Логические основы построения ЭВМ

Понятие логической функции

В ЭВМ информация подвергается не только арифметической, но и логической обработке. При этом машина выполняет преобразования над двоичными числами; в результате этого образуются двоичные числа, являющиеся результатом выполнения соответствующей логической операции.

В основе работы логических схем и устройств ЭВМ используется **математическая логика**, изучающая применение математических методов для решения различных логических задач. Наибольшее применение из области математической логики получила **алгебра логики**, часто называемая **исчислением высказываний**.

1 Под **высказыванием** понимается любое утверждение, которое в зависимости от смысла бывает **истинным** или **ложным**. В логике высказываний интересуются не содержанием высказываний, а лишь их истинностью или ложностью; другие признаки высказываний в алгебре логики не рассматриваются.

Если высказывание истинно, то говорят, что его значение равно «1», если высказывание ложно, то его значение равно «0». Таким образом, **значения высказываний можно рассматривать как переменную величину, принимающую только 2 дискретных значения: «0» и «1». Это приводит к полному соответствию между логическими высказываниями и цифрами двоичной системы счисления.**

С точки зрения логики высказывания делятся на:

* **постоянно истинные высказывания,** например: *«Солнце - источник дневного света»; «Скорость - производная пути по времени»; «За воскресеньем всегда следует понедельник»* (математически их принимают равными «1»);
* **постоянно ложные высказывания**, например: *«Снег - тёплый»; «Москва стоит на берегу Невы»; «делится на без остатка »*.
* **высказывания, которые могут быть истинными или ложными в зависимости от определённых условий,** т.е. принимают значения «0» или «1» попеременно, например: *«На улице идёт дождь»* — высказывание истинно и равно «1», пока идёт дождь; после окончания дождя оно становится ложным и равным «0».

По содержанию высказывания бывают **простыми** и **сложными (составными)**. Например: из простых высказываний с помощью союзов **И** и **ИЛИ** и отрицательной частицы **НЕ** можно составить сложные высказывания:

* 1. «Москва не стоит на берегу Невы, и Ленинград стоит на берегу Невы».
  2. «Москва не стоит на берегу Невы».
  3. «Москва стоит на берегу Невы или Ленинград стоит на берегу Невы».

Все три высказывания истинны.

Простое высказывание при вхождении в состав сложного является **логическим аргументом**, а сложное высказывание является **логической функцией**, зависящей от ложности или истинности аргумента.

Обычно принято простые высказывания обозначать строчными буквами латинского или русского алфавита: x, y, z, m, p, l, a, b,… .

Сложные высказывания, или логические функции, обозначают прописными буквами латинского или русского алфавита: F, P, Q, X, М, П, А… .

Всякое устройство ЭВМ, выполняющее арифметические действия над двоичными числами, можно рассматривать как функциональный преобразователь, входными переменными (аргументами) которого является исходное двоичное число, а выходной функцией от них — новое двоичное число, которое образовалось в результате выполнения данной операции. При этом как входные переменные, так и выходные функции могут принимать лишь одно из двух возможных значений — 0 и 1.

В каждом случае количество входных переменных может быть различным. В простейшем случае — одна переменная х, принимающая значение или 0, или 1.

В общем случае таких переменных может быть n: , причем каждая из этих переменных принимает лишь одно из двух возможных значений - 0 или 1.

Логические двоичные функции получили название Булевых по имени английского математика XIX века Дж. Буля.

2 Совокупность значений входных переменных (аргументов) называется **набором** и обозначается , где равно 0 или 1.

Функция , определяемая на наборах двоичных аргументов и принимающая в качестве своих возможных значений 0 или 1, называется **логической**, или **булевой (переключательной)** функцией.

Для задания булевой функции достаточно построить таблицу её значений, отвечающей всевозможным различным наборам аргументов. Таблица получила название **таблица истинности**.

3 Некоторые свойства переключательных функций:

1. Любая переключательная функция n аргументов определяется на наборах.

Набор n аргументов — это двоичное n-разрядное число. Но количество различных поразрядных чисел равно .

Например:

* + 1. переключательные функции 2-х аргументов определены на 4 наборах: (0,0); (0,1); (1,0); (1,1).
    2. переключательные функции трёх аргументов определены на восьми наборах:(0,0,0); (0,0,1); (0,1,0); (0,1,1); (1,0,0); (1,0,1); (1,1,0); (1,1,1).

