Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Институт информационных технологий

Кафедра информационных систем и технологий

А. Г. Савенко, А. В. Матвеев

ЛОГИЧЕСКИЕ И СТРУКТУРНЫЕ ОСНОВЫ ЦИФРОВЫХ УСТРОЙСТВ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве учебно-методического пособия для специальности 6-05-0611-05 «Компьютерная инженерия»

Минск БГУИР 2025

УДК 004.3’144(076)
ББК 32.971.32я73
С12

Р е ц е н з е н т ы:

кафедра информационных технологий и математики
учреждения образования «БИП – Университет права
и социально-информационных технологий»
(протокол № 9 от 30.04.2024);

ректор учреждения дополнительного образования
«Институт информационных технологий и бизнес-администрирования»
кандидат технических наук, доцент В. К. Дюбков

Савенко, А. Г.

С12 Логические и структурные основы цифровых устройств : учеб.-метод. пособие / А. Г. Савенко, А. В. Матвеев. – Минск : БГУИР, 2025. – 115 с. : ил.
ISBN 978-985-543-823-7.

Предназначено для студентов первого курса специальности «Компьютерная инженерия». Содержит теоретический материал, практические работы, а также примеры решения практических задач. Можно использовать для самостоятельной работы студентов указанной специальности очной, заочной и дистанционной форм обучения. Будет полезно студентам всех специальностей профиля образования 06 «Информационно-коммуникационные технологии» и специалистам в области информационных технологий.

УДК 004.3’144(076)
ББК 32.971.32я73

ISBN 978-985-543-823-7

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цифровые устройства играют ключевую роль в современной технологии, охватывая широкий спектр областей: от информационных технологий и коммуникаций до научных и промышленных приложений. Эти устройства, основанные на принципах арифметики и логики, предоставляют нам средства для обработки, хранения и передачи данных в цифровой форме, что является фундаментальным элементом современного информационного общества.

Цифровые устройства воплощают принципы арифметики и логики в физическую реальность, позволяя нам выполнять разнообразные задачи с высокой скоростью и эффективностью. Они являются основой для работы компьютеров, мобильных устройств, сетей связи, автоматизированных систем управления и многих других технологий, которые мы используем каждый день.

Основные компоненты цифровых устройств, такие как логические вентили, регистры, счетчики и микропроцессоры, обеспечивают выполнение арифметических операций, принятие логических решений и управление потоком данных в системе. Изучение и понимание их принципов работы является ключевым элементом для специалистов в области информационных технологий, электроники, автоматизации и других смежных областей.

В рамках дисциплины «Арифметические и логические основы цифровых устройств» изучаются основные принципы и методы работы цифровых устройств, а также их применение в различных областях техники и технологии. Рассматривается алгебра логики, методы минимизации функций, структурные элементы цифровой техники и их использование для создания сложных систем обработки информации.

К основным целям учебно-методического пособия относятся:

– предоставление студентам базовых знаний о принципах работы цифровых устройств, арифметической и логической алгебре, которые лежат в основе современных информационных технологий;

– обеспечение студентов информацией для развития необходимых навыков для анализа, проектирования и моделирования цифровых систем на основе логических и арифметических операций;

– подготовка студентов к решению практических задач, связанных с проектированием и разработкой цифровых устройств и систем;

– знакомство студентов с основными методами минимизации функций алгебры логики и их применением в цифровой технике.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1 ЛОГИЧЕСКИЕ ОСНОВЫ ЦИФРОВЫХ УСТРОЙСТВ

Для описания и синтеза устройств цифровой вычислительной техники используется математический инструментарий, известный как алгебра логики или булева алгебра. Алгебра логики применяется также для формализации, упрощения, вычисления и преобразования логических выражений.

Алгебра логики – это раздел математики, который изучает логические высказывания, анализируя их логические значения (истинность или ложность), а также проводит операции над ними.

Этот раздел математики начал развиваться в середине XIX века благодаря работам Дж. Буля и последователей, таких как Ч. Пирс, П. С. Порецкий, Б. Рассел, Д. Гильберт и др. Алгебра логики возникла как попытка решения традиционных логических проблем с помощью алгебраических методов.

С появлением теории множеств в 1870-х годах часть исходных задач алгебры логики была включена в область теории множеств, а дальнейшее развитие математической логики, особенно в последней четверти XIX века и первой половине XX века, привело к смещению акцентов в предмете алгебры логики. Основной объект изучения в алгебре логики – это логические высказывания.

1.1 Основные понятия алгебры логики

Основными понятиями алгебры логики являются «логическое высказывание», «логическая константа», «логическая переменная» и «логическая функция».

Логическое высказывание – это утверждение, которое имеет определенное значение истинности и ложности. Оно может быть сформулировано в виде повествовательного предложения и быть подвержено оценке как истинное или ложное.

Любое высказывание можно обозначить каким-либо символом, например *x*, при этом если высказывание истинно, то $x = 1$, а если высказывание ложно, то $x = 0$.

Используя логические операторы И, ИЛИ, НЕ и др., можно объединять простые высказывания в более сложные. Важно отметить, что истинность или ложность сложного высказывания зависит от истинности или ложности составляющих его простых высказываний и логических операторов, которые их объединяют.

Например, утверждения «два, умноженное на три, будет равно шести», «студенты первого курса изучают алгебру логики», «три больше пяти», «число

10 является нечетным», «на улице идет дождь» являются логическими высказываниями.

Из приведенных примеров высказываний некоторые являются абсолютно истинными или абсолютно ложными во всех возможных ситуациях. Эти высказывания называются логическими константами. Например, высказывание «Земля – это планета Солнечной системы» всегда истинно, независимо от дополнительных условий. Остальные высказывания могут быть истинными или ложными в зависимости от конкретной ситуации. Например, «на улице идет дождь» может быть истинным или ложным в зависимости от погодных условий в определенный момент времени и в определенном месте.

Логическая константа – это такая постоянная величина, значением которой может быть только «истинно» или «ложно» («да» или «нет», «единица» или «нуль»).

Логические (*булевы, двоичные*) *переменные* представляют собой основные элементы алгебры логики, которые могут принимать только одно из двух значений: ложь или истину (нуль или единицу), т. е. если *x* – логическая переменная, то $x = \{0, 1\}$. Эти переменные используются для представления логических состояний или значений в цифровых системах, где они являются основой для построения логических выражений и выполнения логических операций. Логические переменные являются аргументами логических функций.

Логическая (*булева*) *функция* – это такая функция алгебры логики, которая может принимать только одно из двух значений: ложь или истину (нуль или единицу) в зависимости от текущего(-их) значения(-ий) аргументов, в качестве которых могут выступать логические переменные. То есть это математическое отображение из множества всех возможных комбинаций значений входных логических переменных в множество значений выходных логических переменных.

Таким образом, булева функция определяет логическую зависимость между входными и выходными булевыми переменными. Каждой комбинации значений входных переменных соответствует определенное значение выходной переменной. Например, булева функция f, зависящая от n переменных $x_1, x_2, \dots, x_n$, называется булевой, если функция f и любая из ее переменных x_i $(i = 1, 2, \dots, n)$ принимают только значения 0 или 1. Булеву функцию можно задавать перечислением значений при различных значениях переменных. Например, для задания функции f от n переменных необходимо определить значения для всех возможных наборов. Количество таких наборов определяется как 2^n (с нулевого по $2^n - 1$) (таблица 1.1).

Таблица 1.1 – Обозначение функции для всех возможных наборов переменных

Номер набора переменных	**Значения логических переменных $x_1 x_2 \dots x_n$ (двоичный код)**	**Логическая функция $f(x_1, x_2, \dots, x_n)$**
0	00…00	$f(0, 0, \dots, 0, 0)$
1	00…01	$f(0, 0, \dots, 0, 1)$
…	…	…
$2^n - 2$	11…10	$f(1, 1, \dots, 1, 0)$
$2^n - 1$	11…11	$f(1, 1, \dots, 1, 1)$

Таким образом осуществляется задание функции. При этом набор значений $f(0, 0, \dots, 0), f(0, 0, \dots, 1), \dots, f(1, 1, \dots, 0), f(1, 1, \dots, 1)$ представляет собой вектор значений функции. Булевы функции могут служить аргументами более сложных булевых функций.

Значение (нуль или единица), которое принимает абстрактная логическая функция $f(x_1, x_2, \dots, x_n)$ при определенных значениях аргументов (т. е. на определенном наборе), в общем случае неконстантно, т. е. не имеет предопределенного значения и задается исходя из логики, которую должна реализовывать функция.

Предопределенное значение (только такое и никакое другое), которое принимает логическая функция на конкретном наборе значений переменных, имеет место для «именных» логических функций, например: для функции логического сложения (ИЛИ, дизъюнкция), функции логического умножения (И, конъюнкция), функции логического отрицания (НЕ), функции Пирса, функции Шеффера и т. д. (подраздел 1.4). Такие функции реализуют определенную логику, поэтому их значения при определенных наборах значений аргументов предопределены.

В общем же случае логическая функция может реализовывать логику решения любой задачи и принимать значения, описывающие решение конкретной задачи.

Для иллюстрации логики работы «неименной» функции рассмотрим следующий пример. Трое друзей (Петя, Оля и Вася) имеют желание поехать отдыхать на море в июле. Каждый из друзей в данном случае является аргументом логической функции, а значениями данных аргументов будут являться их возможности (ложь, или нуль – отсутствие возможности поехать в июле, истина, или единица – наличие возможности). Следовательно, наша логическая функция является функцией трех переменных $f(\text{П}, \text{О}, \text{В})$ и может иметь $2^3 = 8$ наборов значений аргументов и соответствующих каждому набору восемь значений самой функции. При этом значением функции может быть только «ложь» (нуль) или

«истина» (единица). В случае ложности значения функции $f(\text{П}, \text{О}, \text{В})$ поездка не состоится, а в случае истинности – состоится.

Для наглядности значения аргументов и самой функции на каждом наборе представлены в таблице 1.2.

Таблица 1.2 – Значения аргументов и функции $f(\text{П}, \text{О}, \text{В})$ на всех возможных наборах

Номер набора	**Значения аргументов**			**Значение функции *f***
	Петя	**Оля**	**Вася**	
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Как видно из таблицы 1.2, поездка состоится только в половине из возможных случаев (наборы 3, 4, 6, 7). Поскольку наша логическая функция описывает конкретную ситуацию с конкретными людьми (аргументами), для пояснения того, почему именно при таких наборах значений аргументов (возможностях друзей) поездка состоится и почему не состоится при других, рассмотрим каждый из наборов значений:

– нулевой набор. Ни Петя, ни Оля, ни Вася не имеют возможности совершить поездку в июле, соответственно поездка не состоится (f (П, О, В) = 0);

– первый набор. Петя и Оля не имеют возможности совершить поездку в июле, а Вася имеет возможность. Однако поездка не состоится, поскольку Вася не имеет финансовой возможности оплатить одноместное размещение в отеле (f (П, О, В) = 0);

– второй набор. Петя и Вася не имеют возможности совершить поездку в июле, а Оля имеет возможность. Однако поездка не состоится, поскольку Оля не хочет ехать одна (f (П, О, В) = 0);

– третий набор. Петя не имеет возможности совершить поездку в июле, а Оля и Вася имеют возможность. Поездка состоится, поскольку Васе не нужно будет оплачивать одноместное размещение и Оля будет не одна (f (П, О, В) = 1);

– четвертый набор. Петя имеет возможность совершить поездку в июле, а Оля и Вася не имеют такой возможности. Но поездка состоится, поскольку Петя

очень хочет отдохнуть и имеет финансовую возможность оплатить одноместное размещение в отеле (f (П, О, В) = 1);

– пятый набор. Оля не имеет возможности совершить поездку в июле, а Петя и Вася имеют такую возможность. Однако поездка не состоится, поскольку Петя и Вася знакомы через Олю и не очень хорошо знают друг друга и решают остаться вместе с Олей (f (П, О, В) = 0);

– шестой набор. Петя и Оля имеют возможность совершить поездку в июле, а Вася не имеет такой возможности. Поездка состоится, поскольку Оля поедет не одна и друзья очень хотят отдохнуть (f (П, О, В) = 1);

– седьмой набор. И Петя, и Оля, и Вася имеют возможность совершить поездку в июле. Поездка состоится, поскольку друзья хотят отдохнуть и в большой компании будет веселее (f (П, О, В) = 1).

Как видно из приведенного примера, логическая функция f (П, О, В) описывает конкретную ситуацию для конкретных аргументов, поэтому функция на определенных наборах аргументов имеет именно такие значения. При изменении какого-либо из аргументов изменится и сама ситуация, соответственно и значения функции для другой ситуации могут быть иными.

Логическая функция f из приведенного примера могла бы быть задана и как функция шести переменных, где для каждого из трех друзей было бы по два аргумента: возможность уйти в отпуск в июле и иные возможности (f ($П_о$, $П_и$, $О_о$, $О_и$, $В_о$, $В_и$)).

Логическая функция не обязательно должна принимать значение «ложь» (нуль), если все аргументы функции принимают значение «ложь» (нуль), или значение «истина» (единица), если все аргументы принимают значения «истина» (единица). Это также зависит от конкретной задачи, которую описывает логическая функция.

Например, происходит венчание пары, на котором присутствует двое свидетелей, и священнослужитель спрашивает у остальных гостей, может ли кто-то назвать причину, по которой венчание не может произойти. В данном случает это может быть описано логической функцией двух переменных (по числу присутствующих гостей) f (C_1, C_2). Тогда каждый из аргументов принимает значение «ложь» (нуль), если свидетель не может назвать такой причины, и значение «истина» (единица), если свидетель может назвать такую причину. Венчание состоится только в том случае, если никто из свидетелей не озвучит такую причину, т. е. только тогда функция может принять значение «истина» (единица) (таблица 1.3).

Таблица 1.3 – Значения аргументов и функции f (C_1, C_2) на всех наборах

Номер набора	Значения аргументов		Значение функции f
	Свидетель 1	Свидетель 2	
0	0	0	1
1	0	1	0
2	1	0	0
3	1	1	0

1.2 Способы задания логических функций

Зависимость логической функции от аргументов (логических переменных) может быть представлена различными формами. Это зависит от конкретного контекста, целей и удобства задания булевой функции.

Логическая функция может быть задана следующими способами:

1 Словесным описанием. Например, логическая функция двух переменных может быть словесно описана следующим образом: «Если завтра будет выходной день и будет хорошая погода, то наша семья отправится в парк на пикник». Аргументами данной функции являются наличие выходного дня и погодные условия, при этом функция примет значение «истина» только в случае истинности обоих аргументов, а во всех остальных случаях функция будет принимать значение «ложь». Или, например: «Функция трех переменных принимает значение «истина», если любые два аргумента или все три аргумента принимают значение «истина»; во всех других случаях функция принимает значение «ложь». Словесное описание может использоваться в случае сравнительно несложной логической функции. Задание сложных логических функций и (или) логических функций большого количества аргументов словесным описанием является непонятным и неудобным для восприятия.

2 Таблицей истинности (матрицей). Таблица истинности логической функции представляет собой матрицу, состоящую из 2^n строк (где n – количество аргументов функции, тогда 2^n – количество уникальных комбинаций значений аргументов, т. е. наборов) и минимум $n + 1$ столбцов (со значениями каждой из n переменных и значением самой функции, которое она принимает при определенной комбинации значений аргументов. Также может быть столбец, обозначающий номер набора значений аргументов). Каждая переменная (аргумент) и сама функция может принимать одно из двух возможных значений: 0 (ложь) или 1 (истина). Пример задания логической функции таблицей истинности представлен ниже (таблица 1.4).

Таблица 1.4 – Таблица истинности логической функции четырех переменных

Номер набора	**Значения аргументов**				**Значение функции** $f(x_1, x_2, x_3, x_4)$
	x_1	x_2	x_3	x_4	
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	1

Как видно из таблицы 1.4, номер набора представляет собой десятичный эквивалент двоичной комбинации, образуемой значениями аргументов функции. В таблице истинности наборы значений аргументов должны располагаться именно в возрастающей двоичной последовательности (от 00…00 до 11…11).

В одной таблице истинности может задаваться несколько логических функций, зависящих от одних и тех же переменных. Пример задания трех логических функций y_1, y_2, y_3, зависящих от одних и тех же двух аргументов x_1 и x_2, представлен в таблице 1.5.

Таблица 1.5 – Задание трех логических функций, зависящих от одних и тех же двух аргументов, одной таблицей истинности

Номер набора	**Аргументы**		**Функции**		
	x_1	x_2	y_1	y_2	y_3
0	0	0	0	0	1
1	0	1	1	1	0
2	1	0	1	1	0
3	1	1	1	0	1

Таблица истинности является универсальным средством задания логической функции. Однако задание булевых функций, зависящих от большого количества переменных, с использованием таблицы истинности представляется неудобным из-за ее громоздкости. Например, для булевой функции с восемью аргументами таблица истинности будет содержать $2^8 = 256$ строк.

Для описания функций с множеством переменных удобно использовать ***модификацию таблицы истинности***. Рассмотрим способ построения такой таблицы истинности для функции n переменных. Множество из n переменных функции разбивается на два подмножества: $x_1, x_2, \dots, x_{j-1}$ и $x_j, x_{j+1}, \dots, x_n$. Строки модифицированной таблицы истинности помечаются переменными $x_1, x_2, \dots, x_{j-1}$, принимая значение соответствующего двоичного набора длиной $j-1$. Столбцы модифицированной таблицы истинности помечаются переменными $x_j, x_{j+1}, \dots, x_n$, принимая значение соответствующего двоичного набора длиной $n-j+1$. Значение функции записывается в ячейке, находящейся на пересечении соответствующей строки и столбца (таблица 1.6).

Таблица 1.6 – Задание функции модифицированной таблицей истинности

$x_1, x_2, \dots, x_{j-1}$	$x_j, x_{j+1}, \dots, x_n$				
	00…0	**00…1**	**…**	**11…0**	**11…1**
00…0	1	1	…	1	1
00…1	0	1	…	0	1
…	…	…	…	…	…
11…0	0	0	…	1	1
11…1	1	0	…	0	1

Пример модифицированной таблицы истинности логической функции четырех аргументов (приведенной в таблице 1.4) представлен в таблице 1.7.

Таблица 1.7 – Пример модифицированной таблицы истинности, задающей логическую функцию четырех переменных $f(x_1, x_2, x_3, x_4)$

x_1, x_2	x_3, x_4			
	00	**01**	**10**	**11**
00	1	1	1	
01	1	0	0	0
10	1	0	0	0
11	0	0	0	1

Как видно, таблица 1.7 значительно компактнее таблицы 1.4.

Рассмотренный матричный способ задания булевых функций удобен и позволяет наглядно представлять различные логические функции. Однако для выполнения анализа свойств функций алгебры логики ни классическая таблица истинности, ни модифицированная не используются и не считаются компактными.

3 Логическим выражением (аналитически). Аналитическое задание логической функции представляет собой логическое выражение (формулу).

Логическое выражение – это комбинация логических переменных и констант, связанных логическими операторами (И (логическое умножение), ИЛИ (логическое сложение), НЕ (логическое отрицание)), которые также могут содержать оператор группировки (скобки).

Пример задания логической функции четырех аргументов (также заданной таблицей истинности в таблицах 1.4 и 1.7) выглядит в дизъюнктивной нормальной форме (ДНФ) (подраздел 1.3) следующим образом:

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

а в конъюнктивной нормальной форме (КНФ) (подраздел 1.3):

$$f(x_1, x_2, x_3, x_4) = (x_1 + x_2 + \bar{x}_3 + \bar{x}_4)(x_1 + \bar{x}_2 + x_3 + \bar{x}_4) \times$$
$$\times (x_1 + \bar{x}_2 + \bar{x}_3 + x_4)(x_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4)(\bar{x}_1 + x_2 + x_3 + \bar{x}_4) \times$$
$$\times (\bar{x}_1 + x_2 + \bar{x}_3 + x_4)(\bar{x}_1 + x_2 + \bar{x}_3 + \bar{x}_4)(\bar{x}_1 + \bar{x}_2 + x_3 + x_4) \times$$
$$\times (\bar{x}_1 + \bar{x}_2 + x_3 + \bar{x}_4)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3 + x_4).$$

Аналитический подход к заданию логических функций играет важную роль в проектировании цифровых устройств.

Практически все операции, необходимые для разработки цифровых устройств, осуществляются с использованием аналитического представления функций.

4 Логической схемой (схемотехнический). Такое задание представляет собой графическое обозначение логической функции с использованием логических элементов и связей между ними, определяющими последовательность выполнения логических операций в задаваемой сложной функции. Такой способ позволяет визуализировать логическую структуру функции.

Пример задания логической функции логической схемой представлен на рисунке 1.1.

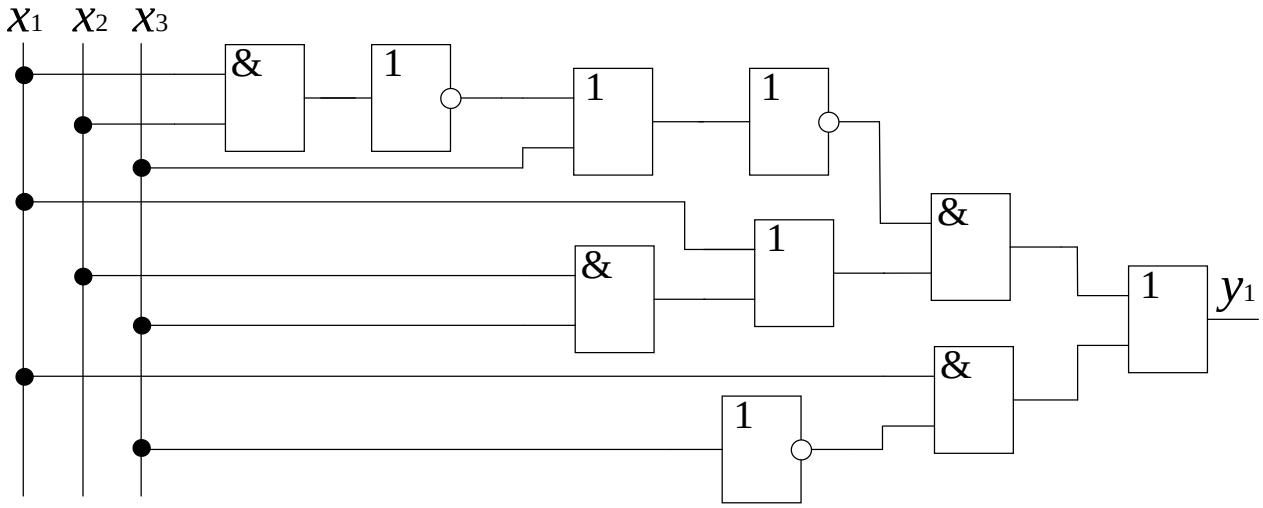


Рисунок 1.1 – Пример задания логической функции логической схемой

5 Картами (диаграммами) Венна, Вейча, Карно. Данный способ представляет собой графическое задание логической функции и несет в себе такую же информацию, как и при других способах задания. Более подробно задание логических функций картами (диаграммами) будет рассмотрено в подразделах 1.7–1.9.

6 Числовым способом. Данный способ предполагает перечисление наборов, на которых функция принимает единичное значение (истина), тогда говорят, что функция задана на единичных наборах, или перечисление наборов, на которых функция принимает нулевое значение (ложь), тогда говорят, что функция задана на нулевых наборах.

Примеры задания числовым способом функции, приведенной в таблицах 1.4 и 1.7 и заданной ранее аналитически в пункте 3:

а) функция, заданная на единичных наборах:

$f(x_1, x_2, x_3, x_4) = \{0, 1, 2, 4, 8, 15\}$;

б) функция, заданная на нулевых наборах:

$f(x_1, x_2, x_3, x_4) = \{3, 5, 6, 7, 9, 10, 11, 12, 13, 14\}$.

Как видно из примеров задания логической функции, данный способ наиболее компактный, но менее наглядный. По записи числового задания логической функции можно восстановить всю информацию о ней: при каких значениях аргументов функция принимает единичное значение (истина), а при каких значениях аргументов – нулевое значение (ложь).

1.3 Формы представления логических функций

Одну и ту же логическую функцию, заданную аналитически, можно представить различными логическими выражениями, т. е. в различных формах представления. Для понимания формы представления необходимо ввести понятия простой конъюнкции и простой дизъюнкции.

Простая конъюнкция – логическое произведение любого конечного числа различных логических переменных, в котором каждая переменная встречается не более одного раза в прямой или инверсной форме (со знаком инверсии или без него).

Простая дизъюнкция – логическая сумма любого конечного числа различных логических переменных, в которой каждая переменная встречается не более одного раза в прямой или инверсной форме (со знаком инверсии или без него).

Различают дизъюнктивную нормальную форму представления логических функций и конъюнктивную нормальную форму.

Дизъюнктивная нормальная форма (*ДНФ*) – это дизъюнкция (логическая сумма) конечного числа *простых конъюнкций* (логических сомножителей). Например:

$$f = \underbrace{x_1 \cdot \overline{x_2} \cdot x_3 \cdot \overline{x_4}}_{\text{простая конъюнкция (ПК)}} + \underbrace{\overline{x_3} \cdot x_4 \cdot x_6}_{\text{ПК}} + \underbrace{\overline{x_1} \cdot x_5 \cdot x_7}_{\text{ПК}}$$

дизъюнкция *дизъюнкция*

Конъюнктивная нормальная форма (*КНФ*) – это конъюнкция (логическое произведение) конечного числа *простых дизъюнкций* (логических слагаемых). Например:

$$f = \underbrace{(x_2 + x_4 + x_5 + \overline{x_7})}_{\text{простая дизъюнкция (ПД)}} \cdot \underbrace{(\overline{x_1} + \overline{x_2} + x_3 + \overline{x_5})}_{\text{ПД}} \cdot \underbrace{(x_3 + x_4 + x_7)}_{\text{ПД}}$$

конъюнкция *конъюнкция*

Различают также такие канонические формы представления логических функций, как совершенная дизъюнктивная нормальная форма и совершенная конъюнктивная нормальная форма.

Совершенная дизъюнктивная нормальная форма (*СДНФ*) – это форма, которая представляет собой дизъюнкцию простых конъюнкций, причем простые конъюнкции содержат все аргументы функции в своей прямой или инверсной форме (полные конъюнкции) и отражают собой наборы, на которых функция принимает единичные значения. Такие полные конъюнкции называются конституентами единицы.

Конституента единицы – набор (дизъюнкция переменных в прямой или инверсной форме), на которых функция принимает единичные значения.

Совершенная конъюнктивная нормальная форма (*СКНФ*) – это форма, которая представляет собой конъюнкцию простых дизъюнкций, причем простые дизъюнкции содержат все переменные в своей прямой или инверсной форме (полные дизъюнкции) и отражают собой инверсию конституент нуля.

Конституента нуля – набор (конъюнкция переменных в прямой или инверсной форме), на котором функция принимает нулевые значения.

СДНФ и СКНФ легко сформировать по таблице истинности логической функции. Например, таблицей истинности заданы две логические функции y_1 и y_2 трех аргументов, зависящие от одних и тех же переменных x_1, x_2 и x_3 (таблица 1.8).

Таблица 1.8 – Таблица истинности булевых функций $y_1(x_1, x_2, x_3)$ и $y_2(x_1, x_2, x_3)$

Номер набора	**Значения аргументов**			**Значения функций**	
	x_1	x_2	x_3	y_1	y_2
0	0	0	0	1	1
1	0	0	1	1	0
2	0	1	0	0	1
3	0	1	1	1	1
4	1	0	0	1	1
5	1	0	1	0	0
6	1	1	0	0	1
7	1	1	1	0	0

СДНФ таких логических функций имеет вид

$y_1 = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2x_3 + x_1\bar{x}_2\bar{x}_3$;

$y_2 = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 + \bar{x}_1x_2x_3 + x_1x_2\bar{x}_3 + x_1\bar{x}_2\bar{x}_3$.

СКНФ таких логических функций имеет вид

$y_1 = (x_1 + \bar{x}_2 + x_3) \cdot (\bar{x}_1 + \bar{x}_2 + x_3) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_3) \cdot (\bar{x}_1 + x_2 + \bar{x}_3)$;

$y_2 = (x_1 + x_2 + \bar{x}_3) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_3) \cdot (\bar{x}_1 + x_2 + \bar{x}_3)$.

