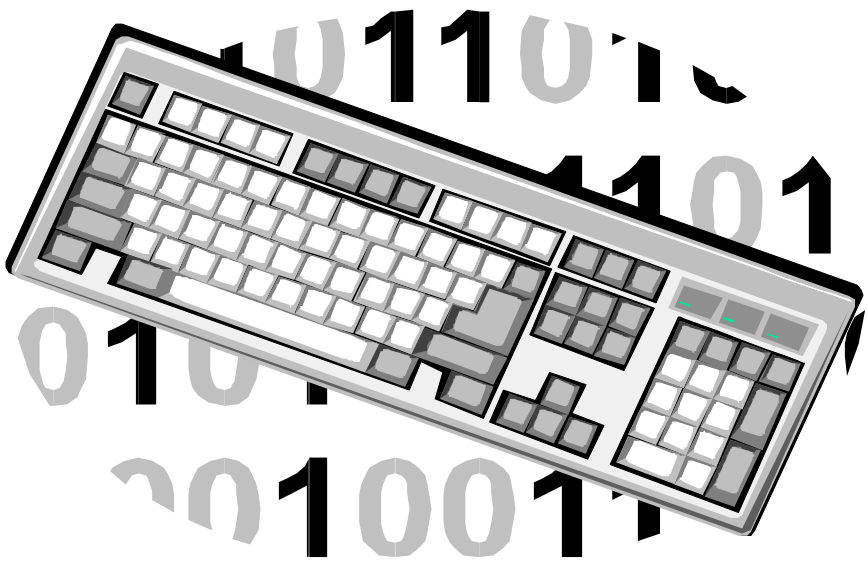
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ

А.Г. Рощин, Р.М. Половов

Теория автоматов

Часть I



Москва- 2007

ФЕДЕРАЛЬНОЕ АГЕНСТВО ВОЗДУШНОГО ТРАНСПОРТА
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ГРАЖДАНСКОЙ АВИАЦИИ

Кафедра вычислительных машин, комплексов,
систем и сетей

А.Г. Рощин, Р.М. Половов

ТЕОРИЯ АВТОМАТОВ

Часть I

Анализ и синтез комбинационных схем

Утверждено Редакционно-
издательским советом МГТУ ГА
в качестве учебного пособия

Москва-2007

ББК 6Ф6.5
Р 81

Печатается по решению редакционно-издательского совета
Московского государственного технического университета ГА.

Рецензенты: канд. техн. наук проф. Н.Н. Горнец
Доктор техн. наук проф. А.В. Балдин
Рощин А.Г., Половов Р.М.

Р 81 Теория автоматов. Часть I. Учебное пособие- М.: МГТУ ГА, 2007. -96 с.

Учебное пособие предназначено для изучения первой части дисциплины «Тория автоматов» студентами 2-го курса специальности 230101.

В пособии рассмотрены элементы теории алгоритмов, основы алгебры логики, а также методы анализа и синтеза комбинационных схем. Подробно рассмотрены методы минимизации логических функций. Особое внимание уделено вопросам минимизации частично определённых логических функций. Подробно освещены вопросы синтеза комбинационных схем с использованием интегральных элементов и элементов с ограниченным количеством входов, а также схем с несколькими выходами.

Данное учебное пособие издаётся в соответствии с рабочей программой учебной дисциплины СД.01 «Теория автоматов» по Учебному плану специальности 230101, утвержденному 25 января 2007 г. для студентов II курса специальности дневного обучения.

Рассмотрено и одобрено на заседаниях кафедры (протокол № 6) и методического Совета по специальности 230101 (протокол № 4) 20 марта 2007 г.

Рощин Алексей Григорьевич
Половов Радислав Михайлович
ТЕОРИЯ АВТОМАТОВ
Часть I
Учебное пособие

Содержание

I. Элементы теории алгоритмов
1.1. Основные понятия теории алгоритмов.....6
1.1.1. Алгоритм и его свойства.....6
1.1.2. Способы задания алгоритмов.....6
1.1.3. Машина Тьюринга.....10
Контрольные вопросы.....16
1.2. Универсальная машина Тьюринга.....17
1.2.1. Композиция машин Тьюринга.....17
1.2.2. Универсальная машина Тьюринга.....18
Контрольные вопросы.....22

II. Основы алгебры логики

2.1. Основные понятия……..24
2.2. Элементарные логические функции……...24
2.3. Формы логических функций……..26
2.4. Законы алгебры логики……...27
2.5. Техническая реализация логических функций……...29
Контрольные вопросы……....31

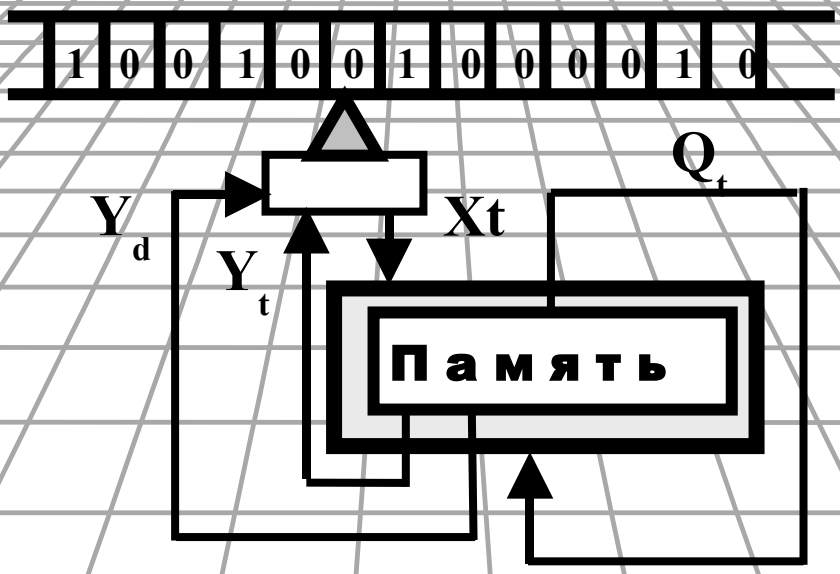
III. Комбинационные схемы

3.1. Анализ и синтез комбинационных схем……..…….....34
3.1.1. Типы цифровых автоматов…….. 34
3.1.2. Задачи анализа и синтеза комбинационных схем…..……....35
3.1.3. Анализ комбинационных схем……..…….... 37
3.1.4. Канонический метод синтеза комбинационных схем……....42
Контрольные вопросы…….……47
3.2. Минимизация логических функций…….……48
3.2.1. Методы минимизации логических функций……..…….....…48
3.2.2. Общая последовательность минимизации……..…….....…48

3.2.3. Метод непосредственных преобразований……..……51
3.2.4. Метод Карно……..……..…..
…..…53
Контрольные вопросы……..…..
…..……..…61
3.3. Метод Квайна……..…..…….62
3.3.1. Сущность метода Квайна…….……..…...
…62
3.3.2. Определение сокращенной ДНФ……..……...…
62
3.3.3. Определение тупиковых форм…….…..…...
…...64
3.3.4. Приближенный метод минимизации……...……67
Контрольные вопро-
сы……..…69
3.4. Метод интервалов……..……..
…….70
3.4.1. Сущность метода……..
……..70
3.4.2.Пример реализации метода ……..……..……...
71
Контрольные вопро-
сы……..76
3.5. Частные случаи синтеза комбинационных схем……..77
3.5.1. Синтез частично определенных схем……..…...77
3.5.2. Синтез схем на интегральных элементах……..…...80
3.5.3. Синтез комбинационных схем с несколькими

выходами……..
……..83
3.5.4. Использование скобочных форм для упрощения схемы……91
3.5.5. Синтез схем на элементах с ограниченным числом входов……93
Контрольные вопросы……95
литературА……96

Раздел I.
Элементы теории алгоритмов



1.1. Основные понятия теории алгоритмов

1.1.1. Алгоритм и его свойства

лгоритм - точное предписание, ведущее (вычислительный) процесс от варьируемых исходных данных к конечному результату.

(А. Марков).

В соответствии с этим определением алгоритму должны быть присущи следующие свойства:

- определенность;
- массовость;
- результативность.

Свойство определенности означает, что все указания или правила, составляющие алгоритм, должны быть однозначными и не допускать различных толкований. Алгоритм должен быть сформулирован четко настолько, чтобы он мог быть выполнен одинаково успешно любым пользователем.

Свойство массовости предполагает возможность применения алгоритма для решения не только отдельной задачи, но и целого класса задач. При этом задачи должны решаться для различных исходных данных.

Алгоритм, как последовательность действий, выполняется по шагам. Количество шагов может быть большим. Свойство результативности означает, что результат решения задачи должен быть достигнут за конечное число шагов.

Кроме приведенных выше свойств алгоритм обладает также такими свойствами как:

- дискретность;
- альтернативность и эквивалентность;
- самоуправляемость;
- сложность;
- адекватность;
- ресурсоемкость;
- уровень формализации.

В зависимости от типа обрабатываемых данных и основных операций по их обработке различают:

- вычислительные алгоритмы;
- логические алгоритмы;
- алгоритмы обработки графической информации и т.д.

1.1.2.Способы задания алгоритмов

Для описания алгоритма могут быть использованы различные средства.

В зависимости от типа преимущественно используемых средств различают следующие способы описания:

⇨словесное описание;

⇨табличное описание;
⇨графическое описание;
⇨аналитическое описание;
⇨описание с помощью специальных средств.

Словесное описание представляет собой описание алгоритма на естественном языке и обычно используется для обмена алгоритмами между различными пользователями, особенно на начальной стадии разработки алгоритма. Словесное описание может быть использовано для описания любых алгоритмов, однако оно имеет ряд недостатков, основными из которых являются громоздкость, неоднозначность и отсутствие наглядности. Отметим, что элементы словесного описания используются во всех других способах описания алгоритмов.

Основу табличного описания алгоритмов составляют таблицы различного рода: таблицы работы, таблицы переходов и выходов и т.д. Табличное описание является компактным и довольно наглядным, особенно при описании алгоритмов работы цифровых автоматов с памятью. Однако такое описание не всегда удобно.

При графическом описании могут быть использованы направленные графы, граф - схемы, временные диаграммы, сети Петри и т.д. Главной особенностью такого описания является наглядность. Однако для сложных алгоритмов на-

глядность может быть недостаточной, особенно при расположении графического описания на нескольких листах. Для обеспечения наглядности в этом случае алгоритм рассматривается на нескольких уровнях детализации.

При аналитическом описании используются математические формулы, логические функции и т.д. Особенно удобен этот способ для описания вычислительных и логических алгоритмов.

Все рассмотренные выше способы описания алгоритмов используют такие средства, которые применяют и в других случаях. В настоящее время для описания алгоритмов часто используют такие средства, которые предназначены специально для этой цели. К таким средствам относятся:

- ✓*алгоритмические языки (языки программирования);*
- ✓*машина Тьюринга;*
- ✓*рекурсивные функции;*
- ✓*нормальные алгоритмы Маркова;*
- ✓*операторы Ляпунова и т.д.*

Из этих специальных средств машина Тьюринга является объектом, который может использоваться не только для описания алгоритмов, но и как пример цифрового автомата с памятью. Поэтому принцип построения и работы машины Тьюринга далее будет рассмотрен подробно. Здесь же приведем пример использования различных способов для описания одного и того же алгоритма, в качестве которого рассмотрим алгоритм решения квадратного уравнения с произвольными коэффициентами.

Пример. Пусть задано квадратное уравнение

$$Ax^2 + Bx + C = 0 \qquad (1.1)$$

Необходимо описать алгоритм решения уравнений такого вида.

⇨*Словесное описание.* При решении квадратного уравнения вначале необходимо проверить, не является ли оно вырожденным. Для этого проверяется условие $A = 0$. Если условие выполняется, то квадратное уравнение выродилось в уравнение первого порядка вида $Bx + C = 0$. В этом случае уравнение имеет два одинаковых действительных корня, т.е. решение уравнения имеет следующий вид:

$$x_{1,2} = -\frac{C}{B} \qquad (1.2)$$

Если условие не выполняется, то необходимо определить дискриминант

$$D = B^2 - 4AC \qquad (1.3)$$

и проверить условие $D < 0$.

Если это условие не выполняется, то уравнение имеет действительные корни, значения которых определяются по формуле:

$$x_{1,2} = \frac{-B \pm \sqrt{D}}{2A}$$

(1.4)

Если условие выполняется, то уравнение имеет мнимые корни, значения которых могут быть определены по формуле:

$$x_{1,2} = \frac{-B \pm i\sqrt{-D}}{2A} \qquad (1.5)$$

⇨*Табличное описание.* Табличное описание может иметь вид таблицы 1.1.

Таблица 1.1

Значение коэффициента	Значение дискриминанта	
A	D < 0	D ≥ 0
A = 0	$x_{1,2} = -\frac{C}{B}$	$x_{1,2} = -\frac{C}{B}$
A ≠ 0	$x_{1,2} = \frac{-B \pm i\sqrt{-D}}{2A}$	$x_{1,2} = \frac{-B \pm \sqrt{D}}{2A}$

⇨*Графическое описание.* В качестве примера рассматривается описание при помощи схемы алгоритма, которая приведена на рисунке 1.1.

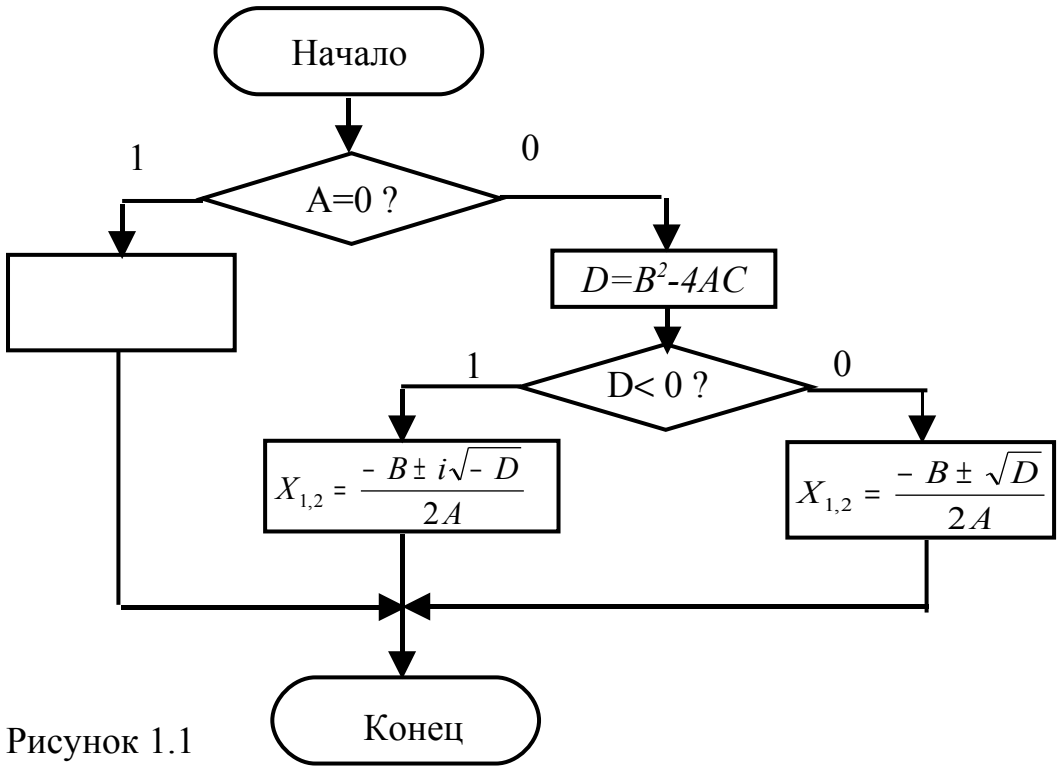


Рисунок 1.1

⇨*Аналитическое описание.* В данном случае аналитическое описание может иметь следующий вид:

Если $A = 0$, то $X_1 = X_2 = -\frac{C}{B}$, иначе вычислить $D=B^2-4AC$.

Если $D<0$, то $x_1 = \frac{-B - i\sqrt{-D}}{2A}$, $x_2 = \frac{-B + i\sqrt{-D}}{2A}$

иначе $x_1 = \frac{-B - \sqrt{D}}{2A}$, $x_2 = \frac{-B + \sqrt{D}}{2A}$

⇨*Описание на алгоритмическом языке.* В качестве примера приведем фрагмент описания алгоритма на языке Паскаль:

```
begin
        if  A = 0  then
                begin  X1 := - C/B;
                       X2 := X1;
                 end
           else
              begin   D := B*B - 4*A*C;
                 if  D < 0  then
                            begin

                                  X1:= (-B - i* SQRT(-D))/2/A;
                                  X2:= (-B + i* SQRT(-D))/2/A;
                             end
                           else
                                  begin
                                  X1 := (-B - SQRT(D))/2/A;
                                  X2 := (-B + SQRT(D))/2/A;
                                  end
                    end
end
```

1.1.3. Машина Тьюринга

Идея машины Тьюринга выдвинута английским математиком А.Тьюрингом в 1936 г. С точки зрения теории алгоритмов машина Тьюринга представляет собой средство для описания алгоритмов преобразования информации. Для понимания принципа ее работы машина Тьюринга представляется в виде следующего технического устройства, структура которого показана на рисунке 1.2.

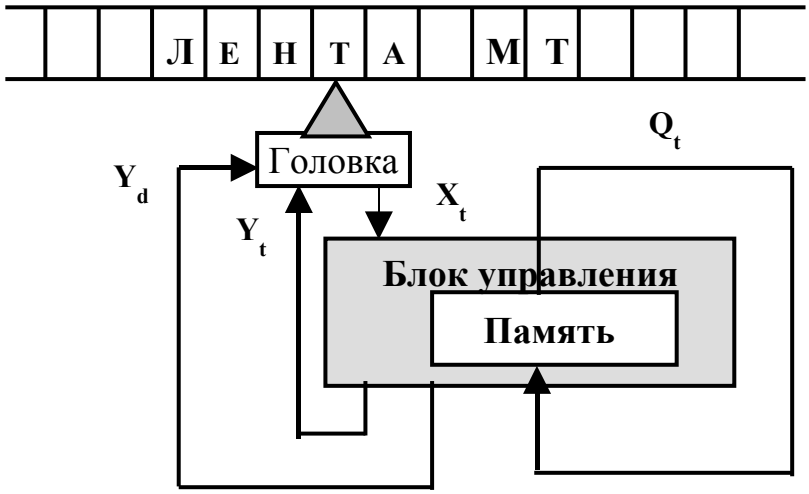


Рисунок 1.2

Машина Тьюринга состоит из ленты, головки и блока управления. Лента является внешней памятью машины. Лента служит для записи на нее исходных данных и результатов решения задачи. Физические принципы записи данных на ленту и чтения их с ленты не имеют значения. Лента бесконечна в обе стороны. Лента состоит из отдельных ячеек. В каждую ячейку может быть записан один символ. Запись и чтение данных с ленты производится при помощи головки. Будем считать, что при чтении информация в ячейке стирается.

Головка служит для записи данных на ленту и чтения информации с ленты. Головка может перемещаться по ленте. Перемещение головки может производиться на один шаг влево или вправо к соседней ячейке. Для перемещения головки используется специальный привод, не показанный на рисунке 1.2.

Блок управления обеспечивает взаимодействие всех частей машины. Он воспринимает символ, считанный с ленты, формирует новый символ, который записывается на ленту, а также управляет перемещением головки. Блок управления имеет внутреннюю память. Информация, записанная во внутренней памяти, представляет собой ***состояние*** машины. Состояние машины зависит от того, какая информация была считана с ленты до текущего момента времени. Состояние машины может изменяться после чтения и анализа очередного символа.

Для начала работы машины Тьюринга на ленте должна быть записана информация, которая подлежит преобразованию в процессе решения задачи. Головка должна быть установлена над первым обрабатываемым символом. Машина должна находиться в начальном состоянии.

Работа машины Тьюринга происходит по шагам. На каждом шаге выполняются следующие действия:

⇨С ленты считывается символ X_t, находящийся под головкой.

⇨В соответствии с алгоритмом решения задачи в зависимости от текущего состояния машины Q_t и считанного символа X_t блок управления формирует новый символ Y_t, который записывается на ленту вместо символа X_t.

⇨В соответствии с алгоритмом решения задачи блок управления формирует сигнал Y_d , который поступает на привод перемещения головки и вызывает перемещение головки.

⇨В соответствии с алгоритмом решения задачи блок управления формирует код нового состояния Q_{t+1} и записывает его во внутреннюю память машины.

Далее последовательность действий повторяется. При этом возможны два случая.

В *первом случае* после определенного количества шагов машина переходит в конечное состояние. Это означает, что решение задачи закончено, и результат ее решения записан на ленте.

Во *втором случае* машина не переходит в конечное состояние, и работа машины происходит бесконечно. Это означает, что алгоритма решения данной задачи не существует (или что алгоритм решения задачи сформулирован некорректно).

Алгоритм работы машины Тьюринга считается заданным, если заданы:

1 Алфавит входных сигналов $X_t = \{X_1, X_2, X_3 \dots X_k\}$.

2 Алфавит состояний $Q_t = \{Q_1, Q_2, Q_3 \dots Q_l\}$.

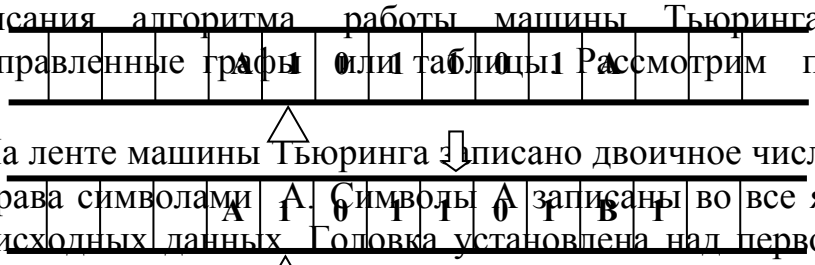
3 Алфавит выходных сигналов $Y_t = \{Y_1, Y_2, Y_3 \dots Y_n\}$.

4 Функция выходов $Yt = Y(Xt, Qt)$.

5 Функция переходов $Q_{t+1} = Q(X_t, Q_t)$.

6 Функция перемещения головки $Y_d = Y(X_t, Q_t)$.

Для описания алгоритма работы машины Тьюринга обычно используются направленные графы или таблицы. Рассмотрим пример такого описания.



Пример. На ленте машины Тьюринга записано двоичное число, ограниченное слева и справа символами А. Символы А записаны во все ячейки ленты, свободные от исходных данных. Головка установлена над первой слева цифрой. Машина должна просмотреть информацию на ленте и определить четность количества единиц в этом числе. После просмотра вместо правого символа А записывается символ В, и если количество единиц четное, то в следующую ячейку записывается символ Т, иначе – 0. Затем головка должна быть установлена в исходное положение.

Граф работы такой машины представлен на рисунке 1.3.