Условимся упорядочивать аргументы переключательной функции по возрастанию индексов: . Тогда каждому набору можно приписать номер, равный двоичному числу, соответствующему данному набору.

Пример: 0,0,0,0,0,0 - нулевой набор;

0,0,0,0,0,1 - первый набор;

0,0,0,0,1,0 - второй набор;

0,0,1,0,1,0 - десятый набор;

1,1,1,1,1,1 - 63-й набор;

Набор аргументов, содержащий все единицы (1,1,1,1,1,1), называют **единичным набором**.

1. Число различных переключательных функций n аргументов конечно и равно .

Переключательная функция n аргументов определена на наборах, на которых она принимает значение 0 или 1. Поэтому каждой переключательной функции можно поставить в соответствие *-*разрядное двоичное число, а поскольку этих чисел , то такое же количество и различных переключательных функций.

В дальнейшем будем приписывать каждой переключательной функции номер, равный двоичному числу, образованному значениями переключательной функции, например:

функция двух аргументов F(х, y) принимает значения:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0 | 0 | **1** |
| 0 | 1 | **1** |
| 1 | 0 | **0** |
| 1 | 1 | **1** |

№ функции — 13.

Задача синтеза логических схем ЭВМ эквивалентна математической задаче представления (с использованием принципа суперпозиции) сложных переключательных функций через простые переключательные функции, описывающие работу логических элементов.

Суперпозиция булевых функций состоит в замене аргументов булевой функции другими булевыми функциями и в перестановке (или переименовании) аргументов булевых функций. При её осуществлении необходимо учитывать ещё и порядок выполнения, т.е. старшинство, булевых операций, причем функция НЕ должна выполнятся первой, И — второй, в последнюю очередь - ИЛИ. Для изменения порядка выполнения операций используются скобки.

Преобразования над аргументами, в результате которых получены значения Булевых функций, получили название **операций алгебры логики**.

**4 Переключательные функции одного аргумента**

Существует 4 различные переключательные функции одного аргумента.

# Переключательные функции одного аргумента

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | 0 | 1 | Условное обозначение | Название функции |
| f0(x) | 0 | 0 | 0 | const 0 |
| f1(x) | 0 | 1 | u | Переменная х |
| f2(x) | 1 | 0 |  | Инверсия х |
| f3(x) | 1 | 1 | 1 | const 1 |

# 

# 5 Функция f0(x) ≡ 0. Эту функцию называют **константный нуль** и обозначают f0(x)=0.

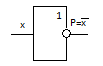
Функция f3(x) ≡ 1. Эту функцию называют **константой единицы** и обозначают f3(x)=1.

Функция f1(x) повторяет значение аргумента х, поэтому f1(x) ≡ х.

Функция f2(x) принимает значение, противоположное значению аргумента х: если х=0, то f2(x)=1; если х=1, то f2(x)=0. Эта функция названа **инверсией** переменной х, или **отрицанием** х (х с отрицанием).

**Х-черта** называется знаком отрицания.

Математически это выражается формулой: Р = x̅ (Р есть не х). Элемент, реализующий функцию НЕ, или операцию отрицания, функцию инверсии, называется элементом НЕ (инвертором). На функциональных схемах обозначается так:



Переключательные функции двух аргументов

(Логические функции двух переменных)

1. **Логическое сложение** двух или нескольких простых высказываний — это функциональная зависимость, в результате которой сложное высказывание Р будет истинно, если хотя бы одно из составляющих его простых высказываний истинно, и ложно, когда одновременно ложны все составляющие его простые высказывания.

Формула логической связи: (P есть х ИЛИ у), где ˅ — знак дизъюнкции — логическое сложение (Р = х+у).

Логика работы элемента дизъюнкции задается таблицей истинности:

|  |  |  |
| --- | --- | --- |
| х | у | Р = х˅у |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

F7(x,y) № функции 7

Схема элемента логического сложения называется **схемой ИЛИ**, **дизъюнктором** или **собирательной схемой**. Её функциональное обозначение:

x

1

P = x˅y

y

1. **Логическое умножение** двух или более высказываний заключается в том, что сложное высказывание P будет истинно в том и только в том случае, когда составляющие его простые высказывания будут одновременно истинны.

