



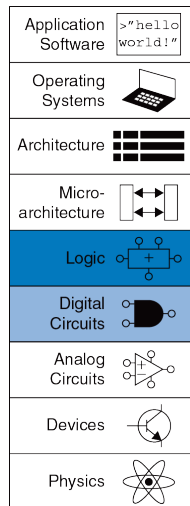
Проектирование комбинационной логики

2

- 2.1 Введение
- 2.2 Булевы уравнения
- 2.3 Булева алгебра
- 2.4 От логики к логическим элементам
- 2.5 Многоуровневая комбинационная логика
- 2.6 Что за х и z?
- 2.7 Карты карно
- 2.8 Базовые комбинационные блоки
- 2.9 Временные характеристики
- 2.10 Резюме

Упражнения

Вопросы для собеседования



2.1 ВВЕДЕНИЕ

В цифровой электронике под *схемой* понимают электрическую цепь, которая обрабатывает дискретные сигналы. Такую схему можно рассматривать как «черный ящик», как показано на [Рис. 2.1](#), при этом схема имеет:

- ▶ Один или более дискретных входов;
- ▶ Один или более дискретных выходов;
- ▶ Функциональную спецификацию (*functional specification*), описывающую взаимосвязь между входами и выходами;
- ▶ Временную спецификацию (*timing specification*), описывающую задержку между изменением сигналов на входе и откликом выходного сигнала.

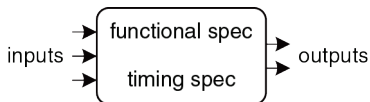


Рис. 2.1 Схема как «черный ящик» с входами, выходами и спецификациями

Если заглянуть внутрь такого «черного ящика», мы увидим, что схемы состоят из соединений, также называемых узлами (*nodes*), и элементов.

Элемент также представляет собой схему с входами, выходами и спецификацией. Соединение – это проводник, напряжение на котором соответствует дискретной переменной. Соединения подразделяются на входы, выходы и внутренние соединения. Входы получают сигналы извне. Выходы отправляют сигналы во внешний мир. Соединения, которые не являются входами или выходами, называются внутренними соединениями. На **Рис. 2.2** показана электронная схема с тремя элементами E1, E2 и E3 и шестью соединениями. Соединения A, B и C – входы, Y и Z – выходы, а n1 – внутреннее соединение между E1 и E3.

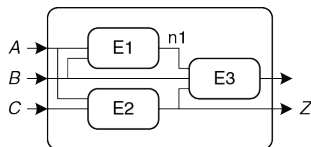


Рис. 2.2 Элементы и соединения

Цифровые схемы разделяются на комбинационные (combinational) и последовательностные (sequential). Выходы комбинационных схем зависят только от текущих значений на входах; другими словами, такие схемы комбинируют текущие значения входных сигналов для вычисления значения на выходе. Например, логический элемент – это

комбинационная схема. Выходы последовательностных схем зависят и от текущих, и от предыдущих значений на входах, то есть зависят от последовательности изменения входных сигналов. У комбинационных схем, в отличие от последовательностных схем, память отсутствует. Данная глава посвящена комбинационным схемам, а в [главе 3](#) мы рассмотрим последовательностные схемы.

Функциональная спецификация комбинационной схемы описывает зависимость значений на выходах от текущих входных значений. Временная спецификация комбинационной схемы состоит из нижней и верхней граничных значений задержки сигнала по пути от входа к выходу. В этой главе мы сначала рассмотрим функциональную спецификацию, а потом вернемся к временной.

На [Рис. 2.3](#) показана комбинационная схема с двумя входами и одним выходом. Входы A и B расположены слева, справа изображен выход Y . Символ \boxplus в прямоугольнике означает, что этот элемент реализован с использованием исключительно комбинационной логики. В этом примере функция F определена как «ИЛИ»: $Y = F(A, B) = A + B$.

Другими словами, мы говорим, что выход Y – это функция двух входов A и B , а именно $Y = A$ ИЛИ B . На [Рис. 2.4](#) показаны два возможных способа построения комбинационной логической схемы, приведенной на [Рис. 2.3](#). Как мы неоднократно увидим в этой книге, зачастую

существует множество способов реализации одной и той же функции. Вы сами выбираете, как реализовать требуемую функцию, исходя из имеющихся в распоряжении «строительных блоков», а также ваших проектных ограничений. Эти ограничения часто включают в себя занимаемую на кристалле микросхемы площадь, скорость работы, потребляемую мощность и время разработки.



$$Y = F(A, B) = A + B$$

Рис. 2.3 Комбинационная логическая схем

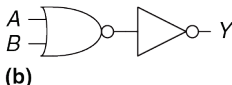
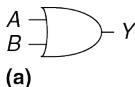
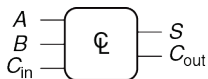


Рис. 2.4 Два варианта схемы ИЛИ

На **Рис. 2.5** показана комбинационная схема с несколькими выходами. Данная комбинационная схема называется полным сумматором, мы ещё вернёмся к ней в **разделе 5.2.1**. Два уравнения определяют значения на выходах S и C_{out} как функции входных сигналов A , B и C_{in} .

Для упрощения чертежей мы часто используем перечеркнутую кривой чертой линию и число рядом с ней для обозначения *шины* (bus), то есть группы сигналов. Число показывает, сколько сигналов в шине (прим. переводчика: это число обычно называется *шириной шины*). Например, на **Рис. 2.6 (а)** показан блок комбинационной логики с тремя входами и двумя выходами. Если количество разрядов не имеет значения или очевидно из контекста, то косая черта может быть без числа рядом.



$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + AC_{in} + BC_{in}$$

Рис. 2.5 Многовыходная комбинационная схема



(a)



(b)

Рис. 2.6 Обозначение шин на схемах

На **Рис. 2.3 (b)** показаны два блока комбинационной логики с произвольным числом выходов одного блока, которые являются входами для другого блока.

Правила *комбинационной композиции* говорят нам, как мы можем построить большую комбинационную схему из более маленьких

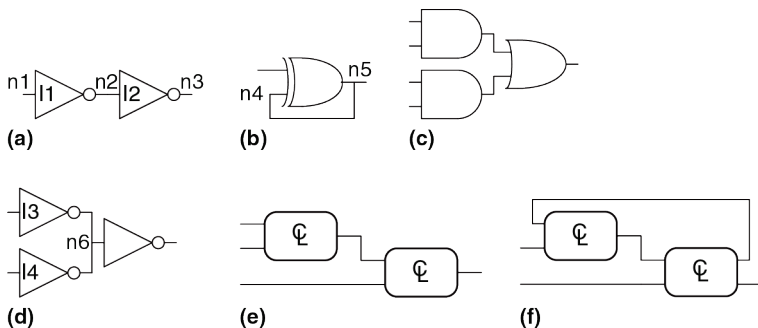
комбинационных элементов. Схема является комбинационной, если она состоит из соединенных между собой элементов и выполнены следующие условия:

- ▶ Каждый элемент схемы сам является комбинационным;
- ▶ Каждое соединение схемы является или входом, или подсоединено к одному-единственному выходу другого элемента схемы;
- ▶ Схема не содержит циклических путей: каждый путь в схеме проходит через любое соединение не более одного раза.

Правила комбинационной композиции схем являются достаточными, но не строго необходимыми. Некоторые схемы, не подчиняющиеся этим правилам, все же являются комбинационными, поскольку значения их выходов зависят только от текущих значений на входах. Однако бывает довольно сложно определить, являются ли некоторые нетипичные схемы комбинационными или нет, поэтому обычно при разработке комбинационных схем мы ограничиваем себя правилами комбинационной композиции.

Пример 2.1 КОМБИНАЦИОННЫЕ СХЕМЫ

Какие из схем на **Рис. 2.7** являются, согласно правилам комбинационной композиции, комбинационными?

**Рис. 2.7** Примеры схем

Решение: Схема (а) – комбинационная. Она составлена из двух комбинационных элементов (инверторы I1 и I2). В ней три соединения: n1, n2 и n3. Соединение n1 – вход схемы и вход для I1; n2 – внутреннее соединение, являющееся выходом для I1 и входом для I2; n3 – выход схемы и выход I2. Схема (b) – это не комбинационная схема, поскольку в ней есть циклический путь: выход элемента «исключающее ИЛИ» подключен к одному из его собственных входов, то есть циклический путь, начинаясь в n4, проходит через «исключающее ИЛИ» к n5, который ведет обратно к n4. Схема (c) – комбинационная, а (d) – не комбинационная, поскольку соединение n6 подключено к выходам двух элементов (I3 и I4). Схема (e) – комбинационная, представляющая собой две комбинационные схемы, соединенные между собой

и образующие более крупную комбинационную схему. Схема (f) не отвечает правилам комбинационной композиции, поскольку в ней есть циклический путь через два элемента. В зависимости от функций этих элементов эта схема может быть, а может и не быть комбинационной.

Большие схемы, такие как микропроцессоры, могут быть очень сложными, поэтому мы будем применять принципы, описанные в **главе 1**, чтобы справиться со сложностью. Рассмотрение схемы как «черного ящика» с тщательно определенными интерфейсом и функцией есть применение принципов абстракции и модульности. Построение схемы из более мелких элементов является применением иерархического подхода к разработке. Правила комбинационной композиции суть применение дисциплины.

Функциональная спецификация комбинационной схемы обычно задается в виде таблицы истинности или булева уравнения. В следующих разделах будет описано, как вывести булево уравнение из любой таблицы истинности и как применять булеву алгебру и карты Карно для упрощения уравнений. Мы рассмотрим, как реализовывать эти уравнения, используя логические элементы, и как анализировать скорость работы таких схем.

2.2 БУЛЕВЫ УРАВНЕНИЯ

Булевы уравнения используют переменные, имеющие значение ИСТИНА или ЛОЖЬ, поэтому они идеально подходят для описания цифровой логики. В этом разделе сначала будет приведена терминология, часто используемая в булевых уравнениях, а затем будет показано, как записать булевы уравнения для любой логической функции по её таблице истинности.

2.2.1 Терминология

Дополнение (complement) переменной A – это ее отрицание \bar{A} . Переменная или ее дополнение называются *литералом*. Например, A , \bar{A} , B и \bar{B} – литералы. Мы будем называть A прямой формой переменной, а \bar{A} – комплементарной формой; «прямая форма» не подразумевает, что значение A равно ИСТИНЕ, а говорит лишь о том, что у A нет черты сверху.

