения последовательно-параллельной структуры, по которой могут быть составлены структурные формулы, например для элементов X и Y, можно разложить исходную схему на рис 2.32 (*а*) по начальным элементам так, как это показано на рис. 2.32 (*б*). На схеме вычерчены отдельные цепи питания элементов X и Y.

Поскольку в схеме все еще имеются мостиковые соединения, то ее следует снова разложить по начальным элементам. В данном случае элементы d и f будут начальными элементами для цепи элемента X, а e и f — для цепи элемента Y.

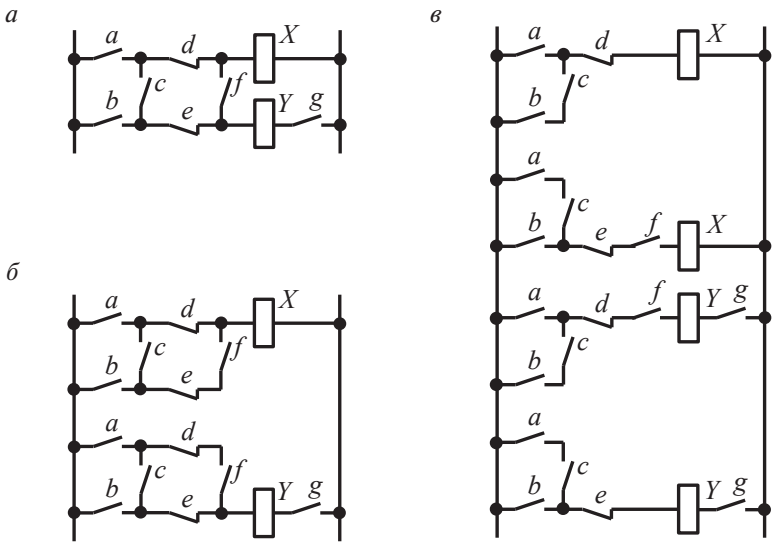


Рис. 2.32. Пример разложения
мостиковой схемы по начальным элементам

При разложении схемы по конечным элементам производятся те же операции, только с другого полюса схемы.

По схеме на рис. 2.32 (*в*) можно записать структурные формулы, необходимые для построения бесконтактной схемы:

$$X = (a + bc)\bar{d} + (ac + b)\bar{e}f;$$

$$Y = (a + bc)\bar{d}fg + (ac + b)\bar{e}g = g\left[(a + bc)\bar{d}f + (ac + b)\bar{e}\right].$$

Метод записи структурных формул для всех возможных цепей включения элементов. Этот метод уже был ранее применен к схеме

на рис. 2.30 (*а*). Применение этого метода к схеме на рис. 2.32 (*а*) для всех возможных цепей включения элементов X и Y и выполнение равносильных преобразований полученных структурных формул приводят к следующим выражениям:

$$X = a\bar{d} + bc\bar{d} + (a + bc)\bar{d} + b\bar{e}f + ac\bar{e}f + (ac + b)\bar{e}f =$$
$$= (a + bc)\bar{d} + (ac + b)\bar{e}f;$$
$$Y = \left[b\bar{e} + ac\bar{e} + (ac + b)\bar{e} + a\bar{d}f + bc\bar{d}f + (a + bc)\bar{d}f \right] g =$$
$$= \left[(ac + b)\bar{e} + (a + bc)\bar{d}f \right] g.$$

2.8. Построение рабочей циклограммы

Исходным материалом является технологическая циклограмма, которая показывает последовательность работы механизмов агрегата, поточной линии и тому подобного в цикле. Включение и отключение каждого механизма осуществляется с помощью электропривода, соленоидов и других электротехнических устройств. Управляющие команды (входные сигналы) формируются аппаратами ручного управления (кнопки, тумблеры) и датчиками (индуктивные, фото, импульсные и другие). Их расположение определяется при проектировании, что позволяет представить состояние выходного сигнала датчика (1 или 0) в течение цикла.

Построение циклограммы рассмотрим на примере относительно простой схемы, которая управляет двумя гидравлическими цилиндрами X и Y, обслуживающими участок автоматической линии (рис. 2.33). Цилиндры работают в следующей последовательности: X, Y, $\bar{Y}$, $\bar{X}$.

Каждое последующее действие начинается только после окончания предыдущего. Сначала включается цилиндр X, и его шток, связанный с устройством, которое зажимает обрабатываемую деталь на рабочей позиции, идет вперед. После окончания его хода включается цилиндр Y, шток которого соединен со сверлильной головкой. После завершения обработки отверстий сверлильная головка возвращается в исходное положение, а затем возвращается и шток цилиндра X, освобождая деталь.

Для контроля положения штоков цилиндров используются путевые переключатели A_1, A_2, B_1, B_2. Сигналы с этих переключателей (a_1, a_2, b_1, b_2) являются входными переменными проектируемой схемы.

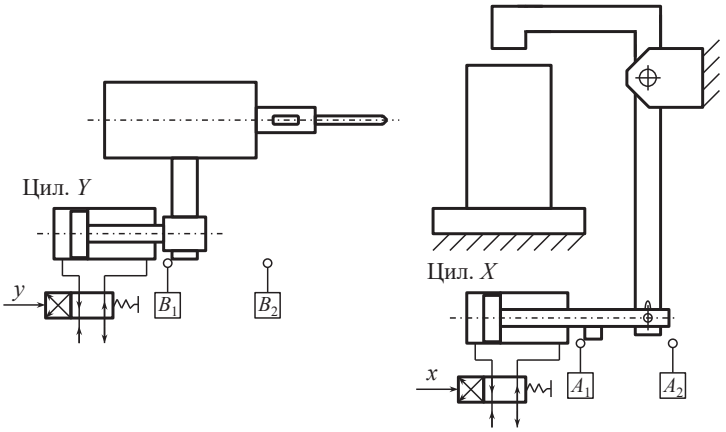


Рис. 2.33. Схема управления двумя гидроцилиндрами

Последовательная запись изменения состояний входных и выходных переменных называется таблицей включений (табл. 2.2). На основании этой таблицы строится начальная циклограмма. Переход от одного состояния к другому происходит почти мгновенно, поэтому на циклограмме он изображается вертикальной линией, т. е. вертикальные линии соответствуют моментам изменения входных и выходных сигналов. Расстояние между соседними вертикальными линиями есть такт работы системы (рис. 2.34).

Таблица 2.2

Таблица включений элементов станка

такт	входная ~	выходная ~
1	$a_1 = 1$	$x = 1$
2	$a_1 = 0$	—
3	$a_2 = 1$	$y = 1$
4	$b_1 = 1$	—
5	$b_2 = 1$	$\bar{y} = 1$
6	$b_2 = 0$	—
7	$b_1 = 1$	$\bar{x} = 1$
8	$a_2 = 0$	—

Следующий шаг — получение промежуточной циклограммы (рис. 2.35), где пунктирной линией показано состояние выходных переменных x, y и инверсных им выходов $\bar{x}$ и $\bar{y}$. Соединив между собой

концы линии, на которых нет штрихов, получим циклограмму состояний входных переменных a_1, a_2, b_1, b_2. С учетом принятых допущений (мгновенное включение выходных переменных) на циклограмме отсутствуют включающие и отключающие такты выходных переменных.

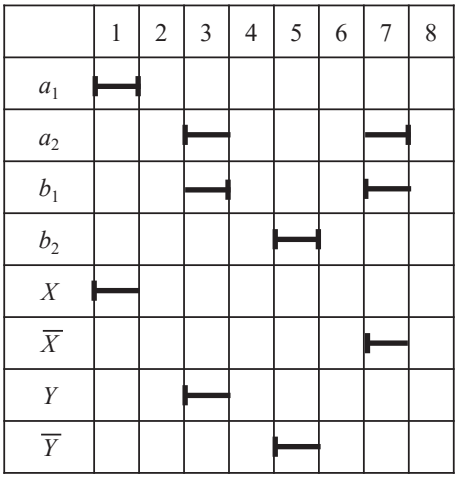


Рис. 2.34. Начальная циклограмма работы элементов станка

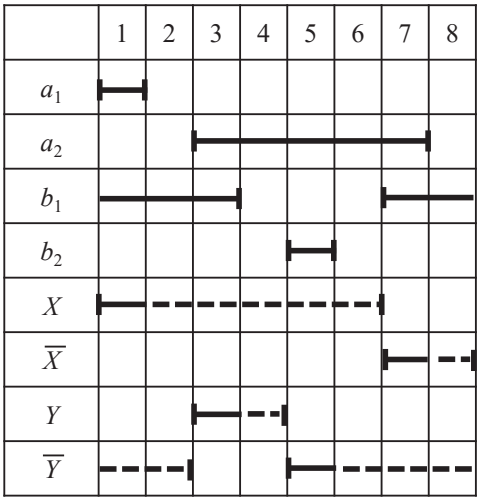


Рис. 2.35. Промежуточная циклограмма

Кроме того, по окончании цикла нужно снять готовую деталь и заменить ее следующей заготовкой. Поэтому сигнал a_1, необходимый для работы схемы, не должен сразу же запускать схему. Эту задачу

решает вводимая кнопка П, которая запускает схему после завершения всех подготовительных работ. В результате получаем окончательную циклограмму (рис. 2.36).

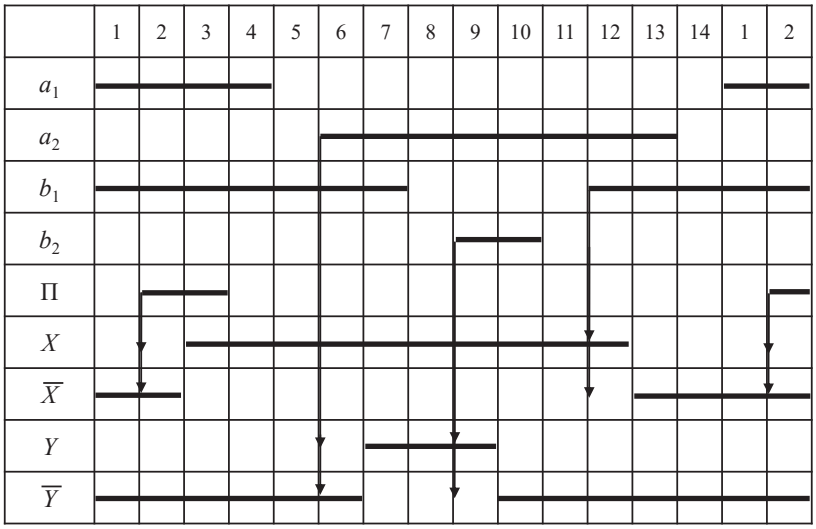


Рис. 2.36. Окончательная циклограмма

Как видно, на окончательной циклограмме (рис. 2.36) количество тактов увеличилось до 14. Это связано с вводом в систему управления кнопки «Пуск» (П), а также с вводом включающих и отключающих тактов и устранения возникающей при этом проблемы многозначности внутри такта. Так, на рис. 2.35 в такте 5 сигнал b_2 отключает y. Если мы введем отключающий такт для y, используя для этого имеющийся такт 5, то окажется, что длительность отключения y (составляющая в бесконтактных схемах микросекунды) и длительность сигнала датчика b_2 (секунды) равны. Поэтому необходимо ввести дополнительный такт. На рис. 2.36 это такт 9 (отключающий для Y), а сигнал b_2 теперь длится 2 такта — девятый и десятый.

Реализацию схемы лучше осуществлять с использованием элементов «Память», поскольку *RS*-триггер имеет прямой и инверсный выходы. Читателю предоставляется возможность найти структурную формулу схемы с использованием элементов «Память». Для выходной переменной X требуется реализовать положительные результаты первой и второй проверок, для Y — третьей проверки.

Контрольные вопросы к главе 2

1. Что есть циклограмма?
2. Как звучит первая проверка, выполняемая при синтезе алгоритма работы системы автоматического управления с использованием циклограммы?
3. Как проводится вторая проверка, выполняемая при поиске формулы логической функции с использованием циклограммы?
4. В чем заключается сущность третьей проверки, выполняемой при синтезе алгоритма логической функции с использованием циклограммы?
5. Как функционирует логический элемент типа «Память»?
6. Каковы основные правила, применяемые при построении релейно-контакторных схем на основе формулы логической функции?
7. Перечислите основные правила, применяемые при реализации логических функций на базе бесконтактных логических элементов.

Глава 3.

Техническая реализация логических функций

Реализация логических функций возможна с использованием релейно-контакторной аппаратуры, бесконтактных логических элементов и программируемых контролеров [8]. Релейно-контакторные схемы для современных систем автоматики не проектируются [6]. Бесконтактные системы автоматики на логических элементах эксплуатируются на ряде объектов (например, в системах управления лифтами), но в процессе модернизации заменяются на современные с использованием программируемых логических контроллеров (ПЛК). Новые проекты выполняются с использованием ПЛК практически всеми ведущими фирмами-производителями средств автоматизации.

Для бесконтактных логических элементов уровень единицы определяется напряжением питания и устанавливается производителем (для элементов серии К155 это 5 В, К511–15 В). В системах автоматизации с программируемыми контроллерами чаще всего это 24 В [9]. Уровень сигналов датчиков (входные сигналы) лежит в широком диапазоне: от долей вольта (шунтовой датчик тока) до 100 В и более (датчик скорости). Поэтому необходимо согласовать сигналы датчиков с системой управления (СУ), для чего применяется устройство сопряжения с объектом ($УСО_1$) (рис. 3.1).

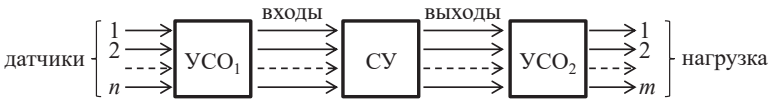


Рис. 3.1. Структура системы управления

Основные задачи, которые решает $УСО_1$:

– нормализация входного сигнала по уровню;

– гальваническая (потенциальная) развязка входов СУ от линий связи с датчиками;

– фильтрация входного сигнала (например, от высших гармоник сигналов датчиков скорости или тока).

К выходам СУ можно подключать относительно маломощные нагрузки, поэтому необходимо усилить мощность выходного сигнала. $УСО_2$ решает следующие задачи:

– усиление мощности выходного сигнала;

– гальваническая развязка выходов СУ от линий связи с нагрузками.