Логическое умножение, конъюнкция обозначается знаком конъюнкции ˄, или знаком & (and), или буквой И.

Формула: Р= х˄у (Р есть х И у).

Логический элемент И работает в соответствии с таблицей истинности:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | 0 | 0 | 1 | 1 |
| у | 0 | 1 | 0 | 1 |
| Р = х^у | 0 | 0 | 0 | 1 |

№ функции 1 F1(x,y)

и называется **конъюнктором**, или **схемой И**, или **схемой совпадения**. Её функциональное обозначение:

x

&

P = x˄y

y

1. **Константа нуль**:

|  |  |  |
| --- | --- | --- |
| х | у | F0(x,y) |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

F0(x,y) № функции 0

Логический элемент, реализующий функцию “константа нуль”, называется “генератор нуля”.

1. **Константа единица:**

|  |  |  |
| --- | --- | --- |
| х | у | F15(x,y) |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

F15(x,y) № функции 15

Логический элемент, реализующий F15(x,y) — “генератор единиц”.

1. **Отрицание дизъюнкции ( стрелка Пирса, или функция Вебба)** —логическая функция истинна только тогда, когда составляющие её простые высказывания одновременно ложные.

Запись м. б. в одном из видов:

F=x↓y; ; ; F=x o y;

Таблица истинности:

|  |  |  |
| --- | --- | --- |
| х | у | F8(x,y) |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

№ функции 8 F8(x,y)

Для реализации операции используется элемент Пирса (ИЛИ-НЕ).

Функциональное изображение элемента ИЛИ-НЕ:

x

1

y

1. **Отрицание конъюнкции** **(Штрих Шеффера)** — функция истинна, если хотя бы одно из составляющих её высказываний ложно.

Алгебраическая запись операции Шеффера:

F(x,y)=x/y; F=x^y; F=xy

Таблица истинности отображает логику работы:

|  |  |  |
| --- | --- | --- |
| х | у | F(x,y) |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

№ функции 14 F14(x,y)

Для реализации операции применяется элемент Шеффера (элемент И-НЕ) функциональное изображение которого:



1. **Эквивалентность (равнозначность)** – логическая операция, в результате которой функция будет истинной, если составляющие её аргументы равноценны или равнозначны.

Обозначение: F(x,y)= x ≡ y;

F(x,y)= x~y;

F(x,y)=x  y;

Логика роботы:

|  |  |  |
| --- | --- | --- |
| х | у | F(x,y) |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

№ функции 9 F9(x,y)

Элемент, реализующий функцию, называется равнозначность.

Функциональное обозначение:

x

=

xy

y

1. **Сумма по модулю 2** — операция, в результате которой логическая функция истинна, если составляющие ее аргументы неравнозначны, неравноценны.

Алгебраическая запись: F(x,y)=ху

Сумма по mod 2 – логическая неравнозначность, т.е. F(x,y)= x ≡ y;

Логика работы:

|  |  |  |
| --- | --- | --- |
| х | у | F6(x,y) |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

№ функции 6 F6(x,y)

Элемент, реализующий операцию, называется “Сложение по mod 2”, или M2.

Функциональное обозначение:

x

M2

М2 ху

y

1. **Импликация от х к у** – логическая операция, в результате которой функция ложна только в том случае, когда х истинна, а у – ложна. X называется посылкой, а Y – следствием.

“Если 2\*2=5, то снег черный”. (ЕСЛИ X, то Y)

Функция обозначается: F(x,y)=х→у

(если Х, то У)

F(x,y)= x̅˅y

Таблица истинности:

|  |  |  |
| --- | --- | --- |
| х | у | F13(x,y) |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

№ функции 13 F13(x,y)

Операция реализуется элементом импликация.

Функциональное обозначение:

1

x

x+y

y

1. **Импликацией от у к х** считается такая логическая операция, в результате которой функция ложна только в том случае, когда величина х ложна, а у - истинна.

Функция обозначается: F(x,y)=y→x (ЕСЛИ y, то x)

F(x,y)=x˅y̅

Таблица истинности:

|  |  |  |
| --- | --- | --- |
| х | у | F11(x,y) |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

№ функции 11 F11(x,y)

Операция реализуется элементом импликация.

Функциональное обозначение :

x

1

x+y

y

Логическая связь “ЕСЛИ-ТО”.