Операция «И» над одним или несколькими литералами называется *конъюнкцией*, *произведением* (product) или *импликантой*. $\bar{A}B$, $\bar{A}\bar{B}\bar{C}$ и B являются импликантами для функции трех переменных. *Минтерм* (minterm, элементарная конъюнктивная форма) – это произведение, включающее все входы функции. $\bar{A}\bar{B}\bar{C}$ – это минтерм для функции трех переменных A , B и C , а $\bar{A}B$ – не минтерм, поскольку он не включает в

себя C . Аналогично, операция «ИЛИ» над одним или более литералами называется *дизъюнкцией* или *суммой*. *Макстерм* (maxterm, элементарная дизъюнктивная форма) – это сумма всех входов функции. $A + B + C$ является макстермом функции трех переменных A , B и C .

Порядок операций важен при анализе булевых уравнений. Означает ли $Y = A + BC$, что $Y = (A \text{ ИЛИ } B) \text{ И } C$ или $Y = A \text{ ИЛИ } (B \text{ И } C)$? В булевых уравнениях наибольший приоритет имеет операция НЕ, затем идет И, затем ИЛИ. Как и в обычных уравнениях, произведения вычисляются до вычисления сумм. Таким образом, правильно уравнение читается как $Y = A \text{ ИЛИ } (B \text{ И } C)$. **уравнение (2.1)** – ещё один пример, показывающий порядок операций.

$$\bar{A} B + BC\bar{D} = ((\bar{A})B + (BC(\bar{D}))) \quad (2.1)$$

2.2.2 Дизъюнктивная форма

Таблица истинности для функции N переменных содержит 2^N строк, по одной для каждой возможной комбинации значений входов. Каждой строке в таблице истинности соответствует минтерм, который имеет значение ИСТИНА для этой строки. На **Рис. 2.8** показана таблица истинности функции двух переменных A и B . В каждой строке показан соответствующий ей минтерм. Например, минтерм для первой строки –

это $\bar{A}\bar{B}$, поскольку $\bar{A}\bar{B}$ имеет значение ИСТИНА тогда, когда $A = 0$ и $B = 0$. Минтермы нумеруют начиная с 0; первая строка соответствует минтерму 0 (m_0), следующая строка – минтерму 1 (m_1), и так далее.

A	B	Y	minterm	minterm name
0	0	0	$\bar{A}\bar{B}$	m_0
0	1	1	$\bar{A}B$	m_1
1	0	0	$A\bar{B}$	m_2
1	1	0	AB	m_3

Рис. 2.8 Таблица истинности и минтермы

Можно написать булево уравнение для любой таблицы истинности путем суммирования всех тех минтермов, для которых выход Y имеет значение ИСТИНА. Например, на **Рис. 2.8** есть только одна строка (минтерм), для которой выход Y имеет значение ИСТИНА, она отмечена синим цветом. Таким образом, $Y = \bar{A}B$. На **Рис. 2.9** показана таблица, в которой выход имеет значение ИСТИНА для нескольких строк. Суммирование отмеченных минтермов дает $Y = \bar{A}B + AB$.

Такая сумма минтермов называется *совершенной дизъюнктивной нормальной формой* функции (sum-of-products canonical form). Она представляет собой сумму (операцию «ИЛИ») произведений (операций «И», образующих минтермы). Хотя существует много

способов записать одну и ту же функцию, такую как $Y = \bar{A}B + AB$, мы будем записывать минтермы в том же порядке, как в таблице истинности, чтобы всегда получать одно и то же булево выражение для одной и той же таблицы истинности. Совершенная дизъюнктивная нормальная форма также может быть записана через символ суммы Σ . При использовании такого обозначения функция на Рис. 2.9 будет выглядеть так:

$$F(A, B) = \Sigma(m_1, m_3)$$

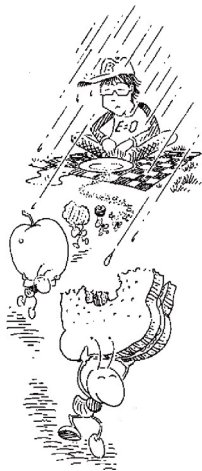
или

$$F(A, B) = \Sigma(1, 3) \quad (2.2)$$

A	B	Y	minterm	minterm name
0	0	0	$\bar{A} \bar{B}$	m_0
0	1	1	$\bar{A} B$	m_1
1	0	0	$A \bar{B}$	m_2
1	1	1	$A B$	m_3

Рис. 2.9 Таблица истинности с несколькими минтермами, равными ИСТИНЕ

Пример 2.2 ДИЗЬЮНКТИВНАЯ ФОРМА



У Бена Битдидла намечается пикник. Он не обрадуется, если пойдёт дождь или появятся муравьи. Постройте схему, в которой выход будет принимать значение ИСТИНА только в том случае, если Бену пикник понравится.

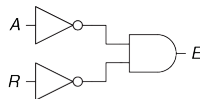
Решение: Сначала определим входы и выходы. Входами будут переменные A и R , что означает муравьёв (ants) и дождь (rain). Значение A принимает значение ИСТИНА, когда муравьи есть, и ЛОЖЬ, когда муравьёв нет. Аналогично, R имеет значение ИСТИНА, когда идёт дождь, и ЛОЖЬ, когда Бену светит солнце. Выход E (enjoyment, радость) показывает настроение Бена. E имеет значение ИСТИНА, когда Бен радуется пикнику, и ЛОЖЬ, когда он страдает. На **Рис. 2.10** показана таблица истинности впечатлений Бена от пикника.

Используя дизъюнктивную форму, запишем уравнение так: $E = \bar{A}\bar{R}$ или $E = \Sigma(0)$. Мы можем реализовать соответствующую схему, используя два инвертора и двухвходовой элемент И, как показано на **Рис. 2.11 (а)**. Вы могли заметить, что эта таблица является точно такой же, как и таблица для функции «ИЛИ-НЕ», рассмотренной в **разделе 1.5.5**: $E = A$ ИЛИ-НЕ $R = \overline{A + R}$

На **Рис. 2.11 (b)** показана реализация на базе элемента ИЛИ-НЕ. В **разделе 2.3** мы покажем, что выражения $\overline{A} + \overline{R}$ и $\overline{A + R}$ эквивалентны.

<i>A</i>	<i>R</i>	<i>E</i>
0	0	1
0	1	0
1	0	0
1	1	0

Рис. 2.10 Таблица истинности Бена



(a)



(b)

Рис. 2.11 Схема Бена

Совершенная дизъюнктивная нормальная форма позволяет записать булево уравнение для любой таблицы истинности с любым количеством переменных. На **Рис. 2.12** показана произвольная таблица истинности для трехвходового элемента. Совершенная дизъюнктивная нормальная форма соответствующей логической функции выглядит так:

$$Y = \overline{A}\overline{B}\overline{C} + A\overline{B}\overline{C} + A\overline{B}C$$

или

$$Y = \Sigma(0, 4, 5)$$

(2.3)

<i>A</i>	<i>B</i>	<i>C</i>	<i>Y</i>
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Рис. 2.12 Произвольная таблица истинности с тремя входами

К сожалению, совершенная дизъюнктивная нормальная форма не всегда позволяет получить простое уравнение. В [разделе 2.3](#) мы покажем, как записать одну и ту же функцию, используя меньшее число членов уравнения.

2.2.3 Конъюнктивная форма

Альтернативный способ выражения булевых функций – это *совершенная конъюнктивная нормальная форма* (products-of-sum forms). Каждая строка таблицы истинности соответствует макстерму, который имеет значение ЛОЖЬ для этой строки. Например, макстерм для первой строки для двухвходовой таблицы истинности – это $(A + B)$, поскольку $(A + B)$ имеет значение ЛОЖЬ, когда $A = 0$ и $B = 0$. Для любой схемы, заданной таблицей истинности, мы можем записать ее булево

уравнение как логическое «И» всех макстермов, для которых выход имеет значение ЛОЖЬ. Совершенная конъюнктивная нормальная форма также может быть записана с использованием символа Π .

Пример 2.3 КОНЪЮНКТИВНАЯ ФОРМА

A	B	Y	maxterm	maxterm name
0	0	0	$A + B$	M_0
0	1	1	$A + \bar{B}$	M_1
1	0	0	$\bar{A} + B$	M_2
1	1	1	$\bar{A} + \bar{B}$	M_3

Рис. 2.13 Таблица истинности с макстермами

что $Y = 0$ для $A = 0$ и $B = 0$, так как логическое «И» любого значения и нуля дает ноль. Аналогично, второй макстерм ($\bar{A} + B$) гарантирует, что $Y = 0$ для комбинации $A = 1$ и $B = 0$. На Рис. 2.13 показана такая же таблица истинности, как и на Рис. 2.9, чтобы продемонстрировать, что одна и та же функция может быть записана более чем одним способом.

Запишите уравнение в совершенной конъюнктивной нормальной форме для таблицы истинности на Рис. 2.13.

Решение: Таблица истинности имеет две строки, в которых выход имеет значение ЛОЖЬ. Следовательно, функция может быть записана в конъюнктивной форме так:

$Y = (A + B)(\bar{A} + B)$. Также функция может быть записана как $Y = \Pi(M_0, M_2)$ или $Y = \Pi(0, 2)$. Первый макстерм, $(A + B)$, гарантирует,

Аналогично, булево уравнение для пикника Бена (**Рис. 2.10**) может быть записано в совершенной конъюнктивной нормальной форме, если обвести три строки с нулями для того, чтобы получить

$$E = (A + \bar{R})(\bar{A} + R)(\bar{A} + \bar{R}) \text{ или } E = \Pi(1, 2, 3).$$

Это не такая красивая запись, как дизъюнктивное уравнение, $E =$, но эти два уравнения логически эквивалентны. Дизъюнктивная форма дает более короткое уравнение, когда выход имеет значение ИСТИНА только в нескольких строках таблицы истинности; конъюнктивная же форма проще, когда выход имеет значение ЛОЖЬ только в нескольких строках таблицы истинности.

2.3 БУЛЕВА АЛГЕБРА

В предыдущем разделе мы изучили, как записывать булевы выражения при наличии таблицы истинности. Однако, выражение, получаемое таким способом, не обязательно приводит к простейшему набору логических элементов. Вы можете использовать булеву алгебру для упрощения булевых уравнений точно так же, как вы используете алгебру для упрощения математических уравнений. Правила булевой алгебры очень похожи на правила обычной алгебры, но в некоторых случаях они проще, потому что переменные имеют только два возможных значения: 0 или 1.

Булева алгебра основана на наборе аксиом, которые мы считаем верными. Аксиомы являются недоказуемыми в том смысле, что определение не может быть доказано. С помощью этих аксиом мы доказываем все теоремы булевой алгебры.

Эти теоремы имеют огромную практическую значимость, потому что с их помощью мы учимся тому, как упрощать логические уравнения, чтобы получать более дешевые и компактные схемы. Аксиомы и теоремы булевой алгебры подчиняются принципу двойственности. Если взаимно заменить символы 0 и 1, а так же взаимно заменить операторы \cdot (И) и $+$ (ИЛИ), то булево выражение останется верным. Мы используем символ «штрих» (') для обозначения двойственного выражения.

