Министерство образования Российской Федерации
Уфимский государственный авиационный технический университет

Н. Г. Ч И К У Р О В

ЛОГИЧЕСКИЙ СИНТЕЗ ДИСКРЕТНЫХ СИСТЕМ УПРАВЛЕНИЯ

Учебное пособие

Уфа 2003

Министерство образования Российской Федерации
Уфимский государственный авиационный технический университет

Н. Г. Ч И К У Р О В

ЛОГИЧЕСКИЙ СИНТЕЗ ДИСКРЕТНЫХ СИСТЕМ УПРАВЛЕНИЯ

Учебное пособие

Уфа 2003

УДК 621.9.06
ББК 34.630.2
Ч-60

Чикуров Н. Г. Логический синтез дискретных систем управления: Учебное пособие / Н. Г. Чикуров; Уфимск. гос. авиац. техн. ун-т; – Уфа, 2003. – 132 с. ISBN 5-86911-432-2

Рассмотрены инженерные методы анализа и синтеза дискретно-логических систем управления промышленных механизмов. на основе аппарата алгебры логики и циклограмм работы этих механизмов. Усовершенствован универсальный метод, позволяющий достаточно быстро синтезировать сложные системы управления на различной элементной базе. Приведены примеры проектирования систем управления устройствами станочной электроавтоматики. Дана методика программирования логических контроллеров в инструментальной среде программирования ISaGRAF.

Учебное пособие соответствует государственному образовательному стандарту дисциплин «Компьютерное управление мехатронными системами» и «Электромеханические и мехатронные системы» по направлению 652000 специальности 071800 «Мехатроника».

Предназначено для использования студентами при выполнении курсовых и дипломных проектов. Оно может быть полезным и для специалистов, занимающихся разработкой дискретно-логических систем управления на основе промышленных контроллеров.

Табл. 2 Ил. 121 Библиогр.: назв. 8

Печатается по решению редакционно-издательского совета УГАТУ.

Научный редактор др. техн. наук, проф. В.Ц.Зориктуев.
Рецензенты:
Уфимский филиал ЗАО РТСофт, Дир. А.А.Болдырев;
ОАО «Стерлитамак-МТЕ», Гл. конструктор В.Л.Зинов.
Уфимский государственный ин-т сервиса, Проф. Л.Н.Касимов.

ISBN 5-86911-432-2

Содержание

Предисловие

Учебное пособие посвящено проблемам проектирования дискретно-логических систем управления промышленной электроавтоматики, реализуемой на различной элементной базе: релейно-контактных схемах, бесконтактных интегральных микросхемах и на основе программируемых логических контроллеров (ПЛК).

Это сложная техническая задача, для решения которой разработчик должен хорошо владеть теорией устройств дискретного действия и математическим аппаратом алгебры логики.

Большинство практических примеров, представленных в учебном пособии, связано с синтезом систем управления механизмами металлорежущих станков. В современных станках автоматизированы многочисленные операции технологического обеспечения: управление автоматической сменой инструмента, управление переключениями в приводах главного движения, управление зажимными приспособлениями, охлаждением, смазкой, перемещением ограждений и др.

В учебном пособии развивается универсальный метод синтеза дискретно-логических систем управления на основе циклограмм работы механизмов. Этот метод дает описание алгоритма управления с помощью логических функций, которые затем достаточно легко реализовать в виде электрической схемы или программы для ПЛК.

Автором внесен в метод ряд усовершенствований, благодаря которым стало возможным проектировать не только последовательные, но и параллельные циклические процессы (циклы), протекающие одновременно в различных частях технологического объекта, а также организовывать условные переходы к циклам и обращаться из основного цикла к другим циклам как к подпрограммам.

Особое внимание уделено программируемым логическим контроллерам и новым возможностям, которые открываются при их использовании в дискретных системах управления.

Описываемая в учебном пособии методика пригодна для синтеза любых промышленных механизмов.

Из-за ограниченного объема в учебное пособие не включены другие известные методы синтеза дискретных автоматов: метод ориентированных графов, метод сетей Петри и др. Описание этих методов можно найти в книгах [1, 6, 8].

Минимизация логических функций представлена только методом карт Карно. О табличном методе минимизации можно прочитать в книгах [1, 8]. Для минимизации логических функций можно использовать также специальную программу (см. приложение).

Учебное пособие написано на основе курса лекций, читаемых в Уфимском государственном авиационном техническом университете (УГАТУ) для студентов специальности «Мехатроника».

При написании учебного пособия использованы некоторые определения, формулы и примеры из книги Гаврилова М.А., Девяткова В.В., Пупырева Е.И. [1], а также из книг Рабиновича А.Н. [5] и Рогинского В.Н. [4]. В описании системы программирования *ISaGRAF* использованы инструкции, которые прилагаются к данному программному продукту.

1. ПРИМЕНЕНИЕ МАТЕМАТИЧЕСКОЙ ЛОГИКИ ДЛЯ ПОСТРОЕНИЯ ДИСКРЕТНЫХ СИСТЕМ УПРАВЛЕНИЯ

1.1. Логические функции

Рассмотрим некоторое устройство, которое имеет n входов и один выход (рис. 1.1)

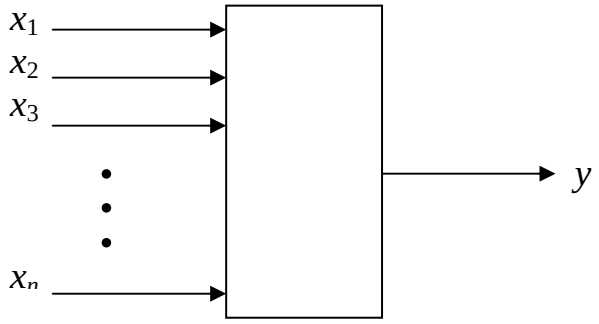


Рис. 1.1. Логическое устройство

На каждый из входов может быть подан сигнал в виде электрического напряжения. Обозначая эти напряжения через x_1 , x_2 и т.д., получим следующий набор:

$$\{x_1, x_2, x_3, \ldots, x_n\}.$$

Каждое из напряжений может принимать только два значения, которые условно обозначим с помощью символов **0** и **1**, т.е. включено или выключено. Следовательно, набор состоит из цифр **0** и **1**.

Наличие или отсутствие сигнала на выходе устройства зависит от комбинации сигналов на его входе, т.е. является функцией

$$f(x_1, x_2, x_3, \ldots, x_n),$$

которая называется *функцией алгебры логики* или просто *логической функцией*. Эта функция также может принимать только два значения: **0** и **1**.

Обозначим логическую функцию через y :

$$y = f(x_1, x_2, x_3, \ldots, x_n).$$

Рассмотрим логические элементы, которые реализуют основные логические функции **НЕ**, **ИЛИ**, **И**.

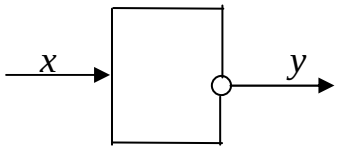
1) Элемент **НЕ** (инвертор).

Имеет только один вход, причем сигнал на выходе элемента всегда имеет значение, обратное значению сигнала на входе, т.е. элемент реализует функцию

$$y = f(x) = \bar{x}.$$

Черта над буквой x означает инверсию или логическое отрицание (**НЕ**) переменной x.

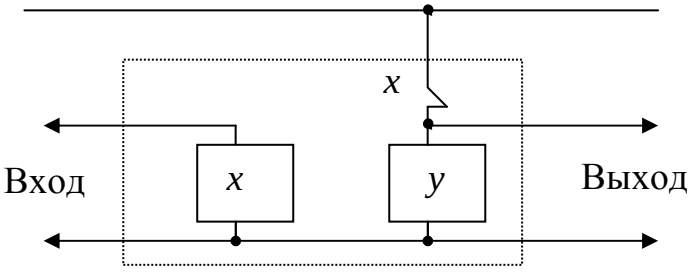
На функциональных схемах инвертор изображают в виде прямоугольника с прозрачным кружком на входе (рис. 1.2, *а*).



а

X	*y*
0	1
1	0

в



б

Рис. 1.2. Элемент **НЕ** (инвертор)

Элемент может быть бесконтактным (на основе полупроводников) и контактным (на основе электромеханических реле).

В контактных схемах инвертор реализуется размыкающим контактом (рис. 1.2, *б*). Состояния сигналов на входе и на выходе инвертора показаны в табличке, на рис. 1.2, *в*.

2) Элемент **ИЛИ** в простейшем случае имеет два входа и осуществляет логическую операцию сложения (дизъюнкцию), которую обозначают знаком « + » или знаком « $\vee$ ».

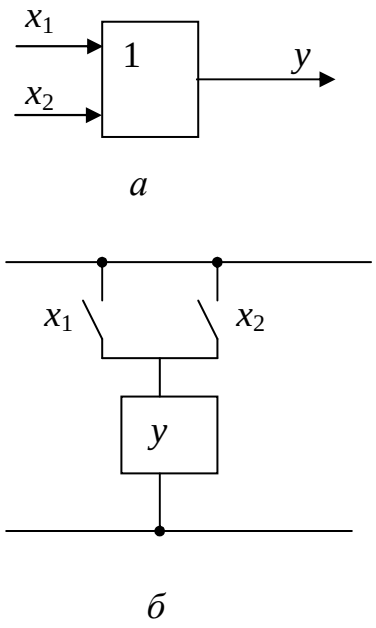
$$y = f(x_1, x_2) = x_1 + x_2 = x_1 \vee x_2.$$

В дальнейшем мы будем использовать первое обозначение.

На функциональных схемах элемент **ИЛИ** изображают в виде прямоугольника с цифрой **1** в верхнем левом углу (рис. 1.3, *а*).

В электрической схеме на электромеханических реле логическое сложение реализуется путем параллельного соединения контактов (рис. 1.3, *б*).

Из таблицы состояния элемента (рис. 1.3, *в*) видно, что *сигнал на выходе элемента ИЛИ равен единице, если хотя бы один сигнал на входе равен единице*. Соответственно *сигнал на выходе равен нулю только тогда, когда равны нулю сигналы на всех входах*.



а

б

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

в

Рис. 1.3. Элемент **ИЛИ**

Свойства логической суммы:

$$x + x = x \;;\;\; x + \bar{x} = 1 \;;\;\; x + 0 = x \;;\;\; x + 1 = 1 \;,$$

где константе **1** соответствует постоянно замкнутая цепь или высокий уровень напряжения, а константе **0** – постоянно разомкнутая цепь или низкий уровень напряжения.

Указанные свойства легко установить, моделируя работу элемента **ИЛИ** в контактном исполнении.

Заметим, что элемент **ИЛИ** в общем случае может иметь число входов больше двух.

3) Элемент **И** в простейшем случае имеет два входа и выполняет логическую операцию умножения (конъюнкцию), которую обозначают знаком умножения (точкой) или знаком " $\wedge$ ".

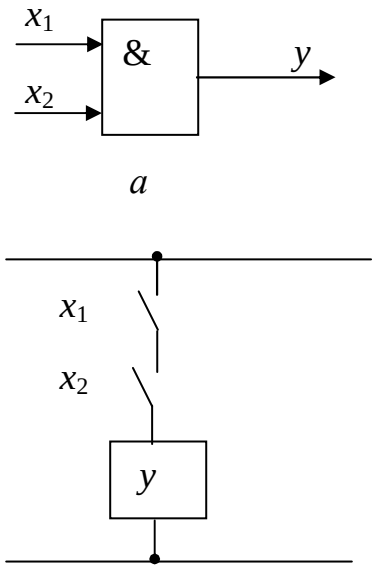
$$y = f(x_1, x_2) = x_1 x_2 = x_1 \wedge x_2 .$$

Далее мы будем применять первое обозначение.

На функциональных схемах элемент **И** обозначают в виде прямоугольника со знаком **&** в верхнем левом углу (рис. 1.4, *а*).

В контактных схемах логическое умножение реализуют путем последовательного соединения контактов (рис. 1.4, *б*).

Проанализировав работу элемента (рис. 1.4, *в*), можно заключить, что *сигнал на выходе элемента* **И** *равен нулю, если хотя бы один сигнал на входе равен нулю*. Соответственно *сигнал на выходе равен единице только тогда, когда равны единице сигналы на всех входах*.



а

б

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

в

Рис. 1.4. Элемент **И**

Свойства логического произведения:

$$xx = x \ ; \quad x\bar{x} = 0 \ ; \quad x1 = x \ ; \quad x0 = 0 \ .$$

Данные свойства легко доказать, если смоделировать их с помощью контактного элемента **И**.

Также, как и элемент **ИЛИ**, элемент **И** может иметь число входов больше двух.

1.2. Законы алгебры логики

1) Закон нулевого множества

$0\,x = 0;$

$0\,x_1\,x_2\,\ldots\,x_n = 0.$

Произведение любого числа переменных обращается в нуль, если хотя бы одна переменная имеет значение нуль.

2) Закон тавтологии

$x\,x = x;$
$x\,x \ldots x = x;$
$x + x = x;$
$x + x + x + \ldots + x = x.$

Истина, повторенная несколько раз, остается истиной.

3) Закон двойной инверсии

$\overline{\overline{x}} = x.$

Отрицание отрицания величины равно самой величине.

4) Переместительные (коммутативные) законы

$x\,y = y\,x;$
$x + y = y + x.$

Результат выполнения операций умножения и сложения не зависит от того, в каком порядке следуют переменные.

5) Сочетательные (ассоциативные) законы

$x\,(y\,z) = (x\,y)\,z = x\,y\,z;$
$x + (y + z) = (x + y) + z = x + y + z.$

Для записи умножения или сложения скобки можно опустить.

6) Распределительные (дистрибутивные) законы

$x\,(y + z) = x\,y + x\,z;$
$x + y\,z = (x + y)\,(x + z).$

7) Законы поглощения

$x\,(x + y) = x;$
$x + x\,y = x;$
$x(\overline{x} + y) = xy;$
$x + \overline{x}y = x + y.$

8) Законы склеивания

$$x y + x \bar{y} = x;$$
$$(x + y)(x + \bar{y}) = x.$$

9) Законы инверсии (Де Моргана)

а) для двух переменных

$$\overline{xy} = \bar{x} + \bar{y},$$

т. е. *инверсия произведения равна сумме инверсий*;

$$\overline{x + y} = \bar{x}\bar{y},$$

инверсия суммы есть произведение инверсий;

б) для n переменных

$$\overline{x_1 x_2 x_3 \ldots x_n} = \bar{x}_1 + \bar{x}_2 + \bar{x}_3 + \ldots + \bar{x}_n$$
$$\bar{x}_1 + \bar{x}_2 + \bar{x}_3 + \ldots + \bar{x}_n = \bar{x}_1 \bar{x}_2 \bar{x}_3 \ldots \bar{x}_n.$$

Справедливость рассмотренных законов может быть доказана различными методами.

Примеры:

1) $x + y\,z = (x + y)\,(x + z)$ – распределительный закон.

Выполним доказательство от противного:

$$(x + y)\,(x + z) = x\,x + x\,y + x\,z + y\,z = x + x\,y + x\,z + y\,z =$$
$$= x\,(1 + y + z) + y\,z = x + y\,z.$$

Данному произведению соответствуют две эквивалентные схемы (рис. 1.5).

2) $x\,(x + y) = x$ – закон поглощения.

Выполним преобразования:

$$x\,(x + y) = xx + xy = x + xy = x(1 + y) = x.$$

Это доказывает, что две разные схемы на рис. 1.6 эквивалентны в смысле алгебры логики.

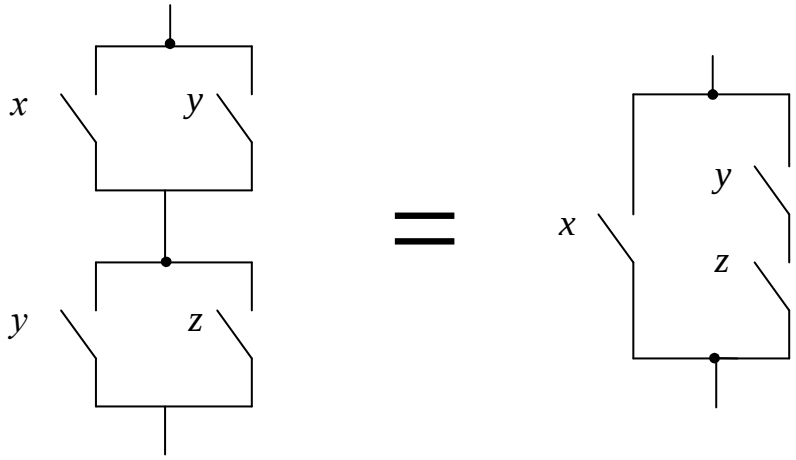


Рис. 1.5. К примеру 1

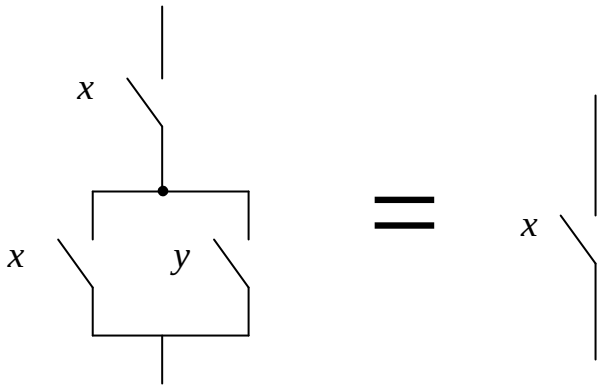


Рис. 1.6. К примеру 2

3) $x + \bar{x}y = x + y$ – закон поглощения.

Обозначим $\bar{x} = z$. Тогда

$$x + \bar{x}y = x + yz = (x + y)(x + z) = (x + y)(x + \bar{x}) = x + y.$$

Данному уравнению соответствуют контактные схемы на рис. 1.7. Раздел математики, занимающийся изучением законов алгебры логики, называется *булевой алгеброй*.

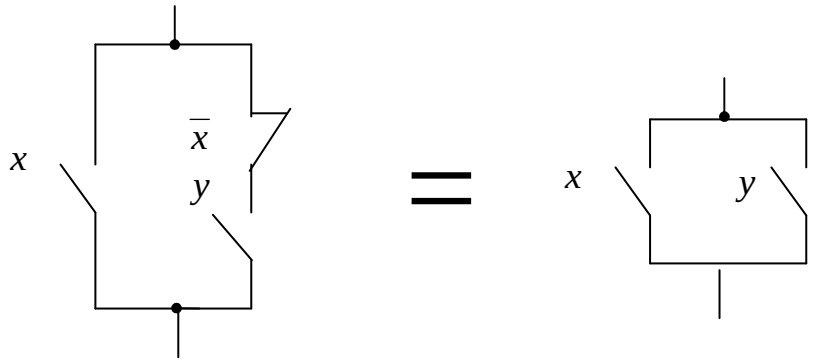


Рис. 1.7. К примеру 3

1.3. Выражение одних логических функций через другие

Для записи любого логического выражения достаточно иметь только две логические функции **НЕ**, **И** ; либо **НЕ**, **ИЛИ**.

Допустим, что схема **ИЛИ** не существует. Возможно ли любую булеву функцию реализовать только с помощью схем **И** и **НЕ** ?

Это можно проанализировать на основании теоремы Де Моргана.

$$x + y = \overline{\overline{(x + y)}} = \overline{\bar{x}\,\bar{y}}.$$

Представим полученный результат в виде функциональной схемы на бесконтактных логических элементах (рис. 1.8).

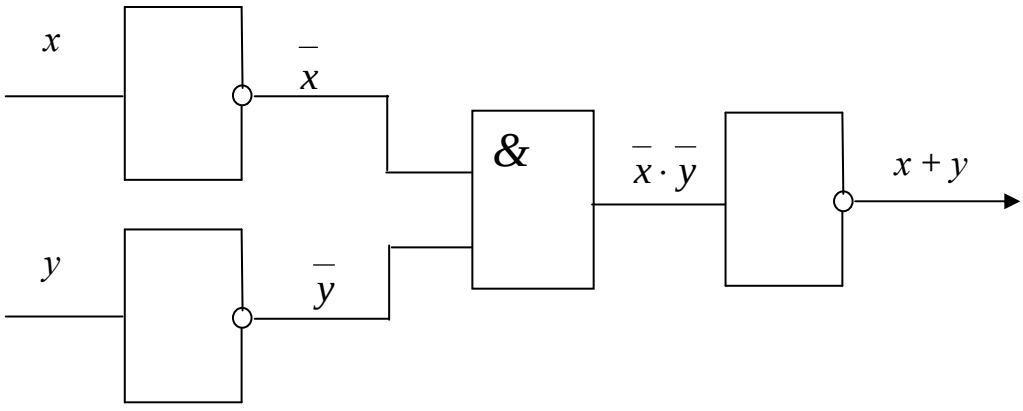


Рис. 1.8. Схема элемента **ИЛИ** на элементах **И** и **НЕ**

Для второго случая

$$x = \overline{\overline{(xy)}} = \overline{\bar{x} + \bar{y}}.$$

Результат представлен на рис. 1.9.

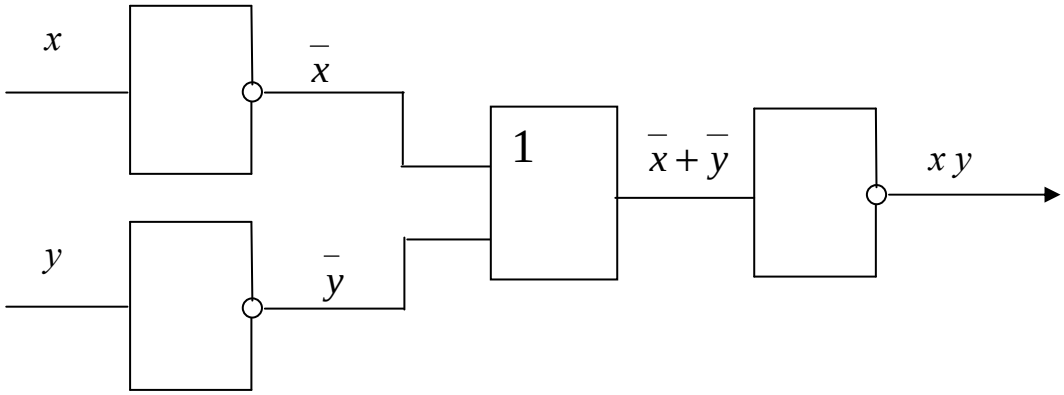


Рис. 1.9. Схема элемента **И** на элементах **И** и **ИЛИ**

2. НОРМАЛЬНЫЕ ФОРМЫ ЛОГИЧЕСКИХ ФУНКЦИЙ

2.1. Элементарные конъюнкции и дизъюнкции

Элементарной конъюнкцией называется выражение, представляющее собой конъюнкцию любого числа независимых переменных, входящих в данное выражение с инверсией или без нее не более одного раза.

Например :

$x\,y\,z$; $\bar{x}y\bar{z}$; $\bar{x}\bar{z}$; $\bar{x}$; $\bar{y}$; $z\,1$; 1 – элементарные конъюнкции.

$(x+\bar{y})z$; $xy\bar{y}z$; $\overline{xy}$ – не являются элементарными конъюнкциями.

Элементарной дизъюнкцией называется выражение, представляющее собой дизъюнкцию любого числа независимых переменных, входящих в данное выражение с инверсией или без нее не более одного раза.

Например :

$x + y + z$; $\bar{a}+\bar{z}$; $\bar{x}+\bar{y}+z$; x ; $y + 0$; 0 – элементарные дизъюнкции.

$x\,y + z$; $x + y + x$ – не элементарные дизъюнкции.

Количество переменных в элементарном выражении называется его *длиной* и определяет его *ранг*.

Например :

Конъюнкция $x\bar{y}z$ является конъюнкцией третьего ранга.

2.2. Нормальные формы дизъюнкций и конъюнкций

Дизъюнктивной нормальной формой (ДНФ) называется дизъюнкция любого числа элементарных конъюнкций.

Например :

$$x + yz + \bar{x}yz + x\bar{y}\bar{z}.$$

Конъюнктивной нормальной (КНФ) называется конъюнкция любого числа элементарных дизъюнкций.

Например :

$$x(x + y)(\bar{y} + z)(\bar{x} + y + \bar{z}).$$

Логическую функцию, заданную любым аналитическим выражением, можно непосредственно привести к конъюнктивной нормальной форме. Для этого необходимо выполнить следующее :

- избавиться от инверсий над целыми выражениями, перейдя к форме, в которой имеются инверсии только отдельных переменных;
- раскрыть скобки, применяя закон дистрибутивности;
- привести конъюнкции (дизъюнкции) к элементарным.

Пример

$$f = \overline{\{\bar{x} + [(x + z)(\bar{z} + \bar{x})]\}}(\bar{y} + \bar{z}) + \overline{(\bar{x} + z)}.$$

а) Избавиться от знаков инверсии, применяя законы Де Моргана:

$$f = [\overline{x(x + z)(\bar{z} + \bar{x})}](\bar{y} + \bar{z}) + x\bar{z} =$$
$$= x[\overline{(x + z)} + \bar{z} + \bar{x})](\bar{y} + \bar{z}) + x\bar{z} = x(\bar{x} \cdot \bar{z} + z \cdot x)(\bar{y} + \bar{z}) + x\bar{z}$$

б) Раскрыть скобки, применяя первый закон дистрибутивности

$$\boxed{x\,(y + z) = x\,y + x\,z}$$

$$f = (x\bar{x}\bar{z} + xzx)(\bar{y} + \bar{z}) + x\bar{z} == x\bar{x}\bar{z}\bar{y} + xzx\bar{y} + x\bar{x}\bar{z}\bar{z} + xzx\bar{z} + x\bar{z};$$

в) Привести конъюнкции к элементарным

$$f = 0\bar{z}\bar{y} + xz\bar{y} + 0\bar{z} + x0 + x\bar{z} = x\bar{y}z + x\bar{z}.$$

Чтобы привести данную функцию к конъюнктивной нормальной форме, нужно применить несколько раз второй дистрибутивный закон

$$x + y\,z = (x + y)\,(x + z)$$

Тогда :

$$f = x\bar{y}z + x\bar{z} = (x\bar{y}z + x)(x\bar{y}z + \bar{z}) = (x + x)(x + \bar{y}z)(\bar{z} + x)(\bar{z} + \bar{y}z) =$$
$$= (x + x)(x + \bar{y})(x + z)(\bar{z} + x)(\bar{z} + \bar{y})(\bar{z} + z) = x(x + \bar{y})(x + z)(x + \bar{z})(\bar{y} + \bar{z})$$

2.3. Конституенты единицы и нуля

Элементарные конъюнкции (дизъюнкции) называются конституентами единицы (нуля), если они содержат все независимые переменные функции.

Например :

Для функции $f\ (x_1,\ x_2,\ x_3,\ x_4)$ элементарные конъюнкции $\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$; $\bar{x}_1x_2\bar{x}_3x_4$ являются конституентами единицы, а элементарные дизъюнкции $(x_1 + x_2 + \bar{x}_3 + x_4)$; $(\bar{x}_1 + \bar{x}_2 + x_3 + x_4)$ являются конституентами нуля.

Для функции n переменных конституенты единицы имеют вид $\tilde{x}_1\,\tilde{x}_2 \ldots \tilde{x}_n$, а конституенты нуля $\tilde{x}_1 + \tilde{x}_2 + \ldots + \tilde{x}_n$.

Конституента единицы принимает единичное значение тогда и только тогда, когда все буквы принимают единичные значения (это происходит только на одном наборе).

Например :

Конституенте единицы $x_1\bar{x}_2x_3\bar{x}_4$ соответствует набор **1010**, при котором она принимает единичное значение; на всех же остальных наборах – **0**.

Конституента нуля принимает нулевое значение тогда и только тогда, когда все входящие в конституенту буквы принимают нулевые значения. Из этого следует, что конституенте нуля соответствует только один набор, на котором она принимает значение **0**.

Например :

Конституенте нуля $\bar{x}_1 + x_2 + \bar{x}_3 + \bar{x}_4$ соответствует набор **1011**, при котором она принимает нулевое значение, на всех же остальных наборах данная конституента принимает значение **1**.

2.4. Совершенные дизъюнктивные и конъюнктивные нормальные формы

Дизъюнктивная (конъюнктивная) нормальная форма называется совершенной, если все её элементарные конъюнкции (дизъюнкции) являются конституентами единицы (соответственно нуля).

Любая логическая функция имеет одну и только одну совершенную дизъюнктивную нормальную форму (СДНФ) и одну и только одну совершенную конъюнктивную нормальную форму (СКНФ).

Например :

Функцию $f = x + y$ можно записать в СДНФ в виде

$$f = \bar{x}y + x\bar{y} + xy.$$

От любой нормальной формы можно перейти к совершенной нормальной форме при помощи равносильных преобразований. Такой переход называется *развертыванием*.

Для этого необходимо :

- ввести недостающие переменные в каждое произведение умножением его на равносильность вида $x + \bar{x} = 1$, где x – недостающая переменная;
- раскрыть скобки, применяя коммутативный закон ($xy = yx$);
- избавиться от повторяющихся произведений на основании закона тавтологии ($x + x = x$).

Пример

Рассмотрим нормальную форму:

$$f = x\bar{y}z + x\bar{z}.$$

Преобразуем её:

$$f = x\bar{y}z + x\bar{z}(y + \bar{y}) = x\bar{y}z + xy\bar{z} + x\bar{y}\bar{z}.$$

Аналогично можно перейти и к совершенной конъюнктивной нормальной форме.

Совершенные нормальные формы обладают следующими особенностями:

1) Если при каком-либо наборе значений переменных функция равна единице, то в СДНФ только один из её членов принимает единичное значение.

2) Если функция равна нулю, то в СКНФ только один из её членов принимает нулевое значение.

3. МИНИМИЗАЦИЯ ЛОГИЧЕСКИХ ФУНКЦИЙ

3.1. Метод непосредственного упрощения

Тот факт, что одна и та же функция может быть выражена в различных формах, позволяет утверждать, что одна из этих форм окажется минимальной.

При проектировании системы наибольший интерес представляет та форма функции, на которую расходуется минимальное количество элементов.

Примеры

1) В некоторой схеме управления имеются три входа a, b и c. Сигнал на выходе должен появиться, когда имеются сигналы на двух любых или на всех трех входах (рис. 3.1).

Согласно приведенным условиям функция будет иметь следующий вид:

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

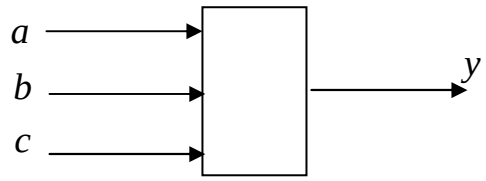


Рис. 3.1. К примеру 1

Прибавим два раза член abc.

$$y = ab\bar{c} + a\bar{b}c + \bar{a}bc + abc + abc + abc = ab(\bar{c} + c) + ac(\bar{b} + b) + bc(\bar{a} + a) = ab + ac + bc$$

На рис. 3.2 представлена схема, соответствующая полученному результату.

Применив к последнему выражению распределительный закон $x + yz = (x + y)(x + z)$, поищем другое решение :

$$y = ab + ac + bc = a\,(b + c) + bc = [a\,(b + c) + b]\,[a\,(b + c) + c] =$$
$$= (b + a)(b + b + c)(c + a)(c + b + c) = (a + b)(b + c)(a + c).$$

Этому результату соответствует функциональная схема, показанная на рис. 3.3.

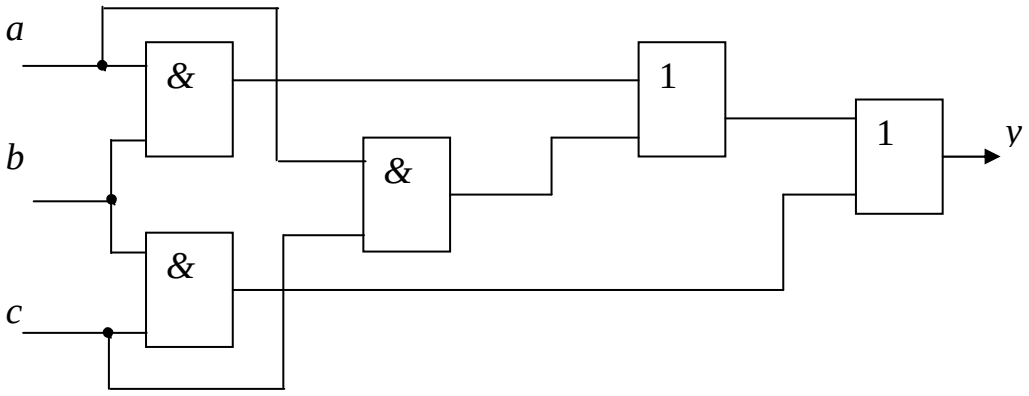


Рис. 3.2. К примеру 1

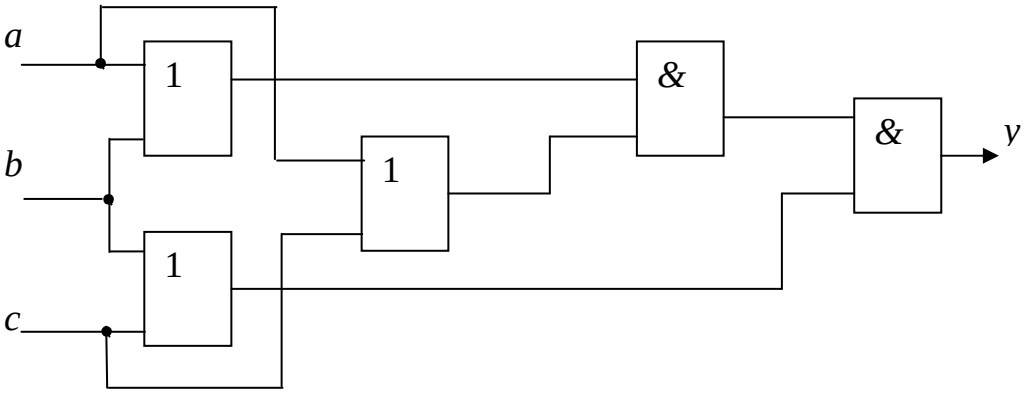


Рис. 3.3. К примеру 1

2) Упростить функцию

$$y = a\bar{b} + c + \bar{a}\bar{c}d + b\bar{c}d\,.$$

Применим к трем последним членам закон поглощения

$$x + \bar{x}y = x + y;$$

$$y = a\bar{b} + c + \bar{a}d + b\bar{c}d = a\bar{b} + c + \bar{a}d + bd = a\bar{b} + c + d\overline{\overline{\bar{a} + b}} = a\bar{b} + c + d\,\bar{\bar{a}}\bar{\bar{b}}\,.$$

К первому и последнему членам снова применим закон поглощения:

$$y = a\bar{b} + c + d\,.$$

Результат представлен на рис. 3.4.

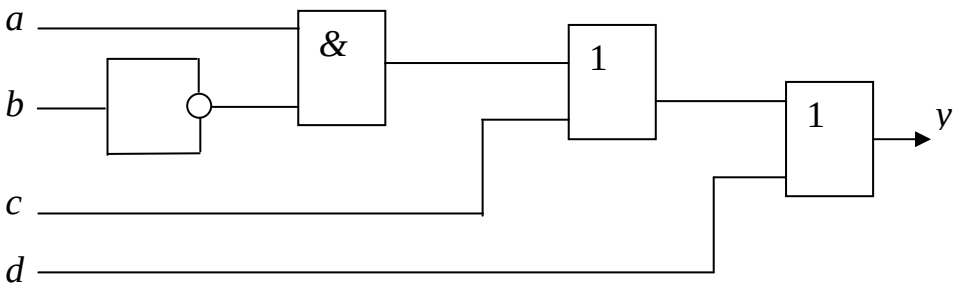


Рис. 3.4. К примеру 2

3.2. Метод Карно

Метод основан на применении карт Карно (рис. 3.5).

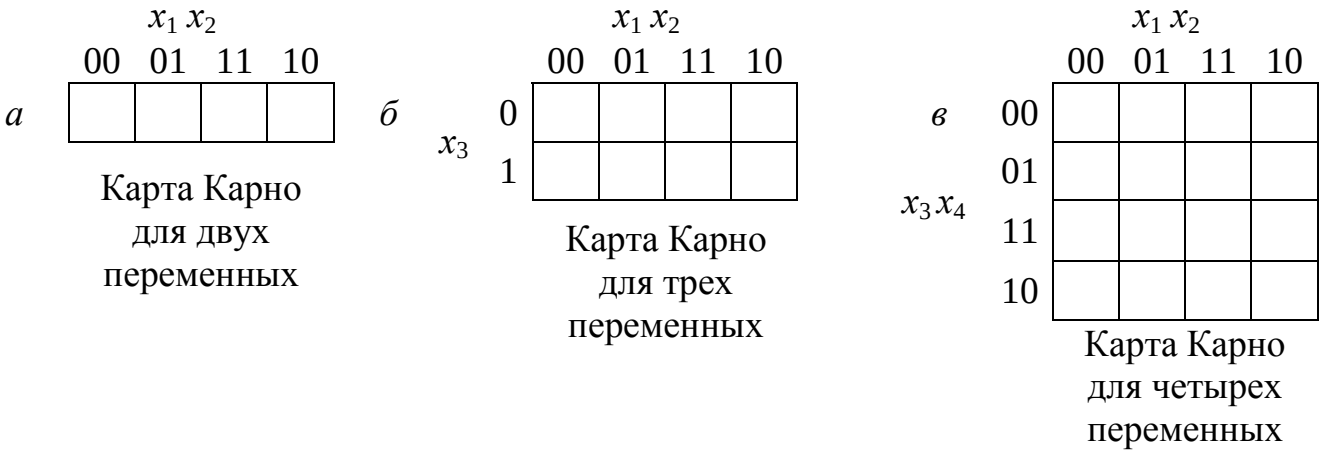


Рис. 3.5. Карты Карно

Для упрощения обозначений строки и столбцы, содержащие некоторую переменную, равную **1**, обозначим скобкой, так что значение **0** эта переменная будет иметь в неотмеченных местах (рис. 3.6).

Соседние (по строке или столбцу) клетки отличаются значением только одной переменной. Клетки на противоположных концах карты тоже являются соседними. При этом можно полагать, что карта размещена на торе.

Чтобы представить функцию на карте, достаточно в те клетки карты, где функция имеет значение **1**, поместить единицы (рис. 3.7).

Клетки, в которых записаны **1**, называют *конституентами единицы* функции или просто конституентами.

Две соседние конституенты склеиваются и образуют *простую импликанту* (рис. 3.8).

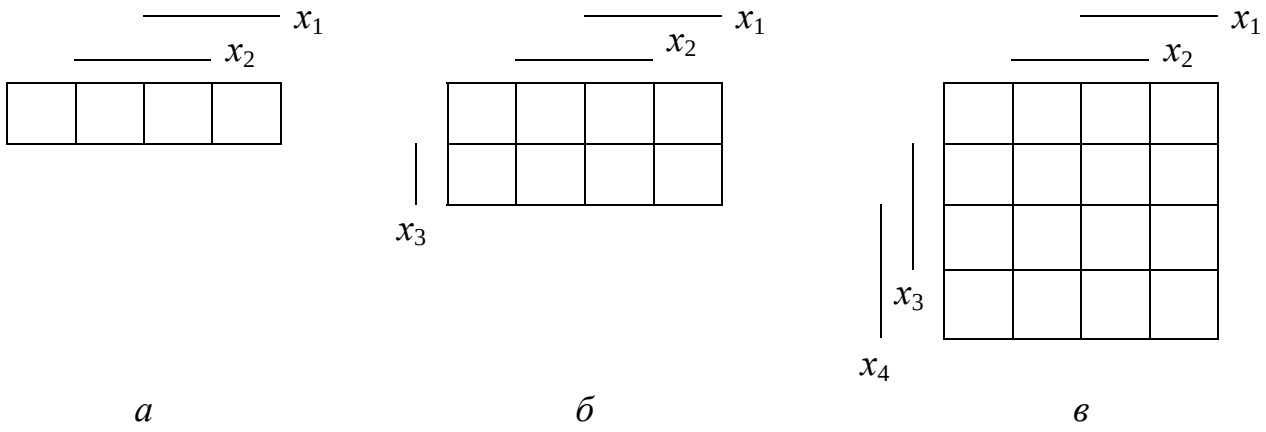
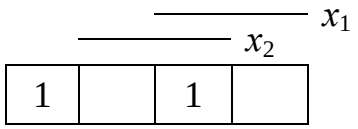
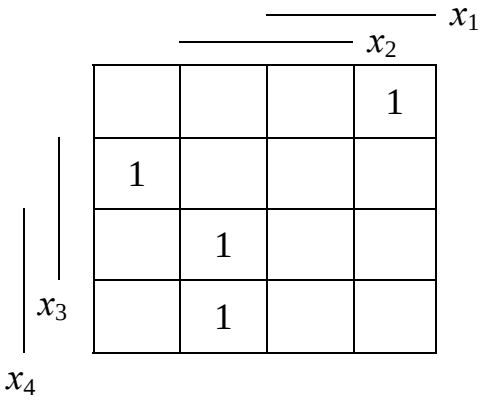


Рис. 3.6. Карты Карно (второй вариант)

а) $F = \overline{x_1}\,\overline{x_2} + x_1 x_2$



б) $F = \overline{x_1}\,\overline{x_2} x_3 \overline{x_4} + \overline{x_1} x_2 \overline{x_3} x_4 + \overline{x_1} x_2 x_3 x_4 + x_1 \overline{x_2} \overline{x_3} \overline{x_4}$



в) $F = x_1 \overline{x_2}\,\overline{x_3} + \overline{x_1}\,\overline{x_2} x_3 + x_1 x_2 x_3 + \overline{x_1} x_2 \overline{x_3}$

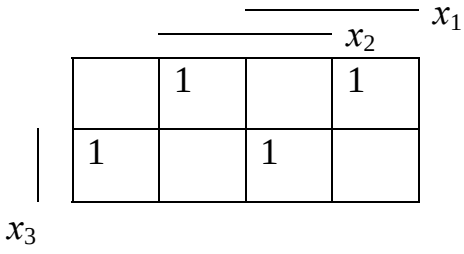


Рис. 3.7. Представление логических функций на картах Карно

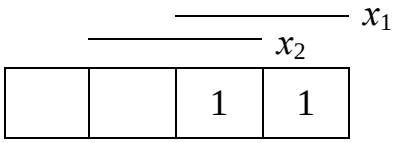


Рис. 3.8. Простая импликанта

Эта простая импликанта может быть представлена произведением, содержащим одну переменную x_1 , так как вхождение **1** в этой функции образуют произведения x_1x_2 и $x_1\overline{x_2}$.

Таким образом, склеивание

$$x_1x_2 + x_1\overline{x_2} = x_1(x_2 + \overline{x_2}) = x_1 \cdot 1 = x_1$$

мы выполнили графически (рис. 3.8).

Переменная, отсутствующая в произведении, имеет различные значения для двух конституент соответствующей импликанты (рис. 3.9).

$$F = \overline{x_1}x_2\overline{x_3}x_4 + \overline{x_1}x_2x_3x_4.$$

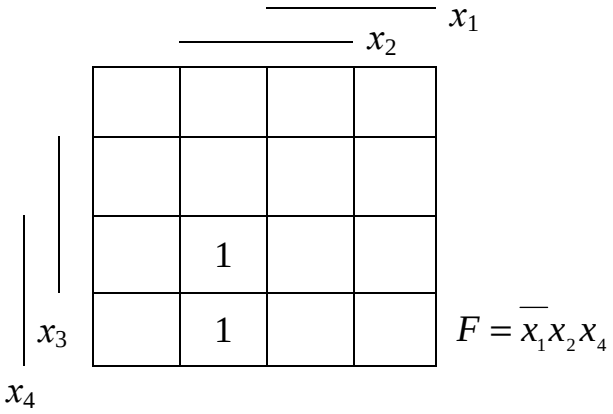


Рис. 3.9. Простая импликанта для карты четырех переменных

В общем случае простая импликанта соответствует произведению, в котором всегда отсутствует одна переменная.

Четыре соседних конституенты образуют импликанту, которая соответствует произведению без двух переменных. Те переменные, которые не сохраняют постоянное значение на этой импликанте, опускают (рис. 3.10):

$$F = \overline{x_1}x_2\overline{x_3}x_4 + \overline{x_1}x_2x_3x_4 + x_1x_2\overline{x_3}x_4 + x_1x_2x_3x_4.$$

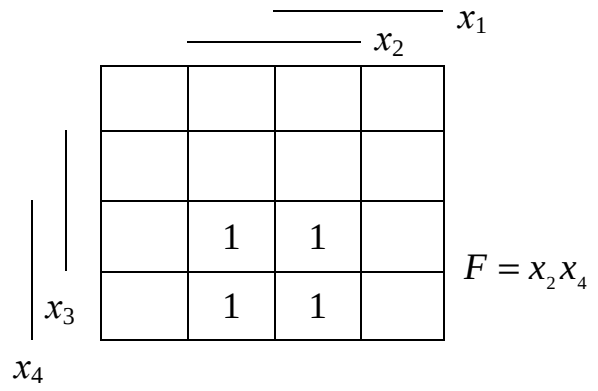
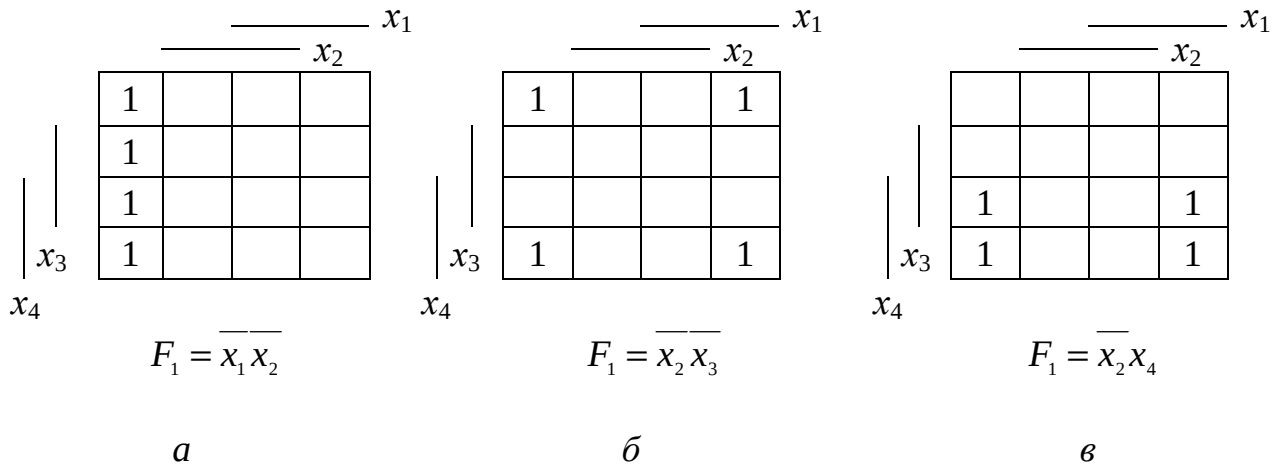


Рис. 3.10. Импликанта из четырех конституент

Аналогичные виды простых импликант представлены на рис. 3.11.