СДНФ и СКНФ взаимосвязаны. Можно переходить из одной канонической формы в другую и наоборот. Такой переход можно выполнить, составив по заданной СДНФ таблицу истинности для необходимой функции, а на основе полученной таблицы составить СКНФ. Помимо такого способа, данную задачу можно решить, используя правило де Моргана (подраздел 1.5).

1.4 Основные логические операции, функции и элементы

В основе электронных схем лежат логические элементы, которые, в свою очередь, реализуют некоторые операции алгебры логики или, другими словами, выполняют логические операции. Логические операции булевой алгебры являются основными строительными блоками цифровой логики и используются для обработки и манипулирования логическими значениями.

Среди всех булевых функций можно выделить две основные группы: функции одной переменной и функции многих переменных.

Функции одной переменной в булевой алгебре представляют собой выражения, зависящие от одной логической переменной, таких функций четыре (таблица 1.9):

– константа 0, или абсолютно ложная функция (независимо от значения переменной функция принимает значение, равное 0, $f_0 = 0$);

– тождественная функция (функция принимает значение, равное значению переменной, $f_1 = x$);

– функция отрицания (функция принимает значение, противоположное значению переменной, $f_2 = \bar{x}$);

– константа 1 или абсолютно истинная функция (независимо от значения переменной функция принимает значение, равное 1, $f_3 = 1$).

Таблица 1.9 – Функции одной переменной

Аргумент	**Значения функций одной переменной**			
x	f_0	f_1	f_2	f_3
0	0	0	1	1
1	0	1	0	1

Функции многих переменных в булевой алгебре представляют собой выражения, зависящие от нескольких логических переменных.

В таблице 1.10 представлены всевозможные варианты функции двух переменных.

Таблица 1.10 – Варианты функции двух переменных

Номер набора	**Аргументы**		**Значения функций двух переменных**															
	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

Описание функций двух переменных представлено в таблице 1.11.

Таблица 1.11 – Описание вариантов функций двух переменных

Функция f_i	**Название функции**	**Чтение функции**	**Аналитическая запись**
f_0	Константа нуля	Тождественный нуль	0
f_1	Конъюнкция	И x_1, и x_2	x_1x_2; $x_1 \,\&\, x_2$; $x_1 \wedge x_2$
f_2	Запрет по x_2	Неверно, что если x_1, то x_2	$x_1\overline{x_2}$
f_3	$f(x_1)$	Функция одной переменной	x_1
f_4	Запрет по x_1	Неверно, что если x_2, то x_1	$\overline{x_1}x_2$
f_5	$f(x_2)$	Функция одной переменной	x_2
f_6	Функция неравнозначности (сложение по модулю 2)	x_1 не равно x_2	$\overline{x_1}x_2 + x_1\overline{x_2}$; $x_1 \oplus x_2$
f_7	Дизъюнкция	Или x_1, или x_2	$x_1 + x_2$; $x_1 \vee x_2$
f_8	Функция Пирса (стрелка Пирса, ИЛИ-НЕ, функция Вебба)	Ни x_1, ни x_2	$\overline{x_1 + x_2}$; $x_1 \downarrow x_2$
f_9	Функция равнозначности	x_1 равно x_2	$\overline{x_1x_2} + x_1x_2$; $x_1 \equiv x_2$
f_{10}	$f(x_2)$	Функция одной переменной	$\overline{x_2}$
f_{11}	Импликация («Если-то»)	Если x_2, то x_1	$\overline{x_1x_2} + x_1$; $x_2 \rightarrow x_1$
f_{12}	$f(x_1)$	Функция одной переменной	$\overline{x_1}$
f_{13}	Импликация («Если-то»)	Если x_1, то x_2	$\overline{x_1x_2} + x_2$; $x_1 \rightarrow x_2$
f_{14}	Функция Шеффера (штрих Шеффера, И-НЕ)	Неверно, что и x_1, и x_2	$\overline{x_1x_2}$; x_1/x_2
f_{15}	Константа единицы	Тождественная единица	1

В булевой алгебре возведение в степень и извлечение корня являются вырожденными операциями. При возведении в степень или извлечении корня в булевой алгебре значения аргументов остаются неизменными, поскольку умножение на единицу или нуль не изменяет значение. Это происходит из-за того, что в булевой алгебре произведение единицы на любое число равно этому числу, а произведение нуля на любое число равно нулю. Следовательно, при возведении в степень или извлечении корня независимо от значения аргументов результатом будет значение аргумента.

Например, $1^2 = 1$ и $0^2 = 0$, а также $\sqrt{1} = 1$ и $\sqrt{0} = 0$.

Операции вычитания и деления не рассматриваются в булевой алгебре, т. к. они не имеют смысла в контексте логических переменных, принимающих только значение нуля или единицы.

Основные логические операции булевой алгебры

Основными логическими операциями, используемыми как в языках программирования высокого уровня (в поразрядной арифметике), так и в логике, являются: логическое сложение (операция ИЛИ (OR), дизъюнкция), логическое умножение (операция И (AND), конъюнкция), логическое отрицание (операция НЕ (NOT), инверсия, поразрядное дополнение) и логическая операция исключающего ИЛИ (операция XOR, сложение по модулю 2). Перечисленные логические операции реализуются соответствующими логическими функциями, которые имеют предопределенный результат, зависящий от действующих значений аргумента (для функций одного аргумента) или определенной комбинации аргументов (для функций нескольких аргументов).

Логический элемент – это элемент, реализующий определенную логическую зависимость между входными и выходными сигналами. Логические элементы используются для построения логических схем цифровых устройств. Для всех видов логических элементов независимо от их физической природы характерны дискретные значения входных и выходных сигналов.

Логическое сложение также называется дизъюнкцией, логическим ИЛИ.

Операцию логического сложения реализует логическая функция ИЛИ, являющаяся функцией двух аргументов.

Возможные варианты обозначения логического сложения (функции ИЛИ) и примеры записи представлены в таблице 1.12.

Таблица 1.12 – Варианты обозначения логического сложения

Обозначение	**В аналитических выражениях**		**В языках программирования**	
	Вариант 1	**Вариант 2**	**Delphi**	**C/C++**
	+	∨	OR	||
Пример использования	$y = x_1 + x_2$	$y = x_1 \vee x_2$	`y:= x1 or x2;`	`y = x1 || x2;`

В зависимости от аргументов функции логического сложения (входных данных) сама функция будет принимать значения (выходные данные), представленные в таблице 1.13. Такая таблица называется таблицей истинности функции ИЛИ (логического сложения).

Таблица 1.13 – Таблица истинности функции логического сложения (ИЛИ)

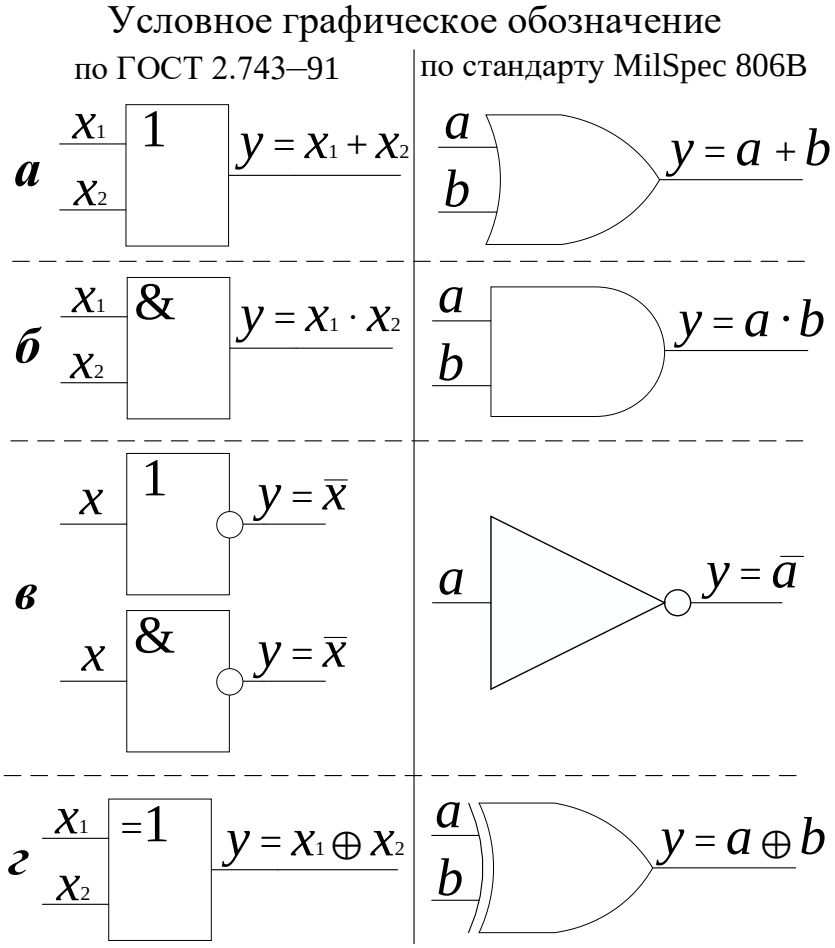
Значения аргументов		**Значение функции ИЛИ**
x_1	x_2	
0	0	0
0	1	1
1	0	1
1	1	1

Как видно из таблицы 1.13, для того чтобы функция приняла значение «истина» (единица), достаточно, чтобы хотя бы один из двух аргументов функции имел значение «истина» (единица): или первый аргумент, или второй аргумент, или оба. Отсюда и вытекает название функции: *ИЛИ*.

На схемах логическое сложение (функция ИЛИ) изображается логическим элементом ИЛИ (рисунок 1.2, *а*). Условные графические обозначения логических элементов выполняются по ГОСТ 2.743–91 или стандарту MilSpec 806B.

Технически элемент ИЛИ может быть реализован, например, на базе двух МОП-транзисторов, подключенных параллельно (рисунок 1.3).

Как видно из рисунка 1.3, при отсутствии сигналов (нули) на обоих затворах транзисторы будут закрыты и на их стоках сигнала не будет (нуль). При подаче сигнала (единица) на затвор одного из транзисторов – транзистор будет открыт, а при отсутствии сигнала (нуль) на другом – он будет закрыт, но при этом на стоке будет сигнал (единица), поскольку ток будет течь через открытый транзистор. При подаче сигнала (единица) на затворы обоих транзисторов оба транзистора будут открыты и на стоке также будет сигнал, т. к. ток будет течь через оба открытых транзистора.



а – элемент ИЛИ; *б* – элемент И; *в* – элемент НЕ; *г* – элемент исключающее ИЛИ

Рисунок 1.2 – Изображение основных логических элементов на схемах

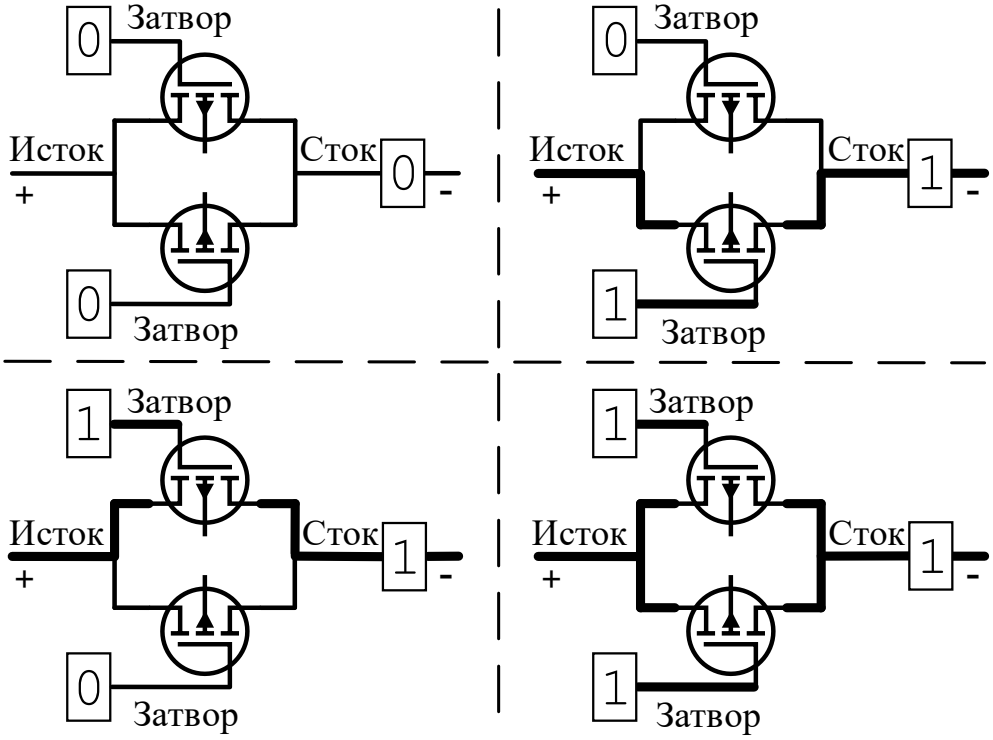


Рисунок 1.3 – Работа элемента ИЛИ на базе параллельных МОП-транзисторов

Логическое умножение также называется конъюнкцией, логическим И.

Операцию логического умножения реализует логическая функция И, являющаяся функцией двух аргументов.

Возможные варианты обозначения логического умножения (функции И) и примеры записи представлены в таблице 1.14.

Таблица 1.14 – Варианты обозначения логического умножения

Обозначение	**В аналитических выражениях**		**В языках программирования**	
	Вариант 1	**Вариант 2**	**Delphi**	**C/C++**
	·	∧	AND	&&
Пример использования	$y = x_1 \cdot x_2$ $y = x_1 \times x_2$ $y = x_1 x_2$	$y = x_1 \wedge x_2$	`y:= x1 and x2;`	`y = x1 && x2;`

В зависимости от аргументов функции логического умножения (от входных данных) сама функция будет принимать значения (выходные данные), представленные в таблице 1.15. Такая таблица называется таблицей истинности функции И (логического умножения).

Таблица 1.15 – Таблица истинности функции логического умножения (И)

Значения аргументов		**Значение функции И**
x_1	x_2	
0	0	0
0	1	0
1	0	0
1	1	1

Как видно из таблицы 1.15, для того чтобы функция приняла значение «истина» (единица), необходимо, чтобы каждый из двух аргументов функции имел значение «истина» (единица): и первый аргумент, и второй аргумент. Отсюда и вытекает название функции: *И*.

На схемах логическое умножение (функция И) изображается логическим элементом И (см. рисунок 1.2, *б*). Условные графические обозначения логических элементов выполняются по ГОСТ 2.743–91 или стандарту MilSpec 806B.

Технически элемент И может быть реализован, например, на базе двух МОП-транзисторов, подключенных последовательно (рисунок 1.4).

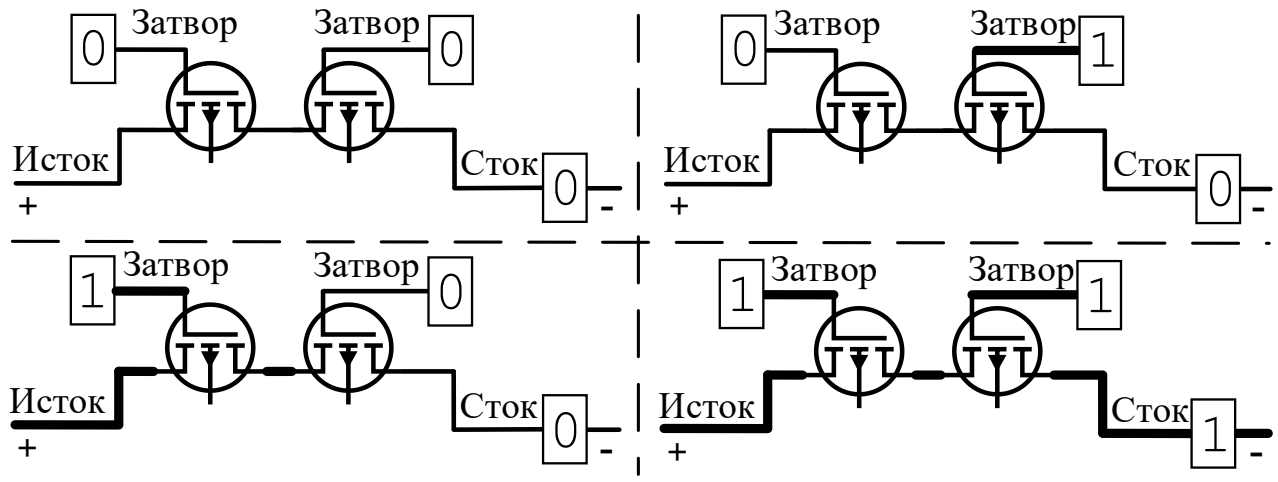


Рисунок 1.4 – Работа элемента И на базе последовательных МОП-транзисторов

Как видно из рисунка 1.4, при отсутствии сигналов (нули) на обоих затворах транзисторы будут закрыты и на их стоках сигнала не будет (нуль). При подаче сигнала (единица) на затвор одного из транзисторов этот транзистор будет открыт, а при отсутствии сигнала (нуль) на другом – закрыт, и поскольку они подключены последовательно, на стоке заключительного транзистора сигнала не будет (нуль), т. к. ток не будет течь через один из закрытых транзисторов. При подаче сигнала (единица) на затворы обоих транзисторов оба транзистора будут открыты и на стоке заключительного транзистора также будет сигнал, т. к. ток будет течь через оба открытых транзистора.

Логическое отрицание также называется инверсией, логическим НЕ.

Операцию логического отрицания реализует логическая функция НЕ, являющаяся функцией одного аргумента.

Возможные варианты обозначения логического отрицания (функции НЕ) и примеры записи представлены в таблице 1.16.

Таблица 1.16 – Варианты обозначения логического отрицания

Обозначение	**В аналитических выражениях**	**В языках программирования**	
		Delphi	**C/C++**
	$\bar{x}$	not	!
Пример использования	$y = \bar{x}$	`y:= not x;`	`y = !x;`

В зависимости от аргумента функции логического отрицания (от входных данных) сама функция будет принимать значения (выходные данные), представленные в таблице 1.17. Такая таблица называется таблицей истинности функции НЕ (логического отрицания).

Таблица 1.17 – Таблица истинности функции логического отрицания (НЕ)

Значение аргумента *x*	**Значение функции НЕ**
0	1
1	0

Как видно из таблицы 1.17, для того чтобы функция приняла значение «истина» (единица), необходимо, чтобы аргумент функции имел значение «ложь» (нуль), а для того чтобы функция приняла значение «ложь» (нуль), необходимо, чтобы аргумент функции имел значение «истина» (единица). То есть функция инвертирует значение аргумента.

На схемах логическое отрицание (функция НЕ) изображается логическим элементом НЕ (см. рисунок 1.2, *в*). Условные графические обозначения логических элементов выполняются по ГОСТ 2.743–91 или стандарту MilSpec 806B.

Технически элемент НЕ может быть реализован, например, на базе одного МОП-транзистора, подключенного параллельно выходу (рисунок 1.5).

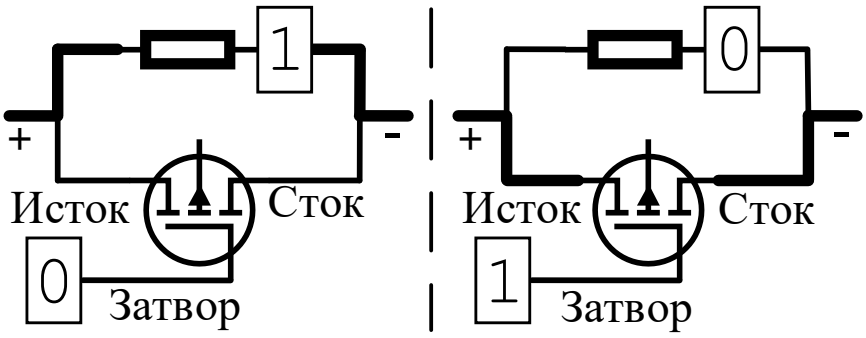


Рисунок 1.5 – Работа элемента НЕ на базе МОП-транзистора

Как видно из рисунка 1.5, при отсутствии сигналов (нуль) на затворе транзистор будет закрыт, ток потечет через параллельно подключенную нагрузку (выход) и сигнал на выходе будет (единица). При подаче сигнала (единица) на затвор транзистора этот транзистор будет открыт и ток будет течь через него, а не через нагрузку с большим сопротивлением, на выходе сигнала не будет (нуль).

Логическое исключающее сложение также называется логическим сложением по модулю 2, исключающим ИЛИ, XOR.

Операцию логического исключающего сложения реализует логическая функция исключающее ИЛИ, являющаяся функцией двух аргументов.

Возможные варианты обозначения логического исключающего сложения (функции XOR) и примеры записи представлены в таблице 1.18.

Таблица 1.18 – Варианты обозначения логического исключающего сложения

Обозначение	**В аналитических выражениях**	**В языках программирования**	
		Delphi	**C/C++**
	$\oplus$	XOR	*Такого оператора нет*
Пример использования	$y = x_1 \oplus x_2$	`y:= x1 xor x2;`	Реализуется комбинацией других функций: `y =(x1 || x2) && !(x1 && x2);`

В зависимости от аргументов функции логического исключающего сложения (входных данных) сама функция будет принимать значения (выходные данные), представленные в таблице 1.19. Такая таблица называется таблицей истинности функции исключающего ИЛИ (логического исключающего сложения).

Таблица 1.19 – Таблица истинности функции логического исключающего ИЛИ

Значения аргументов		**Значение функции исключающее ИЛИ**
x_1	x_2	
0	0	0
0	1	1
1	0	1
1	1	0

Как видно из таблицы 1.19, для того чтобы функция приняла значение «истина» (единица), необходимо, чтобы только один из двух аргументов функции имел значение «истина» (единица): или первый аргумент, или второй аргумент, комбинация, когда оба аргумента истинны, исключается. Отсюда и вытекает название функции: *исключающее ИЛИ*.

На схемах логическое исключающее ИЛИ (функция XOR) изображается логическим элементом XOR (см. рисунок 1.2, *г*). Условные графические обозначения логических элементов выполняются по ГОСТ 2.743–91 или стандарту MilSpec 806B.

Технически элемент исключающее ИЛИ может быть реализован, например, на логических элементах И, ИЛИ и НЕ, подключенных, как показано на рисунке (рисунок 1.6).

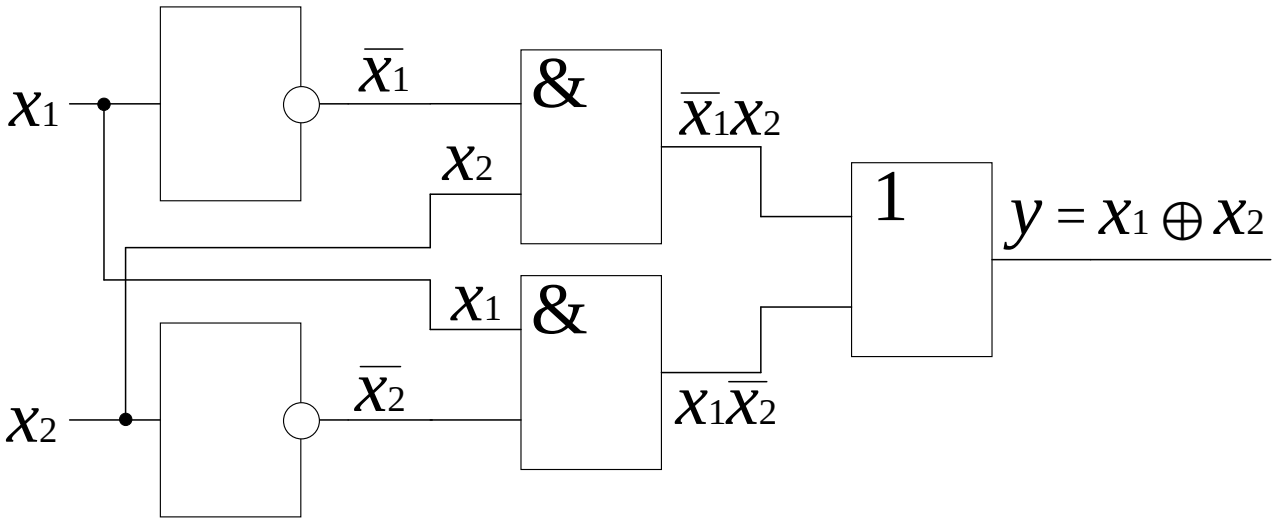


Рисунок 1.6 – Реализация логического элемента исключающее ИЛИ

При последовательном подключении логических элементов ИЛИ и НЕ можно получить логический **элемент ИЛИ-НЕ**, также называемый **NOR** и реализующий функцию Пирса (функцию ИЛИ-НЕ, отрицание дизъюнкции).

В зависимости от аргументов функции ИЛИ-НЕ (входных данных) сама функция будет принимать значения (выходные данные), представленные в таблице 1.20. Такая таблица называется таблицей истинности функции ИЛИ-НЕ.

Таблица 1.20 – Таблица истинности логической функции ИЛИ-НЕ

Значения аргументов		**Значение функции ИЛИ-НЕ**
x_1	x_2	
0	0	1
0	1	0
1	0	0
1	1	0

Как видно из таблицы 1.20, для того чтобы функция приняла значение «истина» (единица), необходимо, чтобы оба аргумента функции одновременно имели значение «ложь» (нуль). В остальных случаях функция ИЛИ-НЕ будет иметь значение «ложь» (нуль).

На схемах логический элемент ИЛИ-НЕ изображается, как показано на рисунке 1.7, *а*.

При последовательном подключении логических элементов И и НЕ можно получить логический **элемент И-НЕ**, также называемый **NAND** и реализующий функцию Шеффера (функцию И-НЕ, отрицание конъюнкции).

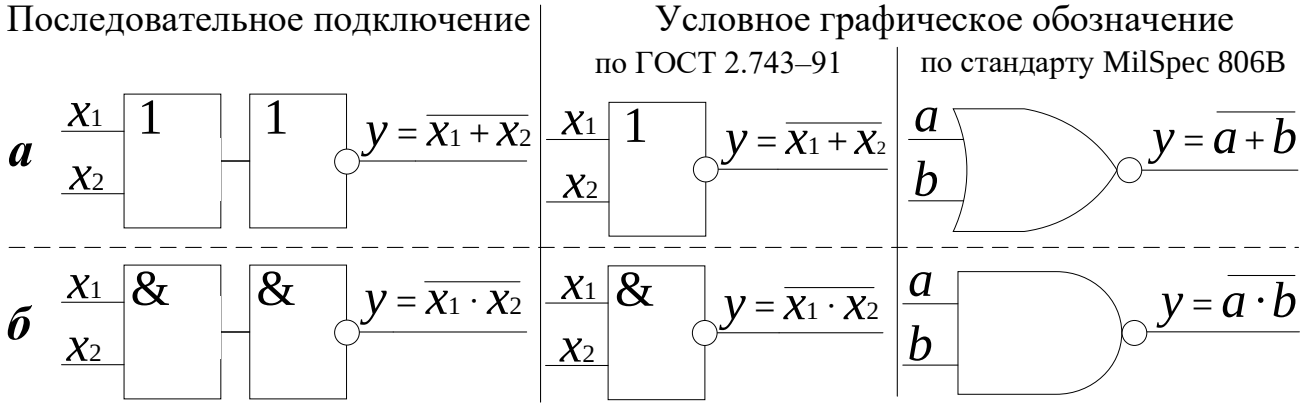
В зависимости от аргументов функции И-НЕ (входных данных) сама функция будет принимать значения (выходные данные), представленные в таблице 1.21. Такая таблица называется таблицей истинности функции И-НЕ.

Таблица 1.21 – Таблица истинности логической функции И-НЕ

Значения аргументов		**Значение функции И-НЕ**
x_1	x_2	
0	0	1
0	1	1
1	0	1
1	1	0

Как видно из таблицы 1.21, для того чтобы функция приняла значение «ложь» (нуль), необходимо, чтобы оба аргумента функции одновременно имели значение «истина» (единица). В остальных случаях функция И-НЕ будет иметь значение «истина» (единица).

