Функции алгебры логики

Цифровые устройства — устройства, предназначенные для преобразования цифровой информации и работающие с цифровыми (логическими) сигналами. Цифровой сигнал — такое представление информации, при котором различают только два его возможных значения: «логический 0» и «логическая 1». Эти сигналы обычно представляются на практике в виде напряжений низкого и высокого уровней. Если логическому нулю соответствует низкий уровень напряжения, а логической единице — высокий, такое представление информации называется **положительной логикой**. Применяется также и **отрицательная логика**, при которой логическому нулю соответствует высокий уровень напряжения, а логической единице — низкий. Тот факт, что любой сигнал в цифровой схемотехнике может принимать только одно из двух возможных значений, позволяет применить для описания и расчета цифровых устройств **алгебру логики**.

Основным понятием алгебры логики является высказывание. **Высказывание —** это любое утверждение, содержание которого можно определить как истинное или ложное. С помощью логических связок из высказываний можно составлять сложные высказывания, которые тоже будут истинны или ложны.

Функцией алгебры логики *n* переменных называется любая функция *n* переменных $f(x_1, x_2, \ldots x_n)$, аргументы которой $x_1, x_2, \ldots x_n$ принимают 2 значения 0 и 1, и сама функция принимает одно из двух значений 0 или 1.

В классической математике для задания функции обычно используются два способа: аналитический (запись формулой) и табличный (таблицами значения функций). Подобными же способами могут задаваться логические функции, однако табличный способ задания функций алгебры логики предпочтительнее по следующим причинам. Функции и аргументы в алгебре логики определены на множестве {0, 1} и, следовательно, могут принимать только два значения. Все возможные комбинации значений аргументов называются наборами.

Ограниченность значений аргументов приводит к тому, что и область определения любой функции тоже ограничена. Область определения, т.е. число наборов, для функции *n* аргументов составляет 2^n. На каждом наборе аргументов функция может принимать только два значения, следовательно, для *n* аргументов можно получить 2^{2^n} различных функций. Таким образом, любую функцию алгебры логики можно задать, перечислив все ее возможные значения на всех наборах аргументов, т.е. таблично. Эта таблица называется **таблицей истинности**. В левой части таблиц обычно перечисляют все наборы значений аргументов функции, а в правой части — значения функции на этих наборах. Число строк в таблице равно числу наборов аргументов. Для удобства слева добавляют еще один столбец с номером набора.

Например, с помощью таблиц истинности функция алгебры логики двух аргументов $f = f1(x_1, x_2)$ будет определена на четырех наборах, а функция трех аргументов $f = f2(x_1, x_2, x_3)$ – на восьми наборах.

N	x_1	x_2	f_1
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

N	x_1	x_2	x_3	f_2
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Так как число таких сочетаний конечно, таблица истинности позволяет определять значение функции для любых значений аргументов (в отличии от таблиц математических функций, которые позволяют задавать значения функции не для всех, а лишь для некоторых значений аргументов).

Логические операции

Из множества логических функций выделяют некоторые, так называемые элементарные функции или **операции**, с помощью которых можно записать аналитически любую другую функцию алгебры логики.

Отрицание

Когда логическая переменная ***е*** равна 0, то ***п*** равно 1, и наоборот. Таблица истинности операции отрицания приведена ниже (табл. 1.2):

x	$\bar{x}$
0	1
1	0

Операция **дизъюнкция** (иначе называемая операцией **ИЛИ**, реже **логическим сложением**) определена для двух и более высказываний. Обозначается операция значком **v** или знаком +, например, X_1 **v** X_2. Результат операции **дизъюнкции** истинен, когда хотя бы одно из вхо- дящих в него высказываний истинно. Если все аргументы ложны, то и результат операции будет ложен (табл. 1.3).

x_1	x_2	$x_1 \vee x_2$
0	0	0
0	1	1
1	0	1
1	1	1

Операция **конъюнкция** (иначе называемая операцией **И**, реже **логическим умножением**) также определена для двух и более выска-зываний. Обозначается операция значком **&** или значком умножения (точкой) или совсем опускается, например, X_1 **&** X_2 или $X_1 X_2$. Резуль-тат операции **конъюнкции** истинен, когда все входящие в него выска-зывания истинны, и ложен в противном случае (табл. 1.4).

x_1	x_2	$x_1 \& x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Операции **дизъюнкция**, **конъюнкция** и **отрицание** чаще всего используются в алгебре логики для записи и упрощения функций ал-гебры логики. Но в цифровой схемотехнике удобнее использовать ло-гические элементы, реализующие другие логические операции. В первую очередь это операции **И-НЕ** и **ИЛИ-НЕ**.

