**Лекция №5**

**Тема «Базовые логические операции и схемы»**

**Цель: дать определение логики и логическим операциям, рассмотреть основные логические операции**

**1.1 Основные понятия и определения.**

В цифровых компьютерах информация представляется и обрабатывается с помо­щью электронных логических схем. Логические схемы оперируют двоичными пе­ременными, принимающими одно из двух значений (обычно таковыми являются нуль и единица). В данном разделе вы познакомитесь с понятием логиче­ских функций и узнаете, как строятся реализующие их логические схемы. Здесь же приведен краткий обзор технологий создания логических схем.

**2.1.** **Базовые логические функции**

Введение в двоичную логику проще всего начать с простого примера, знакомого многим из вас. Представьте себе обычную электрическую лампочку, состояние которой (включена/выключена) управляется двумя выключателями, *х1* и *х2*. Ка­ждый из выключателей может находиться в одном из двух возможных положе­ний, 0 или 1 (рис. 2.1, а). Это означает, что его можно представить как двоичную переменную. Поэтому пусть имена переключателей служат и именами соответст­вующих им двоичных переменных. То, как выключатели будут управлять включением и выключени­ем лампочки, зависит от соединения их проводов. Свет горит лишь в том случае, если образуется замкнутый контур, соединяющий лампочку с источником пита­ния.

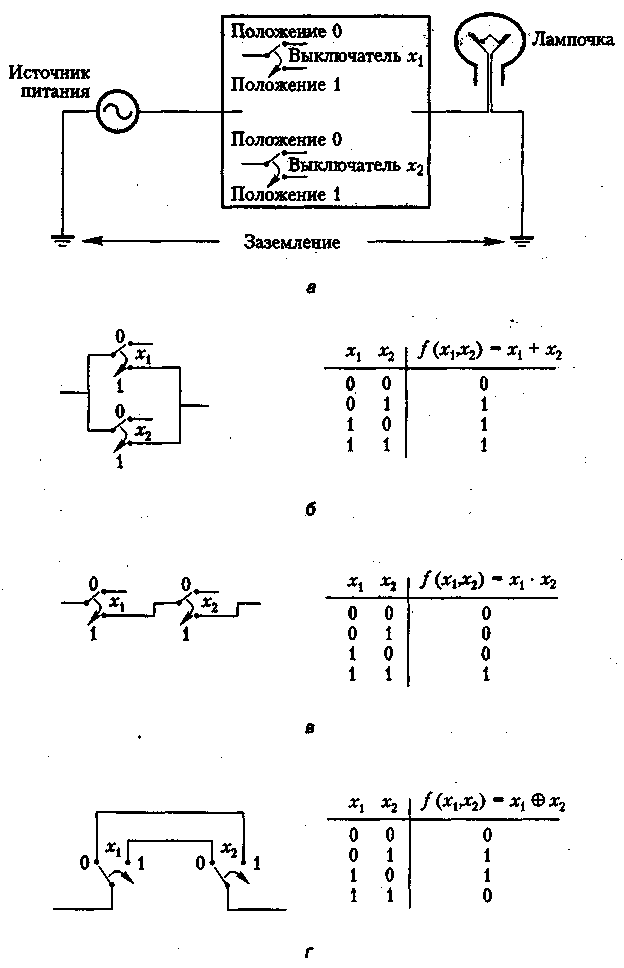


Рис. 2.1. Схемы включения электрической лампочки: лампочка, управляемая двумя выключателями (а); параллельное соединение выключателей — схема **ИЛИ** (б); последовательное соединение выключателей — схема **И** (в); соединение выключателей по схеме **Исключающее ИЛИ** (г)

Пусть условие включения лампочки представляет двоичная переменная *f*. Если лампочка включена, значит *f* = 1, а если она выключена, то *f* = 0. Таким об­разом, условие *f* = 1 указывает, что в цепи существует как минимум один замкну­тый контур, а условие *f* = 0 означает, что замкнутого контура нет. Очевидно, что *f* является функцией двух переменных, *х*1 и *х*2Теперь давайте рассмотрим существующие способы управления лампочкой. Для начала предположим, что она будет гореть при условии, что хотя бы один из переключателей находится в положении 1, то есть *f* = 1, если

*x*1 *=* 1 *и x*2 *=* 0

или

*x*1 *=* 0 *и x*2 *=* 1

или

*x*1 *=* 1 *и x*2 *=* 1

Соединения, реализующие этот тип управления, показаны на рис. 2.1, б. Рядом со схемой приведена представляющая эту ситуацию логическая таблица истинно­сти. В таблице перечислены все возможные пары установок переключателей и соответствующие им значения функции *f.*

В терминах математической логики эта таблица представляет функцию **ИЛИ (OR)** переменных *х*1 и *x*2.