На схемах логический элемент И-НЕ изображается, как показано на рисунке 1.7, *б*.



а – элемент ИЛИ-НЕ; *б* – элемент И-НЕ

Рисунок 1.7 – Изображение логических элементов ИЛИ-НЕ, И-НЕ на схемах

Импликация – это логическая операция, которая определяет логическое следствие между двумя высказываниями. Формально импликация выражает связь «если-то» и возвращает «ложь» только в случае, если предпосылка истинна, а заключение ложно. Операция импликации обозначается символом «стрелка вправо» (→). Таблица истинности функции импликации $f = x_1 \to x_2$ представлена в таблице 1.22.

Таблица 1.22 – Таблица истинности логической функции импликации

Значения аргументов		**Значение функции импликации**
x_1	x_2	
0	0	1
0	1	1
1	0	0
1	1	1

Функции двух переменных $f_3(x_1, x_2)$, $f_5(x_1, x_2)$, $f_{10}(x_1, x_2)$, $f_{12}(x_1, x_2)$ (см. таблицу 1.11) фактически зависят только от одной переменной. Логические переменные могут быть действительными или фиктивными (существенными или несущественными).

Переменная x_i действительна, если значение функции $f(x_1, \dots, x_i, \dots, x_n)$ изменяется при изменении значения x_i.

Переменная x_i фиктивна, если значение функции $f(x_1, \dots, x_i, \dots, x_n)$ не изменяется при изменении значения x_i.

Таким образом, для функций $f_3(x_1, x_2)$, $f_{12}(x_1, x_2)$ переменная x_2 будет фиктивной, x_1 – действительной, по аналогии для $f_5(x_1, x_2)$, $f_{10}(x_1, x_2)$: x_1 – фиктивная, x_2 – действительная. Для функций $f_0(x_1, x_2)$, $f_{15}(x_1, x_2)$ обе переменные x_1, x_2 являются фиктивными. Для остальных функций переменные x_1, x_2 являются действительными.

В данном подразделе рассмотрены наиболее часто встречаемые операции булевой алгебры на примере функций одной и двух переменных. В таблице 1.10 представлены выражения для всевозможных функций двух переменных. Все функции в таблице 1.10 являются элементарными, т. е. выполняются над одной или несколькими булевыми переменными. Эти функции являются основой для создания более сложных логических выражений и цифровых схем.

1.5 Основные законы булевой алгебры

Рассмотренные элементарные функции (отрицания, дизъюнкции, конъюнкции, импликации, суммы по модулю 2 и др.) имеют определенную взаимосвязь друг с другом. Эти связи можно описать с помощью свойств или законов булевой алгебры.

Основные законы булевой алгебры представляют собой фундаментальные правила, которые определяют, как выполняются логические операции над булевыми значениями.

Аксиомы для логических переменных (аксиомы)

Предположим, что x – некоторая логическая переменная. Тогда справедливы следующие аксиомы:

1) $x = \bar{\bar{x}}$ – правило двойного отрицания, исходя из которого появляется возможность исключения из логического выражения всех членов, имеющих двойное отрицание, заменив их исходной величиной;

2) $x + x + \ldots + x = x$;
   $x \cdot x \cdot \ldots \cdot x = x$;

Правила подобных преобразований, которые позволяют сокращать длину логических выражений

3) $x + 0 = x$;
4) $x + 1 = 1$;
5) $x \cdot 0 = 0$;
6) $x \cdot 1 = x$;

Правила операций с константами

7) $x \cdot \bar{x} = 0$;
8) $x + \bar{x} = 1$.

Правила операций с отрицаниями

Правила (законы) для операций логического сложения и умножения

Логические операции умножения и сложения также обладают рядом свойств, схожих со свойствами обычных арифметических операций:

1) свойство ассоциативности (сочетательный закон):

$x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3$;

$x_1 \cdot (x_2 \cdot x_3) = (x_1 \cdot x_2) \cdot x_3$;

2) свойство коммутативности (переместительный закон):

$x_1 + x_2 = x_2 + x_1$;

$x_1 \cdot x_2 = x_2 \cdot x_1$;

3) свойство дистрибутивности (распределительный закон):

а) для конъюнкции относительно дизъюнкции:

$x_1 \cdot (x_2 + x_3) = (x_1 \cdot x_2) + (x_2 \cdot \overline{x_3})$;

б) для дизъюнкции относительно конъюнкции:

$x_1 + x_2 \cdot x_3 = (x_1 + x_2) \cdot (x_1 + x_3)$.

Данные свойства демонстрируют правила раскрытия скобок при вычислении значения логического выражения и доказываются с помощью рассмотренных ранее аксиом либо с помощью таблиц истинности.

В таблице 1.23 проиллюстрировано доказательство справедливости соотношения свойства ассоциативности для дизъюнкции. Доказательство остальных соотношений аналогично.

Теперь докажем соотношение свойства дистрибутивности для дизъюнкции относительно конъюнкции:

$x_1 + x_2 \cdot x_3 = (x_1 + x_2) \cdot (x_1 + x_3)$.

Получаем, что $(x_1 + x_2) \cdot (x_1 + x_3) = x_1 \cdot x_1 + x_1 \cdot x_3 + x_1 \cdot x_2 + x_2 \cdot x_3 =$
$= x_1 + x_1 \cdot x_3 + x_1 \cdot x_2 + x_2 \cdot x_3 = x_1 \cdot (1 + x_3 + x_2) + x_2 \cdot x_3,$
$1 + x_3 + x_2 = 1$ – это следует из четвертой аксиомы, поэтому
$x_1 \cdot (1 + x_3 + x_2) + x_2 \cdot x_3 = x_1 + x_2 \cdot x_3.$

Таблица 1.23 – Доказательство справедливости свойства ассоциативности

Номер набора	x_1	x_2	x_3	$x_1 + x_2$	$x_2 + x_3$	$(x_1 + x_2) + x_3$	$x_1 + (x_2 + x_3)$
0	0	0	0	0	0	0	0
1	0	0	1	0	1	1	1
2	0	1	0	1	1	1	1
3	0	1	1	1	1	1	1
4	1	0	0	1	0	1	1
5	1	0	1	1	1	1	1
6	1	1	0	1	1	1	1
7	1	1	1	1	1	1	1

4) законы де Моргана:

$\overline{x_1 x_2} = \overline{x_1} + \overline{x_2}$;

$\overline{x_1 + x_2} = \overline{x_1} \cdot \overline{x_2}$.

Доказать справедливость соотношений законов де Моргана можно с помощью таблицы истинности (таблица 1.24).

Таблица 1.24 – Доказательство справедливости законов де Моргана

Номер набора	x_1	x_2	$x_1 x_2$	$\overline{x_1 x_2}$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_1} + \overline{x_2}$	$x_1 + x_2$	$\overline{x_1 + x_2}$	$\overline{x_1} \cdot \overline{x_2}$
0	0	0	0	1	1	1	1	0	1	1
1	0	1	0	1	1	0	1	1	0	0
2	1	0	0	1	0	1	1	1	0	0
3	1	1	1	0	0	0	0	1	0	0

Из таблицы 1.24 видно, что векторы функции для левой и правой частей соотношений законов де Моргана совпадают.

Из законов де Моргана вытекает возможность выразить конъюнкцию через отрицание и дизъюнкцию, а дизъюнкцию – через операции отрицания и конъюнкцию:

$x_1 \cdot x_2 = \overline{\overline{x_1 \cdot x_2}} = \overline{\overline{x_1} + \overline{x_2}}$;

$x_1 + x_2 = \overline{\overline{x_1 + x_2}} = \overline{\overline{x_1} \cdot \overline{x_2}}$.

Законы де Моргана справедливы для любого количества операндов;

5) закон поглощения:

$x_1 + x_1x_2 = x_1;$

$x_1(x_1 + x_2) = x_1.$

Доказать соотношение

$x_1 + x_1x_2 = x_1$

можно следующим образом. Вынесем x_1 за скобки:

$x_1 + x_1x_2 = x_1(1 + x_2).$

Выражение $1 + x_2$, согласно рассмотренным аксиомам, всегда принимает значение 1, точно так же, как выражение $x_1 \cdot 1$ принимает значение, равное x_1, поэтому $x_1 + x_1x_2 = x_1(1 + x_2) = x_1 \cdot 1 = x_1$. Второе соотношение закона поглощения доказывается аналогичным образом;

6) правило склеивания:

$x_1x_2 + x_1\overline{x_2} = x_1.$

Доказать данное правило можно с помощью рассмотренных свойств и аксиом булевой алгебры. Для этого применим закон дистрибутивности для левой части выражения, а затем упростим полученный результат, используя аксиомы 8 и 6:

$x_1x_2 + x_1\overline{x_2} = x_1(x_2 + \overline{x_2}) = x_1 \cdot 1 = x_1.$

Аксиомы для исключающего ИЛИ (сложения по модулю 2)

Для функции сложения по модулю 2 применимы рассмотренные ранее свойства, а также справедливы следующие аксиомы:

1) $x \oplus x = 0;\ \ x \oplus \bar{x} = 1;$

2) $x \oplus 1 = \bar{x};\ \ x \oplus 0 = x.$

На основании рассмотренных свойств и аксиом можно определить соотношения между операциями И, ИЛИ, НЕ и суммой по модулю 2:

$\overline{x_1} = x_1 \oplus 1;$

$x_1 + x_2 = x_1 \oplus x_2 \oplus x_1 \cdot x_2;$

$x_1 \cdot x_2 = (x_1 \oplus x_2) \oplus (x_1 + x_2).$

Аксиомы для функции импликации

Для функции импликации справедливы следующие аксиомы:

1) $x \to x = 1;$

2) $x \to 1 = 1;$

3) $0 \to x = 1;$

4) $x \to \bar{x} = \bar{x};$

5) $x \to 0 = \bar{x};$

6) $1 \to \bar{x} = x.$

Из аксиом следует, что для данной функции справедливо только свойство коммутативности, но в измененном виде:

$x_1 \to x_2 = \overline{x_2} \to \overline{x_1}$.

Свойство ассоциативности для функции импликации является несправедливым, о чем свидетельствует соответствующая таблица истинности (таблица 1.25).

Если выразить функции И, ИЛИ, НЕ через импликацию, то получатся следующие соотношения:

$x_1 + x_2 = \overline{x_1} \to x_2$;

$x_1 \cdot x_2 = \overline{x_1 \to \overline{x_2}}$;

$\overline{x_1} = x_1 \to 0$.

Таблица 1.25 – Доказательство несправедливости свойства ассоциативности для функции импликации

Номер набора	x_1	x_2	x_3	$x_1 \to x_2$	$x_2 \to x_3$	$(x_1 \to x_2) \to x_3$	$x_1 \to (x_2 \to x_3)$
0	0	0	0	1	1	0	1
1	0	0	1	1	1	1	1
2	0	1	0	1	0	0	1
3	0	1	1	1	1	1	1
4	1	0	0	0	1	1	1
5	1	0	1	0	1	1	1
6	1	1	0	1	0	0	0
7	1	1	1	1	1	1	1

Из таблицы 1.25 (столбцы 7, 8) видно, что векторы функций $(x_1 \to x_2) \to x_3$ и $x_1 \to (x_2 \to x_3)$ не совпадают.

Аксиомы для функции Шеффера

Функция Шеффера может быть выражена с помощью операций И, НЕ следующим образом:

$x_1/x_2 = \overline{x_1 x_2}$

Для этой функции свойственны аксиомы:

1) $x/x = \bar{x}$;

2) $x/\bar{x} = 1$;

3) $x/0 = 1$;

4) $x/1 = \bar{x}$;

5) $\bar{x}/0 = 1$;

6) $\bar{x}/1 = x$.

Помимо перечисленных аксиом, для функции справедливо только свойство коммутативности:

$x_1/x_2 = x_2/x_1$.

Докажем неприменимость свойства ассоциативности для функции Шеффера на примере функции трех переменных.

Приведем выражение $(x_1/x_2)/x_3$ к виду, записанному с помощью операций И, НЕ:

$(x_1/x_2)/x_3 = \overline{(\overline{x_1x_2})x_3}$.

Применим правило де Моргана, получим

$(x_1/x_2)/x_3 = \overline{(\overline{x_1x_2})x_3} = \overline{(\overline{x_1x_2})} + \overline{x_3} = x_1x_2 + \overline{x_3}$.

Аналогичные действия проведем с выражением $x_1/(x_2/x_3)$:

$x_1/(x_2/x_3) = \overline{x_1(\overline{x_2x_3})} = \overline{x_1} + x_2x_3$.

Получаем $\overline{x_1} + x_2x_3 \neq x_1x_2 + \overline{x_3}$, из чего следует, что

$(x_1/x_2)/x_3 \neq x_1/(x_2/x_3)$, что в свою очередь значит, что свойство ассоциативности несправедливо для функции трех и более переменных.

Свойство дистрибутивности также неприменимо для функции Шеффера. В качестве доказательства рассмотрим следующие примеры:

$x_1/(x_2/x_3) = \overline{x_1(\overline{x_2x_3})} = \overline{x_1} + x_2x_3 = (\overline{x_1} + x_2)(\overline{x_1} + x_3) =$

$= (\overline{\overline{x_1} + x_2})(\overline{\overline{x_1} + x_3}) = \overline{x_1\overline{x_2}} \cdot \overline{x_1\overline{x_3}} = x_1/\overline{x_2} \cdot x_1/\overline{x_3} =$

$= \overline{\overline{x_1/\overline{x_2} \cdot x_1/\overline{x_3}}} = (x_1/\overline{x_2}/x_1/\overline{x_3})/1 = (x_1/(x_2/1)/x_1/(x_3/1))/1;$

$(x_1/x_2)/(x_1/x_3) = \overline{(\overline{x_1x_2})(\overline{x_1x_3})} = \overline{\overline{x_1x_2}} + \overline{\overline{x_1x_3}} = x_1x_2 + x_1x_3 =$

$= x_1(x_2 + x_3) = x_1(\overline{\overline{x_2 + x_3}}) = x_1(\overline{\overline{x_2} \cdot \overline{x_3}}) = x_1(\overline{x_2}/\overline{x_3}) =$

$= x_1((x_2/1)/(x_3/1)) = \overline{\overline{x_1((x_2/1)/(x_3/1))}} =$

$= \overline{x_1/((x_2/1)/(x_3/1))} = (x_1/((x_2/1)/(x_3/1)))/1.$

На основании данных примеров можно сделать вывод, что для функции трех и более переменных свойство дистрибутивности неприменимо.

Применяя известные аксиомы и свойства, можно записать основные операции И, ИЛИ, НЕ посредством функции Шеффера:

$x_1x_2 = \overline{\overline{x_1/x_2}} = x_1/x_2/x_1/x_2;$

$\bar{x} = x/x;$

$x_1 + x_2 = \overline{\overline{x_1} \cdot \overline{x_2}} = \overline{x_1}/\overline{x_2} = x_1/x_1/x_2/x_2.$

Аксиомы для функции Пирса (Вебба)

Функцию Пирса (Вебба) можно описать выражением

$x_1 \downarrow x_2 = \overline{x_1 + x_2} = \overline{x_1} \cdot \overline{x_2}.$

Для функции Пирса актуальны следующие аксиомы:

1) $x \downarrow x = \bar{x}$;

2) $x \downarrow 0 = \bar{x}$;

3) $x \downarrow \bar{x} = 0$;

4) $x \downarrow 1 = 0$.

Исходя из аксиом, единственное свойство, которое применимо для функции Пирса – это свойство коммутативности:

$x_1 \downarrow x_2 = x_2 \downarrow x_1$.

Через функцию Пирса можно представить функции И, ИЛИ, НЕ:

$x_1 x_2 = (x_1 \downarrow x_1) \downarrow (x_2 \downarrow x_2)$;

$\bar{x} = (x \downarrow x)$;

$x_1 + x_2 = (x_1 \downarrow x_2) \downarrow (x_1 \downarrow x_2)$.

Рассмотренные аксиомы, свойства и законы играют важную роль в анализе, преобразовании и упрощении выражений в булевой алгебре.

1.6 Взаимосвязь форм записи логических функций

Основным понятием, лежащим в основе представления булевых функций в различных формах, является понятие терма.

Функцию $F_i(x_0, x_1, \dots, x_{n-1})$ называют термом, если она соответствует условию

$$F_i = \begin{cases} 0, \text{если номер набора равен } i; \\ 1, \text{если номер набора не равен } i, \end{cases}$$

где номер набора i – это произвольное двоичное число, соответствующее набору переменных $x_0, x_1, \dots, x_n$, для которого

$i = 2^{n-1}x_0 + 2^{n-2}x_1 + \dots + 2^0 x_{n-1}$.

Разделяют дизъюнктивный и конъюнктивный термы.

Дизъюнктивный терм, или *макстерм* – это терм, связывающий переменные булевой функции, представленные в прямой или инверсной форме, знаком дизъюнкции. Например, $F_2 = \overline{x_0} + x_1$, $F_{10} = \overline{x_0} + x_1 + \overline{x_2} + x_3$. Дизъюнктивный терм зачастую называют конституентой нуля.

Конъюнктивный терм, либо *минтерм* – это терм, связывающий переменные булевой функции, представленные в прямой или инверсной форме, знаком конъюнкции. Минтерм можно записать следующей условной функцией:

$$F_i = \begin{cases} 1, \text{если номер набора равен } i; \\ 0, \text{если номер набора не равен } i. \end{cases}$$

Например, $F_4 = x_0 \cdot \overline{x_1} \cdot \overline{x_2}$, $F_9 = x_0 \cdot \overline{x_1} \cdot \overline{x_2} \cdot x_3$. Конъюнктивный терм, содержащий все переменные функции, часто называют конституентой единицы.

Ранг терма равен количеству переменных, входящих в тот либо иной терм. Например, для минтерма $F_3 = x_0 x_1$ ранг равен 2, а для макстерма $F_9 = x_0 + \overline{x_1} + + \overline{x_2} + x_3$ ранг – 4.

Помимо термов, в основе представления функций в различных формах встречаются понятия элементарной конъюнкции и элементарной дизъюнкции.

Элементарная дизъюнкция – это логическая сумма любого конечного количества различных между собой булевых переменных, представленных в прямой или инверсной форме. Если давать определение элементарной дизъюнкции через понятие дизъюнктивного терма, то элементарная дизъюнкция – это минтерм ранга, меньшего либо равного количеству переменных в функции.

Элементарная конъюнкция – это логическое произведение любого конечного количества различных между собой булевых переменных, представленных в прямой или инверсной форме. Аналогично элементарная конъюнкция – это макстерм ранга, меньшего либо равного количеству переменных в функции.

Любая таблично заданная функция алгебры логики (ФАЛ) может быть представлена аналитически в виде

$$f(x_0, x_1, \dots, x_{n-1}) = F_0 + F_1 + \dots + F_{n-1} = \bigvee_i F_i, \qquad (1.1)$$

где i – номера наборов, на которых функция принимает значение 1 (единичные наборы);

F_i – термы, соответствующие единичным наборам. При этом каждому набору i, для которого $f_i = 1$, соответствует элемент $F_i = 1$, а наборам i, на которых $f_i = 0$, не соответствует ни одного элемента $F_i = 1$.

Выражение (1.1) принято называть объединением термов.

Таким образом, используя данную терминологию, можно по-другому дать определение понятиям ДНФ, КНФ, СДНФ, СКНФ.

Дизъюнктивная нормальная форма (*ДНФ*) – это объединение термов, включающее минтермы переменного ранга. Другими словами, можно сказать, что ДНФ – это дизъюнкция конечного числа элементарных конъюнкций:

$$f(x_1, x_2, x_3, x_4)_{\text{ДНФ}} = \overline{x_1} x_2 x_3 + x_1 \overline{x_2} x_4 + x_1 x_2 x_3 \overline{x_4}.$$

Любая таблично заданная ФАЛ может быть представлена в аналитической форме:

$$f(x_0, x_1, \dots, x_{n-1}) = F_0 \cdot F_1 \cdot \dots \cdot F_{k-1},$$

где k – количество наборов, для которых $F = 0$.

Конъюнктивная нормальная форма (*КНФ*) – объединение термов, включающее в себя макстермы разных рангов. Или, другими словами, КНФ – это конъюнкция конечного числа элементарных дизъюнкций.

КНФ и ДНФ не дают однозначного представления функции, такое представление возможно только в совершенных нормальных формах.

Совершенная дизъюнктивная нормальная форма (*СДНФ*) – это такая ДНФ от n переменных, в которой все минтермы имеют ранг n.

В общем виде СДНФ записывается как

$$f(x_0, x_1, \dots, x_{n-1}) = \bigvee_i x_0^{a_{i,0}} \cdot x_1^{a_{i,1}} \cdot \dots \cdot x_{n-1}^{a_{i,n-1}},$$

при этом $x^0 = \bar{x};\ x^1 = x;\ \{a_{i,0}, a_{i,1}, \dots, a_{i,n-1}\}$ – двоичный набор с номером i, на котором значение функции равно 1.

Основные свойства СДНФ:

– в СДНФ нет двух одинаковых минтермов;

– в СДНФ ни один минтерм не содержит двух одинаковых переменных;

– в СДНФ ни один минтерм не содержит вместе с переменной и ее отрицание.

Таким образом, можно сформулировать правило для записи СДНФ по таблице истинности: для каждого набора переменных, на котором булева функция принимает единичное значение, записывается конъюнкция ранга n, и все эти конъюнкции объединяются знаками дизъюнкции; переменная имеет знак инверсии, если на соответствующем наборе имеет нулевое значение.

Например, необходимо представить функцию, заданную таблицей истинности (таблица 1.26) в СДНФ.

Таблица 1.26 – Функция для представления в СДНФ

Номер набора	**Значения аргументов**			**Значение функции**
	x_0	x_1	x_2	f
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Используя правило для записи СДНФ, получаем

$$f(x_1, x_2, x_3) = \overline{x_1} x_2 x_3 + x_1 \overline{x_2 x_3} + x_1 x_2 \overline{x_3}.$$

Совершенная конъюнктивная нормальная форма (*СКНФ*) – это такая КНФ от n переменных, в которой все макстермы имеют ранг n.

В общем виде СКНФ можно записать как

$$f(x_0, x_1, \dots, x_{n-1}) = \bigwedge_i x_0^{\overline{a_{i,0}}} + x_1^{\overline{a_{i,1}}} + \dots + x_{n-1}^{\overline{a_{i,n-1}}},$$

при этом $x^0 = \bar{x};\ x^1 = x;\ \{a_{i,0}, a_{i,1}, \dots, a_{i,n-1}\}$ – двоичный набор с номером i, на котором значение функции равно 0.

Правило записи СКНФ по таблице истинности можно записать так: для каждого набора переменных, на котором булева функция принимает нулевое значение, записывается дизъюнкция ранга n, и все эти дизъюнкции объединяются знаками конъюнкции; переменная имеет знак инверсии, если на соответствующем наборе имеет единичное значение.

Например, для функции заданной таблицей истинности (см. таблицу 1.26) СКНФ примет вид

$$f(x_1, x_2, x_3) = (\overline{x_1} + \overline{x_2} + \overline{x_3})(\overline{x_1} + \overline{x_2} + x_3) \times$$
$$\times (\overline{x_1} + x_2 + \overline{x_3})(x_1 + \overline{x_2} + x_3)(x_1 + x_2 + x_3).$$

Совершенная нормальная форма отличается от нормальной формы тем, что всегда содержит термы максимального ранга и дает однозначное представление функции.

Переход от нормальной формы к совершенной нормальной форме

Переход от нормальной формы к совершенной форме построен на основе правила склеивания. Например, пусть $f_{\text{ДНФ}} = F_1$, тогда

$$f_{\text{СДНФ}} = F_1 x_i + F_1 \overline{x_i},$$

где x_i – переменная, которой недостает в терме F_1.

Аналогично осуществляется переход от КНФ к СКНФ. Например, пусть $f_{\text{КНФ}} = F_1$, тогда

$$f_{\text{СКНФ}} = (F_1 + x_i)(F_1 + \overline{x_i}),$$

где x_i – переменная, которой недостает в терме F_1.

Пример

Логическая функция задана в ДНФ:

$$f(x_1, x_2, x_3, x_4) = x_1\overline{x_3}x_4 + x_2\overline{x_4} + \overline{x_1}x_2x_3x_4 + x_1\overline{x_2}x_4.$$

Необходимо преобразовать заданную функцию в СДНФ.

Применим преобразования, рассмотренные выше, к термам, ранг которых меньше 4.

$$F_1 = x_1\overline{x_3}x_4 = x_1\overline{x_3}x_4(x_2 + \overline{x_2}) = x_1x_2\bar{x}_3x_4 + x_1\bar{x}_2\bar{x}_3x_4;$$
$$F_2 = x_2\bar{x}_4 = x_2\bar{x}_4(x_1 + \bar{x}_1) = x_1x_2\bar{x}_4 + \bar{x}_1x_2\bar{x}_4 =$$
$$= x_1x_2\bar{x}_4(x_3 + \bar{x}_3) + \bar{x}_1x_2\bar{x}_4(x_3 + \bar{x}_3) =$$
$$= x_1x_2x_3\bar{x}_4 + x_1x_2\bar{x}_3\bar{x}_4 + \bar{x}_1x_2x_3\bar{x}_4 + \bar{x}_1x_2\bar{x}_3\bar{x}_4;$$
$$F_4 = x_1\overline{x_2}x_4 = x_1\overline{x_2}x_4(x_3 + \bar{x}_3) = x_1\bar{x}_2x_3x_4 + x_1\bar{x}_2\bar{x}_3x_4.$$

После приведения подобных членов определяем СДНФ:

$$f_{\text{СДНФ}}(x_1, x_2, x_3, x_4) = x_1x_2\bar{x}_3x_4 + x_1\bar{x}_2\bar{x}_3x_4 + x_1x_2x_3\bar{x}_4 +$$
$$+ x_1x_2\bar{x}_3\bar{x}_4 + \bar{x}_1x_2x_3\bar{x}_4 + \bar{x}_1x_2\bar{x}_3\bar{x}_4 + \overline{x_1}x_2x_3x_4 + x_1\bar{x}_2x_3x_4.$$

Пример

Необходимо преобразовать в СНКФ логическую функцию следующего вида:

$f(x_1, x_2, x_3) = (\bar{x}_1 + x_3)(x_2 + \bar{x}_3)(x_1 + \bar{x}_2 + x_3)$.

Применяем рассмотренные правила преобразований к термам F_1, F_2:

$F_1 = \bar{x}_1 + x_3 = (\bar{x}_1 + x_3) + x_2\bar{x}_2 = (\bar{x}_1 + x_2 + x_3)(\bar{x}_1 + \bar{x}_2 + x_3)$;

$F_2 = x_2 + \bar{x}_3 = (x_2 + \bar{x}_3) + x_1\bar{x}_1 = (x_1 + x_2 + \bar{x}_3)(\bar{x}_1 + x_2 + \bar{x}_3)$.

После упрощения функция в СКНФ примет вид

$f_{\text{СКНФ}}(x_1, x_2, x_3) = (\bar{x}_1 + x_2 + x_3)(\bar{x}_1 + \bar{x}_2 + x_3) \times$

$\times (x_1 + x_2 + \bar{x}_3)(\bar{x}_1 + x_2 + \bar{x}_3)(x_1 + \bar{x}_2 + x_3)$.

Переход из одной совершенной формы в другую

Формы представления функций СДНФ и СКНФ тесно связаны между собой. Переход от одной формы представления к другой возможен как через построение таблицы истинности, так и посредством преобразований в аналитическом представлении.

Для примера рассмотрим следующую функцию:

$f_{\text{СДНФ}}(x_1, x_2, x_3, x_4) = x_1x_2\bar{x}_3x_4 + x_1\bar{x}_2\bar{x}_3x_4 + x_1x_2x_3\bar{x}_4 + x_1x_2\bar{x}_3\bar{x}_4 +$

$+ \bar{x}_1x_2x_3\bar{x}_4 + \bar{x}_1x_2\bar{x}_3\bar{x}_4 + \overline{x_1}x_2x_3x_4 + x_1\bar{x}_2x_3x_4$;

Таблица истинности для данной функции представлена ниже (таблица 1.27).