Операция **стрелка Пирса** (иначе называемая операцией **ИЛИ-НЕ**) определена для двух и более высказываний. Обозначается операция значком ↓ , например, $X_1 \downarrow X_2$. Результат операции **стрелка Пирса** ло- жен, когда хотя бы одно из входящих в него высказываний истинно. Если все аргументы ложны, то результат операции будет истинен (табл. 1.5)

x_1	x_2	$x_1 \downarrow x_2$
0	0	1
0	1	0
1	0	0
1	1	0

Операция **штрих Шеффера** (иначе называемая операцией **И-НЕ**) также определена для двух и более высказываний. Обозначается опе-рация значком $\mid$, например, $X_1 \mid X_2$. Результат операции **штрих Шеффера** ложен, когда все входящие в него высказывания истинны и истинен в противном случае (табл. 1.6)

x_1	x_2	$x_1 \mid x_2$
0	0	1
0	1	1
1	0	1
1	1	0

Логические элементы **И-НЕ** (реализующие операцию штрих Шеффера) чаще всего используются в цифровой схемотехнике для создания про-стых схем.

Операция ***исключающее ИЛИ***

Операция **исключающее ИЛИ** определена для двух и более вы-сказываний. Обозначается операция значком ^ , например, ***X1 ^ X2***. Ре-зультат операции **исключающее ИЛИ** истинен, когда истинно одно и только одно из входящих в него высказываний, и ложен в противном случае. Данную операцию еще называют операцией **один и только один** (табл. 1.7).

x_1	x_2	$x_1 \wedge x_2$
0	0	0
0	1	1
1	0	1
1	1	0

Операция **эквиваленция** определена для двух и более высказы-ваний. Обозначается операция значком ~ , например, $X_1 \sim X_2$. Резуль-тат операции **эквиваленция** истинен, когда все входящие в нее вы-сказывания истинны или все входящие в нее высказывания ложны, и ложен в противном случае(табл. 19.).

x_1	x_2	$x_1 \sim x_2$
0	0	1
0	1	0
1	0	0
1	1	1

Операция **импликация** определена только для двух высказыва-ний. Обозначается операция значком → , например, $X_1 \rightarrow X_2$ (читается X_1 имплицирует X_2). Результат операции **импликация** ложен, когда первое входящее в него истинно, а второе ложно, и истинен в против-ном случае. Для этой операции несправедлив коммутативный закон (табл. 1.10).

x_1	x_2	$x_1 \to x_2$	$x_2 \to x_1$
0	0	1	1
0	1	1	0
1	0	0	1
1	1	1	1

Число функций для n аргументов ограничено, например, в случае *n=2* получается всего 16 функций, из которых лишь десять суще-ственно зависят от обоих аргументов *(x1,x2)* (табл. 1.13)*:*

N	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	0	1	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
2	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
3	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Можно использовать определенные выше операции для построения других, более сложных функций.

Одну и ту же функцию можно записать аналитически различными способами, выражая ее через определенные выше элементарные функции. Но для записи функций алгебры логики не обязательно ис-пользовать все вышеприведенные операции. Можно доказать, что ограниченный набор некоторых операций (или даже всего одной) поз-воляет записывать любую, сколь угодно сложную функцию алгебры логики. Такой набор операций называется **функционально полной системой** или **базисом**. Базисами являются, например, наборы опе-раций:

- дизъюнкция, конъюнкция и отрицание;
- импликация и константа 0;
- запрет и константа 1;
- штрих Шеффера;
- стрелка Пирса.

Базис, состоящий из операций дизъюнкция, конъюнкция и отрица-ние называют основным, именно для него разработаны методы мини-мизации функций.

Если логическая функция представлена дизъюнкцией, конъюнкцией и инверсией, то такая форма представления называется ***нормальной***.

Элементарной конъюнкцией, называется конъюнкция, состоящая из нескольких аргументов, причем каждый аргумент входит в нее не более одного раза

Элементарной дизъюнкцией, называется дизъюнкция, состоящая из нескольких аргументов, причем каждый аргумент входит в нее не более одного раза

Дизъюнктивная нормальная форма (ДНФ) содержит элементарные конъюнкции, связанные между собой операцией дизъюнкции.

Конъюнктивная нормальная форма (КНФ) содержит элементарные дизъюнкции, связанные между собой операцией конъюнкции.
Одну и ту же логическую функцию можно представить разными ДНФ и КНФ.

Терм - набор элементарных конъюнктов в ДНФ или дизъюнктов в КНФ

Совершенная дизъюнктивная нормальная форма (СДНФ) отвечает следующим требованиям:
1) в ней нет двух одинаковых элементарных конъюнкций;
2) ни одна элементарная конъюнкция не содержит двух одинаковых переменных;
3) ни одна элементарная конъюнкция не содержит переменную вместе с ее инверсией;

4) все конъюнкции имеют один и тот же ранг.

(***Рангом*** элементарной дизъюнкции (конъюнкции) называется количество аргументов, входящих в нее.)