Рассмотрим структуру УСО на примере системы автоматики с использованием ПЛК (рис. 3.2). Сигналы датчиков аналоговых сигналов ДАС (например, температуры, давления) с помощью схемы нормализации $СН_1$ согласуются по уровню с напряжением ПЛК и с помощью коммутатора К подаются на вход аналого-цифрового преобразователя АЦП, который последовательно преобразует сигналы всех аналоговых датчиков в двоичный код. С помощью канала ввода-вывода КВВ информация загружается в оперативное запоминающее устройство ОЗУ. Сигналы датчиков дискретных сигналов ДДС (индуктивные, фотодатчики, кодовые датчики) с помощью коммутатора дискретных сигналов $КДС_1$ подаются в схему нормализации $СН_2$, которая решает две задачи:

– согласование по уровню с напряжением ПЛК;

– преобразование рабочего кода датчика в двоичный код.

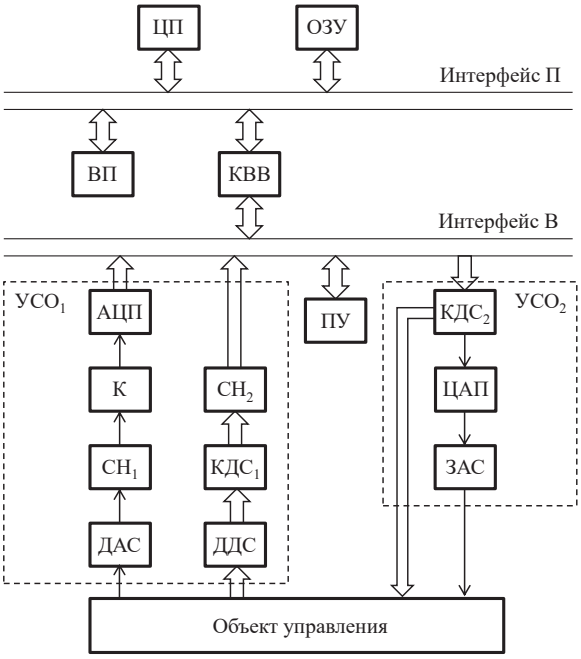


Рис. 3.2. Структура системы автоматики

После этого КВВ загружает информацию в ОЗУ. Когда информация о состоянии всех датчиков загружена в ОЗУ, центральный процессор ЦП обрабатывает программу, и выходные управляющие сигналы с помощью КВВ отправляются на объект управления через коммутатор дискретных сигналов $КДС_2$. Формирование аналоговых сигналов управления осуществляется с помощью цифро-аналогового преобразователя ЦАП и задатчика аналоговых сигналов ЗАС. Последний, как правило, представляет собой усилитель.

Элементная база, используемая в УСО (ЦАП, АЦП, коммутаторы, усилители, датчики), описывается в литературе, посвященной элементам систем автоматики и электронике. Применяемые в УСО преобразователи кодов являются элементами комбинационной логики и выпускаются производителями подобной техники. Поскольку вопросы использования специальных кодов в системах автоматизации нигде не рассматриваются, нам следует устранить этот пробел.

3.1. Коды, применяемые в АСУТП

Код выбирают таким образом, чтобы дальнейшее преобразование информации выполнялось возможно проще, а именно — чтобы легче было:

– реализовать арифметические операции;
– провести подсчет сигналов;
– расшифровать код (декодирование);
– обнаружить и исправить ошибку;
– привести к виду, удобному для использования внешними устройствами.

Классификация кодов представлена ниже.

Все коды, служащие для представления чисел, можно разделить на две группы — взвешенные и невзвешенные [2]. Взвешенные коды — это такие коды, в которых каждому разряду присваивается определенный вес. Например, натуральный двоичный код является взвешенным, поскольку каждому разряду соответствует вес, равный степени числа 2. Невзвешенные коды — это коды, разрядам которых нельзя присвоить веса, а соответствие кода и числа устанавливается каким-либо иным способом.

Рассмотрим двоично-десятичные коды. Для выражения каждой десятичной цифры (0, 1, 2, ..., 9) нужно четыре двоичных цифры, т. е.

четыре бита, которые позволят получить $2^4 = 16$ комбинаций, из которых шесть являются избыточными. На практике используются такие коды, которые обеспечивают самую рациональную и простую обработку цифровой информации, позволяющие переводить десятичные цифры $(X)_{10}$ в двоичные $(X)_2$ по формуле

$$(X)_{10} = a_3x_3 + a_2x_2 + a_1x_1 + a_0x_0, \qquad (3.1)$$

где символы $a_3...a_0$ являются постоянными весовыми коэффициентами существующего кода, а символы $x_3...x_0$ — двоичные цифры 1 или 0. Существуют различные взвешенные двоично-десятичные коды, называемые в соответствии с весовыми коэффициентами $a_3...a_0$: 8421, 7321, 5421, 4311 и др. В коде 8421 $a_3 = 8$, $a_2 = 4$, $a_1 = 2$, $a_0 = 1$. Кодовые комбинации, соответствующие тем или иным символам, в различных кодах легко определяются из выражения (3.1). Например,

$$(7)_{10} = 8\cdot 0 + 4\cdot 1 + 2\cdot 1 + 1\cdot 1 = (0111)_{8421};$$
$$(7)_{10} = 5\cdot 1 + 2\cdot 1 + 1\cdot 0 + 1\cdot 0 = (1100)_{5211};$$
$$(7)_{10} = 3\cdot 1 + 3\cdot 1 + 2\cdot 0 + 1\cdot 1 = (1101)_{3321}.$$

Из большого числа взвешенных кодов широко применяются самодополняющиеся коды. Формула такого кода имеет вид $N + \bar{N} = 9$.

Например,

$$N = (3)_{10} = (0100)_{4311},$$
$$\bar{N} = (6)_{10} = (1011)_{4311},$$
$$N + \bar{N} = (0100)_{4311} + (1011)_{4311} = (1111)_{4311} = 9.$$

Иначе говоря, самодополняющиеся взвешенные коды имеют сумму весов $(a_3 + a_2 + a_1 + a_0)$, равную девяти.

Примеры самодополняющихся кодов: 2421 (код Айкена), 3321, 4321, 5211 (код ЧПУ).

Невзвешенные коды — это циклические коды, в которых комбинации двоичных цифр, отображающие числа, которые отличаются друг от друга на единицу, разнятся только в одном разряде. Самым распространенным в этой группе является код Грея (табл. 3.1), который используется в системах контроля линейного перемещения или кругового движения какого-либо механизма.

Далее рассмотрим коды с обнаружением ошибок. Для ряда производств (системы управления атомными или химическими реакторами) случайные помехи могут вызвать остановку (отключение) систем автоматики, т. к. четырехразрядные самодополняющиеся коды не являются помехоустойчивыми. В таких случаях применяются коды с обнаружением ошибки, которые используют дополнительную информацию, для чего вводится пятый разряд.

Таблица 3.1

Таблица, связывающая код Грея и натуральный двоичный код

Число	**Натуральный двоичный код (8421)**				**Код Грея**			
S	x_3	x_2	x_1	x_0	y_3	y_2	y_1	y_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1

В качестве примера рассмотрим двоично-десятичный код с проверкой на четность (табл. 3.2).

Таблица 3.2

Двоично-десятичный код с проверкой на четность

Число	**ДДК**				***P***
	8	**4**	**2**	**1**	
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1

Окончание табл. 3.2

Число	ДДК				*P*
	8	**4**	**2**	**1**	
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1

Для каждого числа (от 0 до 9) находится сумма единиц кодового набора. Четной сумме соответствует 1 в дополнительном разряде *P*. Если в процессе работы значение текущей суммы не совпадает со значением *P*, то фиксируется ошибка. Использование такого кода требует дополнительных ресурсов — сумматоров и компараторов, что удорожает и усложняет систему.

Существуют коды, которые корректируют ошибку (случайный сбой). В них используется уже два дополнительных разряда и требуется дополнительная память.

3.2. Использование бесконтактных логических элементов при реализации логических функций

Современные системы автоматики проектируются с использованием ПЛК, что позволяет на одном технологическом и электрическом оборудовании производить различную продукцию, меняя последовательность технологических операций за счет изменения программы ПЛК. До появления ПЛК системы автоматики с использованием логических элементов обеспечивали высокое качество управления по сравнению с релейно-контакторными схемами. Поскольку многие системы автоматики этого класса продолжают эксплуатироваться и в наше время, надо знать особенности применяемых в промышленности серий бесконтактных логических элементов, принципы построения и структуры таких систем автоматики.

Наибольшее применение в системах промышленной автоматики нашли следующие серии логических элементов [1]:

– К155, базовый элемент — И-НЕ, напряжение питания 5 В;
– К176, базовый элемент — ИЛИ-НЕ, напряжение питания 9В.

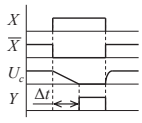
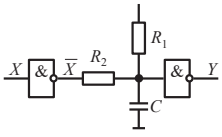
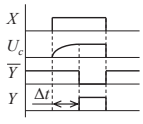
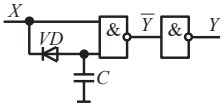
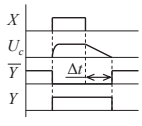
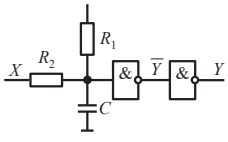
Наиболее проработанной и разветвленной является серия К155. Выпускаются логические элементы И-НЕ, ИЛИ-НЕ, И, ИЛИ с различным числом входов, комбинированные элементы (например, 2И-2ИЛИ-НЕ), исключающие ИЛИ (XOR), различные типы триггеров. Кроме того, для решения других задач предлагаются различные виды дешифраторов, счетчиков, сумматоров, селекторов-мультиплексоров, преобразователей кода, ПЗУ, ОЗУ и тому подобное.

В номенклатурах интегральных микросхем К155, К176 и других отсутствуют элементы времени. Поэтому выдержки времени при включении, отключении, формировании временных импульсов реализуются за счет использования *R-C* цепей и простых логических элементов (И-НЕ, И). Типовые схемы временных элементов представлены в табл. 3.3.

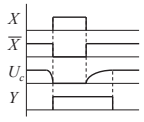
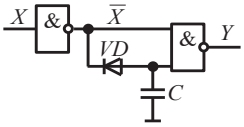
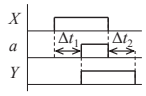
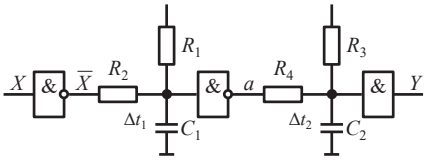
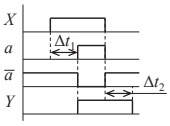
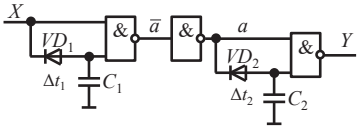
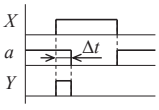
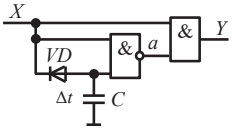
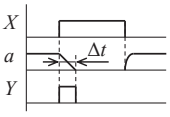
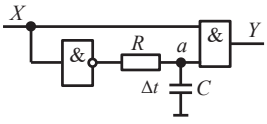
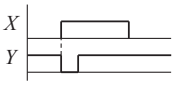
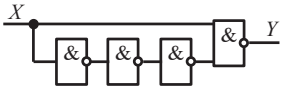
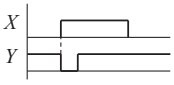
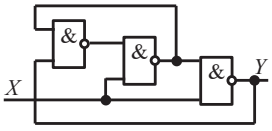
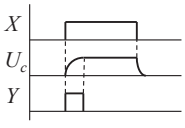
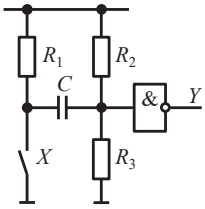
Схемы 1 и 2 — задержка переднего фронта входной переменной *X*, т. е. выдержка времени при включении. Схемы 3 и 4 — задержка заднего фронта входной переменной *X*, т. е. выдержка времени при выключении. Схемы 5 и 6 — задержка обоих фронтов входной переменной *X*, т. е. выдержка времени при включении и выключении, иными словами — сдвиг во времени. Схемы 7...12 — формирование короткого импульса по переднему фронту входного сигнала *X*. Схема 13 — формирование короткого импульса по заднему фронту входного сигнала *X*. Схемы 14 и 15 — одновибраторы.

Таблица 3.3

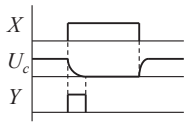
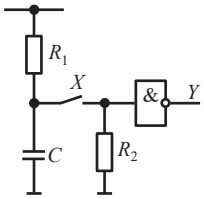
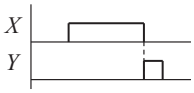
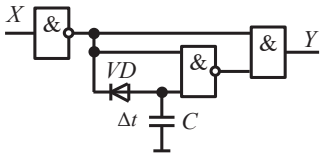
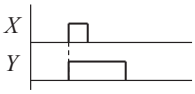
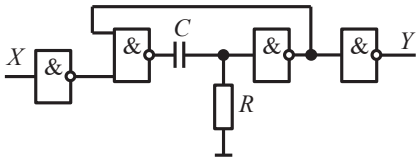
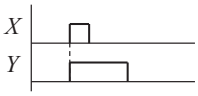
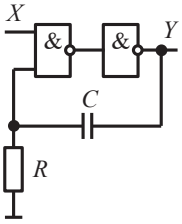
Типовые схемы временных элементов

№	Диаграмма сигналов	Схема реализации
Выдержка времени при включении		
1		
2		
Выдержка времени при отключении		
3		

Продолжение табл. 3.3

№	Диаграмма сигналов	Схема реализации
4		
Выдержка времени при включении и отключении		
5		
6		
Формирование импульса при включении		
7		
8		
9		
10		
11		

Окончание табл. 3.3

№	Диаграмма сигналов	Схема реализации
12		
Формирование импульса при отключении		
13		
Одновибраторы		
14		
15		

Схемы 9 и 10 работают без использования *R*-*C* цепочек. Выдержка времени оказывается очень мала и равна суммарному времени включения трех последовательно включенных элементов И-НЕ. Если в схеме 9 заменить элемент И-НЕ на элемент И, то диаграмма работы будет такая же, как для схемы 11, но длительность импульса будет равна суммарному времени срабатывания трех элементов И.