Таблица 1.27 – Таблица истинности функции, заданной в СДНФ

Номер набора	**Значения аргументов**				**Значение функция**
	x_1	x_2	x_3	x_4	f
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0

Эту функцию можно представить словесно следующим образом: функция четырех переменных $f(x_1, x_2, x_3, x_4)$ принимает единичные значения на наборах {4, 6, 7, 9, 11, 12, 13, 14} или $f_{\text{СДНФ}}(x_1, x_2, x_3, x_4) = \{4, 6, 7, 9, 11, 12, 13, 14\}$.

По таблице 1.27 сразу можно записать СКНФ функции:

$$f_{\text{СКНФ}}(x_1, x_2, x_3, x_4) = (x_1 + x_2 + x_3 + x_4)(x_1 + x_2 + x_3 + \bar{x}_4) \times$$
$$\times (x_1 + x_2 + \bar{x}_3 + x_4)(x_1 + x_2 + \bar{x}_3 + \bar{x}_4)(x_1 + \bar{x}_2 + x_3 + \bar{x}_4) \times$$
$$\times (\bar{x}_1 + x_2 + x_3 + x_4)(\bar{x}_1 + x_2 + \bar{x}_3 + x_4)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4).$$

Выполним некоторые преобразования для функции $f_{\text{СДНФ}}$. Исходная функция представляет собой дизъюнкцию конституент единицы по определению. Следовательно, верно будет то, что дизъюнкция конституент нуля равна отрицанию исходной функции, т. е.

$$\bar{f}(x_1, x_2, x_3, x_4) = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 + \bar{x}_1\bar{x}_2\bar{x}_3x_4 + \bar{x}_1\bar{x}_2x_3\bar{x}_4 +$$
$$+ \bar{x}_1\bar{x}_2x_3x_4 + \bar{x}_1x_2\bar{x}_3x_4 + x_1\bar{x}_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2x_3\bar{x}_4 + x_1x_2x_3x_4.$$

Применим отрицание к левой и правой частям выражения:

$$\bar{\bar{f}}(x_1, x_2, x_3, x_4) = \overline{\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 + \bar{x}_1\bar{x}_2\bar{x}_3x_4 + \bar{x}_1\bar{x}_2x_3\bar{x}_4 +}$$
$$\overline{+ \bar{x}_1\bar{x}_2x_3x_4 + \bar{x}_1x_2\bar{x}_3x_4 + x_1\bar{x}_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2x_3\bar{x}_4 + x_1x_2x_3x_4}.$$

После упрощения получаем

$$f(x_1, x_2, x_3, x_4) = (\overline{\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4})(\overline{\bar{x}_1\bar{x}_2\bar{x}_3x_4})(\overline{\bar{x}_1\bar{x}_2x_3\bar{x}_4}) \times$$
$$\times (\overline{\bar{x}_1\bar{x}_2x_3x_4})(\overline{\bar{x}_1x_2\bar{x}_3x_4})(\overline{x_1\bar{x}_2\bar{x}_3\bar{x}_4})(\overline{x_1\bar{x}_2x_3\bar{x}_4})(\overline{x_1x_2x_3x_4}) =$$
$$= (x_1 + x_2 + x_3 + x_4)(x_1 + x_2 + x_3 + \bar{x}_4)(x_1 + x_2 + \bar{x}_3 + x_4) \times$$
$$\times (x_1 + x_2 + \bar{x}_3 + \bar{x}_4)(x_1 + \bar{x}_2 + x_3 + \bar{x}_4)(\bar{x}_1 + x_2 + x_3 + x_4) \times$$
$$\times (\bar{x}_1 + x_2 + \bar{x}_3 + x_4)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4).$$

Полученное выражение представляет собой не что иное, как СКНФ функции. Запись функции после аналитических преобразований полностью соответствует функции, записанной в форме СКНФ по таблице истинности.

1.7 Диаграммы Венна

Диаграммы Венна, названные по имени английского логика и философа Джона Венна, применявшего их в исследованиях по логике, являются графическим представлением, демонстрирующим соотношения между множествами. Диаграммы Венна часто используются в логике, математике, статистике, информатике и других областях. На основе соответствия между теорией булевой алгебры и теорией множеств диаграммы Венна очень полезны для визуального представления аксиом и законов булевой алгебры, однако их не следует использовать для построения доказательств тождеств, поскольку большинство общих положений эти диаграммы показать не могут. На рисунке 1.8 приведены диаграммы Венна для констант «0», «1» и элементарных функций логического умножения, сложения и инверсии (И, ИЛИ, НЕ). Область, ограниченная кругом на рисунке, соответствует одной переменной.

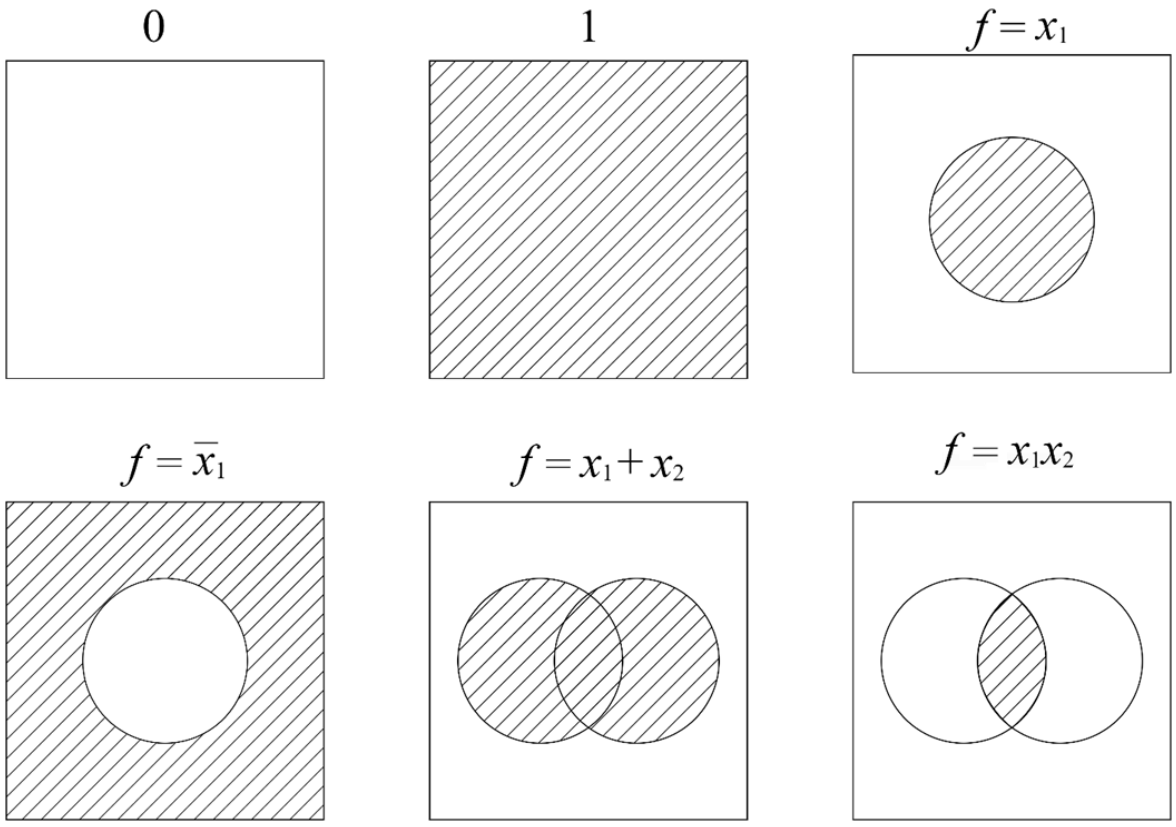


Рисунок 1.8 – Диаграммы Венна для функций 0, 1, И, ИЛИ, НЕ

На рисунке 1.9 представлена диаграмма Венна для функции $f = x_1x_2 + x_3$.

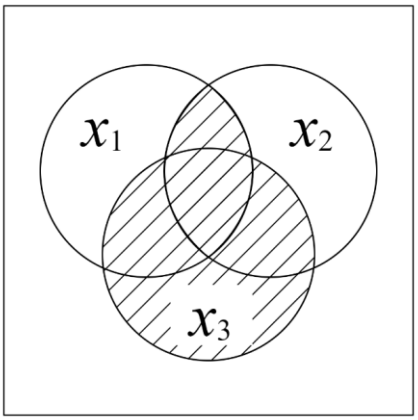


Рисунок 1.9 – Диаграмма Венна для функции $f = x_1x_2 + x_3$

Область пересечения трех кругов на диаграмме Венна (функция $f = x_1x_2x_3$) представляет собой геометрическую фигуру, известную как треугольник Рело (рисунок 1.10).

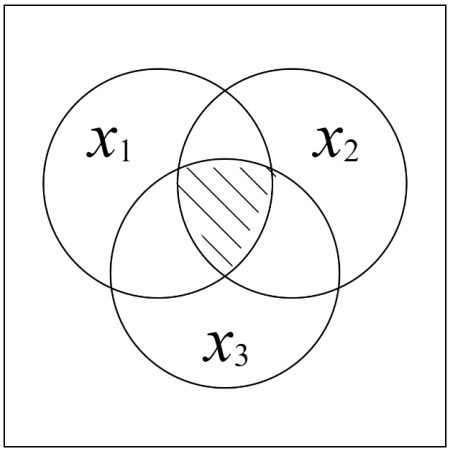


Рисунок 1.10 – Треугольник Рело

1.8 Карты Карно

Карта Карно для n логических переменных представляет собой прямоугольную таблицу, приближенную к форме квадрата. Каждая ячейка таблицы соответствует одному набору логических переменных. Две соседние ячейки должны соответствовать наборам, различающимся значениями одной переменной (рисунок 1.11). Карты Карно также используются для минимизации логических функций (подраздел 2.2).

$n = 1$

	$\overline{x}_1$	x_1

$n = 2$

	$\overline{x}_1$	x_1
$\overline{x}_2$		
x_2		

$n = 3$

	$\overline{x}_1\overline{x}_2$	$\overline{x}_1x_2$	x_1x_2	$x_1\overline{x}_2$
$\overline{x}_3$				
x_3				

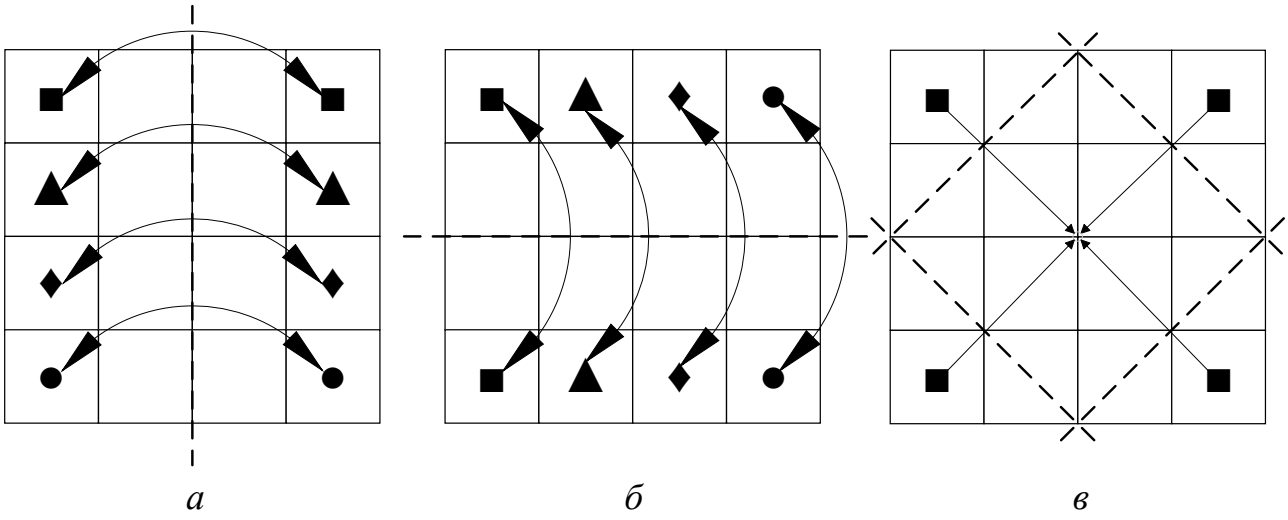
$n = 4$

	$\overline{x}_1\overline{x}_2$	$\overline{x}_1x_2$	x_1x_2	$x_1\overline{x}_2$
$\overline{x}_3\overline{x}_4$				
$\overline{x}_3x_4$				
x_3x_4				
$x_3\overline{x}_4$				

Рисунок 1.11 – Примеры карт Карно для функций одной, двух, трех и четырех переменных

Как видно из рисунка 1.11, в карте Карно ячейки располагаются не по порядку следования наборов, а так, чтобы каждая соседняя ячейка являлась логическим соседом.

Логические соседи – это такие две ячейки, наборы которых отличаются только одной переменной: в одной ячейке эта переменная должна иметь прямое значение, в другой – инверсное. Следует отметить, что логическими соседями также являются ячейки, располагающиеся в соответствующих строках крайнего левого столбца и крайнего правого столбца, а также в соответствующих столбцах крайней верхней строки и крайней нижней строки, поскольку карта Карно является трехмерным объектом. Примеры сворачивания карты Карно функции четырех переменных в «трубочку» по вертикали (рисунок 1.12, *а*) и горизонтали (рисунок 1.12, *б*), а также сворачивания в «конверт» (рисунок 1.12, *в*) представлены ниже.



а – сворачивание в «трубочку» по вертикальной линии изгиба;
б – сворачивание в «трубочку» по горизонтальной линии изгиба;
в – сворачивание в «конверт» по угловым линиям изгиба
Рисунок 1.12 – Примеры карты Карно функции четырех переменных как трехмерного объекта

Как видно из рисунка 1.12, ячейки, обозначенные одинаковыми геометрическими фигурами (квадрат, треугольник, ромб и круг), также являются логическими соседями.

Таким образом, если обозначить номера наборов в соответствии с обозначением переменных (прямым – x или инверсным – $\bar{x}$), можно заметить, что порядок следования имеет закономерность: одновременно меняются местами последняя строка с предпоследней и последний столбец с предпоследним (рисунок 1.13).

Рисунок 1.13 – Примеры карты Карно функции четырех переменных с проставленными номерами наборов

Для построения карты Карно исходная функция должна быть представлена в одной из канонических форм: совершенной конъюнктивной нормальной форме (СКНФ) либо совершенной дизъюнктивной нормальной форме (СДНФ).

СДНФ и СКНФ легко сформировать на основе таблицы истинности. Например, две функции трех переменных заданы на основе таблицы 1.28.

Таблица 1.28 – Таблица истинности функций трех переменных f_1 и f_2

Номер набора	**Значения аргументов**			**Значения функций**	
	x_1	x_2	x_3	f_1	f_2
0	0	0	0	1	1
1	0	0	1	1	0
2	0	1	0	0	1
3	0	1	1	1	1
4	1	0	0	1	1
5	1	0	1	0	0
6	1	1	0	0	1
7	1	1	1	0	0

СДНФ в таком случае примет вид

$f_1 = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2x_3 + x_1\bar{x}_2\bar{x}_3;$

$f_2 = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 + \bar{x}_1x_2x_3 + x_1x_2\bar{x}_3 + x_1\bar{x}_2\bar{x}_3.$

СКНФ примет следующий вид:

$f_1 = (x_1 + \bar{x}_2 + x_3) \cdot (\bar{x}_1 + \bar{x}_2 + x_3) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_3) \cdot (\bar{x}_1 + x_2 + \bar{x}_3);$

$f_2 = (x_1 + x_2 + \bar{x}_3) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_3) \cdot (\bar{x}_1 + x_2 + \bar{x}_3).$

Для различных форм записи СДНФ и СКНФ карты Карно заполняются по-разному.

Карта Карно для функции в СДНФ

В СДНФ функция представляет собой конституенты единицы. Рассмотрим функцию f_1 по конституентам. Первая конституента представляет собой набор $\bar{x}_1\bar{x}_2\bar{x}_3$, следовательно, в ячейку, соответствующую данному набору, необходимо установить 1. Те же действия проведем с остальными конституентами ($\bar{x}_1\bar{x}_2x_3$, $\bar{x}_1x_2x_3$ и $x_1\bar{x}_2\bar{x}_3$). Итоговая запись функции в карту Карно будет иметь вид, приведенный на рисунке 1.14.

	$\overline{X}_1\overline{X}_2$	$\overline{X}_1X_2$	X_1X_2	$X_1\overline{X}_2$
$\overline{X}_3$	1			1
X_3	1	1		

Рисунок 1.14 – Пример записи функции f_1 (в СДНФ) в карту Карно

Карта Карно для функции в СКНФ

В СДНФ функция представляет собой инверсию конституент нуля.

Рассмотрим функцию f_2 по конституентам. Первая конституента представляет собой набор $(x_1 + x_2 + \bar{x}_3)$, следовательно, в ячейку, соответствующую данному набору, необходимо установить 0. Те же действия проведем с остальными конституентами $((\bar{x}_1 + \bar{x}_2 + \bar{x}_3)$ и $(\bar{x}_1 + x_2 + \bar{x}_3))$. Итоговая запись функции в карту Карно будет иметь вид, приведенный на рисунке 1.15.

	$\overline{X}_1\overline{X}_2$	$\overline{X}_1X_2$	X_1X_2	$X_1\overline{X}_2$
$\overline{X}_3$	0	0	0	
X_3				

Рисунок 1.15 – Пример записи функции f_2 (в СКНФ) в карту Карно

1.9 Диаграммы Вейча

Диаграмма Вейча – это вид графического представления логической функции, в которой каждое логическое значение ставится в ячейку, расположение которой зависит от сочетания аргументов функции.

Кодовое расстояние – это количество отличающихся координат (переменных) между двумя различными наборами на диаграмме Вейча.

Переменные, обозначающие клетки диаграммы, расставляются таким образом, чтобы наборы, записанные в двух смежных клетках, имели кодовое расстояние, равное единице. В клетке диаграммы Вейча ставится единица, если булева функция принимает единичное значение на соответствующем наборе.

Количество клеток в карте Вейча соответствует количеству строк в таблице истинности логической функции.

На рисунке 1.15 представлен вид диаграммы Вейча для функций двух, трех и четырех переменных, а также соответствие клеток логическим значениям переменных для функции четырех переменных.

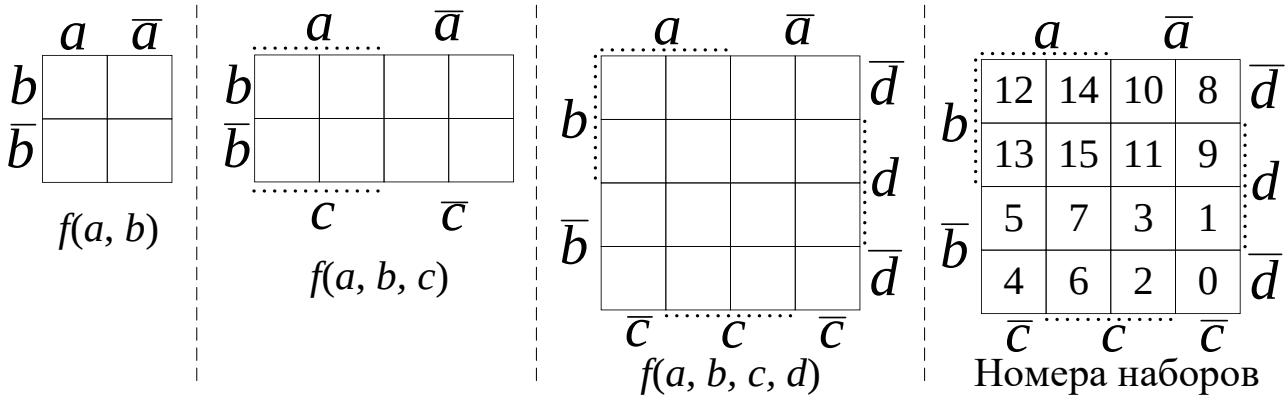


Рисунок 1.16 – Примеры диаграммы Вейча

Таким образом, при переходе в любую соседнюю клетку по вертикали и горизонтали (включая переходы из крайней клетки в клетку, лежащую на противоположном краю, т. е. при сворачивании в «трубочку») в составе переменных, определяющих расположение клеток, меняет свой знак только одна переменная. То есть одинаковые логические значения в соседних клетках позволяют проводить операции склеивания и поглощения.

Соответствие клеток логическим значениям переменных для функции четырех переменных показано на рисунке 1.17.

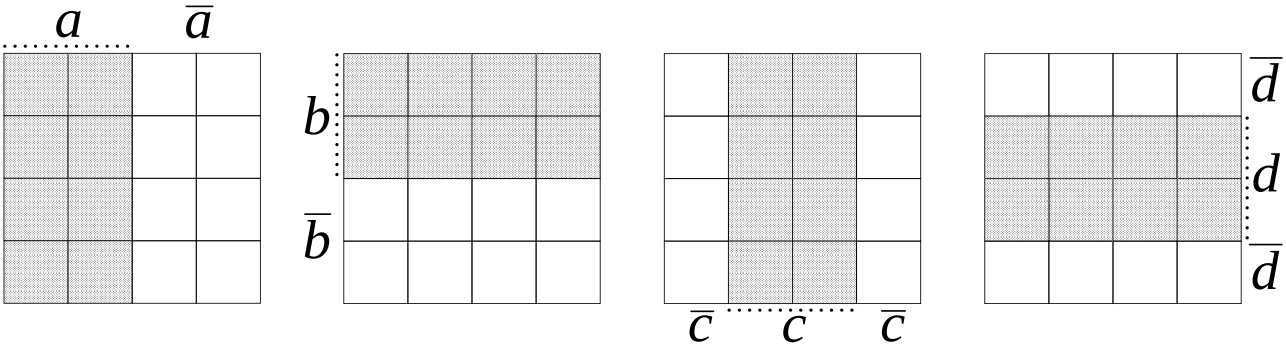


Рисунок 1.17 – Соответствие клеток логическим значениям переменных

1.10 Функционально полная система булевых функций. Классы функций алгебры логики. Базис

Любая булева функция может быть представлена аналитически одной из рассмотренных нормальных форм, которые используют ограниченное число элементарных булевых функций. Например, для СДНФ это функции конъюнкции, дизъюнкции, отрицания (И, ИЛИ, НЕ). Как следствие, существуют системы булевых функций, с помощью которых можно аналитически представить любую булеву функцию. Проблема функциональной полноты является основной проблемой функциональных построений в алгебре логики.

Функционально полной системой булевых функций (*ФПСБФ*) называется совокупность таких булевых функций $\{f_1, f_2, \dots, f_n\}$, что произвольная булева функция f может быть записана в виде формулы через функции этой совокупности.

Функционально полная система булевых функций (*базис*) – это такая совокупность логических функций, через которые можно выразить другие сколь угодно сложные логические функции.

Исходя из определений СДНФ, СКНФ к ФПСБФ относится система булевых функций И, ИЛИ, НЕ.

Булев базис – это функционально полная система логических функций, состоящая из двух функций двух переменных (И и ИЛИ) и одной функции одной переменной – функции НЕ.

Решение задачи определения свойств, которыми должны обладать функции, составляющие ФПСБФ, основано на понятии замкнутого относительно операции суперпозиции класса функций. Класс булевых функций, функционально замкнутый по операции суперпозиции, образует функциональную базу для булевых функций.

Функциональная база состоит из набора базовых булевых функций, с помощью которых можно выразить любую другую булеву функцию. При этом суперпозиция функций из одного класса также будет принадлежать этому классу.

Среди функционально замкнутых классов выделяют классы особого типа, называемые *предполными*, которые обладают определенным свойством. А именно предполный класс *S* не совпадает с множеством *P* всех возможных булевых функций, но если в него включить любую, не входящую в *S*, булеву функцию, то новый функционально замкнутый класс будет совпадать с множеством *P*.

Всего существует пять предполных классов, а для построения ФПСБФ необходимо и достаточно, чтобы ее функции не содержались полностью ни в одном из этих пяти предполных классов.

Предполные классы булевых функций:

1) булевы функции, сохраняющие константу нуль (K_0);

2) булевы функции, сохраняющие константу единица (K_1);

3) самодвойственные булевы функции ($K_С$);

4) линейные булевы функции ($K_Л$);

5) монотонные булевы функции ($K_М$).

Булевы функции, сохраняющие константу нуль, – это такие булевы функции $f\{x_1, x_2, \dots, x_n\}$, для которых справедливо соотношение $f\{0, \dots, 0\} = 0$. Например, к таким функциям относятся функции $f_0, f_1, \dots, f_7$ (таблица 1.29).

Булевы функции, сохраняющие константу единица, – это такие булевы функции $f\{x_1, x_2, \dots, x_n\}$, для которых справедливо соотношение $f\{1, \dots, 1\} = 1$. Например, к таким функциям относятся функции $f_1, f_3, f_6, f_7, f_9, f_{11}, f_{13}, f_{15}$ (таблица 1.29).

Двойственные друг другу булевы функции – это функции $f_1(x_1, \dots, x_n)$ и $f_2(x_1, \dots, x_n)$, для которых справедливо соотношение

$f_1(x_1, \dots, x_n) = \overline{f_2(\bar{x}_1, \dots, \bar{x}_n)}$.

К двойственным относятся функции f_0 и f_{15}, f_1 и f_7, f_2 и f_{11}(таблица 1.29).

Самодвойственная булева функция – это функция, которая на любых двух противоположных наборах принимает противоположные значения.

Например, к самодвойственным относятся функции f_3, f_5, f_{10}, f_{12} (таблица 1.29). Самодвойственная функция полностью определяется на первой половине строк таблицы истинности.

Линейные булевы функции – это такие функции, которые можно представить в следующем виде:

$$f_1(x_1, \dots, x_n) = c_0 \oplus c_1 x_1 \oplus \dots \oplus c_n x_n, \qquad (1.2)$$

где коэффициенты $c_0, \dots, c_n$представляют собой двоичные значения.

Линейными являются булевы функции $f_0, f_3, f_5, f_6, f_9, f_{10}, f_{12}, f_{15}$ (таблица 1.29), т. к. их можно представить в виде соотношения (1.2):

$f_0 = 0;$

$f_3 = x_1;$

$f_5 = x_2;$

$f_6 = x_1 \oplus x_2;$

$f_9 = 1 \oplus x_1 \oplus x_2;$

$f_{10} = \bar{x}_2 = 1 \oplus x_2;$

$f_{12} = \bar{x}_1 = 1 \oplus x_1;$

$f_{15} = 1.$

Линейная функция n переменных однозначно определяется заданием коэффициентов $c_0, \dots, c_n$. Следовательно, количество линейных булевых функций будет равно 2^{n+1}.

Двоичный набор $a = (a_1, a_2, \dots, a_n)$ не меньше двоичного набора $b = (b_1, b_2, \dots, b_n)$ или $a \geq b$, если для каждой пары a_i, b_i, справедливо соотношение $a_i \geq b_i$. Например, $(1,1,0,1) \geq (1,0,0,0)$; $(0,1,0,1) \geq (0,0,0,1)$. Для наборов $(1,0,0,1)$ и $(0,1,0,1)$ не применяется ни одно из соотношений $a \geq b$ и $a < b$, такие наборы являются несравнимыми.

Монотонная булева функция – это такая функция $f(x_1, \dots, x_n)$, при которой для любых двух наборов $a = (a_1, a_2, \dots, a_n)$ и $b = (b_1, b_2, \dots, b_n)$, где $a \geq b$, справедливо неравенство $f(a_1, a_2, \dots, a_n) \geq f(b_1, b_2, \dots, b_n)$.

Примерами монотонных функций являются функции $f_0, f_1, f_3, f_5, f_7, f_{15}$ (см. таблицу 1.28). При этом, несмотря на то что $(1,0) < (1,1)$, функция f_2 не является монотонной, т. к. $f_2(1,0) > f_2(1,1)$.

Для того чтобы система S булевых функций была функционально полной, необходимо и достаточно, чтобы эта система содержала:

– хотя бы одну булеву функцию, не сохраняющую константу 0;

– хотя бы одну булеву функцию, не сохраняющую константу 1;

– хотя бы одну несамодвойственную булеву функцию;

– хотя бы одну нелинейную булеву функцию;

– хотя бы одну немонотонную булеву функцию.