Операцию ИЛИ обычно представляют алгебраическим знаком «+» или «», так что

*f = x1+x2 = x1  x2*

Мы говорим, что x1 и х2 являются входными переменными, а *f*— это выходная функция.

Следует указать некоторые важнейшие свойства операции **ИЛИ**. Прежде все­го, она коммутативна, то есть

*x1+x2 = x2+x1*

Данная операция распространяется на n переменных, так что функция

*f = x1+x2 + … + xn*

принимает значение 1, если это же значение имеет хотя бы одна переменная xi.

Проанализировав таблицу истинности, можно увидеть, что

1 *+ x =* 1

и

0 *+ x = x*

А теперь предположим, что лампочка должна загораться только в том случае, если оба выключателя находятся в положении 1. Такая схема соединения выключателей и соответствующая ей таблица истинности показана на рис. 2.1, в. Эта схема соответствует функции И (AND), для обозначения которой используются символы «∙» или «∩»:

*f = x1 ∙ x2 = x1 ∩ x2*

Вот важнейшие свойства операции И:

*x1 ∙ x2 = x2 ∙x1*

1 *∙ x = x*

0 *∙ x =* 0

Функцию И тоже можно распространить на n переменных:

*f = x1 ∙ x2 ∙ … ∙ xn*

Эта функция имеет значение 1 только в том случае, если все переменные xi имеют значение 1. Она представляет такую же схему, как на рис. 2.1 *в*, в которой, правда, последовательно соединено большее количество выключателей.

Последний вариант соединения выключателей также достаточно распростра­нен. Здесь выключатели подсоединены с двух концов ступенчатого контура, так что лампочку можно включать и выключать с помощью любого из них. Это означа­ет, что если свет включен, изменением положения любого из выключателей его можно выключить, а если свет выключен, изменением положения любого из вы­ключателей его можно включить. Предположим, что лампочка не горит, когда оба выключателя находятся в положении 0. Переключение же любого из них в поло­жение 1 включает лампочку. Теперь предположим, что лампочка горит, если x1 = 1, а *х2* = 0. Переключение x1 в положение 0 выключает лампочку. Более того, для ее выключения можно также установить х2 в положение 1, то есть *f* = 0, если х1 = x2 = 1. Соединение, которое реализует этот способ управления лампочкой, показано на рис.2.1, г. Соответствующая логическая операция, представляемая символом «BD14579_», называется **Исключающее ИЛИ** (**EXCLUSIVE-OR** или **XOR**). Приведем ее важнейшие свойства:



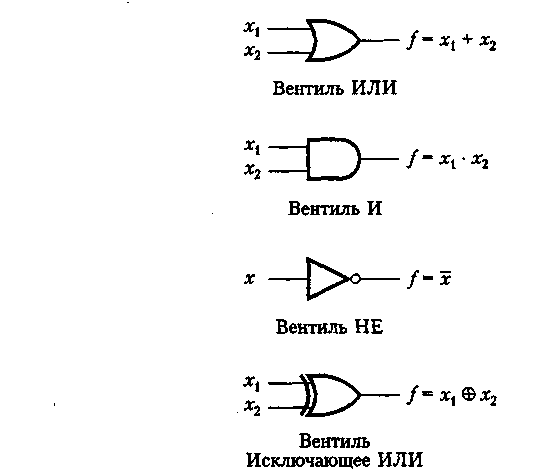
где обозначает функцию НЕ (NOT) от переменной х.

Эта функция переменной f= имеет значение 1, если х = 0, и значение 0, если х = 1. В подобном случае мы говорим, что входное значение х *инвертируется* или *дополняется*.

**2. 1. 1. Электронные логические вентили**

Наш пример с выключателями, замкнутыми и разомкнутыми электрическими цепями и лампочками, иллюстрирующий идею логических переменных и функ­ций, удобен тем, что он очень прост и каждому знаком. При этом представленные им логические концепции применимы к электрическим цепям, используемым для обработки информации в цифровых компьютерах. Физическими переменны­ми в данном случае являются не положения выключателей и замкнутые или ра­зомкнутые цепи, а электрическое напряжение и ток. Для примера рассмотрим схему, предназначенную для работы со входным напряжением +5 В или 0 В. Возможные значения выходного напряжения в ней тоже составляют +5 или 0 В. Если мы договоримся, что значение +5 В представляет логическую единицу, а зна­чение 0 В — логический нуль, тогда функционирование этой схемы можно будет описать с помощью таблицы истинности той логической операции, которую она реализует.