Аналогичным требованиям подчиняется и ***совершенная конъюнктивная нормальная форма*** (СКНФ).

5.5.1. Алгоритм образования СДНФ по таблице истинности

1. Выделить в таблице истинности все наборы переменных, на которых функция принимает единичные значения.
2. Для каждого выбранного набора записать элементарные конъюнкции, содержащие без инверсии переменные, принимающие в соответствующем наборе значение 1 и с инверсией — переменные, принимающие значение 0.
3. Соединить элементарные конъюнкции знаком дизъюнкции.

5.5.2. Алгоритм образования СКНФ по таблице истинности

1. Выделить в таблице истинности все наборы переменных, на которых функция принимает нулевые значения.
2. Для каждого выбранного набора записать элементарные дизъюнкции, содержащие без инверсии переменные, принимающие в соответствующем наборе значение 0 и с инверсией — переменные, принимающие значение 1.
3. Соединить элементарные дизъюнкции знаком конъюнкции.

Минимальной ДНФ (МДНФ) называется ДНФ, содержащая наименьшее операций дизъюнкция и конъюнкция по сравнению с дру-гими ДНФ данной функции. У функции может быть несколько МДНФ.

Минимальной КНФ (МКНФ) называется ДНФ, содержащая наименьшее операций дизъюнкция и конъюнкция по сравнению с дру-гими ДНФ данной функции. У функции может быть несколько МДНФ.

Свойства операций алгебры логики

Для упрощения формул необходимо прежде всего рассмотреть свойства основных операций.

Чаще всего для записи и преобразования формул алгебры логики используют основной базис, поэтому рассмотрим более подробно свойства (или законы) операций дизъюнкция, конъюнкция и отрицание

*1.4.1. Свойства операции **отрицание***

a) Закон двойного отрицания:

$$x \equiv \overline{\overline{x}}$$

*1.4.2. Свойства операций **конъюнкция** и **ди***

a) Законы нулевого множества:

$$x \vee 0 \equiv x$$

$$x \& 0 \equiv 0$$

b) Законы универсального множества:

$$x \vee 1 \equiv 1$$

$$x \& 1 \equiv x$$

c) Законы идемпотентности:

$$x \vee x \equiv x$$

$$x \& x \equiv x$$

Обобщая для случая *n* аргументов, получим:

$$x \vee x \vee \ldots \vee x \equiv x$$

$$x \& x \& \ldots \& x \equiv x$$

d) Закон исключенного третьего:

$$x \vee \overline{x} \equiv 1$$

e) Закон логического противоречия:

$$x \& \overline{x} = 0$$

f) Коммутативные законы:

$$x_1 \vee x_2 \equiv x_2 \vee x_1$$

$$x_1 \& x_2 \equiv x_2 \& x_1$$

g) Ассоциативные законы:

$$x_1 \vee (x_2 \vee x_3) \equiv (x_1 \vee x_2) \vee x_3 \equiv x_1 \vee x_2 \vee x_3$$

$$x_1(x_2 x_3) \equiv (x_1 x_2) x_3 \equiv x_1 x_2 x_3$$

h) Дистрибутивные законы

конъюнкции относительно дизъюнкции:

$$x_1(x_2 \vee x_3) \equiv x_1 x_2 \vee x_1 x_3$$

дизъюнкции относительно конъюнкции:

$$x_1 \vee x_2 x_3 \equiv (x_1 \vee x_2)(x_1 \vee x_3)$$

Не будем рассматривать все

Законы де Моргана:

Операции конъюнкции и дизъюнкции связаны между собой:

$$\overline{x_1 \& x_2} \equiv \overline{x_1} \vee \overline{x_2}$$

или

$$\overline{x_1 \vee x_2} \equiv \overline{x_1} \& \overline{x_2}$$

Этот закон легко обобщить для случая *n* аргументов:

$$\overline{x_1 x_2 \ldots x_n} \equiv \overline{x}_1 \vee \overline{x}_2 \vee \ldots \vee \overline{x}_n$$

$$\overline{x_1 \vee x_2 \vee \ldots \vee x_n} \equiv \overline{x}_1 \& \overline{x}_2 \& \ldots \& \overline{x}_n$$

Правила склеивания:

$$A\,x \vee A\,\overline{x} \equiv A$$

$$(A \vee x)(A \vee \overline{x}) \equiv A$$

Здесь А - выражение, состоящее из нескольких логических переменных.

МИНИМИЗАЦИЯ ФУНКЦИЙ АЛГЕБРЫ ЛОГИКИ

Существует несколько методов минимизации функций, но все онипозволяют найти либо МДНФ, либо МКНФ. В общем случае ни МДНФ, ни МКНФ не являются самыми короткими аналитическими за-писями в базисе, состоящем из операций дизъюнкция, конъюнкция и отрицание. ***Абсолютно минимальным***

представлением функции алгебры логики будем называть такое логическое выражение, короче которого в данном базисе не существует.