Представленные условия для ФПСБФ есть не что иное, как формулировка теоремы Поста – Яблонского. Простыми словами, система булевых функций является функционально полной тогда и только тогда, когда она целиком не содержится ни в одном из предполных классов.

Рассмотрим примеры ФПСБФ. Для наглядности сведем элементарные булевы функции двух переменных и некоторые функции одной переменной из таблицы 1.29 в таблицу 1.30, классифицируя каждую из них по признакам принадлежности к предполным классам в виде, соответствующем теореме Поста – Яблонского.

Таблица 1.29 – Классы функций

Функция	x_1x_2				**Классы**				
	00	**01**	**10**	**11**	K_0	K_1	$K_С$	$K_Л$	$K_М$
f_0	0	0	0	0	+	–	–	+	+
f_1	0	0	0	1	+	+	–	–	+
f_2	0	0	1	0	+	–	–	–	–
f_3	0	0	1	1	+	+	+	+	+
f_4	0	1	0	0	+	–	–	–	–
f_5	0	1	0	1	+	+	+	+	+
f_6	0	1	1	0	+	–	–	+	–
f_7	0	1	1	1	+	+	–	–	+
f_8	1	0	0	0	–	–	–	–	–
f_9	1	0	0	1	–	+	–	+	–
f_{10}	1	0	1	0	–	–	+	+	–
f_{11}	1	0	1	1	–	+	–	–	–
f_{12}	1	1	0	0	–	–	+	+	–
f_{13}	1	1	0	1	–	+	–	–	–
f_{14}	1	1	1	0	–	–	–	–	–
f_{15}	1	1	1	1	–	+	–	+	+

Таблица 1.30 – Виды функций

Функция	**Виды функций**				
	Не сохраняющие константу 0	**Не сохраняющие константу 1**	**Несамодвойственные**	**Нелинейные**	**Немонотонные**
$f_0 = 0$	–	+	+	–	–
$f_1 = x_1x_2$	–	–	+	+	–
$f_2 = x_1\overline{x_2}$	–	+	+	+	+
$f_3 = x_1$	–	–	–	–	–
$f_4 = \overline{x_1}x_2$	–	+	+	+	+
$f_5 = x_2$	–	–	–	–	–
$f_6 = x_1 \oplus x_2$	–	+	+	–	+
$f_7 = x_1 + x_2$	–	–	+	+	–
$f_8 = x_1 \downarrow x_2$	+	+	+	+	+
$f_9 = x_1 \equiv x_2$	+	–	+	–	+
$f_{10} = \overline{x_2}$	+	+	–	–	+
$f_{11} = x_2 \to x_1$	+	–	+	+	+
$f_{12} = \overline{x_1}$	+	+	–	–	+
$f_{13} = x_1 \to x_2$	+	–	+	+	+
$f_{14} = x_1/x_2$	+	+	+	+	+
$f_{15} = 1$	+	–	+	–	–

На основе таблицы 1.30 можно сделать вывод, что каждая из функций f_8 и f_{14} по отдельности представляют собой ФПСБФ. То есть любую булеву функцию можно записать, используя только функцию f_8 или f_{14}. Это следует из того, что каждая из этих функций соответствует всем условиям согласно теореме Поста – Яблонского о ФПСБФ.

ФПСБФ могут составлять и комбинации функций, например f_1 и $f_{10}(f_{12})$, которые представляют систему функций И, НЕ.

К самым распространенным ФПСБФ (базисам) относятся следующие системы:

а) ИЛИ-НЕ;

б) И-ИЛИ-НЕ;

в) И-$\oplus$-НЕ;

г) И-$\oplus$-1.

Если система булевых функций содержит хотя бы одну нелинейную и одну немонотонную функцию, то это означает, что система содержит функции, которые не могут быть выражены только с использованием линейных и монотонных функций.

Следовательно, система, содержащая хотя бы одну нелинейную и одну немонотонную функцию, не может быть выражена только с использованием констант и других линейных и монотонных функций. Это приводит к тому, что эта система не является функционально полной, т. к. не может выразить любую другую булеву функцию.

Таким образом, это утверждение действительно следует из теоремы о функциональной полноте.

Функция «константа 0» несамодвойственна и не сохраняет 1 (см. таблицу 1.30).

Функция «константа 1» несамодвойственна и не сохраняет 0 (см. таблицу 1.30). Также константы являются линейными и монотонными функциями.

На основании теоремы о функциональной полноте из этого вытекает следующее: система булевых функций является ослабленно функционально полной, если она содержит хотя бы одну нелинейную и одну немонотонную булевы функции.

Например:

а) И-$\oplus$;

б) $\rightarrow$;

в) ИЛИ-$\oplus$;

г) ИЛИ-$\equiv$;

д) И-$\equiv$.

Базисом называется функционально полная система ФАЛ, с помощью которой любая ФАЛ может быть представлена в виде комбинации элементарных булевых функций.

К базису относится система функций И, ИЛИ, НЕ, свойства которых были изучены Дж. Булем. Поэтому алгебру высказываний, использующих данный базис, называют булевой алгеброй.

Базисами являются также системы, содержащие функции И-НЕ, ИЛИ-НЕ, состоящие из функции Шеффера, функции Пирса (таблица 1.31).

Из данных перечислений видно, что одни базисы могут включать другие. Например, базис И-ИЛИ-НЕ и ИЛИ-НЕ.

Таблица 1.31 – Основные базисы

Название базиса	**Функции в базисе**
Базис 1	И-ИЛИ-НЕ
Базис 2	И-НЕ
Базис 3	ИЛИ-НЕ
Базис 4	Функция Шеффера
Базис 5	Функция Пирса

Исходя из этого выделяют такие понятия, как «избыточный базис» и «минимальный базис».

Минимальный базис – такой базис, удаление из которого хотя бы одной функции превращает систему булевых функций в неполную. Таким образом, *функционально полный базис* – это набор операций булевой алгебры (и соответствующих им элементов), позволяющих построить любую булеву функцию.

Избыточный базис – такой базис, удаление из которого некоторой функции не превращает систему булевых функций в неполную, а превращает этот базис в другой (возможно, минимальный).

Базис 1 избыточен, т. к. удаление из него функции И превращает его в минимальный базис 3, а удаление функции ИЛИ превращает его в минимальный базис 2.

Базис 2 является минимальным, а значит, с его помощью можно представить любую функцию.

Например, представим функцию ИЛИ, применив закон де Моргана:

$x_1 + x_2 = \overline{\overline{x_1 + x_2}} = \overline{\bar{x}_1 \bar{x}_2}$.

Аналогичным образом можно представить функцию И через базис 3 (ИЛИ-НЕ):

$x_1 x_2 = \overline{\overline{x_1 x_2}} = \overline{\bar{x}_1 + \bar{x}_2}$.

Следует обратить внимание, что многие функции могут быть записаны различными выражениями, в свою очередь в выражениях могут использоваться сочетания более простых операций. Среди всех возможных вариантов выражений, соответствующих определенной функции, есть наиболее простое, например, в котором используется наименьшее количество операндов или наименьшее количество более простых операций.

Поиск логической формулы, соответствующей данному условию, представляет особый интерес в задачах проектирования цифровых устройств.

1.11 Контрольные вопросы

1 Что такое логическое высказывание?

2 Чем отличается логическая константа от логической переменной?

3 Какие значения может принимать логическая функция?

4 Назовите и приведите примеры способов задания логических функций.

5 Какие формы задания логических функций существуют? Какие из них являются каноническими?

6 Поясните основные аксиомы для логических переменных.

7 Поясните основные правила (законы) для функций дизъюнкции и конъюнкции.

8 Поясните основные аксиомы для функций сложения по модулю 2.

9 Поясните основные аксиомы для функций импликации.

10 Поясните основные аксиомы для функций Шеффера.

11 Поясните основные аксиомы для функций Пирса (Вебба).

12 Какая существует взаимосвязь между формами записи логических функций? Как осуществляется переход от одной формы записи к другой?

13 Что представляет собой диаграмма Венна?

14 Что представляет собой карта Карно?

15 Что такое логические соседи? Какие ячейки являются логическими соседями при представлении карты Карно в трехмерном пространстве?

16 Что такое функционально полная система булевых функций?

17 Назовите основные классы предполных логических функций.

18 Что такое булев базис?

19 Чем отличаются минимальный и избыточный базисы?

2 МЕТОДЫ МИНИМИЗАЦИИ ЛОГИЧЕСКИХ ФУНКЦИЙ

При сравнении по минимальности различных форм представления ФАЛ становится ясно, что нормальные формы экономичнее совершенных нормальных форм. Но, с другой стороны, нормальные формы не дают однозначного представления.

Минимальная форма представления ФАЛ – это форма представления ФАЛ, которая содержит минимальное количество термов и переменных в термах (т. е. минимальная форма не допускает никаких упрощений).

Методы минимизации, позволяющие получить минимальную форму ФАЛ, широко применяются при проектировании цифровых автоматов. Общая задача минимизации заключается в поиске аналитического выражения заданной булевой функции в форме, содержащей минимально возможное число букв. Данная задача не имеет решения для общей формы представления, но значительно исследована для дизъюнктивно-конъюнктивных форм.

Минимальной дизъюнктивной нормальной формой (МДНФ) булевой функции называется ДНФ, содержащая наименьшее число букв по сравнению с другими ДНФ, представляющими заданную функцию.

Функция $g(x_1, x_2, \dots, x_n)$ называется импликантой функции $f(x_1, x_2, \dots, x_n)$, если для любого набора переменных, на котором $g = 1$, справедливо $f = 1$. Например, для функции, заданной таблицей истинности, все импликанты запишутся следующим образом (таблица 2.1).

Таблица 2.1 – Примеры импликант

x_1	x_2	x_3	f	g_1	g_2	g_3	g_4	g_5	g_6	g_7
0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
1	1	0	1	0	1	1	0	0	1	1
1	1	1	1	1	0	1	0	1	0	1

Тогда:

$g_1 = x_1 x_2 x_3;$

$g_2 = x_1 x_2 \bar{x}_3;$

$g_3 = x_1x_2x_3 + x_1x_2\bar{x}_3 = x_1x_2(x_3 + \bar{x}_3) = x_1x_2;$

$g_4 = \bar{x}_1x_2x_3;$

$g_5 = \bar{x}_1x_2x_3 + x_1x_2x_3 = (\bar{x}_1 + x_1)x_2x_3 = x_2x_3;$

$g_6 = \bar{x}_1x_2x_3 + x_1x_2\bar{x}_3;$

$f = g_7 = \bar{x}_1x_2x_3 + x_1x_2\bar{x}_3 + x_1x_2x_3.$

Простая импликанта – это импликанта g булевой функции f, являющаяся элементарной конъюнкцией, при этом никакая часть импликанты g не является импликантой функции f.

В рассмотренном примере простыми импликантами являются импликанты g_3 и g_5, т. к. никакая часть этих импликант не является импликантой f. Импликанты g_1, g_2, g_4, g_6 простыми не являются, т. к. их части являются импликантами функции f. Например, импликанта g_1 является частью импликанты g_6.

При получении минимальной ДНФ будут справедливы следующие утверждения:

1) дизъюнкция любого числа импликант булевой функции f также является импликантой этой функции;

2) любая булева функция f эквивалентна дизъюнкции всех своих простых импликант. Такая форма представления булевой функции называется сокращенной ДНФ.

Для рассмотренного примера сокращенная ДНФ функции f запишется следующим образом:

$f = g_3 + g_5 = x_1x_2 + x_2x_3.$

Простые импликанты функции f представлены в таблице 2.2.

Таблица 2.2 – Простые импликанты функции f

g_3	g_5	$g_3 + g_5$	f
0	0	0	0
0	0	0	0
0	0	0	0
0	1	1	1
0	0	0	0
0	0	0	0
1	0	1	1
1	1	1	1

Простые импликанты функции f покрывают своими единицами все единицы функции f. Поиск минимальной ДНФ является первым этапом поиска минимальных форм булевых функций. В некоторых случаях для получения исходной функции можно использовать только часть простых импликант.

Таким образом, импликанты, которые можно исключить, являются *несущественными*, или лишними, а импликанты, исключение которых приводит к нарушению эквивалентности исходной функции, являются *существенными*.

Процесс исключения несущественных импликант представляет собой второй этап минимизации функции. Сокращенная ДНФ булевой функции, в которой отсутствуют лишние простые импликанты, называется *тупиковой*. Булева функция может иметь несколько тупиковых ДНФ, это связано с процессом неоднозначности устранения лишних простых импликант.

Например, для функции f простые импликанты g_1, g_2 являются существенными, а для сохранения эквивалентности значения исходной функции достаточно одной из простых импликант g_3 или g_4 (таблица 2.3).

Таблица 2.3 – Пример нескольких вариантов тупиковых форм для функции f

x_1	x_2	x_3	f	g_1	g_2	g_3	g_4
0	0	0	0	0	0	0	0
0	0	1	1	1	0	1	0
0	1	0	1	0	1	0	1
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0
1	1	0	1	0	1	0	0
1	1	1	0	0	0	0	0

Таким образом, для данного примера существует две тупиковые ДНФ:

$f = g_1 + g_2 + g_3$;

$f = g_1 + g_2 + g_4$.

Тупиковые ДНФ булевой функции f, содержащие минимальное число букв, являются *минимальными*. Минимальных ДНФ также может быть несколько.

Минимизация функции – это процесс нахождения наиболее простого выражения для исходной функции.

Выделяют несколько способов минимизации функций. Все они различаются практически только на этапе получения сокращенной ДНФ. Стоит помнить, что поиск МДНФ всегда связан с некоторым перебором решений.

2.1 Метод Квайна

В методе Квайна простые импликанты находятся по СДНФ булевой функции в результате последовательного применения к ней операции неполного склеивания и операции поглощения.

Для понимания данных операций следует рассмотреть процесс развертывания сокращенной ДНФ в СДНФ. Например:

$f_{\text{сокр}} = x_1x_2 + x_2x_3 + x_3x_4.$

Для преобразования данной функции в СДНФ необходимо к каждому элементарному произведению добавить множители, представляющие собой логическую сумму переменной, отсутствующей в произведении, и отрицания этой переменной:

$$f_{\text{СДНФ}} = x_1x_2(x_3 + \bar{x}_3) + (x_1 + \bar{x}_1)x_2x_3 + (x_1 + \bar{x}_1)x_3x_4 =$$
$$= x_1x_2x_3 + x_1x_2\bar{x}_3 + x_1x_2x_3 + \bar{x}_1x_2x_3 + x_1x_3x_4 + \bar{x}_1x_3x_4 =$$
$$= x_1x_2x_3(x_4 + \bar{x}_4) + x_1x_2\bar{x}_3(x_4 + \bar{x}_4) + x_1x_2x_3(x_4 + \bar{x}_4) +$$
$$+ \bar{x}_1x_2x_3(x_4 + \bar{x}_4) + x_1x_3x_4(x_2 + \bar{x}_2) + \bar{x}_1x_3x_4(x_2 + \bar{x}_2) =$$
$$= x_1x_2x_3x_4 + x_1x_2x_3\bar{x}_4 + x_1x_2\bar{x}_3x_4 + x_1x_2\bar{x}_3\bar{x}_4 + x_1x_2x_3x_4 + x_1x_2x_3\bar{x}_4 +$$
$$+ \bar{x}_1x_2x_3x_4 + \bar{x}_1x_2x_3\bar{x}_4 + x_1x_2x_3x_4 + x_1\bar{x}_2x_3x_4 + \bar{x}_1x_2x_3x_4 + \bar{x}_1\bar{x}_2x_3x_4.$$

Функция $f_{\text{СДНФ}}$ представляет собой искомую СДНФ, в которой некоторые конституенты повторяются.

Процесс перехода от сокращенной ДНФ к СДНФ обратим. Точно таким же образом можно поэтапно перейти от СДНФ к сокращенной ДНФ, используя операции склеивания. Применяя операцию склеивания многократно для каждого промежуточного результата, можно получить сокращенную ДНФ.

При этом следует учесть, что после выполнения операции склеивания участвующие конституенты (склеиваемые члены) не следует сразу же исключать. Поэтому привычная формула операции склеивания

$Ax + A\bar{x} = A$

примет вид

$Ax + A\bar{x} = Ax + A\bar{x} + A.$

Полученное преобразование получило название *операции неполного склеивания*.

Для того чтобы процесс с использованием операции неполного склеивания дал нужный результат, следует исключить члены, участвующие в склеивании, после выполнения с ними всех возможных операций склеивания. Такая операция называется операцией *поглощения* и сводится к последовательному применению формулы элементарного поглощения к членам промежуточной функции.

Следует отметить, что исключить с помощью операции поглощения можно лишь те члены, к которым были применены все возможные операции склеивания.

Таким образом, метод Квайна основывается на применении двух основных соотношений:

1) соотношение склеивания и неполного склеивания:

$Ax + A\bar{x} = Ax + A\bar{x} + A = A,$

где A – любое элементарное произведение;

2) соотношение поглощения:

$A\bar{x} + A = A.$

Полученное логическое выражение не обязательно будет соответствовать минимальной форме. Поэтому следует приступить ко второму этапу минимизации функций – переходу от сокращенной к тупиковой ДНФ. Для этого строится импликантная матрица, строки которой соответствуют простым импликантам, полученным в результате выполнения первого этапа, а столбцы – конституентам единицы исходной СДНФ.

Если конституенты единицы покрываются соответствующей простой импликантой, то в ячейке на пересечении соответствующего столбца и строки проставляется символ астриска (*), т. е. отмечаются те конституенты единицы, которые включают простую импликанту как свою составную часть. После того как импликантная матрица будет построена, можно перейти к непосредственному поиску тупиковых форм.

Тупиковая ДНФ строится следующим образом:

1 В импликантной матрице находятся столбцы, имеющие только один астериск. Соответствующие этим астерискам импликанты должны быть обязательно представлены в тупиковой ДНФ. Иногда данные импликанты называются базовыми и представляют собой ядро булевой функции. Найденный столбец исключается из рассмотрения точно так же, как и столбцы, покрываемые найденной импликантой.

2 Находятся импликанты, покрывающие наибольшее количество еще не покрытых столбцов.

В результате может быть найдено несколько тупиковых ДНФ. Минимальной ДНФ в таком случае будет та тупиковая ДНФ, в которой представлено наименьшее суммарное число букв.

В качестве примера рассмотрим задачу минимизации булевой функции, заданной таблицей 2.4.

Таблица 2.4 – Таблица истинности логической функции четырех переменных для минимизации методом Квайна

Номер набора	**Значения аргументов**				**Значение функции** $f(x_1, x_2, x_3, x_4)$
	x_1	x_2	x_3	x_4	
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	1

Запишем СКНФ заданной функции:

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1\bar{x}_2\bar{x}_3x_4 + \bar{x}_1\bar{x}_2x_3x_4 + \bar{x}_1x_2\bar{x}_3x_4 +$$
$$+ \bar{x}_1x_2x_3x_4 + x_1x_2x_3\bar{x}_4 + x_1x_2x_3x_4.$$

Для удобства записи дальнейших действий пронумеруем все конституенты единицы в полученном выражении:

$$f(x_1, x_2, x_3, x_4) = \overset{1}{\bar{x}_1\bar{x}_2\bar{x}_3x_4} + \overset{2}{\bar{x}_1\bar{x}_2x_3x_4} + \overset{3}{\bar{x}_1x_2\bar{x}_3x_4} +$$
$$+ \overset{4}{\bar{x}_1x_2x_3x_4} + \overset{5}{x_1x_2x_3\bar{x}_4} + \overset{6}{x_1x_2x_3x_4}.$$

Выполним операции склеивания, результаты сведем в таблицу 2.5.

Таблица 2.5 – Результаты склеиваний, выполненных на первом шаге минимизации методом Квайна

Склеиваемая пара	**Результат**	**Переменная, по которой осуществляется склеивание**	**Нумерация полученных конъюнкций**
1, 2	$\bar{x}_1\bar{x}_2x_4$	x_3	7
1, 3	$\bar{x}_1\bar{x}_3x_4$	x_2	8
2, 4	$\bar{x}_1x_3x_4$	x_2	9
3, 4	$\bar{x}_1x_2x_4$	x_3	10
4, 6	$x_2x_3x_4$	x_1	11
5, 6	$x_1x_2x_3$	x_4	12

Продолжим выполнять склеивания (таблица 2.6).

Таблица 2.6 – Результаты последующих склеиваний, выполненных на первом шаге минимизации методом Квайна

Склеиваемая пара	**Результат**	**Переменная, по которой осуществляется склеивание**	**Нумерация полученных конъюнкций**
7, 10	$\bar{x}_1x_4$	x_2	13
8, 9	$\bar{x}_1x_4$	x_3	14

Полученные конъюнкции больше склеить нельзя. В результате после выполнения первого шага минимизации методом Квайна получаем функцию

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1\bar{x}_2\bar{x}_3x_4 + \bar{x}_1\bar{x}_2x_3x_4 + \bar{x}_1x_2\bar{x}_3x_4 +$$
$$+ \bar{x}_1x_2x_3x_4 + x_1x_2x_3\bar{x}_4 + x_1x_2x_3x_4 + \bar{x}_1\bar{x}_2x_4 + \bar{x}_1\bar{x}_3x_4 +$$
$$+ \bar{x}_1x_3x_4 + \bar{x}_1x_2x_4 + x_2x_3x_4 + x_1x_2x_3 + \bar{x}_1x_4 + \bar{x}_1x_4.$$

На следующем шаге выполним операции всевозможного поглощения, в результате чего выражение примет вид

$$f(x_1, x_2, x_3, x_4) = x_2x_3x_4 + x_1x_2x_3 + \bar{x}_1x_4.$$

Полученная функция представляет собой сокращенную ДНФ.

Для того чтобы от сокращенной ДНФ перейти к минимальной, необходимо из сокращенной ДНФ убрать лишние импликанты. Для этого переходим ко второму этапу минимизации булевой функции методом Квайна, а именно составлению импликантной матрицы (рисунок 2.1).

Импликанты	Конституенты единицы					
	1	**2**	**3**	**4**	**5**	**6**
	$\bar{x}_1\bar{x}_2\bar{x}_3x_4$	$\bar{x}_1\bar{x}_2x_3x_4$	$\bar{x}_1x_2\bar{x}_3x_4$	$\bar{x}_1x_2x_3x_4$	$x_1x_2x_3\bar{x}_4$	$x_1x_2x_3x_4$
$x_2x_3x_4$				*		*
$x_1x_2x_3$					*	*
$\bar{x}_1x_4$	*	*	*	*		

Рисунок 2.1 – Импликантная матрица минимизации функции методом Квайна

В строках матрицы необходимо записать найденные на первом этапе простые импликанты, в столбцах – конституенты единицы исходной функции.

Далее отмечаются те конституенты единицы, которые включают простую импликанту как свою составную часть.

Согласно алгоритму необходимо найти столбцы, имеющие только один астериск (*). Таким является столбец 1 с конституентой $\bar{x}_1\bar{x}_2\bar{x}_3x_4$, соответствующая ему импликанта – $\bar{x}_1x_4$. Данная импликанта является обязательной в тупиковой ДНФ. Далее следует исключить из рассмотрения конституенты единицы, которые покрывает базовая импликанта (столбцы 1–4). Таким образом, для рассмотрения остаются только конституенты $x_1x_2x_3\bar{x}_4$ и $x_1x_2x_3x_4$ (столбцы 5, 6).

Аналогичным образом в столбце 5 конституенты единицы $x_1x_2x_3\bar{x}_4$ только один астериск (*), поэтому импликанта $x_1x_2x_3$ является обязательной (базовой) в записи тупиковой ДНФ. Найденная импликанта также покрывает и конституенту единицы $x_1x_2x_3x_4$ (столбец 6). Таким образом, найденные базовые импликанты $\bar{x}_1x_4$ и $x_1x_2x_3$ покрывают все конституенты единицы, следовательно, импликанта $x_2x_3x_4$ является лишней.

В результате найдена одна тупиковая ДНФ, которая соответствует минимальной ДНФ и принимает вид

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1x_4 + x_1x_2x_3.$$

2.2 Метод карт Карно

Одним из наиболее часто используемых методов является минимизация с использованием карт Карно (или диаграмм Вейча – Карно).

Как было рассмотрено ранее (см. подраздел 1.8), карта Карно представляет собой матрицу прямоугольной формы, где каждая ячейка является логическим соседом.

Логическая функция, заданная картой Карно, должна быть представлена в одной из канонических форм (СДНФ или СКНФ).

В процессе минимизации может возникнуть задача перехода между основными каноническими формами записи функции. Например, если функция задана в СДНФ, требуется найти ее СКНФ. Такой переход можно выполнить, составив по заданной СДНФ таблицу истинности для необходимой функции, а на основе полученной таблицы составить СКНФ. Помимо такого способа, данную задачу можно решить, используя правило де Моргана.

Для минимизации функции, заданной картой Карно, необходимо охватить контурами множество ячеек карты Карно, а затем записать минимальное выражение для каждого контура.

Охват ячеек карты Карно контурами выполняется с соблюдением следующих правил:

– в контур заключаются заполненные (единицами) ячейки, являющиеся логическими соседями (в том числе в трехмерном пространстве, т. е. при сворачивании в «трубочку» или «конверт»);

– контур должен иметь прямоугольную форму (квадрат – частный случай прямоугольника);

– в контур может входить только такое количество ячеек, которое равно двойке, возведенной в целую степень (например, $2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, $2^4 = 16$ и т. д.);

– в контур необходимо включить максимальное количество ячеек с учетом вышеприведенных правил;

– контуров должно быть минимальное количество (следует из предыдущего правила);

– контуры могут пересекаться.

Запись минимального выражения для заданной функции формируется таким образом, чтобы конъюнкция или дизъюнкция, соответствующая контуру, включала только те переменные, которые имеют постоянное значение во всех ячейках, охваченных рассматриваемым контуром, т. е. записываются только переменные, которые находятся в одинаковой форме (прямой или инверсной) для всех ячеек контура.

Пример минимизации методом карт Карно логической функции в СДНФ

Функция трех переменных $f_1(x_1, x_2, x_3)$ в СДНФ имеет вид

$$f_1 = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2x_3 + x_1\bar{x}_2\bar{x}_3. \quad (2.1)$$

Карта Карно функции $f_1(x_1, x_2, x_3)$ представлена на рисунке 2.2.

Необходимо выполнить минимизацию функции $f_1(x_1, x_2, x_3)$ методом карты Карно.

	$\bar{x}_1\bar{x}_2$	$\bar{x}_1 x_2$	$x_1 x_2$	$x_1\bar{x}_2$
$\bar{x}_3$	1			1
x_3	1	1		

Рисунок 2.2 – Карта Карно функции f_1

Необходимо охватить контурами все заполненные единицами ячейки по вышеописанным правилам. Таким образом получается три контура (I, II и III на рисунке 2.3).