Остальные 6 функций являются либо повторителями входной переменной, либо запретом по одному из входов (отрицание импликации), либо инверсия одного из входов.   
F2(x,y) х∆у ; Запрет по х Схема запрета

х^у; отрицание импликации

F3(x,y) х Переменная х Повторитель х

F4(x,y) у∆х Запрет по у Схема запрета

х^у; отрицание импликации

F5(x,y) у Переменная у Повторитель у

F12(x,y) х Отрицание х, Инвертор х, НЕ

инверсия х

F10(x,y) у Отрицание у Инвертор у, НЕ

инверсия у

**9 Основные законы алгебры логики**

С помощью этих законов преобразуют и упрощают исходные логические функции.

1. **Переместительный закон**:

* для логического сложения: 
* для логического умножения: 

1. **Сочетательный закон**:

* для логического сложения: 
* для логического умножения: 

1. **Распределительный закон**:

* для логического сложения: 
* для логического умножения: 

1. **Закон инверсии**:
2. для логического сложения: 

Формулы

де Моргана

1. для логического умножения: 
2. Отсюда следует **закон двойного отрицания**: 

Распределительный закон для логического умножения и закон инверсии не имеют аналогов в математике, алгебре и характерны лишь для алгебры логики.

Доказательство этих законов можно получить построением таблиц истинности.

В качестве примера рассмотрим таблицу истинности для формул де Моргана:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Значение аргументов и функций | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  |  |  |  |  | \* |  | \*\* | \* | \*\* |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |

Аналогично можно доказать справедливость других соотношений в алгебре логики.

Из основных законов алгебры логики и свойств логических функций НЕ, И, ИЛИ можно вывести следующие заключения:

**Формулы для дизъюнкции:**

;

;

; - закон исключения третьего

;

**Формулы для конъюнкции**:

;

;

;

; - закон противоречия

**Формулы для инверсии**:

;

;

**Формулы для системы функций**:

* операция поглощения ( поглощает ):



* II операция поглощения [ поглощает ]:



* операция склеивания по переменной :



* II операция склеивания по переменной :



* Правило действия со скобками



Следует отметить, что все рассмотренные соотношения справедливы для любого числа аргументов.

Используя приведенные выше законы и соотношения алгебры логики, можно существенно упростить сложные логические выражения и получить более простые формулы.

Пример: 

Упростить выражение для.



*.*

*{(y+1) и (х+х)≡1}*

Из перечисленных выше законов и соотношений можно сделать выводы:

1. Если логическая сумма двоичных аргументов или функций содержит хотя бы одну пару слагаемых, являющихся взаимно-инверсными величинами, то эта сумма тождественна 1 (т.е. истинна):

 ; 

1. Если логическое произведение двоичных аргументов или функций содержит хотя бы одну пару сомножителей, являющихся взаимно-инверсными величинами, то это произведение ложно, т.е. тождественно 0:

; 

Для упрощения (минимизации) сложных логически выражений используется ряд методов, но наиболее часто применяются приёмы, связанные с основными законами алгебры логики, законами поглощения, исключения третьего, инверсии.

**10 Формы логических функций и их использование для синтеза логических схем**

Табличное представление логических функций значительно усложняется с увеличением числа аргументов. Так, для трёх аргументов мы имеем  логических функций.

Для этой цели более удобным является аналитическое представление логических функций в виде формул. Из-за неоднозначности представления любой переключательной функции через исходные необходимо найти такую форму ее представления, которая соответствует наиболее простой электрической схеме.

Наиболее рациональным является представление логической функции в нормальных формах: совершенной дизъюнктивной и совершенной конъюнктивной.

Введем понятия: элементарная конъюнкция, элементарная дизъюнкция, ранг функции.

**Элементарная конъюнкция (минтерм)** образуется конъюнкцией конечного множества логических переменных и их отрицаний.

Пример: 

**Элементарная дизъюнкция (макстерм)** образуется дизъюнкцией конечного множества логических переменных и их отрицаний.

Пример: 

Число переменных (аргументов), составляющих элементарную конъюнкцию или дизъюнкцию, называется её **рангом**.

Примеры*:*  – минтерм 4 ранга;

– макстерм 3 ранга.

Форма представления логических функций посредством инверсии, конъюнкции и дизъюнкции, называется **нормальной**.

Различают **дизъюнктивную** и **конъюнктивную** нормальные формы.