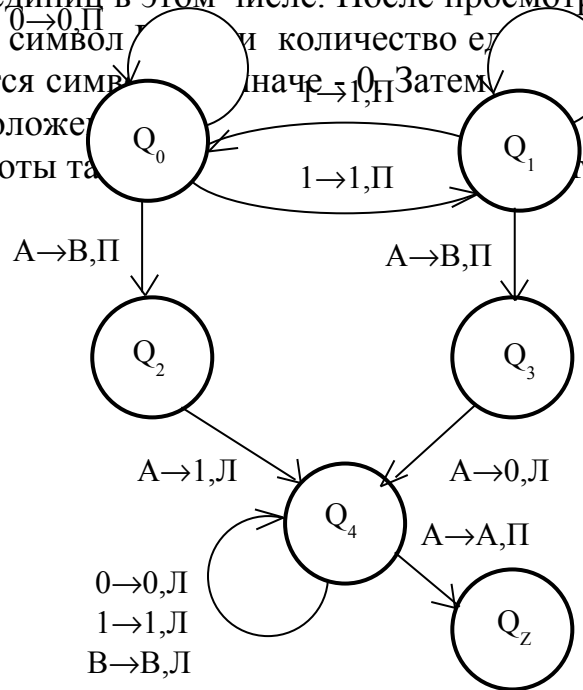


Рисунок 1.3

На рисунке 1.3 состояниям машины соответствуют вершины графа, переходам машины из одного состояния в другое - ребра графа. На ребрах графа для каждого входного сигнала X_t указываются выходной сигнал Y_t и направление перемещения головки (Л - влево, П - вправо, Ст - стоп).

При считывании каждой четной единицы машина переходит в состояние Q_0, при считывании нечетной - в состояние Q_1. Поэтому после просмотра всей ленты (т.е. при считывании правого символа А и замены его символом В) головка делает шаг вправо и на ленту записывается 1 или 0 в зависимости от того, в каком состоянии машины читается символ А. Далее в состоянии Q_4 выполняется перемещение головки в исходное положение. Работа машины заканчивается при переходе машины в конечное состояние Q_z.

В таблице работы машины Тьюринга для каждой комбинации текущего состояния Q_t входного символа X_t указываются:

- ⇨выходной символ Y_t;
- ⇨направление перемещения головки;
- ⇨новое состояние машины Q_{t+1}.

Таблица работы машины Тьюринга, соответствующая графу на рисунке 1.3 , имеет вид таблицы 1.2.

Таблица 1.2

Состояние машины	Входные символы			
	0	1	A	B
Q_0	0,П,Q_0	1,П,Q_1	B,П,Q_2	-
Q_1	0,П,Q_1	1,П,Q_0	B,П,Q_3	-
Q_2	-	-	1,Л,Q_4	-

Q_3	-	-	0,Л, Q_4	-
Q_4	0,Л,Q_4	1,Л,Q_4	А, П, Q_z	В,Л,Q_z
Q_z	-	-	-	-

Как видно из графа и таблицы, каждой паре символов X_t ,Q_t однозначно соответствует тройка символов Y_t, Y_d, Q_{t+1}. Таким образом, логика работы машины на каждом шаге полностью описывается пятеркой символов X_t, Q_t, Y_t, Y_d, Q_{t+1}. Эта пятерка символов называется ***командой*** машины Тьюринга.

Тогда таблица работы машины перечисляет все возможные команды, т.е. задает систему команд машины Тьюринга, представляющую собой алгоритм работы этой машины.

В качестве второго примера рассмотрим машину Тьюринга, которая выполняет сложение двух чисел, представленных в унарном коде. (Число в унарном коде записывается в виде последовательности символов 1, количество которых равно значению числа). Для разделения чисел на ленте используется символ *.

Для случая, когда ни одно из слагаемых не равно нулю, граф машины Тьюринга, решающей эту задачу, может иметь вид, приведенный на рисунке 1.4.

Формально решение задачи сводится к тому, что вместо символа * необходимо записать символ 1 и затем стереть одну из единиц. Так как головка должна быть установлена в исходное положение, то целесообразно стереть левую единицу. Поэтому после замены символа * организуется движение головки

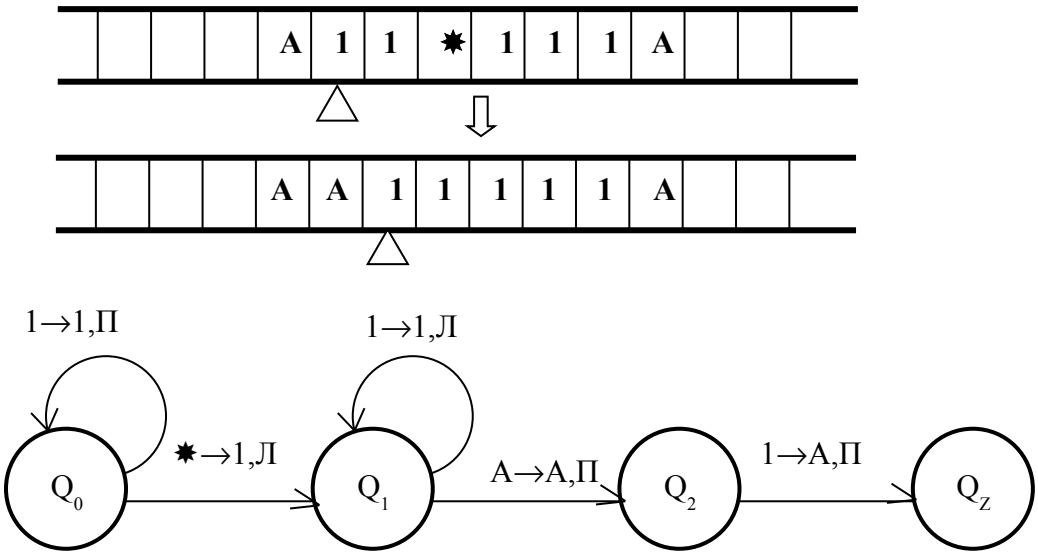


Рисунок 1.4

влево до пробела и замена левой единицы на пробел. Особенность данной задачи состоит в том, что алфавиты входных и выходных символов отличаются друг от друга.

Таблица работы машины Тьюринга приведена в виде таблицы 1.3.

Таблица 1.3

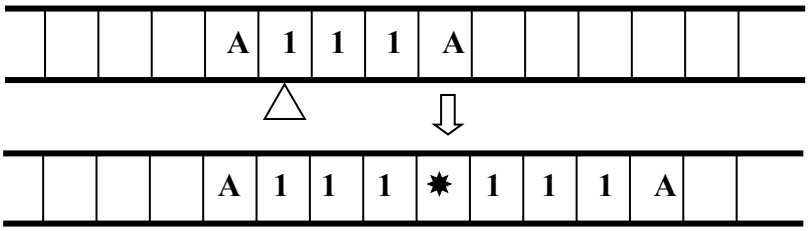
Состояние	Входные символы		
машины	*	1	А
Q_0	1, Л, Q_1	1, П, Q_0	-
Q_1	-	1, Л, Q_1	А, П, Q_2
Q_2	-	А, П, Q_z	-
Q_z	-	-	-

Приведем также пример задачи, при решении которой приходится вводить вспомогательные символы, которых нет ни в алфавите исходных данных, ни в алфавите выходных символов.

Пусть на ленте записана последовательность единиц, ограниченная слева и справа символами А. Необходимо сохранить на ленте исходную информацию и записать на эту же ленту ее копию. Исходную информацию и копию разделить символом *.

Идея решения задачи сводится к тому, что вначале все единицы заменяются вспомогательными символами (символами 0), а вместо правого символа А записывается символ разделителя *. Далее каждый символ 0 последовательно заменяется на символ 1 с одновременной записью символа 1 на ленту справа от разделителя. Граф машины Тьюринга, решающей эту задачу, приведен на рисунке 1.5.

Как видно из графа, задача решается за несколько проходов головки по ленте. Таблица работы машины, соответствующая графу рисунка 1.5, имеет вид таблицы 1.4.



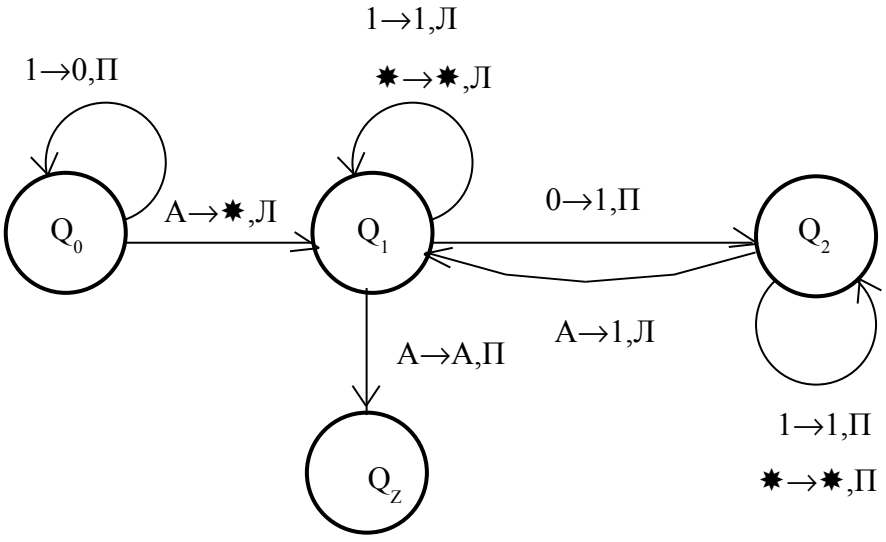


Рисунок 1.5

Таблица 1.4

Состояние машины	Входные символы			
	0	1	*	A
Q_0	-	0, П, Q_0	-	*, Л, Q_1
Q_1	1, П, Q_2	1, Л, Q_1	*, Л, Q_1	A, П, Q_z
Q_2	-	1, П, Q_2	*, П, Q_2	1, Л, Q_1
Q_z	-	-	-	-

При рассмотрении примеров можно сделать следующие выводы:

☞Решение задачи при помощи машины Тьюринга выполняется по шагам.

☞На каждом шаге выполняются элементарные операции:

- ☑ чтение символа из ячейки ленты;
- ☑ запись символа на ленту;
- ☑ перемещение головки на один шаг влево или вправо.

☞Количество шагов может быть большим.

☞Машины Тьюринга, реализующие различные алгоритмы, отличаются структурой блоков управления. Поэтому для решения различных задач необходимо строить различные машины, отличающиеся блоками управления. Это обстоятельство является одной из причин, по которым машина Тьюринга реально не была построена, а рассматривается лишь на абстрактном уровне.

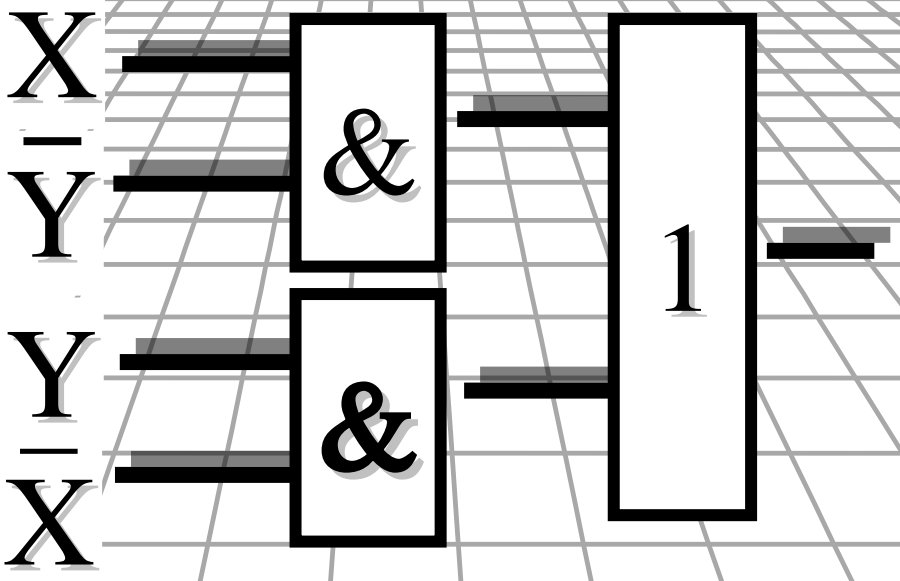
Однако в теории алгоритмов возникает необходимость построения алгоритмов решения при помощи машины Тьюринга различных задач, в том числе и достаточно сложных. Попытки уменьшить зависимость структуры машины от типа решаемых задач привели к идее создания универсальной машины Тьюринга.

Контрольные вопросы

✓1.Что такое алгоритм?
✓2.Перечислите основные свойства алгоритма.
✓3.Какими способами можно описать алгоритм?
✓4.Поясните состав машины Тьюринга.
✓5.Как происходит работа машины Тьюринга?
✓6.Что такое команда машины Тьюринга?
✓7.Какие операции выполняются на каждом шаге работы машины Тьюринга?
✓8.Чем отличаются машины Тьюринга, которые решают разные задачи?

✓9.Что такое состояние машины Тьюринга?
✓10.Для чего используется внутренняя память Тьюринга?

Раздел II.
Основы алгебры логики



$$F(X,Y)=X\oplus Y$$

2 .1 . Основные понятия

лгебра логики - это алгебра, образованная множеством {0,1} и всеми возможными операциями на нем. Алгебра логики является основным инструментом синтеза и анализа цифровых схем различного типа.

Функция алгебры логики (или логическая функция) $F(x_1,\ldots,x_n)$- это функция, принимающая значения 0 или 1, аргументы которой также принимают значения 0 или 1. Элементы 0 и 1 в данном случае не являются числами в обычном смысле, хотя по некоторым свойствам похожи на них. Чаще всего переменная 1 интерпретируется как «да» или «истина», а переменная 0 как «нет» или «ложь».

Если количество переменных равно n, то возможно 2^n различных наборов их значений. Так как для любого набора значений аргументов функция может принимать значение 0 или 1, то количество различных логических функций от n переменных равно 2^k , где $k = 2^n$.

Логические функции могут быть заданы различным образом, в том числе ***таблицей истинности***. Таблица истинности логической функции содержит все возможные наборы значений переменных и соответствующие им значения функции. Наборы значений переменных перечисляются в таблице в стандартном порядке, при котором наборы рассматриваются как двоичные числа и записываются в порядке их возрастания.

2.2. Элементарные логические функции

Среди всех функций алгебры логики выделяют функции одной и двух переменных, которые называют ***элементарными.***

Логических функций одной переменной всего 4. Объединенная таблица истинности этих функций приведена в виде таблицы 2.1.

Таблица 2.1

Переменная X	Значения логических функций			
	$F_0(x)$	$F_1(x)$	$F_2(x)$	$F_3(x)$
0	0	0	1	1
1	0	1	0	1

Функции $F_0(x)$ и $F_3(x)$ называют константами (генераторами) 0 и 1 соответственно [$F_0(x)=0$ и $F_3(x)=1$], функцию $F_1(x)$ – тождественной функцией [$F_1(x)=x$], а функцию $F_2(x)$-отрицанием (инверсией) переменной x или функ - цией НЕ [$F_2(x)=\overline{x}$].

Значения логических функций двух переменных (всего их 16) приведены в таблице 2.2.

Таблица 2.2

Значения переменных		Значения логических функций															
x_1	x_2	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Наиболее часто используемые логические функции имеют специальные названия.

Функции $F_0(x_1, x_2)$ и $F_{15}(x_1, x_2)$ называют константами (генераторами) 0 и 1 соответственно ($F_0(x_1, x_2)=0$, $F_{15}(x_1, x_2)=1$).

Функция $F_1(x_1, x_2)$ называется конъюнкцией x_1 и x_2 и обозначается как x_1 & x_2, или $x_1 \wedge x_2$, или $x_1 \bullet x_2$ (знак конъюнкции часто опускают). Конъюнкция x_1 и x_2 равна 1, если равны 1 соответственно x_1 и x_2, поэтому ее часто называют функцией И. Также ее называют логическим умножением, так как ее таблица совпадает с таблицей умножения для чисел 0 и 1.

Функцию $F_2(x_1, x_2)$ называют левой коимпликацией и обозначают как $x_1 \rightarrow \overline{x_2}$. $F_2(x_1, x_2) = x_1 \cdot \overline{x_2}$.

Функцию $F_4(x_1, x_2)$ называют правой коимпликацией и обозначают как $x_1 \leftarrow \overline{x_2}$. $F_4(x_1, x_2) = \overline{x_1} \cdot x_2$.

Функция $F_6(x_1, x_2)$ называется сложением по модулю два (отрицанием эквивалентности, неравнозначностью, исключающим ИЛИ), она принимает значение 1 только при различных значениях переменных. Эта функция обозначается как $x_1 \oplus x_2$.

Функцию $F_7(x_1, x_2)$ называют дизъюнкцией (логическим сложением, функцией ИЛИ). Она обозначается как $x_1 \vee x_2$. Эта функция равна 1, если равна 1 переменная x_1, или переменная x_2 или обе (все) переменные.

Функция $F_8(x_1, x_2)$ является отрицанием функции ИЛИ и называется функцией ИЛИ-НЕ (стрелкой Пирса). Функция ИЛИ-НЕ обозначается как $\overline{x_1 \vee x_2}$. Она принимает значение 1, только если обе (все) переменные равны 0.

Функция $F_9(x_1, x_2)$ называется функцией эквивалентности (равнозначности) и обозначается как $x_1 \sim x_2$ ($x_1 \equiv x_2$). Эта функция равна 1, только в случае, если значения всех переменных равны.

Функцию $F_{11}(x_1,x_2)$ называют правой импликацией. Она обозначается как $x_1 \leftarrow x_2$ ($x_2 \supset x_1$). $F_{11}(x_1,x_2) = \overline{x_1} \vee x_2$.

Функцию $F_{13}(x_1,x_2)$ называют импликацией (левой). Она обозначается как $x_1 \rightarrow x_2$ ($x_1 \supset x_2$). Эту функцию можно выразить также, как $F_{13}(x_1, x_2) = \overline{x_1} \vee x_2$.

Функция $F_{14}(x_1, x_2)$ является отрицанием конъюнкции и называется функцией И-НЕ (функцией Шеффера). Она принимает значение 1, только если хотя бы одна переменная равна 0. $F_{14}(x_1,x_2)= \overline{x_1 \cdot x_2}$.

Функции $F_3(x_1, x_2)$, $F_5(x_1, x_2)$, $F_{10}(x_1, x_2)$ и $F_{12}(x_1, x_2)$ являются вырожденными и фактически являются функциями одной переменной:

$F_3(x_1, x_2) = x_1$, $F_5(x_1, x_2) = x_2$, $F_{10}(x_1, x_2) = \overline{x_2}$, $F_{12}(x_1, x_2) = \overline{x_1}$.

Из рассмотренных выше элементарных функций при синтезе цифровых схем наиболее часто используются функции И, ИЛИ, НЕ, И-НЕ и ИЛИ-НЕ.

2.3.Формы логических функций

Используя принцип суперпозиции, можно из элементарных функций построить логическую функцию любого числа переменных. Логическая функция в общем случае может быть записана в различной форме. Наиболее часто используют дизъюнктивные и конъюнктивные нормальные формы.

Дизъюнктивная нормальная форма (ДНФ) логической функции представляет собой дизъюнкцию элементарных конъюнкций. ***Элементарная конъюнкция*** – это конъюнкция переменных или их отрицаний. Например, $F(x_1, x_2, x_3) = x_1 \overline{x_2} \vee \overline{x_1} x_3 \vee \overline{x_2} \overline{x_3}$ – функция, записанная в ДНФ.

Аналогично определяется конъюнктивная нормальная форма. ***Конъюнктивная нормальная форма (КНФ)*** логической функции представляет собой конъюнкцию элементарных дизъюнкций. ***Элементарная дизъюнкция*** – это дизъюнкция переменных или их отрицаний.

Например, $F(x_1, x_2, x_3) = (x_1 \vee x_2)\ (\overline{x_1} \vee x_3)\ (\overline{x_1} \vee \overline{x_2} \vee x_3)$.

Совершенная дизъюнктивная нормальная форма (СДНФ) – это ДНФ, каждая конъюнкция которой содержит все переменные в прямом или инверсном виде.

Например, $F(x_1, x_2, x_3) = x_1 \overline{x_2}\, x_3 \vee x_1 x_2 x_3 \vee x_1 x_2 \overline{x_3}$.

Аналогично определяется совершенная конъюнктивная нормальная форма. ***Совершенная конъюнктивная нормальная форма (СКНФ)*** – это КНФ, каждая дизъюнкция которой содержит все переменные в прямом или инверсном виде.

Например, $F(x_1, x_2, x_3) = (\overline{x_1} \,\&\, x_2 \vee \overline{x_3})\ (x_1 \vee \overline{x_2} \vee x_3)\ (x_1 \vee x_2 \vee x_3)$.

Любая логическая функция имеет только одну совершенную ДНФ или КНФ. Совершенные формы логической функции могут быть записаны по таблице истинности.

При записи СДНФ по таблице истинности каждому набору значений переменных, на котором функция равна 1, соответствует конъюнкция всех переменных функции. Если значение переменной в наборе равно 1, то переменная входит в конъюнкцию в прямом виде, иначе – в инверсном. Дизъюнкция всех полученных конъюнкций образует СДНФ.

При записи СКНФ каждому набору значений переменных, на котором функция равна 0, соответствует дизъюнкция всех переменных функции. Если значение переменной в наборе равно 0, то переменная входит в дизъюнкцию в прямом виде, иначе – в инверсном. Конъюнкция всех полученных дизъюнкций образует СКНФ.

Пример. Пусть логическая функция трех переменных описана следующей таблицей истинности (таблица 2.3).

Таблица 2.3.

Значения переменных			Значения функции
a	b	c	F(abc)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Тогда функция будет записана в СДНФ в следующим виде:

$$F(abc) = \overline{a}\,\overline{b}\,\overline{c} \vee \overline{a}\,\overline{b}\,c \vee \overline{a}\,b\,\overline{c} \vee a\,\overline{b}\,\overline{c} \vee a\,b\,c.$$

СКНФ этой функции будет иметь следующий вид:

$$F(abc) = (a \vee b \vee c)\ (a \vee b \vee \overline{c})\ (a \vee \overline{b} \vee \overline{c}).$$

Обычно при разработке цифровых схем используются дизъюнктивные нормальные формы логических функций.

Кроме нормальных (дизъюнктивных и конъюнктивных) форм логических функций могут использоваться и другие, например скобочные формы. Скобочные формы записываются с использованием скобок, уточняющих порядок выполнения логических операций. Скобочные формы в некоторых случаях позволяют уменьшить сложность схемы.

2.4. Законы алгебры логики

Реализация логических функций связана с выполнением эквивалентных преобразований для получения минимальных форм используемых функций. Эквивалентные преобразования изменяют только форму логической функции, не изменяя ее таблицу истинности. Такие преобразования выполняются с использованием законов алгебры логики.

Основные законы алгебры логики имеют следующий вид:

1. $\overline{\overline{a}} = a$. Закон двойного отрицания.