2.3.1 Аксиомы

В **Табл. 2.1** приведены аксиомы булевой алгебры. Эти пять аксиом и двойственные им аксиомы определяют булевы переменные и значения операторов НЕ, И, ИЛИ. Аксиома $A1$ показывает, что булева переменная B имеет значение 0, если она не имеет значение 1. Двойственное выражение для этой аксиомы $A1'$ утверждает, что переменная принимает значение 1, если она не имеет значение 0. Вместе аксиомы $A1$ и $A1'$ говорят нам, что мы работаем в булевом, то есть бинарном поле, состоящем из значений нулей и единиц. Аксиомы

A_2 и A_2' определяют операцию НЕ. Аксиомы с A_3 по A_5 определяют операцию И, а их двойственные аксиомы (A_3' – A_5') определяют операцию ИЛИ.

Табл. 2.1 Аксиомы булевой алгебры

Аксиома		Двойственная аксиома		Название
A_1	$B = 0$ если $B \neq 1$	A_1'	$B = 1$ если $B \neq 0$	Бинарное поле
A_2	$\bar{0} = 1$	A_2'	$\bar{1} = 0$	НЕ
A_3	$0 \cdot 0 = 0$	A_3'	$1 + 1 = 1$	И/ИЛИ
A_4	$1 \cdot 1 = 1$	A_4'	$0 + 0 = 0$	И/ИЛИ
A_5	$0 \cdot 1 = 1 \cdot 0 = 0$	A_5'	$1 + 0 = 0 + 1 = 1$	И/ИЛИ

2.3.2 Теоремы одной переменной

Теоремы с T_1 по T_5 в **Табл. 2.2** описывают, как упростить уравнения, содержащие одну переменную.

Теорема *идентичности* T_1 утверждает, что для любой булевой переменной B выполнено $B \text{ И } 1 = B$. Двойственная ей теорема говорит о том, что $B \text{ ИЛИ } 0 = B$. В аппаратуре, как показано на **Рис. 2.14**, T_1 означает, что если уровень сигнала на одном из входов двухвходового элемента И всегда равен 1, то мы можем удалить этот элемент и заменить его проводом, соединяющим выход этого элемента с входом B , значение которого может меняться. Точно так же теорема

T1' говорит о том, что если один вход двухвходового элемента ИЛИ всегда равен 0, мы можем заменить этот элемент на провод, соединенный с входом B . Как правило, элементы имеют определенную стоимость, энергопотребление и задержку прохождения сигнала, поэтому замена элемента на провод является целесообразной.

Табл. 2.2 Булевы теоремы для одной переменной

Теорема		Двойственная теорема		Название
T1	$B \cdot 1 = B$	T1'	$B + 0 = B$	Идентичность
T2	$B \cdot 0 = 0$	T2'	$B + 1 = 1$	Нулевой элемент
T3	$B \cdot B = B$	T3'	$B + B = B$	Идемпотентность
T4		$\overline{\overline{B}} = B$		Инволюция
T5	$B \cdot \overline{B} = 0$	T'	$B + \overline{B} = 1$	Дополнительность

Теорема о *нулевом элементе* T2 говорит, что B И 0 всегда равно 0. Следовательно, 0 называют нулевым элементом для операции И, потому что он обнуляет эффект любого другого входа. Двойственная ей теорема говорит о том, что B ИЛИ 1 всегда равно 1. Таким образом, 1 – это нулевой элемент для операции ИЛИ. В аппаратуре, как показано на Рис. 2.15, если один вход элемента И равен 0, мы можем заменить элемент И проводом, подключенным к низкому логическому уровню (0). Точно так же, если один из входов элемента ИЛИ равен 1, мы можем

заменить элемент ИЛИ на провод, который подключен к высокому логическому уровню (1).

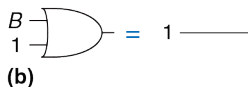
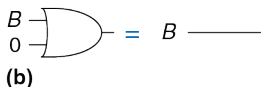
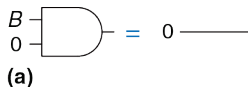
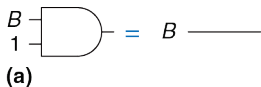


Рис. 2.14 Теорема идентичности в аппаратуре: (a) T1, (b) T1'

Рис. 2.15 Теорема о нулевом элементе в аппаратуре: (a) T2, (b) T2'

Теорема о нулевом элементе приводит к нелепым утверждениям, которые при этом оказываются верными! Эта теорема становится особенно опасной, когда её применяют те, кто делает рекламу: «ВЫ ПОЛУЧИТЕ МИЛЛИОН ДОЛЛАРОВ или мы пришлём вам по почте зубную щётку» (скорее всего, вы получите зубную щётку по почте).

Теорема об *идемпотентности* T3 утверждает, что операция логического «И» двух равных друг другу переменных имеет значение, равное этой переменной. Аналогичное утверждение верно для операции «ИЛИ» с двумя одинаковыми значениями на входах.

Название теоремы происходит от латинских слов «idem» – *тот же*, *такой же* и «potent» – *сила*. Операции возвращают те же значения, которые вы подаете им на вход. На **Рис. 2.16** показано, как идемпотентность позволяет заменить элемент схемы на провод.

Теорема об *инволюции* Т4 – это забавный способ описания того, что двойное отрицание переменной дает её исходное значение. Два последовательно включенных инвертора логически отменяют друг друга, то есть они эквивалентны проводу, как показано на **Рис. 2.17**. Двойственной ей теоремой является она сама.

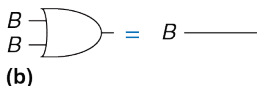
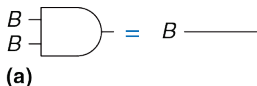


Рис. 2.16 Теорема об идемпотентности в аппаратуре: (a) Т3, (b) Т3'

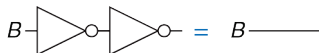


Рис. 2.17 Теорема о дополнительности в аппаратуре: (a) Т5, (b) Т5'

Теорема о *дополнительности* Т5 (**Рис. 2.18**) утверждает, что операция И над переменной и её инверсным значением дает 0 (потому что одна

из них всегда будет равна нулю). И, согласно принципу двойственности, операция ИЛИ над переменной и её инверсным значением всегда дает 1 (так как одна из них всегда будет равна единице).

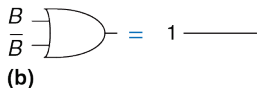
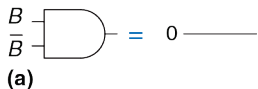


Рис. 2.18 Теорема инволюции в аппаратуре: (a) T4, (b) T4'

2.3.3 Теоремы с несколькими переменными

Теоремы с T6 по T12 в Табл. 2.3 описывают, как упростить уравнения, включающие в себя более одной булевой переменной.

Теоремы T6 о *коммутативности* и T7 об ассоциативности работают так же, как и в традиционной алгебре. В соответствии с принципом коммутативности порядок входов для функций И или ИЛИ не влияет на значение выхода. Согласно принципу ассоциативности любое группирование входов не влияет на значение выхода.

Теорема о *дистрибутивности* Т8 является точно такой же, как и в традиционной алгебре, а двойственная ей теорема Т8' – нет. Согласно теореме Т8 оператор И дистрибутивен относительно операции ИЛИ. Т8' говорит, что оператор ИЛИ дистрибутивен относительно операции И. В традиционной алгебре оператор умножения дистрибутивен относительно операции сложения, но не наоборот, то есть

$$(B + C) \times (B + D) \neq B + (C \times D).$$

Теоремы *поглощения*, *склеивания* и *согласованности* Т9 – Т11 позволяют нам удалять излишние переменные. Если вы немного подумаете, вы сможете убедиться, что эти теоремы справедливы.

Табл. 2.3 Булевы теоремы для нескольких переменных

Теорема		Двойственная теорема		Название
Т6	$B \cdot C = C \cdot B$	Т6'	$B + C = C + B$	Коммутативность
Т7	$(B \cdot C) \cdot D = B \cdot (C \cdot D)$	Т7'	$(B + C) + D = B + (C + D)$	Ассоциативность
Т8	$(B \cdot C) + (B \cdot D) = B \cdot (C + D)$	Т8'	$(B + C) \cdot (B + D) = B + (C \cdot D)$	Дистрибутивность
Т9	$B \cdot (B + C) = B$	Т9'	$B + (B \cdot C) = B$	Поглощение
Т10	$(B \cdot C) + (B \cdot \bar{C}) = B$	Т10'	$(B + C) \cdot (B + \bar{C}) = B$	Склеивание
Т11	$(B \cdot C) + (\bar{B} \cdot D) + (C \cdot D) = B \cdot C + \bar{B} \cdot D$	Т11'	$(B + C) \cdot (\bar{B} + D) \cdot (C + D) = (B + C) \cdot (\bar{B} + D)$	Согласованность
Т12	$\overline{B_0 \cdot B_1 \cdot B_2 \dots} = (\bar{B}_0 + \bar{B}_1 + \bar{B}_2 \dots)$	Т12'	$\overline{B_0 + B_1 + B_2 \dots} = (\bar{B}_0 \cdot \bar{B}_1 \cdot \bar{B}_2 \dots)$	Теорема де Моргана

Теорема де Моргана T12 является особенно мощным инструментом при разработке цифровых устройств. Эта теорема поясняет, что дополнение результата умножения всех термов равно сумме дополнений каждого терма. Аналогично дополнение суммы всех термов равно результату умножения дополнений каждого терма.

В соответствии с теоремой де Моргана, элемент И-НЕ эквивалентен элементу ИЛИ с инвертированными входами. Аналогично, ИЛИ-НЕ эквивалентен элементу И с инвертированными входами. На **Рис. 2.19** показаны эквивалентные по де Моргану элементы И-НЕ и ИЛИ-НЕ. Каждая пара символов, приведенная для каждой функции, называется двойственной. Они логически эквивалентны и взаимозаменяемы.