а *б* *в*

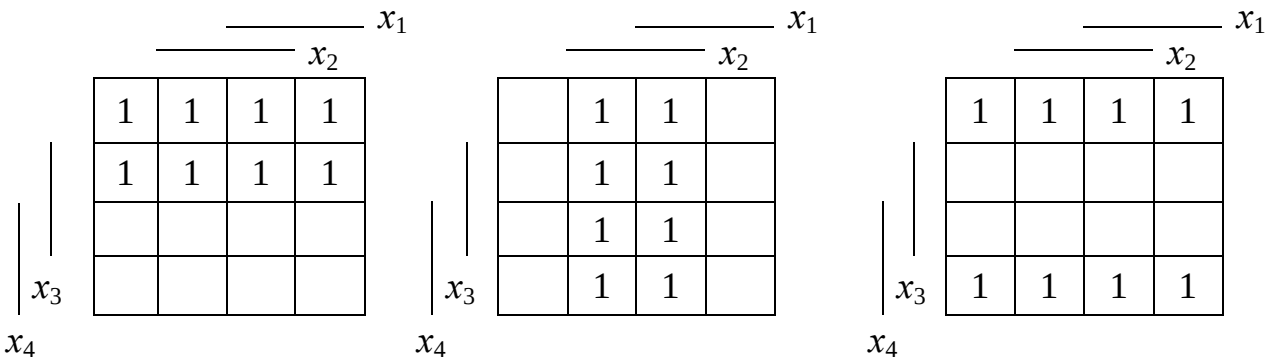
Рис. 3.11. Виды простых импликант

Другие виды простых импликант показаны на рис. 3.12.

Импликанта, полученная в результате склеивания некоторого множества конституент, *покрывает* эти конституенты.

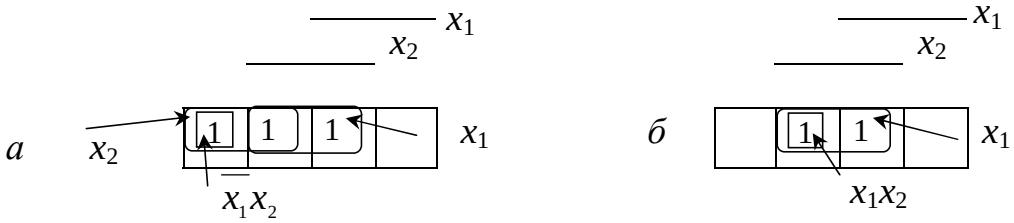
Для представления функции каждая **1** должна быть использована, по крайней мере, в одной импликанте и ни в какой импликанте не должна быть использована ни одна пустая клетка. В этом случае говорят, что все единицы карты должны быть покрыты простыми импликантами (рис. 3.13).

Если выбраны самые большие импликанты и использовано по возможности меньшее число импликант, то будет получена самая простая дизъюнктивная нормальная форма (рис. 3.14).



$F_1 = \overline{x_4}$ *а* $F_1 = x_2$ *б* $F_1 = \overline{x_3}$ *в*

Рис. 3.12. Другие виды простых импликант



$x_1 + \overline{x_1}x_2 = x_1 + x_2$ $x_1 + x_1x_2 = x_1$

Рис. 3.13. Варианты покрытия единиц простыми импликантами

Дальнейшее упрощение возможно за счет вынесения общих множителей за скобки.

Поучительный пример.

Минимизировать функцию

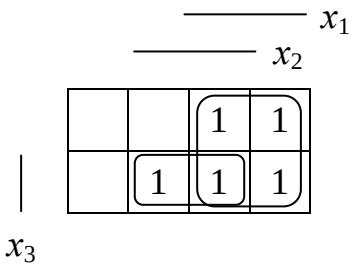
$$F_1 = ab + \overline{a}c + bc.$$

Следует сначала привести функцию к совершенной дизъюнктивной нормальной форме (СДНФ).

$$F = ab + \overline{a}c + bc = ab(c + \overline{c}) + \overline{a}c(b + \overline{b}) + bc(a + \overline{a}) =$$
$$= abc + ab\overline{c} + \overline{a}bc + \overline{a}\overline{b}c + abc + \overline{a}bc = abc + ab\overline{c} + \overline{a}bc + \overline{a}\overline{b}c$$

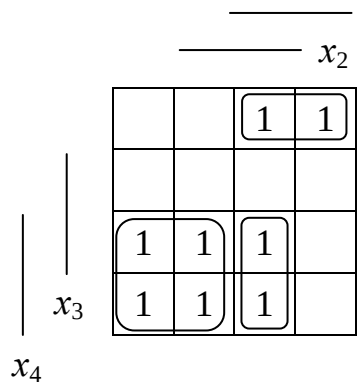
Теперь можно минимизировать логическую функцию обычным способом (рис. 3.15).

а



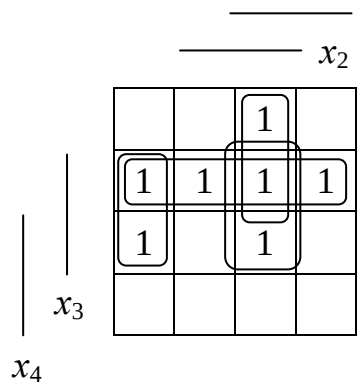
$F = x_1 + x_3 x_2$

б



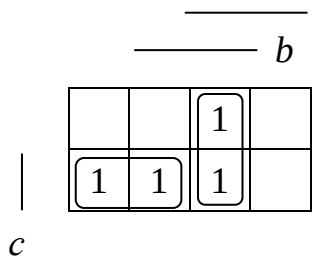
$F = \overline{x_1} x_3 + x_1 \overline{x_3 x_4} + x_1 x_2 x_4$

в



$F = x_3 \overline{x_4} + x_1 x_2 \overline{x_4} + x_1 x_2 x_3 + \overline{x_1 x_2} x_3$

Рис. 3.14. Минимизация логических функций в дизъюнктивной нормальной форме



$F = ab + \overline{a}c$

Рис. 3.15

Принимая во внимание клетки карт Карно, не содержащие единиц, и поступая с ними так же, как мы поступали с клетками с единицами, можно получить двойственную форму, т.е. конъюнктивную нормальную форму (рис. 3.16).

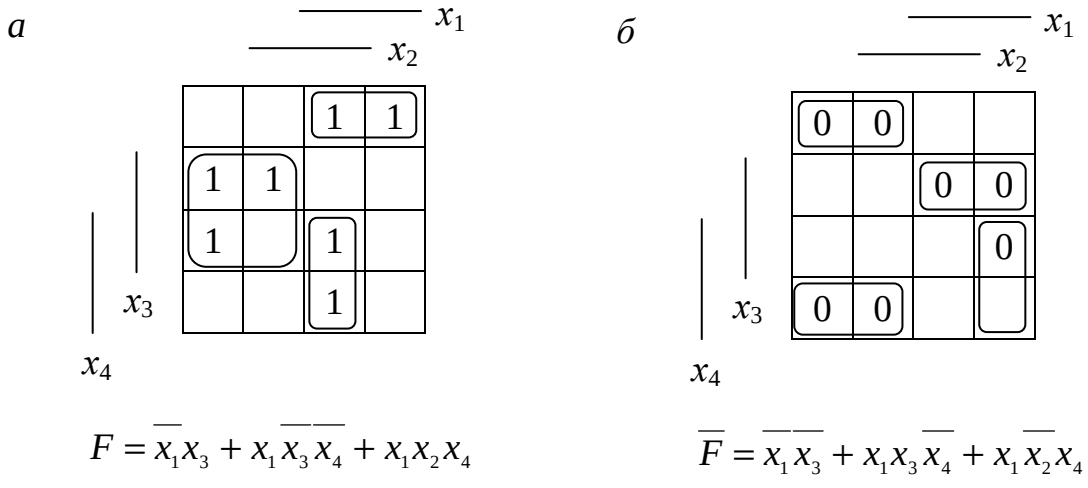


Рис. 3.16. Получение конъюнктивной нормальной формы

Часто при определении работы системы при некоторой комбинации значений входных переменных безразлично, какое будет значение выхода. Такие комбинации называют *безразличными состояниями*. На карте Карно их можно помечать знаком ~ (тильда).

Для увеличения размеров простых импликант в любые клетки, имеющие знак ~ , можно условно помещать единицы (рис. 3.17).

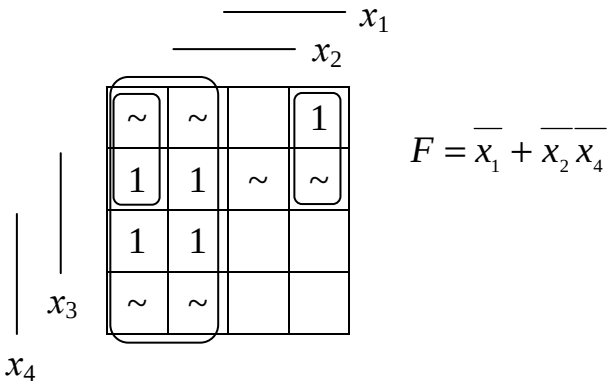


Рис. 3.17. Минимизация логической функции с использованием безразличных состояний

В большинстве реальных систем число безразличных состояний значительно превышает число обязательных и запрещенных состояний. Чтобы не затенять карту большим количеством знаков ~ , мы в дальнейшем

будем обозначать обязательные состояния знаком **1** , запрещенные – знаком **0**, а для безразличных состояний оставлять пустые клетки (рис. 3.18).

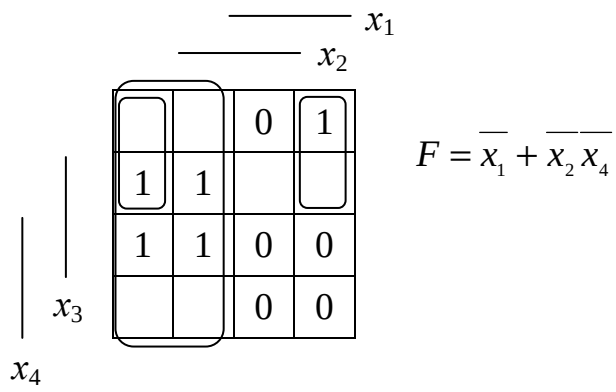


Рис. 3.18. Обозначение безразличных состояний в виде пустых клеток

Для минимизации логических функций с большим количеством $(7 \div 8)$ переменных требуются достаточно обширные карты Карно, что усложняет процедуру минимизации.

Чтобы успешно решать такие задачи, необходимы правила формального выделения простых импликант на картах Карно. Прежде чем сформулировать эти правила, рассмотрим два понятия: ось симметрии и поле симметрии карты Карно.

Ось симметрии – это любая прямая, проведенная на карте Карно параллельно одной из её сторон и касающаяся конца одной из скобок (сами стороны не являются осями симметрии) (рис. 3.19). Оси симметрии, проходящие через середину карты, называются *главными*.

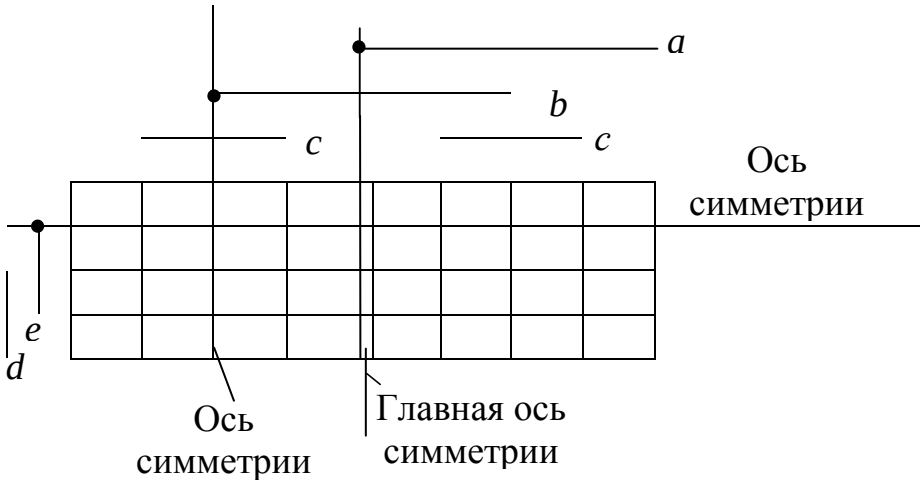


Рис. 3.19. Оси симметрии карты Карно

Поле симметрии – это все клетки, расположенные слева и справа (или сверху и снизу) от оси симметрии на расстоянии, равном 1/2 длины скобки, с которой соприкасается данная ось симметрии (рис. 3.20).

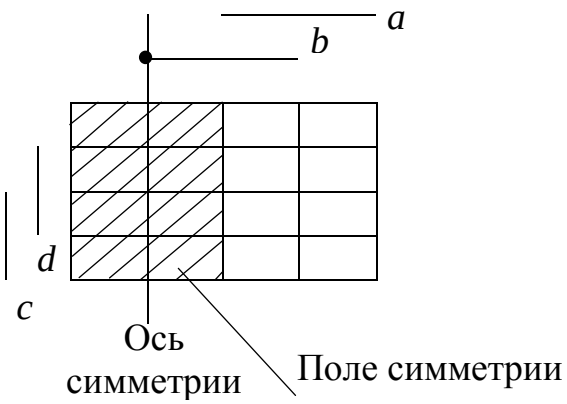


Рис. 3.20. Поле симметрии карты Карно

Замечание:

Длина наружных скобок карты при условном увеличении числа переменных возросла бы в два раза. Поэтому ширину поля симметрии для оси, касающейся наружной скобки, принимают равной не половине её длины, а всей её длине.

Теперь сформулируем правила выделения простых импликант на карте Карно:

1) Фигура простой импликанты должна иметь форму прямоугольника (с учетом замкнутости противоположных краёв карты) и содержать 2^{k1} клеток, где 2^{k1} и 2^{k2} - количество клеток, содержащихся в каждой из сторон прямоугольника ($k1$ и $k2$ – простые числа).

2) Фигура импликанты должна быть симметричной относительно её центральных осей, причем эта фигура не должна выходить за пределы поля симметрии каждой из этих осей (рис. 3.21).

3) Фигура импликанты может состоять из нескольких частей. Тогда общее количество осей симметрии для такой импликанты в одном из направлений (горизонтальном или вертикальном) определяется по формуле

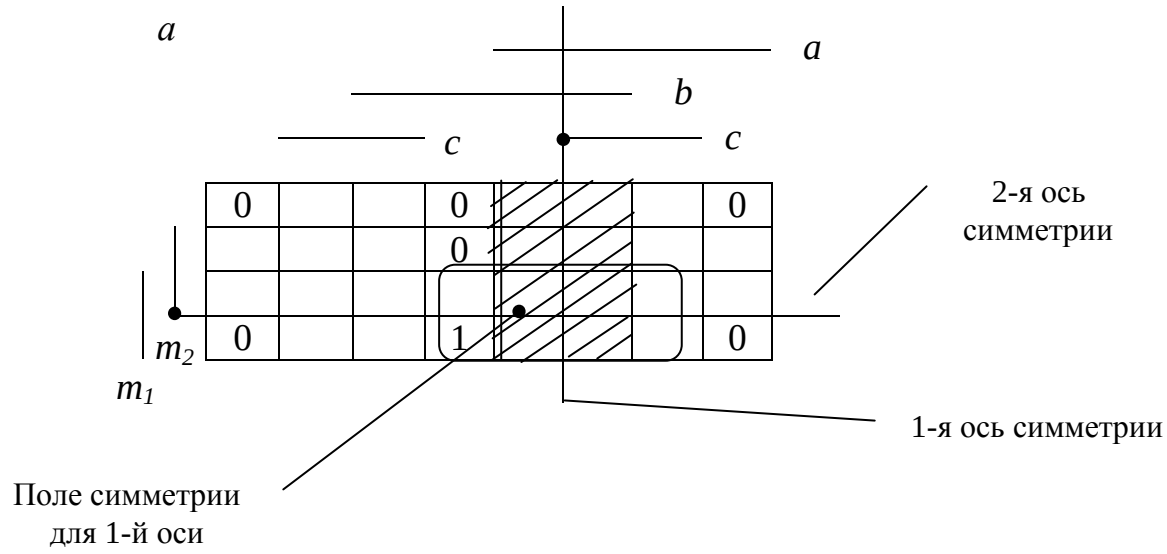
$$N = 2n - 1,$$

где n – количество частей, из которых состоит импликанта в заданном направлении;

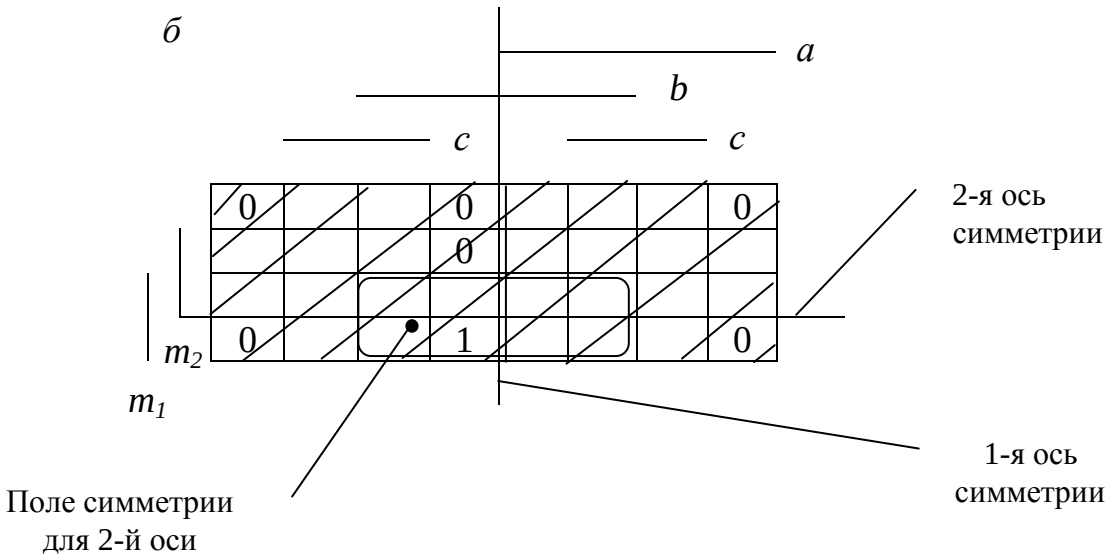
N – количество осей симметрии.

На рис. 3.22 условно показаны три варианта таких импликант.

В этом случае импликанта должна быть симметричной относительно каждой из осей симметрии в пределах её поля симметрии (рис. 3.23).



Неверно! Импликанта вышла за пределы поля симметрии.

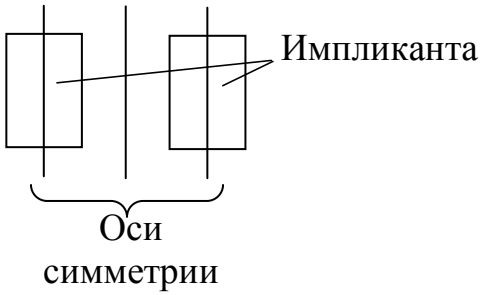


Верно! Импликанта находится внутри поля симметрии

Рис. 3.21. Условия симметрии импликанты

а) $n = 1$; $N = 1$

б) $n = 2$; $N = 3$



в) $n = 3$; $N = 5$

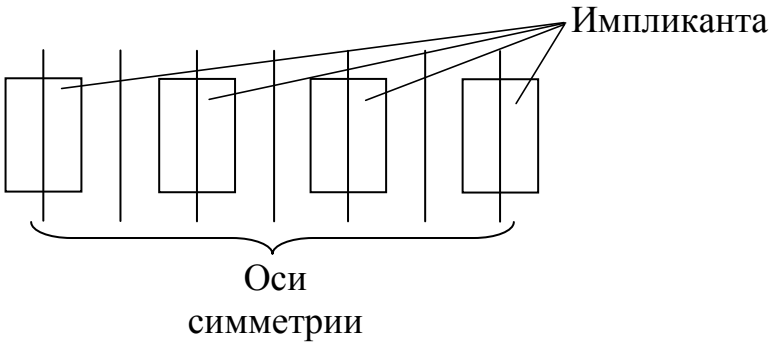
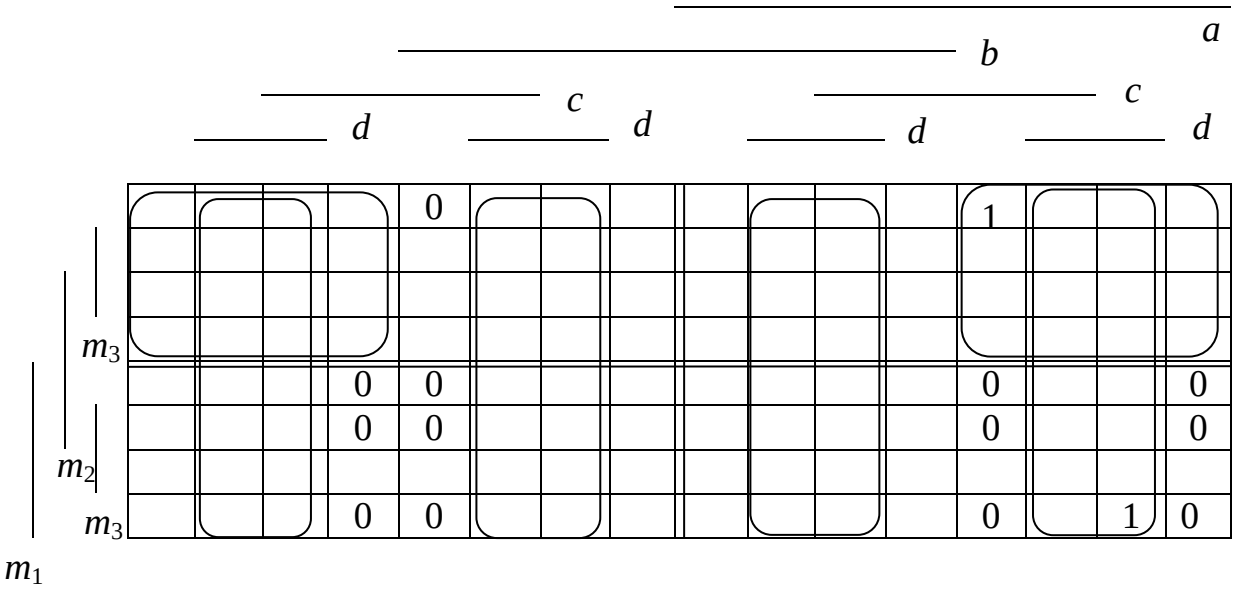


Рис. 3.22. Оси симметрии импликант



$$F = \overline{b\overline{m_1}} + d$$

Рис. 3.23. Пример импликант, состоящих из нескольких частей

4. СИНТЕЗ ОДНОТАКТНЫХ СИСТЕМ УПРАВЛЕНИЯ

4.1. Общие положения

По условиям работы дискретно-логические системы управления разделяются на *однотактные* и *многотактные*. К *однотактным* относятся системы, у которых комбинация сигналов на выходе однозначно определяется комбинацией сигналов, поступивших на вход в течение заданного промежутка времени (такта), и не зависит от комбинаций сигналов, поступивших на вход в предыдущие промежутки времени.

Выходные сигналы появляются с некоторой задержкой во времени после подачи очередной комбинации входных сигналов. Эта задержка происходит из-за инерционности при срабатывании отдельных элементов системы. Однако при синтезе системы время срабатывания элементов не учитывается, так как предполагается, что оно значительно меньше интервалов времени между подачами комбинаций сигналов на входе.

К *многотактным* системам относятся системы, у которых комбинация выходных сигналов определяется не только состояниями входов в заданный момент времени, но и значениями входных сигналов в предыдущих тактах.

Однотактную систему можно рассматривать как комбинационную схему (логическое устройство), реализующую логическое соотношение между входами и выходами. Поэтому процесс построения такой системы принято называть *логическим синтезом*.

4.2. Примеры синтеза однотактных систем управления

Рассмотрим систему управления четырьмя пневматическими цилиндрами (рис. 4.1), в которой в зависимости от комбинации состояний двух переключателей A_1 и A_2 на входе выдвигается шток одного из цилиндров: 1-го – при обоих открытых переключателях A_1 и A_2; 2-го – при открытом A_2 и закрытом A_1; 3-го – при открытом A_1 и закрытом A_2; 4-го – при обоих закрытых переключателях A_1 и A_2.

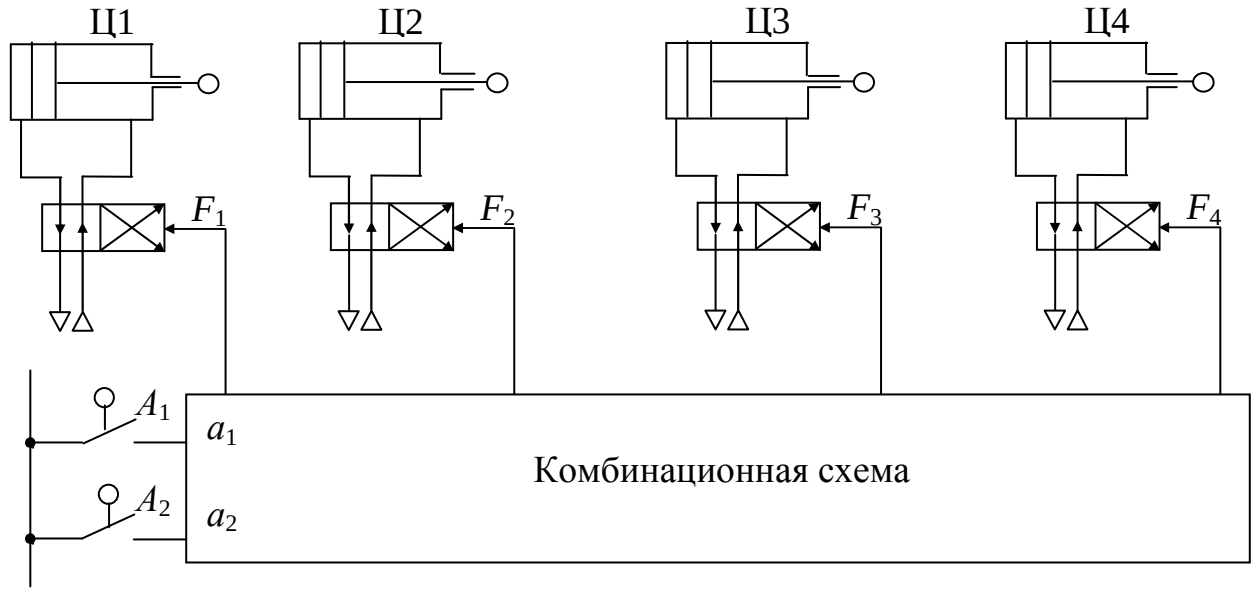


Рис. 4.1. Схема управления четырьмя исполнительными устройствами

На схеме переключатели обозначены большими буквами A_1 и A_2, а сигналы с этих переключателей – малыми буквами a_1 и a_2.

Подобная система используется в контрольно-сортировочных автоматах, где отбраковка производится по двум каким-либо параметрам, например, по размеру наружного диаметра и форме его цилиндрической поверхности (эллипс, огранка).

Работает система следующим образом. От переключателей, служащих в данном случае конечными звеньями размерных датчиков, сигналы поступают через систему управления к распределителям пневматических цилиндров. В зависимости от комбинации входных сигналов переключается один из распределителей, в результате чего поршень соответствующего цилиндра совершает ход вперед. Шток каждого поршня связан с заслонкой бункера. Если заслонка бункера открыта, то изделие после измерения попадет в один из четырех бункеров. Таким образом, в первом бункере будут собраны изделия, забракованные по двум параметрам, во втором и третьем – по одному параметру, а в четвертом бункере будут находиться годные изделия.

Заданные выше условия работы представим в виде таблицы состояний (табл. 4.1) в первом столбце которой отмечено условие срабатывания цилиндра **1** ($a_1 = 1$, $a_2 = 1$), во втором столбце – цилиндра **2** ($a_1 = 0$, $a_2 = 1$) и т. д. Команды F_1, F_2, F_3 и F_4 на включение цилиндров поступают на соответствующие распределители.

Таблица 4.1

Сигнал	Номер состояния			
	1	2	3	4
a_1	1	0	1	0
a_2	1	1	0	0
F_1	1	0	0	0
F_2	0	1	0	0
F_3	0	0	1	0
F_4	0	0	0	1

В рассматриваемом примере использованы все возможные состояния (сочетания) входных сигналов, и каждому из них соответствует команда на ход вперед поршня одного из устройств. Запишем логические функции для выходных сигналов системы управления:

$$F_1 = a_1 a_2; \quad F_2 = \overline{a_1} a_2; \quad F_3 = a_1 \overline{a_2}; \quad F_4 = \overline{a_1}\,\overline{a_2}.$$

Этим функциям соответствует комбинационная схема, показанная на рис. 4.2.

В качестве второго примера рассмотрим систему управления одним исполнительным устройством, поршень которого должен перемещаться вперед при срабатывании не менее двух из трех переключателей A_1, A_2 и A_3 на входе системы (рис. 4.3).

Обязательные и запрещенные состояния для этого случая сгруппированы в табл. 4.2.

Таблица 4.2

Сигнал	Номер состояния							
	1	2	3	4	5	6	7	8
a_1	1	1	0	1	0	1	0	0
a_2	1	0	1	1	0	0	1	0
a_3	0	1	1	1	0	0	0	1
F_1	1	1	1	1	0	0	0	0

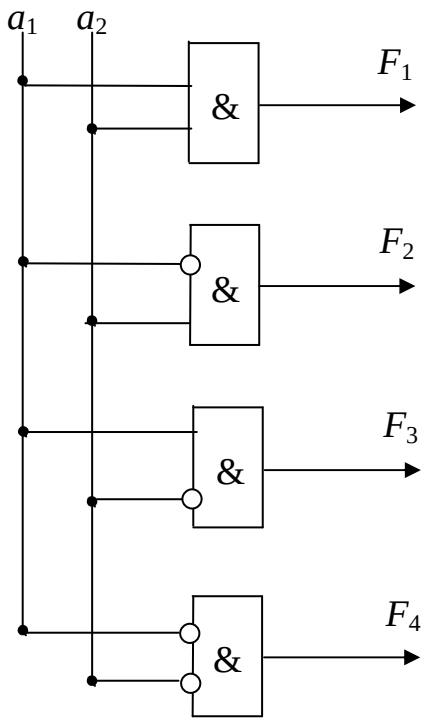


Рис. 4.2. Комбинационная схема автомата-сортировщика

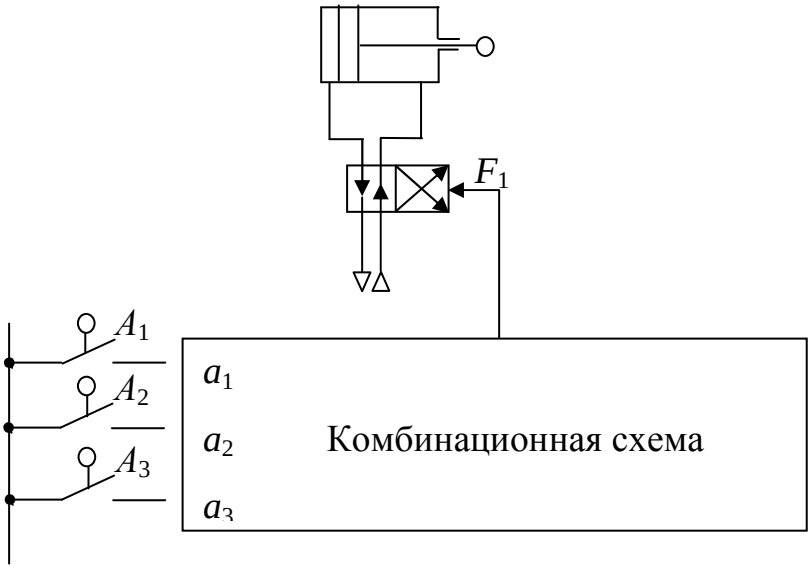


Рис. 4.3. Схема управления одним цилиндром

Согласно табл. 4.2 сигнал F_1 на выходе системы равен единице при комбинациях сигналов на входе 110, 101, 011 и 111. В остальных случаях сигнал F_1 равен нулю. Неиспользованных состояний здесь нет.

Отразим состояния системы управления на карте Карно (рис. 4.4).

$$F_1 = a_1 a_2 \overline{a_3} + a_1 \overline{a_2} a_3 + \overline{a_1} a_2 a_3 + a_1 a_2 a_3;$$

$$\overline{F_1} = \overline{a_1}\,\overline{a_2}\,\overline{a_3} + a_1 \overline{a_2}\,\overline{a_3} + \overline{a_1} a_2 \overline{a_3} + \overline{a_1}\,\overline{a_2} a_3 .$$

После минимизации получаем для логической функции F_1 следующее выражение:

$$F_1 = a_1 a_2 + a_1 a_3 + a_2 a_3 .$$

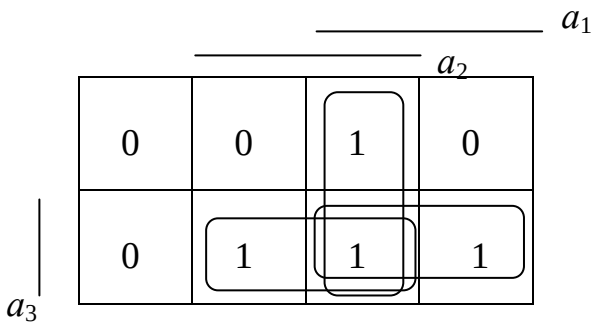


Рис. 4.4. Минимизация логической функции с помощью карты Карно

Комбинационная схема дискретного автомата представлена на рис. 4.5.

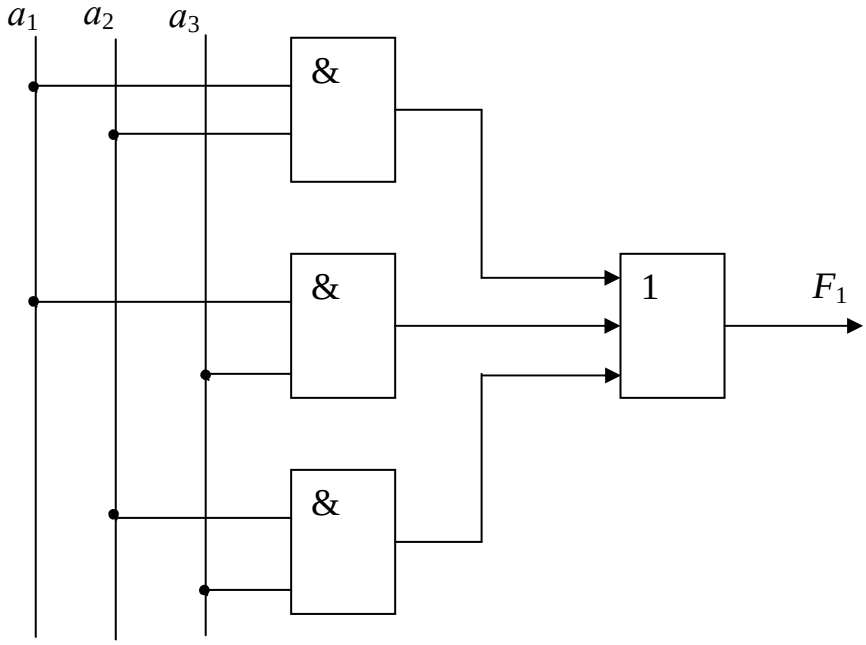


Рис. 4.5. Комбинационная схема дискретного автомата

Методика упрощения (минимизации) логических функций в буквенных выражениях с ростом числа входных переменных становится громоздкой и неудобной для использования. Поэтому целесообразнее принять методы, рассматриваемые в главе 5.

5. СИНТЕЗ МНОГОТАКТНЫХ СИСТЕМ УПРАВЛЕНИЯ

5.1. Общие сведения

В отличие от однотактных систем в многотактных системах сигналы на выходе определяются не только сигналами на входе в данный момент, но и ранее поступившими входными сигналами, т.е. поведение многотактной системы определяется последовательностями сигналов, поступающих на вход [5].

Многотактная система отличается от однотактной тем, что при одинаковых воздействиях на входе на её выходе могут быть разные сигналы. Чтобы реализовать такое свойство, многотактная система управления должна обладать памятью – способностью запоминать происшедшие ранее события.

В дискретных автоматах запоминание осуществляется с помощью *элементов памяти*.

При переходе из одного устойчивого состояния в другое элемент памяти должен сохранять это состояние до тех пор, пока новый сигнал не выведет его из этого состояния. В качестве элементов памяти широко применяются различные триггеры. Элементы памяти условно изображают прямоугольником с двумя входами: включающим S (*Set*) и выключающим R (*Reset*). При подаче сигнала на вход R (выключающий) происходит стирание единичного сигнала на выходе.

5.2. Основные сведения по общей теории дискретных автоматов

Для изучения релейных устройств с единой точки зрения их непосредственное изучение заменяют анализом абстрактной модели, называемой *дискретным автоматом* (рис. 5.1).

Часть дискретного автомата, в которой сосредоточены логические элементы (элементы, реализующие операции алгебры логики **И**, **ИЛИ**, **НЕ**), образующие однотактную комбинационную схему, называют *логическим преобразователем* (ЛП).

К входу логического преобразователя подключены *входные элементы* $A, B, C, \ldots$, которые контролируют работу объекта управления и, соответственно, вырабатывают *входные переменные* $a, b, c, \ldots$, являющиеся независимыми аргументами, а к выходу логического преобразователя присоединены *выходные элементы* X, Y, Z, включение которых определяется функциями F_X, F_Y, F_Z, являющимися *выходными переменными*.

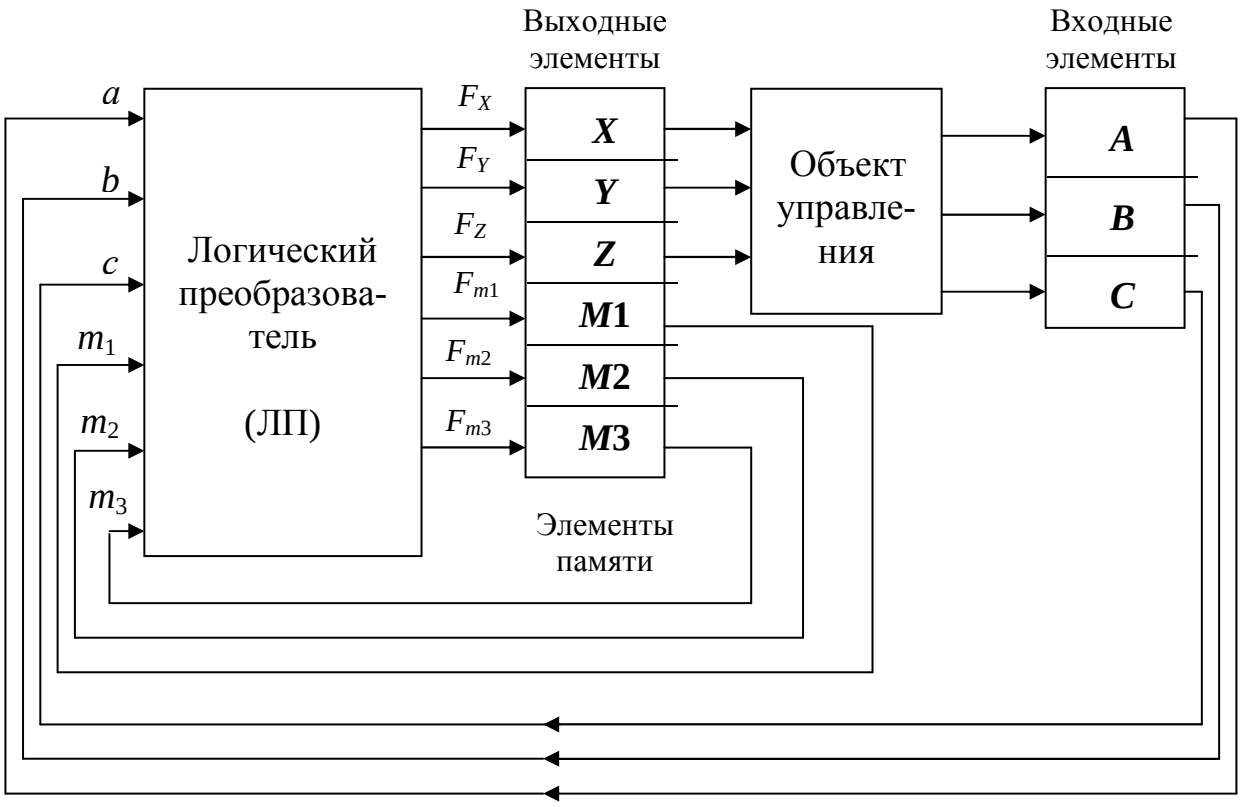


Рис. 5.1. Структурная схема дискретного автомата

Переменные m_1 , m_2 , m_3 , определяемые состояниями элементов памяти M_1 , M_2 , M_3 , называются *внутренними переменными*. Они действуют на вход логического преобразователя совместно с входными переменными и влияют на работу выходных элементов и элементов памяти.

Благодаря элементам памяти M_1 , M_2 , M_3 , в рассматриваемом дискретном автомате сигналы на выходе определяются не только сигналами на входе в данный момент, но и ранее поступившими входными сигналами. Таким образом, при одинаковых воздействиях на входе на выходе дискретного автомата могут быть разные сигналы.

Совокупность состояний элементов памяти называют *внутренним состоянием* дискретного автомата. Совокупность *состояний входа* $\{a , b , c ,\ldots\}$ и внутренних состояний $\{m_1, m_2 , m_3 ,\ldots\}$ называется *полным состоянием* или просто *состоянием* дискретного автомата. Автомат с n входами и S элементами памяти может находиться в 2^{n+S} состояниях.

Период, в течение которого состояние автомата не меняется, называется *тактом*. Длительность всех тактов принимается одинаковой, хотя реальное время работы элементов, связанных с рабочими органами машины, естественно, может быть различным. Такт – условная единица

времени работы дискретного автомата. Каждому такту приписывается порядковый номер.

Переход от одного такта к следующему обусловлен переключением одного из входных элементов или элементов памяти и сопровождается изменением состояния дискретного автомата. Переходы считаются мгновенными.

Чтобы определить состояние дискретного автомата для каждого такта, входным и внутренним переменным в порядке сверху вниз приписываются веса 2^0, 2^1, 2^2,… 2^{n+S} и в каждом такте вес переменной умножают на её значение, т.е. на **0** или **1**, а полученные величины суммируют.

Найденные таким образом состояния дискретного автомата отмечают на карте Карно и с её помощью синтезируют структуру логического преобразователя. Неиспользуемые состояния дискретного автомата отмечают на карте Карно знаком «~» (тильда) или вовсе не отмечают.

5.3. Синтез систем управления по циклограммам работы механизмов

Циклограмма показывает последовательность включения рабочих органов машины.

Состояния рабочих органов в установившемся режиме изображают на циклограмме горизонтальными линиями. Для каждого рабочего органа, а также для каждого элемента памяти проводится своя линия.

Переход от одного состояния к другому происходит почти мгновенно, поэтому он на циклограмме изображается вертикальной линией.

Таким образом, вертикальные линии соответствуют моментам изменения входных сигналов. Расстояние между соседними вертикальными линиями на циклограмме есть такт работы дискретного автомата.

По циклограмме легко определить суммарные веса входных переменных в каждом такте. С этой целью входным переменным (каналам) в порядке сверху вниз приписывают веса 2^0, 2^1, 2^2,… 2^n и в каждом такте вес канала умножают на значение сигнала в нем, т.е. на **0** или на **1** , а полученные величины суммируют.

Построение циклограммы рассмотрим на примере относительно простой схемы, которая управляет двумя гидравлическими цилиндрами X и Y, обслуживающими участок автоматической линии (рис. 5.2).

Цилиндры работают в следующей последовательности:

$$X, Y, \overline{Y}, \overline{X}.$$

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.