2. $a \vee 1 = 1$; $a \& 1 = a$. Законы одинарных элементов.
   $a \vee 0 = a$. $a \& 0 = 0$.

3. $a \vee a = a$; $a \& a = a$. Закон тавтологии (идемпотентности).

4. $a \vee \overline{a} = 1$; $a \& \overline{a} = 0$. Закон исключенного третьего (противоречия).

5. $a \vee b = b \vee a$; $a \& b = b \& a$. Коммуникативный закон.

6. $a \vee (b \vee c) = (a \vee b) \vee c$. Ассоциативный закон.
   $a \& (b \& c) = (a \& b) \& c$.

7. $a \& (b \vee c) = a \& b \vee a \& c$. Дистрибутивный закон.
   $a \vee b \& c = (a \vee b) \&(a \vee c)$.

8. $\overline{a \& b} = \overline{a} \vee \overline{b}$. Закон отрицания отрицания.
   $\overline{a \vee b} = \overline{a} \& \overline{b}$. (Закон де Моргана).

9. $a \vee a \& b = a$. Закон поглощения.
   $a \&(a \vee b) = a$.

10. $a \& b \vee a \& \overline{b} = a$. Закон склеивания.
    $(a \vee b) \& (a \vee \overline{b}) = a$.

11. $a \vee \overline{a} \& b = a \vee b$. Закон Порецкого.
    $a \&(\overline{a} \vee b) = a \& b$.

Следует отметить, что в соответствии с принципом суперпозиции законы алгебры логики применимы не только к простым переменным, но и к функциям.

Пример.

$F(abc) = ac \vee ac\,(b \vee c) = ac$ (закон поглощения).

Законы алгебры логики используются также для вычисления значений логических функций по известным значениям всех переменных. При этом следует учитывать порядок выполнения логических операций с учетом их старшинства:

1. Действия в скобках.
2. Одноместные операции (отрицание или инверсия).
3. Конъюнкция (И).
4. Дизъюнкция (ИЛИ).
5. Сложение по модулю 2.

Пример. Пусть задана логическая функция вида

$F(abc) = \bar{a}\bar{b}c \vee \bar{a}\bar{b}c \vee a\bar{b}c \vee ab\bar{c} \vee abc.$

Если задано

$a = 0;\ b = 1;\ c = 0.$ то

$F(abc) = 0\cdot\bar{1}\cdot\bar{0} \vee 0\cdot1\cdot\bar{0} \vee 0\cdot1\cdot0 \vee 0\cdot1\cdot0 \vee \bar{0}\cdot1\cdot0 = 1\cdot1\cdot1 \vee 0\cdot0\cdot1 \vee 0\cdot0\cdot0 \vee$

$\vee\ 0\cdot1\cdot1 \vee 0\cdot1\cdot0 = 1 \vee 0 \vee 0 \vee 0 \vee 0 \vee 0 = 1.$

Полученное значение совпадает с приведенным в таблице истинности (таблица 2.3) для заданного набора значений переменных.

Правильность выполнения эквивалентных преобразований можно проверить путем составления таблицы истинности для исходной и преобразованной форм логической функции так, как это показано в таблице 2.4 на примере закона Порецкого для дизъюнкции ($a \vee ab = a \vee b$).

Таблица 2.4

Значения переменных		Значения функций			
a	b	$\bar{a}$	$\bar{a}b$	$a \vee \bar{a}b$	$a \vee b$
0	0	1	0	0	0
0	1	1	1	1	1
1	0	0	0	1	1
1	1	0	0	1	1

Как видно из таблицы 2.4, значения функций $a \vee \bar{a}b$ и $a \vee b$ совпадают при всех значениях переменных, т.е. эти функции эквивалентны.

2.5. Техническая реализация логических функций

Вычисление значений логических функций для любых наборов значений переменных выполняется с помощью ***логических схем***. Логические схемы на-

зывают также ***комбинационными схемами***, так как значение сигнала на выходе такой схемы зависит только от комбинации значений сигналов на входе схемы. Комбинационная схема строится из ***логических элементов***. Логические элементы выполняют (реализуют) элементарные логические функции.

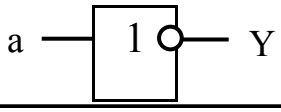
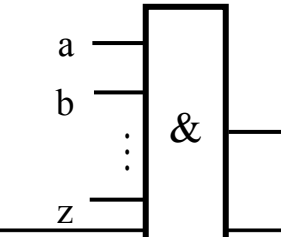
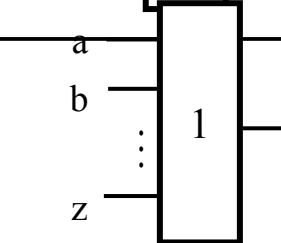
При создании комбинационных схем необходимо стремиться не только к минимальному количеству элементов, но и к их минимальному разнообразию (минимальному числу типов элементов). Набор типов элементов, при использовании которого можно построить любую комбинационную схему, называется ***функционально полной системой логических элементов***.

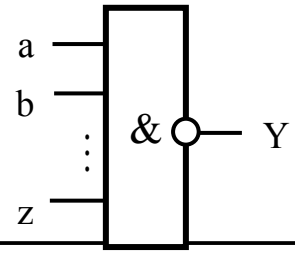
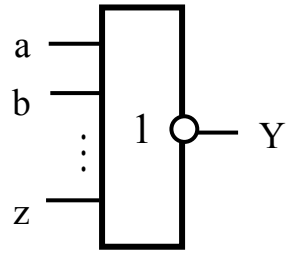
Аналогично можно поставить вопрос о минимальном наборе элементарных логических функций, с помощью которого можно записать любую логическую функцию. Такой набор называется ***функционально полной системой логических функций***. Существует несколько функционально полных систем. Например, известно, что любую логическую функцию можно записать в СДНФ. В такой форме используются только элементарные логические функции И, ИЛИ и НЕ. Поэтому логические функции И, ИЛИ и НЕ составляют функционально полную систему. Эта система представляется вполне естественной, но она избыточна. Доказано, что функционально полную систему могут образовать функции И и НЕ, а также функции ИЛИ и НЕ. Кроме того, функция И-НЕ, как и функция ИЛИ-НЕ, также представляет собой функционально полную систему.

Применительно к логическим элементам это означает, что функционально полную систему элементов могут составлять элементы И, ИЛИ, НЕ или И-НЕ, а также элемент ИЛИ-НЕ. В настоящее время для построения цифровых схем чаще всего используют элементы И-НЕ. Таким образом все электронные схемы компьютера можно построить на элементах одного типа, а именно на элементах И-НЕ (ИЛИ-НЕ).

На схемах логические элементы изображаются в виде условных графических обозначений (УГО). Примеры УГО некоторых логических элементов приведены в таблице 2.5.

Таблица 2.5

Обозначение	Выполняемая функция	Название
	$Y=\overline{a}$	Элемент НЕ (инвертор)
	Y=a&b&…&z	Элемент И (конъюнктор)
	Y=a∨b∨…∨z	Элемент ИЛИ

		(дизъюнктор)
	$Y=\overline{a\&b\&\ldots\&z}$	Элемент И-НЕ
	$Y=\overline{a\vee b\vee\ldots\vee z}$	Элемент ИЛИ-НЕ

Вычерчивание электронных схем, реализующих заданные логические функции, выполняется путем замены элементарных логических функций соответствующими логическими элементами. Затем элементы объединяют с учетом старшинства логических функций. Если реализуемая функция записана в СДНФ, то логическая схема состоит из трех ярусов. В первом ярусе располагаются инверторы, во втором – элементы И. Выходные сигналы элементов И поступают на входы элемента ИЛИ, представляющего третий ярус схемы. Число входов элемента ИЛИ равно числу конъюнкций в СДНФ. В качестве примера на рисунке 2.1 приведена функциональная схема, реализующая СДНФ логической функции, полученной из таблицы 2.3.

Следует отметить, что реализованная на рисунке 2.1 функция не минимизирована.

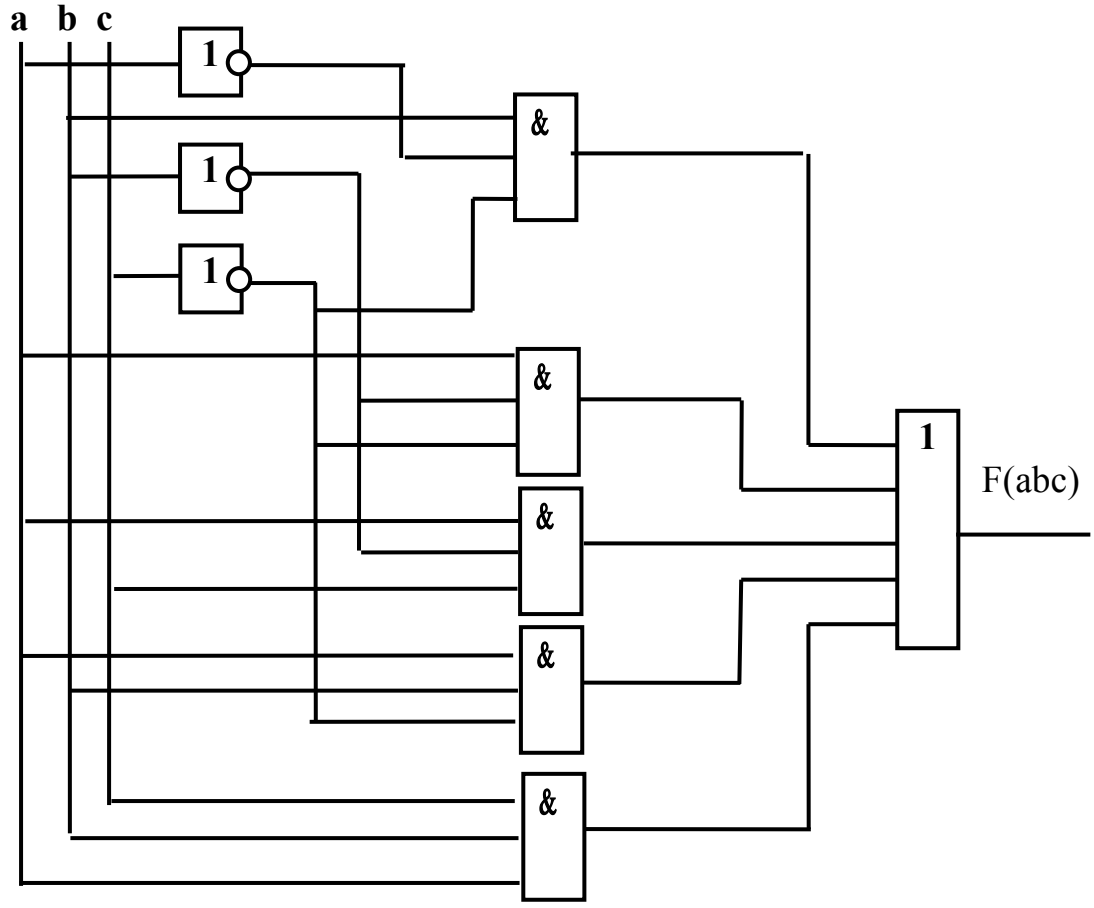


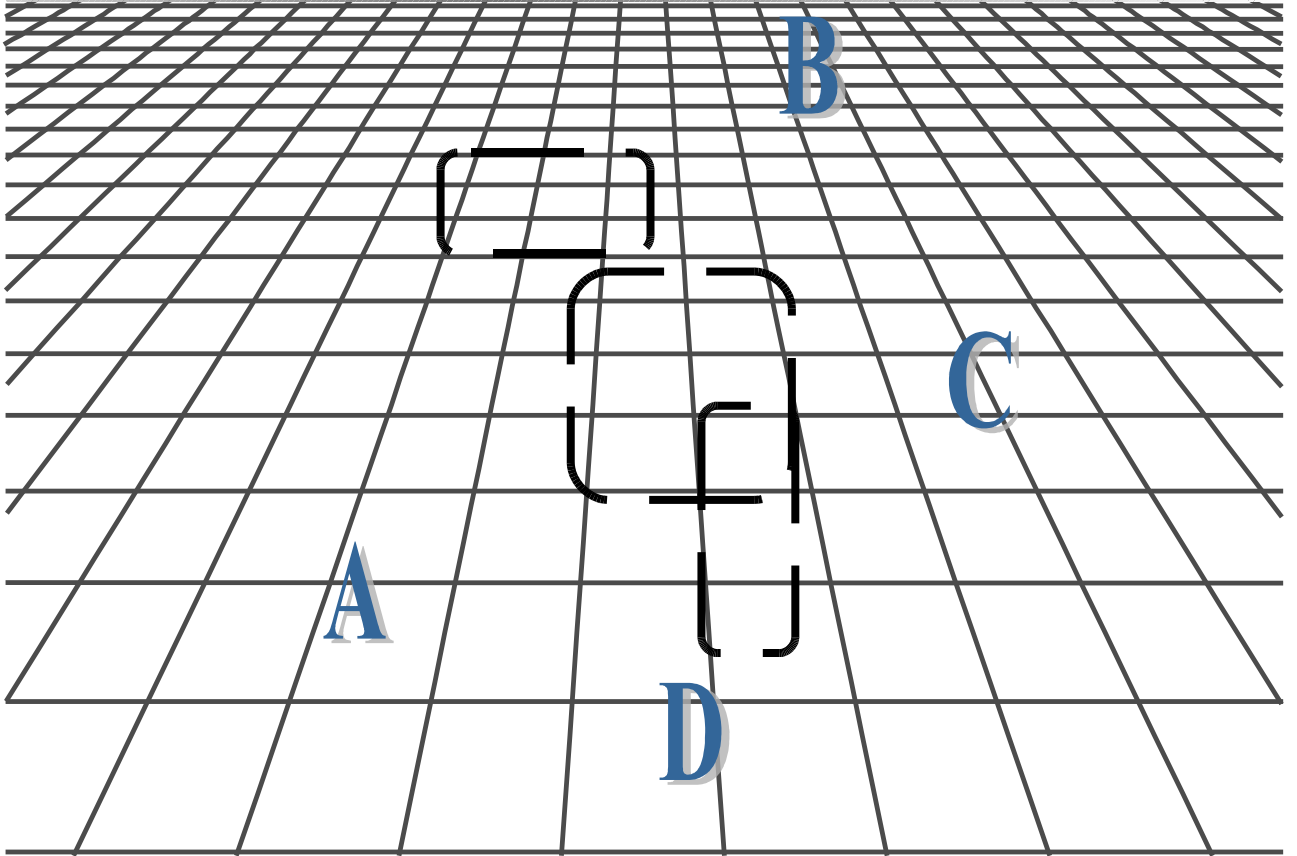
Рисунок 2.1

Контрольные вопросы

1. Что такое логическая функция?
2. Что представляет собой таблица истинности логической функции?
3. Что такое элементарная логическая функция?
4. Что такое дизъюнктивная нормальная форма логической функции?
5. Что такое совершенная дизъюнктивная нормальная форма логической функции (СДНФ)?
6. Что такое конъюнктивная нормальная форма логической функции?
7. Что такое совершенная конъюнктивная нормальная форма логической функции (СКНФ)?
8. Как записать СДНФ логической функции по таблице истинности?
9. Как проверить правильность законов алгебры логики с помощью таблицы истинности?
10. В чем заключаются эквивалентные преобразования логической функции?

11. Что такое логический элемент?
12. Что такое функционально полная система логических элементов?
13. Приведите примеры функционально полных систем логических элементов.
14. Как построить комбинационную схему по СДНФ логической функции?
15. Как вычислить значение логической функции при заданных значениях переменных?

Раздел III.
Комбинационные схемы



3.1. Анализ и синтез комбинационных схем

3.1.1. Типы цифровых автоматов

В зависимости от типа элементов, из которых построен автомат, различают два основных типа цифровых автоматов:

- комбинационные схемы;
- цифровые автоматы с памятью.

Комбинационные схемы состоят только из логических элементов (И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ и т.д.). Комбинационные схемы могут иметь несколько входов и несколько выходов (рисунок 3.1).

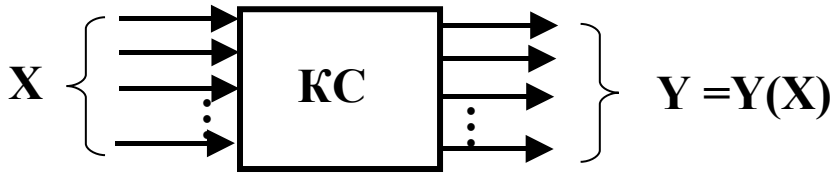


Рисунок 3.1

Главной особенностью комбинационных схем является то, что сигнал на выходе комбинационных схем зависит только от комбинации сигналов на входах схемы и не зависит ни от времени, ни от предыстории входных сигналов. Таким образом, при многократном поступлении одного и того же входного сигнала на выходах комбинационной схемы будет формироваться один и тот же входной сигнал.

Цифровые автоматы с памятью состоят из логических элементов и элементов памяти (рисунок 3.2). Информация, записанная в элементах памяти такого автомата, называется ***состоянием*** цифрового автомата. Основная особенность цифровых автоматов с памятью состоит в том, что сигнал на выходе автомата зависит как от входного сигнала, так и от состояния автомата.

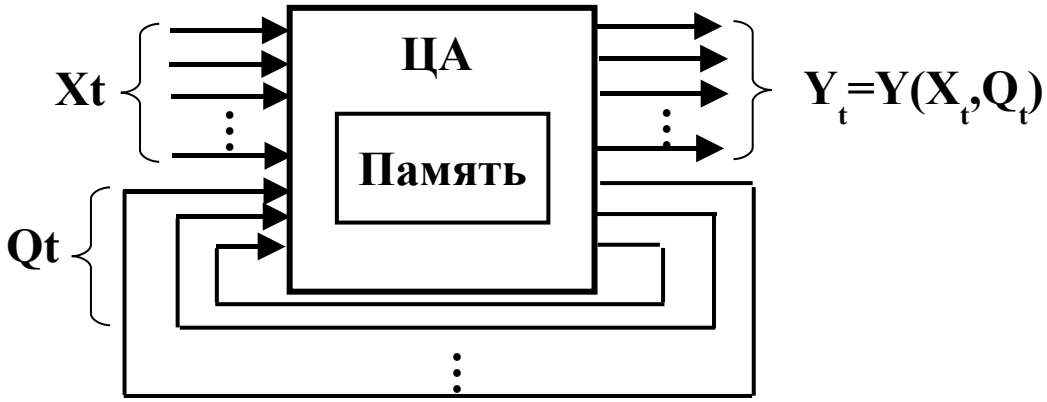


Рисунок 3.2

В свою очередь, состояние автомата зависит от того, какие сигналы поступали на его входы до настоящего момента времени, т.е. от предыстории работы автомата. Поэтому при многократном поступлении одного и того же входного сигнала на выходах автомата с памятью могут формироваться различные сигналы.

При сопоставлении комбинационных схем и цифровых автоматов с памятью можно сделать вывод, что комбинационные схемы имеют единственное состояние. Комбинационные схемы используются для технической реализации логических функций. При помощи комбинационной схемы определяется значение логической функции для заданных значений логических переменных. В теории автоматов обычно считается, что при изменении сигналов на входах

схемы выходной сигнал мгновенно принимает соответствующее значение, т.е. задержка сигналов логическими элементами не учитывается.

В цифровых автоматах с памятью формирование выходного сигнала занимает определенное время, которое обычно разбивается на такты. В каждом такте в зависимости от входного сигнала и состояния автомата формируется выходной сигнал и новое состояние. Таким образом, можно сказать, что выходной сигнал автомата с памятью зависит от последовательности входных сигналов, поэтому цифровые автоматы с памятью называют также последовательностными схемами.

3.1.2. Задачи анализа и синтеза комбинационных схем

Структурно комбинационная схема может быть представлена как совокупность элементарных логических схем – логических элементов. Логические элементы выполняют над входными переменными элементарные логические операции типа **И-НЕ, И, ИЛИ, ИЛИ-НЕ** и т.д. Число входов логического элемента соответствует числу аргументов воспроизводимой им булевой функции. Графическое изображение комбинационной схемы, при котором показаны связи между различными элементами, а сами элементы представлены условными обозначениями, называется ***функциональной*** схемой.

Основными характеристиками комбинационных схем являются сложность и быстродействие.

Сложность комбинационной схемы оценивается количеством оборудования, составляющего схему. При разработке схем на основе конкретной элементной базы количество оборудования обычно измеряется числом корпусов (модулей) интегральных микросхем, используемых в схеме. В теоретических разработках ориентируются на произвольную элементную базу и поэтому для оценки затрат оборудования используется оценка сложности схем по Квайну.

Сложность (цена) по Квайну определяется суммарным числом входов логических элементов в составе схемы. При такой оценке единица сложности – один вход логического элемента. Цена инверсного входа обычно принимается равной двум (если на вход схемы сигналы поступают только в прямой форме). Такой подход к оценке сложности оправдан по следующим причинам:

- сложность схемы легко вычисляется по логическим функциям, на основе которых строится схема: для ДНФ сложность схемы равна сумме количества букв (букве со знаком отрицания соответствует цена 2) и количества знаков дизъюнкции, увеличенного на 1 для каждого дизъюнктивного выражения;
- все классические методы минимизации логических функций обеспечивают минимальность схемы именно в смысле цены по Квайну.

Практика показывает, что схема с минимальной ценой по Квайну обычно реализуется наименьшим числом конструктивных элементов – корпусов интегральных микросхем.

Быстродействие комбинационной схемы оценивается максимальной задержкой сигнала при прохождении его от входа схемы к выходу, т.е. определяется промежутком времени от момента поступления входных сигналов до момента установления соответствующих значений выходных сигналов. Задержка сигнала кратна числу элементов, через которые проходит сигнал от входа к выходу схемы. Поэтому быстродействие схемы характеризуется значением $r\tau$, где τ - задержка сигнала на одном элементе. Значение r определяется количеством уровней комбинационной схемы, которое рассчитывается следующим образом. Входам КС приписывается уровень ***нулевой***. Логические элементы, связанные только с входами схемы относятся к уровню ***первому***. Элемент относится к уровню k, если он связан по входам с элементами уровней k-1, k-2, и т.д. Максимальный уровень элементов r определяет количество уровней КС, называемое ***рангом схемы***. Пример определения ранга r схемы приведён на рисунке 3.3.