**Дизъюнктивная нормальная форма (ДНФ**) – является логической суммой элементарных конъюнкций (минтермов).

*Пример*: 

**Конъюнктивная нормальная форма (КНФ)** – является логическим произведением элементарных дизъюнкций (макстермов).

*Пример*: 

**Минтермом** называют функцию, принимающую единичное значение на одном из всех возможных наборов аргументов, и нулевые – при всех других, а **макстерм** – функция, принимающая нулевое значения на одном из возможных наборов входных аргументов и единичное значение – при всех других.

Алгебраически минтерм – конъюнкция аргументов и их отрицаний, а макстерм – дизъюнкция аргументов и их отрицаний.

Количество минтермов и макстермов совпадает с числом наборов различных аргументов, т.е. для n аргументов их будет .

Наборы элементарных конъюнкций и дизъюнкций для двух аргументов приведены в таблице:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Значение аргументов* | | *Значения минтермов (элементарных конъюнкций)* | | | | *Значения макстермов*  *(элементарных дизъюнкций)* | | | |
|  |  |  |  |  |  |  |  |  |  |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

Выражение логической функции, составленное из минтермов одинакового ранга, связанных дизъюнкцией, называют **совершенной дизъюнктивной нормальной формой**  (СДНФ).

Пример СДНФ: 

Функция  отвечает требованиям, предъявляемым к СДНФ:

1. не содержит одинаковых конъюнкций;
2. ни одна конъюнкция не содержит двух одинаковых переменных;
3. ни в одной из конъюнкций не содержится булевой переменной и её отрицания;
4. все конъюнкции одного ранга.

Если логическая функция содержит конъюнкции разных рангов, то следует использовать свойство  для конъюнкции младшего ранга и повысить её ранг для образования СДНФ, например:

*.*

Логическую функцию, содержащую элементарные дизъюнкции одного ранга, связанные конъюнкцией, называют **совершенной конъюнктивной нормальной формой** (СКНФ).

Пример СКНФ: 

Функция  характеризуется такими свойствами:

1. не содержит одинаковых дизъюнкций;
2. ни одна дизъюнкция не содержит двух одинаковых двоичных аргументов;
3. ни одна дизъюнкция не содержит переменную вместе с её отрицанием;
4. все дизъюнкции одного ранга.

При анализе или синтезе логических схем ЭВМ обычно используют описание их работы, заданное в виде соответствующей таблицы, из которой легко могут быть получены логические формулы в СДНФ и СКНФ.

**Правила образования СДНФ функций и аргументов:**

1. По каждому набору двоичных переменных, при котором функция принимает значение единицы, составить элементарные конъюнкции (минтермы).
2. В элементарные конъюнкции записать неинвертированными переменные, заданные единицей в таблице истинности, а инвертированными – те переменные, которые заданы в таблице истинности нулем.
3. Соединить элементарные конъюнкции знаком дизъюнкции.

**Правила образования СКНФ по таблице истинности:**

1. По каждому набору двоичных переменных, при котором функция принимает значение нуля, составить элементарные дизъюнкции (макстермы).
2. В элементарные дизъюнкции записать неинвертированными переменные, заданные нулями в соответствующем наборе, а инвертированными – те переменные, которые заданы единицей.
3. Элементарные дизъюнкции соединить знаком конъюнкции.

СКНФ используется реже, чем СДНФ при преобразовании логических выражений.

Пример:

Пусть функция  задана таблицей истинности:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | минтермы | макстермы |
| 0 | 0 | 0 | 1 |  | -------- |
| 0 | 0 | 1 | 0 | -------- |  |
| 0 | 1 | 0 | 1 |  | -------- |
| 0 | 1 | 1 | 0 | -------- |  |
| 1 | 0 | 0 | 0 | -------- |  |
| 1 | 0 | 1 | 1 |  | -------- |
| 1 | 1 | 0 | 0 | -------- |  |
| 1 | 1 | 1 | 1 |  | -------- |

Запись функции :

* в СДНФ

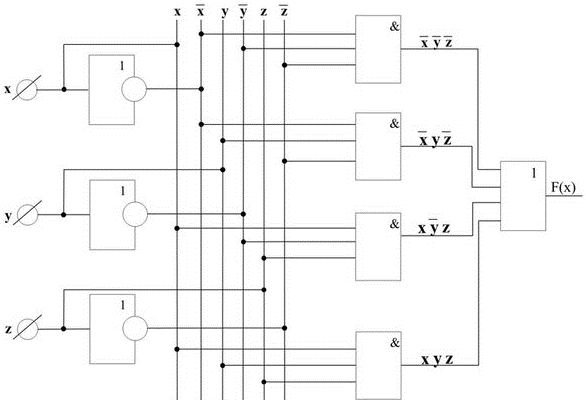


* в СКНФ



Полученные формулы функции  в СДНФ и СКНФ можно использовать для синтеза функциональных схем логических устройств.