Еще одна задача, требующая решения, — устройство сопряжения с входными и выходными цепями. Возможные варианты представлены в табл. 3.4.

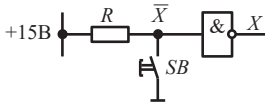
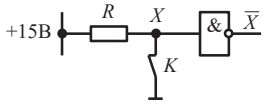
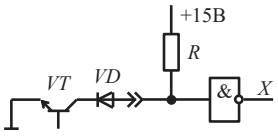
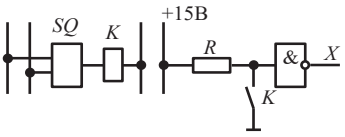
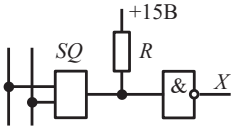
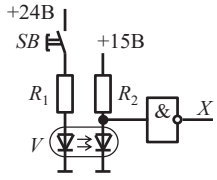
В интегральных микросхемах К155, К176 и других сериях отсутствуют элементы времени и согласующие элементы, кроме того, они рассчитаны на низкий уровень напряжения. Это дало толчок к разра-

ботке и внедрению в 80-х гг. XX в. серии «Логика-И», имеющей напряжение питания 15 В, что обеспечило простоту и удобство управления современными системами электроприводов. Серия включает в себя пять групп элементов:

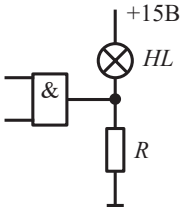
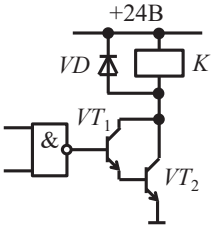
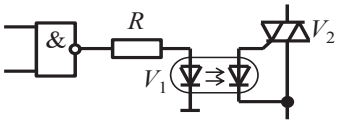
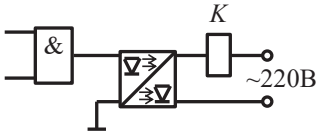
- логические элементы — И-101...И-123;
- функциональные элементы — И201...И-209;
- временные элементы — И-301, И-302;
- выходные элементы — И-401...И-406;
- элементы контроля — И-501, И-502.

Таблица 3.4

Варианты сопряжений с микросхемами

Назначение	Схема
Примеры связи с входными цепями	
Связь с контактными элементами с прямым выходным сигналом *X*	
То же с инверсным выходным сигналом *X*	
Связь с выходными цепями ЧПУ	
Связь с бесконтактным путевым датчиком с релейной развязкой	
Связь с бесконтактным датчиком	
Развязка при помощи оптронного элемента	

Окончание табл. 3.4

Назначение	Схема
Примеры связи с выходными цепями	
Включение индикаторной лампы	
Включение промежуточного реле	
Управление силовым ключом	
Включение силового аппарата переменного тока	

В табл. 3.5 представлено назначение некоторых элементов серии.

Таблица 3.5

Элементы серии Логика-И

Тип	Назначение элемента
Логические элементы	
И-101	Семь логических схем НЕ реализуют логическую функцию $Y = \bar{X}$; коэффициент разветвления по выходу $K_p = 25$
И-102	Две логические схемы 4И с расширением по И (вход Е) и открытым коллекторным выходом (100 мА): $Y = X_1X_2X_3X_4$; $K_p = 200$
И-103	Четыре логические схемы 2И-НЕ: $Y = \overline{X_1X_2}$, $K_p = 25$

Продолжение табл. 3.5

Тип	Назначение элемента
И-104	Две логические схемы 4И-НЕ с расширением по И: $Y = \overline{X_1 X_2 X_3 X_4}$, $K_p = 25$
И-105	Две логические схемы 4ИЛИ с расширением по ИЛИ: $Y = X_1 + X_2 + X_3 + X_4$; $K_p = 25$
И-108	Четыре логические схемы 2ИЛИ-НЕ: $Y = \overline{X_1 + X_2}$, $K_p = 10$
И-109	Логическая схема 2И-2ИЛИ-НЕ
И-110	Два *RS*-триггера с синхронным и асинхронным управлением; хранение информации с $K_p = 24$
И-111	Два универсальных *JK*-триггера; хранение информации, построение счетчиков, делителей частоты, регистров; $K_p = 24$
И-117	Логический дешифратор двоичных четырехразрядных чисел в десятичные $K_p = 10$
И-118	Логический шифратор десятичного кода в двоичный $K_p = 25$
И-119	Цифровой нуль-орган с $K_p = 8$; сравнение двух двоичных двухразрядных чисел
И-120	Сумматор с $K_p = 10$; арифметическое сложение двоичных двухразрядных чисел
Функциональные элементы	
И-201	Выходной согласующий элемент на герконовых реле типа РПГ-6, постоянный ток, 24 В
И-202	То же, 48 В
И-203	То же, 60 В
И-204	То же, 110 В
И-206	Выходной согласующий элемент на переменном токе на реле типа РПГ-6 (~220 В)
И-207	Выходной согласующий низкочастотный элемент на оптронах
И-209	Выходной согласующий высокочастотный элемент на оптронах
Временные элементы	
И-301	Выдержка времени и формирование серий прямоугольных импульсов: выдержка от 0,01 до 10 с; частота импульсов от 0,1 до 100 Гц
Выходные элементы	
И-401	Выходной усилитель на базе РПГ-6; переменный или постоянный ток; 10 мА

Окончание табл. 3.5

Тип	Назначение элемента
И-402	Выходной усилитель на базе РПГ-2; переменный или постоянный ток; 0,5 А
И-403	Выходной усилитель на базе РПГ-5; переменный или постоянный ток; 1 А
И-406	Выходной усилитель на базе тиристорной оптопары; переменный ток до 1 А, напряжение до 220 В

При проектировании системы автоматизации на базе бесконтактных логических элементов исходными данными являются:

– структурные формулы системы автоматизации, полученные с помощью циклограммы или другим способом;

– параметры входных сигналов (датчиков), в первую очередь напряжение;

– параметры нагрузок (реле, контакторов, соленоидов) — напряжение, ток, мощность.

Последовательность действий при проектировании:

1. Выбор элементной базы. Критерии выбора:

– разнообразие типов элементов по назначению;

– стоимость;

– имеющиеся на складе предприятия элементы;

– выбор элементов серии, уже используемых на объекте.

2. Преобразование исходной структурной формулы под выбранную элементную базу, если отсутствуют элементы И или ИЛИ.

3. Выбор устройств согласования на входе и выходе схемы. Проще всего задача решается при использовании элементов серии Логика-И.

4. Построение полной схемы системы автоматизации, включая датчики, устройства согласования и нагрузки с соблюдением всех требований ГОСТ и ЕСКД.

3.3. Использование программируемых контроллеров при реализации логических функций

На рис. 3.3 представлена упрощенная структура ПЛК. Входной модуль выполняет функции рассмотренного ранее $УСО_1$:

– нормализация входного сигнала по уровню;

– гальваническая развязка входов от линий связи с датчиками;
– фильтрация входного сигнала;
– индикация состояния входов, как правило, при помощи светодиодов.

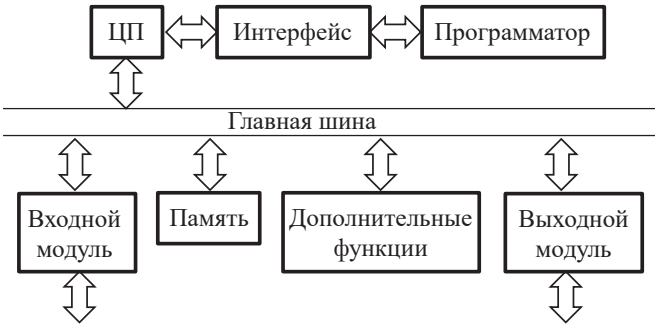


Рис. 3.3. Упрощенная структура программируемого логического контроллера

Выходной модуль выполняет функции рассмотренного ранее $УСО_2$:
– усиление мощности выходного сигнала;
– гальваническая развязка выходов от линий связи с нагрузкой;
– индикация состояния выходов.

Модули содержат обычно число каналов, кратное восьми: 8, 16, 24. Выпускаются смешанные модули вход-выход, например 4+4, 8+8 и более.

Центральный процессор ЦП определяет основные параметры ПЛК:
– число входов и выходов (число переменных);
– число дополнительных функций;
– объем памяти;
– быстродействие.

Он формирует команды на считывание входной информации, следит за последовательностью обработки программы, управляет процессом чтения и записи памяти, загрузки результатов в выходной модуль [9, 10].

Существуют следующие типы процессоров:
– работающие с битом информации;
– работающие с байтами (словами);
– работающие с битами и байтами (самый распространенный тип в технологической автоматике).

Первые два вида процессоров используются для управления простыми агрегатами, современной бытовой техникой, в программиру-

емых (интеллектуальных) реле. Рабочий цикл таких ПЛК включает в себя три этапа:

1. Загрузка в память состояний входов.

2. Последовательная обработка программы с запоминанием результатов в промежуточной или выходной памяти.

3. Передача результатов обработки программы в выходные модули.

Иногда второй и третий этапы объединяются. Работа ПЛК заключается в последовательном повторении рабочих циклов. После выполнения последнего шага программы происходит автоматический возврат к первому шагу. Длительность цикла определяется длиной программы. Длительность первого и третьего этапов составляет примерно по 2 мс. На обработку одного килобайта программы затрачивается время от 5 до 15 мс, т. е. общее время цикла достигает 20 мс.

Память контроллера разделяется на служебную и рабочую [9, 10]. Служебная память используется для хранения программ управления работой ПЛК, она недоступна для пользователя. Рабочая память, наоборот, доступна пользователю и служит для хранения программ и информации пользователя. В рабочей памяти хранят:

– таблицы состояний входов и выходов;

– результаты промежуточных вычислений;

– различные уставки, текущие значения таймеров, счетчиков, регистров.

По местоположению память может быть внешней и внутренней. Внутренняя память обеспечивает автономный режим работы ПЛК. Внешняя память используется для отладки и хранения программ. Внутренняя память, как правило, является памятью EAROM (Electrically Alterable Read Only Memory — электрически изменяемая память только для чтения), располагается в процессорном блоке ПЛК и имеет объем 2, 4, 8, 12, 16 или 32 килобайта.

Существуют следующие типы памяти ПЛК:

1. ПЗУ (постоянное запоминающее устройство), международное обозначение — ROM. Этот вид памяти может быть разделен на три подтипа:

– собственно ПЗУ (ROM), т. е. память, запрограммированная изготовителем контроллера;

– программируемое ПЗУ–ППЗУ или PROM (Programmable ROM), доступно для однократного программирования пользователем, после чего изменение этой памяти невозможно;

– репрограммируемое ПЗУ–РППЗУ. Варианты этой памяти: EPROM (Erasable Programmable Read Only Memory) — память с ультрафи-

олетовым стиранием, и EAPROM (Electronically Alterable Programmable Read-Only Memory) — память с электрическим стиранием.

2. Память данных — память типа ОЗУ-RAM. Позволяет многократно записывать и считывать информацию, но является энергозависимой. При отключении питания информация, хранимая в этой памяти, пропадает.

Дополнительные функции контроллера позволяют расширить его возможности. Способы реализации таких функций зависят от типа ПЛК. Производители могут предлагать модули, реализующие устройства из следующего списка: энергонезависимая память, таймеры, счетчики, регистры, арифметические функции, шифраторы, дешифраторы, преобразователи кодов, схемы сравнения, АЦП и ЦАП.

Программаторы — устройства для ввода программы, стирания и коррекции установленной программы, вывода программы на печать. При проектировании системы используют стационарные программаторы с дисплеем. Для наладочных работ и коррекции уже загруженной программы производители ПЛК предлагают большой спектр переносных компактных программаторов с газоразрядной или матричной индикацией. В настоящее время наладочные работы и всевозможные коррекции программ реализуются с помощью компьютера или ноутбука.

За полувековую историю развития ПЛК было разработано большое количество языков и способов программирования контроллеров. При создании системы управления технологического процесса существует проблема взаимопонимания программиста и технолога. Трудно ожидать от технолога формализованного описания алгоритма, это удел программиста, который, вникнув во все тонкости технологического процесса, создаст программу, в которой будет трудно разобраться технологам. Такая ситуация породила стремление создать технологические языки программирования, доступные инженерам и технологам, и максимально упрощающие процесс программирования [9, 10].

В последние десятилетия появилось несколько технологических языков. Более того, Международной Электротехнической Комиссией (МЭК) разработан стандарт МЭК-61131–3, концентрирующий все передовое в области языков программирования для систем автоматизации технологических процессов. Этот стандарт требует от различных изготовителей ПЛК предлагать команды, являющиеся одинаковыми и по внешнему виду, и по действию.

Стандарт описывает пять языков программирования:

– Sequential Function Chart (SFC) — язык последовательных функциональных блоков;

– Function Block Diagram (FBD) — язык функциональных блоковых диаграмм;

– Ladder Diagrams (LAD) — язык релейных диаграмм;

– Statement List (STL) — язык структурированного текста, язык высокого уровня, похожий на язык *Pascal*;

– Instruction List (IL) — язык инструкций — язык ассемблерного типа с аккумулятором и переходом по метке.

Язык LAD похож на электрические схемы релейной логики, поэтому инженерам, не знающим сложных языков программирования, не составит труда написать на нем программу [10]. Язык FBD напоминает создание схем на логических элементах. В каждом из этих языков есть минусы и плюсы. Поэтому при выборе языка специалисты основываются в основном на личном опыте. Большинство программных комплексов дают возможность конвертировать написанную программу из одного языка в другой. Некоторые задачи могут изящно и просто выглядеть на одном языке и при этом сложно и непонятно на другом.

Наибольшее распространение в настоящее время получили языки LAD, STL и FBD. Большинство производителей ПЛК имеют собственные наработки в области инструментального программного обеспечения. Например, такие как ZelioSoft от Schneider Electric или Step 7 от Siemens. Они не желают отказываться от огромных наработок, уже имеющихся у них в этой области, и могут себе это позволить в силу широкого распространения их продукции на мировом рынке и, в частности, в уральском регионе. Нужно отметить, что сама логика появления языков стандарта МЭК 61131–3 и фирменных языков делает их очень похожими друг на друга.