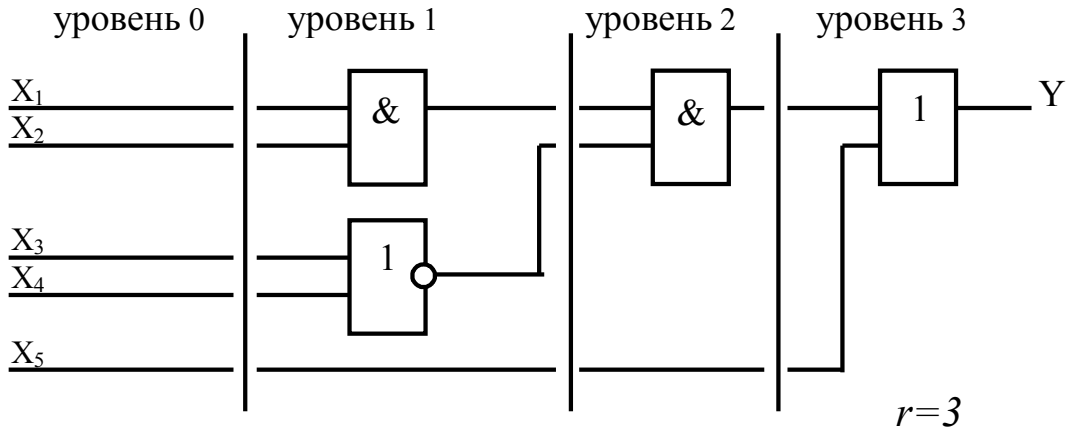


Рисунок 3.3

Как известно, любая логическая функция может быть представлена в ДНФ, которой соответствует двухуровневая комбинационная схема. Следовательно, быстродействие любой КС в принципе можно довести до 2τ.

Минимизация булевой функции с целью уменьшения сложности схем обычно приводит к необходимости представления функций в скобочной форме, которой соответствуют схемы с $r > 2$. Таким образом, уменьшение затрат оборудования в общем случае приводит к снижению быстродействия схем.

В ходе разработки комбинационных схем приходится решать задачи анализа и синтеза.

Задача анализа состоит в определении статических и динамических свойств комбинационной схемы. В статике определяются логические функ-

ции, реализуемые комбинационной схемой по известной ей структуре. В динамике рассматривается способность надёжного функционирования схемы в переходных процессах при смене значений переменных на входах схемы, т.е. определяется наличие на выходах схемы возможных нежелательных импульсных сигналов, которые не следуют непосредственно из выражений для логических функций, реализуемых схемой.

Задача синтеза заключается в построении из заданного набора логических элементов комбинационной схемы, реализующей заданную систему логических функций.

Решение задачи синтеза не является однозначным, можно предложить различные варианты комбинационных схем, реализующих одну и ту же систему логических функций, но отличающихся по тем или иным параметрам. Разработчик комбинационных схем из этого множества вариантов выбирает один, исходя из дополнительных критериев: минимального количества логических элементов, необходимых для реализации схемы, максимального быстродействия и т.д. Существуют различные методы синтеза комбинационных схем, среди которых наиболее разработан канонический метод.

3.1.3. Анализ комбинационных схем

Задачи анализа комбинационных схем (КС) возникают при необходимости проверить правильность синтеза (на этапе проектирования) или определить логическую функцию, реализуемую КС (при анализе или ремонте схем). Все существующие методы анализа делятся на ***прямые*** и ***косвенные***.

В результате анализа КС ***прямым*** методом получается множество наборов входных переменных, обеспечивающих заданное значение на выходе, что позволяет записать в алгебраическом виде логическую функцию, реализуемую схемой. К прямым методам относится метод π- алгоритма.

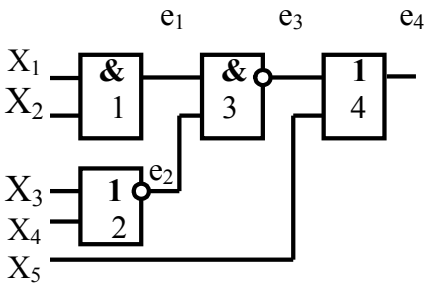
Применение ***косвенных*** методов дает возможность определить реакцию схемы на заданный набор входных переменных в статике или проанализировать переходный процесс смены одного входного набора на другой. Примерами косвенных методов анализа являются методы синхронного и асинхронного моделирования.

Все упомянутые методы анализа являются машиноориентированными, что позволяет выполнить анализ схемы на ЭВМ.

Для всех методов анализа необходимо описать схему в виде схемного списка, в который включаются в общем случае следующие данные:

- номер ЛЭ в схеме;
- логическая функция, реализуемая ЛЭ;
- входные переменные для данного ЛЭ.

Например, на рисунке 3.4 представлена схема и список, описывающий схему.



N элем.	Функция	Входы
1	2И	X1 X2
2	2ИЛИ-НЕ	X3 X4
3	2И-НЕ	e_1 e_2
4	2ИЛИ	e_3 X5

Рисунок 3.4

Метод π- алгоритма

При анализе комбинационных схем методом π-алгоритма определяются наборы входных переменных, обеспечивающих заданное значение выходного сигнала КС. Наборы, обеспечивающие на выходе КС логическую 1, образуют так называемое ***единичное покрытие*** C^1. Аналогично входные наборы, обеспечивающие на выходе КС логический 0, образуют ***нулевое покрытие*** C^0. Рассмотрим покрытия C^0 и C^1 для простейшего логического элемента 2И, выполняющего функцию $Y=X_1X_2$. Таблица истинности для этой функции и условное обозначение элемента 2И приведены на рисунке 3.5.

X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

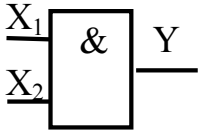


Рисунок 3.5

Как видно из приведенной таблицы только при единственном наборе $X_1=1$ и $X_2=1$ на выходе ЛЭ будет 1, т.е. единичное покрытие включает только один набор $C^1=\{1\ 1\}$. На выходе ЛЭ будет 0 при трех наборах, образующих нулевое покрытие:

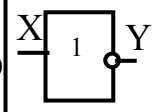
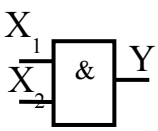
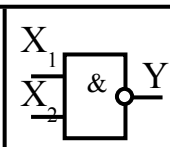
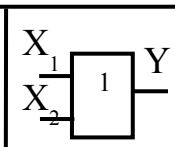
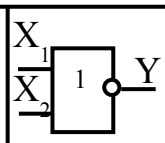
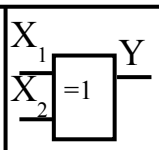
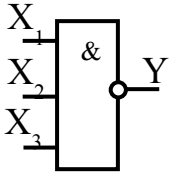
$$C^0 = \begin{bmatrix} 00 \\ 01 \\ 10 \end{bmatrix}.$$

Это покрытие можно упростить, заметив, что первый набор склеивается со вторым и третьим, т.е.

$$C^0 = \begin{bmatrix} 0X \\ X0 \end{bmatrix}$$

Таким образом, для ЛЭ 2И можно сказать, что 1 на его выходе будет только при обеих единицах на входах, а для обеспечения 0 на выходе достаточно подать хотя бы на один вход 0. Рассуждая аналогично, получим таблицу покрытий C^0 и C^1 для основных ЛЭ, представленных ниже в таблице 3.1.

Таблица 3.1

ЛЭ							
	НЕ X	2И X_1 X_2	2И – НЕ X_1 X_2	2ИЛИ X_1 X_2	2ИЛИ–НЕ X_1 X_2	ИСК. ИЛИ X_1 X_2	3И – НЕ X_1 X_2 X_3
C^0	1	0 X X 0	1 1	0 0	1 X X 1	0 0 1 1	1 1 1
C^1	0	1 1	0 X X 0	1 X X 1	0 0	0 1 1 0	0 X X X 0 X X X 0

При анализе схемы методом **π**-алгоритма, задавшись определенным значением сигнала на выходе, заменяют его соответствующим покрытием элемента, формирующего выходной сигнал. В результате этого определяется, какие сигналы должны быть на выходах элементов, подключенных к выходному ЛЭ. В свою очередь, сигналы на выходах этих элементов можно заменить соответствующими покрытиями, т.е. определить значения выходных сигналов для других ЛЭ и т.д. Этот процесс продолжается до тех пор, пока не получатся покрытия, состоящие только из входных переменных, называемых опорными. Совокупность таких покрытий и дает соответствующее покрытие схемы.

Пример анализа комбинационной схемы (рисунок 3.4) методом π - алгоритма представлен в таблицах 3.2 и 3.3. В последней колонке этой таблицы приведен оператор подстановки, в результате работы которого сигнал на выходе ЛЭ заменяется соответствующим покрытием. Необходимо обратить внимание, что все значения переменных, записанные в одной строке, должны одновременно быть в наличии для обеспечения заданного значения выходного сигнала. Поэтому, при замене одного из значений в строке соответствующим покрытием, все остальные значения для других переменных в этой строке должны присутствовать совместно с этим покрытием.

На основании полученного единичного покрытия можно записать логическую функцию, реализуемую схемой:

$$Y = X_5 \vee \bar{X}_1 \vee \bar{X}_2 \vee X_3 \vee X_4 .$$

В дальнейшем можно сравнить полученную логическую функцию с той,

по которой строилась схема, и проверить правильность ее построения. При анализе схемы может оказаться, что некоторая переменная, получившая на одном из предыдущих шагов некоторые значения, на данном шаге должна принять противоположное значение. Возникшее противоречие говорит о том, что данный путь является тупиковым и его необходимо исключить из дальнейшего рассмотрения. Если ни при одной комбинации входных переменных не обеспечивается значение 1(0) на выходе, то это означает, что схема реализует константу 0(1) соответственно.

Таблица 3.2

X_1	X_2	X_3	X_4	X_5	e_1	e_2	e_3	e_4	Оператор
--	--	--	--	--	--	--	--	1	———
--	--	--	--	--	--	--	1	--	$П\,^{1}_{4}$ 2ИЛИ X_5e_3
--	--	--	--	1					
--	--	--	--	--	x	0			$П\,^{1}_{3}$ 2И-НЕ e_1e_2
--	--	--	--	--	0	x			
0	x	--	--	--					$П\,^{0}_{1}$ 2И X_1X_2
x	0	--	--	--					
--	--	1	x	--					$П\,^{0}_{2}$ 2ИЛИ-НЕ X_3X_4
--	--	X	1	--					

$$C^1 = \begin{bmatrix} x\,x\,x\,x\,1 \\ 0\,x\,x\,x\,x \\ x\,0\,x\,x\,x \\ x\,x\,1\,x\,x \\ x\,x\,x\,1\,x \end{bmatrix}$$

а) Получение единичного покрытия C^1

Таблица 3.3

X_1	X_2	X_3	X_4	X_5	e_1	e_2	e_3	e_4	Оператор
--	--	--	--	--	--	--	--	0	———
--	--	--	--	0	--	--	0	0	$П^0_4$ 2ИЛИ $X_5\,e_3$
--	--	--	--	0	1	1			$П^0_3$ 2И-НЕ $e_1\,e_2$
1	1	--	--	0		1			$П^1_1$ 2И $X_1\,X_2$
1	1	0	0	0					$П^1_2$ 2ИЛИ-НЕ $X_3\,X_4$

$C^0 = \{11000\}$

б) Получение нулевого покрытия C^0

Метод синхронного моделирования

При данном методе считается, что все ЛЭ переключаются одновременно, без задержки. В результате применения метода определяется установившееся значение сигнала на выходе схемы.

Рассмотрим метод синхронного моделирования на примере схемы на рисунке 3.4. На первом этапе схему разбиваем на уровни и записываем в порядке возрастания уровня уравнения, описывающие функционирование ЛЭ (таблица 3.4).

Таблица 3.4

Уровень	№ элемента	Уравнение
1	1 2	$e_1 = X_1 \,\&\, X_2$ $e_2 = \overline{X_2 \vee X_2}$
2	3	$e_3 = \overline{e_1 \,\&\, e_2}$
3	4	$Y = e_4 = e_3 \vee X_5$

Проанализируем схему при подаче на вход набора $X_1=0$, $X_2=0$, $X_3=0$, $X_4=1$, $X_5=1$. Для этого решаем записанные уравнения в порядке возрастания уровня. Имеем:

$$e_1 = \underline{X_{1\,\&}\ X_2} = \underline{0\&0} = 0;$$

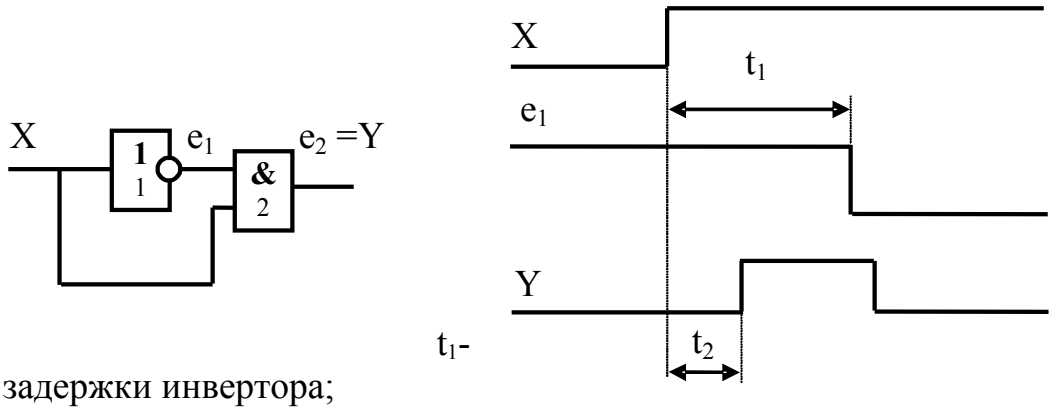
$$e_2 = \underline{X_{3\,\vee}\ X_4} = \underline{0\vee 1} = 0;$$

$$e_3 = e_{1\,\&}\ e_2 = 0\&0 = 1;$$

Следовательно, при подаче на вход набора {00011}, на выходе будет Y=1. Аналогично можно промоделировать работу схемы при подаче на вход любого другого набора.

Метод асинхронного моделирования

Реальный логический элемент ЛЭ переключается за какое-то конечное время, зависящее от технологии изготовления, условий эксплуатации, емкостей нагрузки и т.д. Прохождение сигнала последовательно через несколько ЛЭ будет приводить к накоплению времени задержки и возникновению сдвига во времени выходного сигнала по отношению ко входному. Наличие задержки и порождаемого ею временного сдвига сигналов может приводить к появлению на выходе отдельных ЛЭ и всей схемы в целом кратковременных сигналов, не предусмотренных БФ, реализуемой схемой. Как иллюстрацию рассмотрим схему на рисунке 3.6 .



t_1- время задержки инвертора;

t_2-время задержки элемента 2И

Рисунок 3.6

Данная схема реализует функцию $Y = X \&\overline{X} = 0$, т.е. константу 0 независимо от входного сигнала X. Однако в переходном процессе в результате за-

держки срабатывания ЛЭ возможна ситуация, когда на обоих входах элемента 2И будут логические единицы, что может привести к появлению на выходе схемы логической 1 (см. рисунок 3.6 б). Рассмотренный случай возможен при задержке срабатывания второго элемента больше, чем первого. Такое явление называется ***риском сбоя***. Различают статистический и динамический риски сбоя.

При ***статическом*** риске сбоя до и после переходного процесса состояние выходного сигнала одно и то же, а во время переходного процесса возможно кратковременное появление противоположного сигнала.

При ***динамическом*** риске сбоя до и после переходного процесса состояния выходного сигнала противоположные, но в переходном процессе выходной сигнал несколько раз меняет свое значение. Динамический риск сбоя возможен в схеме (рисунок 3.7).

В данном примере динамический риск сбоя на выходе КС сопровождается статическим риском на выходе элемента 1. Как видно из временных диаграмм, риск сбоя имеет место при наличии определенного временного сдвига между сигналами, поступающими на вход ЛЭ. Нежелательные сигналы на выходе могут и отсутствовать при другом соотношении временных сигналов, однако принципиальная возможность их появления является фактором, снижающим надежность работы схемы. Поэтому очень важно уметь обнаруживать и устранять такие явления.

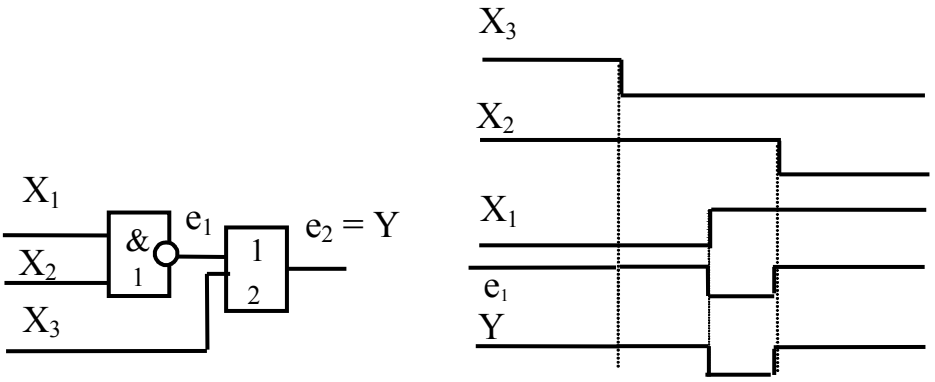


Рисунок 3.7

3.1.4. Канонический метод синтеза комбинационных схем

Задача синтеза решается в тех случаях, когда необходимо построить комбинационную схему с заданной логикой работы. Логика работы комбинационной схемы представляет собой правила соответствия входных и выходных сигналов. Логика работы комбинационных схем может быть описана одним из следующих способов:

- ☑ словесное описание;
- ☑ описание при помощи таблиц;
- ☑ аналитическое описание.

Особенности различных способов описания будут рассмотрены далее на примерах.

Синтез комбинационных схем в общем случае выполняется в следующем порядке:

- ☞ формализация задания;
- ☞составление таблицы истинности;
- ☞ запись логической функции в СДНФ (в совершенной дизъюнктивной нормальной форме);
- ☞минимизация логической функции;
- ☞ выбор системы логических элементов;
- ☞ преобразование логической функции к виду, удобному для реализации на выбранной системе элементов;
- ☞составление функциональной схемы;
- ☞ проверка работоспособности синтезированной схемы.

Содержание операций на каждом этапе синтеза рассмотрим на примере.

Пример. Синтезировать мажоритарную схему голосования 2 из 3-х.

Такая схема имеет три входа и один выход. На каждый вход может поступать сигнал 0 или 1. Сигнал на выходе схемы выбирается из значений входных сигналов по принципу большинства, как при голосовании. Выходной сигнал равен 1, если, по крайней мере, сигналы на двух входах равны 1.

☞1.Формализация задания. При формализации задания определяется количество входов и выходов схемы, определяется алфавит входных и выходных сигналов, принимаются обозначения входов и выходов. В данном случае количество входов и выходов известно. Обозначения входов и выходов, а также алфавит входных и выходных сигналов показаны на рисунке 3.8.

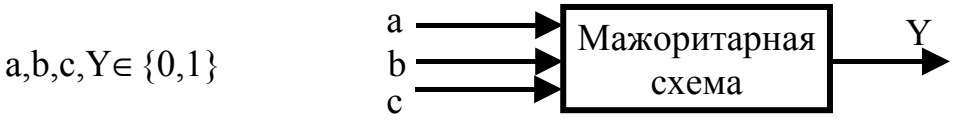


Рисунок 3.8

☞2.Составление таблицы истинности. В таблице истинности перечисляются все комбинации; входных сигналов и для каждой из них задается значение сигнала на выходе схемы. Поэтому число столбцов таблицы равно суммарному количеству входов и выходов схемы, а число строк - количеству комбинаций входных сигналов. Это количество определяется путем возведения числа 2 в степень, равную числу входов схемы. Для данного примера число строк табли-

цы (не считая заглавной) равно 8. Значения выходного сигнала в таблице истинности определяется в соответствии с назначением синтезируемой схемы.

Таблица истинности для мажоритарной схемы голосования 2 из 3-х имеет вид таблицы 3.5.

Таблица 3.5

Входы			Выход
a	b	c	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

☞3.Запись логической функции в СДНФ. По таблице истинности запишем логическую функцию Y в следующем виде:

$$Y = \bar{a}bc \;\&\; a\bar{b}c \;\&\; ab\bar{c} \;\&\; abc \qquad (3.1)$$

☞4. Минимизация логической функции. В формуле (3.1.) каждая из первых трех конъюнкций может быть объединена с последней, например:

$$\bar{a}bc \vee abc = bc\,(\bar{a} \vee a) \qquad (3.2)$$

В результате минимизации получим минимальную форму логической функции, которая имеет следующий вид:

$$Y = ab \vee ac \vee bc \qquad (3.3)$$

☞5. Выбор системы логических элементов. Пусть задана система элементов И, ИЛИ, НЕ. В этом случае логическую функцию не нужно преобразовывать.

☞6. Преобразование логической функции. Преобразование не выполняется.

☞7.Составление функциональной схемы. При составлении схемы проводится анализ логической функции. В данном случае логическая функция состоит из трех конъюнкций, соединенных знаком дизъюнкции. Идея составления схемы заключается в том, что каждая элементарная логическая операция (конъюнкция и

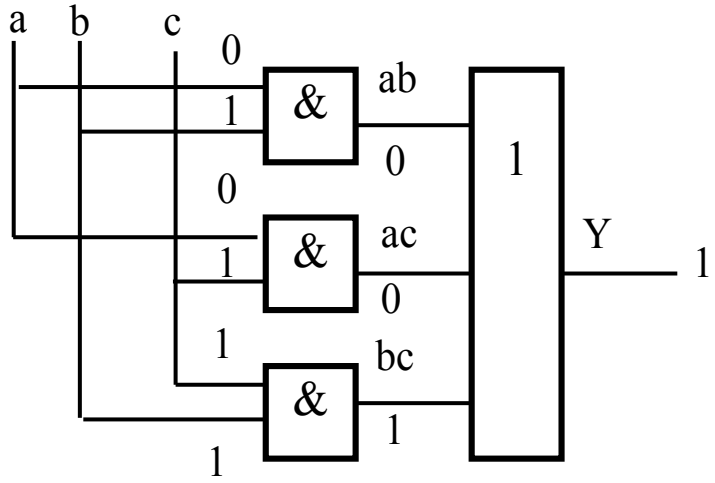


Рисунок 3.9

дизъюнкция) реализуется соответствующим логическим элементом (элементом И и элементом ИЛИ). Тогда для данной схемы потребуются три элемента И на два входа каждый и один элемент ИЛИ на три входа. Общий вид схемы приведен на рисунке 3.9.

☞8.Проверка работоспособности схемы. Работоспособность схемы проверяется методом синхронного моделирования. Для каждой комбинации входных сигналов в соответствии с логикой работы используемых элементов определяются значения сигналов на входах и выходах всех элементов, начиная с входных сигналов схемы. Значения выходных сигналов должны совпадать с данными таблицы истинности. При большом количестве входов такая проверка требует много времени и обычно выполняется при помощи специальных программ на ЭВМ.

На рисунке 3.9 показаны значения сигналов для комбинации входных сигналов a = 0, b = 1, c = 1. При этом значение выходного сигнала (Y = 1) совпадает с данными таблицы истинности.

Рассмотрим пример синтеза схемы, имеющей четыре входа.

Пример. Пусть необходимо построить комбинационную схему, которая сравнивает два двухразрядных двоичных числа (A и B) и выдает сигнал 1, если A < B.