С применением транзисторов можно сконструировать простые электронные схемы, которые будут выполнять логические операции **И**, **ИЛИ**, **Исключающее ИЛИ** и **НЕ**. Эти базовые схемы традиционно называют *вентилями* (gates). Стандартные обозначения вентилей всех четырех типов приведены на рис. 2. 2. Если операция **НЕ** применяется к входному или выходному значению логического вентиля, для нее используется упрощенное обозначение — просто маленький кружок ..



**Рис. 2. 2.** Стандартные обозначения логических вентилей

Об электронной реализации логических вентилей говорится в разделе 2. 5. А пока мы с вами поговорим о том, как с помощью базовых вентилей конструиру­ются логические схемы, реализующие более сложные логические функции

**2. 2. Объединение логических функций**

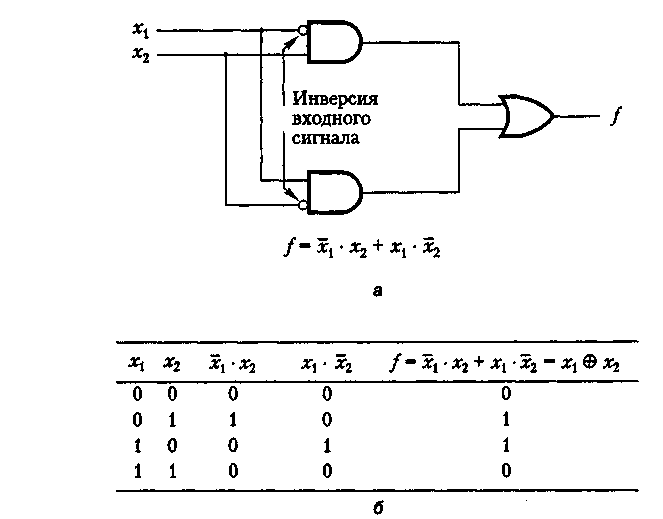
Рассмотрим схему, которая состоит из двух вентилей **И** и одного вентиля **ИЛИ** (рис. 2.3. а). Она может быть представлена выражением



Схема составления таблицы истинности для этого выражения показана на рис. 2.3. *6.* Сначала для каждого входного значения определяются значения тер­мов **И**, затем, с помощью операции **ИЛИ**, — результирующие значения функции f .Таблица истинности функции а идентична таблице истинности функции Исклю­чающее **ИЛИ**, так что схема с тремя вентилями, показанная на рис. 2.3. *а,* реали­зует функцию Исключающее **ИЛИ** с помощью вентилей **И**, **ИЛИ** и **НЕ**. Логиче­ское выражение *∙ x2 + x1 ∙ называется суммой произведений,* поскольку операцию **ИЛИ** иногда называют суммой, а операцию **И** — произведением. Сле­дует отметить, что правильнее было бы записать это выражение так:



Такая форма записи, как вы понимаете, отражает порядок применения логиче­ских операций. Для упрощения подобных выражений определяют иерархию опе­раций **И**, **ИЛИ** и **НЕ**. Если в выражении отсутствуют скобки, логические опера­ции выполняются в следующем порядке: сначала НЕ, затем **И** и только после этого **ИЛИ**. Более того, оператор «•» часто вовсе пропускают, если выражение не допускает двухзначной интерпретации.

**Рис. 2.3.** Реализация функции **Исключающее ИЛИ** с использованием вентилей **И**, **ИЛИ** и **НЕ**: схема для функции **Исключающее ИЛИ** (а); таблица истинности выражения *∙ x2 + x1 ∙ * (б)

Возвращаясь к сумме произведений, мы сейчас покажем, как можно синтези­ровать любую логическую функцию непосредственно на основе ее таблицы ис­тинности (табл. 2.1.).