В программных средах разработки имеется большой набор готовых библиотек элементов, подпрограммы стандартных процедур и шаблонов. Также среда разработки обязательно должна включать в себя программный эмулятор контроллера, позволяющий всесторонне проверить работоспособность программы перед ее переносом на реальный контроллер. Для некоторых серий контроллеров, например, для серии Simatic S7-200 фирмы Siemens, эмулятор контроллера поставляется отдельно от среды разработки.

При написании программ используется набор готовых библиотек-элементов, который фирма-изготовитель предоставляет вместе с оборудованием. Далее будут рассматриваться преимущественно документация и пакеты программ для проектирования систем автоматизации фирмы Siemens. Выбранный для проектируемой системы ПЛК в первую очередь должен иметь необходимое количество входных и выходных

каналов (число входных и выходных переменных). Каждой переменной присваивается адрес, в соответствии с которым в ОЗУ имеется ячейка, хранящая состояние этой переменной. Далее с учетом этих адресов выполняется подключение датчиков и нагрузок к клеммам модуля CPU (ЦП) и модулей расширения цифровых сигналов EM (входных и выходных). У программируемых контроллеров серии Simatic S7-200 (1200) процессорный модуль позволяет подключить определенное количество входных и выходных сигналов, т. к. он имеет устройства согласования. Адрес входа контроллера имеет следующий вид: I0.1, где *I* — символ, означающий область дискретных входов контроллера, 0 — номер байта (все входы разбиты на группы по байтам), 1 — номер бита в выбранном байте (от 0 до 7). Самый первый вход контроллера имеет адрес I0.0. Адрес дискретного выхода контроллера имеет следующий вид: Q0.2, назначение символов аналогично входным адресам. Адреса входов и выходов подписаны на самом процессорном модуле, на модулях расширения подписаны только адреса отдельных битов, т. к. номер байта в адресе зависит от порядка установки модулей расширения (они подключаются последовательно к модулю CPU и далее друг к другу). CPU-224 содержит 14 входов, они имеют адреса I0.0... I0.7, I1.0... I1.5. Адреса входов модуля расширения, подключенного к этому процессорному модулю, будут начинаться с I2.0.

На рис. 3.4 показан вариант адресации входных и выходных сигналов процессорного модуля CPU-222 (8 входов и 6 выходов) и модулей расширения (модуль дискретных входов и комбинированный модуль — входы — выходы). Подключение входных и выходных каналов осуществляется через клеммные контакты, подключение модулей расширения — через разъемы.

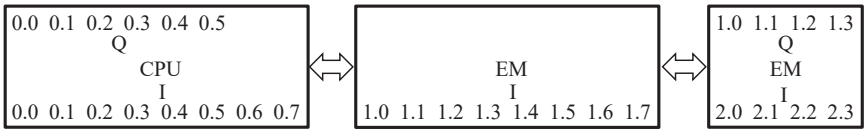


Рис. 3.4. Адреса входов и выходов контроллера с модулями расширения

После опроса всех входных сигналов и загрузки их состояния в ОЗУ начинается обработка программы. Для этого используется стековая память, позволяющая хранить промежуточные результаты выполнения программы. Особенность организации стековой памяти — последовательная загрузка и выгрузка информации через вершину стека (загрузка прямых и инверсных значений входных сигналов, результатов

логического сложения и умножения). Стековая память имеет 9 уровней. При разработке программы нельзя допускать переполнения стека, поскольку в этом случае теряется информация последнего (девятого) уровня стека.

В табл. 3.6 представлены основные команды, используемые при составлении программы на языке STL. При применения этого языка необходимо следить за наполнением стека и не допускать потери информации.

Вернемся к рассмотренной в главе 2 циклограмме (рис. 2.11). После проведения всех проверок окончательная формула системы управления имеет вид:

$$X = d + xa + x\overline{p_1} + \overline{b}\overline{c}p_2;$$
$$Y = \overline{b}ep_3;$$
$$P_1 = a + p_1\overline{c};$$
$$P_2 = b + p_2\overline{c};$$
$$P_3 = b + p_3e.$$

Таблица 3.6

Основные команды языка STL

Двоичные операторы	
LD (LDN)	загрузка битового значения (его инверсии) в вершину стека
A (AN)	логическое И над вершиной стека и переменной (ее инверсией)
O (ON)	логическое ИЛИ над вершиной стека и переменной (ее инверсией)
=	копирование результата из вершины стека в переменную (выходную или промежуточную)
S (R)	установка (сброс) битовой переменной (выходной или промежуточной)
ALD	логическое И между двумя верхними уровнями стека
OLD	логическое ИЛИ между двумя верхними уровнями стека
NOT	инвертирование вершины стека
EU	определение переднего фронта для вершины стека
ED	определение заднего фронта для вершины стека
Временные функции	
TON	таймер с задержкой включения
TOF	таймер с задержкой отключения

Составим программу с использованием только базовых команд (И, ИЛИ, НЕ). Присвоим адреса входным и выходным переменным:

a — I0.0	x — Q0.0
b — I0.1	y — Q0.1
c — I0.2	p_1 — Q0.2
d — I0.3	p_2 — Q0.3
e — I0.4	p_3 — Q0.4

Для промежуточных переменных используем незадействованные адреса выходных переменных. Программа для каждой выходной и промежуточной переменной начинается словом Network с номером и записью реализуемой формулы.

Network 1 // $P_1 = a + p_1\overline{c}$
LD I0.0
LD Q0.2
AN I0.2
OLD
= Q0.2
Network 2 // $P_2 = b + p_2\overline{c}$
LD I0.1
LD Q0.3
AN I0.2
OLD
= Q0.3
Network 3 // $P_3 = b + p_3$e
LD I0.1
LD Q0.4
A I0.4
OLD
= Q0.4
Network 4 // $X = d + xa + x\overline{p_1} + \overline{b}\overline{c}p_2$
LD I0.3
LD Q0.0
A I0.0
LD Q0.0
AN Q0.2
LDN I0.1
AN I0.2
A Q0.3
OLD

OLD
OLD
= Q0.0
Network 5 // $Y = \overline{b}ep_3$
LDN I0.1
A I0.4
A Q0.4
= Q0.1
MEND

Команда MEND возвращает программу в начало, и так повторяется циклически. Составленная программа содержит 31 шаг. Программу можно сократить, если использовать триггерные команды S и R. Структурную формулу для выходной переменной X нужно преобразовать:

$$X = d + xa + x\overline{p_1} + \overline{b}\overline{c}p_2 = d + \overline{\overline{a}p_1}x + \overline{b}\overline{c}p_2.$$

d включает триггер, $\overline{a}p_1$ — сбрасывает.
Network 1 // $P_1 = a + p_1\overline{c}$
LD I0.0
S Q0.2
LD I0.2
R Q0.2
Network 2 // $P_2 = b + p_2\overline{c}$
LD I0.1
S Q0.3
LD I0.2
R Q0.3
Network 3 // $P_3 = b + p_3$e
LD I0.1
S Q0.4
LDN I0.4
R Q0.4
Network 4 // $X = d + \overline{\overline{a}p_1}x + \overline{b}\overline{c}p_2$
LD I0.3
S Q0.0
LDN I0.0
A Q0.2
R Q0.0
LDN I0.1
AN I0.2

```
A Q0.3
= Q0.0
Network 5 // Y = b̄ep3
LDN I0.1
A I0.4
A Q0.4
= Q0.1
MEND
```

Длина программы — 25 шагов.

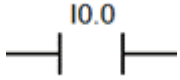
И наконец, решим эту же задачу с использованием команд EU (нарастающий фронт) и ED (спадающий фронт). Эти команды используются в сочетании с командами S и R. При этом пропадает необходимость вводить промежуточные переменные.

```
Network 1 // X
LD I0.3
S Q0.0
LD I0.0
ED
R Q0.0
LD I0.1
ED
S Q0.0
LD I0.2
EU
R Q0.0
Network 2 // Y
LD I0.1
ED
S Q0.1
LD I0.4
ED
R Q0.1
MEND
```

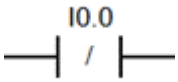
Программа содержит 17 шагов.

Мы подробно рассмотрели систему команд на языке STL. Переходим к рассмотрению основных команд на языках LAD и FBD.

Язык LAD.

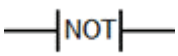


Замыкающий (нормально открытый) контакт замкнут, если значение назначенного бита (I0.0) равно 1.

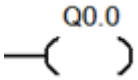


Размыкающий (нормально замкнутый) контакт замкнут, если значение назначенного бита (I0.0) равно 0.

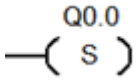
Контакты, соединенные последовательно, образуют логические соединения типа И (AND). Контакты, соединенные параллельно, образуют логические соединения типа ИЛИ (OR).



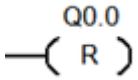
Логическое отрицание (NOT) инвертирует входной сигнал. Если на контакт NOT поступает 1, то на его выходе будет 0. Если на контакт NOT поступает 0, то на выходе будет 1.



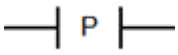
Выходная катушка. Если на этот элемент подается сигнал 1, то выходной бит (Q0.0) устанавливается в 1. Если подан 0, то выходной бит устанавливается в 0.



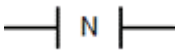
Установка (включение) бита. Если блок S активирован, то значение выхода (Q0.0) устанавливается в 1. Если S не активирован, то выход (Q0.0) не изменяется.



Сброс (выключение) бита. Если блок R активирован, то выход (Q0.0) устанавливается в 0. Если R не активирован, то выход (Q0.0) не изменяется.



Определение нарастающего фронта. Сигнал на выходе контакта P равен единице только в момент переключения входного сигнала из 0 в 1.

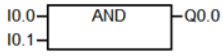


Определение спадающего фронта. Сигнал на выходе контакта N равен единице только в момент переключения входного сигнала из 1 в 0.

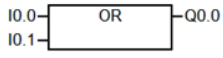
Элементы P и N располагаются после контакта или группы контактов для обнаружения фронта сигнала.

Язык FBD.

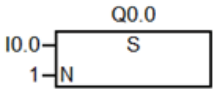
При программировании на языке FBD программа записывается в виде сети блоков И (AND, &), ИЛИ (OR, >=1) и исключающего ИЛИ (XOR), в которых можно задать значения битов для входов и выходов блока. Блоки соединяются между собой в соответствии с алгебраическим выражением, полученным на предыдущем этапе проектирования системы автоматизации.



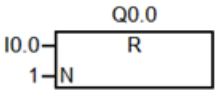
Чтобы выход блока И (Q0.0) принял значение 1, на всех его входах должны быть сигналы 1.



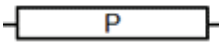
Чтобы выход блока ИЛИ (Q0.0) принял значение 1, на любом его входе должна быть 1.



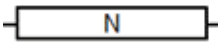
Установка бита. Если блок S активирован, то значение выхода (Q0.0) устанавливается в 1. Если S не активирован, то выход (Q0.0) не изменяется.



Сброс (выключение) бита. Если блок R активирован, то выход (Q0.0) устанавливается в 0. Если R не активирован, то выход (Q0.0) не изменяется.

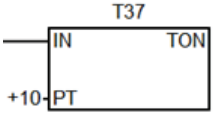


Определение нарастающего фронта. Сигнал на выходе блока P равен единице только в момент переключения входного сигнала из 0 в 1.

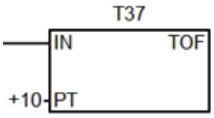


Определение спадающего фронта. Сигнал на выходе блока N равен единице только в момент переключения входного сигнала из 1 в 0. Все команды обнаружения фронта используют бит памяти для хранения предыдущего состояния подлежащего контролю входного сигнала. Фронт обнаруживается путем сравнения текущего состояния сигнала со значением этого бита. Если эти состояния указывают на изменение сигнала на входе в требуемом направлении, то на выходе соответствующего элемента возникает сигнал единицы, иначе выход устанавливается в ноль.

Для реализации временных задержек между событиями, например между поступлением сигнала в контроллер и включением его выхода, используются таймеры. Таймеры различаются временным разрешением — минимально возможной задержкой, реализуемой с их помощью. По величине разрешения они разделены на три группы: 1 мс, 10 мс и 100 мс. К первой группе относятся таймеры с адресами T32 и T96; ко второй — T33...T36 и T97...T100; к третьей — T37...T63 и T101...T255. Ниже рассмотрены таймерные команды.

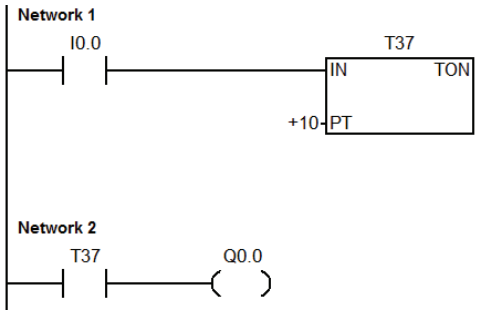


Таймер с задержкой включения (TON) отсчитывает время, когда включен разрешающий вход (IN). Когда текущее значение (T37) становится больше или равно предустановленному (PT), бит таймера устанавливается в 1. Таймер продолжает счет после достижения предустановленного значения, но останавливается при достижении максимального значения, равного 32767. Текущее значение таймера и его бит сбрасываются в 0, когда выключен разрешающий вход (IN).

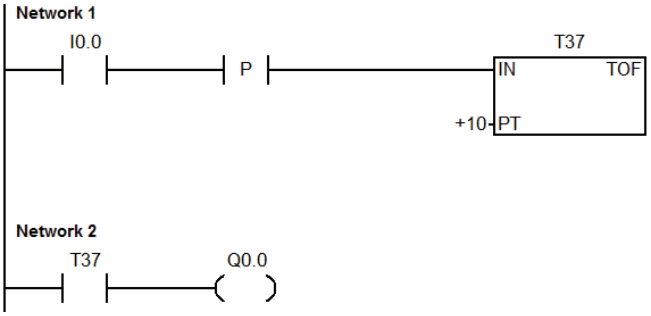


Таймер с задержкой отключения (TOF) используется для задержки выключения выхода на заданный интервал времени после выключения входа IN. При включении разрешающего входа бит таймера включается, а его текущее значение устанавливается в 0. Когда разрешающий вход выключается, таймер начинает отсчет времени, который продолжается пока истекшее время не достигнет предустановленного. Когда заданное время достигнуто, бит таймера выключается, и отсчет текущего значения прекращается. Команда TOF должна обнаружить переход от включенного состояния входа к выключенному, чтобы начать отсчет времени.