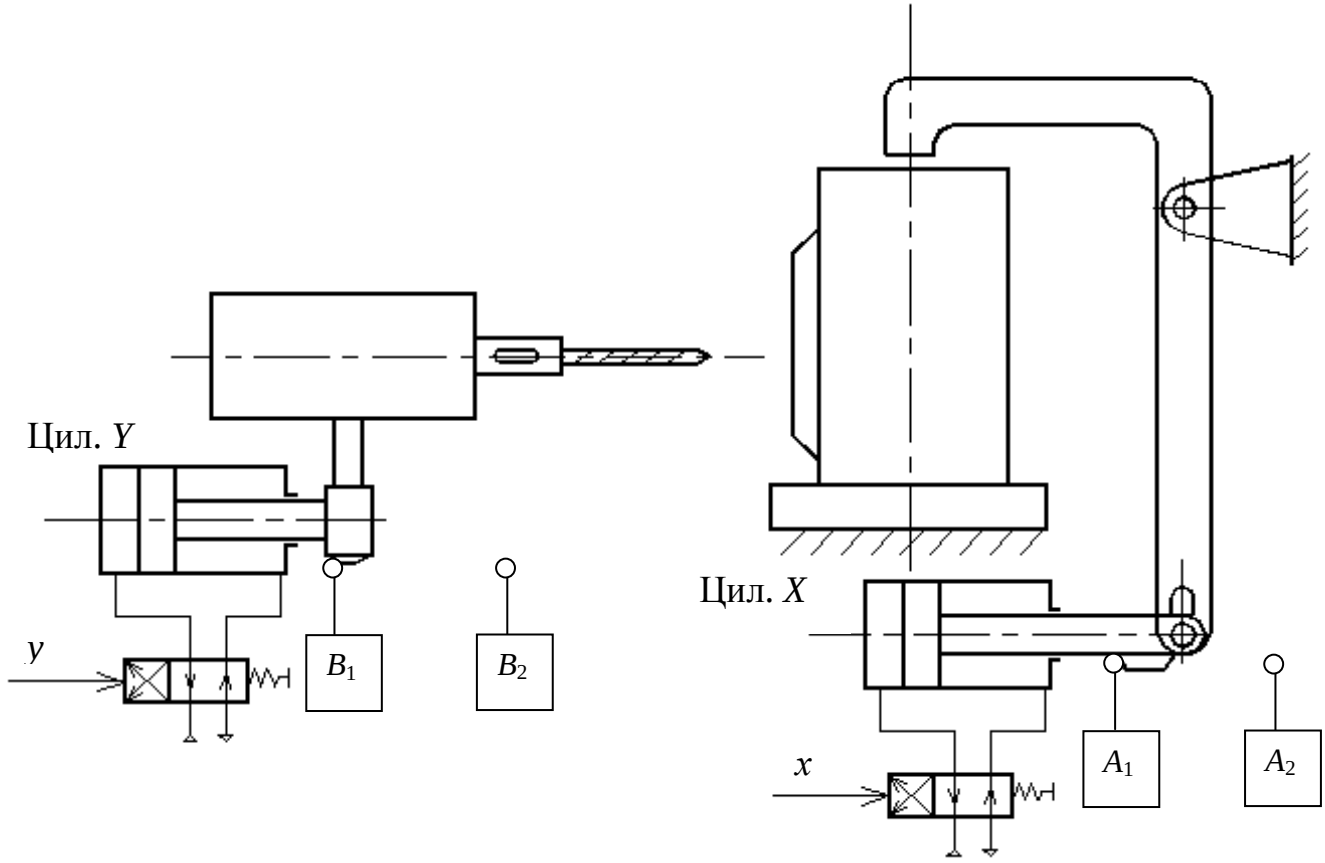


Рис. 5.2. Схема управления двумя гидроцилиндрами

Для контроля положений штоков цилиндров служат путевые переключатели A_1, A_2 и B_1, B_2. Сигналы с этих переключателей (a_1, a_2 и b_1, b_2) являются переменными дискретного автомата (рис. 5.3).

К выходу дискретного автомата подключены выходные элементы памяти (триггеры), с которых поступают выходные сигналы (переменные) x и y. Они через усилители (на схеме не показано) управляют электрогидравлическими кранами цилиндров X и Y.

Значения логических функций $F_X, F_{\bar{X}}, F_Y, F_{\bar{Y}}$ на включение и на выключение выходных элементов памяти зависят от состояний дискретного автомата.

Рассмотрим последовательность включений и выключений входных и выходных переменных в каждом такте.

В первом такте включается цилиндр X (F_X = 1) и его шток идет вперед. Через некоторое время кулачок, связанный с этим штоком, освободит

путевой переключатель A_1 и он выключится. Это событие фиксируется во втором такте. Еще через некоторый промежуток времени включится путевой переключатель A_2, что является событием для третьего такта, и т.д.

Последнее событие цикла – включение путевого переключателя A_2 – приводит дискретный автомат в исходное состояние. Поэтому это последнее событие относится к первому (исходному) такту.

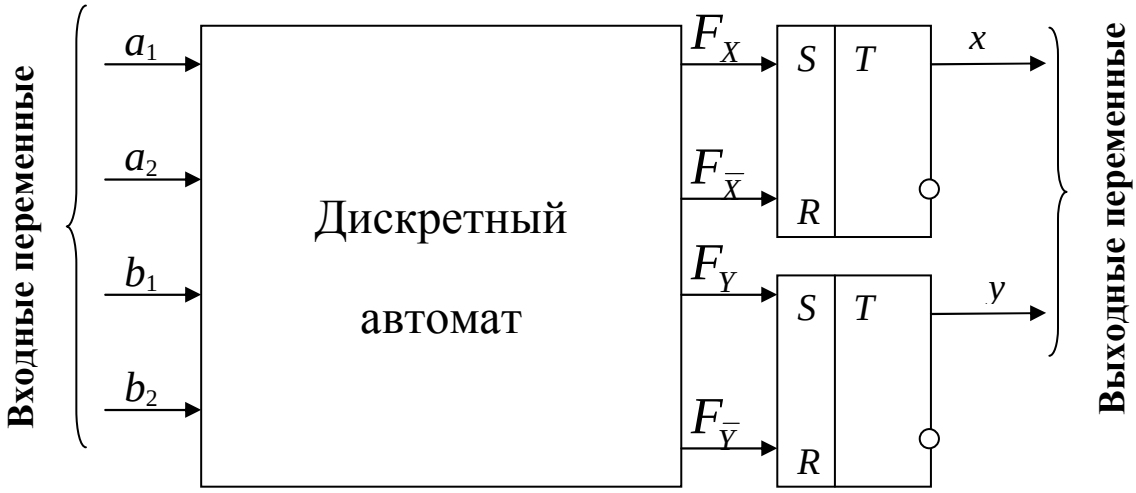


Рис. 5.3. Схема внешних связей дискретного автомата

Последовательная запись изменения состояний входных и выходных переменных называется *таблицей включений*.

Для рассматриваемого устройства таблица включений имеет следующий вид:

1. $a_1 = 1$	$F_X = 1$
2. $a_1 = 0$	
3. $a_2 = 1$	$F_Y = 1$
4. $b_1 = 0$	
5. $b_2 = 1$	$F_{\bar{Y}} = 1$
6. $b_2 = 0$	
7. $b_1 = 1$	$F_{\bar{X}} = 1$
8. $a_2 = 0$	

Правила записи таблицы включений:

1) В первом столбце отмечают изменения состояний входных, а в правом столбце – выходных переменных.

2) Для выходных переменных (правый столбец) записывают только изменения состояний с **0** на **1** (обратные изменения этих переменных с **1** на **0** не надо записывать).

3) Начинают запись алгоритма, отмечая изменение выходной переменной в первом такте (в нашем примере F_X = 1). При этом вначале строка состояния входной переменной в первом такте остается пустой. В эту строку записывают последнее изменение одной из входных переменных в конце цикла.

В соответствии с полученной таблицей включений построим начальную циклограмму работы устройства. Строить её рекомендуется в два приема.

Первый прием состоит в том, что чертят стандартную сетку циклограммы (рис. 5.4), на которой буквами a_1, a_2, b_1 и b_2 обозначены входные переменные, а рядом указаны «веса» этих переменных. В верхней строке циклограммы записаны порядковые номера тактов.

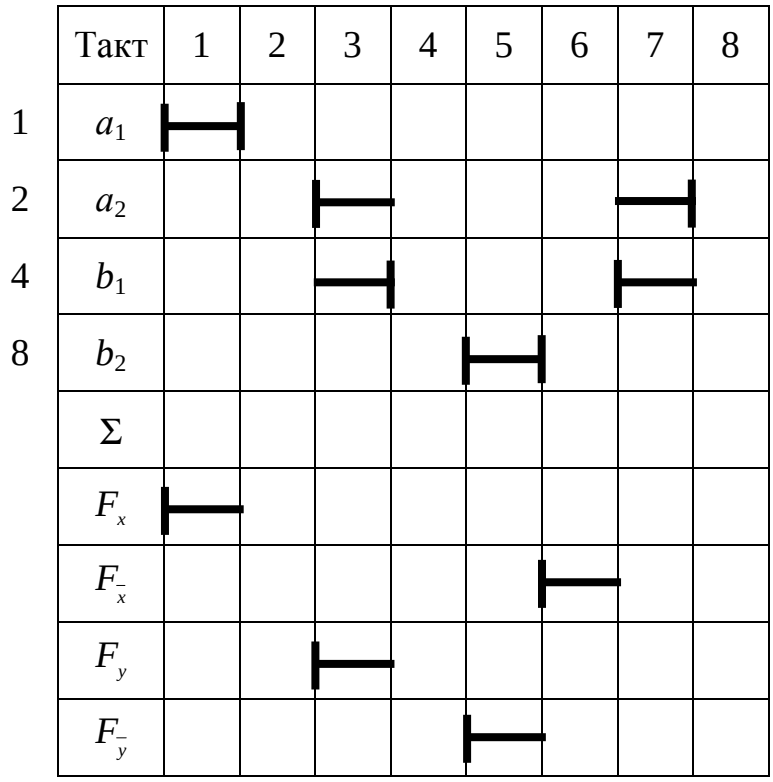


Рис. 5.4. Начальная циклограмма (первый прием)

Затем переносят информацию, содержащуюся в таблице включений, на циклограмму. Для этого в верхней части циклограммы такты, в которых входные переменные изменяют свое состояние, отмечают короткими, равными длительности одного такта, горизонтальными линиями, причем

момент изменения переменной с **0** на **1** указывают поперечным штрихом на левом, а с **1** на **0** - на правом конце линии.

Соединив между собой концы линий, на которых нет штрихов, получим циклограмму состояний входных переменных. Это уже второй прием (рис. 5.5).

В строке **Σ** надо подсчитать суммарные веса входных переменных, которые определяют состояния дискретного автомата.

В нижней части циклограммы показаны состояния выходных логических функций $F_X, F_{\bar{X}}, F_Y, F_{\bar{Y}}$, причем жирной линией отмечены обязательные состояния логических функций, пунктирной – безразличные, а состояния, где нет линии,- запрещенные.

Эту часть циклограммы также строят в два приема. Вначале (рис. 5.4) надо скопировать с таблицы включений изменения выходных функций аналогично тому, как мы это делали для входных переменных. Затем (рис. 5.5) отобразим включенные состояния выходных логических функций, учитывая, что функции F_X и $F_{\bar{X}}$, а также F_Y и $F_{\bar{Y}}$ взаимно исключающие, т.е. они не могут находиться в состоянии **1** одновременно.

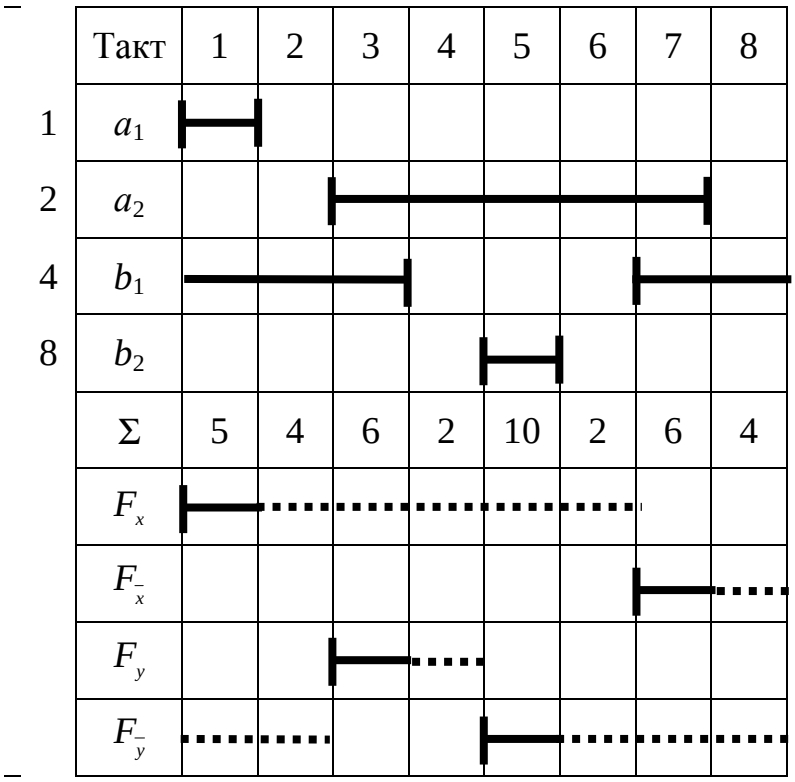


Рис. 5.5. Начальная циклограмма (второй прием)

Часть линии, изображенная пунктиром, показывает безразличные состояния функции. Это значит, что если элемент памяти на входе дискретного автомата включился в некотором такте с обязательным

состоянием, то сигналы на включение этого элемента памяти в последующих тактах лишь подтверждают его включенное состояние, т.е. являются безразличными.

Использование безразличных состояний на этапе минимизации существенно упрощает логические функции.

Важная особенность циклограмм, построенных по рассмотренному принципу, состоит в том, что в каждом такте изменяет значение одна и только одна входная переменная (верхняя часть циклограммы). На выходные переменные (нижняя часть циклограммы) это ограничение не распространяется, т.е. в одном такте допускается изменение нескольких выходных переменных.

Если в циклограмме имеются два одинаковых состояния, различающихся значениями выходных переменных, то это значит, что такая циклограмма не может быть реализована.

В рассматриваемом примере в строке $\mathbf{\Sigma}$ повторяются состояния 4, 6 и 2, 8, и 3, 7 при различных значениях выходных логических функций, т.е. одним и тем же входным сигналам отвечают разные действия.

Следовательно, в дискретный автомат необходимо ввести внутренний элемент памяти, состояние которого во 2-м, 3-м и 4-м тактах отличалось бы от состояния в 6-м, 7-м и 8-м тактах.

Обозначим внутренний элемент памяти буквой M (от английского Memory).

Назначим после 1-го такта включение, а после 5-го выключение элемента памяти M. На включение и на выключение элемента памяти выделим два дополнительных такта, которым присвоим номера, соответственно 1* и 5*, отмеченные знаком звездочка (рис. 5.6).

Теперь задача стала осуществимой, поскольку для разных действий нет повторяющихся признаков.

Циклограмма с введёнными внутренними элементами памяти называется *реализуемой*.

Реализуемая циклограмма отличается от начальной тем, что превышает её длину на количество тактов, отведённых на включение и на выключение внутренних элементов памяти. В верхней части реализуемая циклограмма содержит дополнительные строки для переменных, поступающих с выходов, а в нижней – подаваемых на входы внутренних элементов памяти.

При построении реализуемой циклограммы необходимо учитывать, что переменные, которые включают внутренние элементы памяти, изменяются так же, как и переменные, поступающие с выходов этих элементов, но со смещением по фазе на один такт влево.

Например, в нашем устройстве сигнал m с выхода внутреннего элемента памяти принимает единичное значение в такте 1*, а сигнал F_m на

включение этого элемента – в такте 1. Далее сигнал m принимает нулевое значение в такте 5*, а сигнал $F_{\bar{m}}$ – в такте 5.

Подробная методика составления реализуемой циклограммы будет рассмотрена позже.

Чтобы минимизировать функции $F_X, F_{\bar{X}}, F_Y, F_{\bar{Y}}, F_m, F_{\bar{m}}$, воспользуемся картами Карно для пяти переменных. По реализуемой циклограмме устанавливаем, что обязательным состоянием функции F_X является эквивалентное десятичное число **21**, запрещенные состояния – числа **5**, **6**, **4**, а остальные состояния – безразличные.

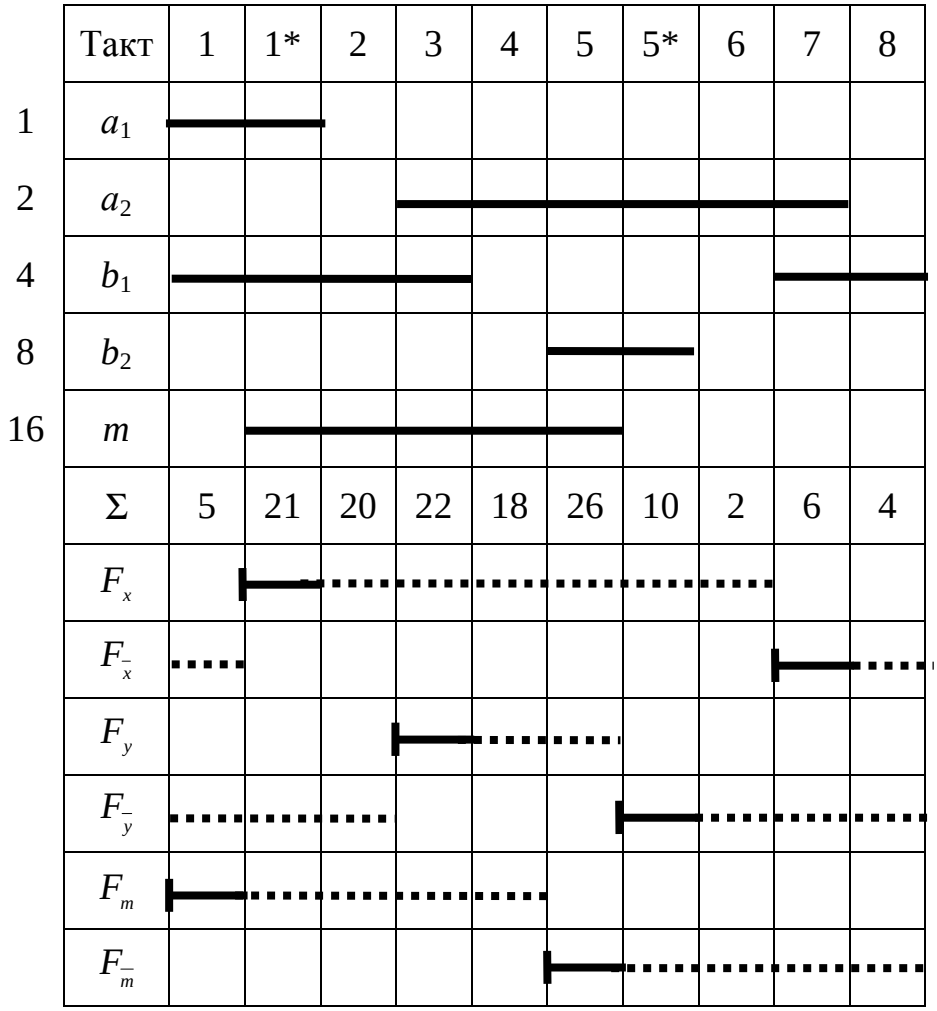


Рис. 5.6. Реализуемая циклограмма

Запишем отмеченные состояния, отметим по периметру карты Карно значения суммарных «весов» входных переменных и выполним минимизацию функции F_X (рис. 5.7).

Аналогично минимизируем функцию $F_{\bar{X}}$ (рис. 5.8).

$F_x = 21$ $\qquad$ $\overline{F_x} = 5,\ 6,\ 4$

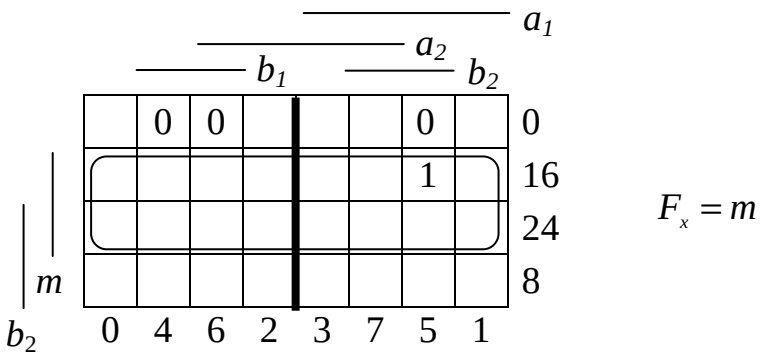


Рис. 5.7. Минимизация логической функции F_x

$F_{\bar{x}} = 6;$ $\qquad$ $\overline{F_{\bar{x}}} = 21,\ 20,\ 22,\ 18,\ 26,\ 10,\ 2$

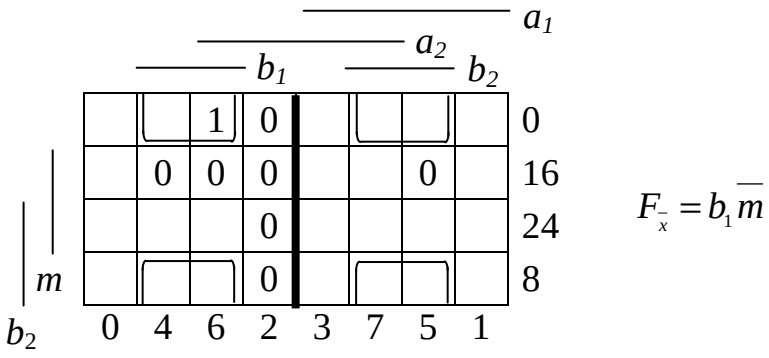


Рис. 5.8. Минимизация логической функции $F_{\bar{x}}$

Минимизируя остальные функции, получаем:

$F_y = a_2 m$, $\quad F_{\bar{y}} = \overline{m}$, $\quad F_m = a_1$, $\quad F_{\bar{m}} = b_2$.

Теперь перейдём к построению функциональной схемы дискретного автомата. Она состоит из четырёх путевых переключателей, с которых поступают входные переменные a_1, a_2, b_1, b_2 , выходных запоминающих элементов x, y, внутреннего запоминающего элемента M и двух логических элементов **И** (рис. 5.9).

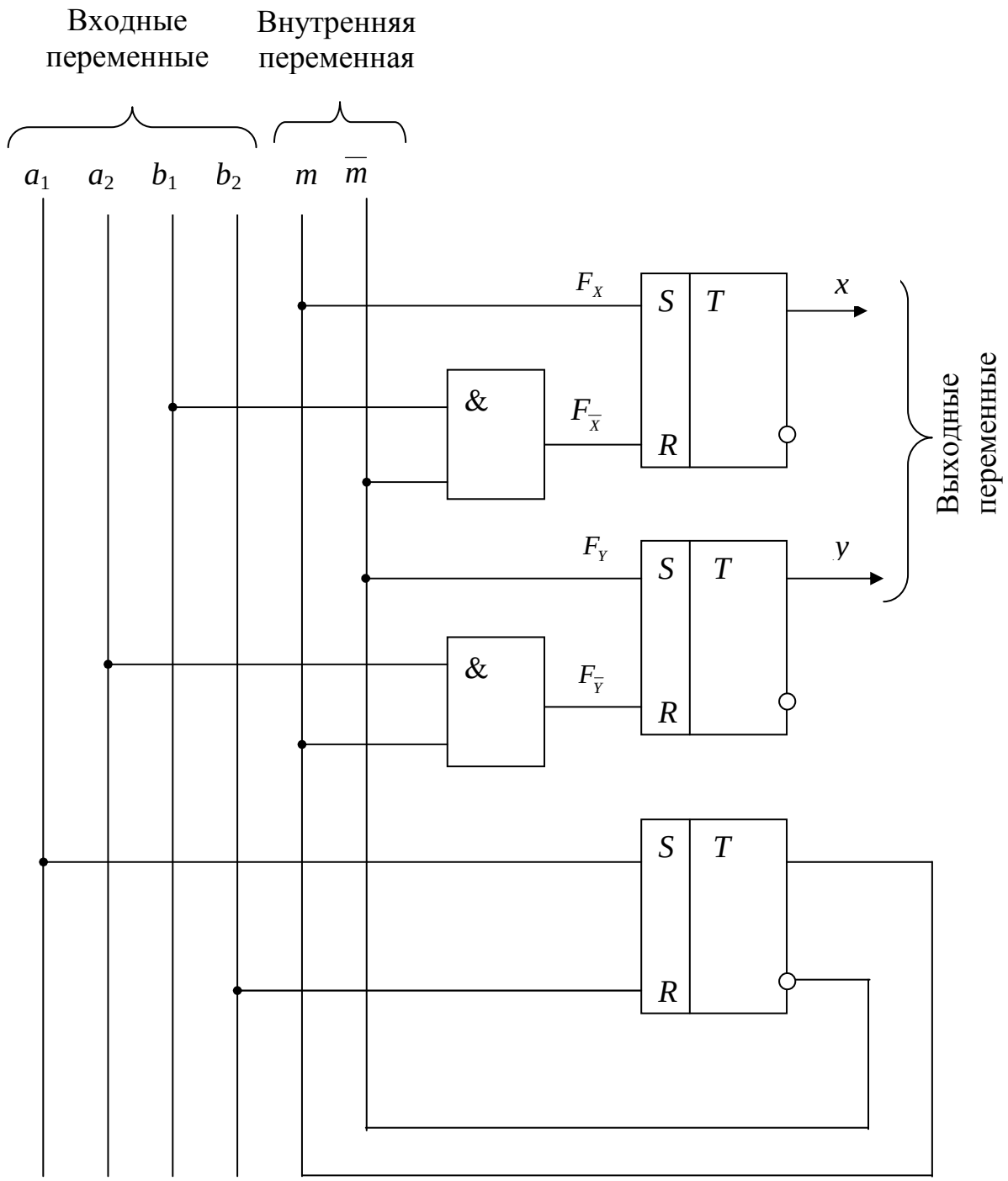


Рис. 5.9. Функциональная схема дискретного автомата

5.4. Методика составления реализуемой циклограммы

Реализуемая циклограмма строится на основе начальной циклограммы путем добавления тактов, учитывающих включение и выключение внутренних элементов памяти. Эти такты называют *дополнительными*.

Количество и место дополнительных тактов в реализуемой циклограмме определяют по следующей методике:

1) Представить последовательность тактов начальной циклограммы в виде ряда *весовых коэффициентов* (рис. 5.10, *а*):

$$N_1^{k1}, N_2^{k2}, N_3^{k3}, \ldots, N_i^{ki},$$

где N_i - суммарный вес входных переменных в *i*-м такте;

I - номер такта;

k_i - номер повторения суммарного веса, соответствующего i-му такту.

Указанные весовые коэффициенты определяют состояния дискретного автомата (или просто «состояния») в каждом такте.

2) Такты с повторяющимися весовыми коэффициентами охватить сверху скобкой, начало которой соответствует моменту включения, а конец – моменту выключения внутреннего элемента памяти (рис. 5.10, *б*).

Над скобкой надо указать имя и вес внутренней переменной, относящейся к этому элементу памяти.

Весовые коэффициенты, охваченные скобкой, увеличиваются на вес новой переменной, и в результате количество повторяющихся состояний дискретного автомата уменьшится.

$1_1^1 0_2^1 2_3^1 0_4^2 1_5^2 0_6^3 2_7^2 0_8^4 4_9^1 0_{10}^5 2_{11}^3 0_{12}^6$

а

m_1=8

$1_1^1 0_2^1 2_3^1 0_4^2 1_5^2 0_6^3 2_7^2 0_8^4 4_9^1 0_{10}^5 2_{11}^3 0_{12}^6$

б

m_2=16

$1_1^1 0_2^1 2_3^1 10_{3*}^1 8_4^1 9_5^1 8_6^2 10_7^2 8_8^3 12_9^1 4_{9*}^1 0_{10}^2 2_{11}^2 0_{12}^3$

в

m_3=32

$1_1^1 17_{1*}^1 16_2^1 18_3^1 26_{3*}^1 24_4^1 25_5^1 9_{5*}^1 8_6^1 10_7^2 8_8^2 12_9^1 4_{9*}^1 0_{10}^1 2_{11}^1 0_{12}^2$

г

$1_1^1 17_{1*}^1 16_2^1 18_3^1 26_{3*}^1 24_4^1 25_5^1 9_{5*}^1 8_6^1 10_7^2 42_{7*}^1 40_8^1 44_9^1 36_{9*}^1 32_{10}^1 34_{11}^1 2_{11*}^1 0_{12}^1$

д

Рис. 5.10. Пример введения в начальную последовательность весовых коэффициентов дополнительных тактов

Конечная цель заключается в том, чтобы исключить в циклограмме такты с повторяющимися состояниями, для чего можно применить 2, 3 или большее число скобок. Длину и расположение скобок надо выбирать так, чтобы использовать по возможности меньшее количество скобок, так как каждая дополнительная скобка добавляет в систему управления внутренний элемент памяти и система управления усложняется.

Циклограмма, в которой нет повторяющихся состояний, становится *реализуемой*.

3) После введения очередного внутреннего элемента памяти найти суммарные веса входных и внутренних переменных во всех тактах и выделить дополнительные такты на включение и на выключение вновь введенного элемента памяти при помощи следующего правила:

Весовые коэффициенты, на которые указывают начало и конец скобки, учитываются дважды: слева от границы скобки и справа от нее (рис. 5.10, *б*, *в*). в результате каждый из этих весовых коэффициентов распадается на два коэффициента, причем второй из них определяет такт включения (выключения) внутреннего элемента памяти. Порядковый номер этого такта обозначают тем же номером, что и номер предыдущего такта, но со знаком звездочка (*).

Например, коэффициент 2_3^1 (рис. 5.10, *б*) порождает коэффициенты 2_3^1 и 10_{3*}^1 (рис. 5.10, *в*), а коэффициент 4_9^1 (рис. 5.10, *б*) – соответственно коэффициенты 12_9^1 и 4_{9*}^1 (рис. 5.10, *в*).

4) Строя скобки, надо учитывать ряд условий:

а) Включение одного и выключение другого внутреннего элемента памяти не могут происходить в одном и том же такте.

б) Последняя, завершающая скобка должна начинаться и заканчиваться только в такте, обозначенном условным числом с номером повторения суммарного веса $k = 1$ (рис. 5.10, *г*).

Для предшествующих скобок выполнение данного условия желательно, но не обязательно.

в) Можно применять, если это целесообразно, многократное включение и выключение одного и того же внутреннего элемента памяти за время цикла.

г) В реализуемой циклограмме допускается повторение суммарных весов входных переменных в некоторых тактах, если в этих же тактах повторяются состояния и выходных переменных.

Применим рассмотренную методику составления реализуемой циклограммы для синтеза системы управления автоматом–перекладчиком (рис. 5.11).

Автомат–перекладчик может быть предназначен для различных загрузочных или сборочных технологических операций и состоит из двух исполнительных цилиндров X и Y, которые работают в следующей последовательности :

$$X,\ \overline{X},\ Y,\ X,\ \overline{X},\ \overline{Y}.$$

Положения штока цилиндра X контролируются путевыми переключателями A_1 и A_2, а штока цилиндра Y – путевыми переключателями B_1 и B_2.

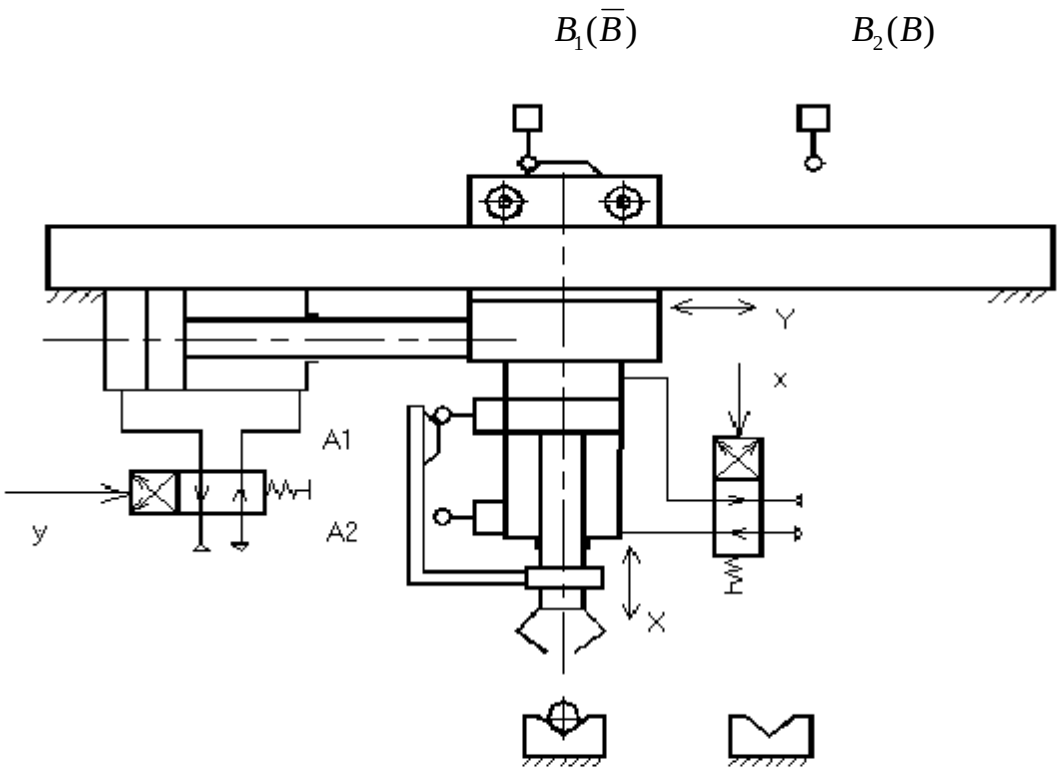


Рис. 5.11. Структурно-кинематическая схема автомата–перекладчика

Составим таблицу включений с учетом сигнала блокировки P (*Pusk*). Если P = 0, то запуск автоматического цикла запрещен. Первое действие автоматического цикла (F_X = 1) разрешается только при P = 1. Этот сигнал может поступать от кнопки «*ПУСК*», т.е. с пульта оператора, или от какой-либо другой подсистемы, являющейся частью общей системы управления.

В результате таблица включений принимает следующий вид:

1. $b_1 = 1$	Если $P = 1$, то $F_X = 1$
2. $a_1 = 0$	
3. $a_2 = 1$	$F_{\bar{x}} = 1$
4. $a_2 = 0$	
5. $a_1 = 1$	$F_y = 1$
6. $b_1 = 0$	
7. $b_2 = 1$	$F_x = 1$
8. $a_1 = 0$	
9. $a_2 = 1$	$F_{\bar{x}} = 1$
10. $a_2 = 0$	
11. $a_1 = 0$	$F_{\bar{y}} = 1$
12. $b_2 = 0$	

На основании таблицы включений строим начальную циклограмму (рис. 5.12).

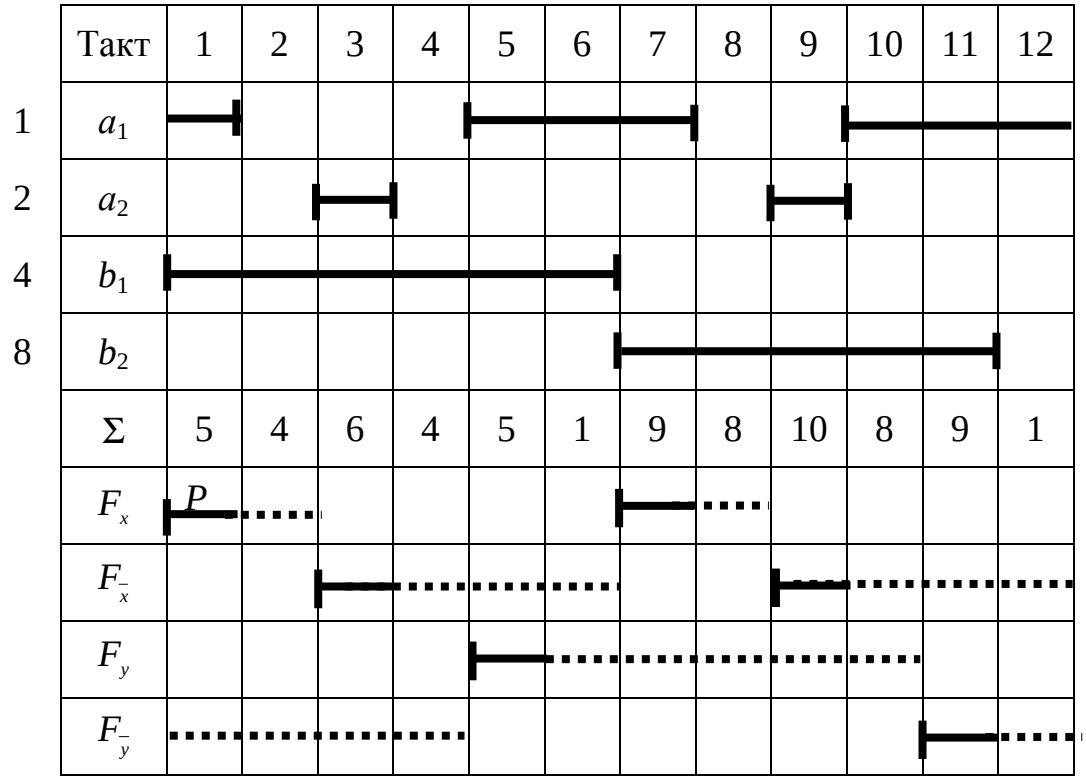


Рис. 5.12. Начальная циклограмма автомата–перекладчика

Буква P над чертой, показывающей изменение состояния функции F_X в первом такте, означает внешний сигнал блокировки *Pusk*.

Данная начальная циклограмма не может быть реализована, так как в строке **Σ** повторяются суммы весов 5, 4, 1, 9 и 8. Чтобы исключить эти повторения, введем в систему управления внутренний элемент памяти и, выписав в ряд все весовые коэффициенты, отметим с помощью скобки такты, в которых следует включить и выключить внутренний элемент памяти (рис. 5.13, *а*).

m_1=16

$5^1_1 4^1_2 6^1_3 4^2_4 5^2_5 1^1_6 9^1_7 8^1_8 10^1_9 8^2_{10} 9^2_{11} 1^2_{12}$

а

$5^1_1 4^1_2 6^1_3 22^1_{3*} 20^1_4 21^1_5 17^1_6 25^1_7 24^1_8 26^1_9 10^1_{9*} 8^1_{10} 9^1_{11} 1^1_{12}$

б

Рис. 5.13. Введение дополнительных тактов на включение и на выключение внутреннего элемента памяти

Внутренний элемент памяти целесообразно включить в 3-м, а выключить в 9-м тактах. Тогда весовые коэффициенты 4, 5, 1, 9 и 8, заключенные внутри скобки, изменятся, и не будут совпадать с такими же весовыми коэффициентами, находящимися за пределами скобки. Следовательно, цель достигнута.

Найденный ряд неповторяющихся весовых коэффициентов (рис. 5.13, *б*) служит основой для построения реализуемой циклограммы. Чтобы её получить, надо выполнить следующие действия:

1) Записать в строке **Σ** ряд неповторяющихся весовых коэффициентов, в котором учтены дополнительные такты на включение и на выключение внутренних элементов памяти (рис. 5.14).

2) В верхней части циклограммы отметить горизонтальными линиями состояния входных и внутренних переменных исходя из того, чтобы суммы весовых коэффициентов в каждом такте были равны значениям, записанным в строке **Σ**.

3) В нижней части циклограммы с помощью коротких, равных длительности одного такта, линий указать переходы выходных логических функций из состояния **0** в состояние **1**, отмечая моменты этих переходов поперечными штрихами на левых концах линий.

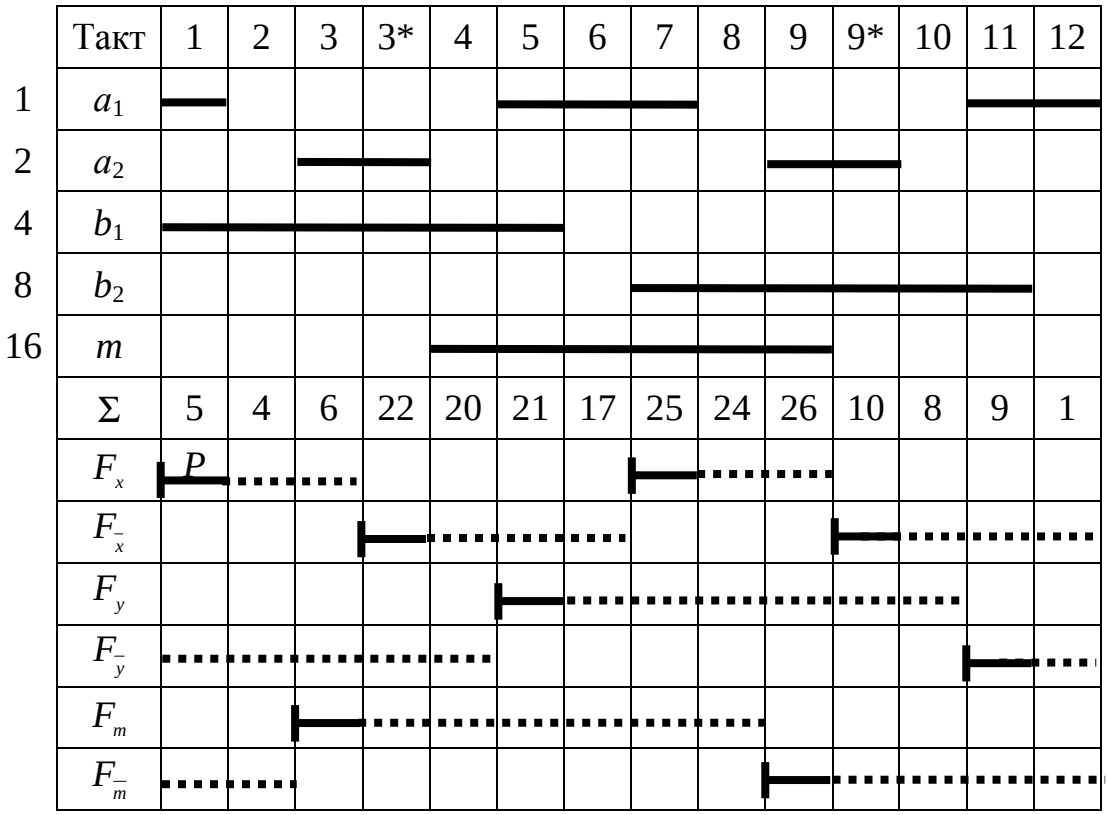


Рис. 5.14. Реализуемая циклограмма автомата–перекладчика

Такты на переходы выходных логических функций в единичное состояние определяют по следующим правилам:

а) Номера тактов на включение и на выключение выходных элементов памяти в начальной и в реализуемой циклограммах должны совпадать.

б) Если изменение состояния выходного элемента памяти выпало на такт, номер которого в реализуемой циклограмме представлен дважды, причем один отмечен звездочкой, а другой нет, то надо выбрать в качестве рабочего такт, отмеченный звездочкой. В противном случае между выходным и внутренним элементами памяти могут возникать состязания (гонки). Если внутренний элемент памяти переключится раньше выходного (выиграет гонку), то он изменит состояние дискретного автомата и выходной элемент, возможно, не успеет переключиться.

Данное условие не распространяется на такты, отмеченные знаком блокировки типа *Pusk*. Например, если в такте **1** включается внутренний элемент памяти, причем функция его включения содержит в этом такте блокировку от внешнего сигнала *Pusk*, то это значит, что такт **1** является исходным и поэтому изменения состояний логических функций на включение и на выключение выходных элементов памяти можно назначать

как в такте **1***, так и в такте **1**. Обычно в такте **1** выключают, а в такте **1*** включают выходные элементы памяти.

3) Указать с помощью пунктирных линий безразличные состояния логических функций.

Чтобы исключить совпадения единичных состояний логических функций на включение и на выключение одного и того же элемента памяти в одном такте, пунктирную линию на включение надо проводить с такта после обязательного включения до такта на обязательное выключение, а пунктирную линию на выключение - с такта после обязательного выключения до такта на обязательное включение каждого элемента памяти.

Из полученной реализуемой циклограммы видно, что для управления автоматом–перекладчиком требуются два выходных элемента памяти с функциями включения, соответственно F_x и F_y, и с функциями выключения $F_{\bar{x}}$ и $F_{\bar{y}}$, а также внутренний элемент памяти с функцией включения F_m и с функцией выключения $F_{\bar{m}}$.

Минимизируем каждую из этих функций с помощью карт Карно (рис. 5.15).

В процессе минимизации логических функций переменную P , означающую внешний сигнал блокировки $Pusk$, надо отметить в соответствующей клетке карты Карно тем же знаком P и, выделяя простые импликанты, принять во внимание следующее правило:

Импликанта, содержащая символ блокировки, не должна полностью поглощать (включать в себя) другие импликанты, содержащие цифры 1 или символы других блокировок (рис. 5.15-1). Пересечение же указанных импликант разрешается.

Полученные логические функции отличаются тем, что содержат входные переменные, которые определяют положения двухпозиционных органов. Отметим два свойства таких логических функций:

1) Логическую функцию можно привести к виду, в котором входные переменные, определяющие положения рабочих органов только в двух позициях, не имеют инверсий.

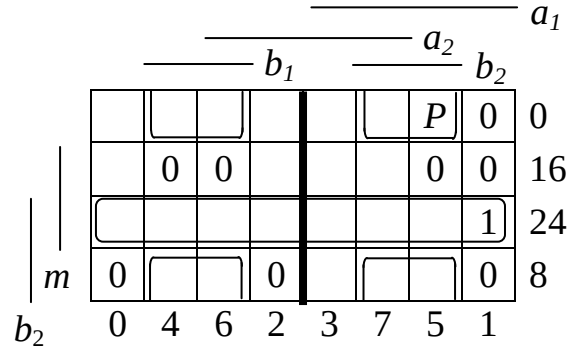
2) Если в логической функции входные переменные, определяющие положения рабочих органов только в двух позициях, не имеют инверсий, то та из указанных входных переменных, которая изменяется в такте перехода логической функции с **0** на **1** или с **1** на **0**, соответственно также изменяется с **0** на **1** или с **1** на **0**.

Не приводя строгого доказательства этих свойств, проверим их на нашем примере.

Среди логических функций автомата–перекладчика (рис. 5.15) функции $F_{\bar{x}}$, F_m и $F_{\bar{m}}$ имеют входные переменные с инверсиями. Поищем для этих логических функций другие минимальные формы, в которых входные переменные не содержат инверсий (рис. 5.16).