Так, для реализации функции  в СДНФ, если не применять никаких преобразований, можно принять схему



Понятие

11 О Функционально полных системах переключательных функций

Одна из задач синтеза схем заключается в выборе типов элементов, из которых должны собираться логические схемы. Основное требование к набору логических элементов - построить с помощью имеющегося этого набора любою сколь угодно сложную схему. Но ввиду того, что законы функционирования элементов описываются переключательными функциями, то сформулированное требование сводится к определению набора таких переключательных функций, с помощью которых можно получить любую сколь угодно сложную функцию, что соответствует операции суперпозиции, заключающейся в замене одних аргументов функции другими.

[ Например, если аргументы функций Z(x,y) являются, в свою очередь, функциями других аргументов Х=Х(a,b) и У=У(c,d), то можно образовать функцию вида Z=Z(a,b,c,d)].

Упрощение логических выражений можно достичь выражением сложных логических функций через другие функции.

Система переключательных функций называется **функционально полной**, если с помощью функций, входящих в эту систему, применяя операции суперпозиции, можно получить любую сколь угодно сложную переключательную функцию.

Примерами функционально полных систем переключательных функций могут служить:

1) инверсия, дизъюнкция и конъюнкция;

2) т.к. закон инверсии позволяет преобразовывать дизъюнкцию в конъюнкцию и наоборот, то функционально полными будут:

а) дизъюнкция и инверсия;

б) конъюнкция и инверсия.

3) Т.к. через отрицание конъюнкции или отрицание дизъюнкции можно представить любую логическую функцию от n переменных, и эти функции являются также функционально полными, то эти две функции называют **универсальными логическими функциями**.

12 Минимизация логических функций

Преобразование логической функции с целью упрощения её аналитического выражения называется **минимизацией**.

Рассмотренные СДНФ и СКНФ используются для первоначального представления заданной переключательной функции через функции основной системы. Однако эти формы неудобны для построения логических схем ЭЦВМ, т.к. схемы, реализующие эти формы, оказываются часто сложными.

Для минимизации логических функций на практике используются следующие методы:

а) метод непосредственных преобразований;

б) метод Квайна;

в) метод Карно-Вейча;

г) метод Блейка;

и другие.

Метод непосредственных преобразований

Алгоритм применения:

* 1. Строится СДНФ логической функции.
  2. С использованием различных законов и соотношений алгебры логики СДНФ преобразуется (и упрощается) в минимальную дизъюнктивную нормальную форму (МДНФ) содержащую min число логических элементарных конъюнкций.

Пример: проведем синтез схем для реализации функции, заданной в СДНФ

P(x,y,z)=xyz +xyz +xyz +xyz

на

1) логических элементах И, НЕ, ИЛИ

* 1. элементах Пирса, НЕ, ИЛИ.

Применив метод склеивания соседних элементарных конъюнкций, получаем:

P(x,y,z)=x̅z̅+xz

Реализация на элементах И, НЕ, ИЛИ:

&

&

1

1

1

x

хz

P = xz+xz

z

xz

Для реализации схемы на элементах ПИРСА, НЕ и ИЛИ, преобразуем выражение

P(x,y,z) = xz + xz

с применением формул де Моргана и закона двойного отрицания:

P(x,y,z) = x+z + x+z = х+z + xz

Схема:

1

1

1

1

1

x

P = x+z + x+z

z

x+z

После дальнейших упрощений данная схема приобретает вид:

x+z

1

1

&

x Ø •

z Ø ◦ P(x,y,z)

xz

Методы Квайна и Карно-Вейча будут рассмотрены в курсе «Математическая логика».