Но методов нахождения абсолютно минимального представления не существует, поэтому на практике обычно находят МДНФ (или МКНФ), а затем, если необходимо упростить эту запись, проводят раз-личные вынесения за скобки на основании дистрибутивных законов и выбирают наиболее простое выражение из этих скобочных форм.

3.1. Метод Квайна

3.1.1. Алгоритм метода Квайна

Метод Квайна имеет строгий алгоритм и позволяет найти все МДНФ функции.

3.2. Метод карт Карно

3.2.1. Построение карт Карно

При минимизации функций алгебры логики в первую очередь ис-пользуется правило склеивания

Куб Карно́ (**ка́рта Карно**, **диагра́мма Карно**) — графический способ представления логических функций с целью наглядной и удобной их минимизации.

Является одним из эквивалентных способов описания или задания логический функций наряду с таблицей истинности. Преобразование представления логической функции, заданной в виде карты Карно в таблицу истинности и обратное преобразование элементарно по простоте.

Основным методом минимизации логических функций, представленных в виде СДНФ или СКНФ, является операция попарного неполного склеивания и элементарного поглощения. Операция попарного склеивания осуществляется между двумя термами (членами), содержащими одинаковые переменные, вхождения которых (прямые и инверсные) совпадают для всех переменных, кроме одной. В этом случае все переменные, кроме одной, можно вынести за скобки, а

оставшиеся в скобках прямое и инверсное вхождение одной переменной подвергнуть склейке. Например:

$$\overline{X}_1 X_2 X_3 X_4 \vee \overline{X}_1 X_2 \overline{X}_3 X_4 = \overline{X}_1 X_2 X_4 (X_3 \vee \overline{X}_3) = \overline{X}_1 X_2 X_4.$$

Возможность поглощения следует из очевидных равенств

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

Таким образом, главной задачей при минимизации СДНФ и СКНФ является поиск термов, пригодных к склейке с последующим поглощением, что для больших форм может оказаться достаточно сложной задачей. Карты Карно предоставляют наглядный способ отыскания таких термов.

Как известно, булевы функции N переменных, представленные в виде СДНФ или СКНФ могут иметь в своём составе 2N различных термов. Все эти члены составляют некоторую структуру, топологически эквивалентную N–мерному кубу, причём любые два терма, соединённые ребром, пригодны для склейки и поглощения.

Для упрощения работы с булевыми функциями большого числа переменных был предложен следующий удобный приём. Куб, представляющий собой структуру термов, разворачивается на плоскость. При этом следует помнить, что порядок кодов термов в таблице (00 01 11 10) не соответствует порядку следования двоичных чисел, а клетки, находящиеся в крайних столбцах таблицы, соседствуют между собой.

Порядок работы с картой Карно

Исходной информацией для работы с картой Карно является таблица истинности минимизируемой функции. Таблица истинности содержит полную информацию о логической функции, задавая её значения на всех возможных 2^N наборах входных переменных $X_1 \ldots X_N$. Карта Карно также содержит 2^N клеток, каждая из которых ассоциируется с уникальным набором входных переменных $X_1 \ldots X_N$. Таким образом, между таблицей истинности и картой Карно имеется взаимно однозначное соответствие, и карту Карно можно считать соответствующим образом отформатированной таблицей истинности.

а

X_1	X_2	X_3	X_4	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

б

X_1X_2 \ X_3X_4	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	0	1	1	0
10	1	0	0	1

в

X_1X_2 \ X_3X_4	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	0	1	1	0
10	1	0	0	1

г

X_1X_2 \ X_3X_4	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	0	1	1	0
10	1	0	0	1

- Склейку клеток карты Карно можно осуществлять по единицам (если необходимо получить ДНФ) или по нулям (если требуется КНФ).
- Склеивать можно только прямоугольные области с числом единиц (нулей) 2^n, где n — целое число, при этом рекомендуется брать максимальное из возможных значений n. В некоторых ситуациях в раскладке образуется единица или ноль, которую невозможно склеить с какой-либо областью. В этом случае единица склеивается «сама с собой». Для карт Карно с числом переменных более четырёх могут получаться более сложные области, о чём будет сказано в следующих разделах.
- Область, которая подвергается склейке, должна содержать только единицы (нули).
- Крайние клетки каждой горизонтали и каждой вертикали также граничат между собой (топологически карта Карно для четырёх переменных представляет собой тор) и могут объединяться в прямоугольники. Следствием этого правила является смежность всех четырёх угловых ячеек карты Карно для N=4. Если во всех четырёх угловых ячейках стоят единицы (нули), они могут быть объединены в квадрат, как показано на рис. 2в.
- Все единицы (нули) должны попасть в какую-либо область.