Пример использования таймера TON показан ниже. В представленной программе выход контроллера включается через одну секунду после появления сигнала на входе. Таймер T37 имеет разрешение 100 мс, поэтому его предустановленное значение задано равным 10.



Для включения выхода на заданное время при появлении сигнала на входе контроллера может быть использован таймер TOF с элементом обнаружения фронта сигнала. Короткий импульс на входе таймера TOF запускает отсчет выдержки времени, в течение которого таймер будет включен.



Контроллер работает по следующему циклу:

1. Считывание состояния входов и загрузка их значений в память ПЛК.
2. Выполнение программы.
3. Обработка коммуникационных запросов. Обрабатываются все сообщения, принятые через коммуникационные порты.
4. Выполнение самодиагностики (проверка программ в ПЗУ, памяти программ и состояния модулей расширения).
5. Запись управляющих сигналов на выходы.

Рассмотрим основные технические характеристики ПЛК серии Simatic S7-200 (табл. 3.7). Хотя эта серия контроллеров уже снята с производства, она по-прежнему эксплуатируется, а предложенные языки, методы программирования и большой объем публикаций позволяют использовать эти наработки для получения навыков проектирования систем автоматизации с использованием ПЛК новых продвинутых серий.

Если встроенных в процессорный модуль «входов — выходов» недостаточно, то для подключения дополнительных логических сигналов используются модули «ввода — вывода» дискретных сигналов EM-221, EM-222, EM-223. Модули предназначены для преобразования входных дискретных сигналов контроллера в его внутренние логические сигналы (модули ввода) и обратного преобразования (модули вывода). Основные характеристики модулей расширения приведены в табл. 3.8 и 3.9.

Таблица 3.7

Характеристики процессорных модулей ПЛК серии Simatic S7-200

Характеристики	CPU-221	CPU-222	CPU-224	CPU-226
Встроенные входы/выходы	6 / 4	8 / 6	14 / 10	24 / 16
Макс. модулей расширения	—	2	7	7
Макс. каналов ввода вывода	10	78	168	168
Аналоговые входы/выходы	—	8 / 4	28 / 14	28 / 14
Память программ/данных, кБ	4 / 2	4 /2	8 / 12	16 / 24
Время выполнения логической операции, мкс	0,22	0,22	0,22	0,22
Количество флагов / счетчиков / таймеров	256 / 256 / 256	256 / 256 / 256	256 / 256 / 256	256 / 256 / 256
Высокоскоростные счетчики	4×30 кГц	4×30 кГц	6×30 кГц	6×30 кГц
Часы реального времени	опция	опция	встроен.	встроен.
Импульсные выходы	2×20 кГц	2×20 кГц	2×20 кГц	2×20 кГц

Окончание табл. 3.7

Характеристики	CPU-221	CPU-222	CPU-224	CPU-226
Коммуникационный порт	1×RS-485	1×RS-485	1×RS-485	2×RS-485
Встроенный потенциометр	1	1	2	2

Таблица 3.8

Характеристики цифровых (дискретных) модулей расширения

Характеристики	EM-221	EM-222	EM-223
Количество дискретных входов	8	—	4 / 8 / 16
Количество дискретных выходов	—	8	4 / 8 / 16
Тип входов	оптопара	—	оптопара
Тип выходов	—	реле / транзистор	реле / транзистор
Макс. задержка по входу, мс	4,5	—	4,5
Входное напряжение, В	+24 В	–	+24 В
Выходное напряжение, В	—	до 250 В ~ тока до 30 В = тока (реле) +24 В (транзистор)	до 250 В ~ тока до 30 В = тока (реле) +24 В (транзистор)

Для процессорных модулей CPU-222, CPU-224 и CPU-226 существуют модули, обеспечивающие связь контроллера по сети Profibus DP (EM-277), подключение термопар (EM-231 Thermocouple) и терморезисторов (EM-231 RTD), модуль для связи по сети Ethernet (CP-243–2). Их подробные характеристики можно найти в описании производителя.

Таблица 3.9

Характеристики аналоговых модулей расширения

Характеристики	EM-231	EM-232	EM-235
Количество аналоговых входов	4	—	4
Количество аналоговых выходов	—	2	1

Окончание табл. 3.9

Характеристики	ЕМ-231	ЕМ-232	ЕМ-235
Входное напряжение, В	0...10; 0...5; ±5; ±2,5	—	0...10; 0...5; 0...1; 0...0,5; 0...0,1; 0...0,05; ±10; ±5; ±2,5; ±1; ±0,5; ±0,25; ±0,1; ±0,05; ±0,025
Разрешающая способность, бит	12	12	12
Макс. задержка по входу / выходу, мс	1,5	0,1	1,5
Выходное напряжение, В	—	±10	—

Итак, мы рассмотрели языки программирования и характеристики контроллеров, предлагаемых фирмой Siemens, а также методы программирования логических контроллеров. Далее рассмотрим последовательность синтеза управляющего устройства при использовании ПЛК:

1. Выбрать программируемый контроллер. В качестве критериев выбора стоит ориентироваться на следующие показатели: наличие необходимого количества входов и выходов, требуемый набор дополнительных функций, стоимость и надежность.

2. Выбрать адреса входных, выходных и промежуточных переменных.

3. Составить программу на языке, используемом данным контроллером.

4. Выполнить отладку программы с использованием эмулятора контроллера.

5. Выбрать все необходимые модули согласования для входных и выходных сигналов.

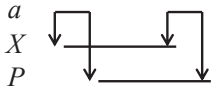


Рис. 3.5. Циклограмма, не имеющая определенного решения

6. Разработать полную схему соединений всех элементов системы, включая датчики, устройства согласования и нагрузки, с соблюдением всех требований ГОСТ и ЕСКД.

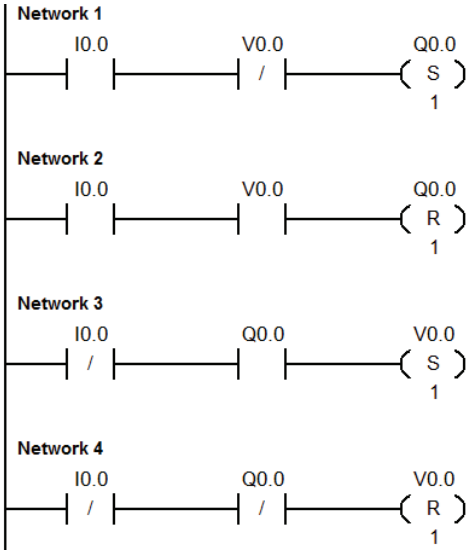
В заключение рассмотрим программу, реализующую циклограмму, не имеющую определенного решения (рис. 3.5).

Структурная формула схемы имеет следующий вид:

$$X:\ S = a\overline{p}\ ,\ R = ap\ ;$$

$$P:\ S = \overline{a}x\ ,\ R = \overline{ax}\ .$$

Программа, реализующая представленные алгоритмы, показана ниже.



Для решения задач автоматизации целого ряда относительно несложных локальных систем (например, управления уличными светофорами, освещения подъездов жилых зданий и тому подобного) ведущие фирмы-производители выпускают компактные ПЛК с относительно небольшим количеством входных и выходных сигналов и поддержкой простейших функций [11]. Такие устройства получили название «интеллектуальные реле». Ниже рассмотрим следующие серии контроллеров:

– LOGO! фирмы Siemens;
– Zelio фирмы Schneider Electric;
– Zen фирмы Omron.

Внешне эти устройства выглядят почти одинаково, но отличаются цветом корпуса. Ниже приведены основные технические характеристики перечисленных интеллектуальных реле.

Серия LOGO! включает в себя:

– LOGO! Basic и LOGO! Pure — логические модули;
– LOGO! Contact — модули бесшумной коммутации трехфазных цепей переменного тока;
– LOGO! Power — блоки питания;

– модули ввода — вывода дискретных сигналов DM8 / DM16;
– модули ввода — вывода аналоговых сигналов AM2;
– коммуникационные модули CM.

Для программирования ПЛК серии LOGO! используется пакет LOGO! Soft Comfort. Данная серия контроллеров предназначена для управления технологическим оборудованием с относительно несложными алгоритмами функционирования (насосы, вентиляторы, компрессоры, прессы, эскалаторы, подъемники и конвейеры, системы отопления и освещения, системы управления дорожным движением). Преимущества контроллеров серии LOGO! заключаются в следующем:

– экономия затрат на готовое изделие (низкая цена по сравнению с ПЛК старших серий);
– экономия времени на разработку благодаря простой системе программирования;
– возможность использования в условиях повышенной влажности.

Модули серии LOGO! выпускаются на напряжение питания 12/24 В постоянного тока (DC) или 115/230 В переменного тока (AC). Выходы модулей могут быть транзисторными (TO – напряжение 24 В постоянного тока силой до 0,3 А) или релейными (RO – до 240 В/3 А). Контроллеры могут иметь дискретные входы (DI), аналоговые входы (AI) и универсальные входы (UI). В табл. 3.10 представлены основные характеристики ПЛК серии LOGO!

Таблица 3.10

Характеристики контроллеров LOGO!

Обозначение	Напряжение питания, В	Входы	Выходы
серия LOGO! Basic (со встроенной клавиатурой и дисплеем)			
LOGO!24	DC 24	6DI + 2UI	4TO
LOGO!24RC	AC 24	8DI	4RO
LOGO!12/24RC	DC 12/24	6DI + 2UI	4RO
LOGO!230RC	AC/DC 115/230	8DI	4RO
серия LOGO! Pure (без встроенной клавиатурой и дисплея)			
LOGO!24o	DC 24	6DI + 2UI	4TO
LOGO!24RCo	AC 24	8DI	4RO
LOGO!12/24RCo	DC 12/24	6DI + 2UI	4RO
LOGO!230RCo	AC/DC 115/230	8DI	4RO

Окончание табл. 3.10

Обозначение	Напряжение питания, В	Входы	Выходы
LOGO! Expansion Modules (модули расширения)			
LOGO! DM8 24	DC 24	4DI	4TO
LOGO! DM8 24R	AC 24	4DI	4RO
LOGO! DM8 12/24R	DC 12/24	4DI	4RO
LOGO! DM8 230R	AC/DC 115/230	4DI	4RO
LOGO! DM16 24	DC 24	8DI	8TO
LOGO! DM16 24R	DC 24	8DI	8RO
LOGO! DM16 230R	AC/DC 115/230	8DI	8RO
LOGO! AM2	DC 12/24	2AI	—
LOGO! AM2 PT100	DC 12/24	2Pt100	—
LOGO! AM2 AQ	DC 12/24	—	2AQ
LOGO! CM AS I	AC/DC 24	4AS	4AS
LOGO! CM LAN	AC/DC 24	16DI+8AI	12DO
LOGO! CM EIB/KNX	AC/DC 24	16DI+8AI	12DO

Логические модули LOGO! Contact характеризуются следующими показателями:

– осуществляют бесшумную коммутацию трехфазных цепей переменного тока;

– напряжение сети до 400 В, активная нагрузка до 20 А;

– осуществляют управление асинхронными короткозамкнутыми двигателями мощностью до 4 кВт.

Фирма Schneider Electric выпускает аналогичные контроллеры — серия Zelio Logic. В табл. 3.11 приведены основные характеристики ПЛК данной серии.

Таблица 3.11

Характеристики контроллеров Zelio

Параметр	Напряжение питания, В	
	DC 24	AC 100...240
Количество дискретных входов	8 / 12	6 / 8 / 12
Количество дискретных выходов	4 / 8	4 / 4 / 8
Модуль с дисплеем и часами	SR2Bxx1B	SR2Bxx1FU
Язык программирования	FBD или Ladder	FBD или Ladder

Окончание табл. 3.11

Параметр	Напряжение питания, В	
	DC 24	AC 100...240
Модуль с дисплеем, без часов	—	SR2Axx1FU
Язык программирования	—	только Ladder
Модуль без дисплея, с часами	SR2Exx1B	SR2Exx1FU
Язык программирования	FBD или Ladder	FBD или Ladder
Модуль без дисплея и часов	—	SR2Dxx1FU
Язык программирования	—	только Ladder
ПО для программирования	ZelioSoft 2 SFT01	ZelioSoft 2 SFT01

Как видно, принципы построения, функциональные возможности, исполнение модулей LOGO! и Zelio очень близки. В равной степени все сказанное относится и к модулям Zen фирмы Omron.

Контрольные вопросы к главе 3

1. Какие способы аппаратной реализации логических функций вам известны?
2. Опишите наиболее общую структуру системы управления технологической автоматики.
3. Какие функции выполняют устройства сопряжения системы управления технологической автоматики с объектом управления?
4. В чем заключается основное отличие взвешенного кода от невзвешенного?
5. Каковы основные правила, применяемые при построении релейно-контакторных схем на основе формулы логической функции?
6. Перечислите основные правила, применяемые при реализации логических функций на базе бесконтактных логических элементов.
7. Каковы преимущества программируемых логических контроллеров перед релейно-контакторными схемами и системами автоматизации, выполненными на базе «жесткой» логики?
8. Перечислите основные языки программирования промышленных контроллеров.
9. Какие основные этапы разработки управляющего логического устройства вам известны?

Глава 4.

Надежность систем автоматики

Расчет надежности систем автоматики необходим для решения следующих задач:

– разработки мер по повышению надежности системы при ее проектировании;

– определение технико-экономического эффекта при замене одной системы другой;

– определения периодичности техобслуживания.

Надежность — свойство системы сохранять работоспособность в течение определенного промежутка времени при заданных условиях эксплуатации [12].

Необходимость решения проблемы надежности возникли относительно недавно в связи с:

– усложнением систем управления и автоматики;

– интенсификации работы систем.

Математические основы теории надежности — теория вероятности и матстатистика.