13 Понятие о синтезе логических схем и цифровых автоматов

При синтезе логических устройств ЭВМ, выполняющих необходимое преобразование цифровой информации, можно выделить следующие этапы:

1. На основании анализа функций, которые должны выполняться данным устройством, формируются логические условия его функционирования в виде соответствующей таблицы.
2. По этой таблице составляется СДНФ логической функции.
3. Производится минимизация логической функции.
4. По упрощенной логической формуле строится функциональная схема устройства, причем минимальному числу и однородности логических элементов отдается предпочтение.

14 Комбинационные схемы и конечные автоматы

По существу, логические устройства ЭВМ выполняют необходимое преобразование поступающей на вход цифровой информации. Их возможно классифицировать по типам:

1. комбинационные схемы, или автоматы без памяти, или примитивные автоматы;
2. конечные, или полные автоматы, или автоматы с памятью.

**Комбинационная схема** – это устройство, в котором совокупность выходных сигналов в дискретный момент времен t*i* однозначно определяется набором входных сигналов, поступивших на вход устройства в тот же момент времени t*i* .

.

Комбинационную схему можно представить в виде m-k полюсника, имеющего *i* входов и j выходов. Входное слово М (входной алфавит) КС задается набором символов m1, m2,….,mi

m1 

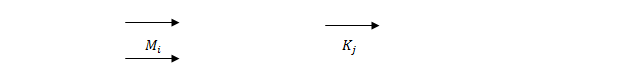
**КС**

**(Mf Kh)**

Входное слово



.





.

Выходное слово

а выходное слово КС К (выходной алфавит) задается набором символов k1,k2,…,kj.

КС характеризуется:

* числом входных сигналов;
* числом выходных сигналов;
* логической формулой или таблицей истинности.

При изменении набора входных сигналов М меняется набор выходных сигналов К. Т.о. выходные сигналы КС полностью определяются входными сигналами и не зависят от внутреннего состояния КС. Поэтому КС называется также **автоматами без памяти**, или **примитивными автоматами**.

Конечно, КС в таком представлении является идеализацией технического устройства. В реальных схемах из-за различной продолжительности переходных процессов в электрических цепях и элементах схем время появления выходных сигналов не совпадает со временем поступления входных сигналов, т.е. выходные сигналы появляются с некоторым запаздыванием по отношению ко времени поступления входного слова. С целью управления работой устройств – разрешение подачи на вход КС нового набора входных сигналов лишь после окончания переходных процессов в КС – используется **тактированный способ** работы устройств, когда новый такт преобразования информации начинается только после окончания предыдущего.

**Конечный автомат** – устройство, выходное слово в котором в дискретный момент времени  определяется не только по входному слову в тот же момент времени , а и по его внутреннему состоянию, обусловленному его предшествующими этапами работы.

Автомат с памятью задается тремя наборами переменных:

M ( - входной набор (входной алфавит);

K ( - выходной набор (выходной алфавит);

Q ( - набор внутренних состояний автомата (внутренний алфавит).

Функционирование конечного автомата

K (

M (m1, m2, .. m3)

Q(

однозначно определено, если установлены связи во времени между тремя его алфавитами. Обычно значение алфавитов сохраняются неизменными на протяжении временных интервалов, называемых **тактами**.

Закон функционирования конечного автомата возможно задать 3 способами:

-табличным;

-аналитическим;

-графическим/

**Табличный способ.** Таблица содержит графы:

- набор входных переменных Mt в момент дискретного времени t;

- внутренние состояния автомата Qt в дискретный момент времени t (до воздействия набора входных переменных);

- внутренние состояния автомата Qt+1 в дискретный момент времени t +1 (после воздействия входных переменных Mt );

- значения выходных переменных Кt+1 (после воздействия входных переменных Mt ) в дискретный момент времени t +1.

Пример:

|  |  |  |  |
| --- | --- | --- | --- |
| Такт t | | Такт t+1 | |
| Mt | Qt | Qt+1 | Кt+1 |
|  |  |  |  |

**Аналитический способ** реализуется заданием формул (уравнений) для функций переходов и выходов, получаемых из табличного представления образованием СДНФ и СКНФ.

**Функции переходов** устанавливают аналитическую связь между внутренним состоянием автомата Qt+1 в момент времени t+1 в зависимости от предыдущего состояния Qt и набора входных переменных Mt .

**Функции выходов** отражают связь между выходным алфавитом Кt+1 в зависимости от внутреннего состояния Qt и входногоMt алфавитов.