1) $F_x = P \cdot 5,\ 25$

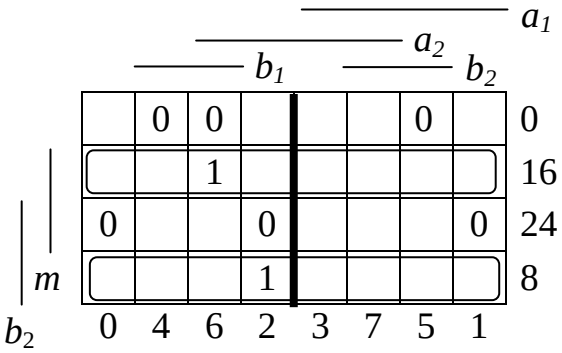
$\overline{F_x} = 22,\ 20,\ 21,\ 17,\ 10,\ 8,\ 9,\ 1$



$F_x = P \cdot b_1\overline{m} + b_2 m$

2) $F^{-}_{x} = 22,\ 10$

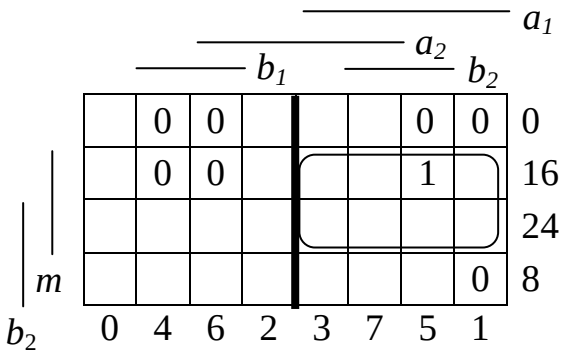
$\overline{F_x} = 5,\ 4,\ 6,\ 25,\ 24,\ 26$



$F_x = \overline{b_2} m + b_2 \overline{m}$

3) $F_y = 21$

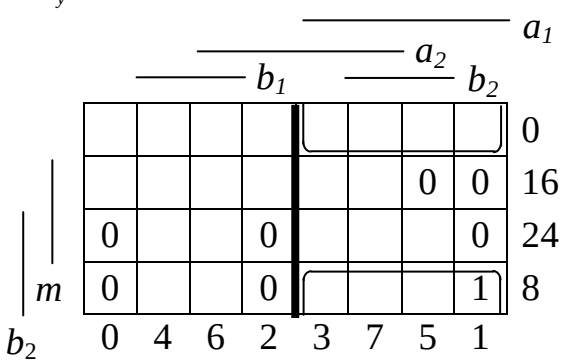
$F_y = 5,\ 4,\ 6,\ 22,\ 20,\ 9,\ 1$



$F_y = a_1 m$

4) $F_y = 9$

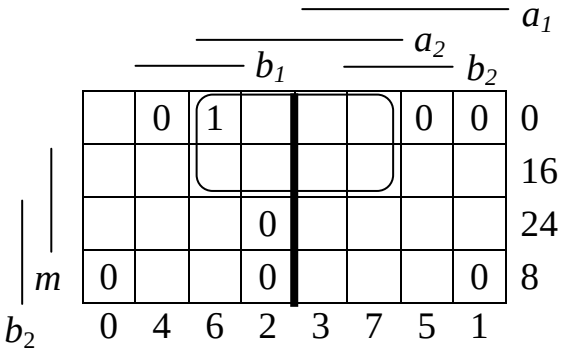
$\overline{F_y} = 21,\ 17,\ 25,\ 24,\ 26,\ 10,\ 8$



$F_y = a_1 \overline{m}$

5) $F_m = 6$

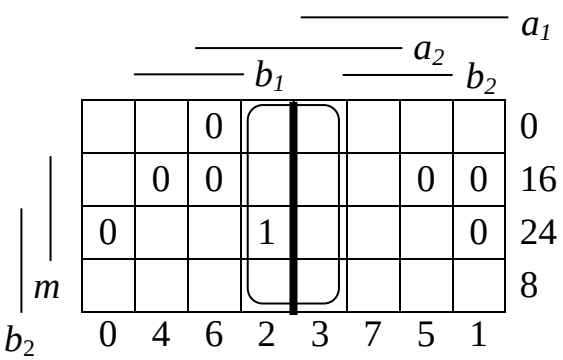
$\overline{F_m} = 5,\ 4,\ 26,\ 10,\ 8,\ 9,\ 1$



$F_m = a_2 \overline{b_2}$

6) $F_m = 26$

$\overline{F_m} = 6,\ 22,\ 20,\ 21,\ 17,\ 25,\ 24$



$F_m = a_2 \overline{b_1}$

Рис. 5.15. Минимизация логических функций автомата–перекладчика

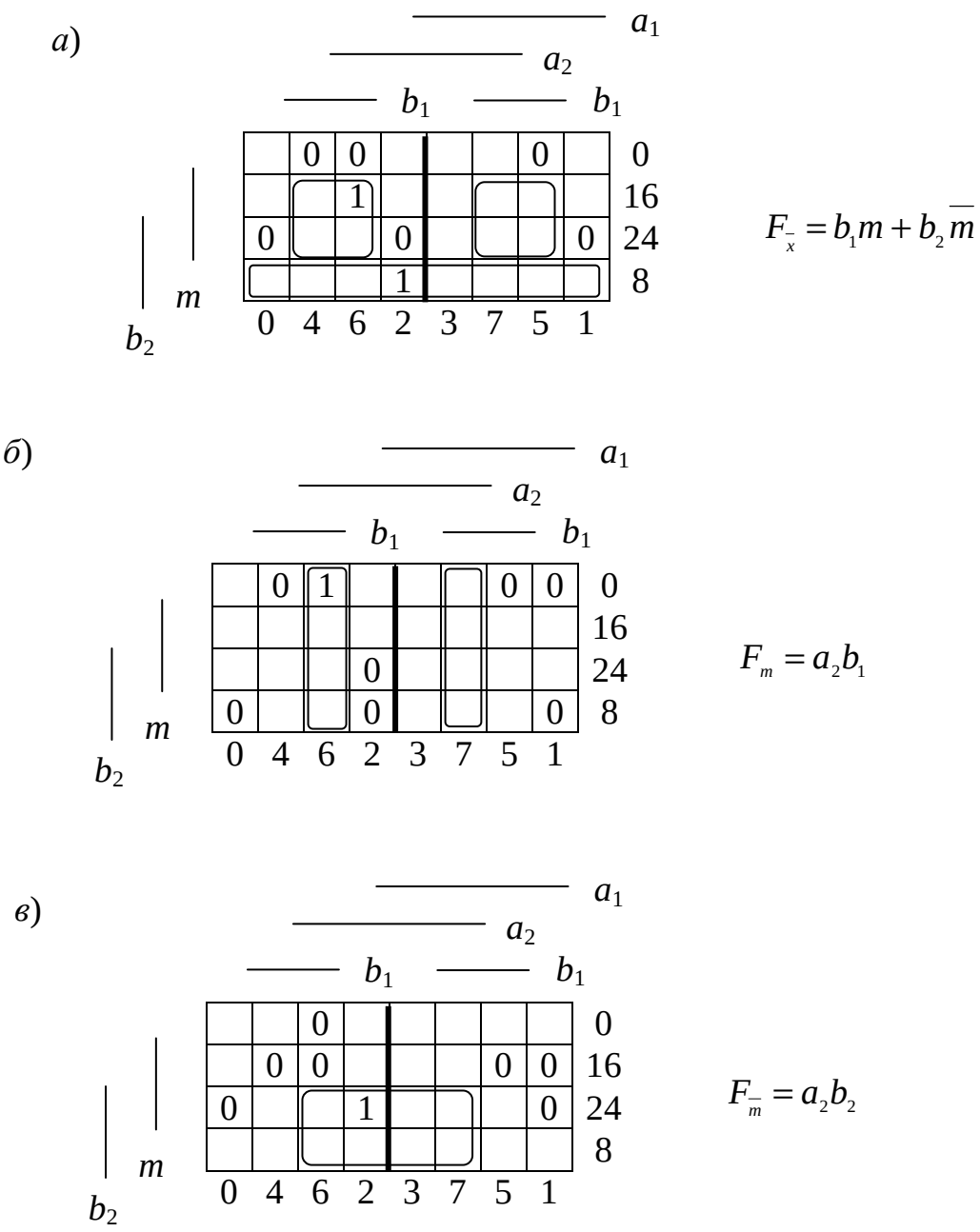


Рис. 5.16. Вторые формы логических функций автомата–перекладчика

Искомые формы логических функций нашлись, что подтверждает справедливость первого свойства.

Заменим логические функции $F_{\bar{x}}$, F_m и $F_{\bar{m}}$ (рис. 5.15) их новыми выражениями (рис. 5.16). Составим общий список логических функций и сопоставим его с реализуемой циклограммой автомата–перекладчика (рис. 5.14).

$$F_x = P \cdot b_1 \cdot \overline{m} + b_2 \cdot m$$

$$F_{\overline{x}} = b_1 \cdot m + b_2 \cdot \overline{m}$$

$$F_y = a_1 \cdot m$$

$$F_{\overline{y}} = a_1 \cdot \overline{m}$$

$$F_m = a_2 \cdot b_1$$

$$F_{\overline{m}} = a_2 \cdot b_2$$

Отмечая значками ⊕ и ⊖ переменные, изменения которых переводят логические функции соответственно в состояние **1** и в состояние 0, видим, что все логические функции обладают вторым выше указанным свойством.

Оба рассмотренных свойства потребуются нам в дальнейшем.

На основании последних выражений построим дискретный автомат. Его функциональная схема представлена на (рис. 5.17).

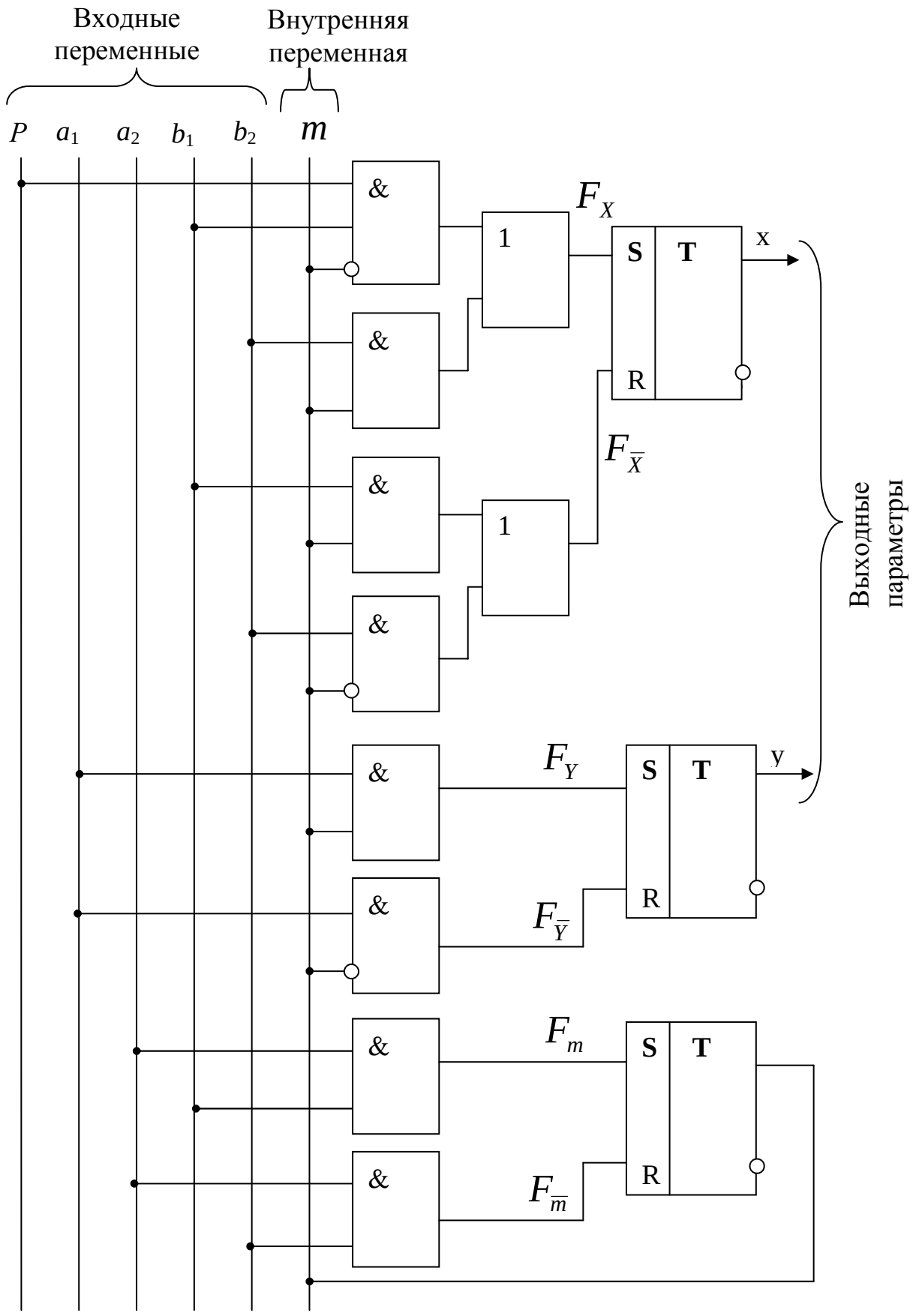


Рис. 5.17. Функциональная схема управления автоматом–перекладчиком

5.5. Методика упрощенного синтеза дискретных систем управления

Рассмотрим два переключателя A_1 и A_2, контролирующих положение двухпозиционного органа (рис. 5.18).

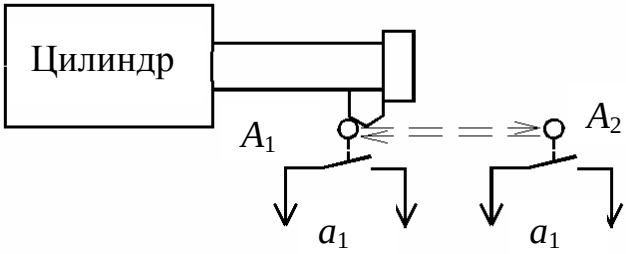


Рис. 5.18. Двухпозиционный орган с реальными переключателями

Переменные a_1 и a_2 существуют в комбинациях $a_1\overline{a_2}$ и $\overline{a_1}a_2$ и не существуют в комбинации a_1a_2. Комбинация переменных $\overline{a_1}\,\overline{a_2}$, хотя и существует, но в тактах циклограммы, в которых встречается эта комбинация, выходные и внутренние переменные не изменяют своих состояний. Следовательно, пару переключателей A_1 и A_2 можно условно представить в виде одного виртуального переключателя, например, A, содержащего замыкающий и размыкающий контакты a и $\overline{a}$ (рис. 5.19).

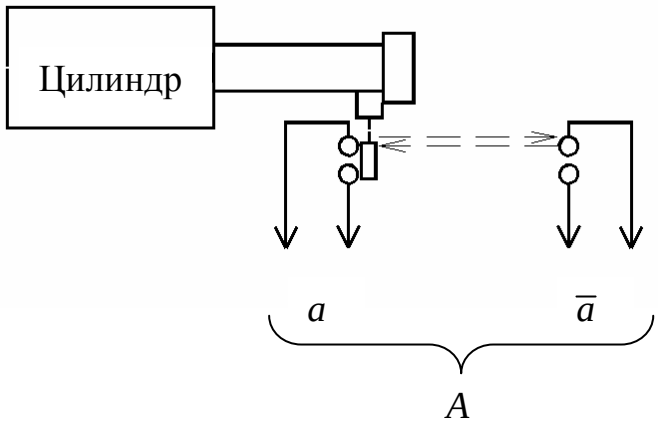


Рис. 5.19. Двухпозиционный орган с виртуальными переключателями

Время переключения переключателя A равно времени перемещения штока цилиндра из одного крайнего положения в другое.

В сложных системах замена реальных переменных виртуальными сокращает общее число переменных, что упрощает процедуру минимизации

логических функций, особенно если минимизацию производят без помощи ЭВМ, т.е. вручную.

Вернемся к ранее рассмотренной структурно-кинематической схеме автомата–перекладчика (рис. 5.11). На этой схеме в скобках обозначены виртуальные переключатели, благодаря которым вместо четырех реальных переменных a_1 , a_2 , b_1 , b_2 остались только две виртуальные переменные a и b.

В результате таблица включений по сравнению с первоначальным вариантом сокращается в два раза:

1. $b = 0$	Если $P = 1$, то $F_X = 1$
2. $a = 1$	$F_{\bar{X}} = 1$
3. $a = 0$	$F_y = 1$
4. $b = 1$	$F_X = 1$
5. $a = 1$	$F_{\bar{X}} = 1$
6. $a = 0$	$F_{\bar{y}} = 1$

На основании таблицы включений строим начальную циклограмму (рис. 5.20).

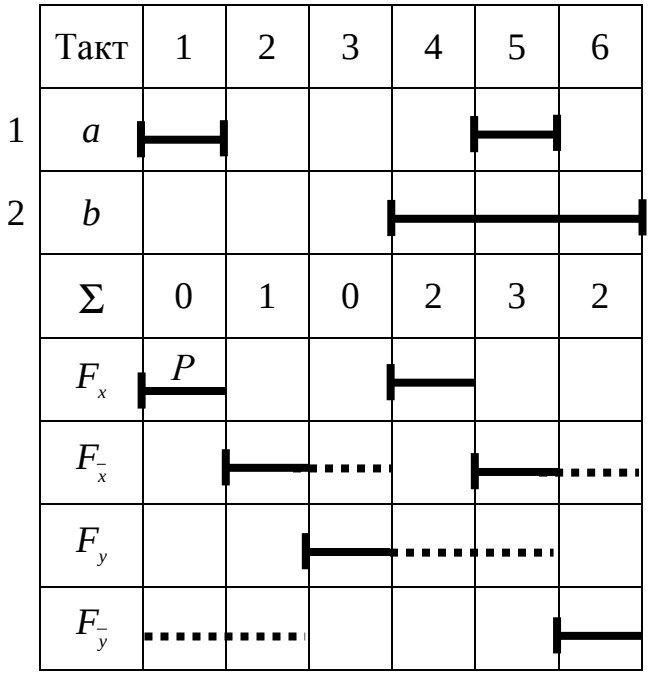


Рис. 5.20. Начальная циклограмма автомата-перекладчика (упрощенный вариант)

Для исключения повторяющихся весовых коэффициентов вводим в дискретный автомат внутренний элемент памяти (рис. 5.21).

m=4

$0^1_1 1^1_2 0^2_3 2^1_4 3^1_5 2^2_6$

а

$0^1_1 1^1_2 5^1_{2*} 4^1_3 6^1_4 7^1_5 3^1_{5*} 2^1_6$

б

Рис. 5.21. Введение в дискретный автомат элемента памяти (упрощенный вариант)

Далее строим реализуемую циклограмму (рис. 5.22).

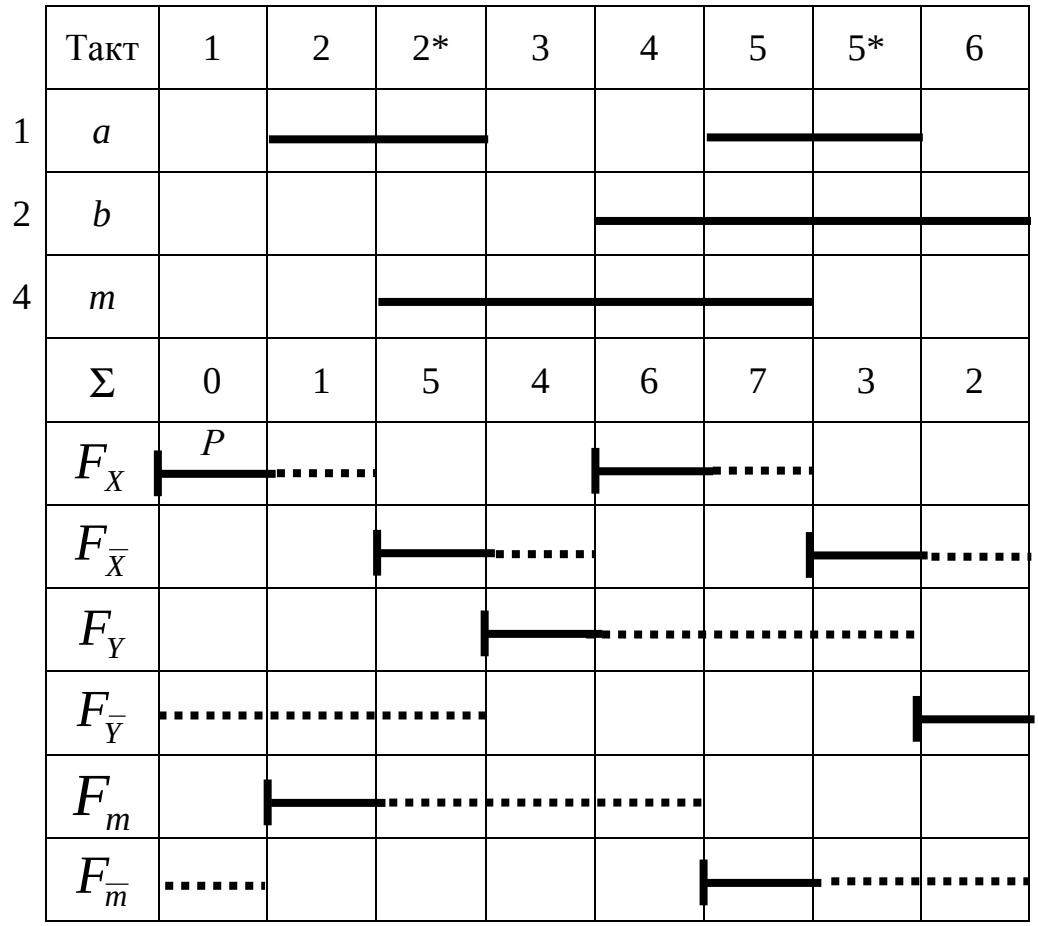


Рис. 5.22. Реализуемая циклограмма автомата-перекладчика (упрощенный вариант)

Прежде чем приступить к минимизации логических функций, рассмотрим правило перехода от виртуальных переменных к реальным.

На основании первого свойства логических функций, управляющих двухпозиционными органами (см. предыдущий параграф), будем искать наши логические функции в форме, в которой входные переменные не содержат инверсий. Тогда в соответствии со вторым свойством тех же логических функций правило перехода от виртуальных переменных к реальным можно сформулировать очень просто: *виртуальные входные переменные, определяющие положения рабочих органов только в двух позициях, надо заменить реальными входными переменными в соответствии с обозначениями, принятыми на структурно-кинематической схеме устройства.*

Применительно к рассматриваемому автомату–перекладчику (рис. 5.11) указанная замена выглядит следующим образом:

$$\bar{a} \to a_1$$

$$a \to a_2$$

$$\bar{b} \to b_1$$

$$b \to b_2$$

Используя данную замену переменных, минимизируем логические функции, содержащиеся в полученной выше реализуемой циклограмме (рис. 5.23).

Найденные логические функции и логические функции, которые были получены ранее (см. предыдущий параграф), совпадают, что подтверждает корректность предлагаемой методики упрощенного синтеза дискретных систем управления.

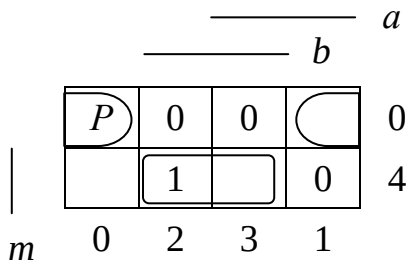
5.6. Состязания в дискретных автоматах

Ранее считалось, что при изменении состояний дискретного автомата входные и внутренние переменные изменяются мгновенно. В реальных условиях изменение значения сигнала на входе дискретного автомата вызывает изменения сигналов в промежуточных узлах комбинационной схемы с некоторым отставанием по времени, определяемым инерционными свойствами логических элементов. В результате если между входом и выходом дискретного автомата имеется несколько путей с разными временами прохождения сигнала, то в переходный период на выходе возможно появление кратковременного сигнала (всплеска), не соответствующего статическим состояниям входов.

В контактных схемах появление такого сигнала связано с неодновременным переключением (замыканием или размыканием) разных контактов одного реле.

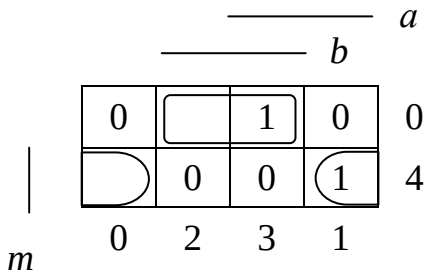
Такое явление получило название *состязаний цепей* или *сигналов* в комбинационных схемах.

1) $F_x = P \cdot 0,6 \qquad \overline{F_x} = 5,4,3,2$



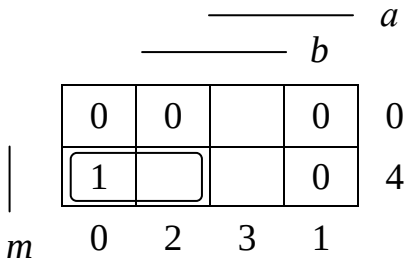
$F_x = P\bar{b}\bar{m} + bm = Pb_1\bar{m} + b_2 m$

2) $F_{\bar{x}} = 5,3 \qquad \overline{F_{\bar{x}}} = 0,1,6,7$



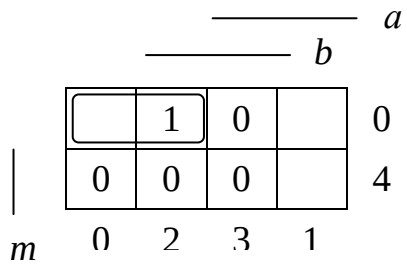
$F_{\bar{x}} = \bar{b}m + b\bar{m} = b_1 m + b_2\bar{m}$

3) $F_y = 4 \qquad \overline{F_y} = 0,1,5,2$



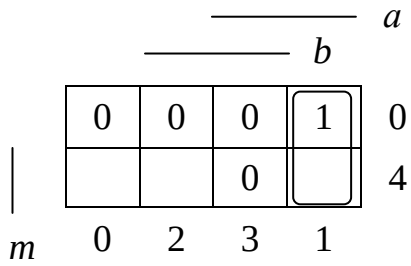
$F_y = \bar{a}m = a_1 m$

4) $F_{\bar{y}} = 2 \qquad \overline{F_{\bar{y}}} = 4,6,7,3$



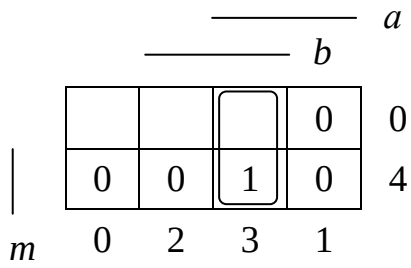
$F_{\bar{y}} = \bar{a}\bar{m} = a_1\bar{m}$

5) $F_m = 1 \qquad \overline{F_m} = 0,7,3,2$



$F_m = a\bar{b} = a_2 b_1$

6) $F_{\bar{m}} = 7 \qquad \overline{F_{\bar{m}}} = 1,5,4,6$



$F_{\bar{m}} = ab = a_2 b_2$

Рис. 5.23. Минимизация логических функций автомата-перекладчика (упрощенный вариант)

Если в результате состязания не нарушается функционирование дискретного автомата (нет непредвиденных включений или выключений выходных элементов и элементов памяти), то такие состязания называются *допустимыми* (*некритическими*), в противном случае состязания будут *недопустимыми* (*критическими*).

Определим условия появления состязаний цепей в комбинационной схеме при единичном изменении сигнала на одном из входов дискретного автомата.

Рассмотрим реализацию функции F в двух формах: в форме ДНФ и в форме КНФ.

$$F = ax + b\bar{x} = (b + x)(a + \bar{x})$$

Этим формам соответствуют схемы, приведенные на рис. 5.24.

В статическом состоянии обе схемы эквивалентны. Однако в переходные периоды при $a = b = 1$ и изменении состояния входа x в схеме на рис. 5.24, *а* возможен кратковременный разрыв цепи (появление нулевого всплеска). Это произойдет, если контакт $\bar{x}$ разомкнется раньше, чем замкнется x. Во второй схеме при заданных условиях цепь будет замкнута все время. Аналогично при $a = b = 0$ в статическом состоянии все цепи будут разорваны, но при изменении состояния входа x в схеме на рис. 5.24, *в* возможно кратковременное замыкание цепи (появление единичного всплеска).

Не приводя здесь строгих доказательств, которые имеются в специальной литературе [4], отметим следующие важные свойства комбинационных схем, связанные с состязаниями:

1) Появление всплесков, вызванных состязаниями на выходе дискретного автомата, возможно только при наличии взаимоинверсных сигналов (разноименных контактов) в цепи комбинационной схемы.
2) Когда некоторая цепь комбинационной схемы выражена в ДНФ (в виде суммы произведений), то при изменении одной переменной в этой цепи возможен только нулевой всплеск.
3) Если цепь комбинационной схемы моделируется КНФ (произведением сумм), то при изменении одной переменной в данной цепи возможен только единичный всплеск.

Следовательно, если при синтезе дискретных систем управления применять логические функции в форме ДНФ и использовать на выходе дискретного автомата в качестве выходных элементов и элементов памяти статические триггеры, то в таких системах не могут возникнуть недопустимые состязания, так как возникающие при состязаниях нулевые всплески не влияют на состояния триггеров.

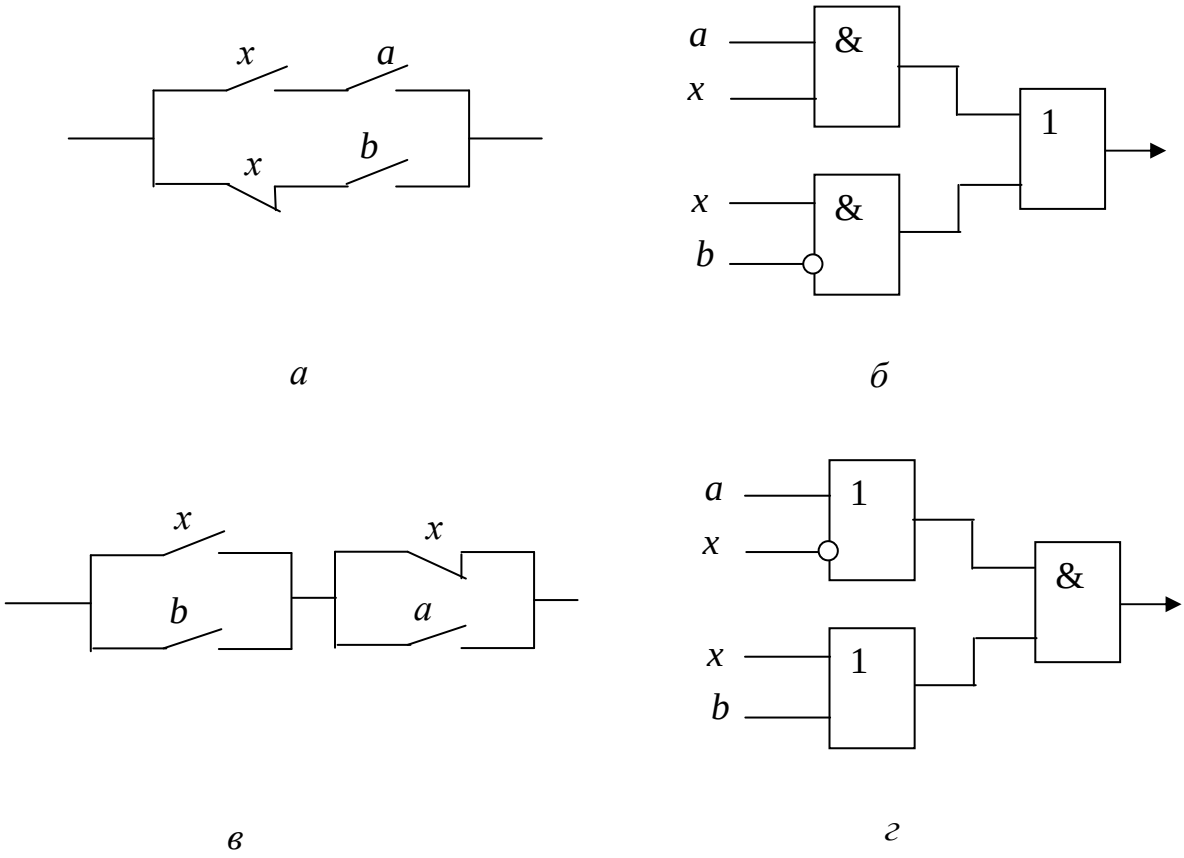


Рис. 5.24. Элементарные системы, создающие состязания

5.7. Непрерывные и прерывистые логические функции

Используемые при синтезе дискретных систем управления логические функции можно разделить на два вида:

- прерывистые;
- непрерывные.

Прерывистая логическая функция принимает значение **1** в обязательных состояниях, а в безразличных состояниях она может принимать значения как **1** так и **0**. Поэтому для сохранения требуемого значения выходной переменной (**0** – в запрещенных и **1** – в обязательных и в безразличных состояниях) необходим специальный запоминающий элемент – статический триггер.

В ранее рассмотренных примерах мы уже применяли метод синтеза дискретных систем, основанный на формировании прерывистых логических функций в сочетании с запоминающими элементами в виде триггеров.

Преимущество дискретных систем управления с прерывистыми логическими функциями состоит в том, что в таких системах не возникают состязания цепей (см. 5.6).

Непрерывная логическая функция сохраняет постоянное значение (0 или 1) во всех тактах циклограммы, в течение которых выходная переменная равна соответственно нулю или единице. Сформированный таким способом сигнал может быть передан непосредственно на выходной элемент (реле,

контактор, электромагнит, электромагнитную муфту и т.д.) без применения специального запоминающего элемента.

В качестве примера спроектируем дискретную систему управления электрифицированной агрегатной головкой (рис. 5.25) с использованием непрерывных логических функций.

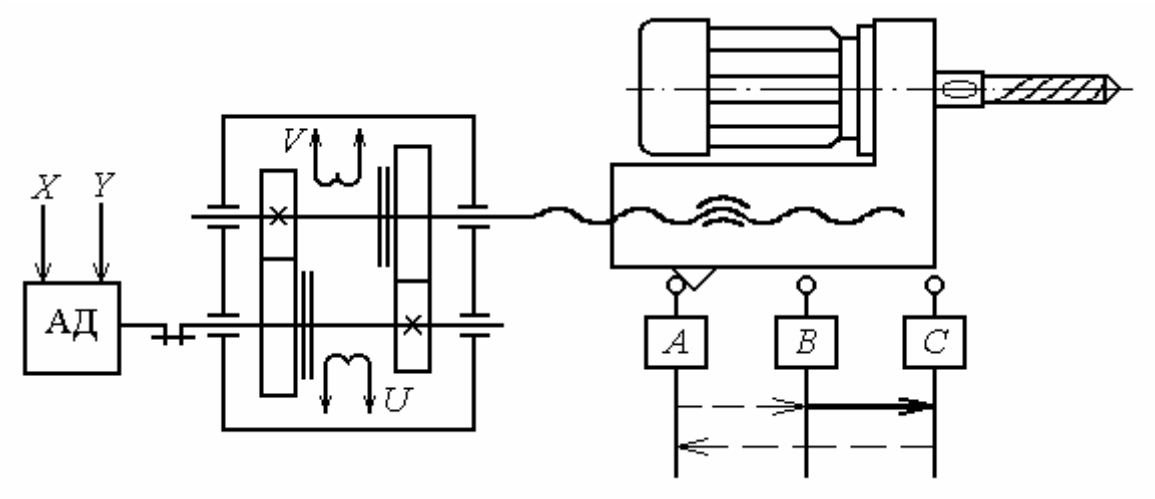


Рис. 5.25. Электрифицированная агрегатная головка:

АД – асинхронный двигатель; *U* – электромагнитная муфта быстрого хода;
V – электромагнитная муфта медленной (рабочей) подачи;
A, *B*, *C* – путевые выключатели; *x*, *y* – сигналы «вперед» и «назад» управления вращением ротора электродвигателя.

Цикл работы головки удобно представить в виде следующей условной записи:

$A \longrightarrow B$ Быстрый подвод;
$B \longrightarrow C$ Рабочая подача;
$C \longrightarrow A$ Быстрый отвод.

Общая схема дискретной системы управления показана на рис. 5.26.

На вход дискретного автомата системы управления поступают сигнал *P* («Пуск») и сигналы *a*, *b*, *c* с путевых выключателей *A*, *B*, *C*. К выходу дискретного автомата подключены электрические аппараты: катушки контакторов *X*, *Y* и катушки электромагнитных муфт *U*, *V*.

Включенное или выключенное состояние электрических аппаратов определяются логическими функциями f_x, f_y, f_u и f_v. Катушки этих аппаратов не обладают свойством запоминания поданных сигналов. Поэтому, если в соответствии с циклограммой некоторый электрический аппарат должен быть включен в течение *n*-го количества тактов, то во всех этих *n* тактах значение соответствующей логической функции должно быть равно единице. В этом заключается принципиальное отличие методики синтеза схем на основе

непрерывных логических функций от методики синтеза схем, содержащих на выходе элементы памяти (триггеры).

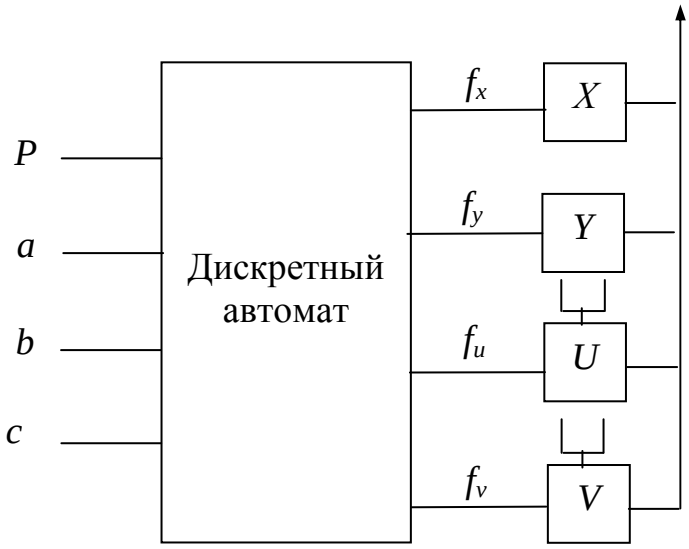


Рис. 5.26. Общая схема дискретной системы управления

С учетом данного замечания составляем таблицу включений дискретной системы управления.

Таблица включений:

1.	$a = 1$	Если $P = 1$, то $f_x = 1$, $f_u = 1$	Вперед быстро
2.	$a = 0$	$f_x = 1$, $f_u = 1$	Вперед быстро
3.	$b = 1$	$f_x = 1$, $f_v = 1$	Вперед медленно
4.	$b = 0$	$f_x = 1$, $f_v = 1$	Вперед медленно
5.	$c = 1$	$f_y = 1$, $f_u = 1$	Назад быстро
6.	$c = 0$	$f_y = 1$, $f_u = 1$	Назад быстро
7.	$b = 1$	$f_y = 1$, $f_u = 1$	Назад быстро
8.	$b = 0$	$f_y = 1$, $f_u = 1$	Назад быстро

Заметим, что в правой части таблицы включений теперь показаны не изменения состояний, как это мы делали раньше, а сами состояния логических функций.

В соответствии с таблицей включений строим начальную циклограмму (рис. 5.27).

Вводим в систему управления элементы памяти, причем с целью уменьшения их числа в качестве первого элемента памяти используем выходной элемент x. Это означает, что элемент x будет одновременно выполнять две роли: роль выходного элемента и роль элемента памяти (рис. 5.28).

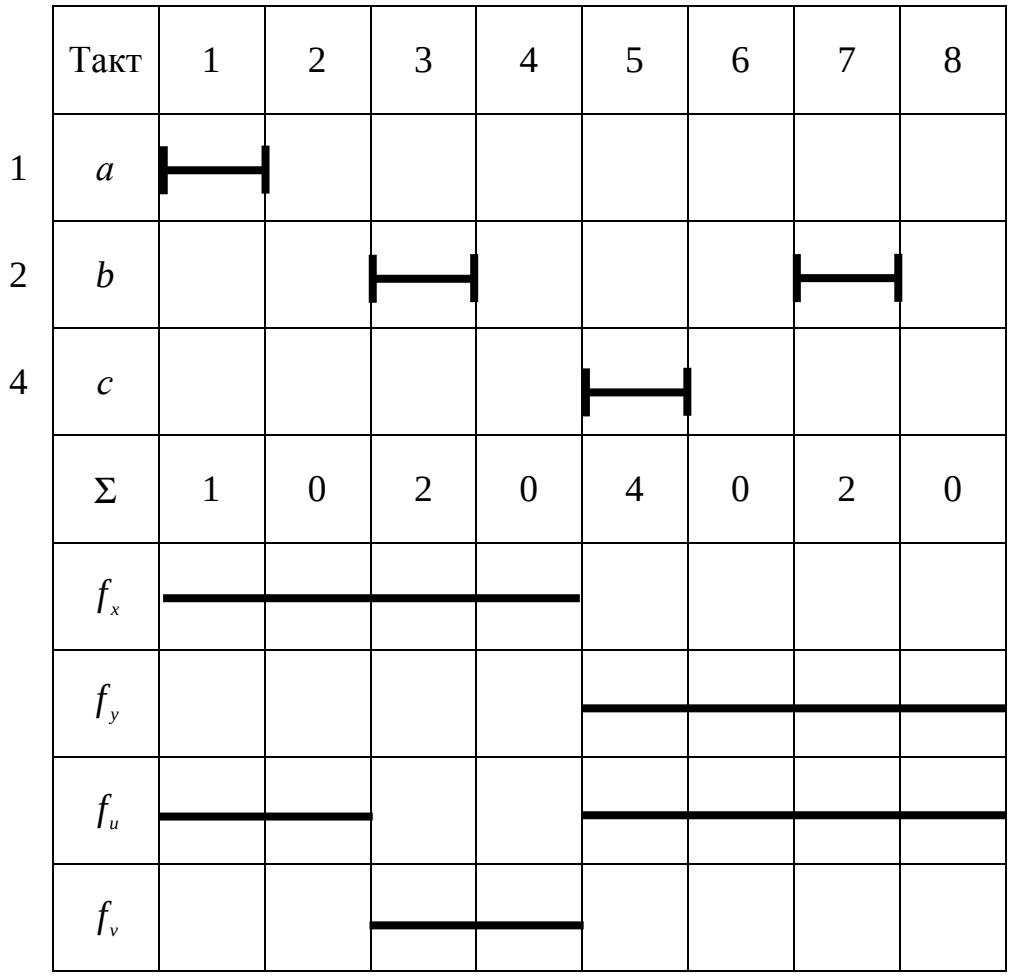


Рис. 5.27. Начальная циклограмма

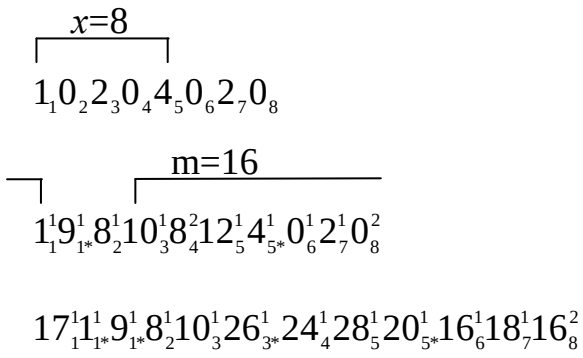


Рис. 5.28. Введение в систему управления элементов памяти

В 6-м и в 8-м тактах циклограммы повторяется состояние "16" входных переменных. Однако в этих же тактах повторяется состояние и выходных переменных. Поэтому такое повторение состояний входных переменных допустимо.

Переходим к построению реализуемой циклограммы (рис. 5.28).

	Такт	1	1*	1*	2	3	3*	4	5	5*	6	7	8
1	a	—	—	—									
2	b					—	—					—	
4	c								—	—			
8	x			—	—	—	—	—	—				
16	m	—					—	—	—	—	—	—	—
	Σ	17	1	9	8	10	26	24	28	20	16	18	16
	f_x		P	—	—	—	—	—					
	f_y									—	—	—	—
	f_u			—	—	—				—	—	—	—
	f_v						—	—	—				
	f_m					—	—	—	—	—	—	—	—

Рис. 5.28. Реализуемая циклограмма

Нижняя часть циклограммы построена по следующим правилам:

1) В первую очередь отобразить состояния логических функций, которые определяют моменты включения и выключения элементов памяти (в данном примере это функции f_x и f_m), выходные сигналы (x и m) которых уже обозначены в верхней части циклограммы.
2) Скопировать из начальной циклограммы состояния остальных логических функций, причем если сигнал на включение или выключение рабочего органа приходится на такт, который в реализуемой циклограмме имеет один или несколько дополнительных тактов с тем же номером, но обозначенных символом «*», то следует выбирать последний в очереди дополнительный такт.

Исключением из этого правила является первый такт. При наличии дополнительных тактов 1* сигнал на включение рабочего органа следует подавать в последнем такте 1*, а на выключение – в такте 1.

Приступаем к минимизации логических функций. На рис. 5.29 показано расположение на карте Карно используемых конституент.

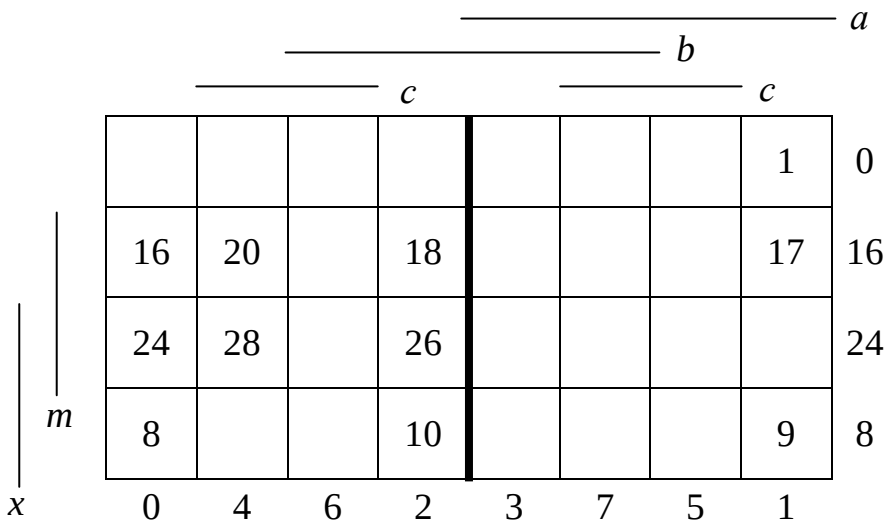


Рис. 5.29. Расположение на карте Карно используемых конституент

Процедура минимизации всех логических функций последовательно изображена на рис. 5.30.