- С точки зрения минимальности ДНФ (КНФ) число областей должно быть как можно меньше (каждая область представляет собой терм), а число клеток в области должно быть как можно больше (чем больше клеток в области, тем меньше переменных содержит терм. Терм размером 2^n ячеек содержит N-n переменных).
- Одна ячейка карты Карно может входить сразу в несколько областей.
- В отличие от СДНФ (СКНФ), ДНФ (КНФ) не единственны. Возможно несколько эквивалентных друг другу ДНФ (КНФ), которые соответствуют разным способам покрытия карты Карно прямоугольными областями.

Существенно, что в карте Карно соседние клетки обязательно имеют соседние коды, то есть различаются только состоянием — с инверсией или без, одной и только одной из переменных. Так как перестановка переменных в логической функции не изменяет саму функцию то существует несколько вариантов отображения таблицы истинности на карту Карно с сохранением «соседства» клеток. Но практически наиболее часто карту Карно заполняют, используя нарастающий код Грея для обозначения строк и столбцов. Такой подход гарантирует порождение карты Карно с избеганием субъективных ошибок.

Код Гре́я — двоичный код, в котором две «соседние» кодовые комбинации различаются только цифрой в одном двоичном разряде.

Код Грея для n бит может быть рекурсивно построен на основе кода для n-1 бит путём переворачивания списка бит (то есть записыванием кодов в обратном порядке), конкатенации исходного и перевёрнутого списков, дописывания нулей в начало каждого кода в исходном списке и единиц — в начало кодов в перевёрнутом списке.

При заполнении карты на пересечении строки и столбца проставляется соответствующее значение из таблицы истинности — 0 или 1. После того как карта заполнена, приступают к минимизации.

Если необходимо получить минимальную ДНФ, то в Карте рассматриваем только те клетки, которые содержат единицы, если нужна КНФ, то рассматриваем те клетки, которые содержат нули. Сама минимизация производится по следующим правилам (на примере ДНФ).

1. Объединяем смежные клетки, содержащие единицы, в область так, чтобы одна область содержала 2^n клеток (помним про то, что крайние строки и столбцы являются соседними между собой), в области не должно находиться клеток, содержащих нули;
2. Область должна располагаться симметрично оси(ей) (оси располагаются через каждые четыре клетки);
3. Несмежные области, расположенные симметрично оси(ей), могут объединяться в одну;
4. Область должна быть как можно больше, а количество областей как можно меньше;
5. Области могут пересекаться;
6. Возможно несколько вариантов покрытия.

Далее берём первую область и смотрим, какие переменные не меняются в пределах этой области, выписываем конъюнкцию этих переменных; если неменяющаяся переменная нулевая, проставляем над ней инверсию. Берём следующую область, выполняем то же самое, что и для первой, и т. д. для всех областей. Конъюнкции областей объединяем дизъюнкцией.

Для КНФ всё то же самое, только рассматриваем клетки с нулями, неменяющиеся переменные в пределах одной области объединяем в дизъюнкции (инверсии проставляем над единичными переменными), а дизъюнкции областей объединяем в конъюнкцию. На этом минимизация считается законченной.

Синтез логических устройств в базисе ИЛИ_НЕ и И-НЕ

Построение логического устройства на элементах ИЛИ-НЕ (И-НЕ) может быть выполнено при следующей последовательности действий: заданная функция минимизируется с получением МКНФ (МДНФ); производится запись полученного логического выражения через операции ИЛИ-НЕ (И-НЕ). При этом используются логические операции двойной инверсии, а затем правило ДеМоргана.

Для перевода сложных логических выражений в базис И-НЕ, надо получить МДНФ, охватить все элементарные конъюнкции скобками, дважды инвертировать логическое выражение, внести одно из отрицаний внутрь скобок (по формуле де Моргана) и записать полученное выражение в базисе И-НЕ

Рассмотрим последовательность синтеза на примере построения логического устройства, реализующего функцию, приведенную в табл. 3.28.

Для минимизации функции воспользуемся методом Вейча. В табл. 3.29 приведена карта Вейча для рассматриваемой функции.