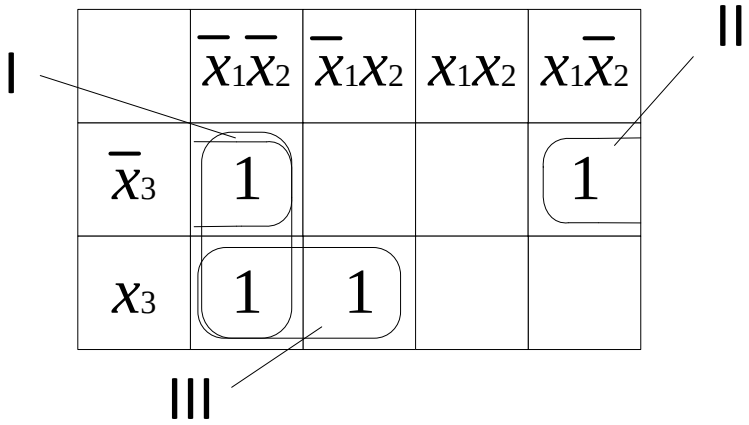


Рисунок 2.3 – Контуры функции f_1

Контур I представляет собой две ячейки, соответствующие набору $\bar{x}_1\bar{x}_2\bar{x}_3$ и $\bar{x}_1\bar{x}_2 x_3$. Постоянными переменными для ячеек в данном контуре являются $\bar{x}_1\bar{x}_2$. Тогда минимальное логическое выражение для контура I примет вид

$$\bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2 x_3 = \bar{x}_1\bar{x}_2.$$

Контур II образуется при сворачивании карты Карно в «трубочку» по вертикальной линии изгиба и представляет собой две ячейки, соответствующие набору $\bar{x}_1\bar{x}_2\bar{x}_3$ и $x_1\bar{x}_2\bar{x}_3$. Постоянными переменными для ячеек в данном контуре являются $\bar{x}_2\bar{x}_3$. Тогда минимальное логическое выражение для контура II примет вид

$$\bar{x}_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2\bar{x}_3 = \bar{x}_2\bar{x}_3.$$

Контур III представляет собой две ячейки, соответствующие набору $\bar{x}_1\bar{x}_2 x_3$ и $\bar{x}_1 x_2 x_3$. Постоянными переменными для ячеек в данном контуре являются $\bar{x}_1 x_3$. Тогда минимальное логическое выражение для контура III примет вид

$$\bar{x}_1\bar{x}_2 x_3 + \bar{x}_1 x_2 x_3 = \bar{x}_1 x_3.$$

Минимальное логическое выражение для функции $f_1(x_1, x_2, x_3)$ примет вид

$$f_1(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2 + \bar{x}_1 x_3 + \bar{x}_2\bar{x}_3, \quad (2.2)$$

где $\bar{x}_1\bar{x}_2$ – минимальное логическое выражение для I, $\bar{x}_1 x_3$ – для II и $\bar{x}_2\bar{x}_3$ – для III контуров.

Полученное выражение (2.2) является результатом минимизации функции $f_1(x_1, x_2, x_3)$ методом карт Карно и, как можно заметить, оно значительно компактнее выражения (2.1).

Пример минимизации методом карт Карно логической функции в СКНФ

Функция трех переменных $f_2(x_1, x_2, x_3)$ в СКНФ имеет вид

$$f_2(x_1, x_2, x_3) = (x_1 + x_2 + \bar{x}_3) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_3) \cdot (\bar{x}_1 + x_2 + \bar{x}_3). \quad (2.3)$$

Необходимо выполнить минимизацию функции $f_2(x_1, x_2, x_3)$ методом карты Карно.

В качестве наборов, соответствующих ячейкам карт Карно, используются простые дизъюнкции, а в соответствующие ячейки устанавливаются значения «нуль» (рисунок 2.4).

	$\overline{X}_1\overline{X}_2$	$\overline{X}_1 X_2$	$X_1 X_2$	$X_1\overline{X}_2$
$\overline{X}_3$	0	0	0	
X_3				

Рисунок 2.4 – Карта Карно функции f_2

Необходимо охватить контурами все заполненные нулями ячейки по вышеописанным правилам. Таким образом, получается два контура (I и II на рисунке 2.5).

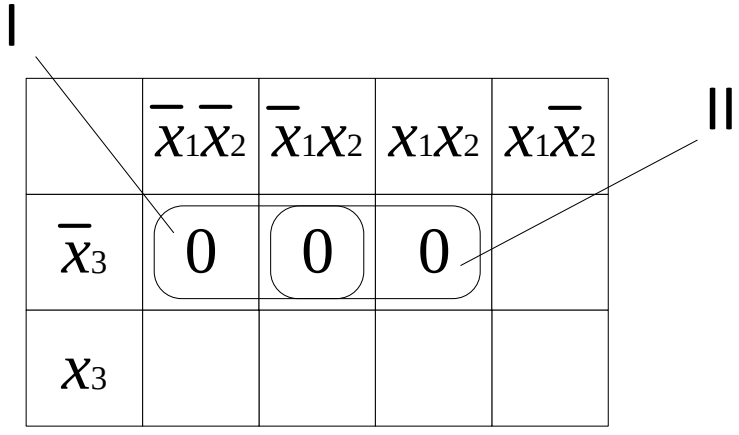


Рисунок 2.5 – Контуры функции f_2

Контур I представляет собой две ячейки, соответствующие набору $\bar{x}_1 + \bar{x}_2 + \bar{x}_3$ и $\bar{x}_1 + x_2 + \bar{x}_3$. Постоянными переменными для ячеек в данном контуре являются $\bar{x}_1 + \bar{x}_3$. Тогда минимальное логическое выражение для контура I примет вид

$$(\bar{x}_1 + \bar{x}_2 + \bar{x}_3) \cdot (\bar{x}_1 + x_2 + \bar{x}_3) = \bar{x}_1 + \bar{x}_3.$$

Контур II представляет собой две ячейки, соответствующие набору $\bar{x}_1 + x_2 + \bar{x}_3$ и $x_1 + x_2 + \bar{x}_3$. Постоянными переменными для ячеек в данном контуре являются $x_2 + \bar{x}_3$. Тогда минимальное логическое выражение для контура II примет вид

$$(\bar{x}_1 + x_2 + \bar{x}_3) \cdot (x_1 + x_2 + \bar{x}_3) = x_2 + \bar{x}_3.$$

Минимальное логическое выражение для функции $f_2(x_1, x_2, x_3)$ примет вид

$$f_1(x_1, x_2, x_3) = (\bar{x}_1 + \bar{x}_3) \cdot (x_2 + \bar{x}_3), \qquad (2.4)$$

где $\bar{x}_1 + \bar{x}_3$ – минимальное логическое выражение для I, а $x_2 + \bar{x}_3$ – для II контуров.

Полученное выражение (2.4) является результатом минимизации функции $f_2(x_1, x_2, x_3)$ методом карт Карно и, как можно заметить, оно значительно компактнее выражения (2.3).

Пример минимизации методом карт Карно логической функции четырех переменных в СДНФ

Выполнить минимизацию функции $f_3(x_1, x_2, x_3, x_4)$, заданной на единичных наборах (т. е. в СДНФ) в числовой форме $f_3(x_1, x_2, x_3, x_4) = \{0, 2, 5, 7, 8, 10, 13, 15\}$. Записать минимальную функцию в форме ДНФ.

Для наглядности запишем исходную функцию $f_3(x_1, x_2, x_3, x_4)$ в форме СДНФ:

$$f(x_1x_2x_3x_4) = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 + \bar{x}_1\bar{x}_2x_3\bar{x}_4 + \bar{x}_1x_2\bar{x}_3x_4 + \\ + \bar{x}_1x_2x_3x_4 + x_1\bar{x}_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2x_3\bar{x}_4 + x_1x_2\bar{x}_3x_4 + x_1x_2x_3x_4. \qquad (2.5)$$

Приведем словесное описание исходной функции $f_3(x_1, x_2, x_3, x_4)$:

«Функция $f_3(x_1x_2x_3x_4)$ будет иметь значение «истина», если одновременно аргументы x_1, x_2, x_3 и x_4 будут иметь значение «ложь» (0-й набор), или если одновременно аргументы x_1, x_2 и x_4 будут иметь значение «ложь», а x_3 – «истина» (2-й набор), или если одновременно аргументы x_1 и x_3 будут иметь значение «ложь», а x_2 и x_4 одновременно будут иметь значение «истина» (5-й набор), или если одновременно аргументы x_2, x_3 и x_4 будут иметь значение «истина», а x_1 – «ложь» (7-й набор), или если одновременно аргументы x_2, x_3 и x_4 будут иметь значение «ложь», а x_1 – «истина» (8-й набор), или если одновременно аргументы x_1 и x_3 будут иметь значение «истина», а x_2 и x_4 одновременно будут иметь значение «ложь» (10-й набор), или если одновременно аргументы x_1, x_2 и x_4 будут иметь значение «истина», а x_3 – «ложь» (13-й набор),

или если одновременно аргументы x_1, x_2, x_3 и x_4 будут иметь значение «истина» (15-й набор)».

Также для наглядности представим исходную функцию $f_3(x_1, x_2, x_3, x_4)$ в виде таблицы истинности (таблица 2.7).

Таблица 2.7 – Функция $f_3(x_1, x_2, x_3, x_4)$, заданная в виде таблицы истинности

Номер набора	x_1	x_2	x_3	x_4	$f(x_1x_2x_3x_4)$
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

Разместим простые конъюнкции на карте Карно и объединим все ячейки с единичными значениями в контуры по вышеописанным правилам. Таким образом получим два контура (рисунок 2.6).

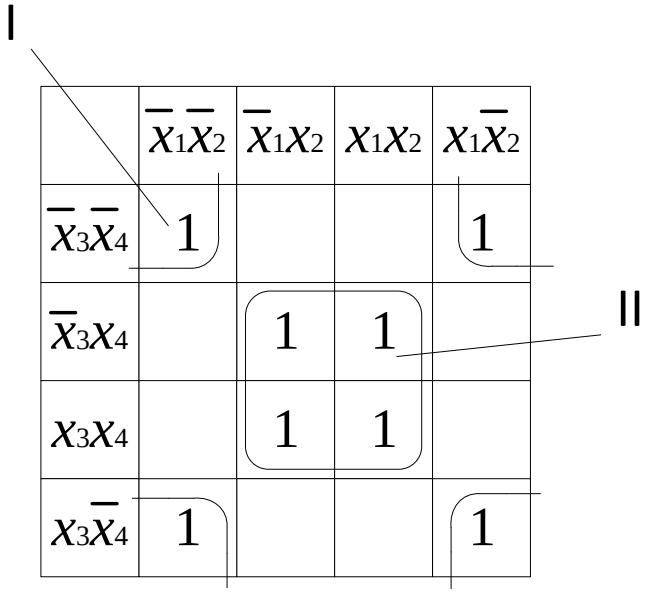


Рисунок 2.6 – Карта Карно и контуры для функции $f_3(x_1, x_2, x_3, x_4)$

В результате минимальное выражение функции $f_3(x_1, x_2, x_3, x_4)$ примет вид ДНФ:

$$f_3(x_1, x_2, x_3, x_4) = \bar{x}_2\bar{x}_4 + x_2x_4. \quad (2.6)$$

Приведем словесное описание минимальной функции $f_3(x_1, x_2, x_3, x_4)$:

«Функция $f_3(x_1, x_2, x_3, x_4)$ всегда будет иметь значение «истина», если одновременно аргументы x_2 и x_4 будут иметь значение «ложь» или одновременно значение «истина».

Сравнивая словесное описание исходной и минимальной функции $f_3(x_1, x_2, x_3, x_4)$ или выражения (2.5) и (2.6), можно понять, насколько проще стала реализация функции $f_3(x_1, x_2, x_3, x_4)$ после минимизации при полном сохранении логики ее выполнения.

В случае когда результат минимизации необходимо записать в КНФ, необходимо либо изначально по таблице истинности записать СКНФ исходной функции, либо прибегнуть к решению задачи перехода из одной канонической формы к другой.

Переход от формы записи выражения в СДНФ к форме записи в СКНФ, помимо способа с использованием таблицы истинности, может быть реализован при помощи правила де Моргана.

Пример перехода от одной формы записи логической функции к другой

По заданной СДНФ функции f_4 найти запись этой функции в СКНФ.

$f_4 = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 + \bar{x}_1x_2x_3 + x_1x_2\bar{x}_3 + x_1\bar{x}_2\bar{x}_3.$

Для начала запишем отрицание данной функции, т. е. запишем дизъюнкцию простых конъюнкций, где простые конъюнкции представляют собой конституенты нуля. В свою очередь, конституенты нуля это те наборы, которые не являются наборами конституент единиц:

$$\bar{f}_4 = \bar{x}_1\bar{x}_2x_3 + x_1x_2x_3 + x_1\bar{x}_2x_3. \quad (2.7)$$

Запишем отрицание правой и левой частей выражения (2.7):

$$\bar{\bar{f}}_4 = \overline{\bar{x}_1\bar{x}_2x_3 + x_1x_2x_3 + x_1\bar{x}_2x_3}.$$

Для левой части применим правило двойного отрицания, а для правой – правило де Моргана:

$$\bar{\bar{f}}_4 = f_4.$$

$$f_4 = \overline{\bar{x}_1\bar{x}_2x_3} \cdot \overline{x_1x_2x_3} \cdot \overline{x_1\bar{x}_2x_3}.$$

Еще раз применим правило де Моргана к конъюнкциям в правой части:

$$f_4 = (\bar{\bar{x}}_1 + \bar{\bar{x}}_2 + \bar{x}_3) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_3) \cdot (\bar{x}_1 + \bar{\bar{x}}_2 + \bar{x}_3).$$

Применив правило двойного отрицания, получаем следующее выражение:

$$f_4 = (x_1 + x_2 + \bar{x}_3) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_3) \cdot (\bar{x}_1 + x_2 + \bar{x}_3). \quad (2.8)$$

Приведение формы записи функции, представленной в СКНФ, к форме СДНФ производится аналогичным образом.

2.3 Контрольные вопросы

1 Что такое минимальная форма представления логической функции?

2 Что такое простая импликанта?

3 Чем отличаются несущественные и существенные импликанты?

4 Что такое тупиковая простая импликанта?

5 Какие операции и в какой последовательности применяются при минимизации логической функции методом Квайна?

6 Как осуществляется развертывание сокращенной ДНФ в СДНФ?

7 Что такое операция неполного склеивания?

8 Опишите алгоритм построения тупиковой ДНФ.

9 Как осуществляется переход от сокращенной ДНФ к минимальной?

10 Каким образом строится импликантная матрица?

11 Назовите правила охвата ячеек карты Карно контурами.

12 Как описывается каждый контур карты Карно при минимизации?

13 Как можно осуществить переход от одной нормальной формы к другой?

3 СТАНДАРТНЫЕ ФУНКЦИОНАЛЬНЫЕ УЗЛЫ ЦИФРОВЫХ УСТРОЙСТВ

3.1 Элементы электронных вычислительных машин

Элементы ЭВМ представляют собой простейшие радиотехнические детали, на основе которых строятся более крупные части ЭВМ – узлы. Общее условное графическое обозначение абстрактного элемента представлено на рисунке 3.1 и предполагает обозначение трех полей:

а) поля входов (слева). Количество входов n, где $n \geq 1$;

б) поля выходов (справа). Количество выходов m, где $m \geq 1$;

в) поле обозначения элемента (по центру), несущее информацию о типе элемента и, возможно, об исполняемых им функциях.

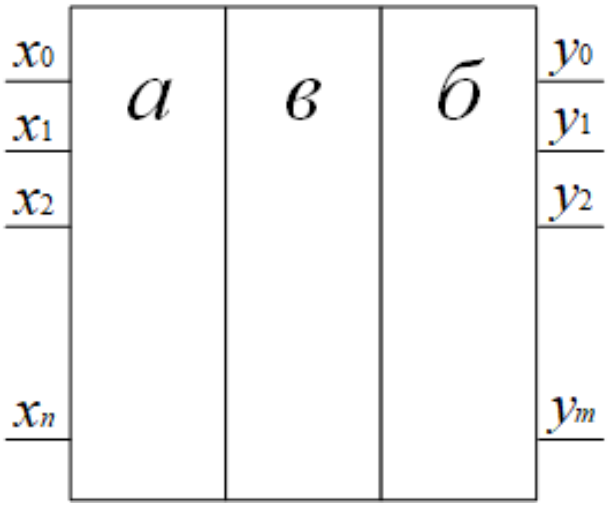


Рисунок 3.1 – Общее обозначение элемента ЭВМ

Элементы ЭВМ бывают трех основных разновидностей:

1 *Логические* – реализующие какие-либо логические функции алгебры логики.

2 *Запоминающие* – триггеры (автомат Мура), реализующие хранение одного из двух устойчивых состояний.

3 *Специальные (вспомогательные)* – усилители, формирователи и генераторы сигналов, преобразователи логических уровней, индикаторы и др.

3.2 Основные узлы электронных вычислительных машин

Все основные узлы ЭВМ строятся на базе элементов и, как правило, представляют собой совокупность нескольких логических элементов или элементов памяти.

Выделяются следующие основные узлы ЭВМ:

1 *Комбинационные* – выходные сигналы зависят только от действующих в настоящее время входных сигналов.

2 *Накапливающие* – выходные сигналы зависят не только от действующих в настоящее время входных сигналов, но и от поступавших ранее.

К **комбинационным узлам** относятся:

1 *Полусумматор* – узел, выполняющий операцию поразрядного сложения над одним двоичным разрядом. Входами такого сумматора являются разряд первого слагаемого *a*, разряд второго слагаемого *b*, а выходами являются сумма в текущем разряде *S* и признак переноса в старший разряд *P*.

Обозначение полусумматора на схемах и его реализация на логических элементах И и исключающее ИЛИ представлены на рисунке 3.2, *а* и *б* соответственно.

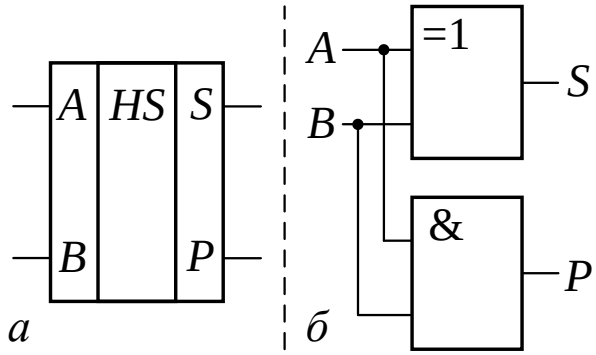


Рисунок 3.2 – Полусумматор

2 *Одноразрядный сумматор* – узел, выполняющий операцию поразрядного сложения над одним двоичным разрядом. Входами такого сумматора являются разряд первого слагаемого *a*, разряд второго слагаемого *b*, признак переноса из младшего разряда *p*, а выходами являются сумма в текущем разряде *S* и признак переноса в старший разряд *P*.

Обозначение одноразрядного сумматора на схемах и его реализация на логических элементах И, ИЛИ и исключающее ИЛИ представлены на рисунке 3.3, *а* и *б* соответственно.

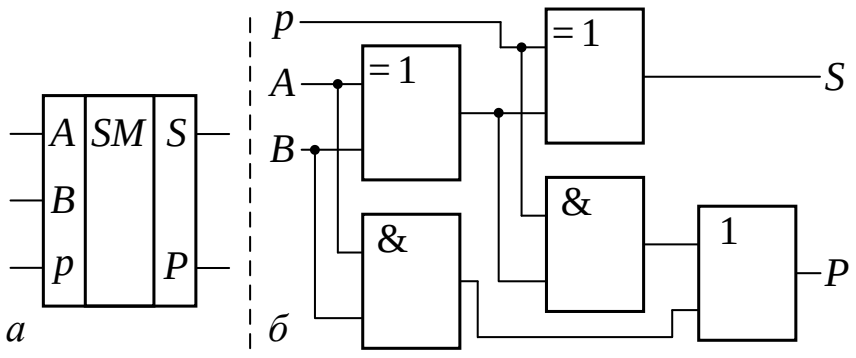
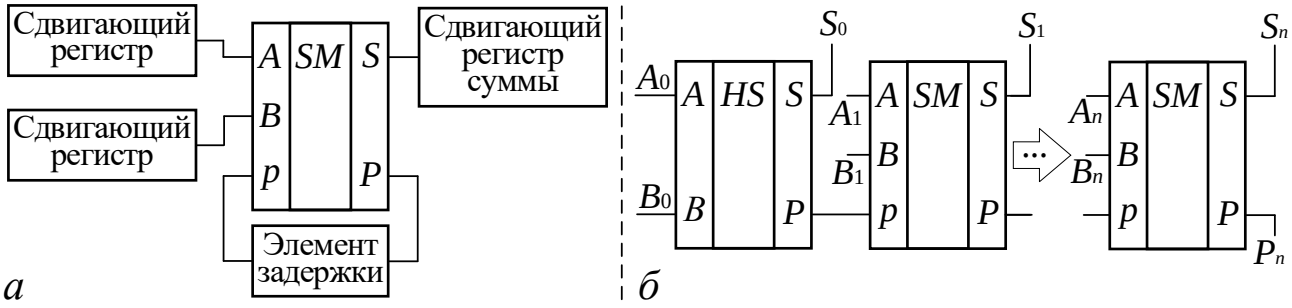


Рисунок 3.3 – Одноразрядный сумматор

3 *Многоразрядный сумматор* – узел, построенный на базе одноразрядных сумматоров с подключением соответствующих выходов к соответствующим входам («*p*» к «*P*») и выполняющий сложение многоразрядных двоичных чисел. Многоразрядные сумматоры бывают с последовательным (рисунок 3.4, *а*) и со сквозным переносом (параллельный) (рисунок 3.4, *б*).



а – последовательный; *б* – параллельный

Рисунок 3.4 – Многоразрядный сумматор

4 *Сумматор по модулю 2* – узел, реализующий поразрядное сложение по модулю 2 (логическую операцию исключающее ИЛИ).

5 *Шифратор* (*кодер*) – узел, преобразующий сигнал на одном из n входов в комбинацию сигналов на m выходах и использующийся в случае, когда необходимо передавать большое количество данных при небольшом количестве линий связи. При этом количество входов n и выходов m связано соотношением $2^m \geq n$. Также имеет вход синхронизации.

6 *Дешифратор* (*декодер*) – узел, преобразующий комбинацию сигналов на n входах в сигнал на одном из m выходов, т. е. с помощью n входов можно задавать выход, на который будет подаваться единичный сигнал. Количество входов n и выходов m связано соотношением $2^n \geq m$. Также имеет вход синхронизации. Дешифратор применяется для построения мультиплексора и демультиплексора.

7 *Мультиплексор* – узел, обеспечивающий подключение одного из $2^n + n$ своих входов ($x_0, x_1, \ldots, x_{2n-1}, s_0, s_1, \ldots, s_{n-1}$) к единственному выходу m, на который подается значение на входе x_i, где i – число, которое кодируется входами $s_0, s_1, \ldots, s_{n-1}$. Также имеет вход синхронизации. Используется при мультиплексированной передаче данных в шинных архитектурах связей.

8 *Демультиплексор* – узел, обеспечивающий логическое подключение одного входа к одному из нескольких выходов, т. е. обратную мультиплексору функцию. Также имеет вход синхронизации.

Накапливающие узлы строятся на базе триггеров, каждый из которых предназначен для хранения отдельного разряда.

Триггер – это логическое устройство последовательного типа, предназначенное для записи и хранения информации.

Любой триггер имеет два выхода: прямой Q и инверсный $\bar{Q}$. Число входов триггера зависит от его типа.

Классификация триггеров представлена на рисунке 3.5.

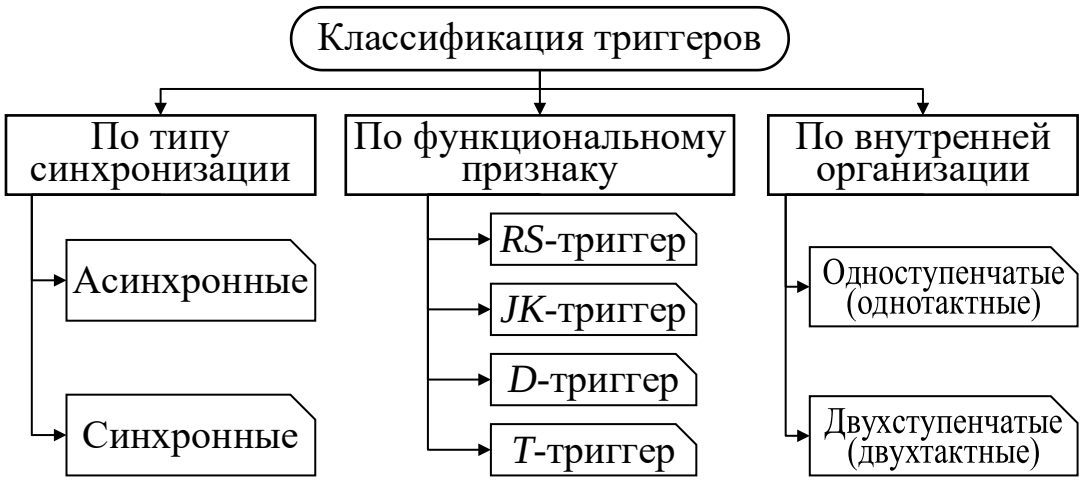


Рисунок 3.5 – Классификация триггеров

Асинхронные триггеры изменяют свое состояние (на выходе Q) в момент появления информационного сигнала на соответствующем входе.

Синхронные триггеры изменяют свое состояние (на выходе Q) в соответствии с информационными сигналами на входе, но только при наличии информационного сигнала на входе синхронизации (С).

Синхронизация может выполняться различными способами:

– по высокому уровню сигнала (на рисунке 3.6 обозначено буквой *а*);

– по низкому уровню сигнала (на рисунке 3.6 обозначено буквой *б*);

– по переднему фронту (при переходе с низкого уровня сигнала на высокий) (на рисунке 3.6 обозначено буквой *в*);

– по заднему фронту (при переходе с высокого уровня сигнала на низкий) (на рисунке 3.6 обозначено буквой *г*).

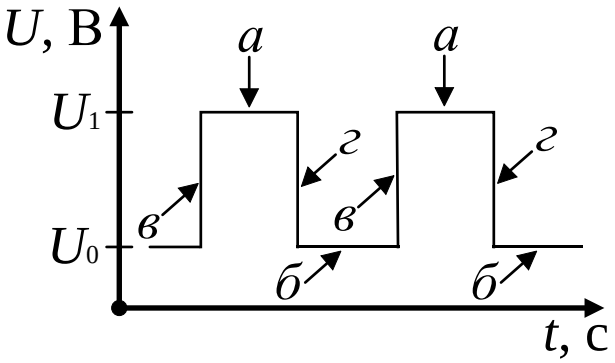


Рисунок 3.6 – Способы синхронизации синхронных триггеров

Одноступенчатые (*однотактные*) *триггеры* – это триггеры, в которых запоминание информации происходит за один такт.

Двухступенчатые (*двухтактные триггеры*) – это триггеры, в которых запоминание информации происходит за два такта (на двух ступенях) (рисунок 3.7).

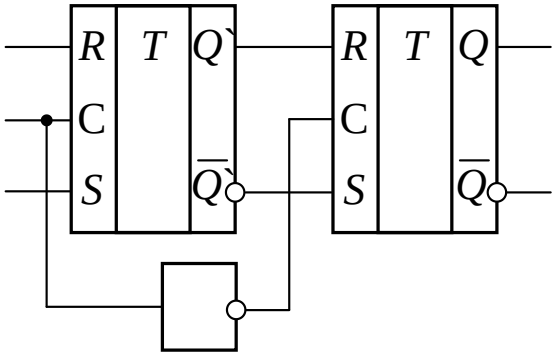


Рисунок 3.7 – Двухступенчатый (двухтактный, *TT*-триггер) синхронный *RS*-триггер

Как видно из рисунка 3.7, по сигналу синхронизации информация записывается в первый триггер, затем уже по инверсному сигналу синхронизации перезаписывается во второй триггер и появляется на его выходе *Q*.

По функциональному признаку триггеры бывают следующих типов:

1 *RS-триггер*. Имеет два входа q_R (*reset*, сброс) и q_S (*set*, установка) и два выхода Q и $\bar{Q}$ (обратное значение выхода Q, т. е. инверсия разряда). Для установки триггера на значение «0» (Q = 0, $\bar{Q}$ = 1) необходимо подать единичный сигнал на вход q_R, а для установки триггера в значение «1» (Q = 1, $\bar{Q}$ = 0) необходимо подать единичный сигнал на вход q_S. Комбинация из двух единичных сигналов на обоих входах запрещена.

Асинхронный *RS*-триггер может быть реализован на двух логических элементах ИЛИ-НЕ (рисунок 3.8, *а*) с переключением сигналом «1» или на двух логических элементах И-НЕ (рисунок 3.8, *б*) с переключением сигналом «0». Синхронный *RS*-триггер на четырех элементах И-НЕ представлен на рисунке 3.8, *в*.