Чтобы реализовать систему управления электрифицированной агрегатной головкой на бесконтактных логических элементах **И**, **ИЛИ**, **НЕ**, необходимо заменить обозначения непрерывных логических функций f_x, f_y, f_u, f_v, f_m соответствующими переменными x, y, u, v, m. В результате получим следующие логические уравнения:

$$x = p\overline{m} + \overline{c}x$$

$$y = \overline{a}\,\overline{x}$$

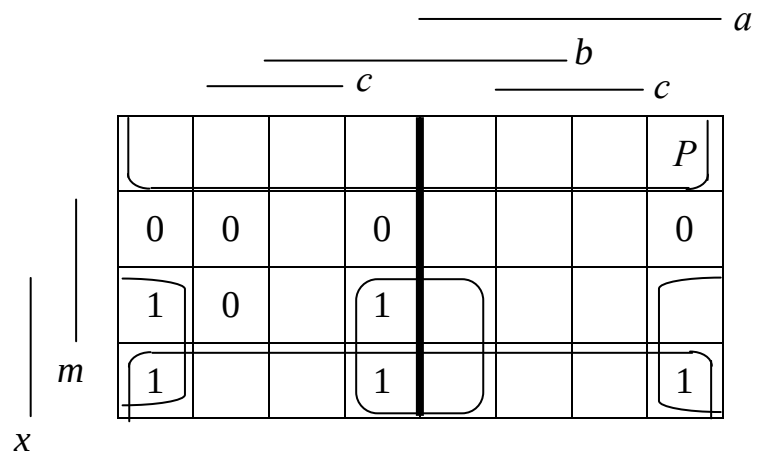
$$u = x\overline{m} + \overline{a}\,\overline{x} = x\overline{m} + y$$

$$v = xm$$

$$m = \overline{a}(b + \overline{m})$$

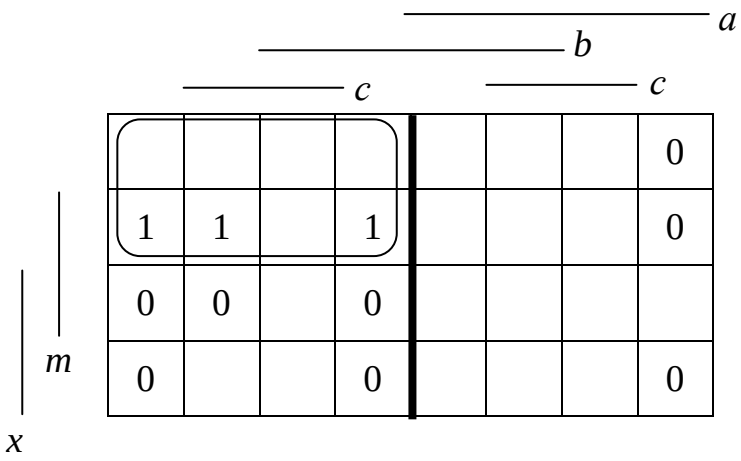
На основании данных уравнений строим схему на бесконтактных логических элементах (рис. 5.31).

а) $f_x = P \cdot 1,\ 9,\ 8,\ 10,\ 26,\ 24$ $\quad \overline{f}_x = 17,\ 28,\ 20,\ 16,\ 18$



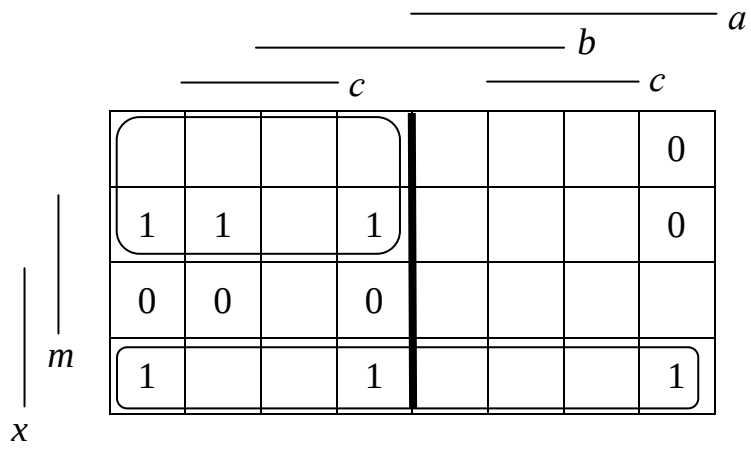
$f_x = P\overline{m} + \overline{c}x$

б) $f_y = 20,\ 16,\ 18$ $\quad \overline{f}_y = 17,\ 1,\ 9,\ 8,\ 10,\ 26,\ 24,\ 28$



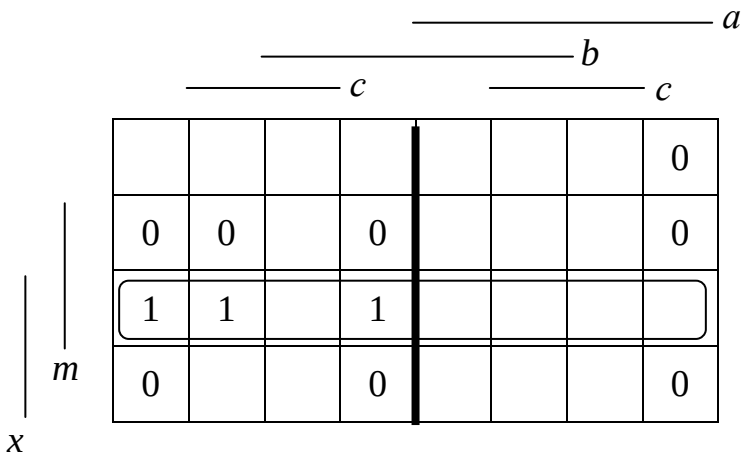
$f_y = \overline{\overline{a}x}$

в) $f_u = 9,\ 8,\ 10,\ 20,\ 16,\ 18$ $\quad \overline{f}_u = 17,\ 1,\ 26,\ 24,\ 28$



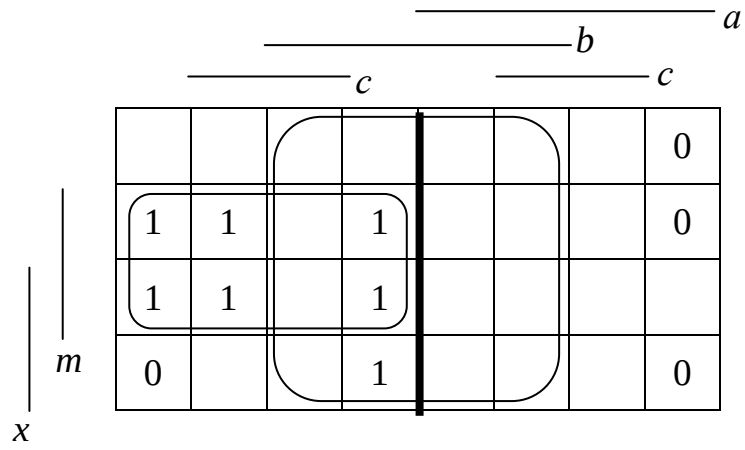
$f_u = x\overline{m} + \overline{\overline{a}x}$

г) $f_м = 26,\ 24,\ 28$ $\qquad \overline{f}_м = 17,\ 1,\ 9,\ 8,\ 10,\ 20,\ 16,\ 18$



$f_м = xm$

д) $f_m = 10,\ 26,\ 24,\ 28,\ 20,\ 16,\ 18$ $\qquad \overline{f}_m = 17,\ 1,\ 9,\ 8$



$f_m = b + \overline{a}m$

или равносильный по числу переменных вариант

$f_m = \overline{a}b + \overline{a}m = \overline{a}(b + m)$

Рис. 5.30. Процедура минимизации логических функций

Недостаток систем управления с непрерывными логическими функциями – возможность возникновения в таких системах состязаний цепей. Чтобы выявить состязания, полезно дискретную систему управления представить в релейно-контактном исполнении (рис. 5.32).

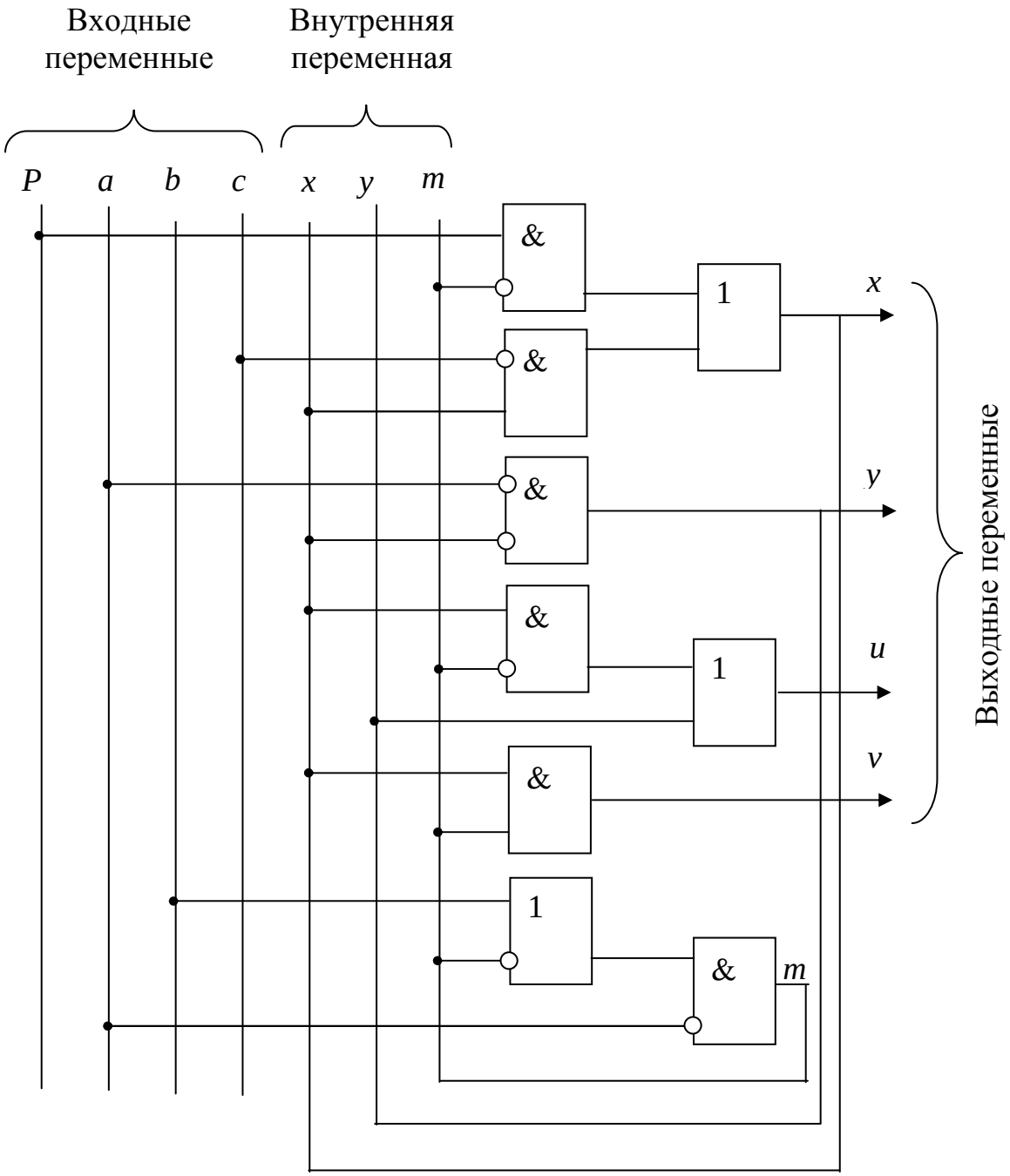


Рис. 5.31. Система управления электрифицированной агрегатной головкой на бесконтактных элементах

Анализ её работы показывает, что в данной системе управления критических состязаний не происходит. Однако результат будет иным, если несколько изменить структуру системы. Назначим включение элемента памяти (m = 16) в 3-м такте, а выключение - в 7-м такте. В результате получим второй вариант реализуемой циклограммы (рис. 5.33).

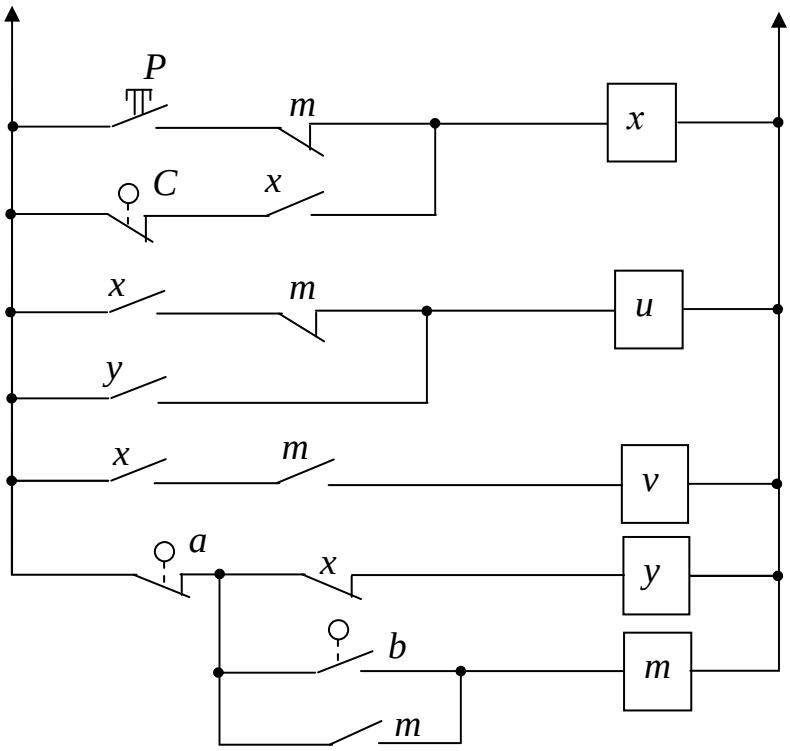


Рис. 5.32. Релейно-контактная система управления электрифицированной агрегатной головкой

	Такт	1	1*	2	3	3*	4	5	5*	6	7	7*	8
1	a	—	—										
2	b				—	—					—	—	
4	c							—	—				
8	x		—	—	—	—	—	—					
16	m					—	—	—	—	—	—		
	Σ	1	9	8	10	26	24	28	20	16	18	2	0
	f_x	P	—	—	—	—	—						
	f_y								—	—	—	—	—
	f_u		—	—	—				—	—	—	—	—
	f_v					—	—	—					
	f_m				—	—	—	—	—	—			

Рис. 5.33. Второй вариант реализуемой циклограммы

После минимизации логические функции f_x, f_y, f_u и f_v принимают такой же вид, как и в первом варианте.

Минимизируем последнюю логическую функцию f_m (рис. 5.34).

$f_m = 10,\ 26,\ 24,\ 28,\ 20,\ 16$ $\qquad$ $\overline{f}_m = 1,\ 9,\ 8,\ 18,\ 2,\ 0$

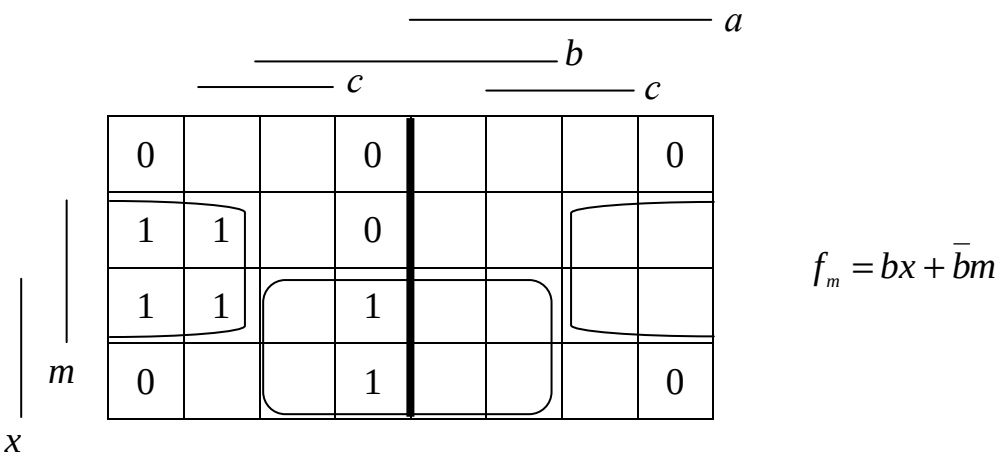


Рис. 5.34. Минимизация функции f_m

Рассмотрим полученную функцию f_m в релейно-контактном исполнении (рис. 5.35).

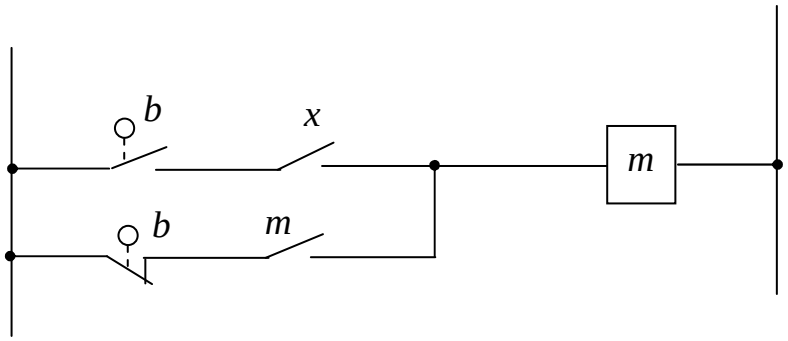


Рис. 5.35. Релейно-контактное исполнение функции F_m

В данной схеме могут быть критические состязания. Действительно, если в момент выключения путевого выключателя *b* его замыкающий контакт разомкнет цепь раньше, чем замкнется размыкающий контакт, то на катушке реле появится нулевой всплеск напряжения.

Критическое состязание цепи можно устранить двумя способами:

1) применить известную равносильность вида

$$xa + \bar{x}b = xa + \bar{x}b + ab;$$

2) применить для формирования внутренней переменной запоминающий элемент в виде статического триггера.

Рассмотрим первый способ. Преобразуем логическую функцию f_m в вид:

$$f_m = bx + \bar{b}m = bx + \bar{b}m + xm .$$

В результате приходим к схеме (рис. 5.36), в которой критических состязаний нет.

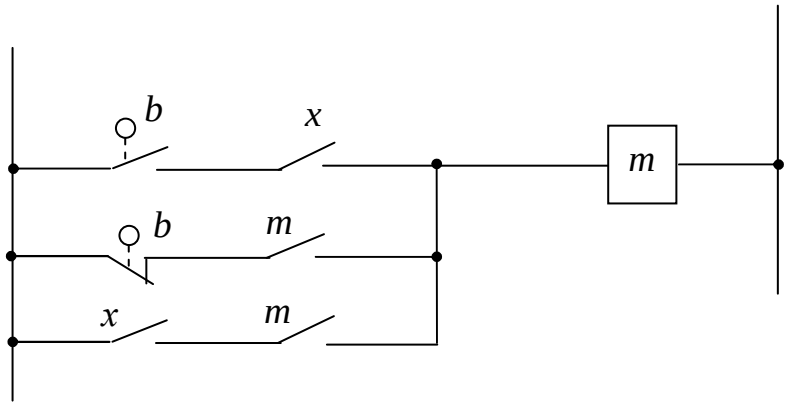


Рис. 5.36. Устранение критических состязаний в схеме

Второй способ сводится, в сущности, к замене непрерывной логической функции прерывистой. Обозначим в нашей задаче импульсную функцию через F_m и найдем её (рис. 5.37).

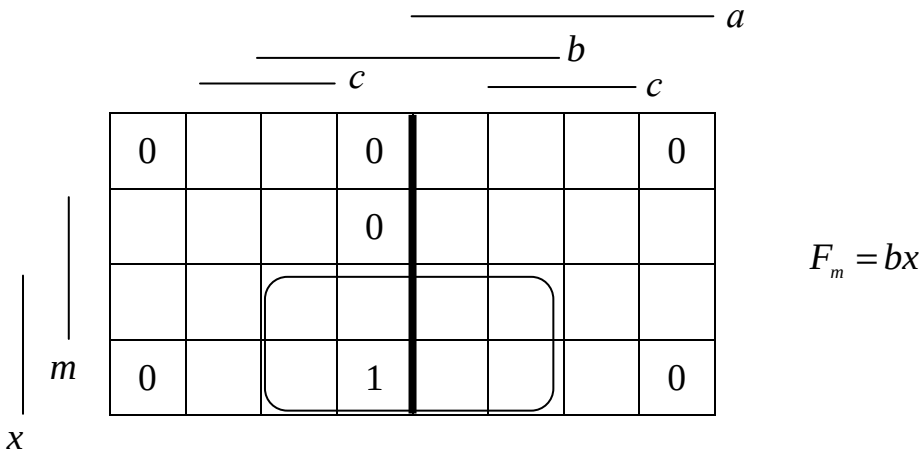
Существует переход от схемы с использованием статических триггеров к эквивалентной (в логическом смысле) релейно-контактной схеме. В свою очередь, релейно-контактную схему легко перевести на бесконтактные логические элементы.

Методика указанных преобразований рассматривается в следующем параграфе.

5.8. Особенности синтеза релейно-контактных систем управления

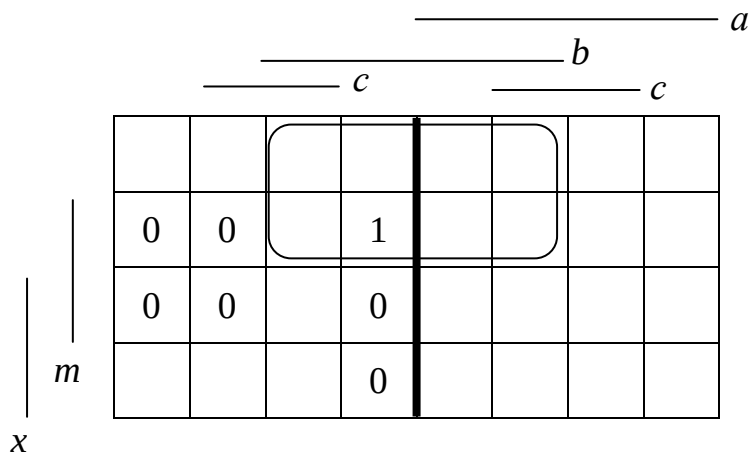
В настоящее время системы управления на электромеханических реле применяются редко. Вместе с тем языки релейно-контактных схем широко распространены при программировании логических контроллеров. Это объясняется тем, что релейные структуры имеют определенные преимущества перед схемами на бесконтактных логических элементах. В частности, в релейных схемах легче анализировать последовательность протекания автоматического цикла, обнаруживать явления «гонок» и «состязаний» и др.

а) $F_m = 10$ $\quad\quad$ $\overline{F}_m = 1,\ 9,\ 8,\ 18,\ 2,\ 0$



$F_m = bx$

б) $F_{\bar{m}} = 18$ $\quad\quad$ $\overline{F}_{\bar{m}} = 10,\ 26,\ 24,\ 28,\ 20,\ 16$



$F_{\bar{m}} = b\bar{x}$

Рис. 5.37. Минимизация импульсной логической функции

Таким образом, релейно-контактный вариант системы управления можно рассматривать как промежуточную модель, которую затем легко перевести на бесконтактные логические элементы.

Синтез релейно-контактных систем управления имеет свои особенности, которые мы рассмотрим на примере дискретной системы управления гидрофицированной агрегатной головкой (рис. 5.38).

Автоматический цикл управления агрегатной головкой имеет следующую последовательность:

$A \longrightarrow B$ Быстрый подвод;
$B \longrightarrow C$ Рабочая подача;
$C \longrightarrow A$ Быстрый отвод.

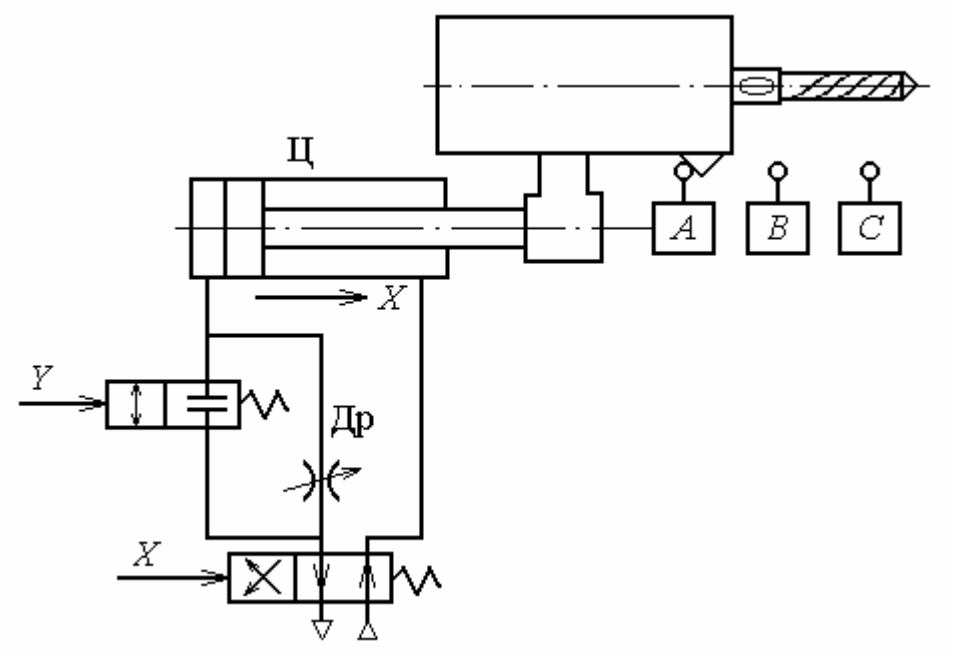


Рис. 5.38. Гидрофицированная агрегатная головка: Ц – гидравлический цилиндр; Др – дроссель; *A*, *B*, *C* – путевые выключатели

Направление движения агрегатной головки можно задавать с помощью гидравлического распределителя (сигнал *x*), причем движения штока цилиндра **Ц** вперед и назад ограничены жесткими упорами.

Для регулирования скорости рабочей подачи предусмотрен дроссель **Др**, который во время быстрых перемещений шунтируется другим гидравлическим распределителем (сигнал *y*).

Чтобы предотвратить в дискретной системе управления состязания цепей, целесообразно на первоначальном этапе синтеза использовать импульсные логические функции в сочетании с запоминающими элементами в виде статических *RS*-триггеров (рис. 5.39).

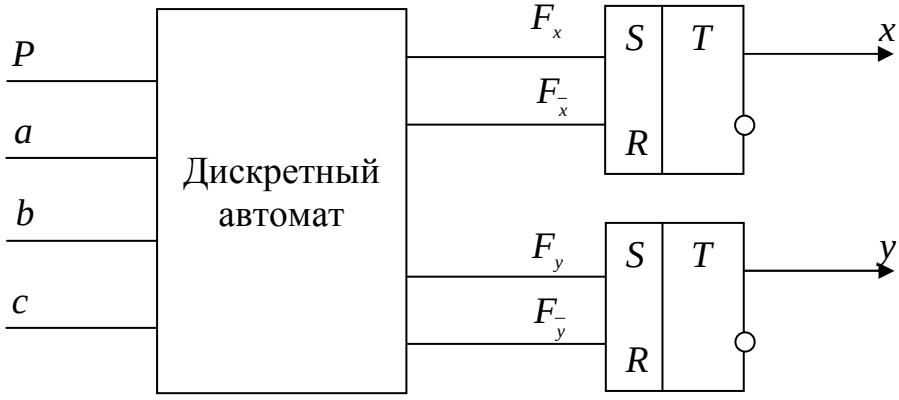


Рис. 5.39. Первоначальная схема системы управления гидрофицированной агрегатной головкой

Таблица включений дискретной системы управления имеет следующий вид:

Таблица включений.

1.	$a = 1$	Если $P = 1$, то $F_x = 1$, $F_u = 1$	Вперед быстро
2.	$a = 0$		
3.	$b = 1$	$F_{\bar{y}} = 1$	Вперед медленно
4.	$b = 0$		
5.	$c = 1$	$F_{\bar{x}} = 1$; $F_y = 1$	Назад быстро
6.	$c = 0$		
7.	$b = 1$		
8.	$b = 0$		

В соответствии с таблицей включений построена начальная циклограмма (рис. 5.40).

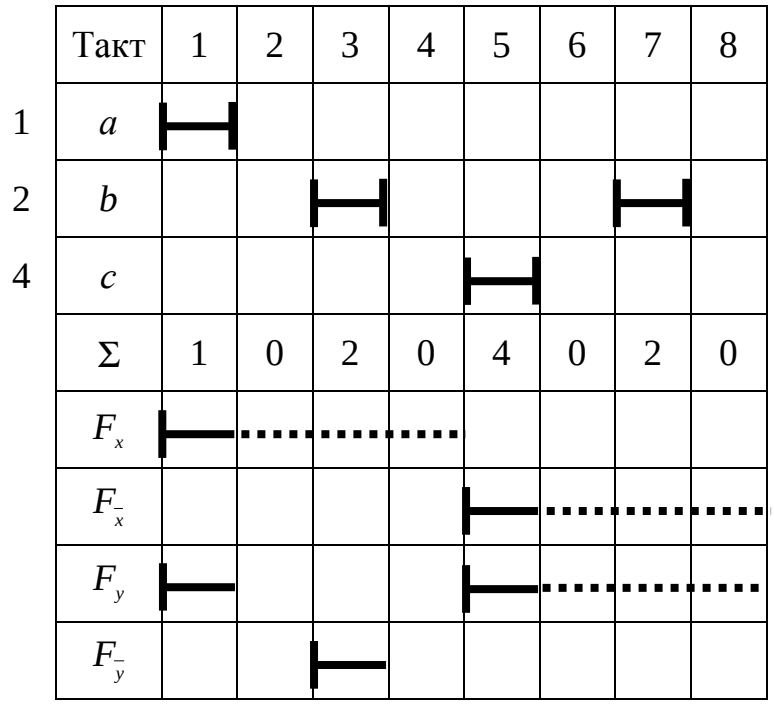


Рис. 5.40. Начальная циклограмма гидрофицированной агрегатной головки

Заметим, что повторение веса «0» входных переменных в 6-м и в 8-м тактах допустимо, так как в этих же тактах повторяется и состояние выходных переменных.

С целью сокращения числа элементов памяти повторение веса «0» во 2-м и в 4-м тактах также искусственно сделано допустимым за счет того, что в указанных тактах логические функции F_y и $F_{\bar{y}}$ имеют запрещенные состояния.

В результате для получения реализуемой циклограммы достаточно применить только один элемент памяти, роль которого выполняет выходной элемент x (рис. 5.41).

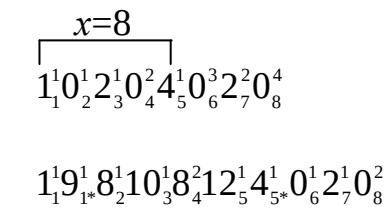


Рис. 5.41. Введение в дискретную систему управления элемента памяти

В соответствии с начальной циклограммой построена реализуемая циклограмма (рис. 5.42).

	Такт	1	1*	2	3	4	5	5*	6	7	8
1	a										
2	b										
4	c										
8	x										
	Σ	1	9	8	10	8	12	4	0	2	0
	F_x										
	$F_{\bar{x}}$										
	F_y										
	$F_{\bar{y}}$										

Рис. 5.42. Реализуемая циклограмма гидрофицированной агрегатной головки

Чтобы облегчить минимизацию логических функций, на рис. 5.43 показано расположение используемых конституент на карте Карно.

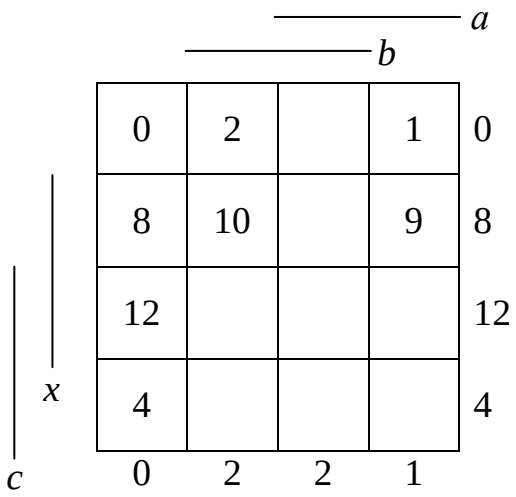


Рис. 5.43. Расположение конституент на карте Карно

В процессе минимизации (рис. 5.44) получены выражения импульсных логических функций.

Запоминающие элементы в виде статических триггеров заменим электромеханическими реле. Чтобы определить правила такой замены, рассмотрим некоторый триггер и некоторое реле, которые формируют один и тот же выходной сигнал x (рис. 5.45).

Из рисунка видно, что цепь включения реле (включающая цепь) реализует функцию F_x, а цепь выключения реле (выключающая цепь) – функцию $x \cdot \overline{F_x}$. Общая функция включения реле определяется по формуле

$$f_x = F_x + x \cdot \overline{F_x},$$

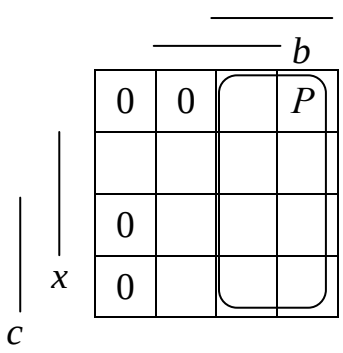
где F_x – функция включения триггера;

$\overline{F_x}$ – инверсная функция выключения триггера;

x – замыкающий контакт реле x.

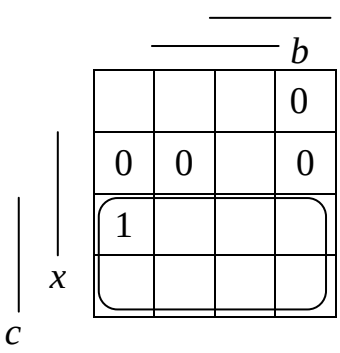
Важно отметить, что функция F_x включающей цепи выражена в дизъюнктивной нормальной форме (ДНФ) и при состязаниях цепей может создавать только нулевые всплески, а функция $x \cdot \overline{F_x}$ выключающей цепи представляет собой конъюнктивную нормальную форму (КНФ) и поэтому может создавать только единичные всплески (5.6). В том и в другом случае возможные состязания цепей в комбинационных схемах не нарушают нормальной работы реле.

а) $F_x = P \cdot 1$; $\overline{F_x} = 12,\ 4,\ 0,\ 2$



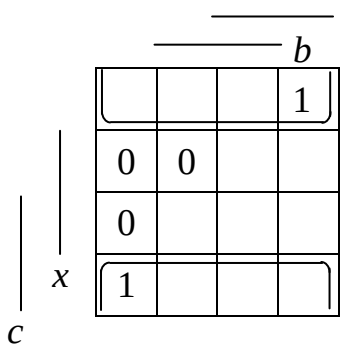
$F_x = P \cdot a$

б) $F_{\bar{x}} = 12$; $\overline{F_{\bar{x}}} = 1,\ 9,\ 8,\ 10$



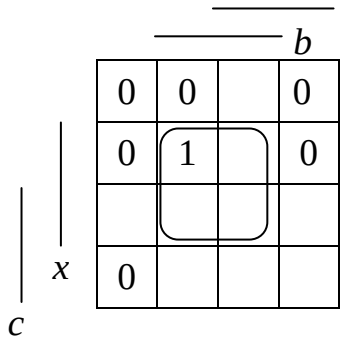
$F_{\bar{x}} = c$

в) $F_y = 1,4$; $\overline{F_y} = 10,\ 8,\ 0,\ 12$



$F_y = \bar{x}$

г) $F_{\bar{y}} = 10$; $\overline{\overline{F_y}} = 1,\ 9,\ 8,\ 4,\ 0,\ 2$



$F_{\bar{y}} = bx$

Рис. 5.44. Минимизация логических функций с помощью карт Карно

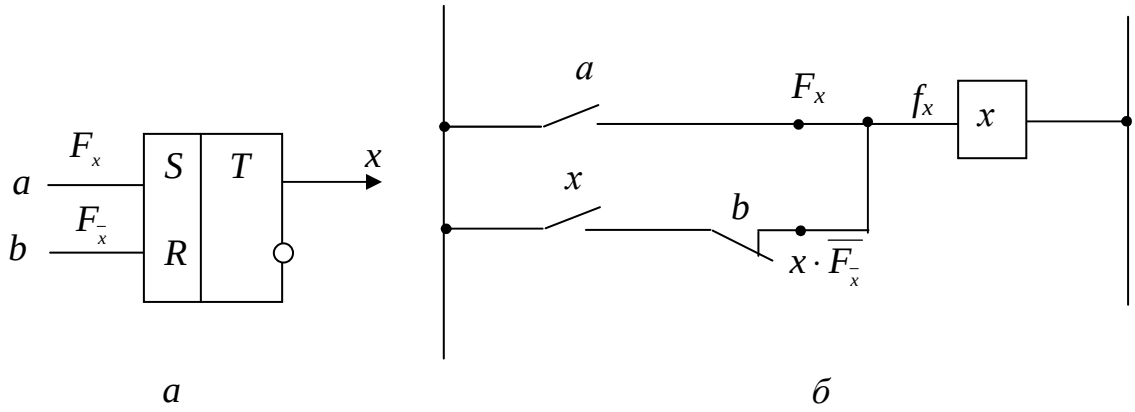


Рис. 5.45. Запоминающие элементы:
а – на основе триггера; *б* – на основе реле

С учетом изложенного запишем логические функции включения реле, которые управляют движением гидрофицированной агрегатной головкой.

$$f_x = F_x + x\overline{\bar{F}_x} = P \cdot a + x\bar{c};$$

$$f_y = F_y + y\overline{\bar{F}_y} = \bar{x} + y\bar{b}\bar{x} = \bar{x} + y(\bar{b} + \bar{x}) = \bar{x}(1 + y) + y\bar{b} = \bar{x} + y\bar{b}.$$

Теперь реализуем полученные логические функции в виде релейно-контактной схемы (рис. 5.46).

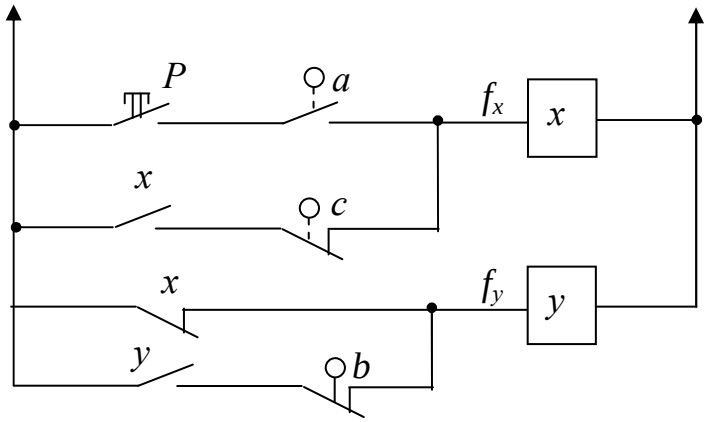


Рис. 5.46. Релейно-контактная схема управления
гидрофицированной агрегатной головкой

Особенность логических функций f_x и f_y в том, что они непрерывные, а прототипы, на основе которых получены эти функции, – импульсные.

Если требуется перейти от контактного варианта системы управления к бесконтактному на элементах **И**, **ИЛИ**, **НЕ**, то достаточно в выражениях логических функций заменить символы f_x и f_y соответствующими переменными x и y:

$$x = Pa + x\bar{c};$$

$$y = \bar{x} + y\bar{b}.$$

На основании данных уравнений строим бесконтактную схему управления гидрофицированной агрегатной головкой (рис. 5.47).

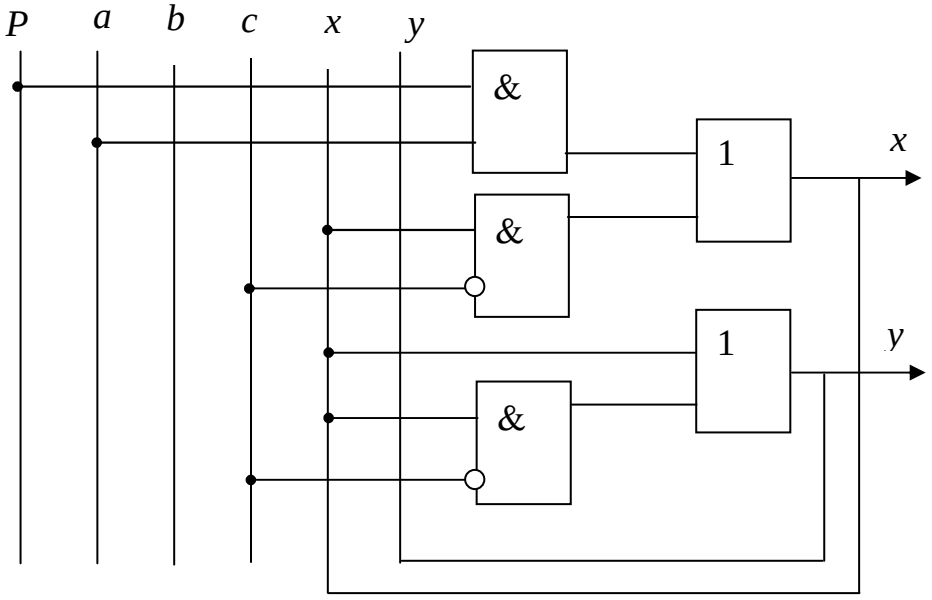


Рис. 5.47. Бесконтактная схема управления гидрофицированной агрегатной головкой

В отличие от первоначальной схемы (рис. 5.39) данная схема не содержит в явном виде статические *RS*-триггеры.

5.9. Построение дискретных систем управления в базисе элементов И – НЕ

Среди интегральных микросхем серии ТТЛ наиболее распространен активный элемент **И – НЕ**. Наиболее удобным методом построения схем на элементах **И – НЕ** является предварительный синтез системы управления в базисе **И**, **ИЛИ**, **НЕ**, а затем переход к базису **И – НЕ** по приведенному ниже правилу.

Если имеется готовая структура в базисе **И**, **ИЛИ**, **НЕ**, то можно перейти к структуре в базисе **И – НЕ** с помощью графических преобразований, т. е. не прибегая к алгебраической записи логических функций. Указанные преобразования следует выполнять в следующем порядке:

1) Составляется исходная схема, в которой инвертируются отдельные внутренние связи, входы и выходы:

а) инвертируются все внутренние связи между одноименными элементами **И** и **ИЛИ**;

б) инвертируются все входы, подаваемые непосредственно на элементы **ИЛИ**;

в) инвертируются все выходы, идущие с элементов **И**.

2) Составляется промежуточная схема, в которой учтены инверсии, произведенные во внутренних, входных и выходных цепях.

3) Составляется окончательная схема, в которой все элементы **И** и **ИЛИ** заменяются на элементы **И – НЕ** с соответствующим числом входов.

В качестве примера приведем к базису элементов **И – НЕ** систему управления электрифицированной агрегатной головкой (рис. 5.31).

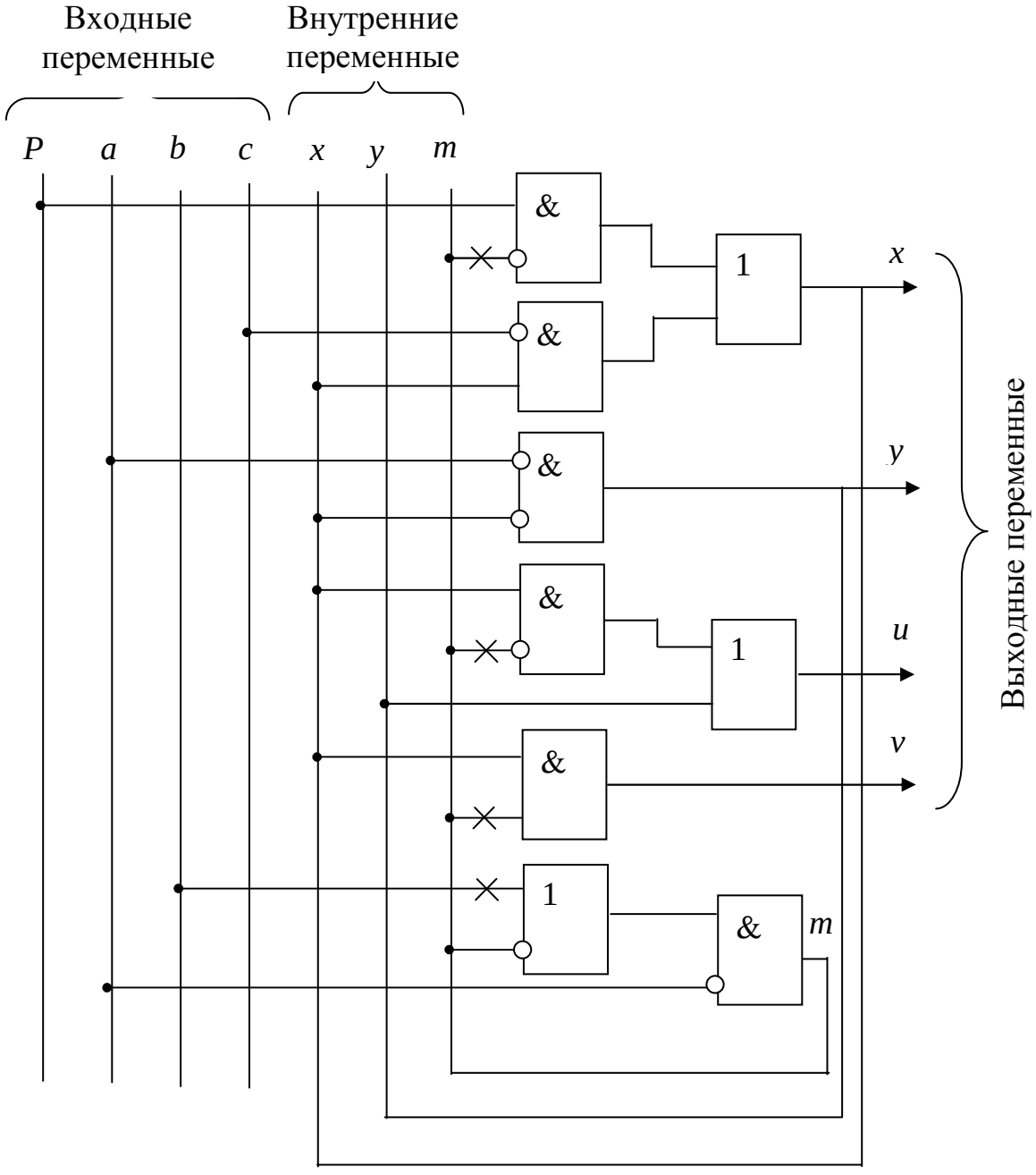


Рис. 5.48. Исходная схема

От исходной схемы (рис. 5.48), на которой крестами отмечены связи, требующие инвертирования, переходим к промежуточной схеме (рис. 5.49), а затем к структуре (рис. 5.50).