4.1. Основные понятия и термины теории надежности [12]

Работоспособность — свойство изделия выполнять заданные функции с параметрами, установленными в технической документации.

Долговечность — свойство изделия сохранять работоспособность до предельного состояния, оговоренного в технической документации при условии правильного обслуживания.

Предельное состояние — это состояние изделия, при котором его дальнейшая эксплуатация становится нецелесообразной или недопустимой.

Срок службы — календарная продолжительность эксплуатации изделия до момента наступления предельного состояния.

Наработка — продолжительность (в часах, циклах) или объем работы (га, км).

Отказ — случайное событие, в результате которого изделие (или его элемент) перестает выполнять свои функции.

Типы отказов:

– зависимые (следствие отказа другого элемента);
– независимые (отказ элемента сам по себе);
– постепенные;
– внезапные;
– окончательные (ремонт изделия или замена);
– перемежающиеся (многократно возникающие и самоустраняющиеся).

Для числовой оценки надежности системы используют следующие показатели [13]: вероятность безотказной работы — $p(t)$, интенсивность отказов — $\lambda(t)$, вероятность отказа — $q(t)$, среднее время безотказной работы — $T_{ср}$, частота отказов — $a(t)$.

Вероятность безотказной работы — вероятность того, что в заданном интервале времени при соблюдении паспортных условий эксплуатации не произойдет ни одного отказа — $p(t)$. $P(t)$ является монотонно убывающей функцией. Очевидно, что $0 \leqslant p(t) \leqslant 1$, $p(0) = 1$, $p(\infty) = 0$.

Вероятность отказа — вероятность того, что в заданном интервале времени произойдет хотя бы один отказ, — $q(t)$. Очевидно, что $q(t) = 1 - p(t)$.

Интенсивность отказов — вероятность отказов неремонтируемого изделия в единицу времени при условии, что до этого момента времени отказ не возник — $\lambda(t)$. Значение $\lambda(t)$ определяется производителями изделия как число отказов в единицу времени, отнесенное к среднему числу элементов, исправно работающих в данный отрезок времени, и обычно выражается в % на 1000 ч либо на один час работы.

Среднее время безотказной работы (средняя наработка до отказа) — $T_{ср}$ — математическое ожидание времени безотказной работы.

$$T_{ср} = \frac{\sum_{i=1}^{n} t_i}{n},$$

где t_i — время исправной работы i-го элемента, n — число элементов.

Частота отказов (плотность распределения времени безотказной работы) — $a(t)$.

Графическая интерпретация зависимости $\lambda(t)$ показана на рис. 4.1. Опыт эксплуатации технических систем показывает, что график функции $\lambda(t)$ можно разделить на три периода [14].

Первый период характеризуется высокой интенсивностью отказов, которая уменьшается с течением времени $(0-t_1)$. На этом участке выявляются грубые дефекты, это период приработки, его длительность составляет сотни часов (сроки гарантии).

Второй период (t_1-t_2) — это участок нормальной эксплуатации, на котором интенсивность отказов имеет постоянное значение. Его длительность может составлять тысячи и десятки тысяч часов.

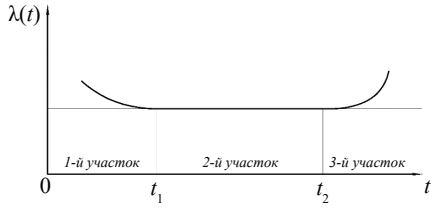


Рис. 4.1. Графическая интерпретация зависимости $\lambda(t)$

На третьем участке $(t_2-\infty)$ из-за старения элементов интенсивность отказов начинает возрастать. Время (t_2) — начало периода износа, при достижении которого система должна сниматься с эксплуатации или выводиться на капитальный ремонт.

Рассмотрим зависимости между основными показателями надежности.

Изготовители элементов определяют интенсивность отказов каждого вида изделия по результатам испытания партии готовой продукции. Партия изделий, включающая в себя n элементов (чем больше, тем результат будет точнее), ставится на испытания с постоянной паспортной нагрузкой. Элементы постепенно выходят из строя вплоть до последнего (рис. 4.2).

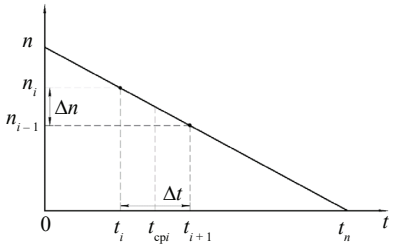


Рис. 4.2. Зависимость числа отказов от времени

Интенсивность отказов $\lambda(t)$ определяется как число отказов Δn_i в единицу времени Δt, отнесенное к среднему числу элементов, исправно работающих в данный отрезок времени.

$$\lambda(t) = \frac{\Delta n_i}{n_x \Delta t},$$

где n_i, n_{i+1} — число исправных работающих элементов в начале (t_i) и в конце (t_{i+1}) интервала времени Δt; Δt — интервал времени; Δn_i — число отказавших элементов за интервал времени Δt; n_x — среднее число исправно работающих элементов в интервале времени Δt.

$$n_x = \frac{n_i + n_{i+1}}{2} = \frac{nP_i + nP_{i+1}}{2} = \frac{n\left(P_i + P_{i+1}\right)}{2};$$

$$\Delta n_i = n_{i+1} - n_i = nP_{i+1} - nP_i = -n\Delta P;$$

$$\lambda(t) = \frac{2n\Delta P}{n\left(P_i + P_{i+1}\right)\Delta t};$$

$$\Delta t \to 0,\ P_i \to P_{i+1} \to P.$$

Отсюда $\lambda = -\dfrac{2dP}{2\Delta P dt} = -\dfrac{P'}{P}$.

$$\int_0^t \lambda \cdot dt = -\int_0^t \frac{dP}{P} = -\ln P \Big|_0^t = -\left[\ln P(t) - \ln P(0)\right];$$

$$\ln P(t) = -\int_0^t \lambda dt;$$

отсюда

$$p(t) = \exp\left(-\int_0^t \lambda(t) dt\right). \tag{4.1}$$

Для периода нормальной эксплуатации (интервал $t_1 - t_2$, рис. 4.1). $\lambda = \text{const}$. Отсюда

$$p(t) = e^{-\lambda t}. \tag{4.2}$$

Среднее время безотказной работы $T_{\text{ср}}$ на практике определяется в результате испытания партии изделий по формуле:

$$T_{\text{ср}} = \frac{\sum_{i=1}^{m} \Delta n_i \cdot t_{\text{ср}i}}{n},$$

где $t_{\text{ср}i}$ — продолжительность исправной работы Δn_i элементов, вышедших из строя в интервале времени Δt (рис. 4.2); m — число интервалов времени Δt за время испытаний (от 0 до t_n, t_n — время выхода из строя последнего элемента испытуемой партии, рис. 4.2).

$$m = \frac{t_n}{\Delta t};\ t_{cpi} = \frac{t_{i+1} + t_i}{2}$$

при $\Delta t \to 0,\ m \to \infty$.

$$T_{\text{ср}} = \frac{\int_0^{\infty} -t \Delta n dp}{n} = -\int_0^{\infty} t dP;$$ (табличный интеграл: $\int udv = uv - \int vdu$).

$$T_{cp} = -\left[t \cdot p \Big|_0^{\infty} - \int_0^{\infty} p \cdot dt \right]$$

$t \cdot p(\infty) = 0$, т. к. p становится равной нулю через конечный интервал времени.

$t \cdot p(0) = 0 \cdot 1 = 0$. Следовательно, $T_{\text{ср}} = \int_0^{\infty} p dt = \int_0^{\infty} e^{-\int_0^{\infty} \lambda dt} dt.$

Поскольку в период нормальной эксплуатации $\lambda(t) = \text{const}$, то

$$T_{\text{ср}} = \int_0^{\infty} e^{-\lambda t} dt = -\frac{1}{\lambda} \int_0^{\infty} e^{-\lambda t} d(\lambda t) = -\frac{1}{\lambda} \left| e^{-\lambda t} \right[_0^{\infty} = -\frac{1}{\lambda} [0 - 1] = \frac{1}{\lambda}.$$

Итак,

$$T_{\text{ср}} = \frac{1}{\lambda}. \qquad (4.3)$$

Частота отказов $a(t)$ является производной от вероятности отказов — $q'(t)$. $a(t) = q'(t) = -p'(t)$ (т. к. $q = 1 - p$.

Отсюда

$$a(t) = -p'(t) = -\left(e^{-\lambda t}\right)' = \lambda e^{-\lambda t} = \lambda p(t)$$

$$a(t) = \lambda p(t) \tag{4.4}$$

$$T_{\text{ср}} = \frac{P(t)}{a(t)}. \tag{4.5}$$

Формулы, связывающие $a(t)$ с другими показателями надежности, представлены для периода нормальной эксплуатации, когда $\lambda = \text{const}$ (рис. 4.1).

Как уже отмечалось, изготовитель по результатам испытаний партии элементов указывает интенсивность отказов λ (1/час), либо время безотказной работы $T_{\text{ср}}$ (час), их значения стандартизированы и выбираются из ряда:

для λ — 10^{-4}; $5 \cdot 10^{-5}$; $3 \cdot 10^{-5}$; $2 \cdot 10^{-5}$; 10^{-5}; $5 \cdot 10^{-6}$ ... 10^{-6} ... 10^{-7} ... 10^{-10}.

Для $T_{\text{ср}}$ — 200, 500, 750, 1000, 1500, 2000, 3000, 4000, 5000, 7500, 10000, 15000, 20000 и более.

При проектировании систем автоматики увеличение количества элементов при их последовательном соединении существенно уменьшает надежность [12]. Использование резервных элементов (параллельное соединение) повышает стоимость системы. Отсюда вытекает необходимость обоснованного выбора надлежащего уровня надежности. Кроме того, нужно учитывать ремонтопригодность (удобство ремонта) системы.

Таким образом, надежность системы обусловлена сочетанием следующих показателей: безотказность, долговечность и ремонтопригодность.

Ремонтопригодность — приспособленность системы к предупреждению, обнаружению и ликвидации отказов. Для безотказности характерными являются закономерности возникновения отказов, для ремонтопригодности — закономерности их устранения.

Системы могут быть ремонтируемыми и неремонтируемыми. При возникновении отказа ремонтируемая система может работать после ремонта, тогда как неремонтируемая система ремонту не подлежит по экономическим или техническим соображениям.

Системы автоматики, являясь ремонтируемыми, имеют ресурс, который определяется не старением и износом, а снижением эффективности и нецелесообразности дальнейшей эксплуатации.

Поэтому надежность систем автоматики достаточно полно определяется безотказностью и ремонтопригодностью.

Для систем автоматики полагают, что вероятность появления в какой-то момент времени более одного отказа пренебрежимо мала, а вероятность появления некоторого числа отказов в интервале времени

$\Delta(t)$ зависит только от длительности интервала и не зависит от *t*. Отказы носят случайный характер и не зависят друг от друга [13].

Из рассмотренных показателей для неремонтируемых систем главным показателем является безотказность, т. к. первый же отказ выводит систему из строя.

Для систем технологической автоматики расчеты надежности необходимы для решения следующих задач:

– разработки мер по повышению надежности системы при ее проектировании;

– определения технико-экономического эффекта при замене одной системы другой;

– определения периодичности техобслуживания.

4.2. Аппаратурная и структурная надежности

Аппаратурная надежность — это надежность отдельно взятого элемента, определяемая изготовителем (λ или $T_{ср}$).

Структурная надежность — результирующая надежность системы при заданной ее структуре и известных значениях надежности всех входящих в нее элементов. При проектировании системы ее структура минимизируется, поэтому выход из строя любого элемента приводит к отказу всей системы. Следовательно, необходимо определить показатели надежности системы, которая включает в себя последовательно включенные все элементы, составляющие систему.

$P_{\Sigma} = P_1 \cdot P_2 \cdot \ldots P_n$; т. к. при $\lambda = \text{const}$ $p = e^{-\lambda t}$, $P_{\Sigma} = e^{\lambda_1 t} \cdot e^{\lambda_2 t} \ldots e^{\lambda_n t} = e^{-(\lambda_1+\lambda_2+\ldots+\lambda_n)t} = e^{-\lambda_{\Sigma} t}$.

Таким образом, $\lambda_{\Sigma} = \lambda_1 + \lambda_2 + \ldots + \lambda_n = \sum_{i=1}^{i=n} \lambda_i$. Отсюда $T_{ср} = \frac{1}{\lambda_{\Sigma}}$.

4.3. Методы расчета надежности систем

Устройства промышленной автоматики, как правило, не требуют очень высоких показателей надежности, т. к. доступны для ремонта и обслуживания. Значение $p(t)$ принимается равным 0,98–0,99. При

проектировании систем автоматики обычно принимается экспоненциальное распределение времени безотказной работы, т. е. вероятность отказа не зависит от того, сколько времени проработало устройство до отказа.

Существует несколько методов расчета надежности.

1. Метод расчета по среднегрупповым значениям интенсивности отказов. Количество элементов может быть большим (сотни и тысячи), интенсивность отказов λ каждого элемента известна. Систему разбивают на *r*-группы с примерно одинаковыми λ внутри. $\lambda_{\Sigma} = \sum_{i=1}^{r} N_i \lambda_i$, где λ_i — усредненная интенсивность отказов *i*-ой группы, а N_i — количество элементов в группе. Среднее время безотказной работы $T_{\text{ср}} = \frac{1}{\lambda_{\Sigma}}$.

2. Метод с использованием данных системы-аналога. Метод применяется при проектировании системы, имеющей действующий аналог, показатели надежности которого известны (в частности, получены при анализе опыта эксплуатации). Проектируемая система имеет аналогичную структуру и аналогичную элементную базу.

Исходными являются следующие данные:

– $T_{\text{ср}\,a}$ — среднее время безотказной работы системы-аналога;
– N_a — количество элементов системы-аналога;
– N_n — количество элементов проектируемой системы.

Среднее время безотказной работы проектируемой системы $T_{\text{ср}\,n}$ находится просто: $T_{\text{ср}\,n} = \frac{N_a}{N_n} T_{\text{ср}\,a}$; $\lambda_n = \frac{1}{T_{\text{ср}\,n}}$.