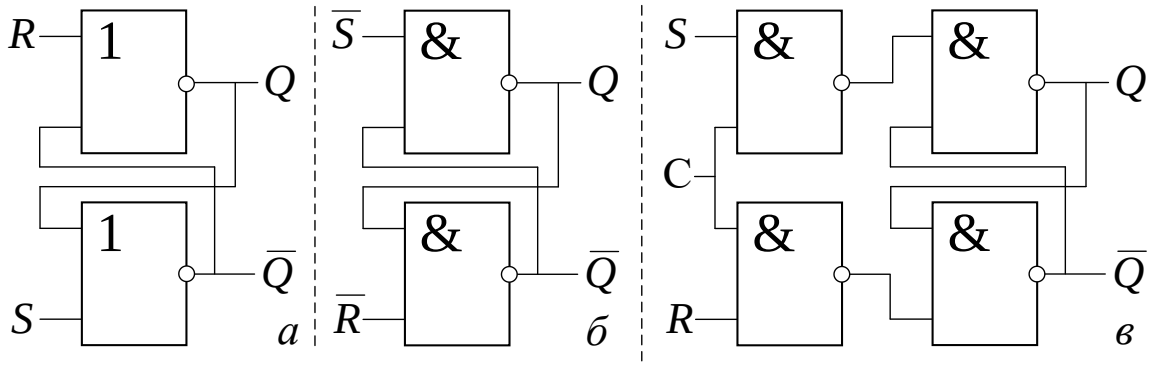


Рисунок 3.8 – Примеры *RS*-триггеров на логических элементах

Ниже представлена таблица состояний *RS*-триггера (таблица 3.1).

Таблица 3.1 – Таблица состояний входов, выходов и режимов работы *RS*-триггера

Вход C	**Вход *R***	**Вход *S***	**Выход $Q_{предыдущее}$**	**Выход $Q_{текущее}$**	**Режим работы**
Асинхронный RS-триггер					
–	0	0	0	0	Хранение информации
–	0	0	1	1	
–	0	1	0	1	Запись 1
–	0	1	1	1	
–	1	0	0	0	Запись 0
–	1	0	1	0	
–	1	1	0	x	Запрещенная комбинация
–	1	1	1	x	
Синхронный RS-триггер					
0	x	x	0	0	Хранение информации
0	x	x	1	1	
1	0	0	0	0	Хранение информации
1	0	0	1	1	
1	0	1	0	1	Запись 1
1	0	1	1	1	
1	1	0	0	0	Запись 0
1	1	0	1	0	
1	1	1	0	x	Запрещенная комбинация
1	1	1	1	x	

2 *JK-триггер*. Имеет два входа q_J и q_K и два выхода Q и $\bar{Q}$. Аналогичен *RS*-триггеру, только комбинация из двух единичных сигналов на обоих входах не запрещена и переводит выходы *JK*-триггера в противоположное состояние (аналогично *T*-триггеру). *JK*-триггер также называют универсальным триггером.

JK-триггер может быть только синхронным. На базе *JK*-триггера можно построить любые другие триггеры. Сам же он может быть построен на базе двухступенчатого синхронного *RS*-триггера, двух элементов И и одного элемента НЕ (рисунок 3.9, *а*). Пример *JK*-триггера на двух элементах 3И-НЕ и двух И-НЕ представлен на рисунке 3.9, *б*.

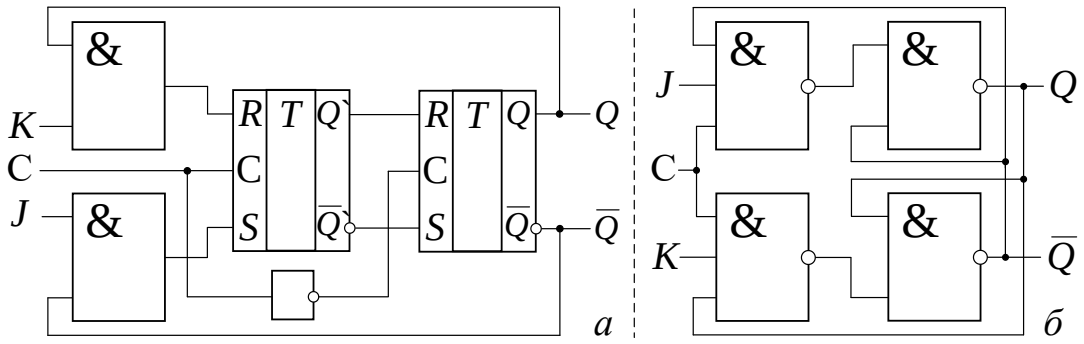


Рисунок 3.9 – *JK*-триггер

Ниже представлена таблица состояний *JK*-триггера (таблица 3.2).

Таблица 3.2 – Таблица состояний входов, выходов и режимов работы *JK*-триггера

Вход C	**Вход *J***	**Вход *K***	**Выход $Q_{\text{предыдущее}}$**	**Выход $Q_{\text{текущее}}$**	**Режим работы**
0	x	x	0	0	Хранение информации
0	x	x	1	1	
1	0	0	0	0	
1	0	0	1	1	
1	0	1	x	1	Запись 1
1	1	0	x	0	Запись 0
1	1	1	0	1	Счетный режим
1	1	1	1	0	

3 *D-триггер*. Имеет один вход q_D (*digital*, цифра) и два выхода Q и $\bar{Q}$. Для установки триггера в значение «0» ($Q = 0$, $\bar{Q} = 1$) необходимо подать нулевой сигнал на вход q_D, а для установки триггера в значение «1» ($Q = 1$, $\bar{Q} = 0$) необходимо подать единичный сигнал на вход q_D. То есть триггер хранит значение, поданное на вход. Также его называют триггером-защелкой.

D-триггер всегда синхронный. Он может быть реализован на базе *RS*-триггера (рисунок 3.10, *а*). *D*-триггер на четырех элементах И-НЕ представлен на рисунке 3.10, *б*.

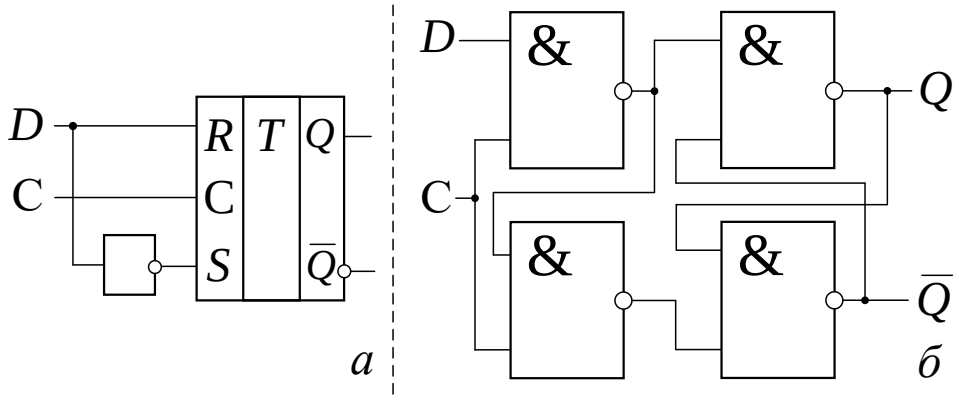


Рисунок 3.10 – *D*-триггер

Ниже представлена таблица состояний *D*-триггера (таблица 3.3).

Таблица 3.3 – Таблица состояний входов, выходов и режимов работы *D*-триггера

Вход C	**Вход *D***	**Выход $Q_{\text{предыдущее}}$**	**Выход $Q_{\text{текущее}}$**	**Режим работы**
0	x	$Q_{\text{предыдущее}}$	$Q_{\text{предыдущее}}$	Хранение
1	1	Не имеет значения	1	Запись 1
1	0	Не имеет значения	0	Запись 0

4 *T-триггер*. Имеет один вход q_T и два выхода Q и $\bar{Q}$. Единичный сигнал, поданный на вход q_T, переводит выходы *T*-триггера в противоположное состояние. Также его называют «счетным триггером». *T*-триггер всегда асинхронный.

T-триггер можно построить на основе *RS*-триггера (рисунок 3.11, *а*) или *D*-триггера (рисунок 3.11, *б*). *T*-триггер на четырех элементах И-НЕ представлен на рисунке 3.11, *в*.

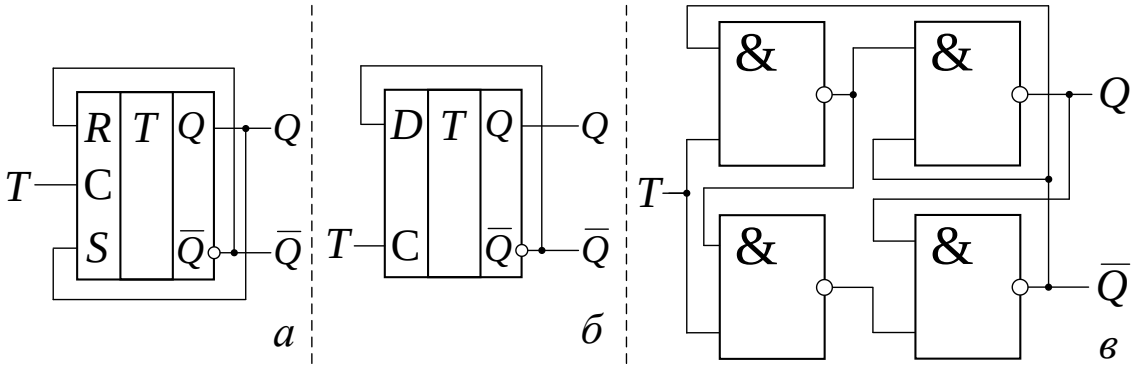


Рисунок 3.11 – Асинхронный *T*-триггер

Ниже представлена таблица состояний *T*-триггера (таблица 3.4).

Таблица 3.4 – Таблица состояний входов, выходов и режимов работы *T*-триггера

Вход *T*	**Выход $Q_{предыдущее}$**	**Выход $Q_{текущее}$**	**Режим работы**
0	1	1	Хранение информации
0	0	0	
1	1	0	Режим переброса
1	0	1	

Зависимость хранимого триггерами значения от сигналов на входах наглядно показана в обобщенной таблице 3.5.

Таблица 3.5 – Зависимость хранимого триггером значения от сигналов на входах

***RS*-триггер**				***T*-триггер**			***JK*-триггер**				***D*-триггер**		
Входы триггера		Выходы триггера		Входы триггера	Выходы триггера		Входы триггера		Выходы триггера		Входы триггера	Выходы триггера	
q_R	q_S	Q	$\bar{Q}$	q_T	Q	$\bar{Q}$	q_J	q_K	Q	$\bar{Q}$	q_D	Q	$\bar{Q}$
0	0	Хранение текущих значений Q и $\bar{Q}$		0	Хранение текущих значений Q и $\bar{Q}$		0	0	0	1	0	0	1
1	0	0	1				1	0	0	1			
0	1	1	0	1	Инверсия текущих значений Q и $\bar{Q}$		0	1	1	0	1	1	0
1	1	Недопустимая комбинация q_R и q_S					1	1	Инверсия текущих значений Q и $\bar{Q}$				

К **накапливающим узлам** относятся:

1 *Регистр* – узел, предназначенный для хранения многоразрядных значений (отдельного числа).

Регистр – группа триггеров, количество которых соответствует количеству разрядов в хранимом регистром двоичном числе.

Разрядность регистра определяется архитектурой ЭВМ.

Машинное слово – двоичное число, хранимое в регистре.

Для упрощения доступа к регистрам они имеют уникальные номера. Этот уникальный номер называется *адресом регистра*.

Совокупность всех регистров основной памяти – это *оперативная память*.

2 *Счетчик* – узел, предназначенный для хранения, как правило, многоразрядных значений, который по каждому сигналу изменяет (увеличивает на единицу) хранимый код числа.

Счетчики могут работать в двух режимах: в режиме счетчика событий и в режиме таймера. В режиме счетчика происходит увеличение на единицу хранимого значения по каждому единичному сигналу, поступающему на вход счетчика по факту наступления какого-либо события. В режиме таймера счетчик считает тактовые импульсы процессора или импульсную последовательность с выхода управляемого делителя частоты (например, для счета времени).

Существует множество различных счетчиков/таймеров, каждый из которых в ЭВМ выполняет свою функцию, например:

– счетчик команд – служит для хранения адреса ячейки оперативной памяти (по сути регистра) текущей команды, выполняемой процессором;

– сторожевой таймер – предназначен для защиты микропроцессорной системы (МПС) от зависания программы. При запуске на выполнение программы сторожевой таймер начинает считать тактовые импульсы. Сторожевой таймер сбрасывается на программном уровне, т. е. при зависании программы на аппаратном уровне еще продолжится счет тактов и при достижении счетчиком максимального кода (в зависимости от разрядности) генерируется сигнал внутреннего сброса (внутреннее прерывание), перезагружающий МПС.

3.3 Контрольные вопросы

1 Какие бывают виды основных узлов ЭВМ?

2 Какие узлы относятся к комбинационным, а какие к накапливающим?

3 Какие виды триггеров бывают?

4 В чем заключается отличие *RS*- от *JK*-триггера?

5 Что такое регистр?

6 В каких режимах может работать счетчик?

4 ВВЕДЕНИЕ В ТЕОРИЮ КОНЕЧНЫХ АВТОМАТОВ

4.1 Основные понятия теории конечных автоматов

Цифровой автомат – это устройство, которое выполняет прием, хранение и преобразование информации по определенному алгоритму.

К цифровым автоматам могут относиться реальные устройства или абстрактные машины. К реальным относятся вычислительные машины, живые организмы и т. п., а к абстрактным – математические машины, аксиоматические теории и др. Теорию автоматов разделяют на структурную и абстрактную. Абстрактная теория не затрагивает способ его построения, т. е. структуры самого автомата, и касается только поведения автомата относительно внешних данных. Другими словами, абстрактная теория изучает только те переходы, которые автомат претерпевает под воздействием входных сигналов, и те выходные сигналы, которые автомат выдает.

Структурная теория в отличие от абстрактной интересуется структурой самого автомата, а также структурой входных воздействий на автомат и реакций автомата на них. Структурная теория включает способы кодирования входных воздействий и реакций автомата, а также способы построения автоматов. Структурная теория является условным продолжением абстрактной. На базе абстрактной теории автоматов и основ булевых функций структурная теория предоставляет эффективные рекомендации по разработке реальных устройств вычислительной техники.

Абстрактный цифровой автомат S представляет собой математическую модель, которая определяется совокупностью пяти объектов $\{Z, A, W, \delta, \lambda\}$:

1) конечное множество Z входных сигналов, представляющих собой входной алфавит автомата:

$Z = \{z_1, z_2, \dots, z_m\}$;

2) конечное множество W выходных сигналов, представляющих собой выходной алфавит автомата:

$W = \{w_1, w_2, \dots, w_n\}$;

3) конечное множество A состояний автомата:

$A = \{a_1, a_2, \dots, a_s\}$;

4) функция перехода автомата из одного состояния в другое, $\delta(a, z)$;

5) функция выходов автомата, $\lambda(a, z)$.

Абстрактный цифровой автомат называют *инициальным*, если на множестве его состояний A выделяется особое состояние a_0, соответствующее

начальному состоянию автомата. Определение начального состояния объясняется практической необходимостью задания условий начала работы автомата. Таким образом, можно считать, что инициальный цифровой автомат определяется совокупностью шести объектов:

$S = \{Z, A, W, \delta, \lambda, a_0\}$.

Рассмотренную модель абстрактного автомата можно увидеть на рисунке 4.1.

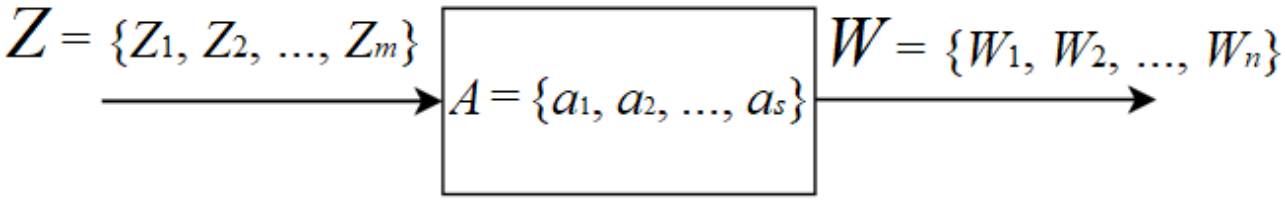


Рисунок 4.1 – Абстрактная модель автомата с одним входом и одним выходом

Функция переходов $\delta(a, z)$ показывает, что автомат, находясь в некотором состоянии a_i при появлении входного сигнала z_j, переходит в некоторое состояние a_k:

$a_k = \delta(a_i, z_j)$.

Функция выходов $\lambda(a, z)$ показывает, что автомат, находясь в некотором состоянии a_i при появлении входного сигнала z_j, выдает выходной сигнал w_l:

$w_l = \lambda(a_i, z_j)$.

По способу формирования функции выходов выделяют три типа абстрактных автоматов:

1) автомат Мили;

2) автомат Мура;

3) *C*-автомат.

В абстрактном автомате Мили выходной сигнал однозначно определяется входным сигналом и состоянием автомата до текущего момента времени. Такой автомат также называют автоматом первого рода.

В абстрактном автомате Мура выходной сигнал однозначно определяется входным сигналом и состоянием автомата в текущий момент времени. Такой автомат также называют автоматом второго рода.

C-автомат представляет собой совмещенную модель. Для данного автомата вводятся две функции выходов: λ_1 и λ_2. При этом функция λ_1 соответствует поведению функции выходов автомата первого рода, а функция λ_2 – второго рода (рисунок 4.2).

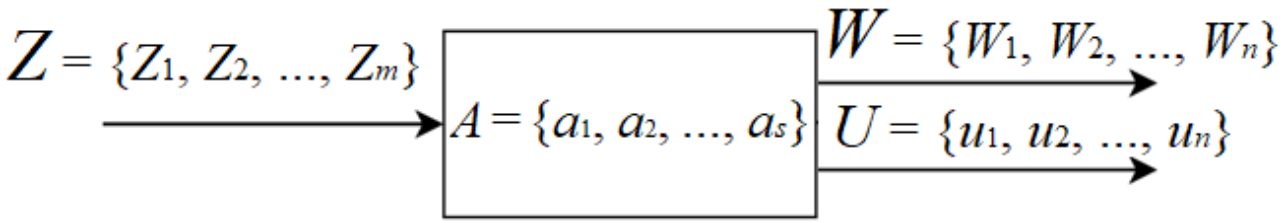


Рисунок 4.2 – Абстрактная модель *C*-автомата с одним входом и двумя выходами

С точки зрения определения или задания абстрактные цифровые автоматы могут быть полностью определенными или частично определенными.

Полностью определенным называют такой абстрактный цифровой автомат, для которого функция переходов $\delta(a, z)$ и функция выходов $\lambda(a, z)$ определены для всех пар a_i, z_j.

Частично определенным называют такой абстрактный цифровой автомат, для которого функция переходов $\delta(a, z)$, или функция выходов $\lambda(a, z)$, или обе эти функции определены не для всех пар a_i, z_j.

Подразумевается, что абстрактный автомат функционирует в дискретном автоматном времени. Это означает, что функционирование автомата рассматривается через дискретные интервалы времени конечной продолжительностью или интервалы дискретности. Переключения автоматов в новые состояния и формирование выходных сигналов осуществляется моментально. Каждому интервалу дискретности t соответствует свой выходной сигнал $w(t)$. При этом выходной сигнал на выходах автомата появляется только после появления в этот же интервал текущего времени t входного сигнала $z(t)$ с одновременным переходом автомата из состояния $a(t)$ в состояние $a(t+1)$, где $t+1$ – это следующий интервал времени.

На основании этого автомат Мили можно описать следующей системой уравнений:

$$\begin{cases} w(t) = \lambda\big(a(t), z(t)\big); \\ a(t+1) = \delta\big(a(t), z(t)\big). \end{cases}$$

Автомат Мура:

$$\begin{cases} w(t) = \lambda\big(a(t)\big); \\ a(t+1) = \delta\big(a(t), z(t)\big). \end{cases}$$

C-автомат:

$$\begin{cases} w(t) = \lambda_1\big(a(t), z(t)\big); \\ u(t) = \lambda_2\big(a(t)\big); \\ a(t+1) = \delta\big(a(t), z(t)\big). \end{cases}$$

Таким образом, если на вход инициального автомата подавать некоторую последовательность букв входного алфавита $z(0), z(1)$, …, образующих входное слово, то на выходе автомата последовательно будут формироваться буквы выходного алфавита $w(0), w(q)$, ..., образующие выходное слово. Для *C*-автомата, на выходе формируется две последовательности: $w(0), w(1)$, ... и $u(0), u(1)$, ...

На уровне абстрактной модели функционирование автомата можно описать преобразованием входных слов в выходные слова. Состояния автоматов необходимы для понимания реализуемыми процессами предыстории своего развития. Например, работа светофора: в случае если на вход автомата подается только сигнал «переключить светофор», то включение определенного светового сигнала основывается на том, какой сигнал светофора был включен в предшествующий момент. Организация хранения предшествующих состояний возможна только при условии наличия у цифровых автоматов памяти. Однако существуют процессы, которыми управляют реальные автоматы, которые не требуют предыстории развития процесса во времени. То есть выходной сигнал автомата определяется лишь входным сигналом. Например, электрический фонарь, где входным сигналом является нажатие клавиши «Включить», а выходным – зажигание лампочки. В таком случае автоматы определяются тремя объектами $S = \{Z, W, \lambda\}$, где λ функция формирования выходного сигнала W на основе входного сигнала Z. Такие автоматы называются абстрактными автоматами с тривиальной памятью или комбинационными автоматами.

4.2 Способы задания автоматов

Задание абстрактных цифровых автоматов является ключевым этапом при проектировании и анализе различных цифровых систем и устройств. Входы, состояния автоматов и выходы обычно задаются множествами. Для задания функций переходов и функций выходов автомата существуют различные способы, например с помощью матриц или графических представлений, также могут использоваться аналитические представления.

Матричный способ задания автомата

Матричный способ подразумевает использование таблиц для задания автоматов, а именно для задания цифрового автомата могут использоваться *таблица переходов состояний* и *таблица выходов состояний*.

Таблица переходов идентична для различных типов автоматов (Мили, Мура, *C*-автомат) и состоит из столбцов и строк, где столбцам соответствуют буквы алфавита состояний автоматов $A = \{a_1, a_2, \dots, a_s\}$, а строкам – буквы алфавита входных сигналов $Z = \{z_1, z_2, \dots, z_m\}$. В клетке таблицы переходов,

соответствующей пересечению столбца с состоянием a_i и строки с входным сигналом z_j, проставляется состояние a_k, где $a_k = \delta(a_i, z_j)$ и является результатом перехода состояния автомата a_i в состояние a_k при воздействии входного сигнала z_j. Например, для полностью определенного конечного автомата с определенным входным множеством $Z = \{z_1, z_2, z_3\}$ и множеством состояний $A = \{a_1, a_2, a_3, a_4\}$ таблица переходов примет вид, представленный в таблице 4.1.

Таблица 4.1 – Таблица переходов для полностью определенного автомата

Входные сигналы	**Состояния**			
	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$	$\boldsymbol{a_3}$	$\boldsymbol{a_4}$
z_1	a_1	a_2	a_3	a_2
z_2	a_1	a_4	a_1	a_4
z_3	a_3	a_1	a_2	a_1

Для частично определенного автомата в клетке таблицы переходов, соответствующей пересечению столбца с состоянием a_i и строки с входным сигналом z_j, где для пары a_i, z_j состояние автомата не определено, проставляется прочерк. При этом входящее слово, при котором осуществляется этот переход, является запрещенным. Состояния в остальных клетках проставляются в соответствии с алгоритмом для полностью определенного автомата. Например, для частично определенного конечного автомата с определенным входным множеством $Z = \{z_1, z_2, z_3\}$ и множеством состояний $A = \{a_1, a_2, a_3, a_4\}$ таблица переходов примет вид, представленный в таблице 4.2.

Таблица 4.2 – Таблица переходов для частично определенного автомата

Входные сигналы	**Состояния**			
	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$	$\boldsymbol{a_3}$	$\boldsymbol{a_4}$
z_1	a_1	a_2	–	a_2
z_2	–	a_4	a_1	a_4
z_3	a_3	a_1	a_2	a_1

Таблицы выходов в отличие от таблиц переходов не одинаковы для различных типов задаваемых автоматов.

Таблица выходов для полностью определенного автомата Мили строится по следующему принципу. Заголовки строк и столбцов полностью соответствуют заголовкам таблицы переходов автомата. В клетке таблицы выходов, соответствующей пересечению столбца с состоянием a_i и строки с входным

сигналом z_j, проставляется выходной сигнал w_n, где $w_n = \lambda(a_i, z_j)$ и является результатом выхода автомата Мили, находящегося в состоянии a_i при воздействии входного сигнала z_j. Например, для полностью определенного конечного автомата с определенным входным множеством $Z = \{z_1, z_2, z_3\}$ и множеством состояний $A = \{a_1, a_2, a_3, a_4\}$ таблица выходов примет вид, представленный в таблице 4.3.

Таблица 4.3 – Таблица выходов для полностью определенного автомата Мили

Входные сигналы	**Состояния**			
	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$	$\boldsymbol{a_3}$	$\boldsymbol{a_4}$
z_1	w_2	w_4	w_1	w_2
z_2	w_1	w_3	w_1	w_2
z_3	w_4	w_1	w_3	w_1

Таблица выходов для автомата Мура строится проще. Каждому состоянию автомата ставится в соответствие выходной сигнал, присущий этому состоянию (таблица 4.4).

Таблица 4.4 – Таблица выходов для полностью определенного автомата Мура

Состояния	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$	$\boldsymbol{a_3}$	$\boldsymbol{a_4}$
Выходные сигналы	w_2	w_3	w_1	w_2

Абстрактный цифровой *C*-автомат задается двумя таблицами выходов, одна из которых соответствует таблице автомата Мили, другая – автомата Мура.

В случае частично определенного автомата Мили в клетке таблицы выходов, соответствующей пересечению столбца с состоянием a_i и строки с входным сигналом z_j, где для пары a_i, z_j выходной сигнал автомата не определен, ставится прочерк. При этом выходной сигнал должен быть не определен для пар a_k, z_l, для которых также не определено состояние в таблице переходов автомата Мили. Или, другими словами, если в частично определенном автомате Мили существует пара a_k, z_l, для которой переход из состояния a_k при воздействии входного сигнала z_l не определен, то не определено и значение выходного сигнала на таком переходе. Пример таблицы выходов частично определенного автомата Мили, заданного таблицей переходов (см. таблицу 4.2), представлен в таблице 4.5.

Таблица 4.5 – Таблица выходов для частично определенного автомата Мили

Входные сигналы	**Состояния**			
	a_1	a_2	a_3	a_4
z_1	w_2	w_4	–	w_3
z_2	–	w_1	w_2	–
z_3	w_1	w_3	–	w_4

С автоматом Мура иная ситуация. Неопределенные выходные сигналы из таблицы выходов не связаны с неопределенными состояниями из таблицы переходов автомата Мура. Это связано с тем, что выходной сигнал автомата Мура зависит только от состояния, в котором находится автомат. Поэтому таблица выходов частично определенного автомата Мура может быть задана как аналогичной таблицей для полностью определенного автомата Мура (см. таблицу 4.4), где для всех состояний определены выходные сигналы, так и иной, в которой некоторые выходные сигналы могут быть не определены (таблица 4.6).

Таблица 4.6 – Таблица выходов для частично определенного автомата Мура

Состояния	a_1	a_2	a_3	a_4
Выходные сигналы	w_2	w_3	w_1	–

Зачастую таблицы переходов и выходов различных автоматов удобно совмещать в одну. Таким образом, полностью определенный автомат Мили, заданный таблицей состояний (см. таблицу 4.1) и таблицей выходов (см. таблицу 4.3), можно задать совмещенной таблицей (таблица 4.7).

Таблица 4.7 – Совмещенная таблица (переходов/выходов) для полностью определенного автомата Мили

Входные сигналы	**Состояния**			
	a_1	a_2	a_3	a_4
z_1	w_2/a_1	w_4/a_2	w_1/a_3	w_2/a_2
z_2	w_1/a_1	w_3/a_4	w_1/a_1	w_2/a_4
z_3	w_4/a_3	w_1/a_1	w_3/a_2	w_1/a_1

Аналогично частично определенный автомат Мура, заданный таблицей состояний (см. таблицу 4.2) и таблицей выходов (см. таблицу 4.4), можно задать совмещенной таблицей (таблица 4.8).