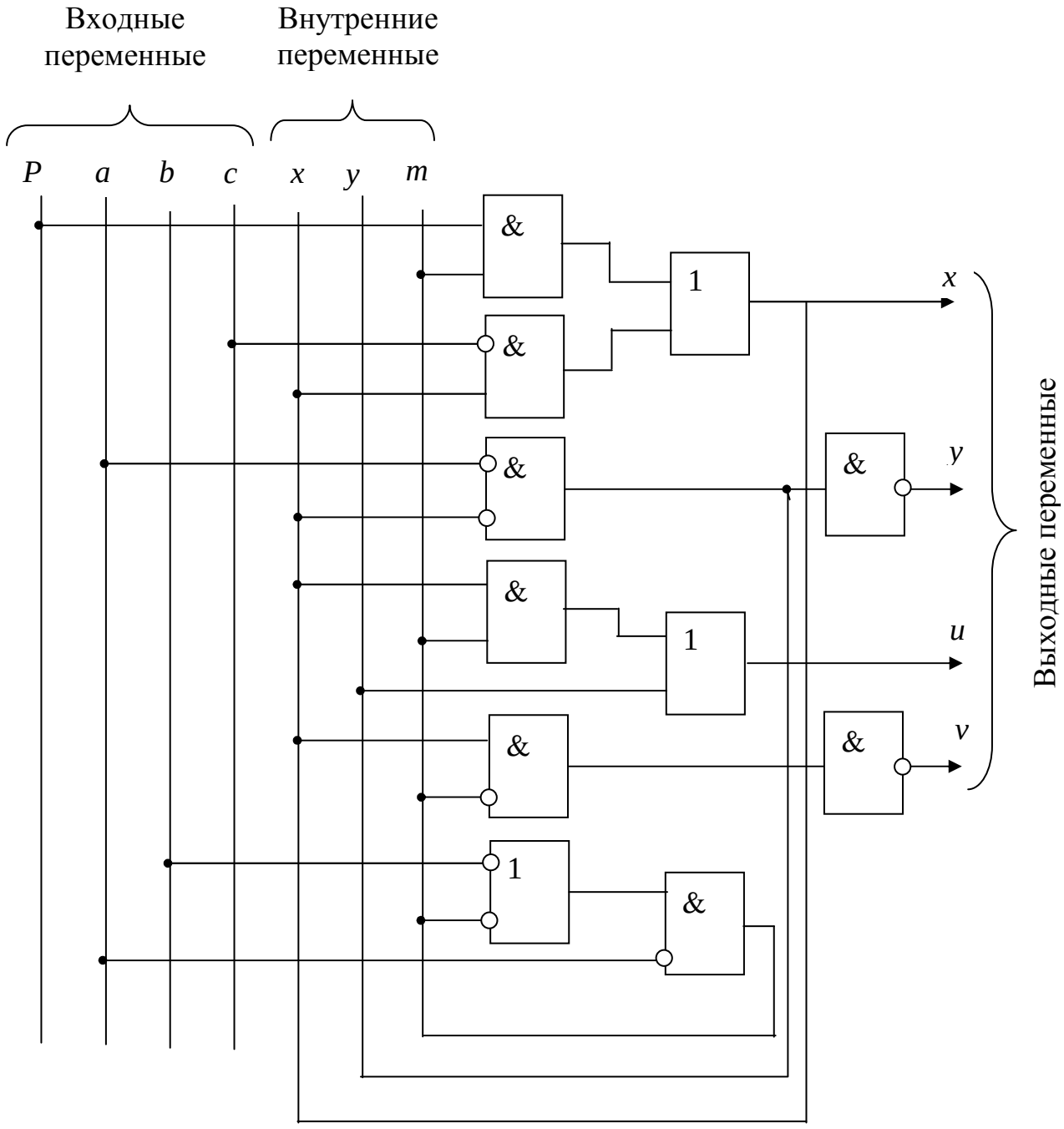


Рис. 5.49. Промежуточная схема

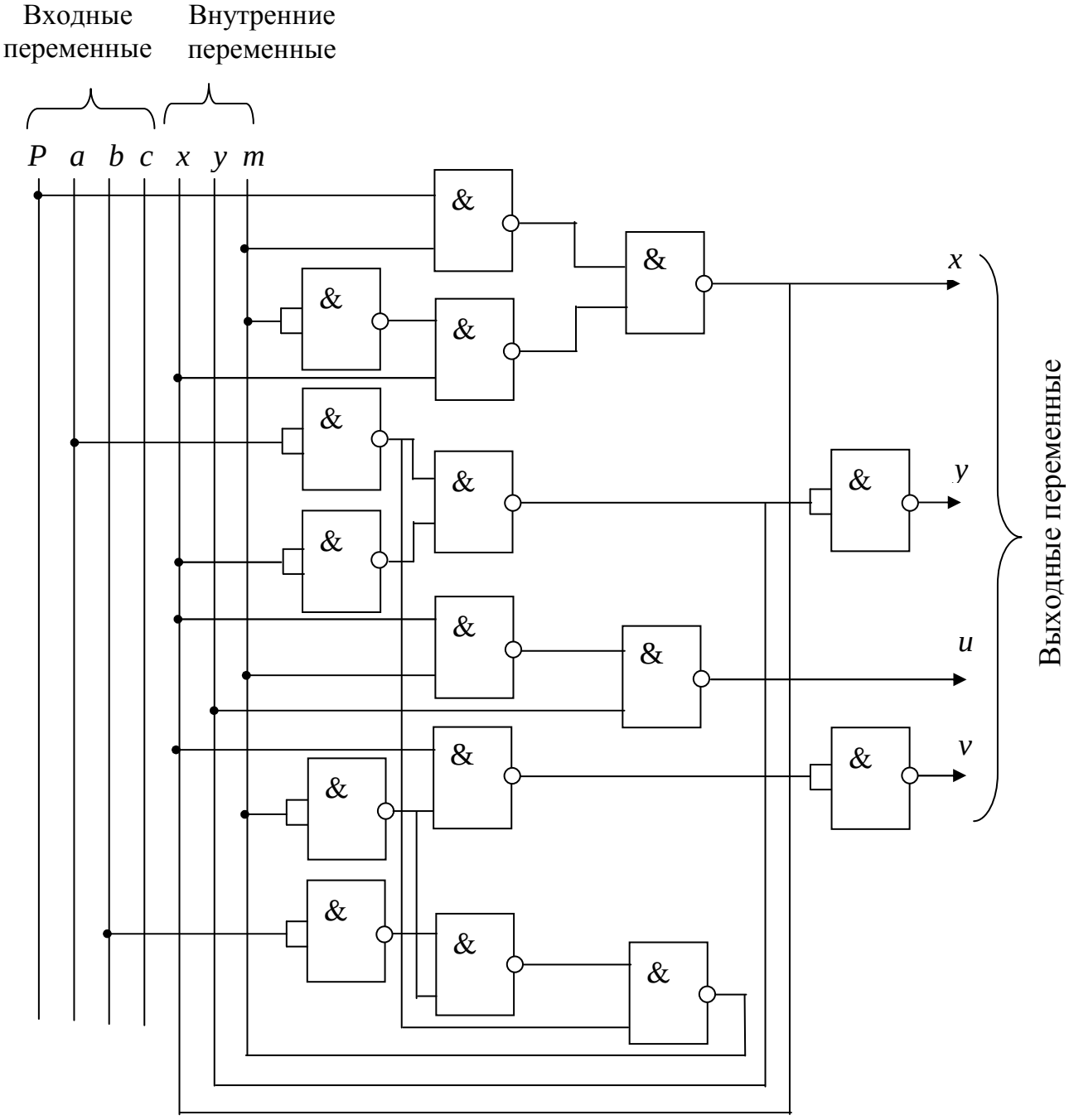


Рис. 5.50. Окончательная схема

6. СИНТЕЗ СИСТЕМ УПРАВЛЕНИЯ СО СЛОЖНЫМИ ЦИКЛАМИ

6.1. Постановка задачи

Современные дискретные системы управления машиностроительным оборудованием (агрегатные станки и станки с ЧПУ, автоматические линии, транспортно-накопительные устройства, автоматические склады и т.п.) отличаются повышенной сложностью и поэтому часто реализуются на основе

программируемых логических контроллеров (ПЛК), причем система управления, как правило, разбивается на отдельные, более простые подсистемы.

Логический синтез дискретных систем управления на основе ПЛК принципиально не отличается от синтеза твердотельных дискретных систем. В реальном проекте необходимо учесть следующие основные факторы:

1) Большую (от десятков до сотен переменных) размерность систем управления.

2) Условия безаварийного функционирования технологических установок.

3) Нештатные режимы работы исполнительных органов и сервисные функции.

Перечисленные факторы следует рассматривать в сочетании с тем, что сложные дискретные системы управления промышленными объектами состоят из отдельных подсистем (или циклов), которые включаются в общий цикл работы последовательно, параллельно или последовательно-параллельно (рис. 6.1).

Сложность проектирования систем управления, содержащих параллельные подсистемы, заключается в том, что при одновременном запуске двух или более параллельных подсистем часто бывает неизвестно, какая из них закончит работу раньше.

В циклограммах работы механизмов, которые мы рассматривали выше, данная задача не ставилась. Поэтому обратимся теперь к изучению методики синтеза дискретных систем управления, содержащих последовательные, параллельные и последовательно-параллельные циклы.

Рассмотрим сначала методику синтеза дискретных систем управления с последовательными циклами.

6.2. Методика синтеза дискретных систем управления с последовательными циклами

Последовательные циклы дискретной системы управления представляют собой автономные подсистемы. С поступлением разрешающей команды на запуск в работу включается 1-я подсистема. Завершив работу, она передает команду на запуск 2-й подсистемы и т.д.

Каждая подсистема имеет специальный триггер управления, который показывает состояние подсистемы. Если триггер управления включен – значит подсистема активна, а если триггер управления выключен – значит подсистема находится в состоянии ожидания. Алгоритм функционирования всех триггеров управления идентичен.

Разбиение проектируемой системы управления на отдельные подсистемы в определенной мере субъективно, т.е. проектировщик сам должен определить, какие механизмы или группы механизмов должны войти в ту или иную подсистему. Обычно этот вопрос решают исходя из законченности конструктивного оформления какой-то части объекта управления (часто такую

часть называют модулем или агрегатом) или из законченности части технологического процесса.

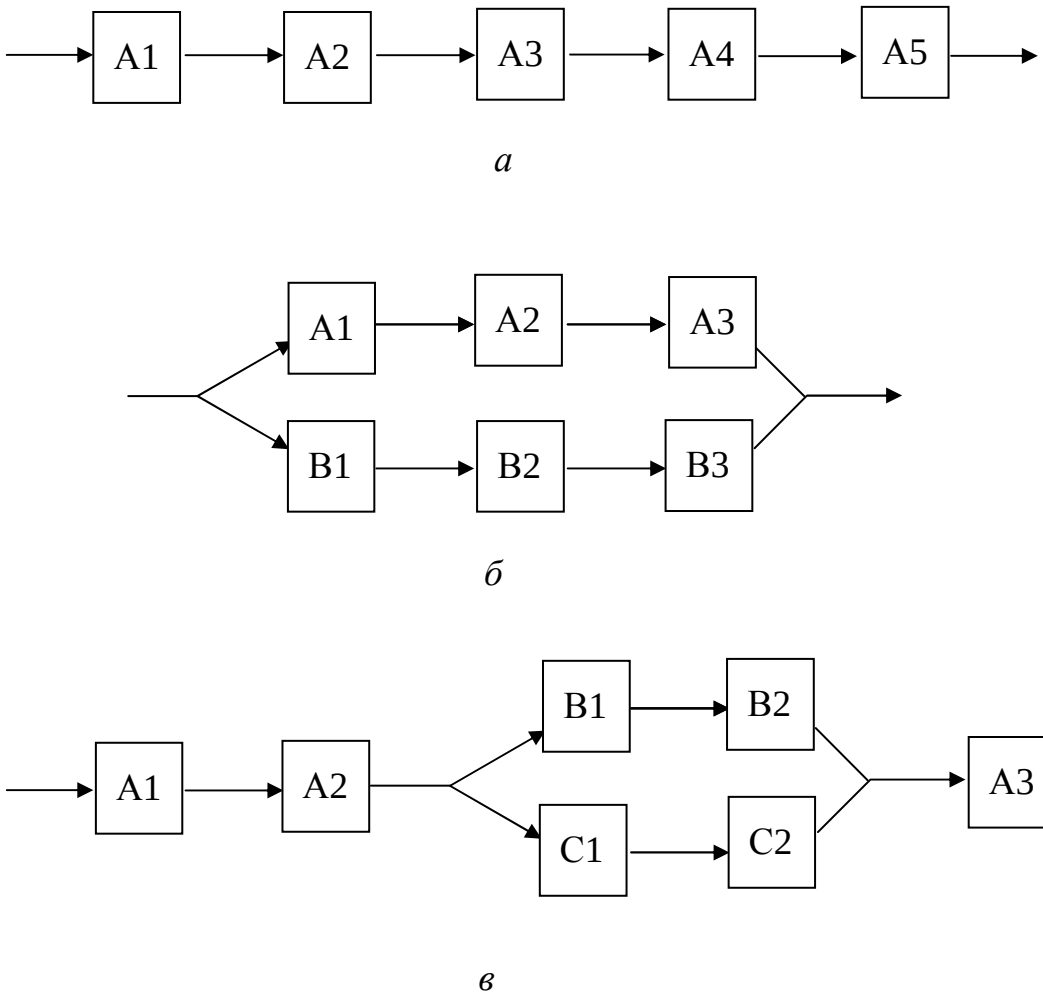


Рис. 6.1. Последовательность работы подсистем : *а* – последовательная; *б* – параллельная; *в* – последовательно-параллельная

Количество исполнительных органов и датчиков в подсистеме определяет число входных и выходных переменных. Чем больше этих переменных, тем сложнее становится циклограмма работы механизмов и тем труднее ее анализировать. В сложной подсистеме могут возникнуть проблемы, связанные с безопасностью и надежностью ее работы, особенно в аварийных ситуациях. Поэтому, исходя из практики, максимальное число входных переменных в подсистеме не рекомендуют увеличивать больше 10 – 12.

Разбивая систему управления на отдельные подсистемы, необходимо учитывать следующие критерии:

1. Выделенная подсистема должна иметь завершенный цикл работы, т.е. по окончании автоматического цикла любая подсистема должна возвратиться в исходное состояние.

2. Цикл работы подсистемы может быть приостановлен один или несколько раз. Для продолжения работы приостановленной подсистемы на ее вход необходимо подать дополнительную команду на запуск.

3. Циклы работы подсистем могут быть вложенными один в другой любое количество раз.

Последнее означает, что любая подсистема может приостановить цикл своей работы и передать управление другой подсистеме, а та, в свою очередь, приостановив свой цикл работы, может запустить третью подсистему и т.д. Завершить свою работу подсистемы могут в обратной последовательности (рис. 6.2, *а*), либо в какой-то иной (рис. 6.2, *б*).

Теоретически любую сколь угодно сложную систему можно представить в виде подсистемы и вложить ее цикл работы в цикл другой, еще более сложной системы.

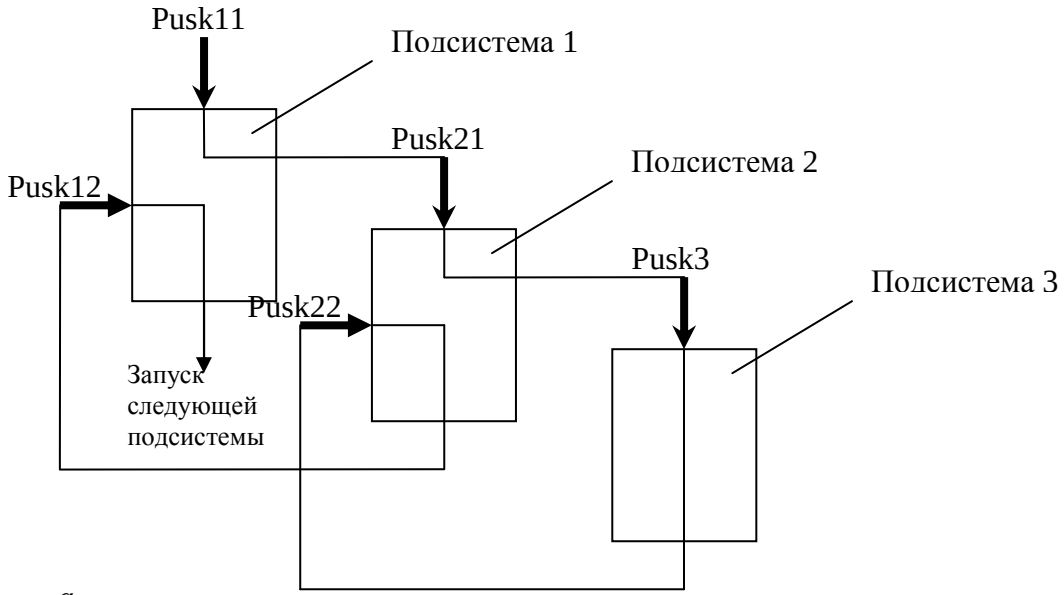
Важно подчеркнуть, что каждая подсистема, несмотря на сложность всей системы, в которую она входит, проектируется автономно, т.е. независимо от остальных подсистем.

Изучим методику синтеза дискретной системы управления с последовательными циклами на примере механизма автоматической смены инструмента (МАСИ) станка с ЧПУ (рис. 6.3).

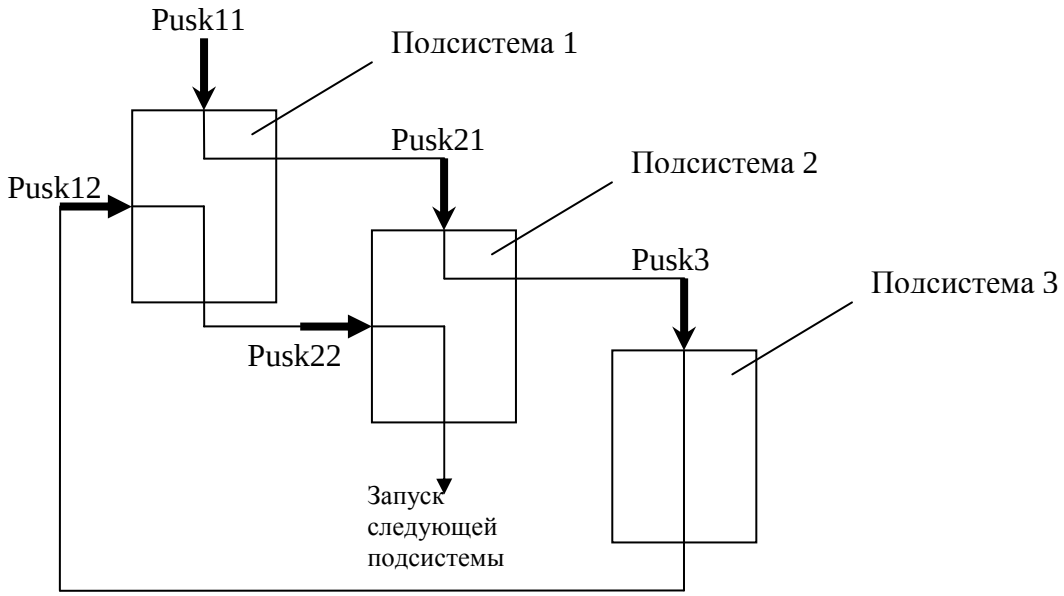
МАСИ имеет два автооператора (захватных устройства) ЗУ1 и ЗУ2. Первое захватное устройство ЗУ1, совершая вращательное и поступательное движения, переносит отработавший инструмент из шпинделя (Ш) в промежуточное загрузочное место (ПЗМ), а новый инструмент – из ПЗМ в Ш.

Вращательное движение ЗУ1 происходит при помощи электродвигателя Д, а поступательное – посредством гидравлического цилиндра Ц1. В шпинделе инструмент зажат пружиной. Для ее отжатия и освобождения инструмента служит цилиндр Ц2.

Второе захватное устройство ЗУ2 может совершать два поступательных движения – горизонтальное и вертикальное, причем в горизонтальном направлении положение механизма контролируется в 3-х точках (1, 2 и 3), а в вертикальном – в 2-х. Привод ЗУ2 гидравлический (цилиндры Ц3 и Ц4).



а



б

Рис. 6.2. Схемы запуска подсистем с вложенными циклами

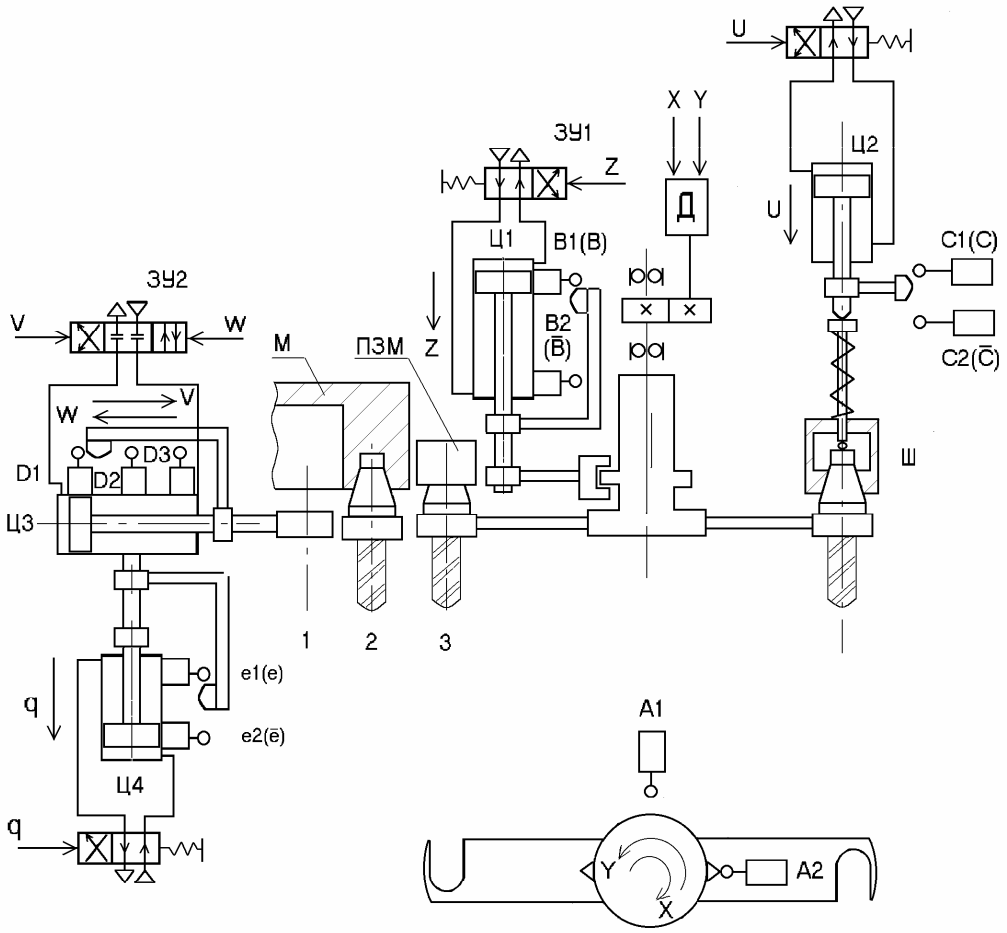


Рис. 6.3. Структурно-кинематическая схема устройства автоматической смены инструмента. Д – двигатель; М – магазин; ПЗМ – промежуточное загрузочное место; Ш – шпиндель; (Ц1-Ц4) - гидравлические цилиндры

Рассмотренную последовательность работы МАСИ удобно представить в виде следующей условной записи:

1-я подсистема:

ЗУ1 (+90°); ← И1 → ; ЗУ1 (↓); ЗУ1 (+180°);
ЗУ1 (↑); °); → И2 ← ; ЗУ1(-90°);

2-я подсистема:

ЗУ2 (1 → 3); ЗУ2 (↓); ЗУ2 (3 → 2); ЗУ2 (↑); ЗУ2 (2 → 1);

3-я подсистема:

Работа магазина М (в данном примере подробно не рассматривается);

2-я подсистема:

ЗУ2 (1 → 2); ЗУ2 (↓); ЗУ2 (2 → 3); ЗУ2 (↑); ЗУ2 (3 → 1).

Из данного алгоритма видно, что работа 2-й подсистемы, которая управляет захватным устройством ЗУ2, состоит из двух частей и за время отработки всего цикла смены инструмента эта подсистема запускается дважды.

Каждая из рассмотренных подсистем синтезируется автономно, т.е. независимо от других.

а) *Синтез подсистемы № 1*

Первая подсистема управляет всеми манипуляциями захватного устройства ЗУ1, а также зажимом и разжимом инструмента в шпинделе Ш. Назовем эту подсистему условно подсистемой ЗУ1. Исполнительными устройствами первой подсистемы являются гидравлические цилиндры Ц1, Ц2 и реверсивный электродвигатель Д. В качестве датчиков обратной связи используются путевые выключатели. Выключатели $A1$ и $A2$ контролируют вращательные движения захватного устройства ЗУ1, а выключатели $B1$ и $B2$ – поступательные. Положения штока цилиндра Ц2, который разжимает инструмент в шпинделе Ш, контролируются выключателями $C1$ и $C2$.

В соответствии с заданным алгоритмом работы запишем таблицу включений для 1-й подсистемы.

На основании таблицы включений строим начальную циклограмму работы подсистемы ЗУ1 (рис. 6.4).

Чтобы превратить начальную циклограмму в реализуемую, вводим в подсистему ЗУ1 элемент памяти (рис. (6.5).

На рис. 6.6 представлена реализуемая циклограмма подсистемы ЗУ1.

Эта циклограмма отличается от ранее рассматривавшихся следующими особенностями:

1. В нижней части циклограммы (под чертой) показаны состояния триггера управления $T1$, который управляет работой данной подсистемы и формирует сигнал P_{21} для запуска второй подсистемы.

2. Чтобы в момент включения питания произвести инициализацию выходных элементов (триггеров), состояния всех выходных переменных в 1-м такте приняты обязательными.

1-я подсистема

Таблица включений

1.	a_1=1	$F_{\bar{y}}$=1. Если P_1=1, то F_x=1	+ 90°
2.	a_1=0		
3.	a_2=1	$F_{\bar{x}}$=1, F_u=1	Разжим
4.	c=0	F_z=1	Вниз
5.	b=0	F_x=1	+ 180°
6.	a_2=0		
7.	a_1=1		
8.	a_1=0		
9.	a_2=0	$F_{\bar{x}}$=1, $F_{\bar{z}}$=1	Вверх
10.	b=1	$F_{\bar{u}}$=1	Зажим
11.	c=1	F_y=1	- 90°
12.	a_2=0		

Рассмотрим реализуемую циклограмму подсистемы ЗУ1. В момент включения питания принудительно выключаются выходные элементы памяти *X, Y, Z, U*. В результате электродвигатель поворота захватного устройства ЗУ1 оказывается выключенным, а штоки цилиндров Ц1 и Ц2 приходят в исходные положения (если они там не были).

При отсутствии сигнала на запуск первой подсистемы ($P1=0$, $\overline{P1}=1$) элемент памяти $M1$ выключен и сигнал m_1 = 0. Триггеры управления всех подсистем в момент инициализации также сбрасываются.

В момент подачи сигнала «ПУСК» ($P1=1$, $\overline{P1}=0$) формируется сигнал F_{m1} = 1 на включение внутреннего элемента памяти $M1$. С его выхода снимается сигнал m_1 = 1 и система управления переходит в новое состояние (такт 1*), когда сумма входных переменных равна 29. В этом такте включается триггер управления $T1$ и формируется сигнал F_x = 1 на включение поворота захватного устройства ЗУ1.

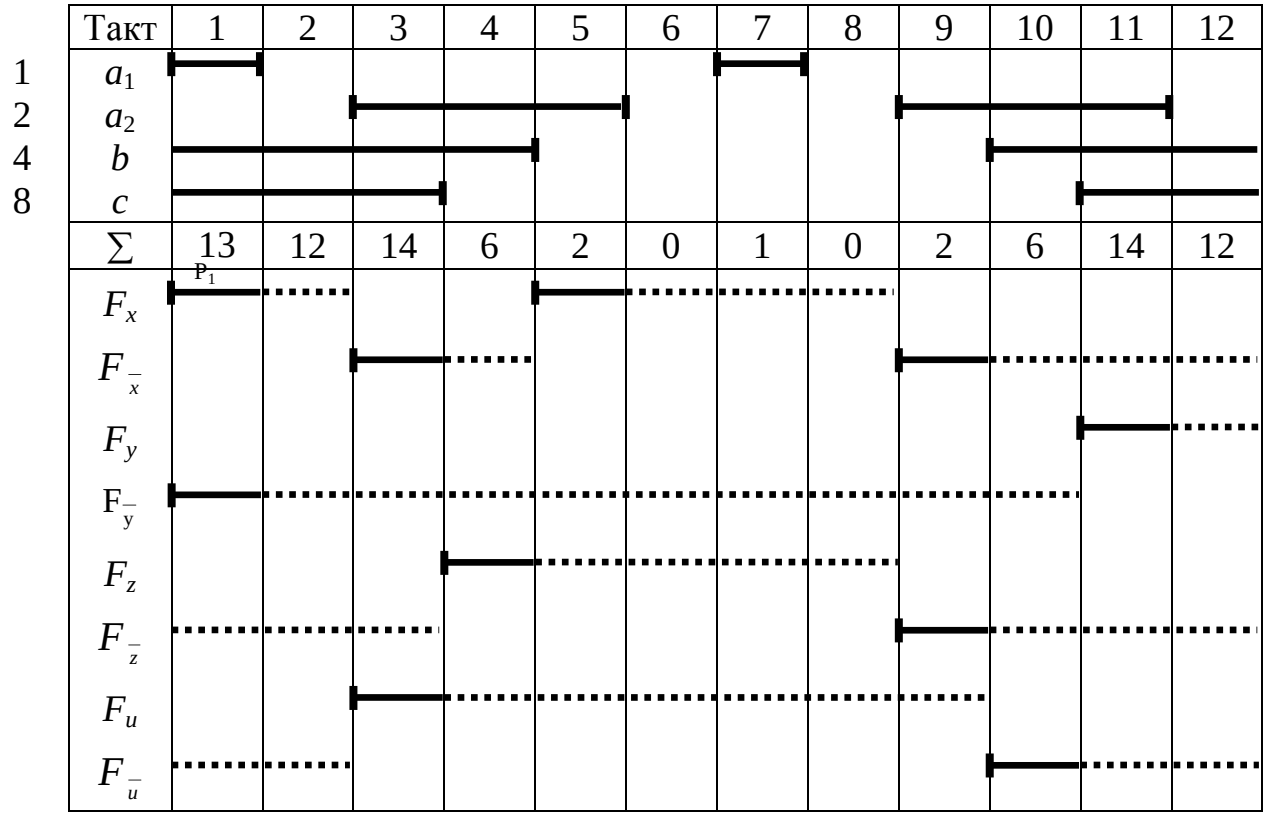


Рис. 6.4. Начальная циклограмма подсистемы ЗУ1

$m_1=16$

$$13^{1}_{1}12^{1}_{2}14^{1}_{3}6^{1}_{4}2^{1}_{5}0^{1}_{6}1^{1}_{7}0^{2}_{8}2^{2}_{9}6^{2}_{10}14^{2}_{11}12^{2}_{12}$$

$$13^{1}_{1}29^{1}_{1*}28^{1}_{2}30^{1}_{3}22^{1}_{4}18^{1}_{5}16^{1}_{6}17^{1}_{7}1^{1}_{7*}0^{1}_{8}2^{1}_{9}6^{1}_{10}14^{1}_{11}12^{1}_{12}$$

Рис. 6.5. Введение элементов памяти в подсистему ЗУ1

Далее продолжается автоматический цикл работы первой подсистемы по заданному алгоритму, причем триггер управления $T1$ все это время включен. Важно подчеркнуть, что *дополнительные такты на включение и на выключение триггера управления Т1 в циклограмме не отводятся*.

По окончании цикла работы 1-я подсистема приходит в начальное состояние (такт 1), а сумма весов входных переменных становится равной 13. К этому времени сигнал пуска $P1 = 0$ и автоматический цикл завершается. Поскольку триггер управления в это время все еще включен, то сигнал блокировки $T1 = 1$ и в результате в такте **1** формируется сигнал $P_{21} = 1$, который предназначен для запуска 2-й подсистемы. Она запускается и триггер управления 2-й подсистемы включается. Сигнал обратной связи с выхода этого триггера устанавливается в единичное состояние ($T2 = 1$) и тем самым

разрешается сброс триггера управления в первой подсистеме. Формируется сигнал $F_{\overline{T1}}=1$, который и сбрасывает триггер управления $T1$ в первой подсистеме. Это означает, что завершилась передача управления от первой подсистемы ко второй.

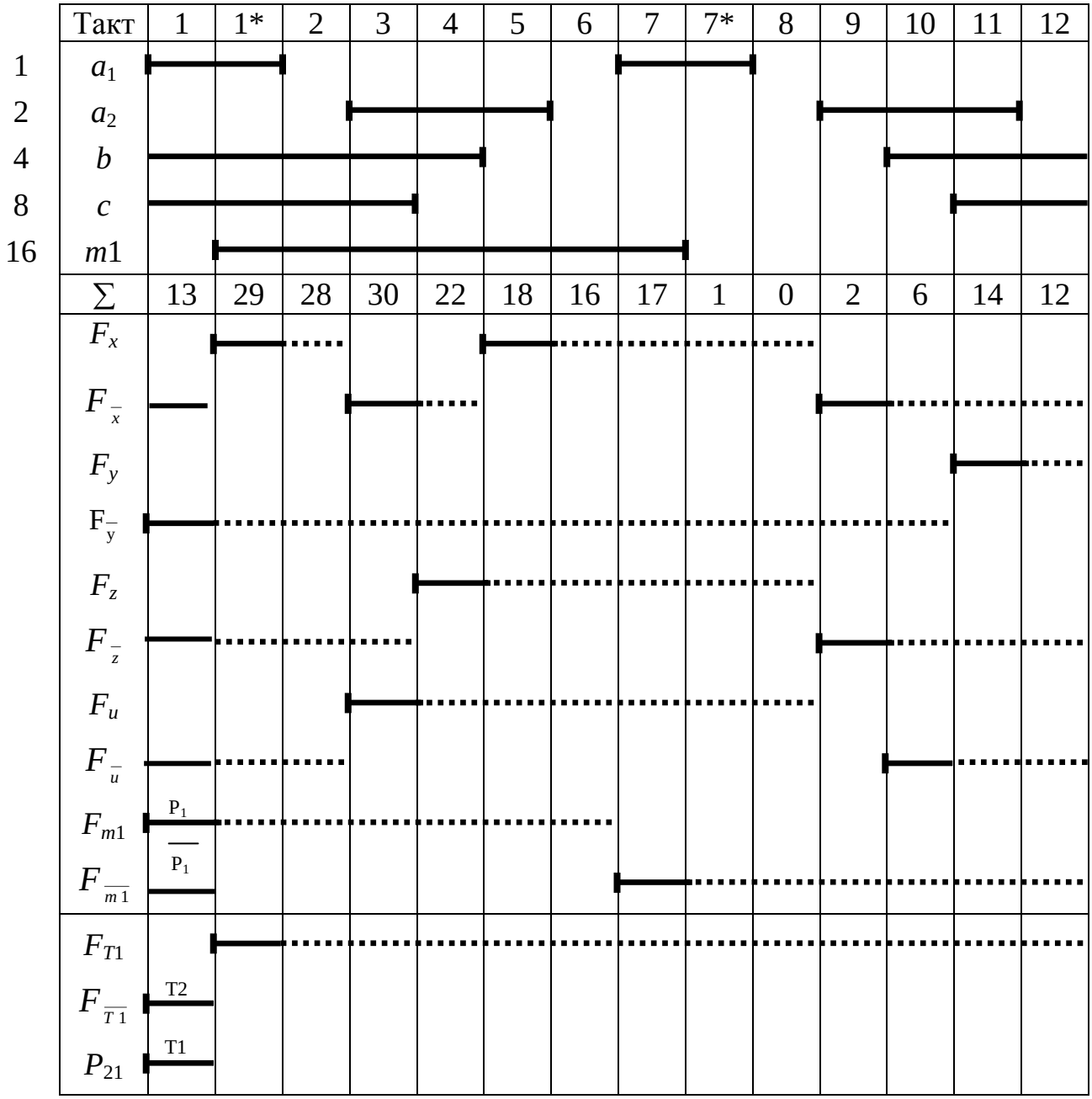


Рис. 6.6. Реализуемая циклограмма подсистемы ЗУ1

Приступаем к минимизации логических функций подсистемы ЗУ1. Схема расположения используемых конституент на карте Карно представлена на рис. 6.7.

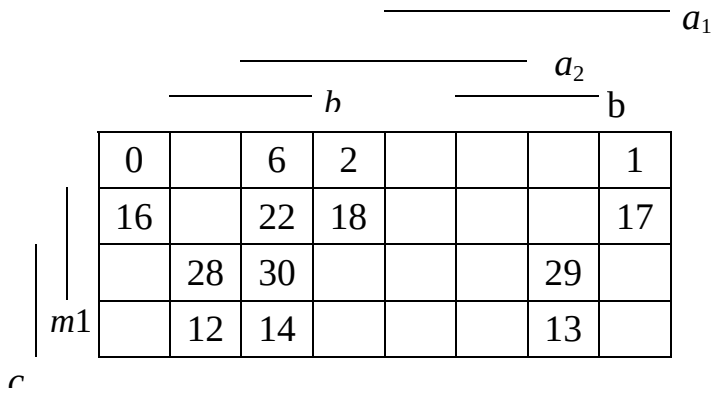
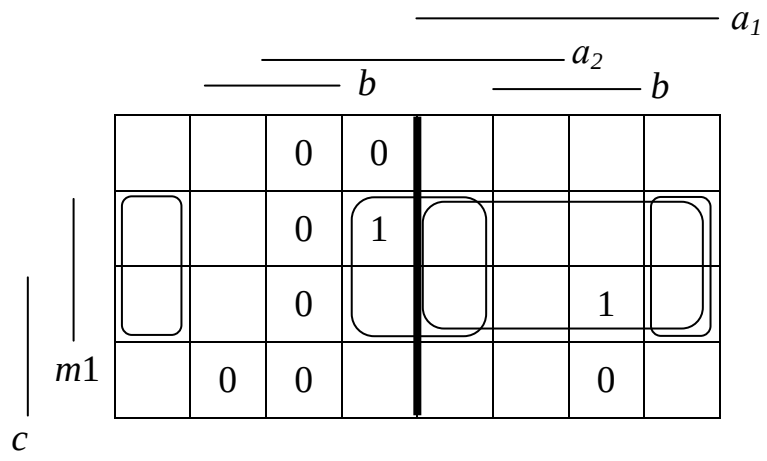


Рис. 6.7. Карта используемых конституент в подсистеме ЗУ1

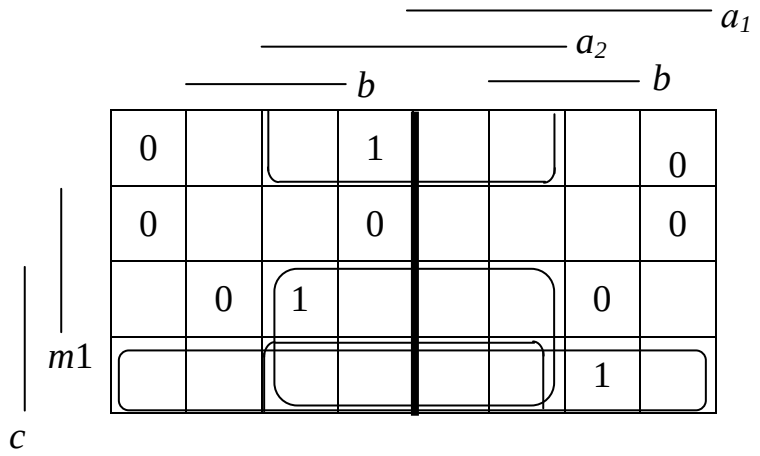
Далее минимизируем каждую логическую функцию с помощью карт Карно (рис.6.8).

а) $F_x = 29,\ 18$ $\qquad \overline{F}_x = 30,\ 22,\ 2,\ 6,\ 14,\ 12,\ 13$



$$F_x = m1 \cdot a_1 + \bar{b} \cdot m1$$

б) $F_{\bar{x}} = 13,\ 30,\ 2$ $\qquad \overline{F}_{\bar{x}} = 29,\ 28,\ 18,\ 16,\ 17,\ 1,\ 0$



$$F_{\bar{x}} = a_2\overline{m_1} + a_2 c + c\overline{m_1} =$$
$$= a_2\overline{m_1} + a_2 c_1 + c_1\overline{m_1}$$

Рис. 6.8. Минимизация логических функций 1-й подсистемы

Минимизируя аналогично оставшиеся функции, получаем:

$$F_y = a_2 c \overline{m_1}$$

$$F_{\bar{y}} = a_1$$

$$F_z = \bar{c} m_1$$

$$F_{\bar{z}} = a_2 \overline{m_1} + c$$

$$F_u = a_2 m_1$$

$$F_{\bar{u}} = b \overline{m_1}$$

$$F_{m1} = P_1 a_1 c$$

$$F_{\overline{m1}} = \overline{P_1}\,\overline{m_1} + a_1 \bar{b}$$

$$F_{T1} = m_1$$

$$F_{\overline{T1}} = T_2 a_1 c \overline{m_1}$$

$$P_{21} = T_1 a_1 c \overline{m_1}$$

б) *Синтез подсистемы № 2*

Вторая подсистема управляет захватным устройством ЗУ2. Поэтому назовем эту подсистему подсистемой ЗУ2. Исполнительными устройствами второй подсистемы служат гидравлические цилиндры Ц3 и Ц4. Положение цилиндра Ц3 контролируется в трех позициях (путевые переключатели Д1, Д2 и Д3, а положение цилиндра Ц4 – в двух позициях (путевые переключатели E_1 и E_2).

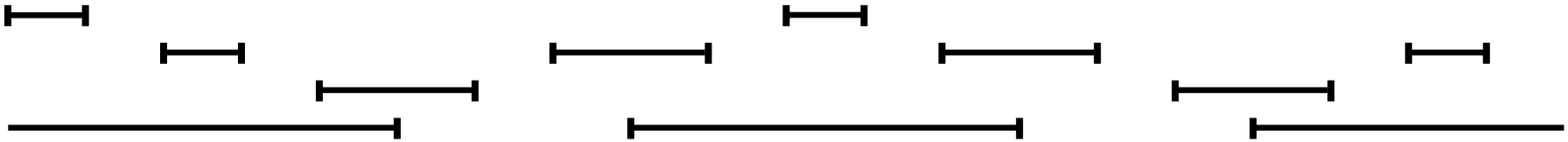
В соответствии с вышеописанным алгоритмом составим таблицу включений для 2-й подсистемы.

Начальная циклограмма, которая соответствует данной таблице включений, представлена на рис. 6.9. Чтобы исключить повторение весов входных переменных, в подсистему ЗУ2 необходимо ввести три элемента памяти (рис. 6.10). В результате приходим к реализуемой циклограмме, изображенной на рис.6.11 и на рис. 6.12.