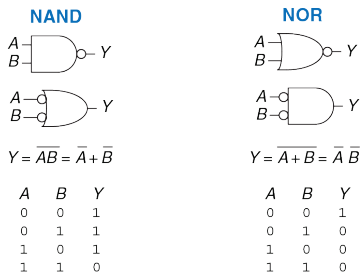


Рис. 2.19 Эквивалентные по де Моргану элементы

Кружочек на графическом обозначении элементов является обозначением отрицания (инверсии). Интуитивно вы можете представить, что если «вдавить» этот кружочек с одной стороны логического элемента, то он «выскочит» на другой, при этом тип элемента изменится с И на ИЛИ (и наоборот). Это называется «перемещением инверсии». Например, элемент И-НЕ на **Рис. 2.19** состоит из элемента И с отрицанием на выходе. Перемещение инверсии влево приводит к получению элемента ИЛИ с двумя отрицаниями на входах. Базовые правила для перемещения инверсии таковы:

- ▶ Перемещение инверсии назад (от выхода) или вперед (от входов) меняет тип элемента с И на ИЛИ и наоборот;
- ▶ Перемещение инверсии с выхода назад ко входам приводит к тому, что на всех входах появляется инверсия;
- ▶ Перемещение инверсии со всех входов элемента к выходу приводит к появлению инверсии на выходе.

В **разделе 2.5.2** принцип перемещения инверсии используется для анализа схем.



Август де Морган

Август де Морган, умер в 1871 г. Британский математик, родился в Индии. Был слепым на один глаз. Его отец умер, когда ему было 10 лет. Поступил в Тринити Колледж в Кембридже, и был назначен профессором математики в возрасте 22 лет в только что открытом в то время Лондонском университете. Много писал на различные математические темы, включая логику, алгебру и парадоксы. В честь де Моргана был назван кратер на Луне. Он придумал загадку про год своего рождения: «Мне было X лет в году X^2 ».

Пример 2.4 ПОЛУЧИТЕ КОНЪЮНКТИВНУЮ ФОРМУ

На **Рис. 2.20** приведена таблица истинности для булевой функции Y и её дополнения \bar{Y} . Используя теорему де Моргана, получите конъюнктивную нормальную форму функции Y из дизъюнктивной формы \bar{Y} .

Решение: На **Рис. 2.21** обведены минтермы, содержащиеся в функции Y . Дизъюнктивная нормальная форма функции Y имеет следующий вид:

$$\bar{Y} = \bar{A} \bar{B} + \bar{A} B \quad (2.4)$$

Применяя операцию инверсии к обеим частям уравнения и дважды используя теорему де Моргана, получаем:

$$\overline{\bar{Y}} = \overline{(Y)} = \overline{\bar{A} \bar{B} + \bar{A} B} = (\overline{\bar{A} \bar{B}})(\overline{\bar{A} B}) = (A + B)(A + \bar{B}) \quad (2.5)$$

A	B	Y	\bar{Y}
0	0	0	1
0	1	0	1
1	0	1	0
1	1	1	0

Рис. 2.20 Таблица истинности, показывающая Y и \bar{Y}

A	B	Y	\bar{Y}	minterm
0	0	0	1	$\bar{A} \bar{B}$
0	1	0	1	$\bar{A} B$
1	0	1	0	$A \bar{B}$
1	1	1	0	$A B$

Рис. 2.21 Таблица истинности, показывающая Y и \bar{Y}

2.3.4 Правда обо всем этом

Любопытный читатель может задать вопрос о том, как же доказать правильность теоремы. В булевой алгебре доказательство теорем с конечным числом переменных является простым: нужно показать, что

теорема верна для всех возможных значений этих переменных. Этот метод называется *совершенной индукцией* и может быть выполнен с использованием таблицы истинности.

Пример 2.5 ДОКАЗАТЕЛЬСТВО ТЕОРЕМЫ СОГЛАСОВАННОСТИ МЕТОДОМ ПОЛНОГО ПЕРЕБОРА

Докажите теорему согласованности T11 из **Табл. 2.3**.

Решение: проверьте обе части уравнения для всех восьми комбинаций переменных B , C и D . Таблица истинности на **Рис. 2.22** иллюстрирует все эти комбинации. Поскольку равенство $BC + \bar{B}D + CD = BC + \bar{B}D$ верно для всех случаев, теорема доказана.

B	C	D	$BC + \bar{B}D + CD$	$BC + \bar{B}D$
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	0	0
1	1	0	1	1
1	1	1	1	1

Рис. 2.22 Таблица истинности, доказывающая T11

2.3.5 Упрощение уравнений

Теоремы булевой алгебры помогают нам упрощать булевы уравнения. Например, возьмём дизъюнктивную форму выражения из таблицы истинности на **Рис. 2.9**: $Y = \bar{A}\bar{B} + A\bar{B}$. В соответствии с теоремой T10, уравнение можно упростить до $Y = \bar{B}$. Возможно, это очевидно при взгляде на таблицу истинности. В общем случае может потребоваться несколько шагов для упрощения более сложных уравнений.

Основной принцип упрощения дизъюнктивных уравнений – это комбинирование термов с использованием отношения $PA + P\bar{A} = P$, где P может быть любой импликантой. Насколько может быть упрощено уравнение? По определению уравнение дизъюнктивной формы является минимизированным, если оно включает в себя минимально возможное количество импликант. Если есть несколько уравнений с одинаковым количеством импликант, минимальным будет то уравнение, в котором меньше литералов.

Импликанта называется простой (prime implicant), если она не может быть объединена с другими импликантами в уравнении для того, чтобы образовать новую импликанту с меньшим количеством литералов. Все импликанты в минимальном уравнении должны быть простыми. Иначе, они могут быть объединены, чтобы уменьшить количество литералов.

Пример 2.6 МИНИМИЗАЦИЯ УРАВНЕНИЯ

Минимизируйте **уравнение (2.3)**: $\overline{A}\overline{B}\overline{C} + A\overline{B}\overline{C} + A\overline{B}C$

Решение: Мы начинаем с исходного уравнения и применяем булевы теоремы шаг за шагом, как показано в **Табл. 2.4**.

Упростили ли мы полностью уравнение на этой стадии? Давайте посмотрим внимательно. В оригинальном уравнении минтермы $\overline{A}\overline{B}\overline{C}$ и $A\overline{B}\overline{C}$ отличаются только переменной A . Поэтому мы объединяем минтермы и получаем $\overline{B}\overline{C}$. Однако, если мы посмотрим на исходное уравнение, мы заметим, что последние два минтерма $A\overline{B}\overline{C}$ и $A\overline{B}C$ также отличаются одним литералом (C и \overline{C}). Таким образом, используя тот же самый метод, мы могли бы объединить эти два минтерма и получить минтерм $A\overline{B}$. Можно сказать, что импликанты $\overline{B}\overline{C}$ и $A\overline{B}$ делят между собой минтерм $A\overline{B}\overline{C}$.

Итак, остановились ли мы на упрощении только одной пары минтермов, или мы можем упростить обе? Используя теорему об идемпотентности, мы можем дублировать минтермы столько раз, сколько нам нужно: $B = B + B + B + B \dots$ Используя этот принцип, мы полностью упрощаем уравнение до его простых импликант, $\overline{B}\overline{C} + A\overline{B}$, как показано в **Табл. 2.5**.

Табл. 2.4 Минимизация выражения

Шаг	Выражение	Объяснение
	$\bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C$	
1	$\bar{B}\bar{C}(\bar{A} + A) + A\bar{B}C$	T8: дистрибутивность
2	$\bar{B}\bar{C}(1) + A\bar{B}C$	T5: дополнительность
3	$\bar{B}\bar{C} + A\bar{B}C$	T1: идентичность

Табл. 2.5 Улучшенная минимизация выражения

Шаг	Выражение	Объяснение
	$\bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C$	
1	$\bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C$	T3: идемпотентность
2	$\bar{B}\bar{C}(\bar{A} + A) + A\bar{B}(\bar{C} + C)$	T8: дистрибутивность
3	$\bar{B}\bar{C}(1) + A\bar{B}(1)$	T5: дополнительность
4	$\bar{B}\bar{C} + A\bar{B}$	T1: идентичность

Хотя это немного нелогично, расширение импликанты (например, превращение AB в $ABC + AB\bar{C}$) иногда полезно при минимизации уравнений. Делая так, вы можете повторять один из расширенных минтермов для его объединения с другим минтермом.

Вы могли заметить, что полное упрощение булевых уравнений при помощи теорем булевой алгебры может потребовать нескольких попыток,

некоторые из которых будут ошибочными. В разделе 2.7 описана методика, позволяющая упростить процесс минимизации – карты Карно.

Зачем же трудиться над упрощением булева уравнения, если оно остается логически эквивалентным? Упрощение уменьшает количество элементов, используемых при физическом воплощении функции в аппаратуре, тем самым делая схему меньше, дешевле и, возможно, быстрее. В следующем разделе рассказывается, как воплощать булевы уравнения при помощи логических элементов.

2.4 ОТ ЛОГИКИ К ЛОГИЧЕСКИМ ЭЛЕМЕНТАМ

Принципиальная схема – это изображение цифровой схемы, показывающее элементы и соединяющие их проводники. Например, схема на **Рис. 2.23** показывает возможную аппаратную реализацию нашей любимой логической функции (**уравнение (2.3)**):

$$Y = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + \bar{A}B\bar{C}$$

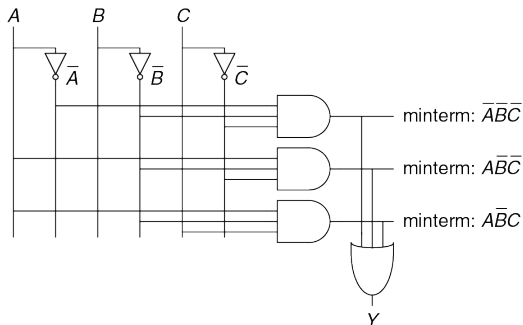


Рис. 2.23 Схема $Y = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + \bar{A}B\bar{C}$

Изображая принципиальные схемы в унифицированном виде, нам становится легче читать их и отлаживать. В большинстве случаев мы будем придерживаться следующих правил:

- ▶ Входы изображаются на левой (или верхней) части схемы;
- ▶ Выходы изображаются на правой (или нижней) части схемы;
- ▶ Всегда, когда это возможно, элементы необходимо изображать слева направо;
- ▶ Проводники лучше изображать прямыми линиями, чем линиями с множеством углов (неровные рваные линии отвлекают внимание: приходится следить за тем, куда ведут провода, а не думать о том, что делает схема);
- ▶ Проводники всегда должны соединяться в виде буквы «Т»;
- ▶ Точка в месте пересечения проводников обозначает их соединение;
- ▶ Проводники, пересекающиеся без точки, не имеют соединения друг с другом.

Три последних правила показаны на **Рис. 2.24**.