Таблица 3.28

x_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$f(x_1,x_2,x_3,x_4)$	0	0	1	1	0	0	0	1	1	1	0	0	1	1	0	1

МКНФ функции

$$f(x_1,x_2,x_3,x_4) = (x_1 \vee x_2) \,\&\, (\overline{x_2} \vee \overline{x_3} \vee x_4) \,\&\, (\overline{x_1} \vee x_2 \vee \overline{x_3})$$

Для перехода от базиса И, ИЛИ, НЕ, в котором $\vee$представлено полученное логическое выражение, к базису ИЛИ-НЕ проводим следующие действия:

дважды инвертируем правую часть выражении

$$f(x_1,x_2,x_3,x_4) = \overline{\overline{(x_1 \vee x_3)\cdot(\overline{x_2} \vee \overline{x_3} \vee x_4)\cdot(\overline{x_1} \vee x_2 \vee \overline{x_3})}},$$

проводим преобразование по формуле де Моргана

$$f(x_1,x_2,x_3,x_4)=\overline{(\overline{x_1\vee x_3})\cdot(\overline{\overline{x_2}\vee\overline{x_3}\vee\overline{x_4}})\cdot(\overline{\overline{x_1}\vee x_2\vee\overline{x_3}})}$$

записываем выражение с использованием символа операции ИЛИ-НЕ

$$f(x_1,x_2,x_3,x_4)=(x_1\downarrow x_3)\downarrow(\overline{x_2}\downarrow\overline{x_3}\downarrow x_4)\downarrow(\overline{x_1}\downarrow x_2\downarrow\overline{x_3})$$

Заметим, что в (3.18) наличие поставленных скобок обязательно, иначе исказится функция.

Построенная в соответствии с (3.18) схема логического устройства приведена на рис. 3.32.

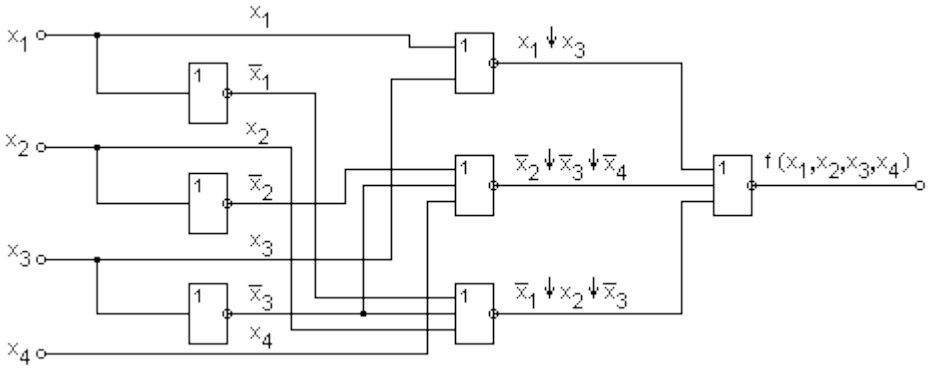


рис 3.32

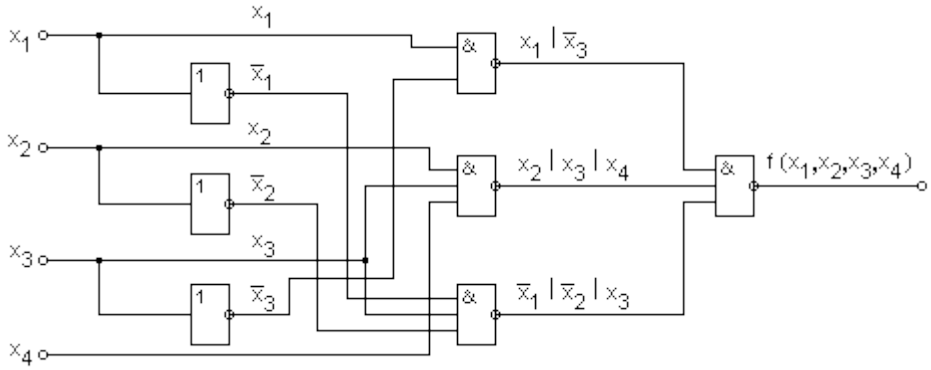


рис 3.33

Методика синтеза устройства в базисе И-НЕ сходна с рассмотренной выше методикой синтеза в базисе ИЛИ-НЕ. Имеющиеся особенности рассмотрим на примере построения с использованием элементов И-НЕ логического устройства, реализующего функцию, заданную таблицей истинности (табл. 3.28).

Минимизируем функцию. В отличие от синтеза в базисе ИЛИ-НЕ, при котором в процессе минимизации получают МКНФ функции, при синтезе в базисе И-НЕ

должна быть получена МДНФ функции. Минимизацию проведем с помощью карты Вейча (табл. 3.30).