☞1.Формализация задания. Пусть A = ab и B = cd, где a и c - старшие разряды чисел A и B, b и d - младшие разряды. Тогда схема сравнения может иметь общий вид, приведенный на рисунке 3.10.

Рисунок 3.10

☞2.Составление таблицы истинности. Таблица истинности для данной схемы приведена в виде таблицы 3.6.

Таблица 3.6

Входы				Выход
A		B		
a	b	c	d	Y
0	0	0	0	0
0	0	0	1	**1**
0	0	1	0	**1**
0	0	1	1	**1**
0	1	0	0	0

0	1	0	1	0
0	1	1	0	**1**
0	1	1	1	**1**

Продолжение таблицы 3.6

1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	**1**
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

☞3. Запись логической функции в СДНФ. Из таблицы 3.6 получим:

$$Y = \bar{a}\bar{b}\bar{c}d \vee \bar{a}\bar{b}c\bar{d} \vee \bar{a}b\bar{c}\bar{d} \vee a\bar{b}\bar{c}\bar{d} \vee \bar{a}bcd \vee a\bar{b}cd. \quad (3.4)$$

☞4.Минимизация логической функции. Используя известное соотношение

$$F \vee \bar{F} = 1, \quad (3.5)$$

перепишем (3.4) в следующем виде:

$$Y = (\bar{a}\bar{b}c\bar{d} \vee \bar{a}\bar{b}cd) \vee (\bar{a}bcd \vee \bar{a}bc\bar{d}) \vee (\bar{a}\bar{b}cd \vee a\bar{b}cd) \vee (\bar{a}\bar{b}cd \vee \bar{a}\bar{b}\bar{c}d) =$$
$$(\bar{a}\bar{b}c)(d \vee \bar{d}) \vee \bar{a}bc(d \vee \bar{d}) \vee b\bar{c}d(a \vee \bar{a}) \vee \bar{a}bd(c \vee \bar{c}) = (\bar{a}bc \vee$$
$$\vee \bar{a}\bar{b}c) \vee b\bar{c}d \vee \bar{a}bd = \bar{a}c(b \vee \bar{b}) \vee b\bar{c}d \vee \bar{a}bd = \bar{a}c \vee b\bar{c}d \vee \bar{a}bd \quad (3.6)$$

☞5.Выбор системы элементов. Пусть задана система элементов И, ИЛИ, НЕ.

☞5 , 6.При заданной системе элементов преобразование не требуется.

☞7.Построение функциональной схемы. Функциональная схема, соответствующая логической функции (3.6), приведена на рис. 3.11.

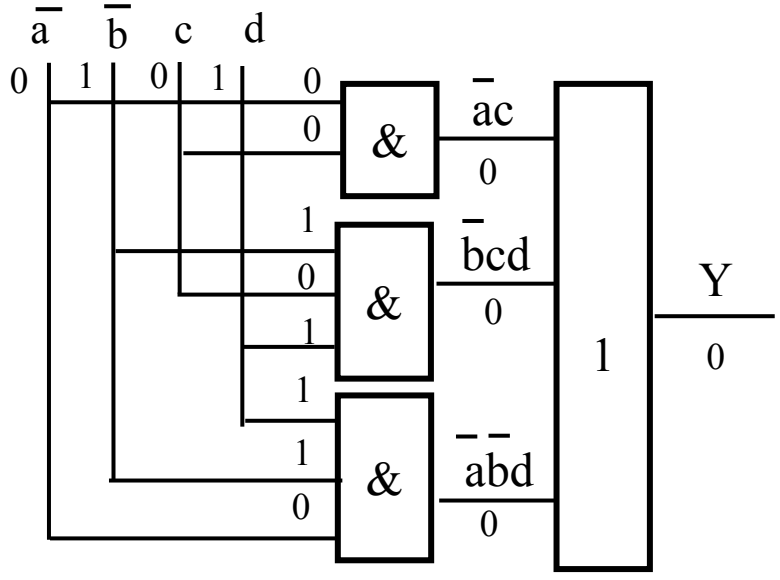


Рисунок 3.11

☞8. Проверка работоспособности схемы. Проверим работу схемы для случая, когда на вход схемы поступает следующая комбинация сигналов:

a=1, b=0,c=0, d=l (A=10, B=01, A>B).

Значения сигналов на входах и выходах элементов схемы показаны на рисунке 3.11. При заданных входных сигналах на выходе схемы формируется выходной сигнал Y = 0, что соответствует таблице истинности.

Контрольные вопросы

✓1.В чем заключается формализация задания при синтезе комбинационной схемы?
✓2.Что такое совершенная дизъюнктивная нормальная форма логической функции?
✓3.Как записать логическую функцию по таблице истинности?
✓4.Для чего выполняется минимизация логических функций?
✓5.Как по логической функции составить комбинационную схему?
✓6.Как проверить правильность работы комбинационной схемы?
✓7.Почему при синтезе комбинационной схемы на элементах И-НЕ (ИЛИ-НЕ) необходимо преобразование дизъюнктивной формы логической функции?
✓8.В чем заключается задача синтеза комбинационной схемы?
✓9.В чем заключается задача анализа комбинационной схемы?
✓10.От чего зависит значение сигнала на выходе комбинационной схемы?
✓11. Как оценивается сложность комбинационной схемы?
✓12. Что такое ранг схемы?
✓13. Как составляется схемный список?
✓14. Что такое единичное и нулевое покрытие?
✓15. Поясните сущность метода π-алгоритма.
✓16. В чем заключается метод синхронного моделирования?
✓17. В чем заключается метод асинхронного моделирования?
✓18. Что такое риск сбоя?

3.2. Минимизация логических функций

3.2.1. Методы минимизации логических функций

Любая логическая функция может быть записана в различной форме. Различные формы логической функции могут иметь различную сложность, т.е. для их реализации в виде комбинационной схемы может требоваться различное количество логических элементов. Поэтому перед реализацией логической функции должна быть предпринята попытка минимизировать (упростить) эту функцию. Для минимизации логических функций могут быть использованы различные методы.

По форме представления исходной логической функции различают следующие методы минимизации:

- аналитические методы;
- графические методы.

В аналитических методах минимизации логическая функция представляется обычно в СДНФ. Из этих методов чаще всего используются метод непосредственных преобразований и метод Квайна.

В графических методах логическая функция задается в виде графа или в виде специальной диаграммы. Наиболее известным из графических методов является метод Карно.

По степени приближения результата минимизации к минимальной форме логической функции различают точные и приближенные методы минимизации. Точные методы позволяют получить строго минимальную форму логиче-

ской функции. К таким методам относятся метод Квайна и метод интервалов. Точные методы требуют много времени для их реализации, поэтому часто применяют приближенные методы. Эти методы не гарантируют получения минимальной формы. Результатом реализации таких алгоритмов является форма функции, близкая к минимальной (может быть и минимальная, но не всегда). Достоинством приближенных методов минимизации является приемлемое время их выполнения. Приближенным методом является, например, метод непосредственных преобразований.

3.2.2.Общая последовательность минимизации

Основной операцией, которая обеспечивает упрощение формы логической функции, является операция склеивания, сущность которой может быть записана следующим образом:

$$ab \vee a\bar{b} = a \qquad (3.7)$$

или в общем виде:

$$aF \vee a\bar{F} = \qquad (3.8)$$

где F - произвольная логическая функция.

Одна и та же конъюнкция может склеиваться с несколькими другими конъюнкциями. В этом случае используется правило неполного склеивания:

$$aF \vee a\bar{F} = a \vee aF \vee a\bar{F} \qquad (3.9)$$

При неполном склеивании результат склеивания дополняется исходными конъюнкциями, которые участвовали в склеивании. Неполное склеивание не приводит к усложнению функции, так как исходные конъюнкции могут быть при необходимости исключены из логической функции путем использования правила поглощения:

$$a \vee aF = a \qquad (3.10)$$

Введем некоторые определения. ***Импликантой*** данной логической функции называется другая логическая функция, которая принимает значение 1 только на части входных комбинаций, на которых данная логическая функция равна 1. Рассмотрим пример (таблица 3.7).

Таблица 3. 7

Переменные			Функции		
a	b	c	F	F1	F2
0	0	0	1	1	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	1	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	0	0	0

1	1	1	1	1	0

В таблице 3.7 заданы значения некоторой функции F. При этом функции F1 и F2 являются импликантами функции F.

Логические функции часто задаются в СДНФ. Для краткости иногда вместо записи СДНФ функция задается перечислением десятичных номеров наборов значений переменных, на которых функция принимает значение «1».

Каждая конъюнкция СДНФ задает один набор значений переменных, на которых логическая функция принимает значение 1 (единичный набор). Поэтому единичный набор является импликантой. Импликантой является также результат склеивания импликант между собой. Импликанты, полученные при склеивании, задают уже не один, а несколько единичных наборов, т.е. интервал единичных значений или единичный интервал. Единичный интервал может включать 1, 2, 4, 8 и т.д. единичных наборов. Таким образом, понятия импликанты и единичного интервала являются синонимами и отличаются лишь по форме. Импликанты записываются в буквенной форме, а единичные интервалы – в цифровой форме (в виде двоичных или десятичных чисел).

Импликанты могут склеиваться друг с другом и порождать новые, более простые, импликанты.

Если импликанта не склеивается с другими импликантами, она называется ***простой импликантой***, а соответствующий ей единичный интервал называется ***максимальным единичным интервалом***.

Каждая логическая функция имеет определенный набор простых импликант. Дизъюнкция всех простых импликант данной логической функции называется ***сокращенной ДНФ*** (сокращенной дизъюнктивной нормальной формой) данной логической функции.

Пример. Пусть $F(abcd) = \overline{a}\overline{b}cd \vee a\overline{b}cd \vee \overline{a}bcd \vee a\overline{b}\overline{c}d \vee ab\overline{c}\overline{d} \vee abcd$ (2, 4, 6, 9, 12, 14)

В данном примере над каждой конъюнкцией указан ее десятичный номер. После склеивания получим сокращенную ДНФ исходной функции:

$F(abcd) = b\overline{d} \vee \overline{a}c\overline{d} \vee \overline{a}\overline{b}cd$

Сокращенная ДНФ данной функции включает три простых импликанты (максимальных единичных интервала). Для записи интервалов в цифровой форме следует записать последовательность символов «0», «1» и «-», длина которой равна числу переменных. Если импликанта содержит переменную в прямой форме, вместо переменной в последовательности записывается символ «1». Вместо переменной в инверсной форме записывается символ «0», а вместо отсутствующих переменных записывается символ «-». Например импликанте bd соответствует интервал ***-1-0***. Так интервалы записываются в двоичной форме.

Для записи десятичной формы интервала следует вместо символов «-» записать все возможные комбинации символов «0» и «1». При этом образуются двоичные номера всех единичных наборов, входящих в данный интервал.

Далее записываются десятичные номера наборов, соответствующих нижней и верхней границам интервала так, как это показано ниже.

$\bar{b}\bar{d} \rightarrow$ ***-1-0*** $\rightarrow$ 0***10*** $\rightarrow$ (4, 14)
0***11***
1***10***
1***11***

$\bar{a}c\bar{d} \rightarrow$ ***0-10*** $\rightarrow$ ***0*****10** $\rightarrow$ (2, 6)
01***10***

$a\bar{b}\bar{c}d \rightarrow$ ***1001*** $\rightarrow$ (9, 9)

В данном примере символы «0» и «1» исходных интервалов выделены полужирным курсивом.

В сокращенной ДНФ могут быть лишние импликанты, т.е. такие, при отбрасывании которых значение логической функции не меняется. При отбрасывании лишних импликант из сокращенной ДНФ получается ***тупиковая*** форма исходной логической функции. Особенность тупиковой формы заключается в том, что эта форма не может быть упрощена.

Логическая функция может иметь несколько тупиковых форм. Различные тупиковые формы одной и той же логической формы могут иметь различную сложность. Для сравнительной оценки сложности логических функций используют такие показатели, как суммарное число букв в логической функции, суммарное число букв плюс число конъюнкций в логической функции, суммарное количество входов элементов, необходимых для реализации логической функции (сложность по Квайну) и др.

При наличии нескольких тупиковых форм логической функции одна из тупиковых форм, имеющая наименьшую сложность, называется ***минимальной*** формой данной логической функции.

В общем случае последовательность минимизации логических функций может иметь вид, показанный на рисунке 3.12.

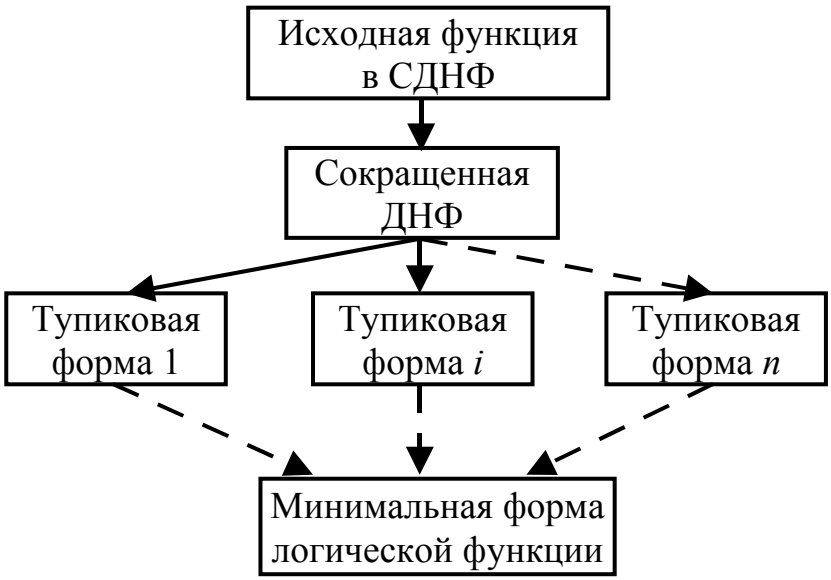


Рисунок 3.12

Минимизируемая логическая функция представляется обычно в совершенной дизъюнктивной нормальной форме (СДНФ). Путем применения логических операций неполного склеивания и поглощения из СДНФ получают сокращенную ДНФ исходной логической функции. После отбрасывания лишних импликант из сокращенной ДНФ получают несколько тупиковых форм логической функции. Далее из тупиковых форм по критерию сложности выбирается минимальная форма исходной логической функции.

Приведенная последовательность реализуется в точных методах минимизации. В приближенных методах минимизации находится лишь одна из тупиковых форм, которая и принимается за минимальную форму логической функции.

Конкретный набор операций и их последовательность при минимизации зависят от используемого метода.

3.2.3. Метод непосредственных преобразований

Метод непосредственных преобразований является аналитическим приближенным методом минимизации. Результат минимизации существенно зависит от квалификации исполнителя. Поэтому метод не является строго алгоритмическим и не гарантирует получения не только минимальной, но даже и тупиковой формы логической функции. Однако этот метод широко используется в инженерной практике для минимизации простых логических функций.

Сущность метода заключается в последовательном применении к исходной логической функции приведенных выше операций склеивания, неполного склеивания и поглощения до тех пор, пока эти операции применимы к исходной форме логической функции или к промежуточным формам исходной функции.

В процессе минимизации могут быть полезными также следующие известные соотношения и правила алгебры логики:

$$x\&x\&x\&x\&x\&\ldots\&x = x; \quad x\vee x\vee x\vee x\vee x\vee\ldots\vee x = x;$$
$$x\&\bar{x} = 0; \quad x\vee\bar{x} = 1;$$
$$x\&1 = x; \quad x\vee 1 = 1;$$
$$x\&0 = 0; \quad x\vee 0 = x; \qquad (3.11)$$
$$\bar{\bar{x}} = x; \quad \overline{x\vee x} = \bar{x}\,\bar{x}$$

В некоторых случаях полезно использовать также правило свертки

$$x \vee \bar{x}y = x \vee y \qquad (3.12)$$

или правило Порецкого

$$xy\vee \bar{x}z \vee yz = xy\vee \bar{x}z$$

(3.13)

Рассмотрим примеры.

Пример. Минимизировать логическую функцию F(abc) методом непосредственных преобразований.

$$F(abc) = \bar{a}\,\bar{b}\,\bar{c} \vee \bar{a}\,b\,\bar{c} \vee a\,\bar{b}\,\bar{c} \vee abc \qquad (3.14)$$

Так как первая конъюнкция может склеиваться со второй и с третьей, то в результате применения операции склеивания получим:

$$F(abc) = (\bar{a}\,\bar{b}\,\bar{c} \vee \bar{a}\,b\,\bar{c}) \vee (\bar{a}\,\bar{b}\,\bar{c} \vee a\,\bar{b}\,\bar{c}) \vee a\,b\,c = \bar{a}\,\bar{c} \vee \bar{b}\bar{c} \vee a\,b\,c \qquad (3.15)$$

В данном примере одна из конъюнкций дважды участвует в склеивании, а последняя конъюнкция не склеивается с другими.

Пример. Минимизировать логическую функцию F(abcd) методом непосредственных преобразований.

$$F(abcd) = a\,b\,c\,d \vee a\,b\,c\,\bar{d} \vee a\,b\,\bar{c}d \vee a\,b\,\bar{c}\,\bar{d} \vee \bar{a}\,b\,c\,d \vee \bar{a}\,\bar{b}\,\bar{c}\,\bar{d} \qquad (3.16)$$

Последовательно применяя операцию склеивания, получим:

$$F(abcd) = (a\,b\,c\,d \vee a\,b\,c\,\bar{d}) \vee (a\,b\,\bar{c}d \vee a\,b\,\bar{c}\,\bar{d}) \vee (a\,\bar{b}\,c\,d \vee \bar{a}\,b\,c\,d)$$
$$\vee \vee \bar{a}\bar{b}cd = (abc \vee \bar{a}bc) \vee bcd \vee \bar{a}bcd = ab \vee bcd \vee \bar{a}bcd \qquad (3.17)$$

В данном примере конъюнкции, полученные в результате склеивания, склеиваются затем между собой. При каждом склеивании число букв в конъюнкциях (ранг конъюнкции) уменьшается на единицу. Поэтому при повторном склеивании вместо конъюнкций из четырех букв получена одна конъюнкция из двух букв.

При большом количестве логических переменных применение метода непосредственных преобразований становится затруднительным, но принципиально возможным. Но при этом трудно гарантировать даже получение хотя бы одной из тупиковых форм.

Приведем пример минимизации логической функции пяти переменных:

$$F(abcdf) = \bar{a}\bar{b}\bar{c}d\bar{f} \vee abcdf \vee \bar{a}bcdf \vee ab\bar{c}df \vee \bar{a}b\bar{c}df \vee \bar{a}\bar{b}cdf \vee abcd\bar{f} \vee$$
$$\vee \bar{a}bcdf \vee ab\bar{c}df.$$

$$F(abcdf) = (\bar{a}\bar{b}\bar{c}d\bar{f} \vee \bar{a}\bar{b}\bar{c}df) \vee (\bar{a}bcd\bar{f} \vee \bar{a}bcdf) \vee (ab\bar{c}d\bar{f} \vee ab\bar{c}df) \vee$$
$$\vee (abcd\bar{f} \vee abcdf) \vee (\bar{a}bc\bar{d}f \vee \bar{a}bcdf) = (\bar{a}\bar{b}cd \vee \bar{a}bcd) \vee (a\bar{b}cd \vee abcd) \vee$$
$$\vee \bar{a}\,b\,c\,f = (\bar{a}bd \vee abd) \vee \bar{a}bcf = bd \vee \bar{a}bcf \qquad (3.19)$$

Из данного примера следует, что даже для такой сравнительно несложной функции поиск конъюнкций, которые могут быть склеены, представляет значительные трудности. Так как при минимизации выполняется большое количество сравнительно несложных операций, для упрощения реальных логических функций, которые встречаются в инженерной практике, целесообразно использовать машинные программы. Для минимизации же логических функ-

ций с числом переменных не более четырех широкое применение нашли графические методы, в частности метод Карно.

3.2.4. Метод Карно

Метод Карно представляет собой приближенный метод минимизации логических функций, который при определенных навыках позволяет получить минимальную форму исходной функции. Исходная логическая функция представляется в СДНФ. Минимизированная логическая функция представлена в дизъюнктивной нормальной форме.

Сущность метода Карно заключается в построении специальных диаграмм, представляющих собой прямоугольные таблицы. Каждой конъюнкции исходной логической функции соответствует одна из клеток таблицы. Главная особенность диаграмм Карно заключается в том, что конъюнкциям, которые могут быть склеены, соответствуют соседние по строке или по столбцу клетки.

Примеры диаграмм для логических функций двух, трех и четырех переменных представлены на рисунке 3.13.

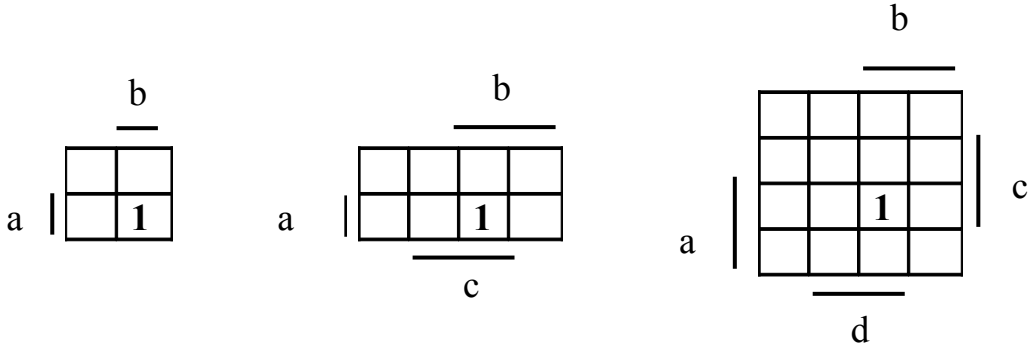


Рисунок 3.13

Диаграмма Карно по каждой переменной делится на две половины. Половина диаграммы, отмеченная соответствующей буквой, содержит все конъюнкции, в которых эта буква записана в прямой форме (без отрицания). На рисунке 3.13 символом 1 отмечены клетки, которые соответствуют следующим конъюнкциям :

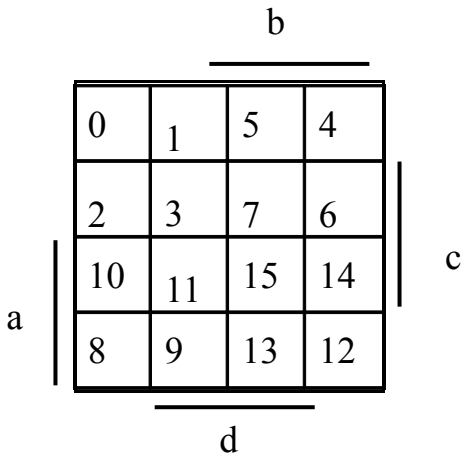
F(ab) = ab; F(abc)=abc; F(abcd) = abcd. (3.20)

В дальнейшем будем рассматривать применение диаграмм Карно на примере функций четырех переменных. Заполнение диаграмм Карно упрощается при использовании эталонных диаграмм. На эталонной диаграмме клетки помечены номерами соответствующих конъюнкций исходной логической функции. Номера конъюнкций соответствуют двоичным комбинациям входных сигналов в таблице истинности логической функции. Эталонная диаграмма для функции четырех переменных, а также пример ее заполнения по таблице истинности приведены на рисунке 3.14.