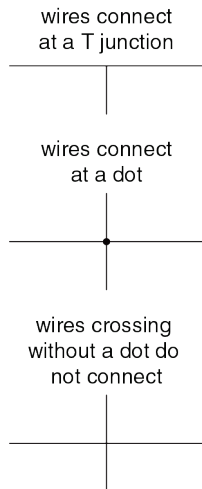


Рис. 2.24
Wireconnections

Любое булево уравнение в дизъюнктивной форме может быть изображено в виде принципиальной схемы с использованием систематического подхода, как показано на [Рис. 2.23](#). Сначала нарисуйте вертикальные проводники для входов. Поместите инверторы на соседних вертикальных линиях для получения комплементарных входов, если это необходимо. Нарисуйте горизонтальные линии, ведущие к элементам И, для каждого минтерма. Затем для каждого выхода нарисуйте элемент ИЛИ, соединенный с минтермом, соответствующим этому выходу. Такой стиль изображения называется программируемой логической матрицей (ПЛМ, PLA), потому что инверторы, элементы И и элементы ИЛИ систематически объединены в массивы. Программируемые логические матрицы будут рассмотрены в [разделе 5.6](#).

На [Рис. 2.25](#) показана реализация упрощенного уравнения, которое мы получили при помощи булевой алгебры в Примере 2.6. Заметьте, что упрощенная схема имеет значительно меньше аппаратных элементов,

чем схема на **Рис. 2.23**. Также ее быстродействие может быть выше, поскольку она использует элементы с меньшим количеством входов.

Мы даже можем ещё уменьшить количество элементов (пусть хотя бы на один инвертор), если воспользуемся преимуществом инвертирующих логических элементов. Заметьте, что $\overline{B}C$ – это элемент И с инвертированными входами. На **Рис. 2.26** показана схема, которая использует эту оптимизацию для исключения инвертора на входе C . Вспомните, что согласно теореме де Моргана логический элемент И с инвертированными входами эквивалентен элементу ИЛИ-НЕ. В зависимости от технологии реализации, использование наименьшего числа элементов или использование элементов определенного типа взамен других может быть выгоднее. Например, в технологии КМОП элементы И-НЕ и ИЛИ-НЕ более предпочтительны, чем И или ИЛИ.

У многих схем имеется несколько выходов, каждый из которых вычисляет независимые булевы функции для входов. Мы можем записать отдельные таблицы истинности для каждого выхода, но часто удобно записать все выходы в одну таблицу истинности и начертить одну схему для всех выходов.

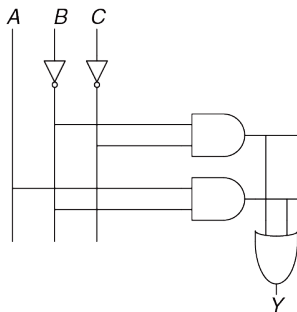


Рис. 2.25 Схема реализации функции
 $Y = B + \bar{C} + A\bar{B}$

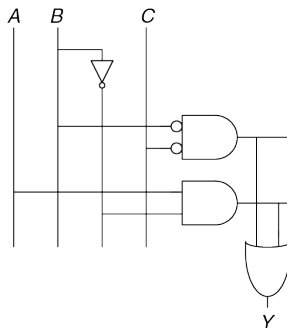


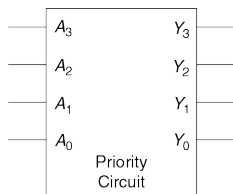
Рис. 2.26 Схема, использующая меньше элементов

Пример 2.7 СХЕМЫ С НЕСКОЛЬКИМИ ВЫХОДАМИ

Декан, заведующий кафедрой, аспирант и председатель совета общежития время от времени используют одну аудиторию. К сожалению, иногда аудитория нужна им одновременно, что приводит к катастрофам, как, например, когда встреча декана с пожилыми и уважаемыми членами попечительского совета была запланирована на то же время, что и пивная вечеринка студентов общежития. Алиса Хакер была приглашена для того, чтобы разработать систему резервирования комнаты.

Система имеет четыре входа (A_3, \dots, A_0) и четыре выхода (Y_3, \dots, Y_0). Эти сигналы также могут быть записаны в виде $A_{3:0}$ и $Y_{3:0}$. Каждый пользователь активирует свой вход, когда запрашивает аудиторию на следующий день. Система активирует только один выход, подтверждая пользование аудиторией самым высокоприоритетным пользователем. Декан, который оплачивает систему, требует наивысший приоритет (3). Заведующий кафедрой, аспирант и председатель совета общежития имеют приоритеты по убыванию. Запишите таблицу истинности и булевы уравнения для этой системы. Начертите схему, которая будет выполнять эту функцию.

Решение: Данная функция называется четырехвходовой схемой приоритета. Её обозначение и таблица истинности показаны на [Рис. 2.27](#). Мы могли бы записать каждый выход в дизъюнктивной форме и упростить уравнения, используя булеву алгебру. Однако достаточно посмотреть на функциональное описание (таблицу истинности), чтобы понять, каковы могут быть упрощенные уравнения: Y_3 имеет значение ИСТИНА всегда, когда подается сигнал A_3 , таким образом $Y_3 = A_3$. Y_2 равен ИСТИНЕ, если подан сигнал A_2 и не подан сигнал A_3 , таким образом $Y_2 = \bar{A}_3 A_2$. Y_1 имеет значение ИСТИНА, если подан сигнал A_1 и ни на какой из более высокоприоритетных входов сигнал не подан: $Y_1 = \bar{A}_3 \bar{A}_2 A_1$. Y_0 имеет значение ИСТИНА при поданном сигнале A_0 и когда ни один из других выходов не активирован: $Y_0 = \bar{A}_3 \bar{A}_2 \bar{A}_1 A_0$. Схема показана на [Рис. 2.28](#). Опытный разработчик часто может реализовать логическую схему, непосредственно глядя в исходные данные. При наличии четко заданной спецификации, просто преобразуйте слова в уравнения, а уравнения в логические элементы схемы.



A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

Рис. 2.27 Схема приоритета

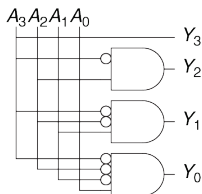


Рис. 2.28 Принципиальная схема

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	X	0	0	1	0
0	1	X	X	0	1	0	0
1	X	X	X	1	0	0	0

Рис. 2.29 Таблица истинности схемы приоритета

Символ «X» используется не только для обозначения переменных, чье состояние нам безразлично, но и для обозначения недопустимых состояний сигналов при симуляции логических схем (см. [Раздел 2.6.1](#)). Старайтесь понять из контекста, о каком варианте использования идет речь. Чтобы избежать такой двусмысленности, некоторые авторы используют символы «D» или «?» для обозначения сигналов, состояние которых нам безразлично.

Обратите внимание, что если в схеме приоритета подается сигнал A_3 , то выходы схемы не будут зависеть от того, какие сигналы присутствуют на остальных входах. Мы используем символ X для описания состояния входов, которые нам безразличны, так как не оказывают влияния на выход. На [Рис. 2.29](#) показано, что таблица

истинности четырехвходовой приоритетной схемы становится гораздо меньше, если убрать значения входов, которыми можно пренебречь. Из этой таблицы истинности мы можем легко получить булевы уравнения в дизъюнктивной форме, опуская входы с Х. Значения, которыми можно пренебречь, также могут возникнуть на выходах в таблице истинности, как мы увидим в [разделе 2.7.3](#).

2.5 МНОГОУРОВНЕВАЯ КОМБИНАЦИОННАЯ ЛОГИКА

Комбинационная логика, построенная как дизъюнкция конъюнкций (сумма произведений), называется двухуровневой, потому что состоит из литералов, соединенных с элементами И (образующими первый уровень), выходы которых соединены с элементами ИЛИ (образующими второй уровень). Разработчики часто создают схемы с большим числом уровней логических элементов. Такая многоуровневая комбинационная схема может использовать меньше логических элементов, чем ее двухуровневая реализация. Эквивалентные преобразования по законам де Моргана и перемещение инверсии особенно полезны при анализе и разработке многоуровневых схем.

2.5.1 Минимизация аппаратуры

Некоторые логические функции требуют огромного количества аппаратуры, если строить их с использованием двухуровневой логики. Показательный пример – это функция ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR) нескольких переменных. Например, рассмотрим построение трехвходового элемента XOR, используя двухуровневую технику, которую мы изучали до сих пор.

Вспомним, что N-входовой XOR выдает на выход значение ИСТИНА, если нечетное число входных операндов имеют значение ИСТИНА. На **Рис. 2.30 (а)** показана таблица истинности трехвходового элемента XOR. В таблице обведены строки, для которых значение выхода будет ИСТИНА. Из таблицы истинности мы понимаем форму логического выражения, соответствующую дизъюнкции конъюнкций (сумме произведений) **уравнения (2.6)**. К сожалению, это выражение невозможно упростить в меньшее количество импликант.

$$Y = \bar{A}B\bar{C} + \bar{A}\bar{B}C + A\bar{B}\bar{C} + ABC \quad (2.6)$$

С другой стороны, $A \oplus B \oplus C = (A \oplus B) \oplus C$ (если вы сомневаетесь, докажите это самостоятельно с помощью совершенной индукции). Следовательно, трехвходовой элемент XOR можно реализовать каскадом двухвходовых элементов XOR, как показано на **Рис. 2.31**.

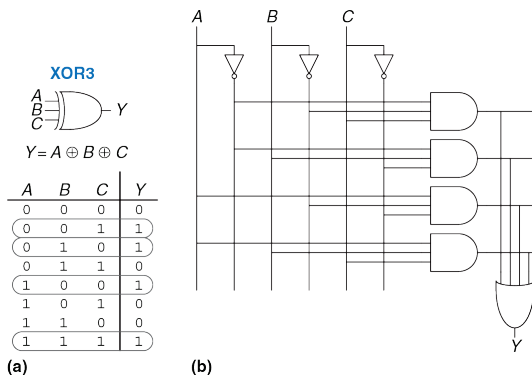


Рис. 2.30 Трехвходовой элемент XOR: функциональная спецификация (a) и реализация с двумя уровнями логики (b)

Аналогично, восьмивходовой XOR требует 128 восьмивходовых элементов И и одного 128-входового элемента ИЛИ для двухуровневой реализации дизъюнкции конъюнкций. Гораздо лучшей альтернативой будет использовать дерево двухвходовых элементов XOR, как показано на [Рис. 2.32](#).



Рис. 2.31 Трехвходовой элемент XOR, собранный из двух двухвходовых элементов XOR

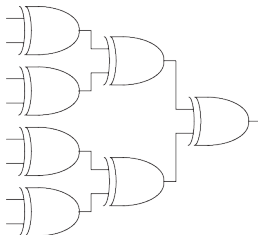


Рис. 2.32 Восьмивходовой элемент XOR, собранный из семи двухвходовых

Выбор наилучшей многоуровневой реализации заданной логической функции – это непростой процесс (выбирать наилучшую многоуровневую реализацию заданной логической функции непросто). Кроме того, «наилучшее» имеет много значений: наименьшее количество элементов, лучшее быстродействие, кратчайшее время разработки, наименьшая стоимость, наименьшее энергопотребление.