3. Коэффициентный метод расчета надежности. Метод основан на использовании достоверно известных высокими показателями надежности элементов. Для механических устройств это могут быть, например, знаменитые неубиваемые лом или кувалда, для систем автоматизации и электропривода — скромные керамический конденсатор или резистор. Показатель надежности *i*-го элемента системы представляется в виде коэффициента $K_i = \frac{\lambda_i}{\lambda_0} = \text{const}$, где λ_i — интенсивность отказов *i*-го элемента; λ_0 — интенсивность отказа элемента, достоверно известное (например, резистора). Обычно берется $\lambda_0 = 10^{-6}$.

Для решения задачи используют методику расчета по среднегрупповым значениям интенсивности отказов. Все элементы системы

разбивают на группы с близкими значениями k, затем определяется интенсивность отказов системы.

$\lambda_i = K_i\lambda_0$. Для группы $\lambda = N_i K_i \lambda_0$.

$$\lambda_\Sigma = \lambda_0 \sum_{i=1}^{m} N_i K_i,$$

где m — общее число групп.

Отсюда, среднее время безотказной работы $T_{\text{ср}\Sigma}$:

$$T_{\text{ср}\Sigma} = \frac{1}{\lambda_0 \sum_{i=1}^{m} N_i K_i} = \frac{T_0}{\sum_{i=1}^{m} N_i K_i},$$

где

$$T_0 = \frac{1}{\lambda_0};$$

$$P_t = \exp\left(-t\lambda_0 \sum_{i=1}^{m} N_i K_i\right).$$

Значения λ, T и k для элементов систем автоматизации и электропривода можно найти в справочниках, в частности, в справочнике по автоматизированному электроприводу под редакцией В. А. Елисеева и А. В. Шинянского [15].

Рассмотрим эксплуатационные расходы для систем автоматики при заданном значении вероятности безотказной работы p и известной общей интенсивности отказов системы. Время t, определенное из $p(t) = \exp(-\lambda t)$, можно считать временем $t_{\text{ппо}}$, через которое необходимо проводить планово-профилактические осмотры и обслуживание, чтобы поддержать вероятность безотказной работы системы на заданном уровне $p_{\text{зад}} = 0{,}98\ldots0{,}99$.

$$t_{\text{ппо}} = -\frac{\text{In}\left(p_{\text{зад}}\right)}{\lambda},$$

где λ — общая интенсивность отказов системы.

В процессе эксплуатации это время может быть скорректировано на основании реального потока отказов.

Подводя итог рассмотрению методов расчета показателей надежности систем автоматики, приведем математические зависимости, связывающие между собой основные показатели надежности.

$$P(t)=\begin{cases} 1-q(t); \\ 1-\int\limits_0^t a(t)dt; \\ e^{-\lambda t}. \end{cases}$$

$$q(t)=\begin{cases} 1-p(t); \\ \int\limits_0^t a(t)dt; \\ 1-e^{-\lambda t}. \end{cases}$$

Если воспользоваться разложением в ряд Маклорена, то

$$q(t)=1-e^{-\lambda t}=1-\left[1-\lambda t+\frac{\lambda^2 t^2}{2}-\frac{\lambda^3 t^3}{6}+\ldots\right]\cong\lambda t;$$

$$a(t)=\begin{cases} -\dfrac{dp(t)}{dt}; \\ \dfrac{dq(t)}{dt}; \\ \lambda(t)e^{-\lambda t}; \end{cases}$$

$$\lambda=\begin{cases} -\dfrac{1}{P(t)}\dfrac{dp}{dt}; \\ \dfrac{1}{1-q(t)}\dfrac{dq}{dt}; \\ \dfrac{a(t)}{\int\limits_0^t a(t)dt}; \\ \lambda\simeq\dfrac{q}{t}; \end{cases}$$

$$T_{ср} = \begin{cases} \dfrac{1}{\lambda}; \\ -\dfrac{p(t)}{\dfrac{dp}{dt}}; \\ \dfrac{1-q(t)}{\dfrac{dq}{dt}}; \\ \dfrac{\int_0^t a(t)dt}{a(t)}. \end{cases}$$

Таким образом, система полученных зависимостей определяет среднее время безотказной работы $T_{ср}$.

4.4. Методы повышения надежности систем автоматики

На этапе проектирования создается минимально необходимый вариант системы автоматики, который позволяет обработать минимально необходимую информацию за минимально допустимое время [13, 14]. Отказ любого элемента приводит к прекращению функционирования всей системы. Таким образом, характеристики надежности минимально необходимого варианта системы не всегда удовлетворяют предъявляемым требованиям, что вынуждает находить способы повышения надежности системы.

Проблема повышения надежности должна решаться в первую очередь на основе разработки и применения высоконадежных элементов. Как показывает практика, этот путь повышения надежности не всегда позволяет создавать высоконадежные системы. Действительно, средний уровень надежности современных элементов характеризуется значениями интенсивности отказов $\lambda = 10^{-6} \div 10^{-8}$ 1/ч. При количестве элементов в системе $n = 10^6$ среднее время безотказной работы $T_{ср} = n\lambda$ составляет 100 ч, и этого явно недостаточно. Необходимая

надежность сложных систем может быть достигнута только при использовании различных видов резервирования.

В соответствии с ГОСТ 27.002–2015[1], «резервирование — способ обеспечения надежности объекта за счет использования дополнительных средств и (или) возможностей, избыточных по отношению к минимально необходимым для выполнения требуемых функций».

Таким образом, «избыточность — это дополнительные средства и возможности сверх минимально необходимых для выполнения объектом заданных функций». Задача введения избыточности — обеспечить нормальное функционирование объекта после возникновения отказов в его элементах. В теории надежности выделяются структурное, информационное и временное резервирование.

При этом структурное резервирование, называемое еще аппаратным, — это резервирование, предусматривающее использование избыточных элементов объекта.

Суть структурного резервирования заключается в том, что в минимально необходимый вариант системы, элементы которой называют основными, вводятся дополнительные элементы, устройства либо даже вместо одной системы предусматривается использование нескольких идентичных систем. При этом эти избыточные структурные элементы, называемые резервными элементами, имеют единственное назначение — взять на себя выполнение рабочих функций при отказе соответствующих основных элементов.

Информационное резервирование — это резервирование, предусматривающее для повышения надежности использование избыточной информации. Его простейшим примером является многократная передача одного и того же сообщения по каналу связи. В цифровых вычислительных машинах применяются коды, обнаруживающие и исправляющие ошибки, которые появляются в результате сбоев и отказов.

Следует заметить, что использование информационного резервирования практически всегда влечет за собой необходимость введения избыточных элементов.

Временное резервирование — это резервирование, предусматривающее для повышения надежности использование избыточного времени. При этом предполагается, что на выполнение системой необходимой работы отводится время, заведомо большее минимально необходимого времени, достаточное для ее восстановления и возобновления функционирования после отказа.

---

[1] Межгосударственный ГОСТ 27.002–2015 «Надежность в технике. Термины и определения» утратил силу в РФ.

Перечисленные выше виды резервирования могут быть применены либо к системе в целом, либо к отдельным элементам системы или к их группам. В первом случае резервирование называется общим, во втором — раздельным.

Наиболее широко используемым, а поэтому и наиболее изученным, является структурное резервирование [13, 14]. При его использовании можно применить различные схемы включения резервных элементов. При этом важное значение имеет режим работы резервных элементов до и после появления отказа в основных элементах, а также кратность резервирования. По этим признакам структурное резервирование можно классифицировать так, как показано на рис. 4.3 на стр. 130.

Резервирование замещением — это такое резервирование, при котором функции основного элемента передаются резервному элементу только после отказа основного элемента.

При использовании резервирования замещением необходимы контролирующие и переключающие устройства для обнаружения отказа основного элемента и переключения с основного на резервный элемент.

Постоянное резервирование — это резервирование, при котором используется нагруженный резерв и при отказе любого элемента в резервированной группе выполнение объектом требуемых функций обеспечивается оставшимися элементами без переключений.

При этом основные и резервные элементы могут иметь общий вход и общий выход, в частности гальваническую связь по входу и выходу, а могут быть и автономными, т. е. такой связи не иметь.

Общее резервирование — это резервирование, при котором резервируется объект в целом.

Раздельное резервирование — это резервирование, при котором резервируются отдельные элементы объекта или их группы.

Скользящее резервирование — это резервирование замещением, при котором группа основных элементов объекта резервируется одним или несколькими резервными элементами, каждый из которых может заменить любой отказавший элемент в данной группе.

В зависимости от режима работы резервных элементов различают нагруженный, облегченный и ненагруженный резервы.

Нагруженный резерв — резерв, который содержит один или несколько резервных элементов, находящихся в режиме основного элемента. Резервный элемент находится в том же режиме, что и основной.

Облегченный резерв — резерв, который содержит один или несколько резервных элементов, находящихся в менее нагруженном режиме, чем основной элемент. Таким образом, принимается, что харак-

теристики надежности резервных элементов в период их пребывания в качестве резервных элементов выше, чем в период их использования в качестве основных после их отказа.

Рис. 4.3. Виды структурного резервирования

Ненагруженный резерв — резерв, который содержит один или несколько резервных элементов, находящихся в ненагруженном режиме до начала выполнения ими функции основного элемента. Таким образом, при ненагруженном резерве резервный элемент практически не несет нагрузки. При этом принимается, что такой резервный элемент, находясь в резерве, отказывать не должен, т. е. обладает в этот период идеальной надежностью; в период же использования этого элемента вместо основного после отказа последнего надежность резервного элемента становится равной надежности основного.

Степень избыточности характеризуется кратностью резервирования.

Кратностью резервирования называется отношение числа резервных элементов к числу резервируемых или основных элементов объекта, выраженное несокращенной дробью. Различают резервирование с целой и дробной кратностью. Резервирование с целой кратностью имеет место, когда один основной элемент резервируется одним и более резервными элементами. Резервирование с дробной кратностью имеет место, когда два и более однотипных элемента резервируются одним и более резервными элементами. Наиболее распространенным

вариантом резервирования с дробной кратностью является такой, когда число основных элементов превышает число резервных.

Резервирование, кратность которого равна единице, называется дублированием.

Рассмотрим принципы расчета надежности систем с резервом. Как уже отмечалось выше, наиболее распространенным является структурное (аппаратное) резервирование. На рис. 4.4 представлены расчетно-логические схемы видов структурного резервирования.

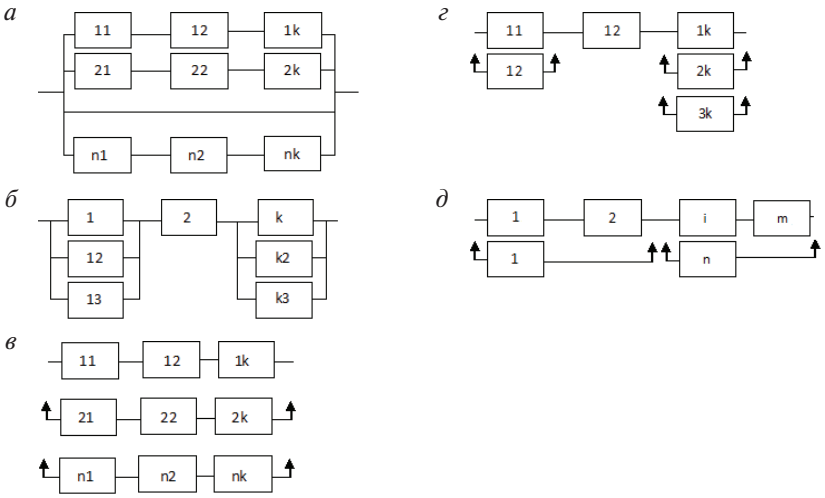


Рис. 4.4. Расчетно-графические схемы видов структурного резервирования:

а — общее резервирование и постоянное включение резерва;
б — раздельное резервирование и постоянное включение резерва;
в — общее резервирование и включение резерва замещением;
г — раздельное резервирование и включение резерва замещением;
д — резервирование с дробной кратностью при скользящем резерве

Как видно, имеет место как параллельное соединение резервных элементов с основными (*в*, *г*, *д*), так и комбинация последовательно-параллельного включения (*а*, *б*).

Расчетные схемы для общего резервирования (рис. 4.4, *а*, *в*) сводятся к расчетно-логической схеме резервированного элемента (рис. 4.4, *б*) путем замены последовательно соединенных элементов (блоков, устройств и т. п.) эквивалентными элементами с показателями надежности, определяемыми по формулам:

$$P(t) = P_1 P_2 \ldots P_k = \prod_{i=1}^{k} P_i(t); \;\; \lambda(t) = \lambda_1 + \lambda_2 \ldots + \lambda_k = \sum_{i=1}^{k} \lambda_i(t),$$

где $P_i(t)$ и $\lambda_i(t)$ — вероятность безотказной работы и интенсивность отказов i-го элемента, k — число последовательно соединенных элементов.

При использовании раздельного резервирования (рис. 4.4, *б*, *г*, *д*) расчет показателей надежности производится путем замены параллельно соединенных элементов (блоков, устройств и т. п.) эквивалентным элементом с показателями надежности, определяемыми по формуле

$$q_p(t) = q_1 q_2 q_k = \prod_{i=1}^{k} q_i(t),$$

где q_i — вероятность отказа i-го элемента, k — число параллельно соединенных элементов.

Поскольку $q(t) = 1 - p(t)$,

$$1 - p(t) = (1 - P_1)(1 - P_2).$$

$$(1 - P_k) = \prod_{i=1}^{k} (1 - P_i).$$

Оценить эффективность такого повышения надежности системы автоматики можно на примере дублирования, когда основной элемент резервируется одним резервным элементом с таким же показателем надежности, как и основной элемент,

$$1 - P_{\partial}(t) = (1 - P_1(t)\left(1 - p_2(t)\right) = \left(1 - P(t)\right)^2.$$

$$1 - P_{\partial}(t) = 1 - 2P(t) + P^2(t).$$

$P_{\partial}(t) = 2P(t) - P^2(t)$, отсюда получаем $P_{\partial}(t)dt = 2P(t)dt - P^2(t)dt$.

$$\int_0^{\infty} P_{\partial}(t)dt = \int_0^{\infty} 2P(t)dt - \int_0^{\infty} P^2(t)dt.$$

$$\int_0^{\infty} P_{\partial}(t)dt = 2\int_0^{\infty} e^{-\lambda t}dt - \int_0^{\infty} e^{-2\lambda t}dt, \text{ т. к. } P(t) = e^{-\lambda t}.$$

Как известно, среднее время безотказной работы $T_{\text{ср}} = \int_0^{\infty} P(t)dt$, а $\int_0^{\infty} e^{-\lambda t}dt = \frac{1}{\lambda}$.