2-я подсистема

Таблица включений

1.	d_1=1	$F_{\bar{w}}$ =1. Если P_{21}=1, то F_v=1	Вперед
2.	d_1=0		
3.	d_2=1		
4.	d_2=0		
5.	d_3=1	$F_{\bar{v}}$ =1; F_q=1	Вниз
6.	e=0	F_w=1	Назад
7.	d_3=0		
8.	d_2=1	$F_{\bar{w}}$ =1; $F_{\bar{q}}$ =1	Вверх
9.	e=1	F_w=1	Назад
10.	d_2=0		
11.	d_1=1	$F_{\bar{w}}$ =1. Если P_{22}=1, то F_v=1	Вперед
12.	d_1=0		
13.	d_2=1	$F_{\bar{v}}$ =1; F_q=1	Вниз
14.	e=0	F_v=1	Вперед
15.	d_2=0		
16.	d_3=1	$F_{\bar{v}}$ =1; $F_{\bar{q}}$ =1	Вверх
17.	e=1	F_w=1	Назад
18.	d_3=0		
19.	d_2=1		
20.	d_2=0		



	Такт	P21										P22									
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	d_1																				
2	d_2																				
4	d_3																				
8	e																				
	$\sum$	9	8	10	8	12	4	0	2	10	8	9	8	10	2	0	4	12	8	10	8
	F_v																				
	$F_{\bar{v}}$																				
	F_w																				
	$F_{\bar{w}}$																				
	F_q																				
	$F_{\bar{q}}$																				

Рис. 6.9. Начальная циклограмма подсистемы ЗУ2

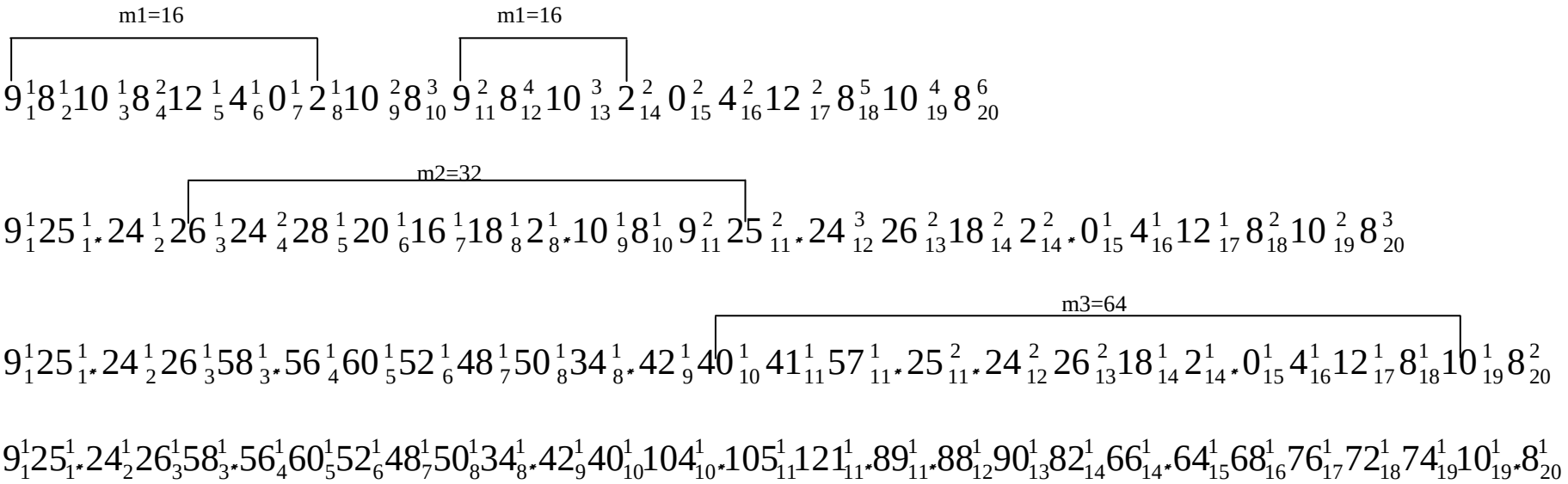


Рис. 6.10. Введение элементов памяти в подсистему ЗУ2

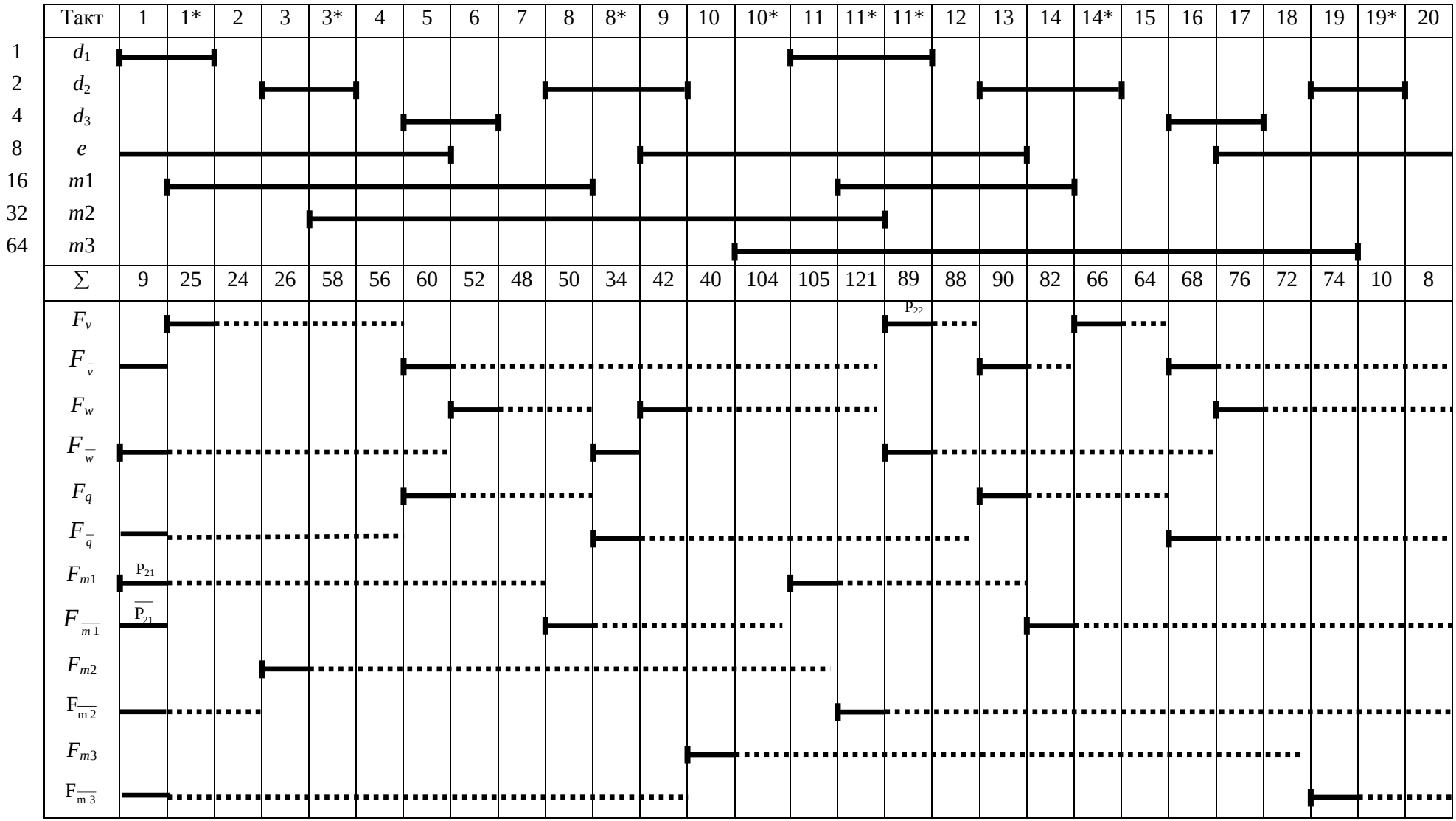


Рис. 6.11. Реализуемая циклограмма для подсистемы ЗУ2

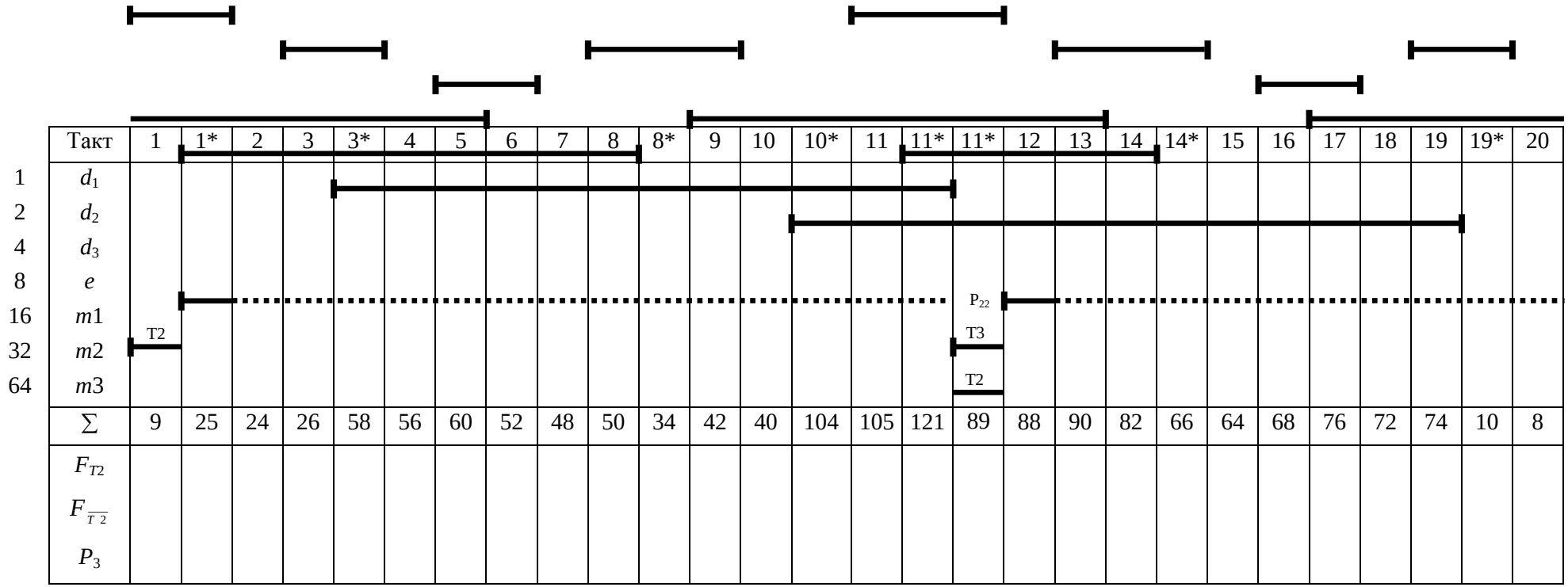


Рис. 6.12. Реализуемая циклограмма для подсистемы ЗУ2 (Продолжение циклограммы)

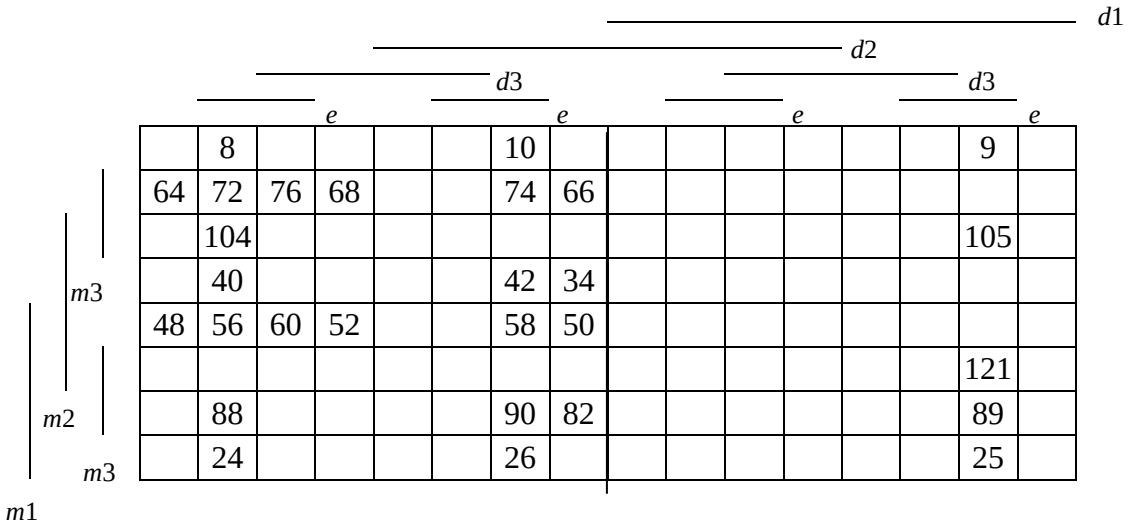
Проанализируем циклограмму подсистемы ЗУ2. В исходном состоянии (такт 1) происходит инициализация всех выходных переменных и сброс триггера управления T_2. При поступлении от первой подсистемы сигнала $P_{21} = 1$ (первый пуск 2-й подсистемы) формируется сигнал $F_{m1} = 1$ на включение первого элемента памяти. Сигнал $m_1 = 1$ с его выхода переводит систему в следующее состояние $\Sigma = 25$ (такт 1*). В этом же такте 1* включается триггер управления второй подсистемы ($F_{T2} = 1$).

Далее развивается автоматический цикл 2-й подсистемы до тех пор, пока она не окажется в состоянии $\Sigma = 89$ (такт 11**). Запрещающий сигнал блокировки $P_{22} = 0$ останавливает подсистему в указанном такте. Одновременно формируется сигнал $P_3 = 1$ на запуске третьей подсистемы (магазина инструментов М). Сигнал $T3 = 1$ с выхода триггера управления 3-й подсистемы выключает триггер управления 2-й подсистемы ($F_{\overline{T2}} = 1$). Таким образом, 2-я подсистема передала управление 3-й подсистеме, а сама перешла в режим ожидания.

Закончив работу, 3-я подсистема передает 2-й подсистеме сигнал P_{22} (второй пуск 2-й подсистемы). В результате вторая подсистема перейдет в состояние $\Sigma = 88$ (такт 12), где вновь включится ее триггер управления T_2 и автоматический цикл 2-й подсистемы продолжится.

По окончании автоматического цикла вторая подсистема возвращается в исходное состояние $\Sigma = 9$ (такт 1). Триггер управления T_2 сбрасывается и на этом работа всей системы в целом завершается.

Переходим к минимизации логических функций второй подсистемы. На рис. 6.13 показано расположение на карте Карно конституент, используемых во 2-й подсистеме, а на рис. 6.14 - результаты минимизации.

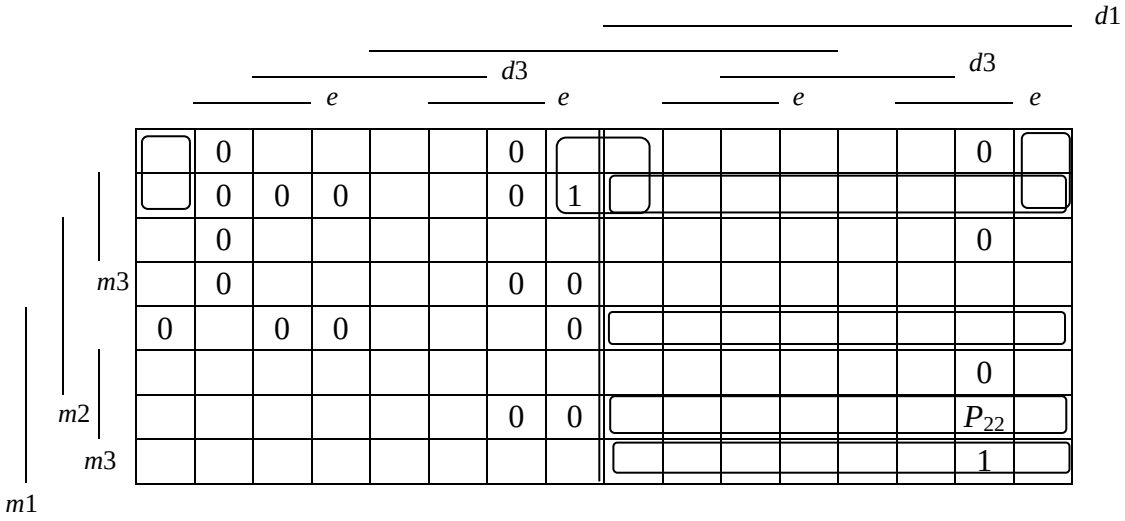


	8					10								9	
64	72	76	68			74	66								
	104													105	
	40					42	34								
48	56	60	52			58	50								
														121	
	88					90	82							89	
	24					26								25	

Рис. 6.13. Карта конституент, используемых в подсистеме ЗУ2

а) $F = 25, P_{22} \cdot 89,\ \ 66$

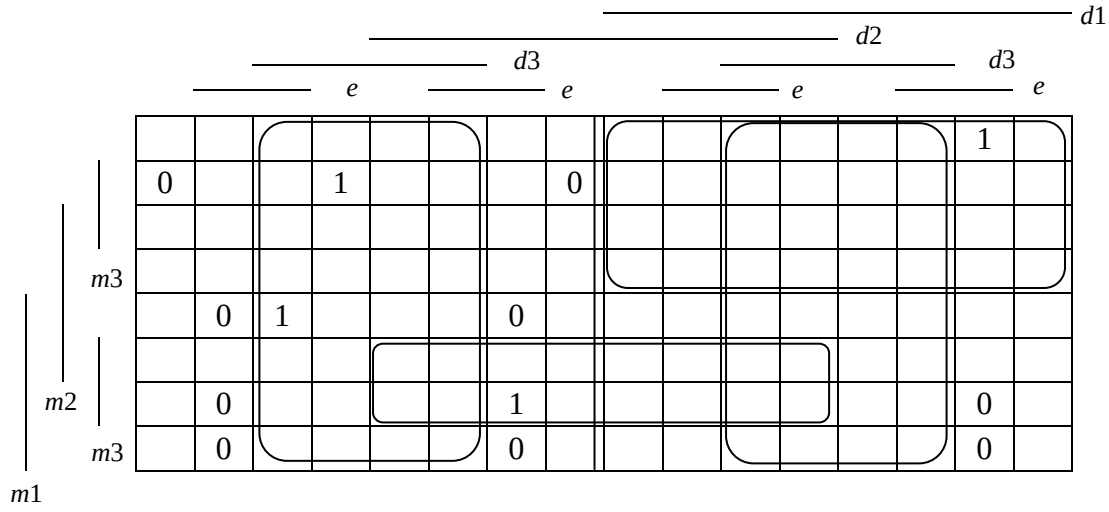
$\overline{F_v} = 9,60,52,48,50,34,42,40,104,105,121,90,82,68,76,72,74,10,8$



$$F_v = P_{22} d_1 \overline{m2} m3 + d_1 m_1 \overline{m3} + \overline{d_3} e m1 \overline{m2} = P_{22} d_1 \overline{m_2} m_3 + d_1 m_1 \overline{m_3} + \overline{d_3} e_1 \overline{m_1} \overline{m_2}$$

б) $\overline{F_v} = 9,60,90,68$

$\overline{\overline{F_v}} = 25,24,26,58,56,89,88,66,64$



$$\overline{F_v} = d_3 + d_1 \overline{m1} + d_2 m1 m3$$

Рис. 6.14. Минимизация логических функций подсистемы ЗУ2

Продолжая процедуру минимизации, получим следующие минимальные формы:

$$F_w = \overline{d_1} e_1 \overline{m_1} + d_3 e_2 m_2$$

$$F_{\overline{w}} = d_1 \overline{m_2} + e_2 \overline{m_1}$$

$$F_q = d_3 m_1 + d_2 m_1 m_3$$

$$F_{\overline{q}} = \overline{m_1}\,\overline{m_3} + d_3 \overline{m_1}$$

$$F_{m1} = P_{21} d_1 \overline{m_2} + d_1 m_2$$

$$F_{\overline{m1}} = \overline{P_{21}}\,\overline{m_1}\,\overline{m_3} + d_2 e_2$$

$$F_{m2} = d_2 m_1 \overline{m_3}$$

$$F_{\overline{m2}} = d_1 \overline{m_2} + d_1 m_1$$

$$F_{m3} = \overline{d_2}\,\overline{m_1} m_2$$

$$F_{\overline{m3}} = d_2 \overline{m_3} + d_2 e_1 \overline{m_1}$$

$$F_{T2} = m_1 \overline{m_3} + \overline{d_1}$$

$$F_{\overline{T2}} = T_2 d_1 \overline{m_1}\,\overline{m_3} + T_3 d_1 \overline{m_2} m_3$$

$$P_3 = T_2 d_1 \overline{m_2} m_3$$

Связанные между собой триггеры управления рассмотренных подсистем образуют своеобразный диспетчер, который передает в заданные моменты времени управление от одной подсистемы к другой.

На рис. 6.15 представлена функциональная схема соединений триггеров управления. Из схемы видно, что после поступления команды «ПУСК» ($P_1 = 1$) включается в работу подсистема ЗУ1 и в состоянии $\Sigma = 29$ включается триггер управления $T1$. В состоянии $\Sigma = 13$ выходной сигнал с этого триггера формирует сигнал P_{21}, который запускает в работу подсистему ЗУ2.

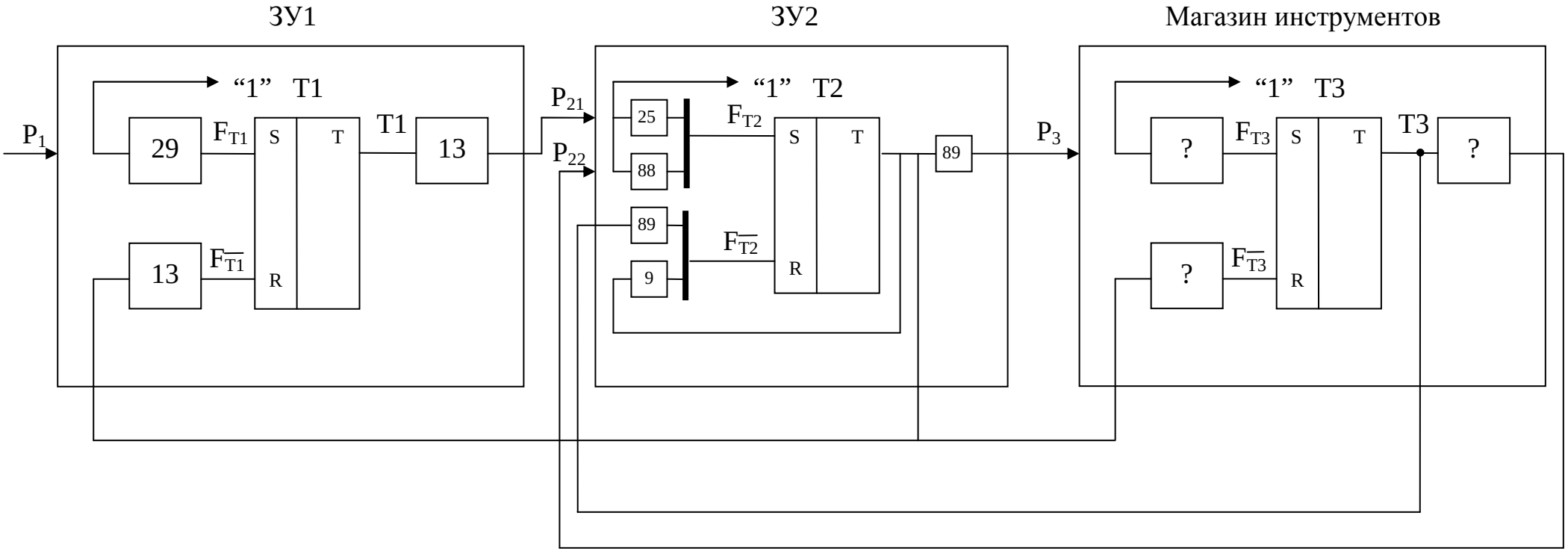


Рис. 6.15. Схема соединений триггеров управления

В состоянии $\Sigma = 25$ второй подсистемы включается триггер управления $T2$ (выделенные вертикальные штрихи на схеме обозначают элементы **ИЛИ**). Сигнал с его выхода сбрасывает триггер управления $T1$. В состоянии второй подсистемы $\Sigma = 89$ триггер управления $T2$ формирует сигнал $P_3 = 1$ на запуск 3-й подсистемы. Её триггер управления $T3$ сбрасывает триггер управления $T2$.

По окончании работы 3-й подсистемы формируется сигнал $P_{22} = 1$, который вновь запускает подсистему ЗУ2 и её триггер управления включается второй раз. Завершив работу, подсистема ЗУ2 приходит в состояние $\Sigma = 9$ и триггер управления $T2$ по цепи обратной связи сбрасывает сам себя.

Таким образом, схема соединений триггеров управления носит регулярный характер и может быть легко расширена на любое количество подсистем.

На самом деле при управлении объектом с помощью ПЛК рассмотренная схема управления реализуется программно, т. е. в данном случае она имеет лишь иллюстративное назначение.

6.3. Параллельные циклы, условные переходы, подпрограммы

Реальные объекты управления работают не только в автоматических режимах, но и в режимах ручного управления, а также в режимах наладки, диагностики, выхода из аварийных ситуаций и т. п. Для этого требуется вводить в систему управления дополнительные подсистемы, подпрограммы, функции и т. п., каждая из которых представляет собой программный модуль (далее просто модуль). Чтобы строить ветвящиеся алгоритмы, необходимы команды условной передачи управления.

Рассмотренный в предыдущем параграфе принцип запуска подсистем с использованием триггеров управления позволяет решить все эти задачи на единой алгоритмической основе.

На рис. 6.16 представлена схема управления параллельными циклами.

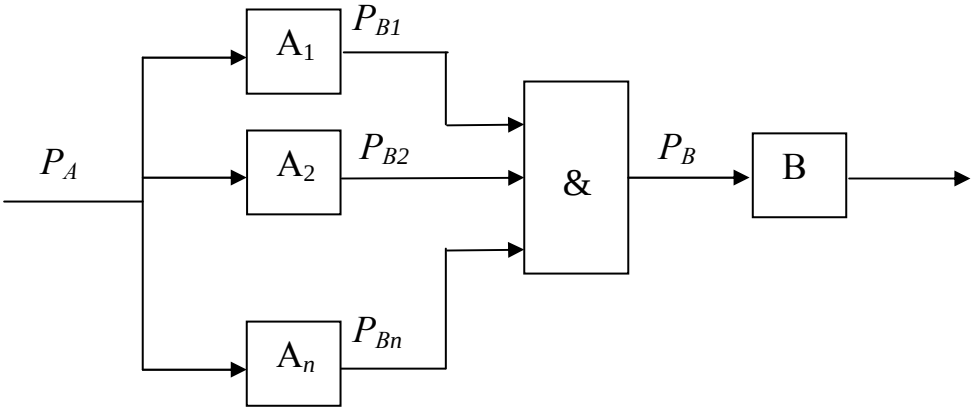


Рис. 6.16. Схема управления параллельными циклами

Сигнал «ПУСК» P_A подается одновременно на n модулей A_1, A_2, …, A_n , которые включаются в работу. Каждый из этих модулей по окончании своего цикла формирует соответствующий сигнал P_{B1}, P_{B2}, …, P_{Bn} на запуск следующего модуля B. Эти сигналы поступают на схему **И**, с выхода которой поступит результирующий сигнал P_B на запуск модуля B только тогда, когда завершит свою работу последний из модулей A_1, A_2, …, A_n.

Для простоты сигналы обратной связи с модуля B на модули A_1, A_2, …, A_n на рисунке не показаны.

Принцип формирования этих сигналов ясен из примера, рассмотренного в предыдущем параграфе. В ПЛК указанные связи и схема **И** реализуются программно.

Условная передача управления, как известно, состоит в том, что в алгоритм управления вводится элемент сравнения двух переменных (в нашем случае булевых переменных). В зависимости от состояния этих переменных продолжение алгоритма управления происходит по первой (ДА) или по второй (НЕТ) ветви.

На рис. 6.17 показана схема условной передачи управления, выполненная с помощью схем **И** и **НЕ-И** (ясно, что в контроллере эти схемы реализованы программно).

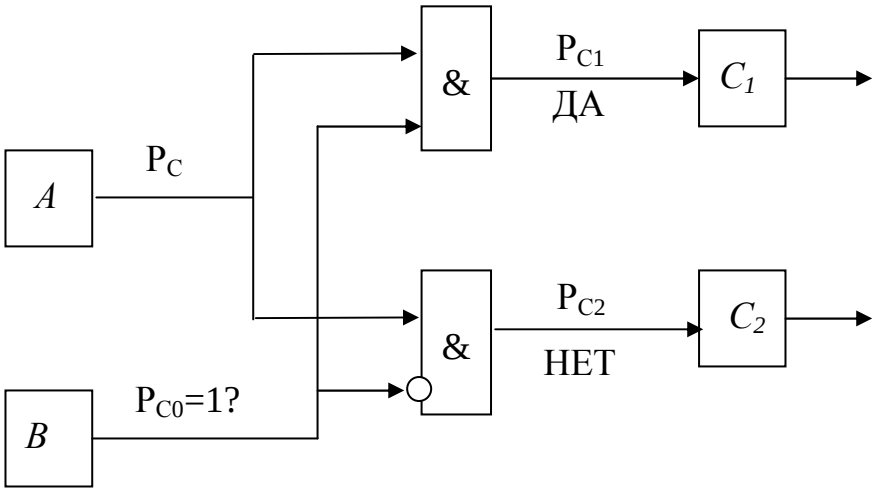


Рис. 6.17. Схема условной передачи управления

Если условие $P_{C0} = 1$ выполнено, то по команде $P_C = 1$ запускается модуль C_1, в противном случае – модуль C_2.

Широко известный в программировании прием, когда основная программа обращается к ряду подпрограмм, можно реализовать в нашем случае с помощью схемы, показанной на рис. 6.18.

Модуль A периодически запускает модули B_1, B_2, …, B_n , в свою очередь, по цепям обратной связи вновь запускает модуль A.

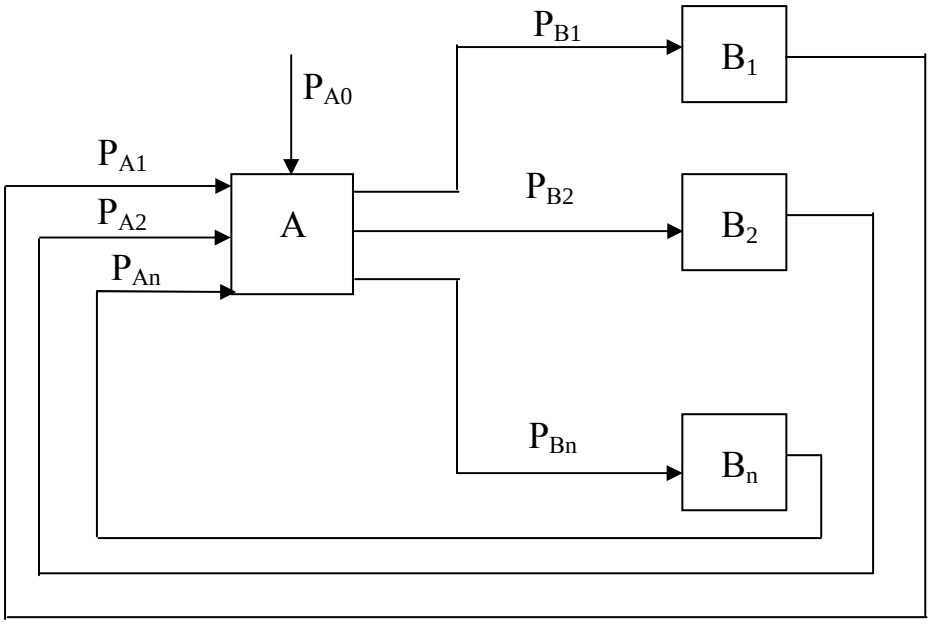


Рис. 6.18. Схема обращения к подпрограммам

Таким образом осуществляется возврат из подпрограмм (модули B_1, B_2, …, B_n) к основной программе (модуль A).

Схема связи триггеров управления и циклограмма их работы именно для такого случая были рассмотрены в предыдущем параграфе. Основной программой там служила подсистема ЗУ2, а подпрограммой – магазин инструментов M.

Подпрограммы могут быть вложенными одна в другую любое количество раз (вложенные подпрограммы) (рис. 6.19).

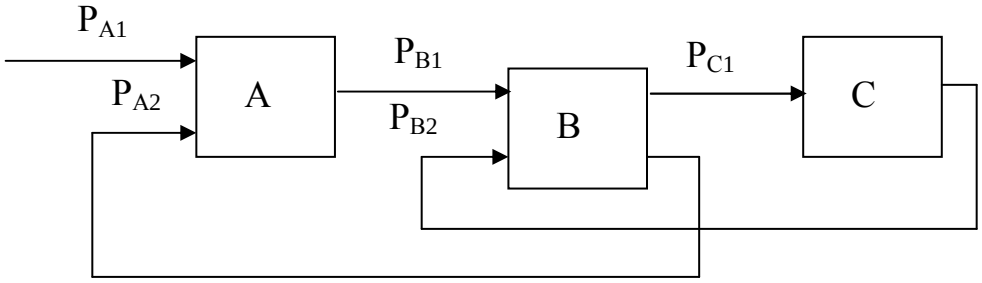


Рис. 6.19. Схема вызова вложенных подпрограмм

Организация вызовов таких подпрограмм и возврат к прерванным (остановленным) программам понятны из рисунка.

Рассмотренная организация сложных циклов в дискретных системах управления похожа на программирование операций логических устройств с

помощью языка последовательных функциональных схем (SFC) в среде ISaGRAF фирмы «CJ International» (Франция).

Среда ISaGRAF, являясь удобным средством программирования, оставляет процесс синтеза системы за программистом. Поэтому для достижения наибольшего эффекта в смысле качества и скорости проектирования полезно рассмотренную в данном учебном пособии методику синтеза дискретных систем управления по циклограммам работы механизмов сочетать с применением системы ISaGRAF.

Вместе с тем имеется ряд ограничений на применение системы ISaGRAF: высокая стоимость программного продукта, отсутствие в контроллере операционной системы OS – 9/9000, поддерживающей ISaGRAF и, наконец, недостаточная мощность контроллера. Например, существуют контроллеры, которые можно программировать только с помощью алгоритмического языка ASSEMBLER.

7. ИНСТРУМЕНТАЛЬНАЯ СИСТЕМА ПРОГРАММИРОВАНИЯ ЛОГИЧЕСКИХ КОНТРОЛЛЕРОВ *ISaGRAF*

7.1. Что такое *ISaGRAF* под *WINDOWS* ?

Задачи программирования контроллеров для систем и устройств связи с объектом весьма специфичны, сложны, трудоемки и требуют для своего решения соответствующих инструментальных средств автоматизации программирования. Использование универсальных языков программирования высокого уровня позволяет решать эти задачи, но требует при этом всеобъемлющих знаний теории и технологии программирования, особенностей конкретной операционной системы и тонкостей аппаратного обеспечения (контроллеров, модулей сопряжения с объектом и т.п.).

В 1992 году Международная Электротехническая Комиссия выпустила стандарт *IEC* 1131-3, определяющий пять языков программирования логических контроллеров (*Programming Logical Controller – PLC*). При разработке стандарта было предложено так много вариантов языка, что было трудно выбирать один из них в качестве общего языка программирования для *PLC*.

Стандарт IEC 1131-3 определяет следующие языки программирования для *PLC*:

1) графический *язык последовательных функциональных схем SFC* **(***Sequential Function Charts***),** описывающих «скелет» программы, - логику ее работы на уровне чередующихся или параллельных процедурных шагов и условных переходов;
2) графический *язык функциональных блоков диаграмм FBD* **(***Function Block Diagrams***),** позволяющий построить комплексную процедуру, состоящую из различных библиотечных функций (арифметических, тригонометрических, строковых) и

функциональных блоков (триггеры, переключатели, таймеры, счетчики и т.п.). Элементы этого языка выглядят как блоки, соединенные проводами в электрическую цепь, делая язык удобным для множества прикладных программ, содержащих передачу информации между различными компонентами;

3) графический *язык релейных диаграмм, или релейной логики LD* **(***Ladder Diagrams***)**, используемый для описания логических выражений различного уровня сложности. Функциональные блоки в языках *LD* и *FBD* используются для программного замещения простых электромеханических элементов;
4) *язык структурированного текста ST* **(***Structured Text***),** относящийся к классу языков высокого уровня и по мнемонике похожий на Паскаль. Язык *ST* предоставляет булевы и арифметические операторы, а также конструкции структурного программирования, такие, как *IF-THEN-ELSE, WHILE-DO, REPEAT-UNTIL*. На основе этого языка можно создавать гибкие процедуры обработки данных;
5) *язык инструкций IL* **(***Instruction List***)**, относящийся к классу текстовых языков низкого уровня (ассемблеров) и позволяющий создавать эффективные, оптимальные процедуры.

В 1990 году французская фирма «*CJ International*» выпустила продукт под названием «*ISaGRAF for DOS*» , а затем и «*ISaGRAF for WINDOWS*», в которых в полной мере реализовала поддержку всех пяти стандартных языков программирования *PLC*.

В *ISaGRAF* заложена методология структурного программирования, которая дает возможность пользователю описать автоматизируемый процесс в наиболее легкой и понятной форме. Интерфейс с пользователем системы *ISaGRAF* соответствует международному стандарту, включающему многооконный режим работы, полнографические редакторы, работу с мышью и т.п.

7.2. Графический язык последовательных функциональных схем SFC

Программа на языке *SFC* – это графический набор *шагов* и *переходов*, соединенных *направленными линиями*. Действия внутри шагов детально описываются на других языках (*ST*, *IL*, *LD* и *FBD*). К каждому переходу присоединяется логическое условие. Шаги и переходы всегда чередуются.

Шаги изображаются *одинарным квадратом*, внутри которого указывается автоматически формируемый *порядковый номер шага*. *На первом уровне программирования*, кроме самого шага, можно задать *необязательный комментарий – описание шага* в прямоугольнике, присоединенном к символу шага (рис. 7.1). Действия внутри шагов, как и

условия перехода, детально описываются на втором уровне программирования.

Рис. 7.1

Инициализация программы на языке *SFC* изображается с помощью *шагов инициализации* в виде *двойного квадрата* (рис. 7.2).

Рис. 7.2

Переходы изображаются маленькими *горизонтальными полосками*, которые пересекают линии соединения. К переходам можно адресоваться по номерам, которые записываются сразу после символа перехода. Правее символа перехода на первом уровне программирования может быть записан необязательный комментарий (рис. 7.3).

Шаги и переходы связаны одинарными направленными линиями соединения (рис.7.3).

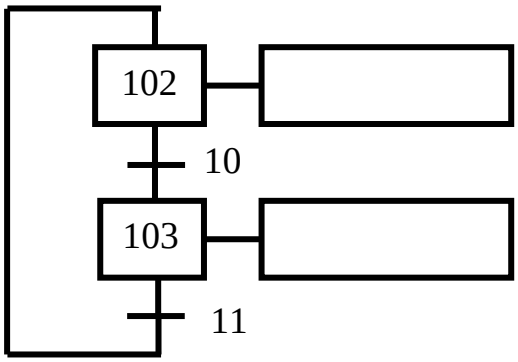


Рис. 7.3

Одинарная дивергенция – это множественное соединение в направлении от одного шага к нескольким переходам.

Одинарная конвергенция – это множественное соединение, направленное от нескольких переходов к одному и тому же шагу.

Одинарная дивергенция и конвергенция изображаются на схемах одинарными горизонтальными линиями, а двойная дивергенция и конвергенция – двойными линиями (рис. 7.4).

Двойная дивергенция – это множественное соединение, направленное от одного перехода к нескольким шагам. Она соответствует параллельному выполнению операций процесса.

Двойная конвергенция – это присоединение нескольких шагов к одному и тому же переходу.

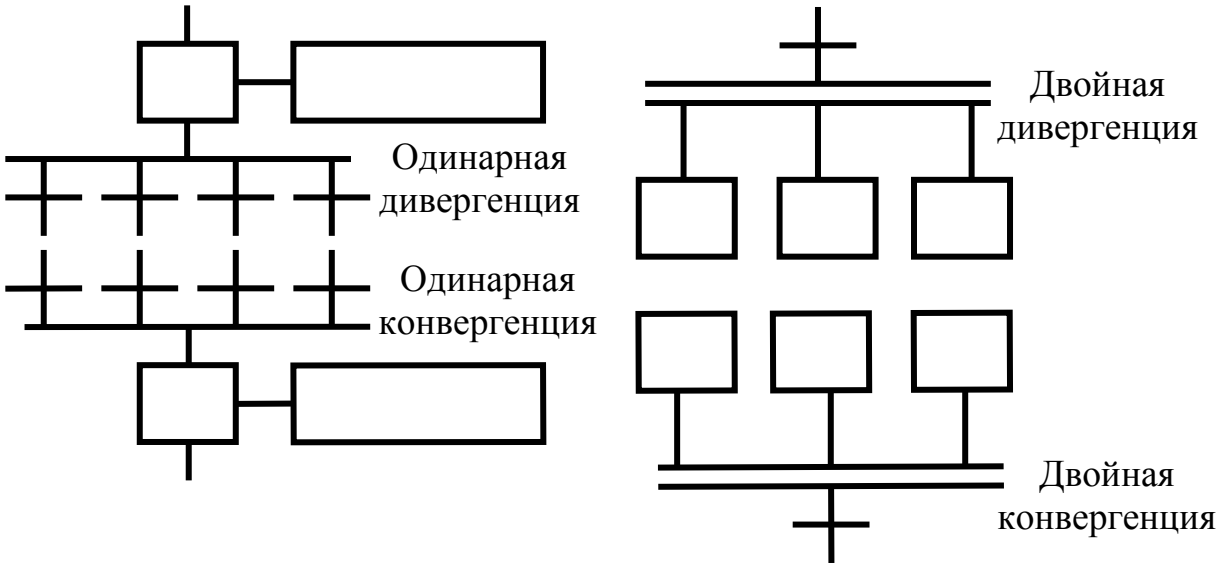


Рис. 7.4

Макро шаг – это уникальная группа шагов и переходов на языке *SFC, записанных отдельно и изображаемых в основной программе в виде одного символа***.** Ссылочный номер, написанный в символе макро шага основной схемы, – это ссылочный номер первого шага в теле макро шага. Сам макро шаг представляет собой автономную схему, где **первый шаг** не имеет верхнего соединения, а **конечный шаг** – нижнего. Символ макро шага может быть помещен в тело другого макро шага (рис.7.5).

*Второй уровень программирования шага SFC является детальным описанием действий, выполняемых во время активности шага***.** К действиям внутри шага относятся:

- *булевы действия и SFC* – действия, описываемые с помощью ограниченных текстовых возможностей самого языка *SFC*;

- «*Pulse*» и «*Non-stored*» – действия, программируемые на языках *ST* и *IL*;
- *вызов подпрограмм*, написанных на любом языке *ISaGRAF*, кроме *SFC*.

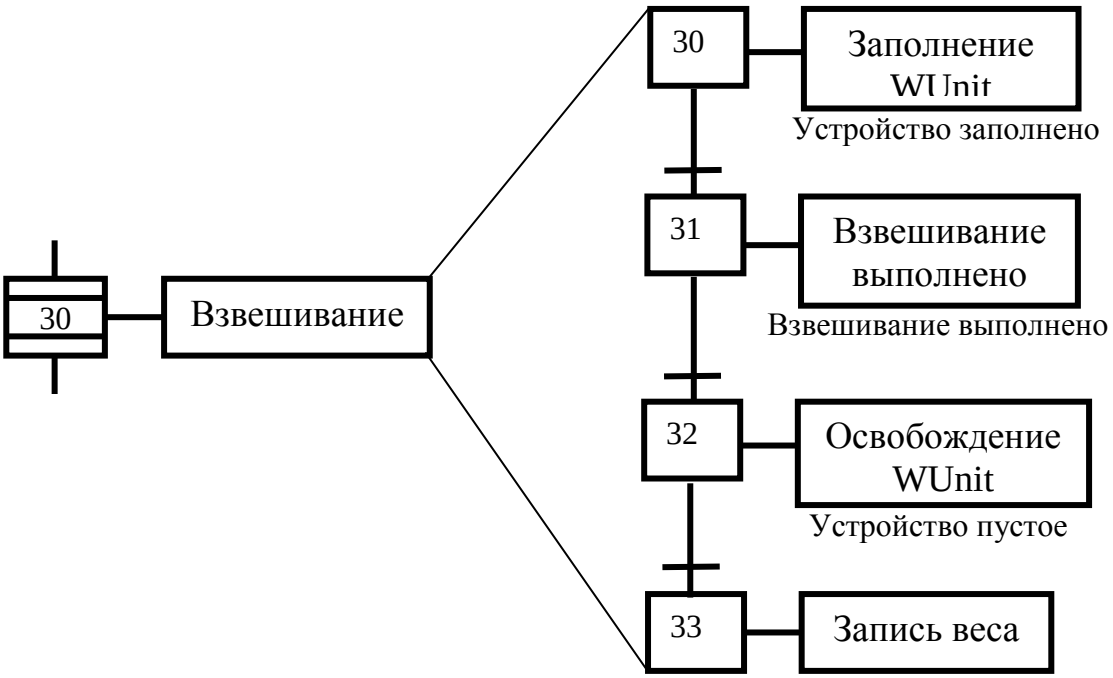


Рис. 7.5

Синтаксис булевых действий следующий:

<*bool_var*>(*N*); присваивает переменной <*bool_var*> значение сигнала активности шага (атрибут *N* является необязательным);

/<*bool_var*>; присваивает переменной <*bool_var*> отрицательное значение сигнала активности шага;

<*bool_var*>(*S*); устанавливает переменную <*bool_var*> в *TRUE*, когда шаг активизируется;

<*bool_var*>(*R*); устанавливает переменную <*bool_var*> в *FALSE*, когда шаг активизируется.

На рис. 7.6 приведен фрагмент программы на языке *SFC*.

«*Pulse***»-***действия – это список команд на языке ST или IL, которые выполняются только один раз в момент активизации шага***.** Синтаксис «*Pulse*»-действий следующий:

ACTION(*P*):

(*Операторы языка *ST* или блок команд на языке *IL**)

END_ ACTION;

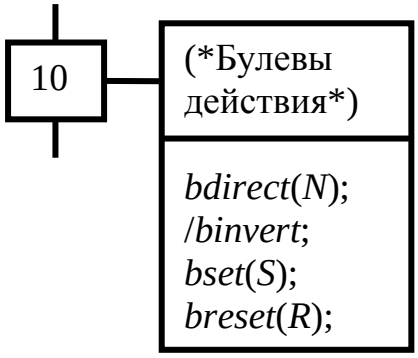


Рис. 7.6

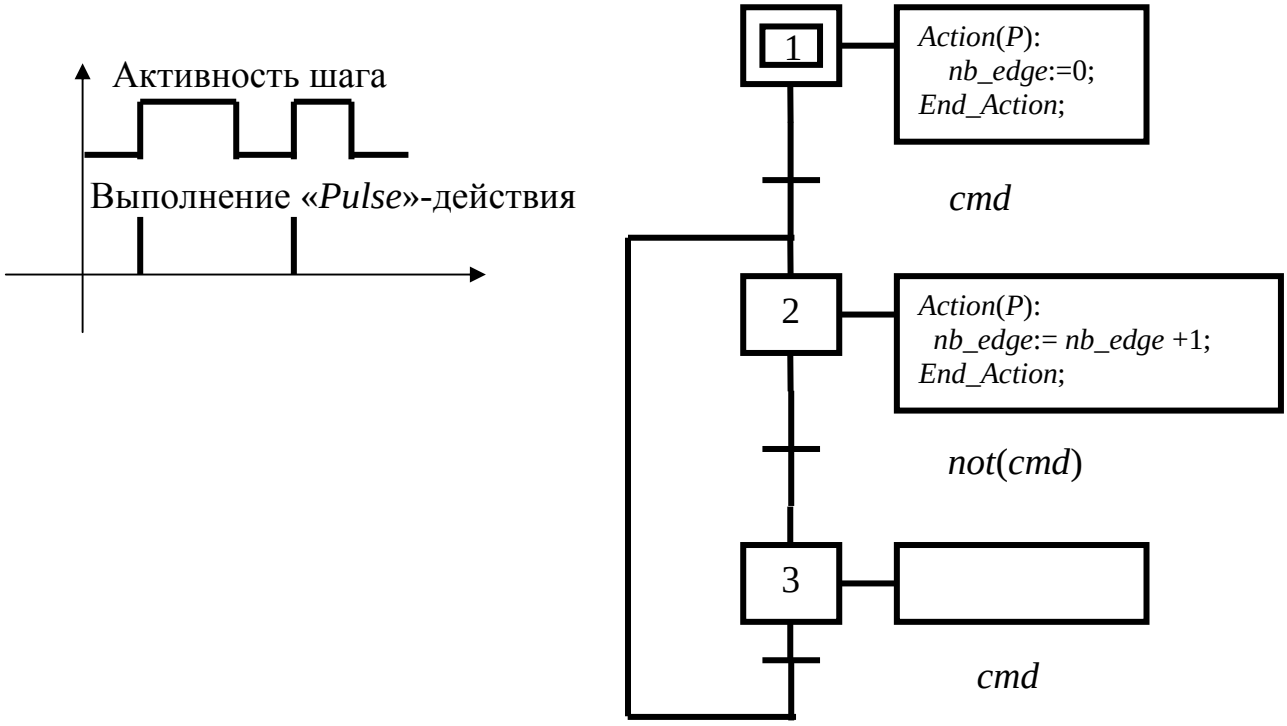


Рис. 7.7

На рис. 7.7 представлена временная диаграмма выполнения «*Pulse*»-действия в рамках шага и пример программы на языке SFC, использующей «*Pulse*»-действия.