Таблица истинности

Номера наборов	Значения переменных				Значение функции
	a	b	c	d	
0	0	0	0	0	0
1	0	0	0	1	**1**
2	0	0	1	0	**1**
3	0	0	1	1	**1**
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	**1**
7	0	1	1	1	**1**
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	**1**
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

Эталонная диаграмма



Отмеченная диаграмма

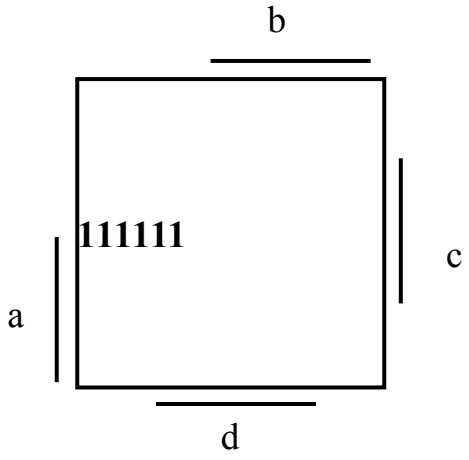
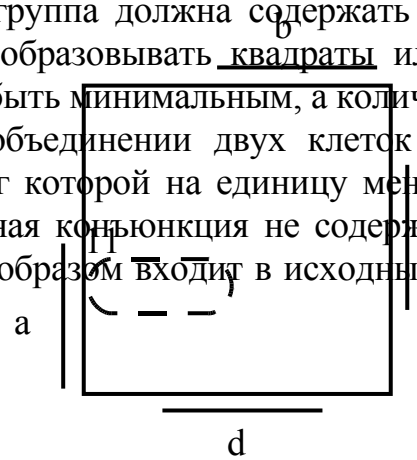


Рисунок 3.14

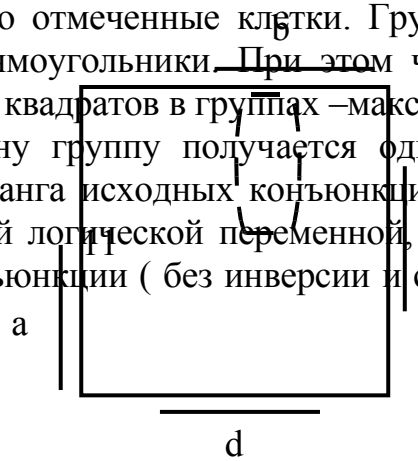
Следует отметить, что вид эталонной диаграммы зависит от способа ее разметки по переменным a, b, c, d.

При использовании метода Карно общая идея заключается в том, что отмеченные клетки диаграммы объединяются в группы по 2, 4, или 8 клеток. Каждая отмеченная клетка должна входить, по крайней мере, в одну группу. Каждая группа должна содержать только отмеченные клетки. Группы клеток должны образовывать квадраты или прямоугольники. При этом число групп должно быть минимальным, а количество квадратов в группах –максимальным .

При объединении двух клеток в одну группу получается одна конъюнкция, ранг которой на единицу меньше ранга исходных конъюнкций. При этом полученная конъюнкция не содержит той логической переменной, которая различным образом входит в исходные конъюнкции (без инверсии и с инверсией).



$F(abcd)=\bar{a}\bar{b}cd \vee \bar{a}\bar{b}cd = \bar{a}\bar{b}c$

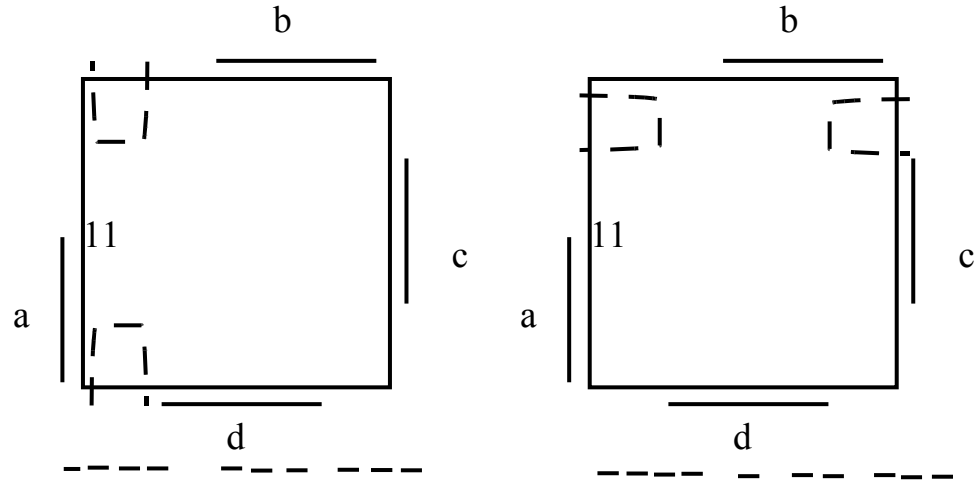


$F(abcd)=\bar{a}\bar{b}cd \vee \bar{a}bcd = \bar{a}cd$

На рисунке 3.15 показаны примеры групп из двух конъюнкций (клеток) и результаты склеивания.

Рисунок 3.15

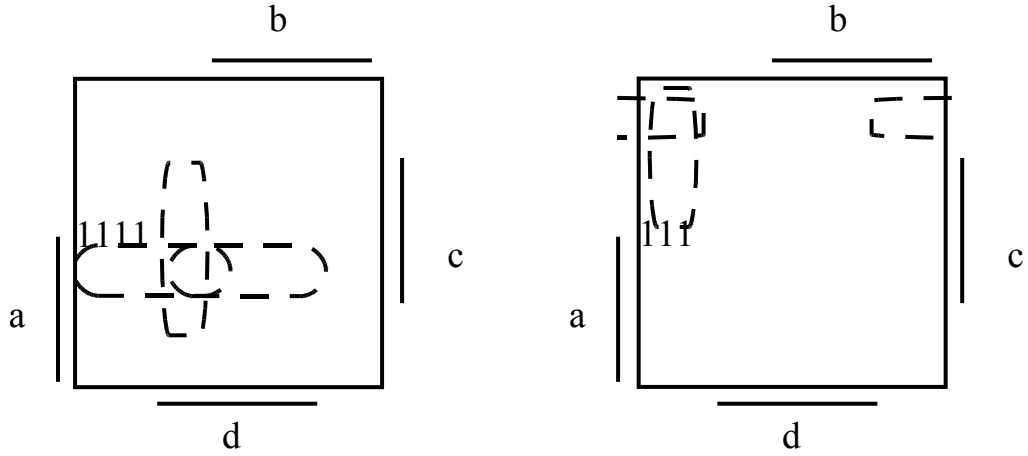
Диаграммы Карно строятся таким образом, что соседними являются крайние в столбце или в строке клетки. Примеры объединения таких клеток в группы приведены на рисунке 3.16.



$F(abcd)=a\,b\,c\,d \vee a\,b\,c\,d = b\,c\,d$ $F(abcd)=a\,b\,c\,d \vee a\,b\,c\,d = a\,c\,d$

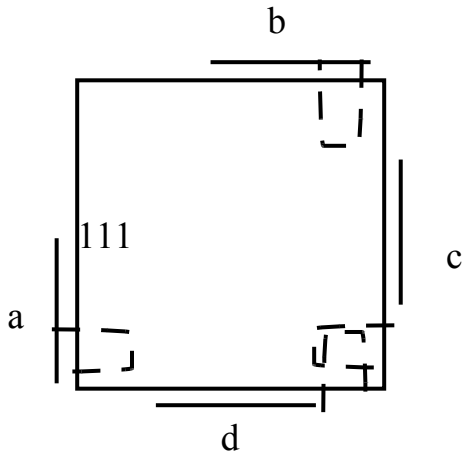
Рисунок 3.16

В соответствии с правилами неполного склеивания и поглощения одна и та же конъюнкция может склеиваться с несколькими другими конъюнкциями. Примеры такого рода приведены на рисунке 3.17.



$F(abcd)=a\,\bar{b}\,c\,\bar{d} \vee a\,\bar{b}\,c\,\bar{d} \vee a\,\bar{b}\,c\,d \vee \vee a\,b\,c\,d = = a\,\bar{b}\,c \vee \bar{b}\,c\,d \vee a\,c\,d$

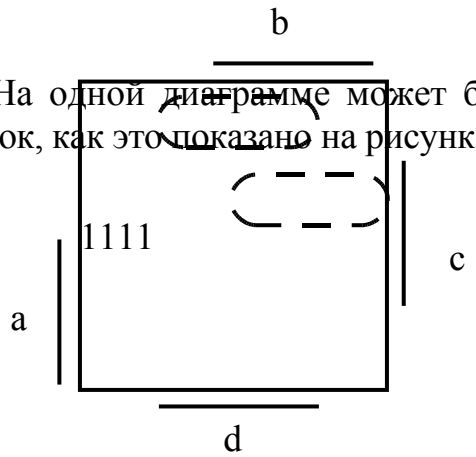
$F(abcd)=\bar{a}\,\bar{b}\,\bar{c}\,\bar{d} \vee \bar{a}\,b\,\bar{c}\,\bar{d} \vee \bar{a}\,\bar{b}\,c\,\bar{d} = = \bar{a}\,\bar{c}\,\bar{d} \vee \bar{a}\,\bar{b}\,\bar{d}$



$F(abcd)=\bar{a}\,b\,\bar{c}\,\bar{d} \vee a\,\bar{b}\,\bar{c}\,\bar{d} \vee a\,b\,\bar{c}\,\bar{d} = a\,\bar{c}\,\bar{d} \vee b\,\bar{c}\,\bar{d}$

Рисунок 3.17

На одной диаграмме может быть несколько групп с одинаковым числом клеток, как это показано на рисунке 3.18.



$F(abcd)=\bar{a}\,\bar{b}\,\bar{c}\,\bar{d} \vee \bar{a}\,\bar{b}\,\bar{c}\,d \vee \bar{a}\,\bar{b}\,c\,\bar{d} \vee \vee \bar{a}\,\bar{b}\,c\,d = = \bar{a}\,\bar{c}\,\bar{d} \vee \bar{a}\,\bar{b}\,c$

$F(abcd)=\bar{a}\,\bar{b}\,\bar{c}\,\bar{d} \vee \bar{a}\,b\,\bar{c}\,\bar{d} \vee \bar{a}\,\bar{b}\,c\,\bar{d} \vee \vee \bar{a}\,\bar{b}\,c\,d \vee \bar{a}\,b\,c\,d \vee a\,\bar{b}\,c\,\bar{d} = \bar{a}\,\bar{c}\,\bar{d} \vee \bar{b}\,c\,\bar{d} \vee \vee \bar{a}\,c\,d$

Рисунок 3.18

При объединении четырех конъюнкций в группу получается конъюнкция, ранг которой на две единицы меньше ранга исходных конъюнкций, при этом полученная конъюнкция не содержит тех переменных, которые по разному входят в исходные конъюнкции. Примеры групп из четырех конъюнкций показаны на рисунке 3.19.

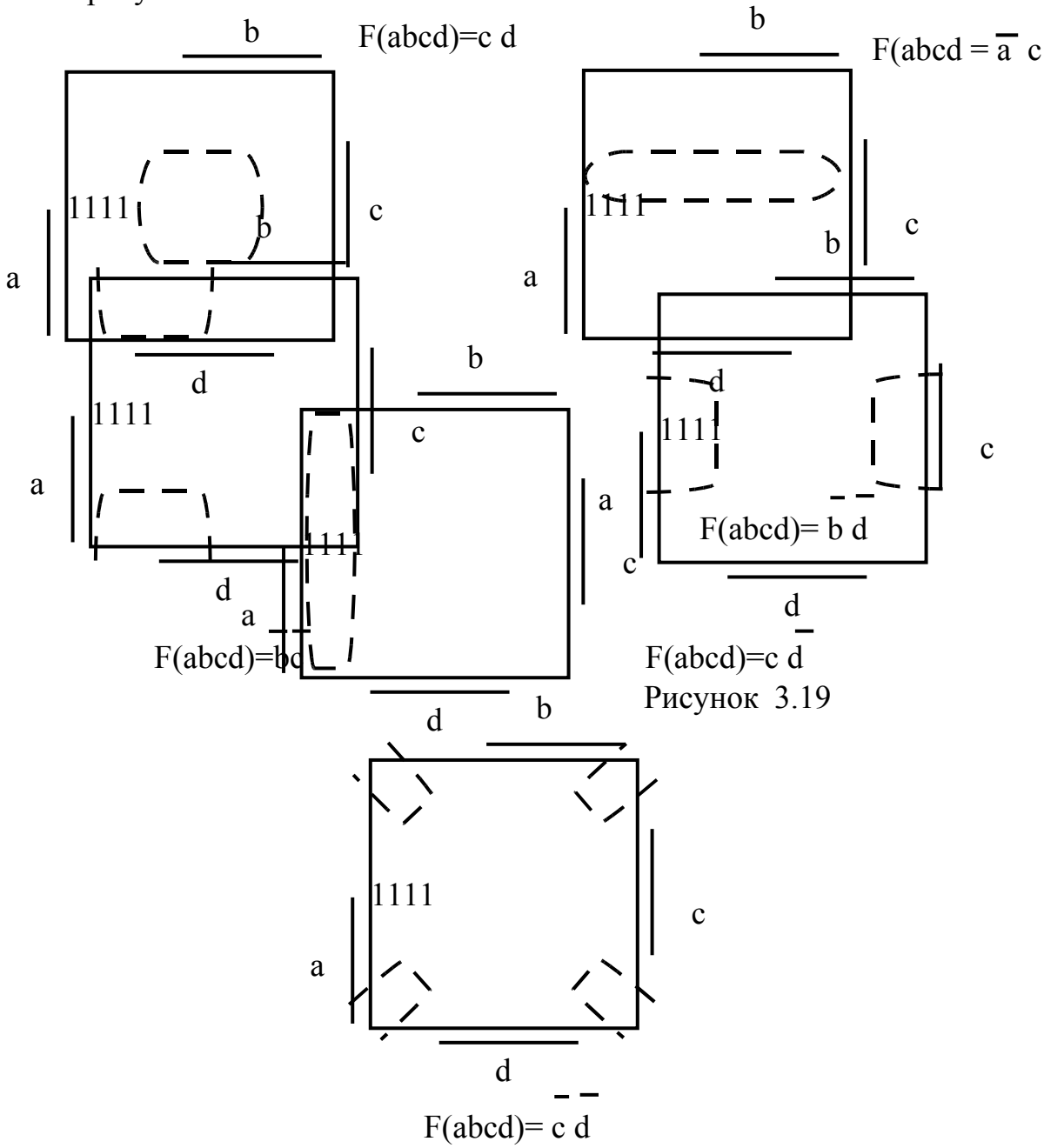
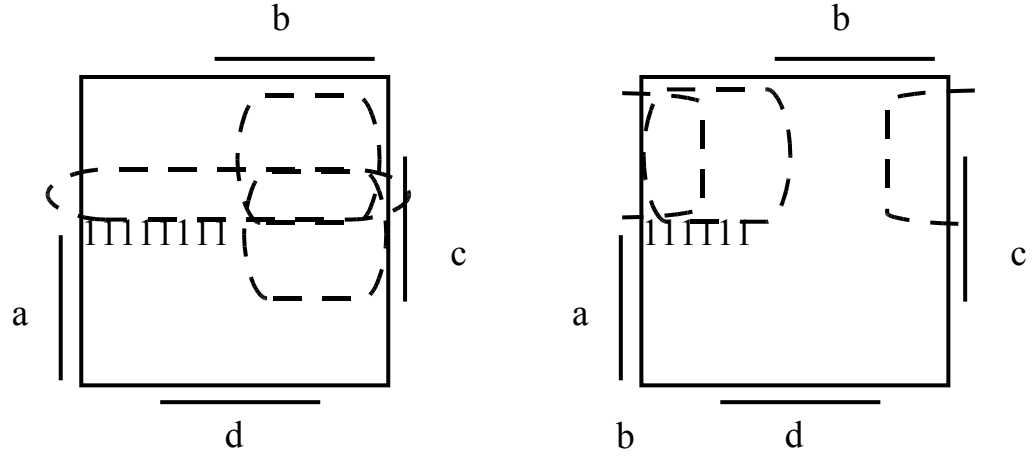


Рисунок 3.19

Рисунок 3.20

Так как соседними являются также верхняя и нижняя строки, а также левый и правый столбцы, это может быть использовано при объединении конъюнкций в группы так, как это показано на рисунке 3.20.

На одной диаграмме может быть несколько групп из четырех конъюнкций, как это показано на рисунке 3.21.



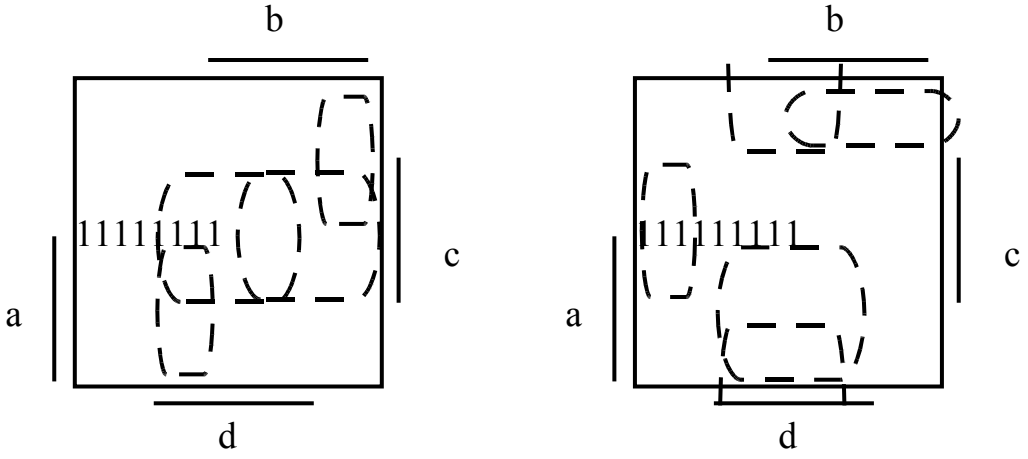
$F(abcd)=\bar{a}c \vee \bar{a}b \vee bc$ $F(abcd)=\bar{a}\,\bar{b} \vee \bar{a}\bar{d}$

Рисунок 3.21

Продолжение рисунка 3.21

$F(abcd)= a\bar{d} \vee b\,\bar{d}$

Следует обратить внимание на то, что в группу не может входить шесть конъюнкций, даже если соответствующие клетки образуют прямоугольник. На рисунке 3.22 показаны примеры минимизации для таких случаев.

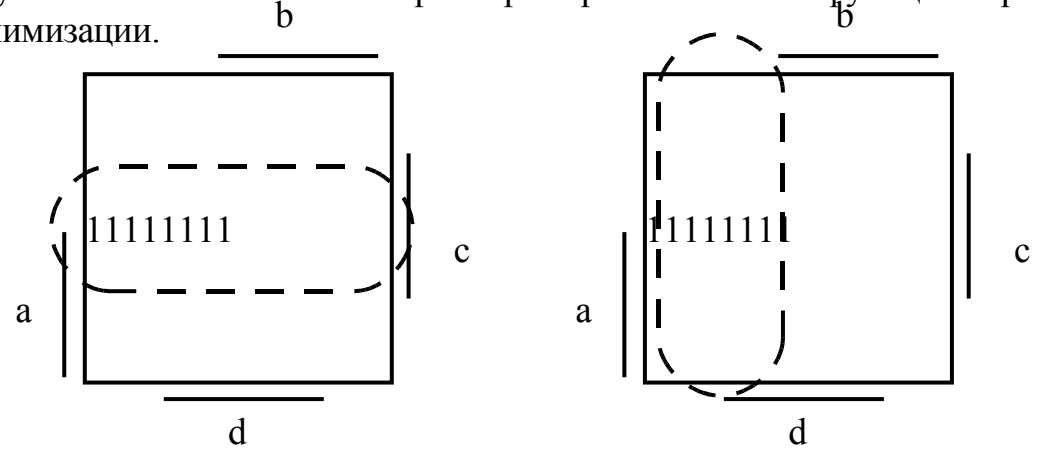


$F(abcd)=cd \vee bc \vee a\bar{b}d \vee \bar{a}b\bar{d}$ $F(abcd)=a\,d \vee \bar{c}d \vee \bar{b}c\bar{d} \vee \bar{a}\bar{b}\bar{c}$

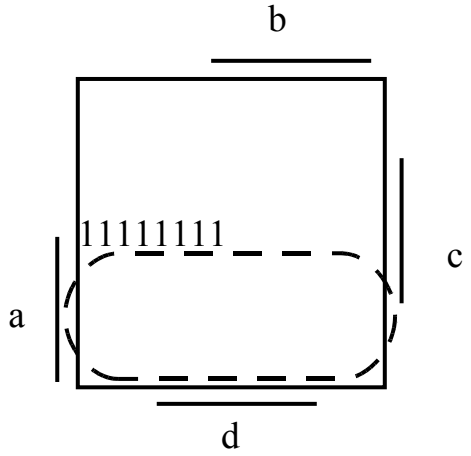
Рисунок 3.22

При объединении восьми конъюнкций в группу результат минимизации представляет собой одну букву (с инверсией или без нее). Некоторые варианты таких групп приведены на рисунке 3.23.

При минимизации на диаграмме могут быть получены различные конфигурации, которые включают группы из различного числа конъюнкций. На рисунок 3.24 показаны некоторые примеры логических функций и результаты их минимизации.

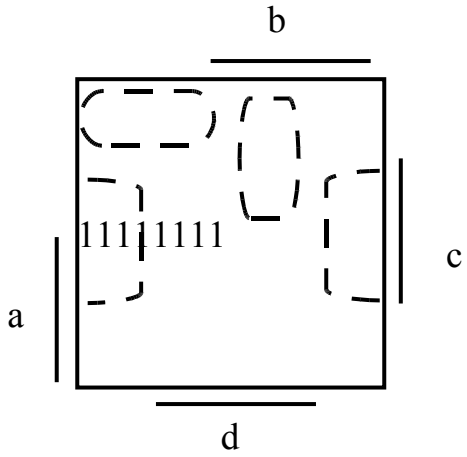


$F(abcd)=c$ $F(abcd)=\bar{b}$



F(abcd) = a

Рисунок 3.23



$F(abcd) = c\bar{d} \vee \bar{a}\bar{b}c \vee \bar{a}bd$

Рисунок 3.24

В заключение следует отметить, что метод Карно практически можно использовать лишь для минимизации логических функций не более чем четырех переменных. Для минимизации более сложных функций используют метод Квайна или его модификации.