В итоге $T_{\text{ср}\partial} = \frac{2}{\lambda} - \frac{1}{2\lambda} = \frac{3}{2}\frac{1}{\lambda}$.

Поскольку $\frac{1}{\lambda} = T_{ср}$ одного элемента, среднее время безотказной работы системы с дублированием $T_{ср\partial} = \frac{3}{2} T_{ср}$.

Оценим показатели надежности элементов основной и резервной систем. Постоянно включенный в работу (нагруженный) резерв: $P_{рез} = P_{раб}$. Облегченный резерв: $P_{рез} > P_{раб}$.

Ненагруженный резерв: $P_{рез} = 1$. Резервный элемент практически не несет нагрузки, т. е. до включения в работу обладает идеальной надежностью. В период его использования после отказа основного элемента (или системы) его надежность становится равной надежности основного элемента.

На практике встречаются системы, для работы которых необходимо нормальное функционирование *m* (или более) элементов из общего количества *n*, соединенных параллельно.

Структурная схема такой системы, для которой $n = 3$, $m = 2$, имеет вид, показанный на рис. 4.5. Таким образом, изображенная на рисунке система допускает отказ не более одного элемента. М — мажоритарный или голосующий элемент. Виды резервирования представлены в табл. 4.1.

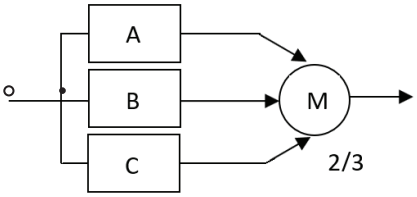
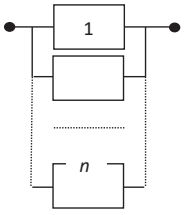


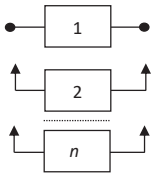
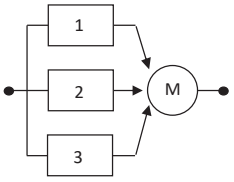
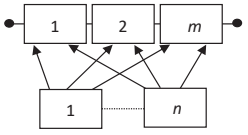
Рис. 4.5. Структурная схема надежности с мажоритарным резервированием

Таблица 4.1

Виды резервирования

Вид резервирования	Структурная схема	Расчетные формулы показателей надежности
Резервирование замещением: нагруженный резерв. Постоянное включение резерва		$P_p(t) = 1 - Q_p(t) = 1 - \prod_{i=1}^{n} \left[1 - P_i(t)\right];$ $P_{\partial}(t) = 2e^{-\lambda t} - e^{-2\lambda t};$ $T_{ср\,p} = \frac{1}{\lambda} \sum_{i=1}^{n} \frac{1}{n-i+1} = T_{ср} \sum_{i=1}^{n} \frac{1}{n-i+1}$

Окончание табл. 4.1

Вид резервирования	Структурная схема	Расчетные формулы показателей надежности
Резервирование замещением: ненагруженный резерв с целой кратностью		$P_p(t)=\sum_{i=0}^{n-1}\frac{(\lambda\Delta t)^i}{i!}e^{-\lambda t};$ $P_{\hat{c}}(t)=(1+\lambda\Delta t)e^{-\lambda t};$ $T_{\text{ср}\,p}=n\lambda^{-1}=nT_{cp}$
Мажоритарное резервирование		$m=2,\ n=1;$ $P_p(t)=P_p(t)\left[3P^2(t)-2P^2(t)\right];$ $T_{\text{ср}\,p}=\frac{5}{6}T_{\text{ср}}$
Ненагруженный резерв: резервирование с дробной кратностью (скользящее резервирование)		$P_p(t)=\sum_{i=0}^{n-1}\frac{(m\lambda\Delta t)^i}{i!}e^{-m\lambda\Delta t};$ $T_{\text{ср}\,p}=\frac{n+1}{m\lambda}=\frac{n+1}{m}T_{\text{ср}}$

Мажоритарное резервирование является разновидностью нагруженного резервирования с дробной кратностью. При таком резервировании вместо одного элемента (канала) включаются три идентичных элемента (канала), выходы которых подключаются к мажоритарному органу М (элементу голосования). На выходе М будет сигнал, совпадающий с большинством сигналов на его входе, т. е. мажоритарный орган осуществляет операцию голосования или выбора по большинству. Таким образом, условие безопасной работы группы при мажоритарном резервировании является безотказная работа любых двух элементов из трех и мажоритарного органа в течение заданного времени *t*.

В общем случае число каналов *n* должно быть нечетным (3, 5, ...), чаще всего *n* = 3. В случае высоких требований по обеспечению безотказной работы системы в течение заданного времени *t* используют пять каналов (например, в системах управления летательных пассажирских аппаратах).

Рассмотрим пример расчета надежности.

Расчет надежности систем с резервом предполагает использование довольно большого числа методов, самым распространенным из которых является метод свертки. Для расчета необходимо иметь полную

структуру системы и показатели надежности каждого ее элемента. Рассмотрим реализацию этого метода на примере системы, показанной на рис. 4.6.

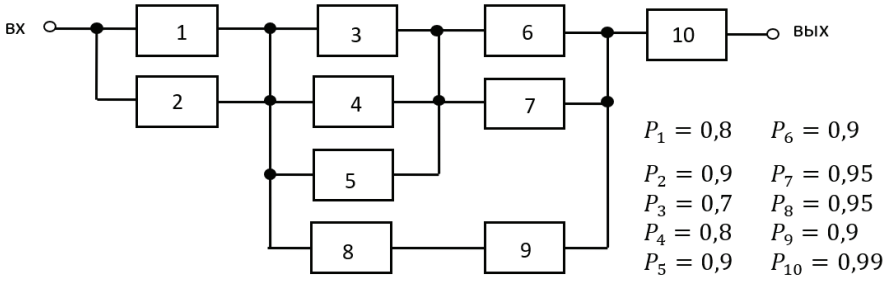


Рис. 4.6. Схема системы

I этап: упрощаются все параллельные соединения, в результате получается следующая структура (рис. 4.7).

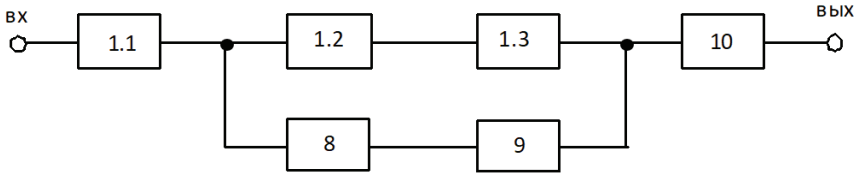


Рис. 4.7. Преобразованная схема системы

При преобразовании параллельных схем имеем

$$q_{1.1} = q_1 q_2 \rightarrow P_{1.1} = 1 - (1 - P_1)(1 - P_2) = 0{,}98;$$
$$q_{1.2} = q_3 q_4 q_5 \rightarrow P_{1.2} = 1 - (1 - P_3)(1 - P_4)(1 - P_5) = 0{,}994;$$
$$q_{1.3} = q_6 q_7 \rightarrow P_{1.3} = 1 - (1 - P_6)(1 - P_7) = 0{,}995.$$

II этап: упрощаются все последовательно соединенные структуры, в результате получаем следующую схему (рис. 4.8).

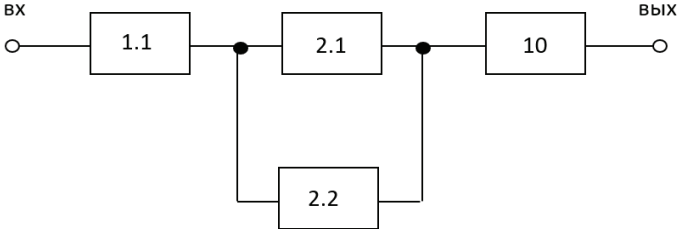


Рис. 4.8. Упрощенная схема системы

$$P_{2.1} = P_{1.2}P_{1.3} = 0,989;$$
$$P_{2.2} = P_8P_9 = 0,885.$$

III этап: вновь упрощаем параллельные соединения, в результате получаем простейшую структуру (рис. 4.9).

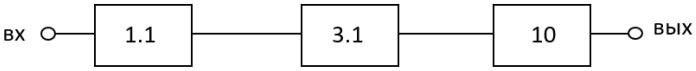


Рис. 4.9. Итоговая расчетная схема системы

$$P_{3.1} = 1 - (1 - P_{2.1})(1 - P_{2.2}) = 0,999.$$

Таким образом, показатель надежности рассмотренной системы $P_p = P_{1.1}P_{3.1}P_{10} = 0,97$.

Контрольные вопросы к главе 4

1. Перечислите основные термины теории надежности.
2. С какой целью выполняются расчеты показателей надежности систем технологической автоматики?
3. Что понимается под термином «Аппаратурная надежность»?
4. Поясните значение термина «Структурная надежность».
5. Какие методы расчета надежности вам известны?
6. Перечислите способы повышения надежности систем технологической автоматики.
7. Опишите суть мажоритарного резервирования.

Список использованных источников

1. Корытин А. М., Петров Н. К., Радимов С. Н. Автоматизация типовых технологических процессов и установок. М. : Энергоатомиздат, 1988. 432 с.
2. Соснин О. М. Основы автоматизации технологических процессов и производств : учебное пособие для студентов. М. : Академия, 2007. 240 с.
3. Грейнер Г. Р., Ильяшенко В. П., Май В. П. Проектирование бесконтактных управляющих логических устройств промышленной автоматики. М. : Энергия, 1977. 384 с.
4. Токмакова Л. И. Системы управления электроприводов : учебное пособие : в 2-х ч. Владивосток : Изд-во ДВГТУ, 2002. Ч. 1. 42 с.
5. Чикуров Н. Г. Логический синтез дискретных систем управления : учебное пособие. Уфа : Уфимский государственный авиационный технический университет, 2003. 132 с.
6. Шандров Б. В., Чудаков А. Д. Технические средства автоматизации : учебник. М. : Академия, 2007. 368 с.
7. Куцин В. В. Синтез алгоритмов систем технологической автоматики и методы их реализации : методические указания. Екатеринбург : УГТУ-УПИ, 1995. 36 с.
8. Чернов Е. А. Проектирование станочной автоматики. М. : Машиностроение, 1989. 304 с.
9. Петров И. В. Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования. М. : СОЛОН-Пресс, 2004. 256 с.
10. Нестеров К. Е., Зюзев А. М. Программирование промышленных контроллеров : учебно-методическое пособие. Екатеринбург : Изд-во Урал. ун-та, 2019. 105 с.
11. Куцин В. В. Автоматизация типовых технологических процессов : лабораторный практикум. Екатеринбург : УГТУ-УПИ, 2007. 67 с.

12. Глазунов Л. П., Грабовецкий В. П. Основы теории надежности автоматических систем управления. Л. : Энергоатомиздат, 1984. 208 с.

13. Ястребенецкий М. А., Иванова Г. М. Надежность автоматизированных систем управления технологическими процессами. М. : Энергоатомиздат, 1989. 264 с.

14. Шкляр В. Н. Надежность систем управления : учебное пособие. Томск : Издательство Томского политехнического университета, 2009. 126 с.

15. Справочник по автоматизированному электроприводу / под ред. В. А. Елисеева, А. В. Шинянского. М. : Энергоатомиздат, 1983. 616 с.

Учебное издание

Куцин Валерий Васильевич
Нестеров Константин Евгеньевич
Кириллов Андрей Владиславович

АВТОМАТИЗАЦИЯ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ

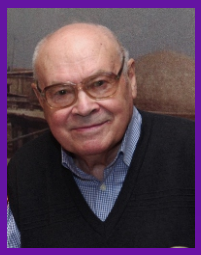
Редактор *З. Р. Картавцева*
Верстка *Ю. В. Ершовой*

Подписано в печать 22.07.2024. Формат 70×100 1/16.
Бумага офсетная. Цифровая печать. Усл. печ. л. 11,3.
Уч.-изд. л. 8,9. Тираж 30 экз. Заказ № 97.

Издательство Уральского университета
Редакционно-издательский отдел ИПЦ УрФУ ЦСД
620062, Екатеринбург, ул. С. Ковалевской, 5.
Тел.: 8 (343) 375-48-25, 375-46-85, 374-19-41
E-mail: rio@urfu.ru

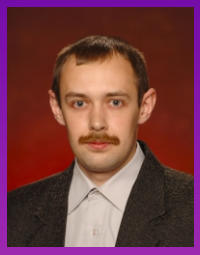
Отпечатано в Издательско-полиграфическом центре УрФУ ЦСД
620062, Екатеринбург, ул. Тургенева, 4
Тел.: 8 (343) 358-93-06, 350-58-20, 350-90-13
http://print/urfu.ru

9 785799 638658



КУЦИН ВАЛЕРИЙ ВАСИЛЬЕВИЧ
Кандидат технических наук, доцент кафедры «Электропривод и автоматизация промышленных установок» УралЭНИН УрФУ.

Область научных интересов: современные электроприводы с цифровым управлением, системы автоматизации зданий и сооружений, энергосбережение и повышение энергетической эффективности систем энергоснабжения и энергопотребления современных бюджетных учреждений.



НЕСТЕРОВ КОНСТАНТИН ЕВГЕНЬЕВИЧ
Кандидат технических наук, доцент кафедры «Электропривод и автоматизация промышленных установок» УралЭНИН УрФУ.

Область научных интересов: асинхронные электроприводы с полупроводниковыми преобразователями, системы промышленной автоматизации.

КИРИЛЛОВ АНДРЕЙ ВЛАДИСЛАВОВИЧ
Кандидат технических наук, доцент кафедры «Электропривод и автоматизация промышленных установок» УралЭНИН УрФУ.

Область научных интересов: электроприводы с цифровым управлением, полупроводниковые приборы, датчики систем автоматики, системы промышленной автоматизации.