«*Non-stored***»–***действие – это список команд на языке ST или IL, которые выполняются на каждом цикле в течение всего периода активности шага***.** Синтаксис описания этих действий следующий:

*ACTION***(***N***):**

(*Операторы языка *ST* или блок команд на языке *IL**)

*END_ ACTION***.**

На рис. 7.8 представлены временная диаграмма выполнения «*Non-stored*»–действий и пример программы на языке *SFC*, использующей «*Non-stored*»–действия внутри шага 2.

На втором уровне программирования *SFC к каждому переходу может быть прикреплено булево выражение*, которое и обуславливает поведение перехода. Условие может быть описано на языках *ST*, *IL* или из перехода может быть вызвана подпрограмма на любом языке.

На *языке ST условие* представляется в виде *булева выражения с точкой с запятой* **« ; »** на конце, содержащего булевы константы, входные и внутренние булевы переменные или комбинацию переменных, результатом которой является булево значение.

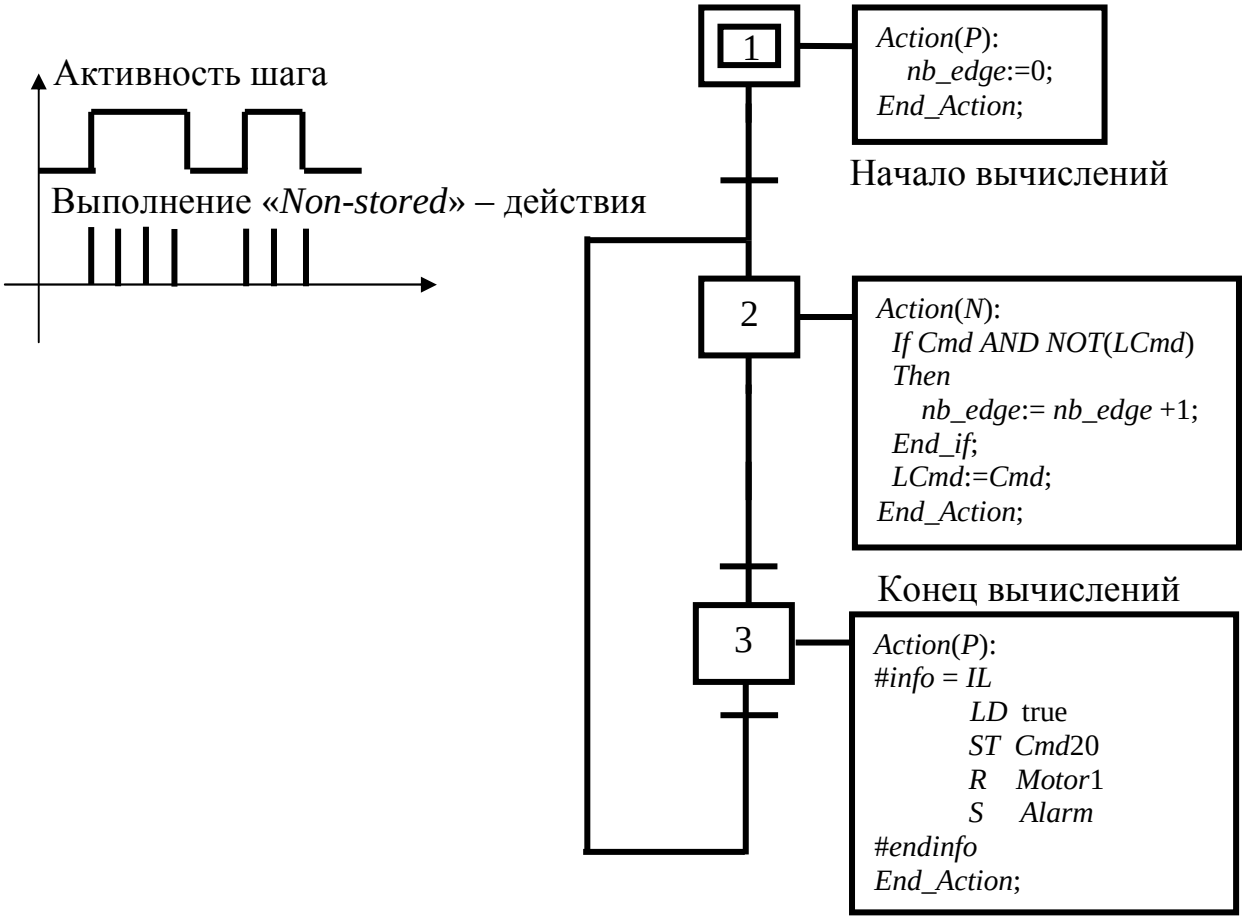


Рис. 7.8

В системе *ISaGRAF* каждая программа на языке *SFC* может управлять другими программами на том же языке, которые называются дочерними программами той программы, которая ими управляет. *SFC*–программы объединяются в иерархическое дерево на базе «родственных» отношений «отец–ребенок». К основным правилам этой иерархической структуры относятся следующие:

- программы на языке *SFC*, не имеющие «родителя», называются главными программами;

- главные программы на языке *SFC* активизируются системой при запуске прикладной программы;
- программа может иметь несколько дочерних программ;
- дочерняя программа не может иметь более одного родителя и управляется только родительской программой;
- программа не может управлять дочерними программами одной из своих дочерних программ.

Дочерняя программа <*child_prog*> может быть запущена или уничтожена в соответствии с изменением сигнала активности шага.

7.3. Графический язык диаграмм функциональных блоков – FBD

Графический язык *FBD* позволяет программисту строить сложные программы и процедуры на основе существующих функций библиотеки *ISaGRAF*, связанных в диаграмму.

Диаграмма *FBD* описывает *функцию* (рис.7.9), определяющую взаимодействие между *входными и выходными переменными*. Функция представляется как набор стандартных *элементарных функциональных блоков*, имеющихся в библиотеке *ISaGRAF*. Элементарный блок выполняет одну *функцию* взаимодействия между своими входами и выходами. Имя этой функции записывается в *прямоугольнике*, изображающем блок. Каждый блок имеет фиксированное число *входных* (слева) и *выходных* (справа) *точек присоединения* (рис.7.10).

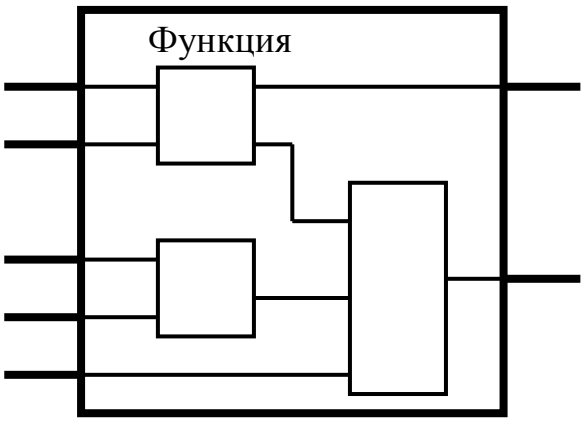


Рис. 7.9

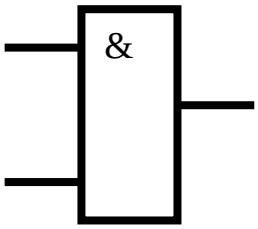


Рис. 7.10

Входные переменные программы на языке *FBD* должны присоединяться к входным точкам присоединения функциональных блоков, а *выходные переменные* - к выходным точкам.

Понятие переменной здесь шире, чем традиционная переменная в языках программирования. *Входом* диаграммы FBD может быть *константа* либо любая *внутренняя*, *входная* или *выходная* переменная. *Выходом*

диаграммы могут служить *внутренняя* или *выходная* переменная либо *имя подпрограммы*, редактируемой в текущий момент совместно с основной программой.

Стандартная библиотека функциональных блоков включает в себя:

- блоки управления данными (например, блок присваивания);
- блоки булевых операций (например, «И» (*AND*), «ИЛИ» (*OR*));
- блоки арифметических операций (например, «+», «-», «*», «/»);
- блоки логических операций (традиционные побитовые операции над числами);
- блоки сравнения.

7.4. Язык релейных диаграмм *LD*

Релейная диаграмма или *диаграмма цепей* **(***LD***)** – это графическое представление булевых уравнений, содержащих *контакты* **(***входные аргументы***)** и *обмотки* **(***выходные результаты***).** Графические символы языка *LD* соответствуют элементам электрической цепи и имеют те же названия и обозначения.

Диаграмма *LD* слева и справа ограничена вертикальными линиями, которые называются *левой силовой шиной и правой силовой шиной* соответственно.

Графические символы диаграммы *LD* присоединяются к силовым шинам и связаны между собой линиями соединения, которые могут быть горизонтальными или вертикальными (рис. 7.11).

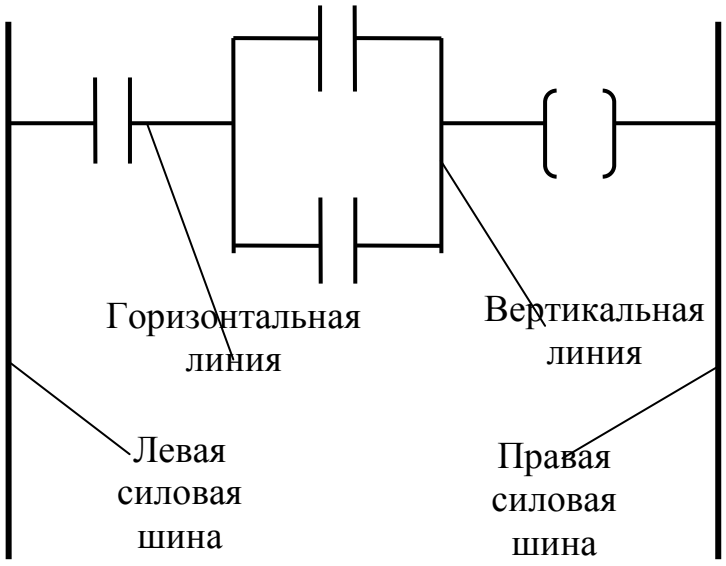


Рис. 7.11

Комбинация горизонтальных и вертикальных линий соединения позволяет строить *множественные соединения***.** Состояния их концов подчиняются правилам булевой алгебры.

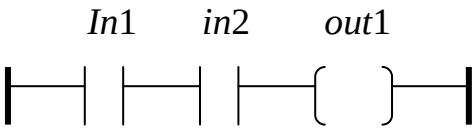
Множественное левое соединение состоит из *нескольких* горизонтальных линий, присоединенных к левой стороне вертикальной линии, и *одной* горизонтальной линии, присоединенной к данной вертикальной линии справа.

Множественное правое соединение состоит из *одной* горизонтальной линии *слева* от вертикальной линии и *нескольких* горизонтальных на ее *правой* стороне.

Множественное левое и *правое состояние* содержит *несколько* горизонтальных линий *слева* и *справа* от вертикальной**.**

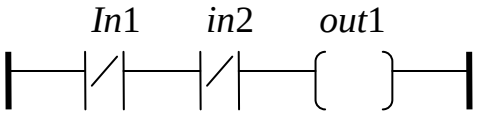
Основные контакты языка *LD*:

- прямой (нормально разомкнутый) контакт – разрешает булеву операцию между состоянием линии соединения и булевой переменной;



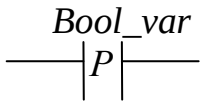

```
(*Эквивалент на ST*)
out:=in1 AND in2;
```

- обратный (нормально замкнутый) контакт – разрешает операцию **ЛОГИЧЕСКОЕ И** между состоянием левой линии соединения и булевым отрицанием значения переменной, связанной с контактом;

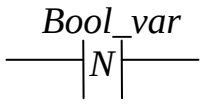



```
(*Эквивалент на ST*)
out1:=NOT(in1) AND NOT(in2);
```

- контакт с обнаружением нарастающего фронта – приводит к тому, что состояние линии соединения справа от контакта устанавливается в *TRUE* тогда, когда состояние линии соединения слева – *TRUE* и состояние переменной, связанной с контактом, изменилось с *FALSE* на *TRUE*;



- контакт с обнаружением падающего фронта – устанавливает в *TRUE* состояние правой линии соединения тогда, когда состояние линии слева – *TRUE* и состояние переменной контакта изменяется с *TRUE* на *FALSE*, и сбрасывает правую линию на *FALSE* во всех остальных случаях.



Основные обмотки языка *LD*:

- прямая обмотка;
- обратная обмотка;
- обмотка с установкой;
- обмотка со сбросом.

Подробнее о библиотеке языка можно узнать из раздела «*HELP*».

7.5. Структурированный текст – *ST*

Структурированный текст **(*ST*)** - это язык высокого уровня, напоминающий по синтаксису язык Паскаль. Язык *ST* предназначен для реализации сложных процедур, которые сложно описать с помощью графических языков. В частности, он по умолчанию используется для описания действий внутри шагов и условий, прикрепленных к переходам, языка *SFC*.

Программа на языке ST – это *список операторов*, заканчивающихся *точкой с запятой***.**

Выражения на языке ST состоят из *операций* и *операндов* (констант и переменных). Чтобы выделять части выражения и изменять приоритет выполнения операций, используют круглые скобки (например, 2+6*3 = 2+18 = 20).

Вызовы подпрограмм на языках *ST*, *IL*, *LD* или *FBD* могут использоваться в любом выражении *ST*-программы или переходе языка *SFC* и синтаксис такого вызова прост:

<имя подпрограммы> (<параметр1>,…,<параметр*N*>).

Основные операции языка *ST*:

- Арифметические операции

(*операции над целыми аналоговыми переменными*)
ares: = (*ax*1+*ax*2)*(12/(*scale*-3));
a: =11/2; (*в результате *a*=5*)
(*операции над вещественными аналоговыми переменными*)
rres: =(*rx*2+1.02*E*5)/(*rx*3-3.14)**rx*8;
rrx: =*rx*1/*rx*2; (*ошибка, если *rx*2=0.0*)
(*операции над таймерными переменными*)
timeprog: = 10*s*+*tm*10-*tm*56;
tresult: = *tm*1-*tm*2; (*если *tm*2>*tm*1, произойдет ошибка исполнения*);

- Операции сравнения

Bt: =(123>12); (* *bt – true**)
Bt: =(1.0>=1.0); (* *bt – true**)
Bt: =(‘*a*’ = ‘*A*’); (* *bt – false**)

- Логические операции (логическое И – «*AND*», логическое ИЛИ – «*OR*», логическое исключающее ИЛИ – «*XOR*», булево отрицание – «*NOT*»).

Основные операторы языка *ST*:

- Оператор присваивания («переменная»:= «любое_выражение»;).
- Оператор *RETURN* – завершает выполнение текущей программы (блока).
- Оператор ветвления *IF-THEN-ELSE*.

IF <бул_выражение> *THEN*
<оператор>;
<оператор>;
ELSE
<оператор>;
<оператор>;
……..
END_IF;

- Операторы цикла *WHILE*, *REPEAT*, *FOR*.

WHILE <бул_условие> *DO*
<оператор>;
<оператор>;
……..
END_WHILE;

REPEAT
<оператор>;
<оператор>;
UNTIL <бул_условие>
END_REPEAT;

FOR <*index*>:=<*mini*> *TO* <*maxi*> *BY* <*step*>
DO
<оператор>;
<оператор>;
……..
END_FOR;

- Оператор *OPERATE* – выполняет указанную операцию ввода/вывода над переменной ввода/вывода.

<переменная>: =*OPERATE*(<*io*>, <*order*>, <*argument*>,

где *io* – входная/выходная переменная;
order – аналоговое выражение;

argument – аналоговое выражение.

7.6. Язык *IL* – список команд

Список команд **(***IL***)** – это язык низкого уровня ассемблерного типа, который может эффективно использоваться для оптимизации отдельных частей прикладной программы. Язык *IL – одноадресный*, и большинство его команд работают с явно неуказанным *текущим результатом* (или *регистром IL*). Оператор *IL* указывает *операцию*, которая должна быть выполнена *между операндом* и *регистром IL*, а *результат* операции сохраняется *в* том же *регистре IL*.

Программа на языке IL - это *список команд*. Каждая команда должна начинаться с новой строки и содержать *оператор*, заканчивающийся необязательным *модификатором* и, если необходимо, *операндом*.

Перед командой может располагаться *метка***,** за которой следует *двоеточие*. Имена меток традиционны для *ISaGRAF* (не более 16 букв, цифр или символов подчеркивания «_», начинающихся с буквы).

Основные операции и операторы:

- *N* – булево отрицание
- (– отложенная операция
- *C* – условная операция
- *LD* – загрузка в регистр; } управление данными
- *ST* – сохранение регистра; }

```
STboo:  LD     false          (* regIL: = false *)
        ST     boo_var1       (*boo_var1: = regIL = false *)
        STN    boo_var2       (*boo_var2: = NOT(regIL) = true*)
```

- *S* – устанавливает в *TRUE* операнд или текущий результат,
- *R* – сбрасывает булеву переменную в *FALSE*, если содержимое регистра результата *TRUE*.

```
SetReset: LD   true       (*regIL: = true*)
          S    boo_var1   (*boo_var1: = true,  regIL не изменился *)
          R    boo_var2   (*boo_var2: = false, regIL не изменился*)
```

- *ADD* – сложение
- *SUB* – вычитание
- *MUL* – умножение } арифметические операторы;
- *DIB* – деление

```
Arif_ex:  LD   a1   (* a1: = 10; a2: = 15; a3: = 5; *)
          ADD  a2   (* regIL: = a1 *)
```

```
SUB   a3   (*regIL: = a1+a2 *)
ST    a0   (* a0: = 20 *)
```

- JMP – переход к метке
- *RET* – возврат из текущей программы

} операторы управления

```
(* вызывающая программа преобразует аналоговое значение в таймерное *)
Main:  SUBPRO          (* вызов программы SUBPRO для *)
                       (* получение аналогового значения *)
                       (* текущий результат: значение *)
                       (* возвращаемое подпрограммой *)
       GT      vmax    (* проверка значения на переполнение *)
       RETC           (* возврат, если переполнение *)
       MUL     100    (* преобразует секунды в сотые доли секунды *)
       ST      tmval  (* запись значения в таймерную переменную *)
```

- Операторы сравнения см. в библиотеке в «*HELP*».

Замечание: язык *IL* не может работать с вещественными аналоговыми и строковыми типами данных.

7.7. Работа с *ISaGRAF*

Запустите пакет *ISaGRAF*. Для этого нажмите кнопку «Пуск» в нижней инструментальной линейке, выберите меню “Программы” подменю *ISaGRAF* и *Projects* и щелкните на этой опции мышкой. После загрузки пакета на экране появится окно менеджера проектов (рис.7.12). Ознакомьтесь с командами меню и линейкой инструментов менеджера.

Создайте свой проект нажатием кнопки «*Create new projekt*» в линейке инструментов или соответствующей командой меню «*Files*». Появится окно атрибутов проекта. Введите имя проекта, а конфигурацию ввода/вывода установите «*none*», что позволит определять переменные ввода/вывода по своему усмотрению. После нажатия кнопки «ОК» в списке проектов появится проект с определенным вами именем. Если необходимо, введите краткий комментарий командой «*Set comment text*» меню «*Edit*» (в данной версии *ISaGRAF* в комментарии применим только английский алфавит). Комментарий будет расположен справа от имени проекта.

Заполните дескриптор (описание) проекта. Для этого нажмите кнопку «*Edit project descriptor*» на линейке инструментов. Откроется окно дескриптора. Текст можете вводить на русском языке. Язык текста выбирается средствами операционной системы *MS Windows*. После ввода текста откройте меню «*Files*», сохраните текст дескриптора командой «*Save*» и вернитесь в окно менеджера проекта командой «*Exit*». Можно пользоваться кнопками линейки инструментов.

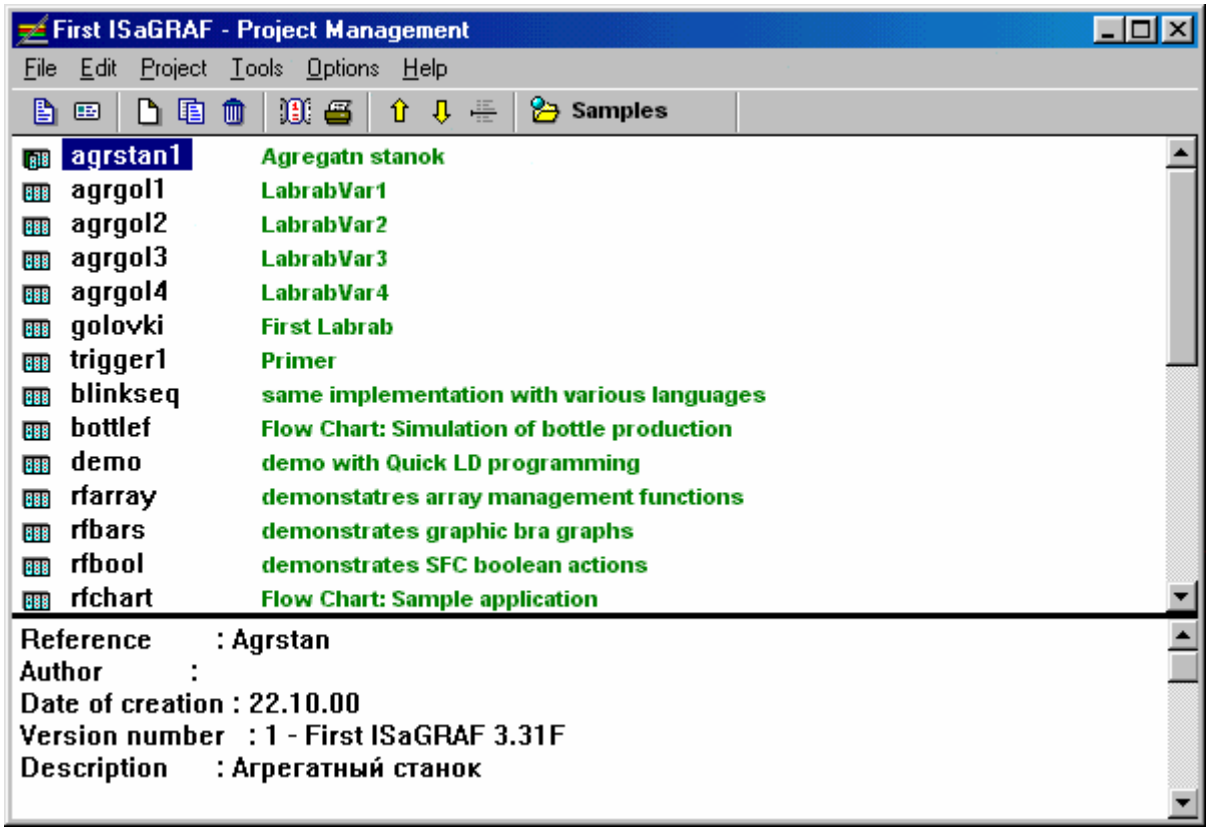


Рис.7.12 Окно менеджера проектов

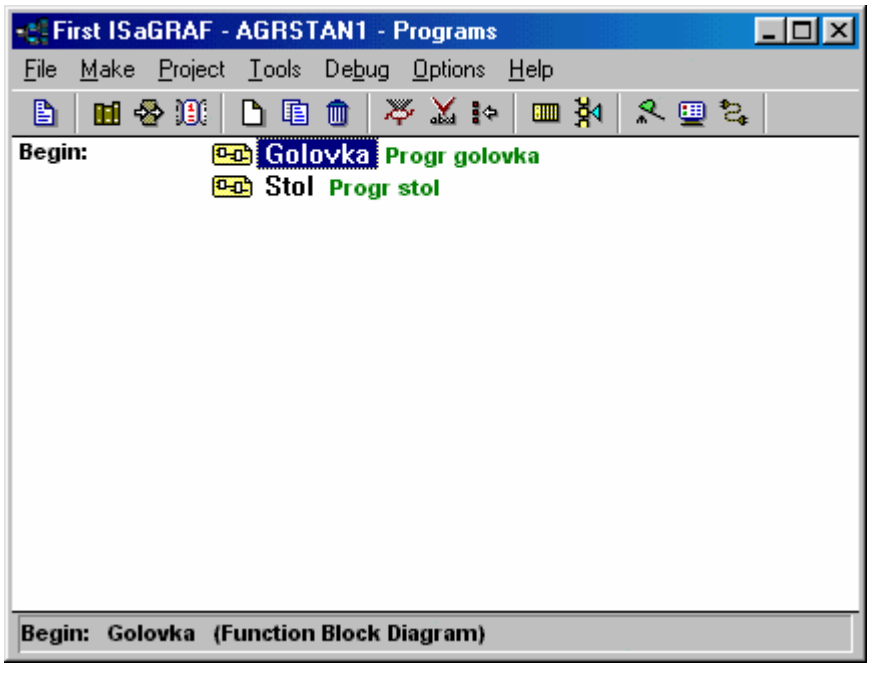


Рис.7.13 Окно программирования и отладки

Откройте созданный Вами проект двойным щелчком мышки по имени проекта или кнопкой «Open» на линейке инструментов. Откроется окно программирования и отладки (рис. 7.13). Ознакомьтесь с командами меню и линейкой инструментов этого окна. Опишите структуру программного обеспечения. Для задания имени программного модуля и его атрибутов нажмите кнопку «*Create new program*» на линейке инструментов. Откроется окно «*New program*». Введите имя программы, необходимый краткий комментарий (только английский алфавит), язык программирования и тип программной секции. После введения атрибутов и нажатия кнопки «ОК» в окне программирования и отладки появится пиктограмма с именем программы и введенным комментарием. Аналогично определяются атрибуты других программных модулей, составляющих структуру программного обеспечения. Если программный модуль является подпрограммой ранее введенной программы, то это необходимо подтвердить выбором соответствующей опции в окне «*Style*». Необходимо отметить, что программные модули будут выполняться в той последовательности, как они расположены в списке программ.

После определения структуры программного обеспечения приступайте к программированию. Для вызова требуемого редактора языка достаточно дважды щелкнуть мышкой на требуемой программе или, выделив ее цветом, нажать кнопку «*Edit*» программы. Откройте окно редактора, назначенного для данной программы языка программирования.

Ознакомьтесь с командами меню и инструментами редактора. Наберите программу. Обязательно определите входные и выходные переменные вашей программы. Сохраните набранную программу, выбрав команду «*Save*» в меню «*File*» или соответствующей кнопкой линейки инструментов. После сохранения появится окно дневника программы. Внесите в дневник необходимые сведения и закройте окно редактора языка.

Опишите входные и выходные переменные вашего проекта. Для этого необходимо нажать на кнопку «*Dictionary*» на линейке инструментов. Появится окно с именем вашей программы и электронной таблицей. Ознакомьтесь с инструментами и командами этого окна. Первая строка электронной таблицы будет выделена цветом. Выберите соответствующий тип описываемых переменных и дважды щелкните мышкой по выделенной цветом строке. Откроется окно атрибутов вводимой переменной. Введите имя переменной, определенное вами при программировании, и соответствующие атрибуты. Чтобы не выходить из окна определения переменных, пользуйтесь кнопками «*Next*» и «*Previous*». После описания всех входных и выходных переменных сохраните таблицу имен кнопкой «*Save*» или соответствующей командой меню «*File*». Закройте окно.

Соедините входные и выходные переменные с устройствами ввода/вывода (УВВ). Для этого нажмите кнопку «*I/O connection*» на линейке инструментов. Появится окно соединений ввода/вывода с именем вашей программы (рис. 7.14).

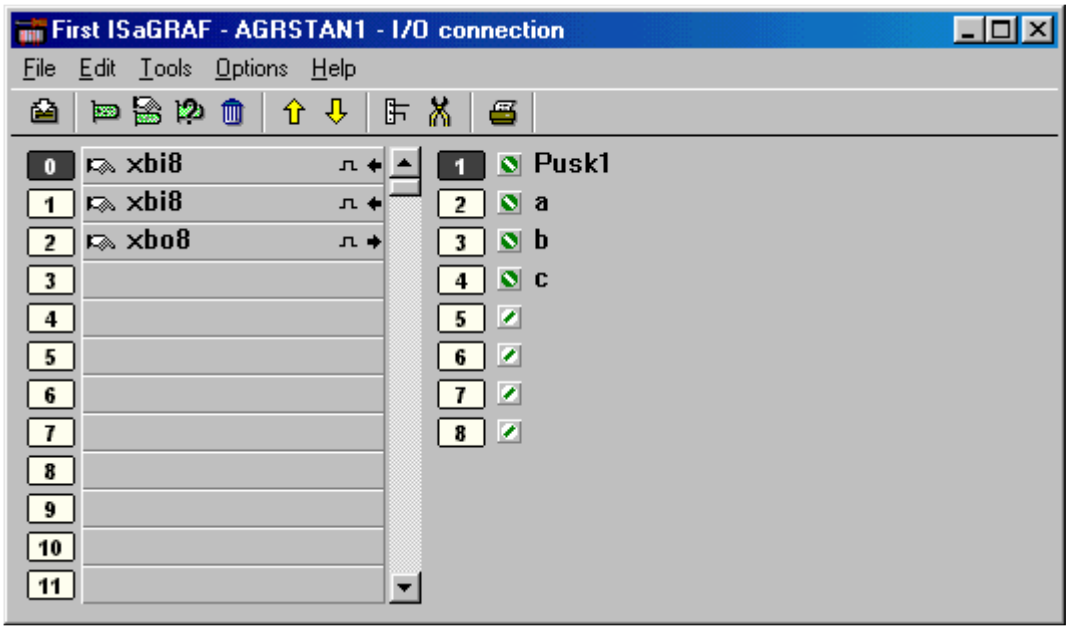


Рис.7.14. Окно соединений ввода/вывода

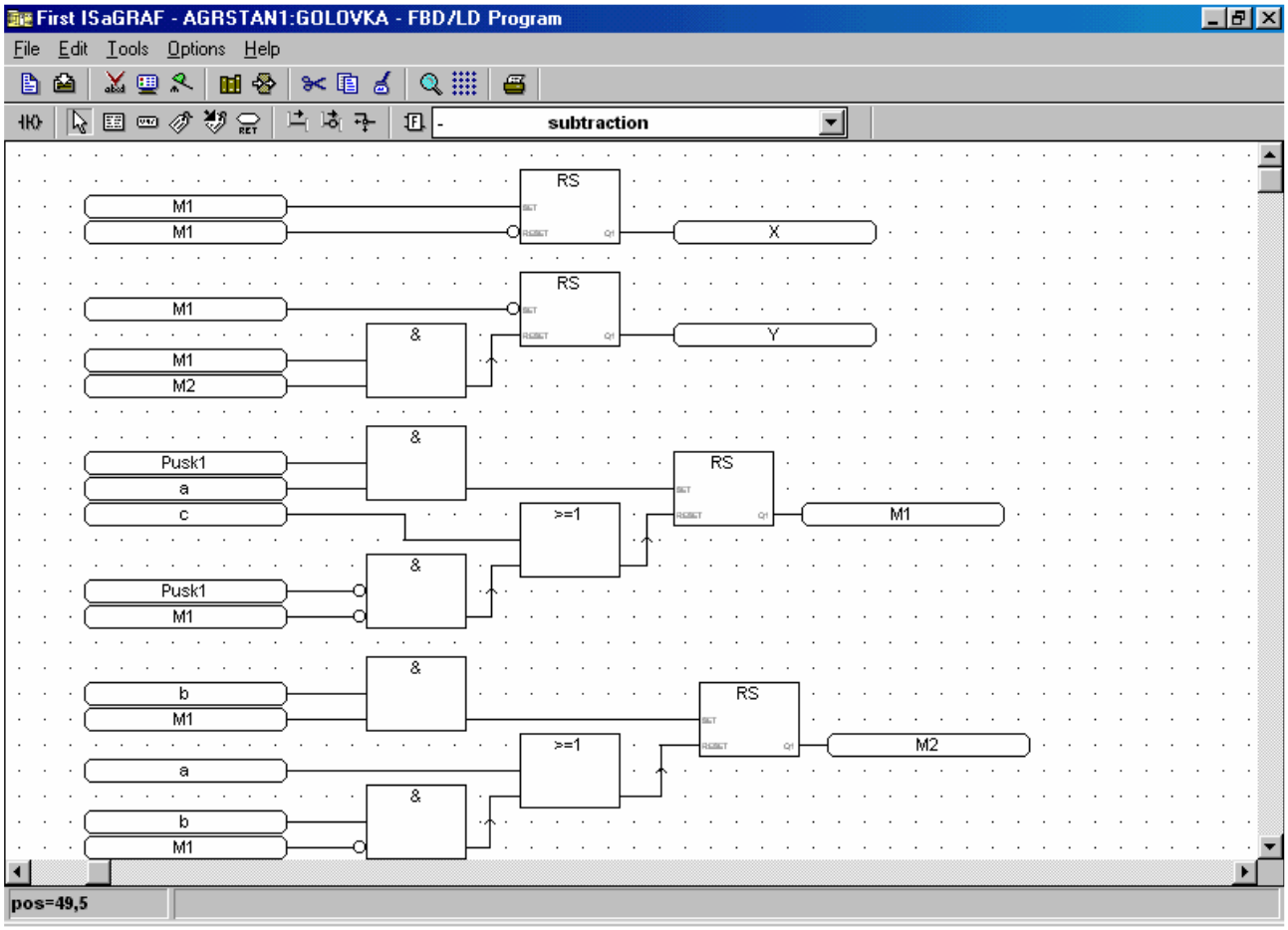


Рис. 7.15. Пример программы управления агрегатной головкой на языке FBD в системе ISaGRAF

Выберите устройство ввода, щелкнув мышкой на выделенном черным цветом номере слота. Откроется окно со списком устройств. В нашем случае необходимо выбрать виртуальное устройство ввода для соответствующего типа переменных, так как отсутствует реальный контроллер с УВВ. Виртуальное устройство имеет атрибут *Simulate*, который следует после имени устройства. Выделите выбранное устройство, щелкнув мышкой на его имени, и нажмите кнопку «OK» (описание устройства вызывается кнопкой «*NOTE*»). В слоте с выбранным номером появится название устройства, а справа – виртуальные клеммы с номерами каналов. Для подключения входной переменной щелкните дважды на номере выбранного канала. Появится окно каналов со списком свободных (неподключенных) переменных.

Выделите из списка имя переменной для первого канала УВВ и нажмите кнопку «*Connect*». Если последовательность номеров входов (выходов) соответствует последовательности номеров переменных в списке, то соединения можно производить не выходя из окна. После завершения процедуры присоединения переменных нажмите кнопку «*Close*». Процедура присоединения выходных переменных аналогична. После выполнения процедуры соединения сохраните конфигурацию УВВ нажатием кнопки «*Save*» или используйте соответствующую команду меню «*File*». Закройте окно соединений. Вы вернетесь в окно программирования и отладки.

Проверьте синтаксис программ, выбрав нужную программу и нажав кнопку «*Verify program*» на линейке инструментов.

Генератор кода проверит синтаксис программы и выдаст в появившемся окне сообщение об ошибках и их позициях в программе. Если ошибок нет, то появится дополнительное окно с предложением продолжить проверку или выйти. Выберите соответствующую вашей ситуации кнопку.

Отладьте программы. Для вызова отладчика выполните команду «*Simulate*» в меню «*Debug*». Появится окно отладчика с виртуальными элементами задания и индикации входных и выходных переменных. Удобнее отлаживать программу из окон редакторов языков, где представлены соответствующие инструменты и команды. В режиме отладки исключен режим редактирования программы.

В качестве примера на рис.7.15 представлена программа управления агрегатной головкой на языке FBD в системе *ISaGRAF*. Структурно-кинематическая схема агрегатной головки была рассмотрена нами ранее (рис. 5.38).

Запомните полезные функции:

1) Переименование проекта
   Project Management/*File*/*Rename*
2) Удаление проекта:
   Project Management/*File*/*Delete*
3) Печать документа:
   Project Management/*Print complete document*

4) Запись проекта на любой диск:
   Project Management/*Tools*/*Archive*/*Projects*/*Browse*(выбор диска)/*Backup*
5) Чтение проекта с любого диска:
   Project Management/*Tools*/*Archive*/*Projects*/*Browse*(выбор диска)/*Restore*
6) Работа с симулятором:
   а) включение имен переменных:
      Options/*Variable names*
   б) постоянное расположение поверх всех окон:
      Options/*Always on top*
   в) сохранение состояний входов:
      Tools/*Save input scheme* (задать имя файла)
   г) восстановление состояний входов:
      Tools/*Load input scheme* (указать имя файла)

Список литературы

1. Гаврилов М.А., Девятков В.В., Пупырев Е.И. Логическое проектирование дискретных автоматов. – М.: Наука, 1977. – 352 с.
2. Глушков В.М. Синтез цифровых автоматов. – М.: Физматгиз, 1962. – 476 с.
3. Лазарев В.Г., Пийль Е.И. Синтез управляющих автоматов. – М.: Энергия, 1970. – 400 с.
4. Рогинский В.Н. Основы дискретной автоматики. – М.: Связь, 1975. – 432 с.
5. Системы управления автоматических машин. /А.Н. Рабинович – К.: Техника, 1973. – 437 с.
6. Теория автоматов /Ю.Г. Карпов – СПб.: Питер, 2002. – 224 с.
7. Чернов Е.А. Проектирование станочной электроавтоматики. – М.: Машиностроение, 1989. – 304 с.
8. Юдицкий С.А., Тагаевская А.А., Ефремова Т.К. Проектирование дискретных систем автоматики. – М.: Машиностроение, 1980. – 232 с.

Приложение

Программа минимизации логических функций на алгоритмическом языке GWBASIC

```
10 REM Программа минимизации логических функций с учетом
безразличных состояний.
20 REM            Разработана ст. гр. АТП-509 Башиным С.В.
30 REM                    УГАТУ 2001г.
40 CLS:PRINT"введи порядок":INPUT N
50 PRINT "кол-во обязательных состояний ":INPUT Q
60 PRINT "кол-во запрещенных состояний ":INPUT P
70 DIM G(Q):DIM X(Q)
80 DIM D(2^N):DIM Z(2^N):DIM A(N):DIM P(N):DIM B(N,P+1):DIM
O(N,Q+1)
90 FOR I=1 TO 2
100 IF I=1 THEN  S=Q ELSE  S=P
110 FOR J=1 TO S
120 IF I=1 THEN PRINT"введите обяз. сост. номер ";J ELSE PRINT"введите
запрещенные сост. номер "J
130 INPUT K
140 IF I=1 THEN  D(K)=1 ELSE  D(K)=2
150 NEXT J
160 NEXT I
170 FOR L=0 TO 2^N
180 IF D(L)<>1 THEN GOTO 1390
190 P1=0
200 FOR L1=0 TO 2^N
210 IF D(L1)<>2 THEN GOTO 350
220 P1=P1+1
230 FOR I=1 TO N
240 S=INT(L/2^(I-1))
250 IF S/2<>INT(S/2) THEN  T1=1 ELSE T1=0
260 S=INT(L1/2^(I-1))
270 IF S/2<>INT(S/2) THEN  T2=1 ELSE T2=0
280 REM PRINT"№ "L" перем "I" = "T1,"№ ="L1" перем"I"="T2
290 IF T2<>T1 THEN B(I,P1)=1  ELSE B(I,P1)=0
300 REM IF T2<>T1 THEN  PRINT"t1<>t2" ELSE PRINT"t1=t2"
310 NEXT I
320 REM FOR I=1 TO N:PRINT B(I,P1);:NEXT I
330 REM PRINT
340 IF P1=P THEN GOTO 360
350 NEXT L1
360 P1=0
```

```
370 FOR L2=1 TO 2^N
380 IF (D(L2))<>1 THEN GOTO 480
390 P1=P1+1
400 FOR I=1 TO N
410 S=INT(L/2^(I-1))
420 IF S/2<>(INT(S/2)) THEN  T1=1 ELSE T1=0
430 S=INT(L2/2^(I-1))
440 IF S/2<>(INT(S/2)) THEN T2=1 ELSE  T2=0
450 IF T2<>T1 THEN O(I,P1)=1 ELSE O(I,P1)=0
460  NEXT I
470 IF P1=Q THEN GOTO 490
480 NEXT L2
490 S1=0
500 FOR I=1 TO Q
510 FOR J=1 TO P
520 FOR R=1 TO N
530 IF (O(R,I)=0) AND (B(R,J)=1) THEN GOTO 550
540 NEXT R:GOTO 580
550 NEXT J
560 S1=S1+1
570 G(S1)=I
580 NEXT I
590 FOR I=1 TO N
600 S=0
610 FOR J=1 TO P
620 S=S+B(I,J)
630 NEXT J
640 B(I,P+1)=S
650 NEXT I
660 FOR M=S1 TO 1 STEP -1
670 FOR I=(2^S1)-1 TO 0 STEP -1
680 M2=0
690 FOR M1=1 TO S1
700 S=INT(I/2^(M1-1))
710 IF S/2=(INT(S/2)) THEN X(M1)=0 ELSE  X(M1)=1
720 IF S/2<>(INT(S/2)) THEN  M2=M2+1
730 NEXT M1
740 IF M2=M THEN  GOTO 770
750 NEXT I
760 NEXT M:GOTO 1040
770 FOR J=1 TO N:O(J,Q+1)=0:NEXT J
780 FOR J=1 TO S1
790 IF X(J)=0 THEN GOTO 830
800 FOR R=1 TO N
```

```
810 O(R,Q+1)=O(R,Q+1)+O(R,G(J))
820 NEXT R
830 NEXT J
840 FOR J=1 TO P
850 FOR R=1 TO N
860 IF (O(R,Q+1)=0) AND (B(R,J)=1) THEN GOTO 880
870 NEXT R:GOTO 750
880 NEXT J
890 FOR I=1 TO N
900 IF O(I,Q+1)>0 THEN B(I,P+1)=0
910 NEXT I
920 FOR I=1 TO S1
930 IF X(I)=0 THEN  GOTO 1030
940 FOR J=1 TO N
950 S=INT (L/2^(J-1))
960 IF (S/2)<>(INT(S/2)) THEN  T1=1 ELSE  T1=0
970 IF O(J,G(I))=T1 THEN O(J,G(I))=0 ELSE O(J,G(I))=1
980 NEXT J:S=0
990 FOR J1=1 TO N
1000 S=S+O(J1,G(I))*2^(J1-1)
1010 NEXT J1
1020 D(S)=0
1030 NEXT I
1040 FOR I=N TO 1 STEP -1
1050 S=-1
1060 FOR J=1 TO N
1070 IF S<B(J,P+1) THEN S1=J
1080 IF S<B(J,P+1) THEN S=B(J,P+1)
1090 NEXT J
1100 A(I)=S1
1110 B(S1,P+1)=-2
1120   REM PRINT "a("I")="A(I),
1130 NEXT I:PRINT
1140 Z(0)=L:E=0
1150 FOR I=1 TO N
1160 F=E:V=0
1170 FOR J=0 TO F
1180 S=INT(Z(J)/2^(A(I)-1))
1190 IF S/2<>INT(S/2)THEN  T1=1 ELSE T1=0
1200 Y=Z(J)+((-1)^T1)*2^(A(I)-1)
1210 IF D(Y)=2 THEN P(A(I))=0
1220 IF D(Y)=2 THEN  E=E-V
1230 IF D(Y)=2 THEN GOTO 1280
1240 V=V+1:E=E+1:Z(E)=Y
```

```
1250 NEXT J
1260 FOR J=(E-V) TO E:D(Z(J))=3:NEXT J
1270 P(A(I))=1
1280 NEXT I
1290 Y=L
1300 FOR I=1 TO N
1310 S=INT(Y/2^(I-1))
1320 IF S/2<>INT(S/2)THEN A(I)=1 ELSE A(I)=0
1330 NEXT I
1340 FOR I=1 TO N
1350 IF P(I)=1 THEN GOTO 1370
1360 IF A(I)=0 THEN  PRINT"a"I"-отриц."ELSE PRINT"a"I
1370 NEXT I
1380 PRINT" V "
1390 NEXT L
```

Пример минимизации

```
введи порядок
? 4
кол-во обязательных состояний
? 2
кол-во запрещенных состояний
? 3
введите обяз. сост. номер  1
? 1
введите обяз. сост. номер  2
? 12
введите  запрещенные сост. номер  1
? 9
введите  запрещенные сост. номер  2
? 8
введите  запрещенные сост. номер  3
? 10

a 4 -отриц.
 V

a 3
 V
Ok_
```

ЧИКУРОВ Николай Георгиевич

ЛОГИЧЕСКИЙ СИНТЕЗ ДИСКРЕТНЫХ СИСТЕМ УПРАВЛЕНИЯ

Учебное издание

Редактор З.Г. Кашаева

Подписано в печать 25.08.03 Формат 60х84 1/16.
Бумага оберточная. Печать плоская. Гарнитура Times New Roman Cyr.
Усл. печ. л. 8,3 . Усл. кр.–отт. 8,3 Уч. – изд. л. 8,2.
Тираж 200 экз. Заказ №.
Уфимский государственный авиационный технический университет
Уфимская типография № 2 Министерства печати и массовой информации Республики Башкортостан.
450000, Уфа-центр, ул. К. Маркса, 12