Контрольные вопросы

✓1.Что такое импликанта логической функции?
✓2.Что такое сокращенная ДНФ логической функции?
✓3.Что такое тупиковая форма логической функции?
✓4.Что такое минимальная форма логической функции?
✓5.В чем заключается особенность приближенных методов минимизации логических функций?
✓6.Какие логические операции выполняются при минимизации логических функций?
✓7.В чем заключается особенность операции неполного склеивания?
✓8.В каких случаях выполняется операция неполного склеивания?
✓9.В чем заключается особенность диаграммы Карно?
✓10.Как занести логическую функцию на диаграмму Карно?
✓11.Сколько конъюнкций можно объединять в группу при использовании диаграммы Карно?
✓12.Как записать результат минимизации по диаграмме Карно?
✓13.Какие клетки диаграммы Карно являются соседними в смысле склеивания конъюнкций?
✓14.Можно ли по диаграмме Карно найти сокращенную ДНФ логической функции?
✓15.Как зависит ранг результирующей конъюнкции от числа конъюнкций в группе при использовании диаграммы Карно?

3.2. Минимизация логических функций

3.2.1. Методы минимизации логических функций

юбая логическая функция может быть записана в различной форме. Различные формы логической функции могут иметь различную сложность, т.е. для их реализации в виде комбинационной схемы может требоваться различное количество логических элементов. Поэтому перед реализацией логической функции должна быть предпринята попытка минимизировать (упростить) эту функцию. Для минимизации логических функций могут быть использованы различные методы.

По форме представления исходной логической функции различают следующие методы минимизации:

- аналитические методы;
- графические методы.

В аналитических методах минимизации логическая функция представляется обычно в СДНФ. Из этих методов чаще всего используются метод непосредственных преобразований и метод Квайна.

В графических методах логическая функция задается в виде графа или в виде специальной диаграммы. Наиболее известным из графических методов является метод Карно.

По степени приближения результата минимизации к минимальной форме логической функции различают точные и приближенные методы минимизации. Точные методы позволяют получить строго минимальную форму логической функции. К таким методам относятся метод Квайна и метод интервалов. Точные методы требуют много времени для их реализации, поэтому часто применяют приближенные методы. Эти методы не гарантируют получения минимальной формы. Результатом реализации таких алгоритмов является форма функции, близкая к минимальной (может быть и минимальная, но не всегда). Достоинством приближенных методов минимизации является приемлемое время их выполнения. Приближенным методом является, например, метод непосредственных преобразований.

3.2.2. Общая последовательность минимизации

Основной операцией, которая обеспечивает упрощение формы логической функции, является операция склеивания, сущность которой может быть записана следующим образом:

$$ab \vee a\bar{b} = a \qquad (3.7)$$

или в общем виде:

$$aF \vee a\bar{F} = \qquad (3.8)$$

где F - произвольная логическая функция.

Одна и та же конъюнкция может склеиваться с несколькими другими конъюнкциями. В этом случае используется правило неполного склеивания:

$$aF \vee a\overline{F} = a \vee aF \vee a\overline{F} \qquad (3.9)$$

При неполном склеивании результат склеивания дополняется исходными конъюнкциями, которые участвовали в склеивании. Неполное склеивание не приводит к усложнению функции, так как исходные конъюнкции могут быть при необходимости исключены из логической функции путем использования правила поглощения:

$$a \vee aF = a \qquad (3.10)$$

Введем некоторые определения. ***Импликантой*** данной логической функции называется другая логическая функция, которая принимает значение 1 только на части входных комбинаций, на которых данная логическая функция равна 1. Рассмотрим пример (таблица 3.7).

Таблица 3. 7

Переменные			Функции		
a	b	c	F	F1	F2
0	0	0	1	1	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	1	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	0	0	0
1	1	1	1	1	0

В таблице 3.7 заданы значения некоторой функции F. При этом функции F1 и F2 являются импликантами функции F.

Логические функции часто задаются в СДНФ. Для краткости иногда вместо записи СДНФ функция задается перечислением десятичных номеров наборов значений переменных, на которых функция принимает значение «1».

Каждая конъюнкция СДНФ задает один набор значений переменных, на которых логическая функция принимает значение 1 (единичный набор). Поэтому единичный набор является импликантой. Импликантой является также результат склеивания импликант между собой. Импликанты, полученные при склеивании, задают уже не один, а несколько единичных наборов, т.е. интервал единичных значений или единичный интервал. Единичный интервал может включать 1, 2, 4, 8 и т.д. единичных наборов. Таким образом, понятия импликанты и единичного интервала являются синонимами и отличаются лишь по форме. Импликанты записываются в буквенной форме, а единичные интервалы – в цифровой форме (в виде двоичных или десятичных чисел).

Импликанты могут склеиваться друг с другом и порождать новые, более простые, импликанты.

Если импликанта не склеивается с другими импликантами, она называется ***простой импликантой***, а соответствующий ей единичный интервал называется ***максимальным единичным интервалом***.

Каждая логическая функция имеет определенный набор простых импликант. Дизъюнкция всех простых импликант данной логической функции называется ***сокращенной ДНФ*** (сокращенной дизъюнктивной нормальной формой) данной логической функции.

Пример. Пусть $F(abcd) = \overline{a}\overline{b}cd \vee a\overline{b}cd \vee \overline{a}bcd \vee a\overline{b}\overline{c}d \vee ab\overline{c}\overline{d} \vee \overline{a}\overline{b}\overline{c}\overline{d}$
(2, 4, 6, 9, 12, 14)

В данном примере над каждой конъюнкцией указан ее десятичный номер. После склеивания получим сокращенную ДНФ исходной функции:

$F(abcd) = b\overline{d} \vee \overline{a}c\overline{d} \vee \overline{a}\overline{b}cd$

Сокращенная ДНФ данной функции включает три простых импликанты (максимальных единичных интервала). Для записи интервалов в цифровой форме следует записать последовательность символов «0», «1» и «-», длина которой равна числу переменных. Если импликанта содержит переменную в прямой форме, вместо переменной в последовательности записывается символ «1». Вместо переменной в инверсной форме записывается символ «0», а вместо отсутствующих переменных записывается символ «-». Например импликанте bd соответствует интервал ***-1-0***. Так интервалы записываются в двоичной форме.

Для записи десятичной формы интервала следует вместо символов «-» записать все возможные комбинации символов «0» и «1». При этом образуются двоичные номера всех единичных наборов, входящих в данный интервал. Далее записываются десятичные номера наборов, соответствующих нижней и верхней границам интервала так, как это показано ниже.

$b\overline{d}$ → ***-1-0*** → 0***1***0***0*** → (4, 14)
0***1***1***0***
1***1***0***0***
1***1***1***0***

$\overline{a}c\overline{d}$ → ***0-10*** → ***0***0***1***0 → (2, 6)
01***1***0***

$a\overline{b}\overline{c}d$ → ***1001*** → (9, 9)

В данном примере символы «0» и «1» исходных интервалов выделены полужирным курсивом.

В сокращенной ДНФ могут быть лишние импликанты, т.е. такие, при отбрасывании которых значение логической функции не меняется. При отбрасывании лишних импликант из сокращенной ДНФ получается ***тупиковая*** фор-

ма исходной логической функции. Особенность тупиковой формы заключается в том, что эта форма не может быть упрощена.

Логическая функция может иметь несколько тупиковых форм. Различные тупиковые формы одной и той же логической формы могут иметь различную сложность. Для сравнительной оценки сложности логических функций используют такие показатели, как суммарное число букв в логической функции, суммарное число букв плюс число конъюнкций в логической функции, суммарное количество входов элементов, необходимых для реализации логической функции (сложность по Квайну) и др.

При наличии нескольких тупиковых форм логической функции одна из тупиковых форм, имеющая наименьшую сложность, называется ***минимальной*** формой данной логической функции.

В общем случае последовательность минимизации логических функций может иметь вид, показанный на рисунке 3.12.

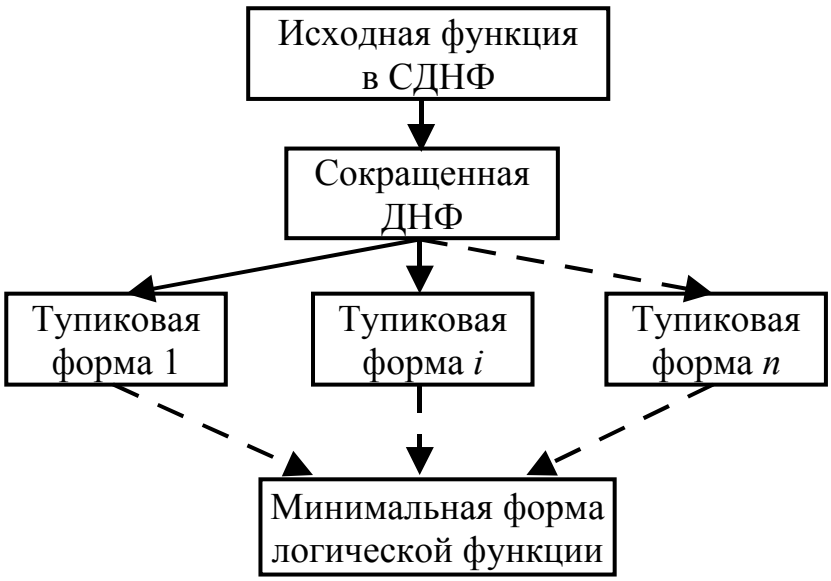


Рисунок 3.12

Минимизируемая логическая функция представляется обычно в совершенной дизъюнктивной нормальной форме (СДНФ). Путем применения логических операций неполного склеивания и поглощения из СДНФ получают сокращенную ДНФ исходной логической функции. После отбрасывания лишних импликант из сокращенной ДНФ получают несколько тупиковых форм логической функции. Далее из тупиковых форм по критерию сложности выбирается минимальная форма исходной логической функции.

Приведенная последовательность реализуется в точных методах минимизации. В приближенных методах минимизации находится лишь одна из тупиковых форм, которая и принимается за минимальную форму логической функции.

Конкретный набор операций и их последовательность при минимизации зависят от используемого метода.

3.2.3. Метод непосредственных преобразований

Метод непосредственных преобразований является аналитическим приближенным методом минимизации. Результат минимизации существенно зависит от квалификации исполнителя. Поэтому метод не является строго алгоритмическим и не гарантирует получения не только минимальной, но даже и тупиковой формы логической функции. Однако этот метод широко используется в инженерной практике для минимизации простых логических функций.

Сущность метода заключается в последовательном применении к исходной логической функции приведенных выше операций склеивания, неполного склеивания и поглощения до тех пор, пока эти операции применимы к исходной форме логической функции или к промежуточным формам исходной функции.

В процессе минимизации могут быть полезными также следующие известные соотношения и правила алгебры логики:

$$x\&x\&x\&x\&x\&\ldots\&x = x;\quad x\vee x\vee x\vee x\vee x\vee\ldots\vee x = x;$$
$$x\&\bar{x} = 0;\quad x\vee\bar{x} = 1;$$
$$x\&1 = x;\quad x\vee 1 = 1;$$
$$x\&0 = 0;\quad x\vee 0 = x; \qquad (3.11)$$
$$\bar{\bar{x}} = x;\quad \overline{\overline{x}} = x$$

В некоторых случаях полезно использовать также правило свертки

$$x \vee \bar{x}y = x \vee y \qquad (3.12)$$

или правило Порецкого

$$xy\vee \bar{x}z \vee yz = xy\vee \bar{x}z \qquad (3.13)$$

Рассмотрим примеры.

Пример. Минимизировать логическую функцию F(abc) методом непосредственных преобразований.

$$F(abc) = \bar{a}\,\bar{b}\,\bar{c} \vee a\,\bar{b}\,\bar{c} \vee a\,b\,\bar{c} \vee abc \qquad (3.14)$$

Так как первая конъюнкция может склеиваться со второй и с третьей, то в результате применения операции склеивания получим:

$$F(abc) = (\bar{a}\,\bar{b}\,\bar{c} \vee a\,\bar{b}\,\bar{c}) \vee (\bar{a}\,\bar{b}\,c \vee a\,\bar{b}\,c) \vee a\,b\,c = \bar{b}\,\bar{c} \vee \bar{a}\,c \vee a\,b\,c \qquad (3.15)$$

В данном примере одна из конъюнкций дважды участвует в склеивании, а последняя конъюнкция не склеивается с другими.

Пример. Минимизировать логическую функцию F(abcd) методом непосредственных преобразований.

$$F(abcd) = a\,b\,c\,d \vee a\,b\,c\,\bar{d} \vee a\,b\,\bar{c}d \vee a\,b\,\bar{c}\,\bar{d} \vee \bar{a}\,b\,c\,d \vee \bar{a}\,b\,\bar{c}\,\bar{d} \qquad (3.16)$$

Последовательно применяя операцию склеивания, получим:

$$F(abcd) = (a\,b\,c\,d \vee a\,b\,c\,\bar{d}) \vee (a\,b\,\bar{c}d \vee a\,b\,\bar{c}\,\bar{d}) \vee (a\,b\,\bar{c}\,d \vee \bar{a}\,b\,c\,d) \vee \vee \bar{a}bcd = (abc \vee \bar{a}bc) \vee bcd \vee \bar{a}bcd = ab \vee bcd \vee \bar{a}bcd \qquad (3.17)$$

В данном примере конъюнкции, полученные в результате склеивания, склеиваются затем между собой. При каждом склеивании число букв в конъюнкциях (ранг конъюнкции) уменьшается на единицу. Поэтому при повторном склеивании вместо конъюнкций из четырех букв получена одна конъюнкция из двух букв.

При большом количестве логических переменных применение метода непосредственных преобразований становится затруднительным, но принципиально возможным. Но при этом трудно гарантировать даже получение хотя бы одной из тупиковых форм.

Приведем пример минимизации логической функции пяти переменных:

$$F(abcdf) = \bar{a}b\bar{c}d\bar{f} \vee abcdf \vee \bar{a}bcdf \vee ab\bar{c}df \vee \bar{a}b\bar{c}d\bar{f} \vee \bar{a}b\bar{c}df \vee abcd\bar{f} \vee \vee \bar{a}bcdf \vee ab\bar{c}df.$$

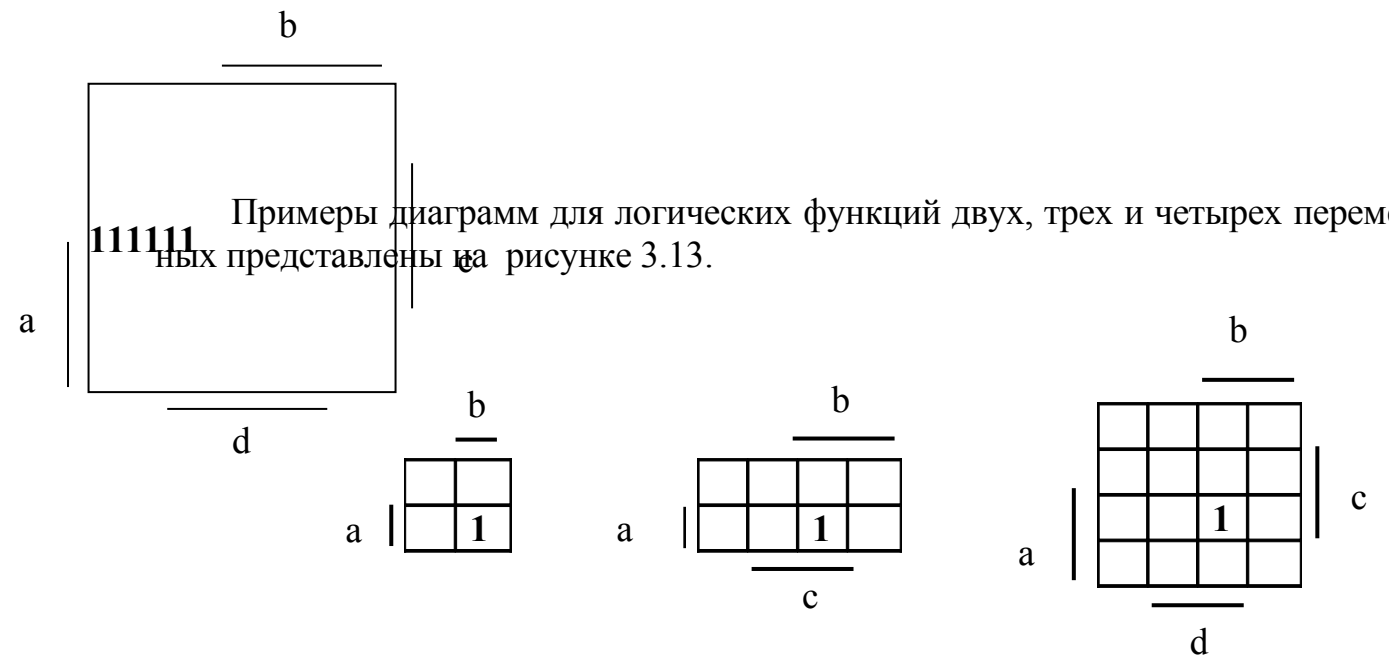
$$F(abcdf) = (\bar{a}b\bar{c}d\bar{f} \vee \bar{a}b\bar{c}df) \vee (\bar{a}bcd\bar{f} \vee \bar{a}bcdf) \vee (ab\bar{c}d\bar{f} \vee ab\bar{c}df) \vee \vee (abcd\bar{f} \vee abcdf) \vee (\bar{a}b\bar{c}df \vee \bar{a}bcdf) = (\bar{a}b\bar{c}d \vee \bar{a}bcd) \vee (a\bar{b}cd \vee abcd) \vee \vee \bar{a}\,b\,c\,f = (\bar{a}bd \vee abd) \vee \bar{a}bcf = bd \vee \bar{a}bcf \qquad (3.19)$$

Из данного примера следует, что даже для такой сравнительно несложной функции поиск конъюнкций, которые могут быть склеены, представляет значительные трудности. Так как при минимизации выполняется большое количество сравнительно несложных операций, для упрощения реальных логических функций, которые встречаются в инженерной практике, целесообразно использовать машинные программы. Для минимизации же логических функций с числом переменных не более четырех широкое применение нашли графические методы, в частности метод Карно.

3.2.4. Метод Карно

Метод Карно представляет собой приближенный метод минимизации логических функций, который при определенных навыках позволяет получить минимальную форму исходной функции. Исходная логическая функция представляется в СДНФ. Минимизированная логическая функция представлена в дизъюнктивной нормальной форме.

Сущность метода Карно заключается в построении специальных диаграмм, представляющих собой прямоугольные таблицы. Каждой конъюнкции исходной логической функции соответствует одна из клеток таблицы. Главная особенность диаграмм Карно заключается в том, что конъюнкциям, которые могут быть склеены, соответствуют соседние по строке или по столбцу клетки.



Примеры диаграмм для логических функций двух, трех и четырех переменных представлены на рисунке 3.13.

Рисунок 3.13

Диаграмма Карно по каждой переменной делится на две половины. Половина диаграммы, отмеченная соответствующей буквой, содержит все конъюнкции, в которых эта буква записана в прямой форме (без отрицания). На рисунке 3.13 символом 1 отмечены клетки, которые соответствуют следующим конъюнкциям :

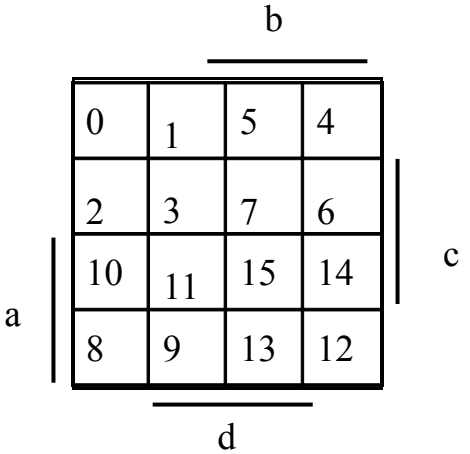
F(ab) = ab; F(abc)=abc; F(abcd) = abcd. (3.20)

В дальнейшем будем рассматривать применение диаграмм Карно на примере функций четырех переменных. Заполнение диаграмм Карно упрощается при использовании эталонных диаграмм. На эталонной диаграмме клетки помечены номерами соответствующих конъюнкций исходной логической функции. Номера конъюнкций соответствуют двоичным комбинациям входных сигналов в таблице истинности логической функции. Эталонная диаграмма для функции четырех переменных, а также пример ее заполнения по таблице истинности приведены на рисунке 3.14.

Номера наборов	Значения переменных				Значение функции
	a	b	c	d	
0	0	0	0	0	0
1	0	0	0	1	**1**
2	0	0	1	0	**1**
3	0	0	1	1	**1**
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	**1**
7	0	1	1	1	**1**
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	**1**
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

Таблица истинности

Эталонная диаграмма



Отмеченная диаграмма

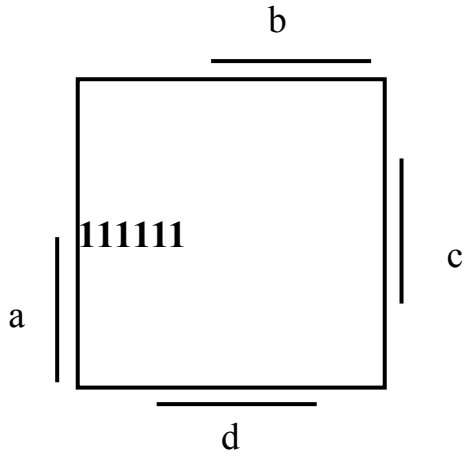
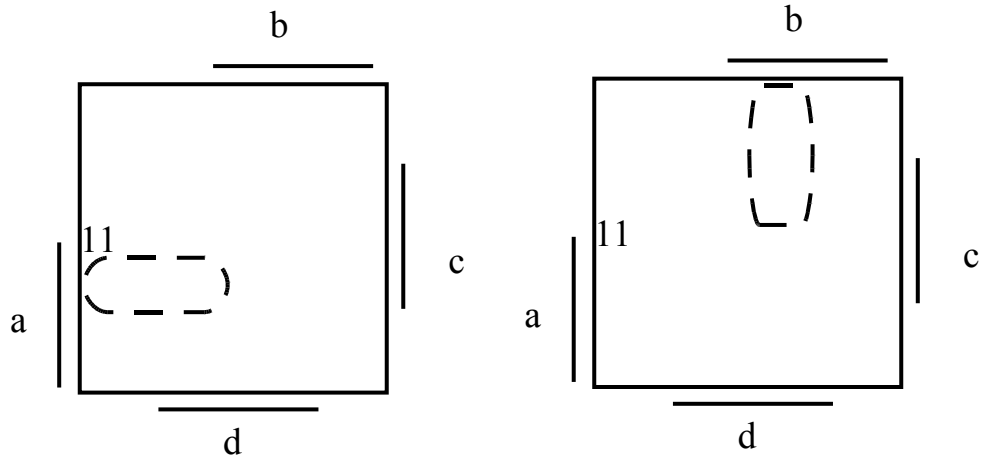


Рисунок 3.14

Следует отметить, что вид эталонной диаграммы зависит от способа ее разметки по переменным a, b, c, d.