Минимальная ДНФ функции

$$f(x_1, x_2, x_3, x_4) = x_1 \cdot \overline{x_3} \vee x_2 \cdot x_3 \cdot x_4 \vee \overline{x_1} \cdot \overline{x_2} \cdot x_3$$

Дважды инвертируем правую часть выражения

$$f(x_1, x_2, x_3, x_4) = \overline{\overline{x_1 \cdot \overline{x_3} \vee x_2 \cdot x_3 \cdot x_4 \vee \overline{x_1} \cdot \overline{x_2} \cdot x_3}}$$

Проводим преобразование по формуле де Моргана

$$f(x_1, x_2, x_3, x_4) = \overline{\left(\overline{x_1 \cdot \overline{x_3}}\right) \cdot \left(\overline{x_2 \cdot x_3 \cdot x_4}\right) \cdot \left(\overline{\overline{x_1} \cdot \overline{x_2} \cdot x_3}\right)}$$

Записываем выражение с использованием символа операции И-НЕ

$$f(x_1, x_2, x_3, x_4) = \left(x_1 \left| \overline{x_3} \right.\right) \left| \left(x_2 \left| x_3 \right| x_4\right) \right| \left(\overline{x_1} \left| \overline{x_2} \right| x_3\right)$$

Выражению (3.20) соответствует схема, приведенная на рис. 3.33

ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ

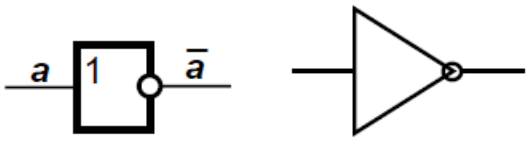
Для условного представления цифровых схем используют систему, принятую Международной Электротехнической Комиссией (МЭК) и применяемую в России в качестве стандарта, а также американскую систему Milspec, которая часто встречается в зарубежной специальной литературе.

Простейшие схемы, работа которых может быть описана с помо-щью основных операций алгебры логики, называются логическими элементами. В системе, принятой МЭК, схема логического элемента обозначается в виде прямоугольника с одним или несколькими входа-ми и одним выходом. Входы соответствуют аргументам функции, а выход – значению функции

Основные логические функции могут быть реализованы с помощью электронных схем, называемых логическими элементами. Условные графические обозначения элементов

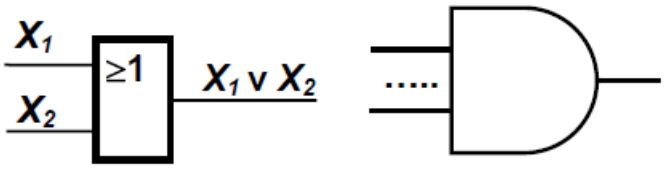
Элемент НЕ

x	y
0	1
1	0



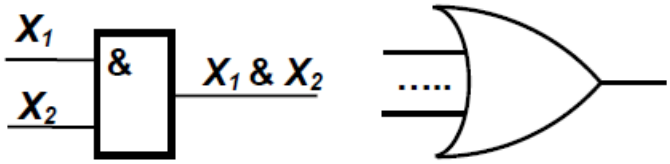
Элемент ИЛИ

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

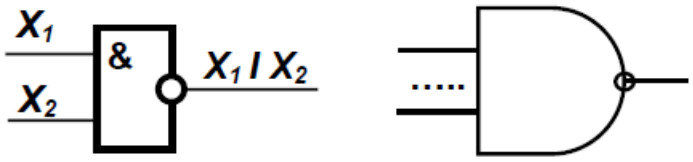


Элемент И

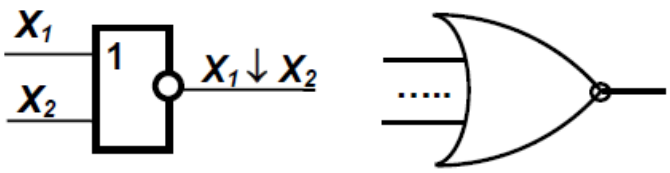
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



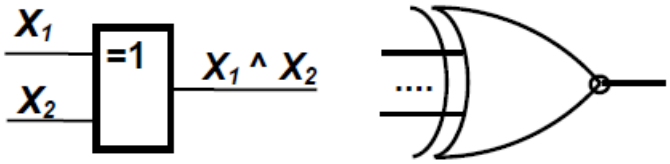
И-НЕ



ИЛИ-НЕ



Исключающее ИЛИ



Комбинационные и последовательностные

Цифровые устройства, выходные сигналы которых зависят от входных сигналов в один и тот же момент времени, называются **ком-бинационными** схемами. Для описания комбинационной схемы с n входами и m выходами необходимо m функций алгебры логики от n переменных. Одной из основных задач при синтезе комбинаци-онных схем является использование минимального числа логических элементов в схеме. Эта задача решается путем минимизации функций алгебры логики.

По способу функционирования логические устройства (и их схемы) делятся на два класса: комбинационные устройства (и соответсвенно комбинационные схемы) и последовательностные устройства (последовательностные схемы).

В комбинационном устройстве (называемом также автоматом без памяти) каждый символ на выходе (логический 0 илил логическая 1) определяются лишь символами (лог. 0 или лог. 1), действующими в данный момент времени на входах устройства, и не зависит от того, какие символы ранее действовали на этих входах. В этом смысле комбинационные устройства лишены памяти (они не хранят сведений о прошлом работы устройства).