В **главе 5** вы увидите, что «наилучшая» схема для одной технологии не обязательно является наилучшей для другой. Например, мы использовали элементы И и ИЛИ, но для КМОП-технологии более эффективны элементы И-НЕ и ИЛИ-НЕ. С опытом, вы увидите, что для

большинства схем вы сможете находить хорошую многоуровневую реализацию, просто рассматривая эти схемы (и действуя по интуиции).

Некоторый опыт вы наработаете, изучая примеры схем остальной части книги. По мере того, как вы учитесь, исследуйте различные варианты разработки и думайте о компромиссах. Сейчас также доступны системы автоматизированного проектирования (САПР), которые позволяют рассматривать огромное пространство возможных многоуровневых реализаций (осуществлять поиск в многомерном пространстве решений) и находить такое, которое наилучшим образом удовлетворяет вашим критериям оптимальности с учетом имеющихся строительных блоков.

2.5.2 Перемещение инверсии

Как вы помните из [раздела 1.7.6](#) для КМОП-схем лучше подходят элементы И-НЕ и ИЛИ-НЕ, а не И и ИЛИ. Однако чтение уравнений многоуровневых схем с элементами И-НЕ и ИЛИ-НЕ может оказаться довольно трудным. На [Рис. 2.33](#) показан пример многоуровневой схемы, функция которой не очевидна непосредственно из схемы. Путем перемещения инверсии можно преобразовать подобные схемы так, что инверсия сократится, и функция может стать более понятной. Построенные на принципах из [раздела 2.3.3](#), правила для перемещения инверсии таковы:

- ▶ Начинайте с выхода цепи и двигайтесь назад к входам.
- ▶ Переместите инверсию с общего выхода на входы так, чтобы вы могли читать выражение в терминах выхода (например, Y), а не инвертированного выхода \bar{Y} .
- ▶ Продвигаясь в обратном направлении, меняйте каждый элемент так, чтобы число инверсий оказалось четным и их можно было сократить. Если текущий элемент имеет входные отрицания, рисуйте предшествующий элемент с выходным отрицанием. Если текущий элемент не имеет входного отрицания, рисуйте предшествующий элемент без выходного отрицания.

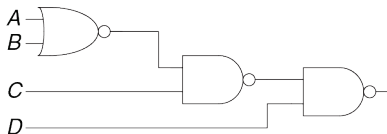


Рис. 2.33 Многоуровневая схема на элементах И-НЕ и ИЛИ-НЕ

Рис. 2.34 показывает, как перерисовать схему из **Рис. 2.33**, следуя изложенным правилам. Начинаем с выхода Y . Элемент И-НЕ имеет отрицание на выходе, которое мы хотим устранить. Мы переставляем выходное отрицание «назад», формируя элемент И с инверсными

выходами, показанный на **Рис. 2.34 (а)**. Двигаясь налево по схеме, мы замечаем, что самый правый элемент теперь имеет входное отрицание, которое может быть отброшено вместе с выходным отрицанием среднего элемента И-НЕ так, что инверсий в этом пути не останется, как показано на **Рис. 2.34 (b)**. Средний элемент не имеет входных инверсий, поэтому мы трансформируем самый левый элемент так, чтобы он не имел выходного отрицания, как показано на **Рис. 2.34 (c)**. Сейчас все отрицания в схеме убраны, за исключением входов, так что функция может быть прочитана в терминах элементов И и ИЛИ с действительными или комплементарными входами: $Y = \overline{A}BC + \overline{D}$.

Чтобы подчеркнуть этот последний пункт, на **Рис. 2.35** показана схема, логически эквивалентная схеме на **Рис. 2.34**. Функции внутренних соединений отмечены синим цветом. Поскольку последовательные отрицания могут быть отброшены, мы можем игнорировать инверсии на выходе среднего и на входе самого правого элементов, получив логически эквивалентную схему на **Рис. 2.35**.

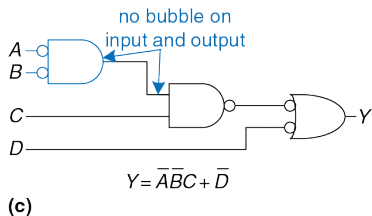
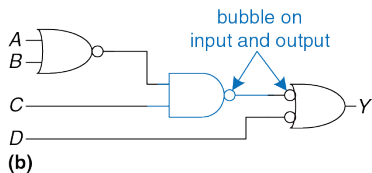
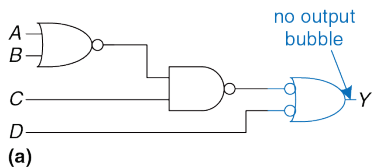
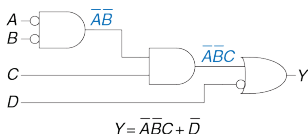


Рис. 2.34 Схема с удаленными инверсиями

**Рис. 2.35** Логически эквивалентная схема

Пример 2.8 ПЕРЕМЕЩЕНИЕ ИНВЕРСИИ В КМОП-ЛОГИКЕ

Большинство разработчиков думают в терминах элементов И и ИЛИ, но предположим, что вы хотели бы реализовать схему из **Рис. 2.36** в КМОП-логике, для которой предпочтительны элементы И-НЕ и ИЛИ-НЕ. Используйте перемещение инверсии, чтобы преобразовать схему в элементы И-НЕ, ИЛИ-НЕ и НЕ.

Решение: прямолинейное решение заключается в простой замене каждого элемента И на И-НЕ с инвертором, а каждого элемента ИЛИ – на И-НЕ с инвертором, как это показано на **Рис. 2.37**. Такая схема потребует 8 элементов. Заметьте, что инверторы изображены с отрицанием на входе, а не на выходе, чтобы подчеркнуть, что последовательное двойное отрицание не меняет логику работы схемы и может быть отброшено.

Обратите внимание, что отрицания могут быть добавлены на выход элемента и на вход следующего элемента без изменения функции, как показано на **Рис. 2.38 (а)**. Выходной элемент И преобразовывается в элемент И-НЕ и инвертор, как показано на **Рис. 2.38 (б)**. Это решение требует только пять элементов.

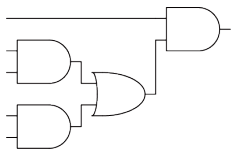


Рис. 2.36 Схема на элементах И и ИЛИ

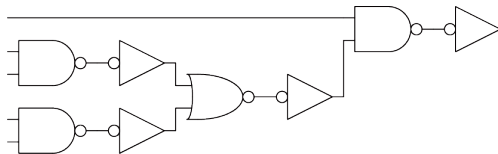
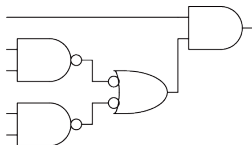
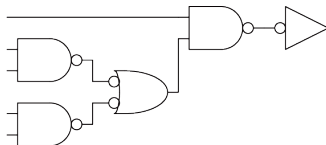


Рис. 2.37 Плохая схема на элементах И-НЕ и ИЛИ-НЕ



(a)



(b)

Рис. 2.38 Улучшенная схема на элементах И-НЕ и ИЛИ-НЕ

2.6 ЧТО ЗА X И Z?

Булева алгебра ограничена значениями 0 и 1. Однако реальные схемы могут также иметь недопустимое и плавающее состояния, представляемые символами X и Z соответственно.

2.6.1 Недопустимое значение: X

Символ X обозначает неизвестное логическое значение или недопустимое значение физического напряжения в соединении, не соответствующее уровням логических 0 и 1. Это обычно происходит, если к соединению подключены выходы других элементов схемы, выдающие значения 0 и 1 одновременно. На **Рис. 2.39** показан такой случай, когда выход Y подключен к элементам, имеющим на выходе ВЫСОКИЙ и НИЗКИЙ уровни.

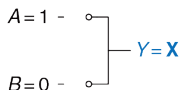


Рис. 2.39 Схема с недопустимым значением на выходе

Эта ситуация, называемая состязанием или конфликтом (contention), считается ошибкой, и её необходимо избегать. Реальное (физическое) напряжение на выходе с конфликтом может быть где-то между нулем и

напряжением питания, в зависимости от соотношения мощностей элементов, выдающих в цепь ВЫСОКОЕ и НИЗКОЕ напряжения. Часто, но не всегда, значение напряжения оказывается в «запрещенной» зоне. Состязание также может стать причиной повышенного потребления энергии конфликтующими элементами, в результате чего схема нагревается и может быть повреждена.

Значение X также иногда используется программами моделирования для обозначения неинициализированного значения. Например, если вы забыли определить входное значение, симулятор присвоит ему значение X для того, чтобы предупредить вас о проблеме.

Как уже упоминалось в [разделе 2.4](#), разработчики цифровых схем также используют символ X для обозначения в таблицах истинности безразличных переменных, от которых не зависит состояние выходов. Не путайте эти два смысла. Когда X появляется в таблицах истинности, он показывает, что значение этой переменной может быть и нулем, и единицей. Когда X появляется в схеме, это означает, что цепь имеет неизвестное или запрещенное значение.

2.6.2 Третье состояние: Z

Символ Z указывает, что напряжение в цепи не определяется ни источником ВЫСОКОГО, ни источником НИЗКОГО напряжения. Говорят, что такая цепь отключена, находится в состоянии высокого импеданса или в третьем состоянии. Типично неправильное представление – это что неподключенная, или плавающая цепь имеет значение логического 0. В реальности логическое состояние неподключенной цепи может быть как 0, так и 1, а напряжение на ней может принять некое промежуточное значение в зависимости от истории изменения состояния системы. Неподключенная цепь не обязательно означает наличие ошибки в схеме. Например, какой-нибудь другой элемент схемы может задать цепи допустимый логический уровень именно в тот момент, когда эта цепь влияет на работу схемы.

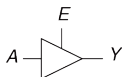
Один из распространенных способов получить неопределенное значение – это забыть подключить вход схемы к источнику напряжения логического уровня или предположить, что неподключенный вход – то же самое, что вход со значением 0. Эта ошибка может привести к тому, что поведение цепи будет хаотичным, так как неопределенные значения на входе могут случайно меняться из 0 в 1. Действительно, касания схемы может быть достаточно, чтобы привести к изменению из-за слабого статического электричества тела. Мы видели схему, которая

корректно работала, только до тех пор, пока студент держал палец на микросхеме.