При использовании метода Карно общая идея заключается в том, что отмеченные клетки диаграммы объединяются в группы по 2, 4, или 8 клеток. Каждая отмеченная клетка должна входить, по крайней мере, в одну группу. Каждая группа должна содержать только отмеченные клетки. Группы клеток должны образовывать квадраты или прямоугольники. При этом число групп должно быть минимальным, а количество квадратов в группах –максимальным .

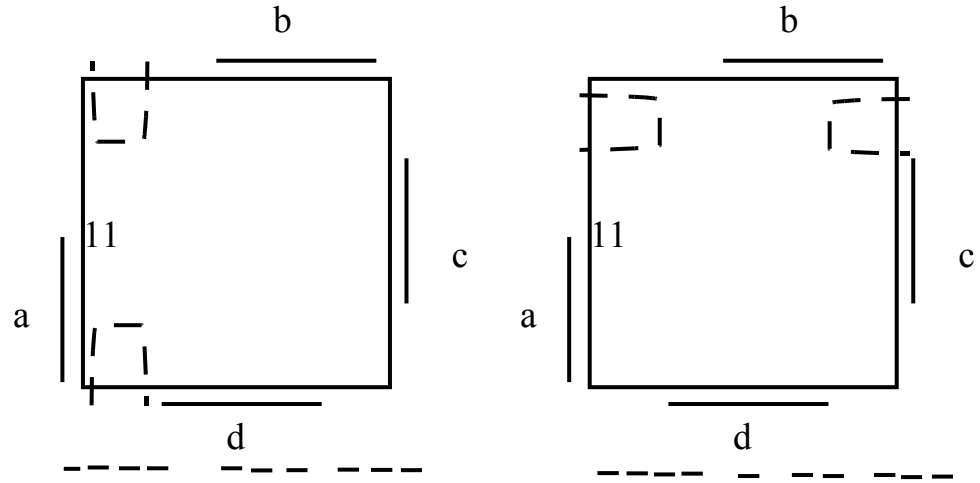
При объединении двух клеток в одну группу получается одна конъюнкция, ранг которой на единицу меньше ранга исходных конъюнкций. При этом полученная конъюнкция не содержит той логической переменной, которая различным образом входит в исходные конъюнкции (без инверсии и с инверсией). На рисунке 3.15 показаны примеры групп из двух конъюнкций (клеток) и результаты склеивания.



$F(abcd)=a\bar{b}c\bar{d} \vee a\bar{b}cd = a\bar{b}c$ $F(abcd)=\bar{a}\bar{b}c\bar{d} \vee \bar{a}bcd = \bar{a}bd$

Рисунок 3.15

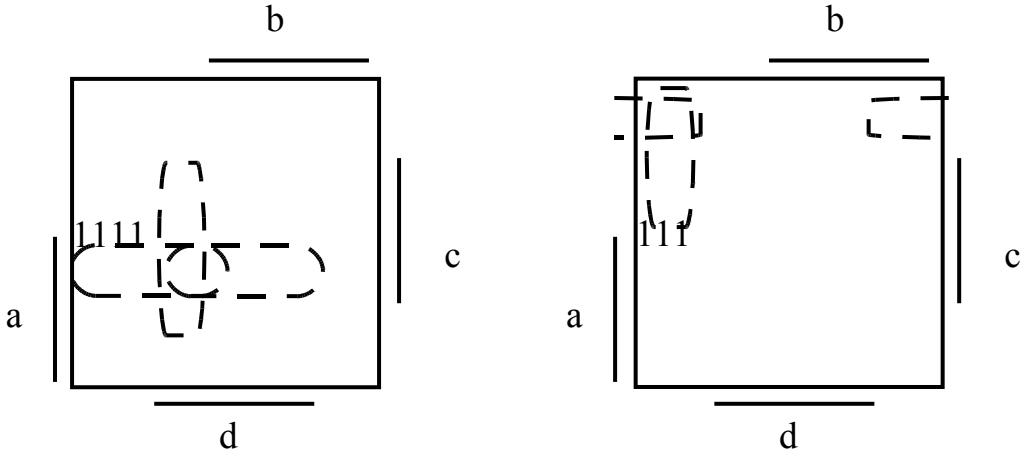
Диаграммы Карно строятся таким образом, что соседними являются крайние в столбце или в строке клетки. Примеры объединения таких клеток в группы приведены на рисунке 3.16.



$F(abcd)=a\,b\,c\,d \vee a\,b\,c\,d = b\,c\,d$ $F(abcd)=a\,b\,c\,d \vee a\,b\,c\,d = a\,c\,d$

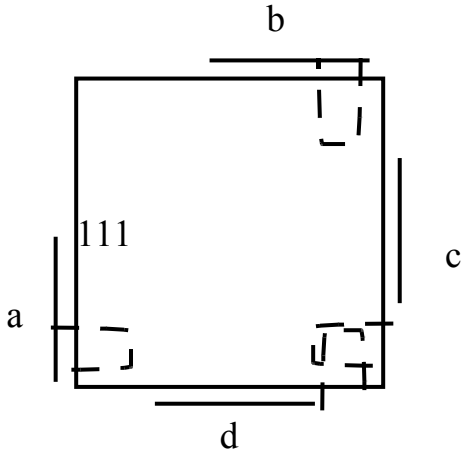
Рисунок 3.16

В соответствии с правилами неполного склеивания и поглощения одна и та же конъюнкция может склеиваться с несколькими другими конъюнкциями. Примеры такого рода приведены на рисунке 3.17.



$F(abcd)=a\,\bar{b}\,c\,\bar{d} \vee a\,\bar{b}\,c\,d \vee a\,b\,c\,d \vee$
$\vee\, a\,b\,c\,d = a\,\bar{b}\,c \vee \bar{b}\,c\,d \vee a\,c\,d$

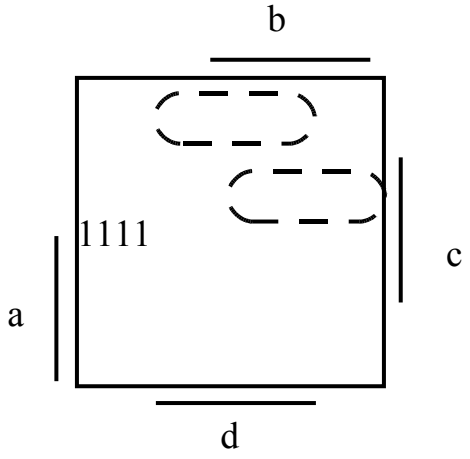
$F(abcd)=\bar{a}\,\bar{b}\,\bar{c}\,\bar{d} \vee \bar{a}\,b\,\bar{c}\,\bar{d} \vee \bar{a}\,\bar{b}\,c\,\bar{d} =$
$= \bar{a}\,\bar{c}\,\bar{d} \vee \bar{a}\,\bar{b}\,\bar{d}$



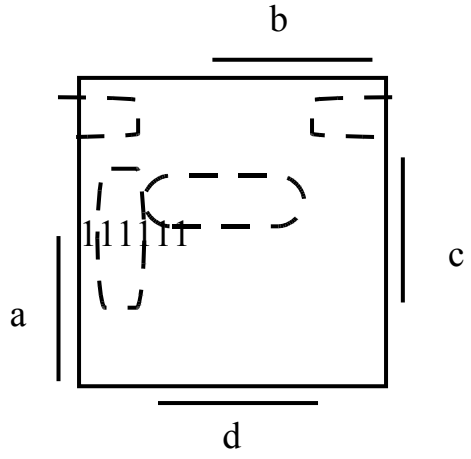
$$F(abcd)=\bar{a}\,b\,\bar{c}\,\bar{d}\vee a\,\bar{b}\,\bar{c}\,\bar{d}\vee\ a\,b\,\bar{c}\,\bar{d}=a\,\bar{c}\,\bar{d}\vee b\,\bar{c}\,\bar{d}$$

Рисунок 3.17

На одной диаграмме может быть несколько групп с одинаковым числом клеток, как это показано на рисунке 3.18.



$$F(abcd)=\bar{a}\,\bar{b}\,\bar{c}\,d\vee\bar{a}\,b\,\bar{c}\,d\vee\bar{a}\,b\,c\,d\vee\vee\bar{a}\,b\,c\,\bar{d}==\bar{a}\,\bar{c}\,d\vee\bar{a}\,b\,c$$

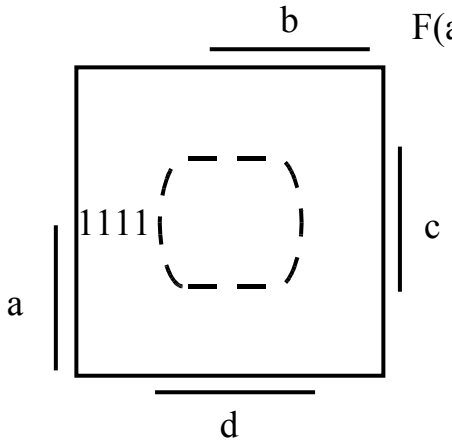


$$F(abcd)=\bar{a}\,\bar{b}\,\bar{c}\,\bar{d}\vee\bar{a}\,b\,\bar{c}\,\bar{d}\vee\bar{a}\,\bar{b}\,c\,\bar{d}\vee\vee\bar{a}\,\bar{b}\,c\,d\vee\ \bar{a}\,b\,c\,d\vee a\,\bar{b}\,c\,\bar{d}=\bar{a}\,\bar{c}\,\bar{d}\vee\ \bar{b}\,c\,\bar{d}\vee\vee\bar{a}\,c\,d$$

Рисунок 3.18

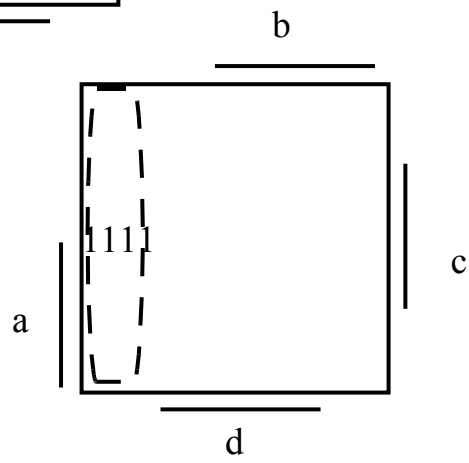
При объединении четырех конъюнкций в группу получается конъюнкция, ранг которой на две единицы меньше ранга исходных конъюнкций, при этом полученная конъюнкция не содержит тех переменных, которые по разному

входят в исходные конъюнкции. Примеры групп из четырех конъюнкций показаны на рисунке 3.19.



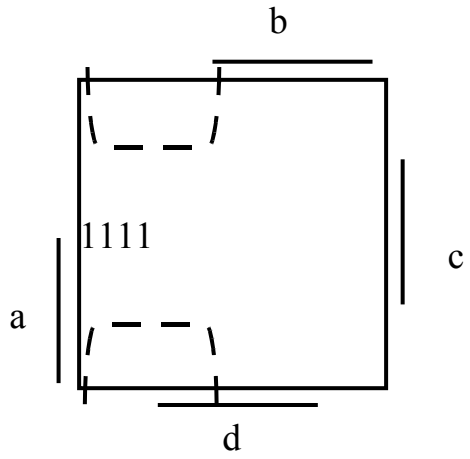
F(abcd)=c d

F(abcd = $\overline{a}$ c

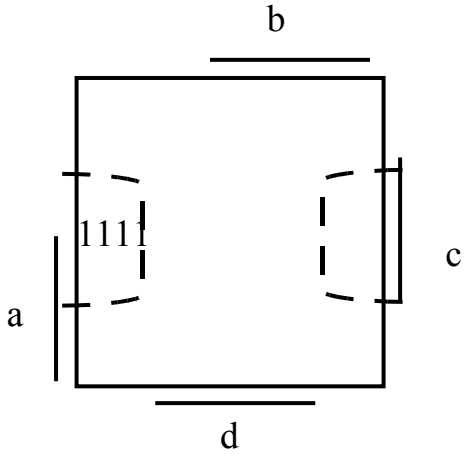


F(abcd)= $\overline{b}$ $\overline{d}$

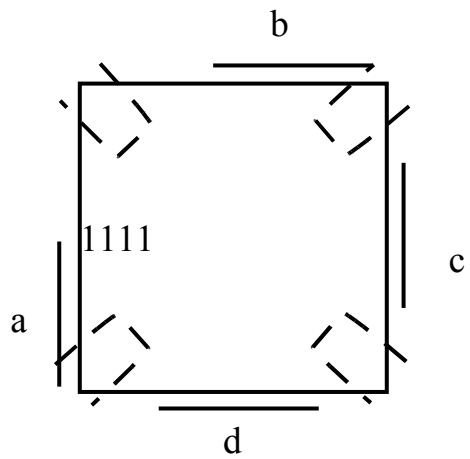
Рисунок 3.19



F(abcd)=$\overline{bc}$



F(abcd)=c $\overline{d}$

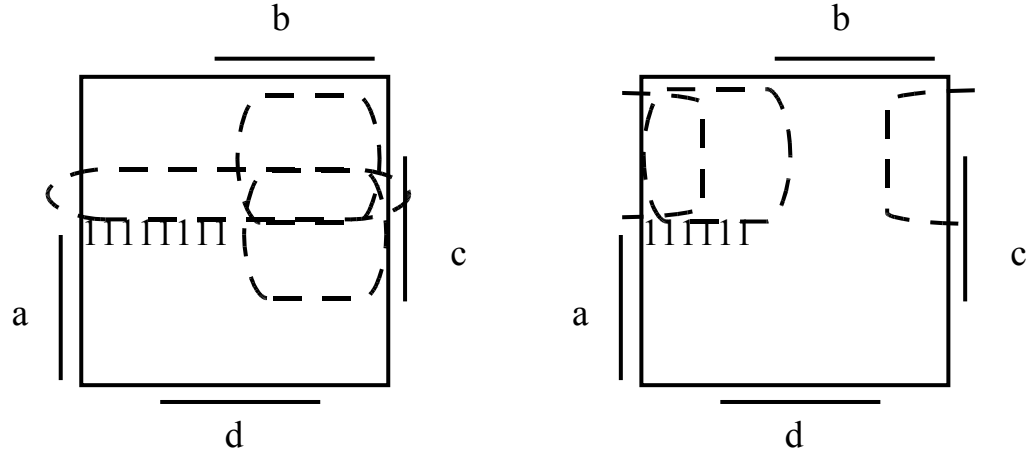


F(abcd)= $\overline{c}$ $\overline{d}$

Рисунок 3.20

Так как соседними являются также верхняя и нижняя строки, а также левый и правый столбцы, это может быть использовано при объединении конъюнкций в группы так, как это показано на рисунке 3.20.

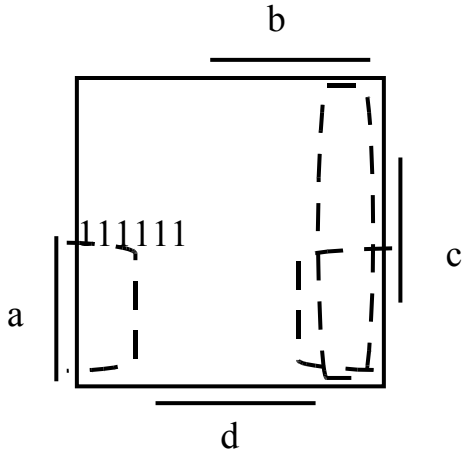
На одной диаграмме может быть несколько групп из четырех конъюнкций, как это показано на рисунке 3.21.



$F(abcd)=\bar{a}c \vee \bar{a}b \vee bc$ $F(abcd)=\bar{a}\,\bar{b} \vee \bar{a}d$

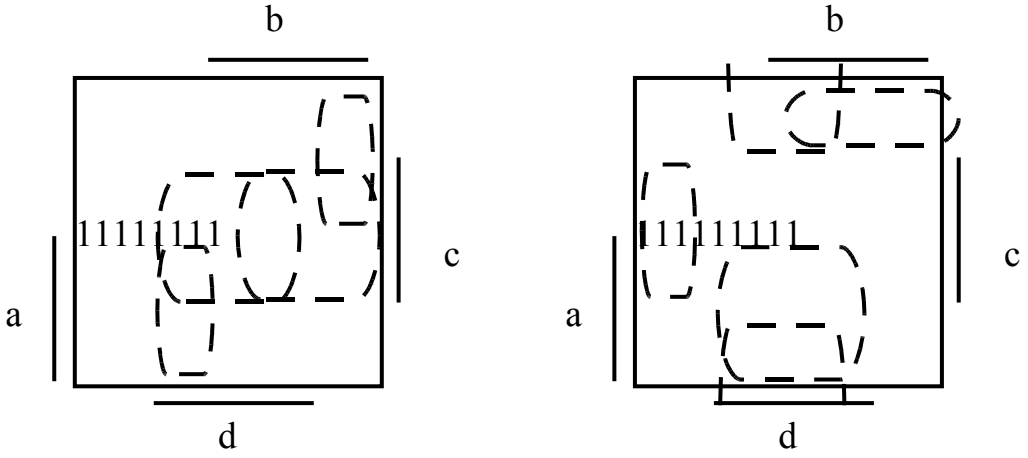
Рисунок 3.21

Продолжение рисунка 3.21



$F(abcd)= a\bar{d} \vee b\,\bar{d}$

Следует обратить внимание на то, что в группу не может входить шесть конъюнкций, даже если соответствующие клетки образуют прямоугольник. На рисунке 3.22 показаны примеры минимизации для таких случаев.

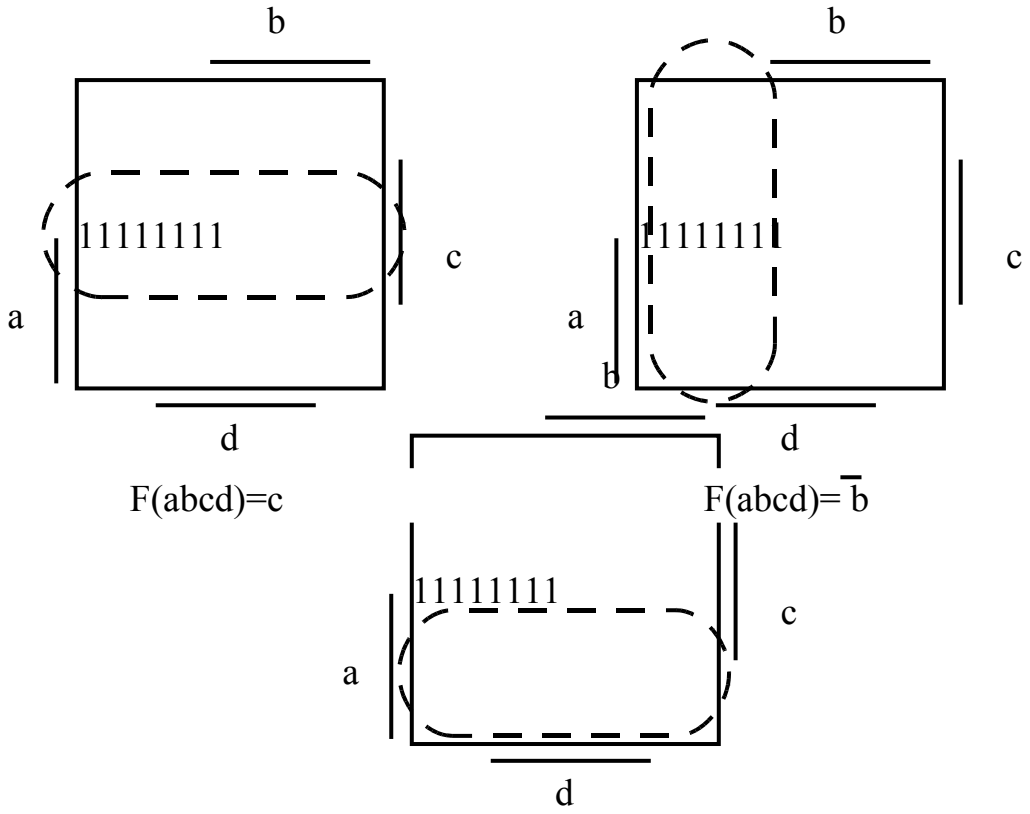


$F(abcd)=cd \vee bc \vee a\bar{b}d \vee \bar{a}b\bar{d}$ $F(abcd)=a\,d \vee \bar{c}d \vee \bar{b}c\bar{d} \vee \bar{a}b\bar{c}$

Рисунок 3.22

При объединении восьми конъюнкций в группу результат минимизации представляет собой одну букву (с инверсией или без нее). Некоторые варианты таких групп приведены на рисунке 3.23.

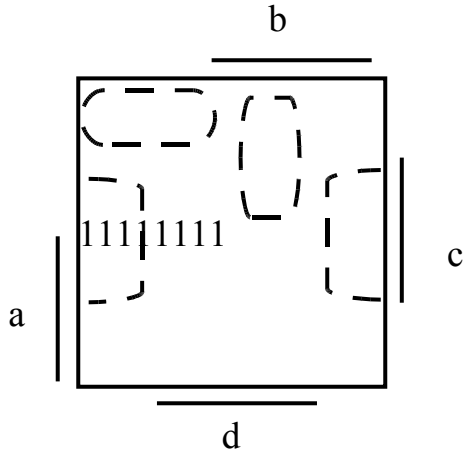
При минимизации на диаграмме могут быть получены различные конфигурации, которые включают группы из различного числа конъюнкций. На рисунок 3.24 показаны некоторые примеры логических функций и результаты их минимизации.



F(abcd)=c F(abcd)= $\bar{b}$

F(abcd) = a

Рисунок 3.23



$$F(abcd) = \bar{c}\bar{d} \vee \bar{a}\bar{b}\bar{c} \vee \bar{a}bd$$

Рисунок 3.24

В заключение следует отметить, что метод Карно практически можно использовать лишь для минимизации логических функций не более чем четырех переменных. Для минимизации более сложных функций используют метод Квайна или его модификации.

Контрольные вопросы

✓1. Что такое импликанта логической функции?
✓2. Что такое сокращенная ДНФ логической функции?
✓3. Что такое тупиковая форма логической функции?
✓4. Что такое минимальная форма логической функции?
✓5. В чем заключается особенность приближенных методов минимизации логических функций?
✓6. Какие логические операции выполняются при минимизации логических функций?

✓7.В чем заключается особенность операции неполного склеивания?
✓8.В каких случаях выполняется операция неполного склеивания?
✓9.В чем заключается особенность диаграммы Карно?
✓10.Как занести логическую функцию на диаграмму Карно?
✓11.Сколько конъюнкций можно объединять в группу при использовании диа-
граммы Карно?
✓12.Как записать результат минимизации по диаграмме Карно?
✓13.Какие клетки диаграммы Карно являются соседними в смысле склеивания конъюнкций?
✓14.Можно ли по диаграмме Карно найти сокращенную ДНФ логической функции?
✓15.Как зависит ранг результирующей конъюнкции от числа конъюнкций в группе при использовании диаграммы Карно?