**Таблица 2. 1**. Функции трех переменных

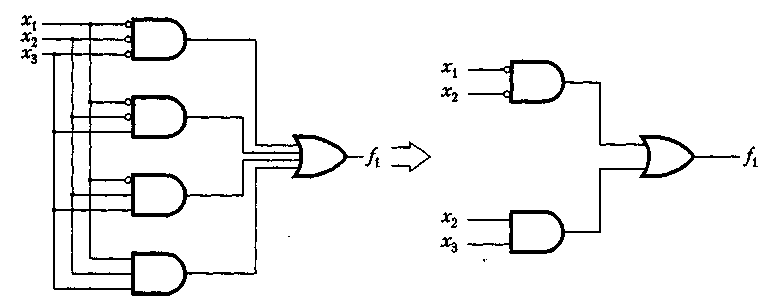
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x1 | x2 | x3 | *f*1 | *f*2 |
| 0  0  0  0  1  1  1  1 | 0  0  1  1  0  0  1  1 | 0  1  0  1  0  1  0  1 | 1  1  0  1  0  0  0  1 | 1  1  1  0  1  1  0  0 |

Предположим, мы хотим составить схему функции f1 на основе вентилей **И**, **ИЛИ** и **НЕ**. Для каждой cтроки таблицы, в которой f1 = 1, в формулу суммы про­изведений включается терм И со всеми тремя входными переменными. К одной, двум или трем из этих переменных по отдельности нужно применить оператор **НЕ** — таким образом, чтобы терм был равен 1 только в том случае, когда значения переменных соответствуют данной строке таблицы истинности. Это означает, что если в этой строке xi = 0, в произведение включается элемент **,а если xi = 1 — элемент xi*.* Например, в четвертой строке таблицы истинности значение функ­ции 1 соответствует входным значениям:



Данной строке соответствует терм *x2x3.* Составив аналогичные термы для всех строк таблицы истинности, в которых функция f1 имеет значение 1, мы полу­чим вот такую сумму произведений:





Логическая схема, соответствующая этому выражению, приведена в левой части рис. 2. 4. В качестве еще одного примера использования описанного алго­ритма можно сформировать сумму произведений для функции **Исключающее ИЛИ**. Этот алгоритм может применяться с целью формирования суммы произве­дений и соответствующей логической схемы на основе таблицы истинности

лю­бого размера.

**Рис. 2. 4.** Логическая схема для функции *f*1 из табл. 2.1 и соответствующая ей минимальная схема реализации

**2. З. Минимизация логических выражений**

Теперь вы знаете, как формируется сумма произведений для произвольной таб­лицы истинности. Фактически для каждой таблицы истинности существует мно­жество эквивалентных выражений и логических схем. *Два логических выраже­ния или две логические схемы считаются эквивалентными, если у них одинако­вые таблицы истинности.* Сформированной нами в предыдущем разделе сумме произведений для функции *f1* эквивалентно, в частности, такое выражение:



Чтобы это доказать, достаточно составить таблицу истинности данного выраже­ния и сравнить ее с таблицей 2. 1. Процесс создания таблицы истинности выраже­ния *+ x2x3,* приведенной в табл. 2. 2, можно разбить на три этапа. Сначала для каждого набора входных значений вычисляется произведение *,* затем — про­изведение *x2x3,* после чего оба результата складываются для получения оконча­тельного значения. Как видите, наша таблица истинности идентична таблице ис­тинности функции *f*1, приведенной в табл. 2. 1.

Для упрощения логических выражений выполняется ряд алгебраических опе­раций. Они основаны на двух логических законах, о которых мы еще не упомина­ли: дистрибутивном и законе исключенного третьего:

****

****

**Таблица 2.2.** Вычисление выражения *+ x2x3,*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x1 | x2 | x3 |  | *x2x3* | *+ x2x3,* |
| 0  0  0  0  1  1  1  1 | 0  0  1  1  0  0  1  1 | 0  1  0  1  0  1  0  1 | 1  1  0  0  0  0  0  0 | 0  0  0  1  0  0  0  1 | 1  1  0  1  0  0  0  1 |

**Таблица 2.3.** Использование таблицы истинности для доказательства эквивалентности выражений

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *w* | *y* | *z* | *y+z* | Значение  *w(y+z)* | *wy* | *wz* | Значение  *wy+wz* |
| 0  0  0  0  1  1  1  1 | 0  0  1  1  0  0  1  1 | 0  1  0  1  0  1  0  1 | 0  1  1  1  0  1  1  1 | 0  0  0  0  0  1  1  1 | 0  0  0  0  0  0  1  1 | 0  0  0  0  0  1  0  1 | 0  0  0  0  0  1  1  1 |

В табл. 2.З. приведено доказательство истинности дистрибутивного закона. Очевидно, что подобные законы всегда можно доказать с помощью таблиц истин­ности выражений, стоящих справа и слева от знака равенства. Совпадение ре­зультирующих значений в этих двух таблицах подтверждает проверяемый закон. Логические законы, подобные дистрибутивному, иногда называют *тождествами*.Вот еще одна форма дистрибутивного закона, которую мы приводим для полноты изложения материала, хотя она нам и не потребуется:



Целью минимизации логического выражения, представляющего заданную логи­ческую функцию, является уменьшение стоимости ее реализации (количества ис­пользуемых логических элементов). Общая схема процесса реализации логической функции такова. Сначала по описанному нами алгоритму для нее составляется сум­ма произведений (дизъюнктивная совершенная нормальная форма). Затем получен­ное выражение минимизируют до эквивалентной *минимальной суммы произведений.* Чтобы определить критерий минимизации, нужно ввести понятие стоимости, или величины, логического выражения.

Обычно при оценке стоимости выражения учитывается общее количество вентилей и их входных значений (входных линий), необходимых для реализации выражения в форме, показанной на рис. 2. 4. Например, стоимость большей схе­мы на этом рисунке равна 21: 5 вентилей плюс 16 входных значений. Инверсия входных значений при подсчете игнорируется. Стоимость более простого выра­жения равна 9: 3 вентиля плюс 6 входных значений.

Теперь можно определить и критерий минимизации.

*Сумма произведений считается минимальной, если не существует эквивалентного ей выражения меньшей стоимости.*

Строгие доказательства минимальности таких функций не является предметом данного курса, а в простых приме­рах, которые мы будем рассматривать, минимальный размер выра­жений будет очевидным.

Стратегия упрощения заданного выражения заключается в следующем. Преж­де всего, термы-произведения разбиваются на пары, отличающиеся единой пере­менной, которая в одном терме дополняется *(),* а во втором используется как есть *(х).* Затем в каждой паре общее произведение двух переменных выносится за скобки, а в скобках остается терм *х +* *,* всегда равный 1. Вот что мы получим, применив эту процедуру к первому выражению для функции *f*1:









Это выражение минимально. Соответствующая ему логическая схема приве­дена на уже упоминаемом нами рис. 2. 4.

Сгруппировать термы попарно, с тем чтобы упростить исходное выражение, не всегда так просто, как в примере с функцией *f*1. В случае затруднений помогает такое правило:

*w + w = w*

Это правило позволяет повторять термы-произведения при необходимости объединить некоторый терм более чем с одним другим термом. Для примера рассмотрим функцию *f*2 из табл. 2. 1. Исходная сума произведений, формируемая на основе таблицы истинности этой функции, такова:



Повторив первый терм и изменив порядок следования термов (на ос­нове коммутативного закона), мы получим:



Сгруппировав термы попарно, и вынеся одинаковые произведения за скобки, мы сможем записать следующее выражение:



Первую пару термов можно упростить еще раз, и тогда получится минималь­ное выражение:



На этом обсуждение способов алгебраического упрощения логических выра­жений завершается. Данное математическое упражнение еще раз доказывает, что более простые логические схемы, содержащие меньше вентилей и входных значе­ний, легче и дешевле реализовать на практике. Так что стремление минимизиро­вать логические выражения имеет под собой чисто экономическое основание. За­коны, которые мы с вами использовали для манипулирования логическими выражениями, объединены в. табл. 2. 4. Они приведены парами, чтобы видна была симметрия функций **И** и **ИЛИ**. До сих пор нам не представилось случая восполь­зоваться законами возведения в степень и де Моргана, но в следующих разделах они нам пригодятся.

**Таблица 2. 4.** Законы двоичной логики

|  |  |  |
| --- | --- | --- |
| **Название закона** | **Алгебраическое тождество** | |
| Коммутативный  Ассоциативный  Дистрибутивный  Идемпотентности  Возведение в степень  Дополнения (Закон исключения третьего)  Закон де Моргана | w + y = y + w  (w+y)+z = y + (w+z)  w + yz = (w+y)(w+z)  w + w = w  = w  w + = 1  =  1 + w = 1  0 + w = w | wy = yw  (wy)z = w(yz)  w(y+z) = wy + wz  ww = w  w = 0  =  +  0 ∙ w = 0  1 ∙ w = w |

**Контрольные вопросы**

1. Что такое логика?
2. Какие логические операции бывают?
3. Что такое логическая операция И?
4. Что такое логическая операция ИЛИ?
5. Что такое логическая операция НЕ?
6. Назовите основные законы двоичной логики

**ДОПОЛНИТЕЛЬНЫ МАТЕРИАЛ К ЛЕКЦИИ**