Буфер с тремя состояниями, показанный на **Рис. 2.40**, имеет три возможных выходных значения: ВЫСОКОЕ (1), НИЗКОЕ (0) и отключенное или плавающее (Z) состояние (прим. переводчика: именно поэтому плавающее состояние называют третьим). Буфер с тремя состояниями имеет вход A, выход Y и сигнал управления E. Когда сигнал разрешения (управления) имеет значение ИСТИНА, буфер с тремя состояниями работает как простой буфер, передавая входное значение на выход. Когда сигнал управления имеет значение ЛОЖЬ, выход буфера переключается в третье состояние и становится плавающим (Z). Буфер с тремя состояниями на **Рис. 2.40** имеет активный высокий уровень. Это значит, что когда сигнал разрешения ВЫСОКИЙ (1), передача разрешена.

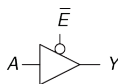
На **Рис. 2.41** показан Буфер с тремя состояниями с активным низким уровнем. Когда сигнал управления НИЗКИЙ (0), передача разрешена. Мы видим, что сигнал имеет активный низкий уровень из-за отрицания, поставленного на его входной цепи. Мы часто обозначаем вход с активным низким уровнем, рисуя черточку (символ отрицания) над его именем (\bar{E}), или добавляя букву "b" или "bar" после имени, E_b или $Ebar$.

Tristate Buffer



E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

Рис. 2.40 Буфер с тремя состояниями



\bar{E}	A	Y
0	0	0
0	1	1
1	0	Z
1	1	Z

Рис. 2.41 Буфер с тремя состояниями с активным низким уровнем

Буферы с третьим состоянием обычно используются в шинах, соединяющих несколько микросхем. Например, микропроцессор, видеоконтроллер и Ethernet-контроллер могут нуждаться во взаимодействии с подсистемой памяти в персональном компьютере. Каждая микросхема может подключаться к общей шине памяти, используя буферы с третьим состоянием, как показано на [Рис. 2.42](#). При этом только одна микросхема имеет право выставить свой сигнал разрешения, чтобы выдать значение на шину. Выходы других микросхем должны находиться в третьем состоянии, чтобы не стать причиной коллизии с микросхемой, осуществляющей обмен данными с

памятью. Однако, любая микросхема может читать информацию с общей шины в любое время. Такие шины на основе буферов с тремя состояниями когда-то были очень распространенными. Однако, в современных компьютерах высочайшие скорости возможны только при соединении микросхем друг с другом напрямую (point-to-point), а не с помощью общей шины.

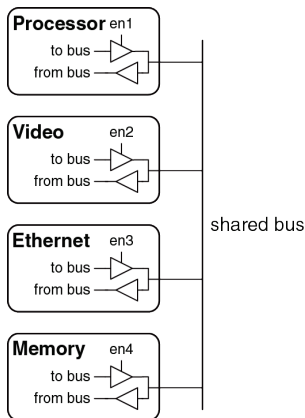


Рис. 2.42 Шина с третьим состоянием, соединяющая несколько микросхем

2.7 КАРТЫ КАРНО

После того, как Вы осуществите несколько преобразований по минимизации булевых уравнений, используя булеву алгебру, Вы поймете, что без соблюдения должной аккуратности, иногда можно получить решение, совершенно отличное от требуемого упрощенного уравнения. Карты Карно представляют собой наглядный метод для упрощения булевых уравнений. Они были изобретены в 1953-м году Морисом Карно, телекоммуникационным инженером из фирмы Bell Labs. Карты Карно очень удобны в случаях, когда уравнение содержит до четырёх переменных. Но, что более важно, они дают понимание сути при манипулировании логическими выражениями.

Морис Карно родился в 1924 году. Получил степень бакалавра по физике в Городском колледже Нью-Йорка в 1948 году, а в 1952 получил степень доктора философии по физике (Ph.D., аналог степени кандидата наук) в Йельском университете.

С 1952 по 1993 годы работал в Bell Labs и IBM. С 1980 по 1999 год являлся профессором информатики в Политехническом университете Нью-Йорка.

Как мы помним, логическая минимизация осуществляется путем склейки термов. Два терма, включающие в себя импликанту P и два логических значения некоторой переменной A , объединяются, при этом

переменная A исключается. Карты Карно позволяют легко находить термы, которые можно склеить, располагая их в виде таблицы.

На **Рис. 2.43** показана таблица истинности и карта Карно для функции трех переменных. Верхняя строка дает 4 возможных значения для переменных A и B . Левая колонка дает 2 возможных значения переменной C . Каждая клетка карты Карно соответствует строке таблицы истинности и содержит значение функции Y из этой строки. Например, верхняя левая клетка соответствует первой строке таблицы истинности и показывает, что значение функции Y будет равно 1, когда $ABC=000$. Как и каждая строка в таблице истинности, каждая клетка карты Карно представляет собой отдельный минтерм. Для лучшего понимания, на **Рис. 2.43 (с)** показаны минтермы, соответствующие каждой клетке карты Карно.

Каждая клетка, или минтерм, отличается от соседней изменением только одной переменной. Это значит, что соседние клетки различаются только в значении одного литерала, значение которого «истинно» в одной клетке и «ложно» в соседней. Например, клетки, представляющие минтермы $\bar{A}\bar{B}\bar{C}$ и $\bar{A}\bar{B}C$ – соседние и различаются только в переменной C . Вы, наверное, также отметили, что переменные A и B комбинируются в верхней строке в особом порядке: 00, 01, 11, 10. Этот порядок называется *кодом Грея* (Gray code). В отличие от битового порядка по возрастанию величины (00, 01, 10, 11), в коде Грея

соседние записи отличаются только на один разряд. Например, 01 : 11 отличается только изменением A с 0 на 1, тогда как 01 : 10 требует изменения A из 1 в 0 и B из 0 в 1. Таким образом, обычный последовательный побитный порядок не дает требуемого нам свойства соседних ячеек, который должны различаться только в одной переменной.

Код Грея был запатентован Фрэнком Греем, исследователем из Bell Labs, в 1953 году (патент США номер 2,632,058). Этот код особенно полезен для электромеханических преобразователей (например, датчиков угла поворота – прим. переводчика), так как он позволяет избавиться от ложных срабатываний. Код Грея может быть любой разрядности. Например, трехбитный код Грея выглядит так: 000, 001, 011, 010, 110, 111, 101, 100.

Льюис Кэрролл опубликовал похожую загадку в журнале Vanity Fair в 1879 году. «Правила просты. Даны два слова одинаковой длины. Нужно соединить их цепочкой слов, в которой два соседних слова отличаются лишь одной буквой» – написал он.

Например, слово SHIP можно превратить в слово DOCK так: **SHIP**, SLIP, SLOP, SLOT, SOOT, LOOT, LOOK, LOCK, **DOCK**. Можете ли вы найти более короткую цепочку?

Карты Карно так же «закольцованы». Клетка с самого правого края таблицы является соседней с самой левой, так как они отличаются

только в одной переменной (A). Можно свернуть карту в цилиндр, соединив края, и даже в этом случае соседние клетки также будут отличаться только в одной переменной.

2.7.1 Думайте об овалах

На карте Карно на **Рис. 2.43** содержится только две единицы, что соответствует числу минтермов в уравнении ($\bar{A} \bar{B} \bar{C}$ и $\bar{A} \bar{B} C$). Чтение минтермов из карт Карно в точности соответствует чтению дизъюнктивной нормальной формы (ДНФ) из таблицы истинности.

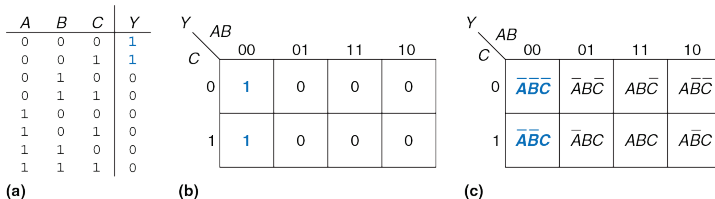


Рис. 2.43 Функция трех переменных: таблица истинности (a), карта Карно (b), карта Карно с минтермами (c)

Как и раньше, мы могли бы использовать булеву алгебру для минимизации:

$$Y = \bar{A} \bar{B} \bar{C} + \bar{A} \bar{B} C + \bar{A} \bar{B} (\bar{C} + C) = \bar{A} \bar{B} \quad (2.7)$$

	AB			
	00	01	11	10
C				
0	1	0	0	0
1	1	0	0	0

Рис. 2.44 Минимизация при помощи карты Карно

Карты Карно помогают нам делать это упрощение графически, обводя единицы в соседних клетках овалами, как показано на **Рис. 2.44**. Для каждого овала мы пишем соответствующую ему импликанту. Вспомните из **раздела 2.2**, что импликанта является произведением одного или нескольких литералов. Переменные, для которых прямая и комплементарная формы попадают в один овал, исключаются из импликанты. В нашем случае обе формы переменной C попадают в овал, так что мы не включаем ее в импликанту. Другими словами, $Y = \text{ИСТИНА}$ когда $A = B = 0$ вне зависимости от C . Так что импликантой

будет $\bar{A}\bar{B}$: карта Карно дает тот же самый ответ, какой мы получили, используя булеву алгебру.

2.7.2 Логическая минимизация на картах Карно

Карты Карно обеспечивают простой визуальный способ минимизации логических выражений. Просто обведите все прямоугольные блоки с единицами на карте, используя наименьшее возможное число овалов. Каждый овал должен быть максимально большим. Затем прочитайте все импликанты, которые обведены.

Напомним, что формально уравнения булевой алгебры являются минимальными, только когда записаны как сумма наименьшего числа первичных импликант. Каждый овал на карте Карно представляет собой импликанту. Максимально возможный овал является первичной импликантой.

Например, на карте Карно на [Рис. 2.44](#) $\bar{A}\bar{B}\bar{C}$ и $\bar{A}\bar{B}C$ импликанты, но не первичные. На этой карте только $\bar{A}\bar{B}$ является первичной импликантой. Правила для нахождения минимального уравнения из карт Карно следующие:

- Использовать меньше всего овалов, необходимых для покрытия всех 1;

- ▶ Все клетки в каждом овале обязаны содержать 1;
- ▶ Каждый овал должен охватывать блок, число клеток которого в каждом направлении равно степени двойки (то есть 1, 2 или 4);
- ▶ Каждый овал должен настолько большим, насколько это возможно;
- ▶ Овал может связывать края карты Карно;
- ▶ Единица на карте Карно может быть обведена сколько угодно раз, если это позволяет уменьшить число овалов, которые будут использоваться.

Пример 2.9 МИНИМИЗАЦИЯ ФУНКЦИИ ТРЕХ ПЕРЕМЕННЫХ ПРИ ПОМОЩИ КАРТЫ КАРНО

Предположим, у нас есть функция $Y = F(A, B, C)$ с картой Карно, показанной на **Рис. 2.45**. Упростим это выражение, используя карту Карно.