Т.o. аналитически поведение автомата описывается:

Qt+1= F1 (Qt, Mt); Задание автомата

Кt+1= F2 (Qt, Mt); Мили

Конечный автомат, выходной сигнал которого зависит от состояния, в котором находится автомат, и от сигналов на его входе, называется автоматом Мили.

В устройствах ЦВМ применяются автоматы другого типа, у которых выходной сигнал Кt в момент дискретного времени t зависит исключительно от состояния автомата Qt в этот момент времени и не зависит от входных сигналов Mt . Такой автомат называется **автоматом Мура**.

Функционирование автомата Мура описывается уравнениями:

Qt+1= F1 (Qt, Mt);

Кt+1= F2 (Qt, );

**Графическое представление** законов функционирования автоматов осуществляется с помощью графов. При этом вершинами графа (узловыми окружностями) отображают внутренние состояние автомата, а переход из одного состояния в другое - дугами (ветвями) графа, на котором указывают входные переменные в момент изменения его внутреннего состояния. Выходные переменные для автоматов Мура наносятся внутри вершин графа, а для автомата Мили - на дугах.

Примеры: Граф автомата Мура Граф автомата Мили

(общий вид) (общий вид)

Qt+1

Kt+1

Qt

Kt

Qt+1

Qt

Kt

Mt

Mt

# Рассмотренные способы используют при анализе работы автоматов и при синтезе их логической структуры.

Примеры: КС – логические элементы; КА – триггер.

Обозначение элементов “И”, ”ИЛИ”, “НЕ”

1

x x X F =

&

X X X

X&Y F = X·Y

Y Y Y

1

X X X

X+Y F=X+Y F

Y Y

## 19. Еволюція програмних модулів в процесі обробки (по Олефіру схемка є)

Програма, записана будь-Якою мовою програмування, назівається початкова модулем (ПМ). початковий модуль

перетворюється відповіднім транслятором в об'єктній модуль (ОМ) - модуль в проміжному форматі, спільному для всіх

трансляторів. Об'єктній модуль галі не призначення для безпосередно виконан, оскількі крім тексту програми (галі не

зовсім готового для виконан на ЕОМ) містіть Додатковий інформацію про організацію зв'язків между програмних модулів та

про налагодження програми на конкретному місце в оператівній пам'яті во время ее завантажування. ОМ винен пройти ще один

етап ОБРОБКИ - редагування. Авторитети етапі ВІН обробляється системних програмою «Редактор зв'язків». У результаті

з'являється програма, готова до завантаження ее в ОП з Настроювання на конкретні адреси. Така програма назівається

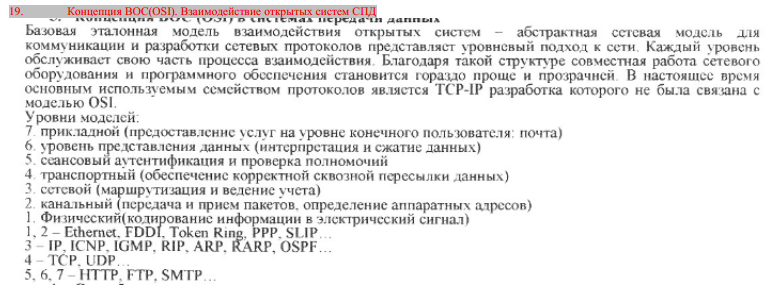
завантажувальну модулем (ЗМ). ПМ, ОМ та ЗМ є переміщуванімі модулями, що не пов'язаними з конкретними адресами пам'яті

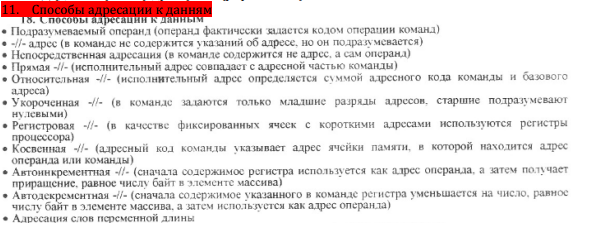
ЕОМ. После завантаження в ОП спеціальною програмою-завантажніком и Настроювання на певні адреси модуль становится

непереміщуванім (абсолютним).

Об'єктній модуль складається з тексту програми в машинному коді та двох Словників - словника зовнішніх імен та словника

Переміщень



****