В последовательностных устройствах (или автоматах с памятью) выходной сигнал определяется не только набором символов, действующих на входах в данный момент времени, но и внутренним состоянием устройства, а последнее зависит от того, какие наборы символов действовали во все предшествующие

моменты времени. Поэтому можно говорить, что последовательностные устройства обладают памятью (они хранят сведения о прошлом работы устройства).

4. ПОСЛЕДОВАТЕЛЬНОСТНЫЕ УСТРОЙСТВА

К последовательностным устройствам относятся схемы, выходные переменные которых зависят не только от текущих значений входных переменных, но и от последовательности значений входных переменных в предшествующих тактах. Это означает, что последовательностные устройства имеют в своем составе, как комбинационные схемы, так и память (рис. 4.1).

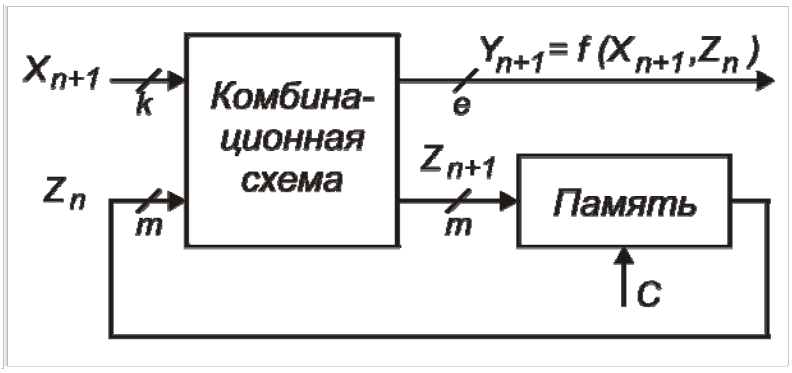


Рис. 4.1. Обобщенная схема последовательностного устройства

Логические состязания

Если известна логическая формула, связывающая входные и выходные переменные проектируемого устройства, то нетрудно составить схему, отдельные элементы которой выполняют логические операции в соответствии с заданной формулой. Но и тогда, когда структура схемы полностью соответствует заданной логической формуле, выходные переменные могут на отдельных интервалах

времени принимать значения, не равные расчетным. Это может быть вызвано тем, что сигналы, представляющие переменные, поступают на входы выполняющего некоторую операцию элемента с различными временными задержками. Такие явления называют логическими состязаниями.

Существо процессов, которые происходят при логических состязаниях, можно выяснить на простом примере. Известно, что произведение логической переменной на ее инверсию тождественно равно 0 (правило отрицания), т.е.:

$$x \cdot x = 0$$

Такое произведение может быть реализовано с помощью схемы, показанной на рис. 2.11а. Временные диаграммы, показывающие процессы в схеме при изменении значения переменной *x*, представлены на рис. 2.11б. Так как сигнал, представляющий переменную *y*, появляется на выходе элемента НЕ с задержкой, то $x \cdot y \neq 0$ в интервале, равном длительности задержки. Временная диаграмма, показывающая переменную *z*, построена с учетом задержки в элементе И. Так как время задержки и длительность фронта сравнимы, то более близкими к реальным осциллограммам являются диаграммы рис. 2.11в.

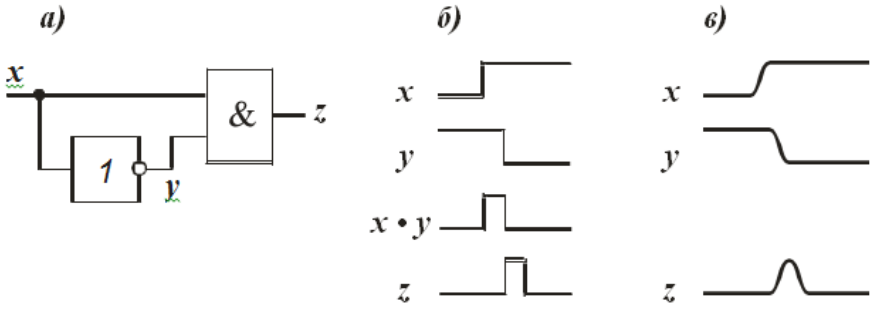


Рис. 2.11. Логические состязания: а – схема;
б, в – временные диаграммы

Таким образом, при изменении значения логической переменной *x* на выходе появляется импульс, в интервале которого выходная переменная не соответствует логической функции. Длительность таких импульсов (их часто называют "иголками") весьма мала (единицы-десятки наносекунд, в зависимости от типа микросхемы), поэтому они могут быть незаметны при осциллографических исследованиях. Однако эти импульсы могут быть причиной сбоев в работе

цифровых схем. Анализ возможностей появления "иголок" и поиск методов нейтрализации эффекта логических состязаний должны производиться на стадии проектирования цифровых устройств.