Решение: Обведем единицы на карте Карно, используя наименьшее возможное количество овалов, как показано на **Рис. 2.46**. Каждый овал на карте Карно представляет собой первичную импликанту, а его размер кратен степени двойки (2×1 и 2×2).

Мы сформируем первичную импликанту для каждого выделенного овала, выписывая только те переменные, которые появляются в нем только в прямой или в комплементарной формах. Например, овал размером 2×1 включает в себя прямую и комплементарную формы переменной B , так что мы не включаем

B в первичную импликанту. Однако, в этом овале есть только прямая форма переменной A (\bar{A}) и комплементарная форма переменной C (\bar{C}), так что мы включаем эти переменные в первичную импликанту A (\bar{A}). Подобным же образом овал размером 2×2 покрывает все клетки, где $B = 0$, так что первичная импликанта будет \bar{B} .

Обратите внимание, что правая верхняя клетка (минтерм) используется дважды, чтобы сделать овалы первичных импликант как можно большими. Как мы видели в булевой алгебре, это эквивалентно совместному использованию минтерма для уменьшения размера импликанты. Также обратите внимание на то, что овал, покрывающий четыре клетки, оборачивается через края карты Карно.

Y

AB

C

	00	01	11	10
0	1	0	1	1
1	1	0	0	1

Рис. 2.45 Карта Карно для примера 2.9

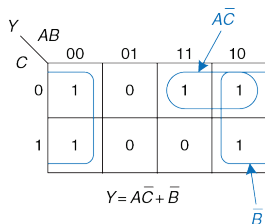
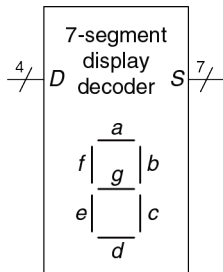
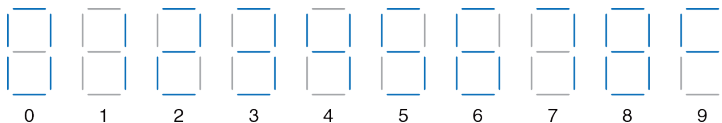


Рис. 2.46 Решение примера 2.9

Пример 2.10 ДЕШИФРАТОР СЕМИСЕГМЕНТНОГО ИНДИКАТОРА

Дешифратор семисегментного индикатора получает на вход четырехбитные данные $D[3:0]$ и формирует семь выходов для управления светодиодами для показа цифр от 0 до 9. Семь выходов часто называют сегментами от a до g , или S_a – S_g , как показано на **Рис. 2.47**. Сами цифры показаны на **Рис. 2.48**. Составим таблицу истинности для выходов и используем карты Карно для нахождения логического уравнения для выходов S_a и S_b . При этом предположим, что запрещенные входные значения (10–15) ничего не выводят на индикатор.

**Рис. 2.47** Семисегментный индикатор

**Рис. 2.48** Цифры на семисегментном индикаторе

Решение: Таблица истинности дана в **Табл. 2.1** Например, вход 0000 должен включать все сегменты, за исключением S_g .

Табл. 2.6 Таблица истинности дешифратора семисегментного индикатора

$D_{3:0}$	S_a	S_b	S_c	S_d	S_e	S_f	S_g
0000	1	1	1	1	1	1	0
0001	0	1	1	0	0	0	0
0010	1	1	0	1	1	0	1
0011	1	1	1	1	0	0	1
0100	0	1	1	0	0	1	1
0101	1	0	1	1	0	1	1
0110	1	0	1	1	1	1	1
0111	1	1	1	0	0	0	0
1000	1	1	1	1	1	1	1
1001	1	1	1	0	0	1	1
Прочие	0	0	0	0	0	0	0

Каждый из семи выходов является независимой функцией от четырех переменных. Карты Карно для выходов S_a и S_b показаны на Рис. 2.49. Помните, что соседние клетки могут отличаться только одной переменной, так что мы промаркируем строки и столбцы в коде Грея: 00, 01, 11, 10. Будьте осторожны и помните этот порядок, когда будете вписывать значения выходов в клетки.

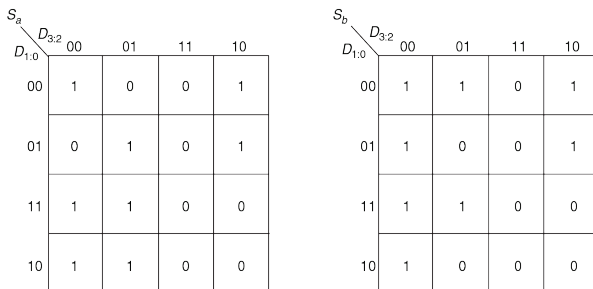


Рис. 2.49 Карты Карно для S_a и S_b

Затем обведем первичные импликаны. При этом используем минимально необходимое количество овалов для покрытия всех единиц. Овалы могут связывать края (вертикальные и горизонтальные), а каждая единица может быть выделена несколько раз. На Рис. 2.50 показаны первичные импликаны и упрощенные логические уравнения.

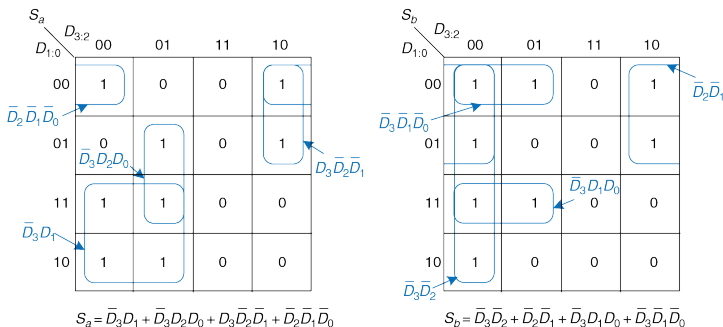


Рис. 2.50 Решение упражнения 2.10

Заметьте, что минимальный набор первичных импликант – не единственно возможный. Например, запись 0000 на карте Карно для S_a может быть выделена вместе с записью 1000, получая минтерм $\bar{D}_2 \bar{D}_1 \bar{D}_0$. Но вместо этого овал может включать в себя запись 0010, получая минтерм $\bar{D}_3 \bar{D}_2 \bar{D}_0$, как показано пунктирной линией на Рис. 2.51.

Рис. 2.52 иллюстрирует распространенную ошибку, когда не первичная импликанта выбирается для покрытия 1 в левом верхнем углу. Этот минтерм $\bar{D}_3 \bar{D}_2 \bar{D}_1 \bar{D}_0$ дает дизъюнкцию конъюнкций (сумму произведений), которая не минимизирована. Его можно было бы скомбинировать с любым из двух соседних

минтермов для получения овала большего размера, как было сделано на предыдущих двух рисунках.

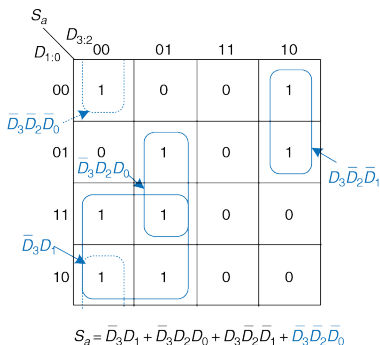


Рис. 2.51 Альтернативная карта Карно для S_a , использующая другой набор первичных импликант

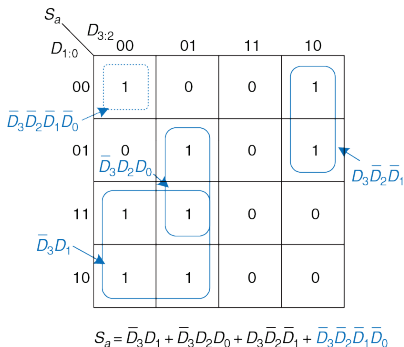


Рис. 2.52 Карта Карно для S_a , использующая некорректную импликанту

2.7.3 Безразличные переменные

Вспомните, что безразличные переменные в таблице истинности были введены в [разделе 2.4](#) для уменьшения числа ее строк в тех случаях, когда соответствующие переменные не влияют на выход. Они обозначаются символом X , который означает, что значение входной переменной может быть или 0, или 1.

Не только входы, но и выходы могут быть безразличными, если состояние выхода не важно или соответствующая комбинация входов никогда не возникает. Такие выходы могут трактоваться или как 0, или как 1, в зависимости от того, как решит разработчик.

В картах Карно безразличные переменные позволяют провести еще большую логическую минимизацию. Их можно включать в овалы, если это помогает покрыть единицы или меньшим количеством овалом, или овалами, большими по размеру, но их можно и не покрывать, если это не помогает минимизации.

Пример 2.11 ДЕШИФРАТОР СЕМИСЕГМЕНТНОГО ИНДИКАТОРА С БЕЗРАЗЛИЧНЫМИ ПЕРЕМЕННЫМИ

Повторим пример 2.10 для случая, когда нас не интересуют значения выходов при запрещенных входных значениях от 10 до 15.

Решение: Карта Карно с безразличными элементами, отмеченными как «X», представлена на **Рис. 2.53**. Поскольку такие элементы могут быть равны как 0, так и 1, мы используем их там, где это поможет покрыть единицы или меньшим количеством овалов, или овалами, большими по размеру. Обведенные значения X трактуются как 1, не обведенные – как 0. Посмотрите, как для сегмента S_a можно выделить овал размером 2×2 , объединяющий все четыре угла. Используйте клетки с безразличными значениями для упрощения логики.

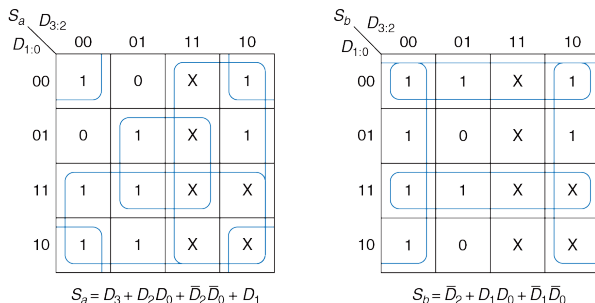


Рис. 2.53 Карта Карно с безразличными переменными

2.7.4 Подводя итоги

Булева алгебра и карты Карно – два метода логического упрощения. В конечном счете, целью является нахождение наименее затратного метода реализации конкретной логической функции.

В современной инженерной практике компьютерные программы, называемые синтезаторами логики (logic synthesizers), проводят упрощение схем по описанию логических функций, как мы увидим в [главе 4](#). Для больших задач программы логического синтеза намного эффективнее людей. Для маленьких же задач человек с некоторым опытом может найти хорошее решение «на глаз». Никто из авторов книги, тем не менее, никогда не использовал карты Карно в реальной жизни для решения практических задач. Но понимание принципов, лежащих в основе карт Карно, крайне важно. Кроме того, знание карт Карно часто спрашивают на собеседованиях!