Таблица 4.8 – Совмещенная таблица для частично определенного автомата Мура

Выходные сигналы		w_2	w_3	w_1	w_2
Состояния		a_1	a_2	a_3	a_4
Входные сигналы	z_1	a_1	a_2	–	a_2
	z_2	–	a_4	a_1	a_4
	z_3	a_3	a_1	a_2	a_1

Графический способ задания автомата

Кроме рассмотренных таблиц переходов и выходов, произвольный абстрактный автомат может быть задан графическим способом. В таком случае абстрактный автомат представляется ориентированным графом, где вершины графа – состояния автомата, а дуги – переходы между состояниями. Каждая дуга подписывается значением z_i, что означает, что переход из одной вершины во вторую (из одного состояния в другое) происходит при внешнем воздействии входного сигнала z_i. Помимо этого, в случае задания автомата Мили для каждой дуги также проставляется значение w_j, означающее, что при данном переходе на выходе автомата устанавливается сигнал w_j. В случае задания автомата Мура значения выходов автомата прописываются рядом с вершинами графа. Пример графов для автомата Мили и автомата Мура, заданных таблицами 4.7, 4.8, представлен на рисунках 4.3 и 4.4 соответственно.

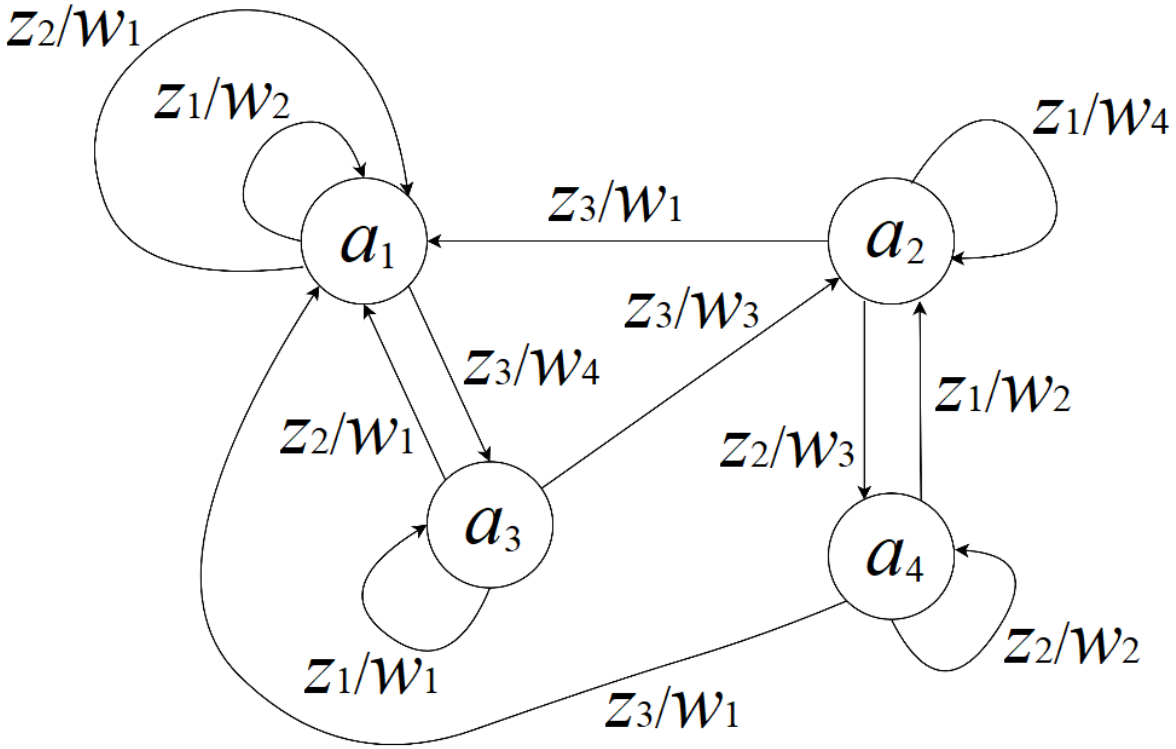


Рисунок 4.3 – Пример задания абстрактного автомата Мили с помощью графа

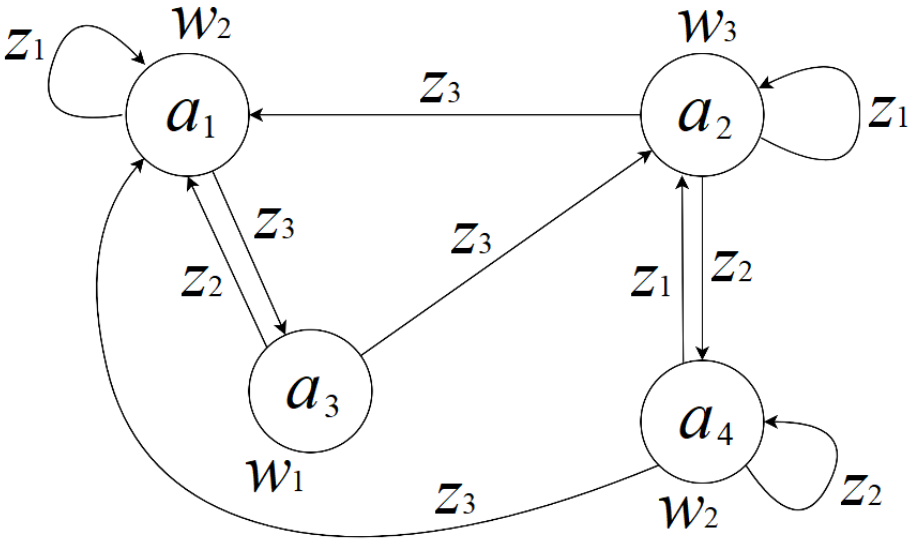


Рисунок 4.4 – Пример задания абстрактного автомата Мура с помощью графа

Состояние автомата, из которого можно выйти, но в которое нельзя перейти, называется *переходящим*. Обратное состояние автомата, т. е. то, из которого нельзя выйти, но в которое можно перейти, называется *тупиковым*.

Изолированным состоянием автомата называется такое состояние, в которое нельзя перейти и одновременно с этим из которого нельзя выйти. Для автомата, представленного с помощью графа, это означает, что вершина графа не имеет ни входящих, ни исходящих дуг.

В качестве примера можно рассмотреть построение абстрактного цифрового автомата для управления переключением сигнала светофора. Предполагается, что на вход автомата подается сигнал, инициирующий переключение светофора (сигнал «переключить светофор»). Таким образом, возможны два варианта входящего сигнала: наличие сигнала о переключении (z_1) и его отсутствие (z_2). Выходные сигналы представляют собой сигналы управления для включения соответствующего света светофора: красный (w_1), желтый (w_2) и зеленый (w_3). Также следует определить возможные состояния автомата. Для зеленого и красного света отдельные состояния очевидны, а для желтого возможны два варианта: он загорается после красного и после зеленого. Помимо того, следует определиться с начальным состоянием светофора. Например, включен красный свет. Таким образом, состояния автомата будут следующие: a_0 – состояние автомата, соответствующее включенному красному свету; a_1 – состояние автомата, соответствующее включенному желтому свету, предшествующее включению зеленого света; a_2 – состояние автомата, соответствующее зеленому свету; a_3 – состояние автомата, соответствующее желтому свету и предшествующее включению красного света. Графы абстрактных автоматов Мили и Мура, реализующих работу светофора, представлены на рисунках 4.5 и 4.6 и в таблицах 4.9 и 4.10 соответственно.

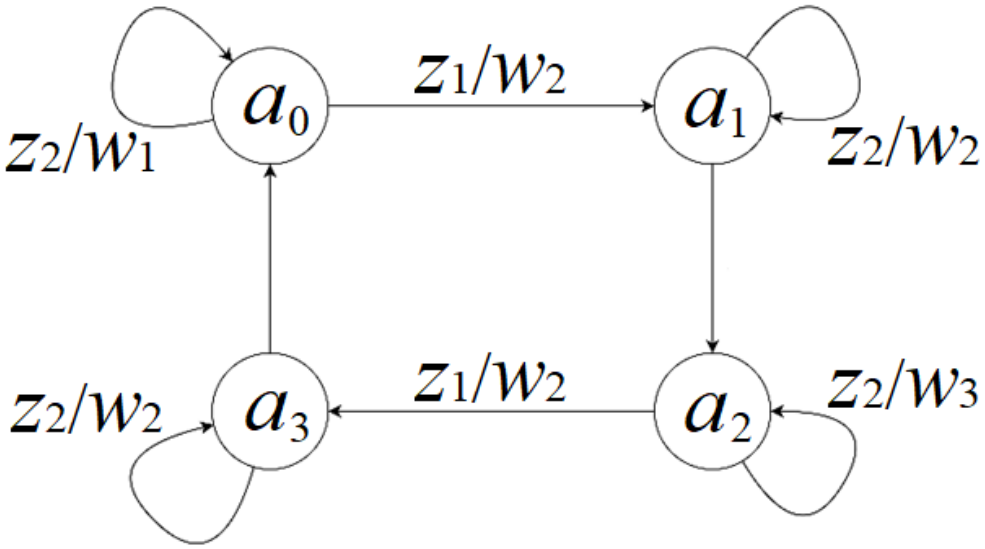


Рисунок 4.5 – Абстрактный автомат Мили, заданный графом и реализующий процесс переключения сигналов светофора

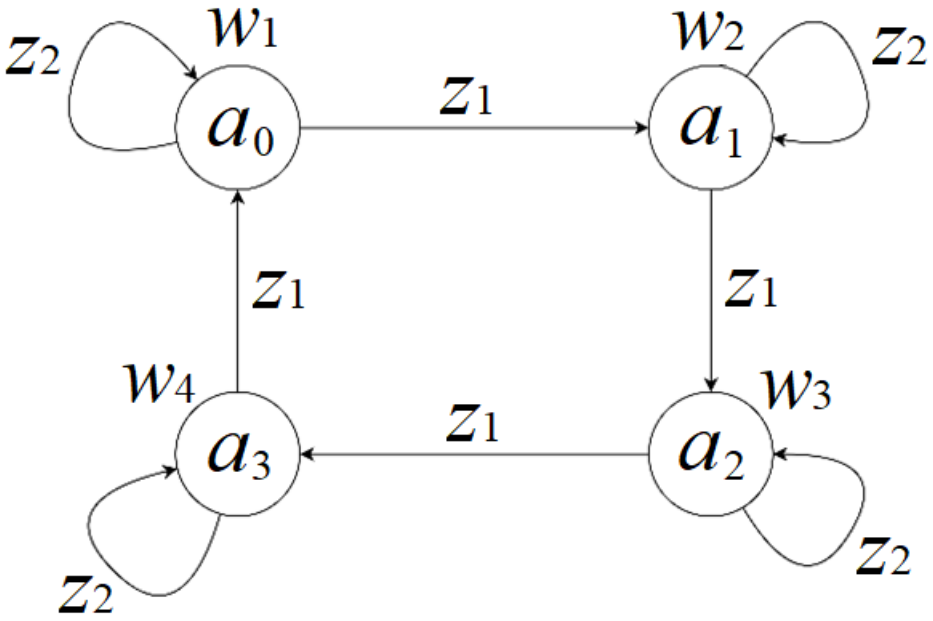


Рисунок 4.6 – Абстрактный автомат Мура, заданный графом и реализующий процесс переключения сигналов светофора

Таблица 4.9 – Автомат Мили, реализующий работу переключения светофора

Входные сигналы	**Состояния**			
	a_0	a_1	a_2	a_3
z_1	w_1/a_3	w_2/a_0	w_3/a_1	w_2/a_2
z_2	w_1/a_0	w_2/a_1	w_3/a_2	w_2/a_3

Таблица 4.10 – Автомат Мура, реализующий работу переключения светофора

Выходные сигналы		w_1	w_2	w_3	w_2
Состояния		a_0	a_1	a_2	a_3
Входные сигналы	z_1	a_3	a_0	a_1	a_2
	z_2	a_0	a_1	a_2	a_3

Для рассмотренного примера абстрактные автоматы Мили и Мура очень схожи и представляют собой эквивалентные автоматы.

Эквивалентными называются автоматы, которые в результате подачи на вход одного и того же набора данных (входного алфавита) на выходе предоставляют одинаковые наборы выходных данных (выходного алфавита).

4.3 Канонический метод синтеза

В общем случае задача синтеза структурного автомата сводится к созданию комбинационной схемы с несколькими выходами. Рассмотрим один из методов синтеза, который позволяет упростить эту задачу, превращая ее в задачу синтеза комбинационных схем. Этот метод, известный как канонический метод структурного синтеза автоматов с памятью, оперирует элементарными автоматами, разделенными на два основных класса. Первый класс включает элементарные автоматы с памятью, называемые элементами памяти, а второй – элементарные комбинационные автоматы, или логические элементы (логика цифрового автомата). Общая схема такого автомата представлена на рисунке 4.7.

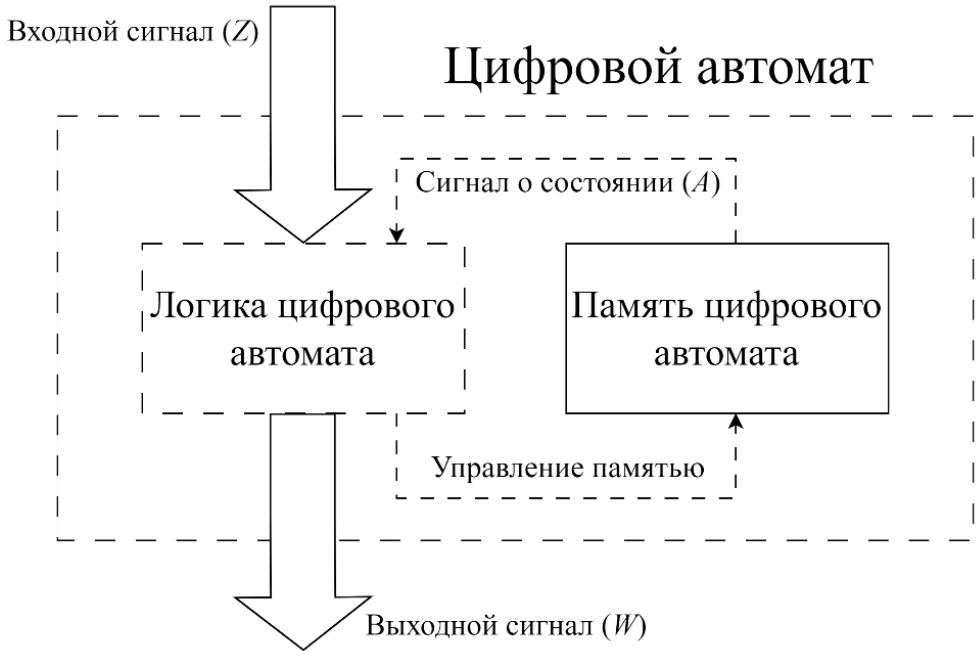


Рисунок 4.7 – Структурная схема автомата

Память содержит данные о предыдущих состояниях цифрового автомата. Логическая часть, основываясь на входном сигнале и информации из памяти, формирует выходной сигнал. Сигнал управления памятью формируется с использованием входного сигнала и сигнала состояния, что обеспечивает переход из текущего состояния автомата в новое.

Память реализуется с использованием триггеров, а сигналы управления памятью должны соответствовать переходам и типу триггеров.

Результатом работы метода являются уравнения булевых функций автоматов в канонической форме представления. Начальными данными для данного метода служит абстрактный цифровой автомат с памятью.

Алгоритм синтеза цифрового автомата представляет собой следующий процесс:

1 Кодирование.

2 Выбор элементов памяти.

3 Построение уравнений булевых функций выходов и состояний автомата, включая минимизацию уравнений.

4 Построение функциональной схемы автомата.

Кодирование

Рассмотрим цифровой автомат с состояниями, описываемыми как $S = \{Z, A, W, \delta, \lambda\}$. При переходе на структурный уровень представления каждая буква z_i из входного алфавита Z автомата представляется в виде двоичного вектора или набора, количество компонентов которого соответствует числу физически реализованных входных каналов автомата. Иными словами, каждая буква z_i из $Z = \{z_1, z_2, \dots, z_n\}$ кодируется двоичным вектором $X = \{x_1, x_2, \dots, x_k\}$. Очевидно, минимальное число физически реализованных входных каналов k в автомате определяется как минимальная целая степень двойки, для которой $2^k \geq n$ или $k =]log_2|Z|[$, где $|Z|$ – количество элементов в множестве; $]n[$ – ближайшее целое число, большее либо равное n. Например, для $Z = \{z_1, z_2, z_3\}$, $k = 2$.

Аналогичным образом кодируются буквы выходного алфавита $W = \{w_1, w_2, \dots, w_m\}$ двоичным вектором $Y = \{y_1, y_2, \dots, y_l\}$, где $l =]log_2|W|[$, а состояния $A = \{a_1, a_2, \dots, a_p\}$ двоичным вектором $Q = \{q_1, q_2, \dots, q_r\}$, где $r =]log_2|A|[$.

Другими словами, кодирование – не что иное, как процесс замены букв алфавита Z, W, A абстрактного автомата двоичными векторами. Данный процесс может быть описан таблицами кодирования, которые составляются следующим образом: в левой части таблицы перечисляются все буквы (например, входного алфавита абстрактного автомата), а в правой части – двоичные векторы, которые соотносятся с этими буквами.

Выбор элементов памяти

Количество элементов памяти структурного автомата напрямую соотносится с числом компонентов его состояния. В качестве элементов памяти структурного автомата обычно применяются *D*-триггеры, *T*-триггеры, *RS*-триггеры и *JK*-триггеры. Таблицы переходов триггеров представлены в таблицах 4.11–4.14.

Таблица 4.11 – Переходы *D*-триггера

Состояние *D*-триггера	**Входной сигнал (*D*)**	
	0	**1**
0	0	1
1	0	1

Таблица 4.12 – Переходы *T*-триггера

Состояние *T*-триггера	**Входной сигнал (*T*)**	
	0	**1**
0	0	1
1	1	0

Таблица 4.13 – Переходы *RS*-триггера

Состояние *RS*-триггера	**Входные сигналы (*R, S*)**		
	00	**01**	**10**
0	0	1	0
1	1	1	0

Таблица 4.14 – Переходы *JK*-триггера

Состояние *JK*-триггера	**Входные сигналы (*J, K*)**			
	00	**01**	**10**	**11**
0	0	0	1	1
1	1	0	1	0

Информационные входы триггеров *D*, *T*, *RS* и *JK* являются основными. Таблицы переходов для триггеров формируются только для информационных входов, в то время как остальные входы считаются вспомогательными. Например, вход *R* для *RS*-триггера служит для установки его состояния в нуль, вход *S* – для установки в единицу, а вход *C* – для синхронизации. У каждого триггера два выхода. Единичный сигнал на выходе триггера указывает на состояние «высокого» уровня, а нулевой сигнал – на состояние «низкого» уровня. Для *RS*-триггера комбинация входов *RS* = 11 является запрещенной, т. к. может вызвать неоднозначное поведение триггера.

Пример синтеза структурного автомата Мили

Автомат задан таблицей 4.15. Необходимо синтезировать структурный автомат Мили. В качестве элементов памяти использовать *T*-триггер.

Таблица 4.15 – Автомат Мили

Входные сигналы	**Состояния**			
	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$	$\boldsymbol{a_3}$	$\boldsymbol{a_4}$
$\boldsymbol{z_1}$	a_1/w_2	a_4/w_3	a_1/w_3	a_2/w_1
$\boldsymbol{z_2}$	a_4/w_1	a_3/w_4	a_2/w_1	a_1/w_4
$\boldsymbol{z_3}$	a_4/w_1	a_4/w_3	a_4/w_3	a_2/w_4

Сначала выполним кодирование (первый шаг алгоритма).

На первом этапе закодируем входные сигналы с помощью набора логических переменных x. Множество входных сигналов содержит три различных сигнала, поэтому для их кодирования будет достаточно использовать комбинации из двух переменных: $k =]log_2|Z|[=]log_2 3[= 2$.

На втором этапе закодируем выходные сигналы с помощью набора переменных y. Множество выходных сигналов содержит четыре различных сигнала, поэтому для их кодирования будет достаточно использовать комбинации из двух переменных: $l =]log_2|W|[=]log_2 4[= 2$.

На третьем этапе закодируем выходные сигналы с помощью набора переменных Q. Множество состояний содержит четыре различных элемента, поэтому для их кодирования будет достаточно использовать комбинации из двух переменных: $r =]log_2|A|[=]log_2 4[= 2$ (рисунок 4.8).

	x_1	x_2
z_1	0	1
z_2	1	0
z_3	1	1

	y_1	y_2
w_1	0	1
w_2	1	0
w_3	1	1
w_4	0	0

	Q_1	Q_2
a_1	0	1
a_2	1	0
a_3	1	1
a_4	0	0

Рисунок 4.8 – Кодирование входных, выходных сигналов и сигналов состояний

В результате кодирования таблица исходного автомата Мили примет следующий вид (таблицы 4.16, 4.17).

Таблица 4.16 – Автомат Мили, кодированная таблица выходов

Входы	**Состояния**			
	$\bar{Q}_1 Q_2$	$Q_1 \bar{Q}_2$	$Q_1 Q_2$	$\bar{Q}_1 \bar{Q}_2$
$\bar{x}_1 x_2$	10	11	11	10
$x_1 \bar{x}_2$	01	00	01	00
$x_1 x_2$	01	11	11	00

Таблица 4.17 – Автомат Мили, кодированная таблица переходов состояний

Входы	**Состояния**			
	$\bar{Q}_1 Q_2$	$Q_1 \bar{Q}_2$	$Q_1 Q_2$	$\bar{Q}_1 \bar{Q}_2$
$\bar{x}_1 x_2$	01	00	01	10
$x_1 \bar{x}_2$	00	11	10	01
$x_1 x_2$	00	00	00	10

Двухразрядный код в клетках таблицы перехода формируется следующим образом: первый разряд отображает значение одной переменной, а второй – значение второй переменной. При этом если значение разряда равно единице, то переменная имеет прямое значение, если нулю, то обратное. В первой таблице в качестве переменных используются y_1 и y_2, которые кодируют выходные сигналы, во второй таблице заданы переходы, а в качестве переменных выступают Q_1 и Q_2.

На втором шаге алгоритма осуществим выбор элементов памяти для синтеза автомата Мили. Начальным условием определено, что необходимо использовать *T*-триггеры. Поэтому в качестве элементов памяти будут использоваться два *T*-триггера, формирующие парафазные выходные сигналы Q и $\bar{Q}$, используемые для кодировки состояний.

На третьем шаге алгоритма построим уравнения булевых функций, формирующих выходные сигналы. Для записи будет использоваться СДНФ.

Для всех случаев в кодированной таблице выходов (см. таблицу 4.16), когда y_i имеет значение, равное 1, запишем конъюнкцию кодированных состояний с кодированными входными сигналами. Все такие случаи объединим дизъюнкциями. Например, конъюнкция $\bar{Q}_1 Q_2 \bar{x}_1 x_2$, соответствует случаю, когда текущее состояние автомата имеет код $\bar{Q}_1 Q_2$, а на вход автомата поступает кодированный сигнал $\bar{x}_1 x_2$, при этом на выходе автомата присутствует кодированный сигнал «10», первый разряд которого соответствует переменной y_1 (рисунок 4.9).

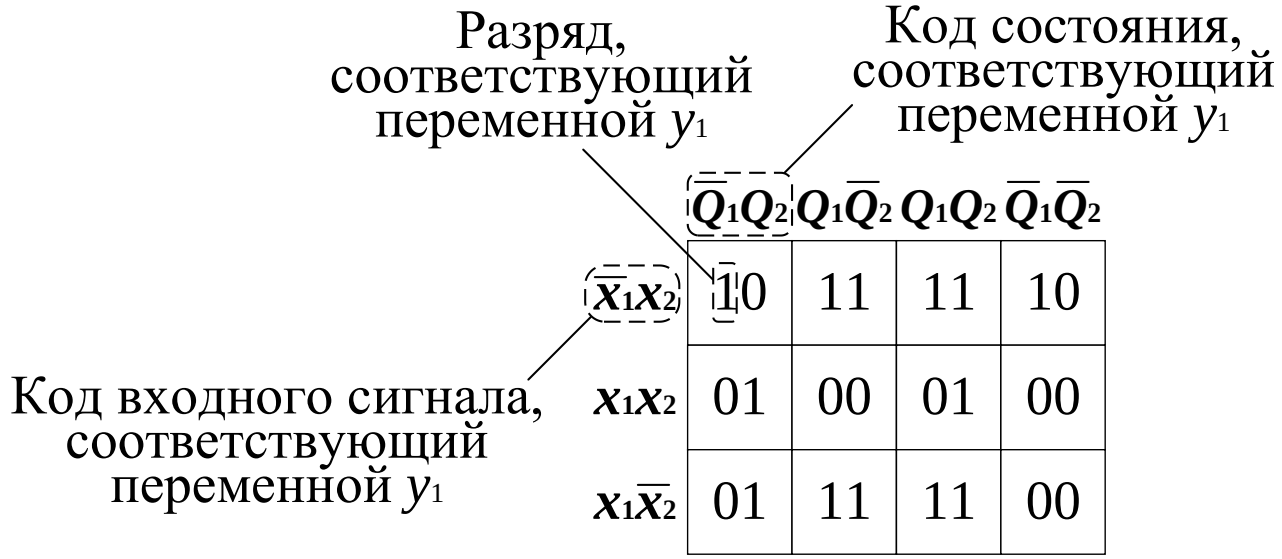


Рисунок 4.9 – Формирование логического выражения для выходного сигнала

Таким образом, функции формирования кода выходного сигнала примут вид

$$y_1 = \bar{Q}_1Q_2\bar{x}_1x_2 + Q_1\bar{Q}_2\bar{x}_1x_2 + Q_1\bar{Q}_2x_1\bar{x}_2 + Q_1Q_2\bar{x}_1x_2 +$$
$$+ \bar{Q}_1\bar{Q}_2\bar{x}_1x_2 + Q_1Q_2x_1\bar{x}_2;$$
$$y_2 = \bar{Q}_1Q_2x_1x_2 + \bar{Q}_1Q_2x_1\bar{x}_2 + Q_1\bar{Q}_2\bar{x}_1x_2 + Q_1\bar{Q}_2x_1\bar{x}_2 +$$
$$+ Q_1Q_2\bar{x}_1x_2 + Q_1Q_2x_1x_2 + Q_1Q_2x_1\bar{x}_2.$$

Далее построим уравнения булевых функций, формирующих сигналы перехода состояний автомата. Для записи будет использоваться СДНФ. Алгоритм записи функции зависит от выбранного типа элемента памяти. В данном случае выбран *T*-триггер, особенностью его работы является то, что по каждому входному сигналу он меняет свое состояние на противоположное (см. таблицу 4.12). Исходя из этого сигнал на вход триггера нужно подавать только в тех случаях, когда состояние автомата отличается от текущего. Таким образом, в функции формирования нового разряда q_i записывается конъюнкция кода текущего разряда и кода входного сигнала только тогда, когда состояние автомата меняется на противоположное. Например, для текущего кода состояния $Q_1\bar{Q}_2$ (рисунок 4.10) и входного сигнала $\bar{x}_1x_2$ значение нового кода разряда q_1 меняется с «1» на «0» (на противоположный), значит, конъюнкцию $Q_1\bar{Q}_2\bar{x}_1x_2$ записываем как конституенту единицы в искомую функцию.

Таким образом, необходимые функции запишутся следующим образом:

$$q_1 = Q_1\bar{Q}_2\bar{x}_1x_2 + Q_1\bar{Q}_2x_1\bar{x}_2 + Q_1Q_2\bar{x}_1x_2 + Q_1Q_2x_1\bar{x}_2 + \bar{Q}_1\bar{Q}_2\bar{x}_1x_2 + \bar{Q}_1\bar{Q}_2x_1\bar{x}_2;$$
$$q_2 = \bar{Q}_1Q_2x_1x_2 + \bar{Q}_1Q_2x_1\bar{x}_2 + Q_1\bar{Q}_2x_1x_2 + Q_1Q_2x_1x_2 + Q_1Q_2x_1\bar{x}_2 + \bar{Q}_1\bar{Q}_2x_1x_2.$$

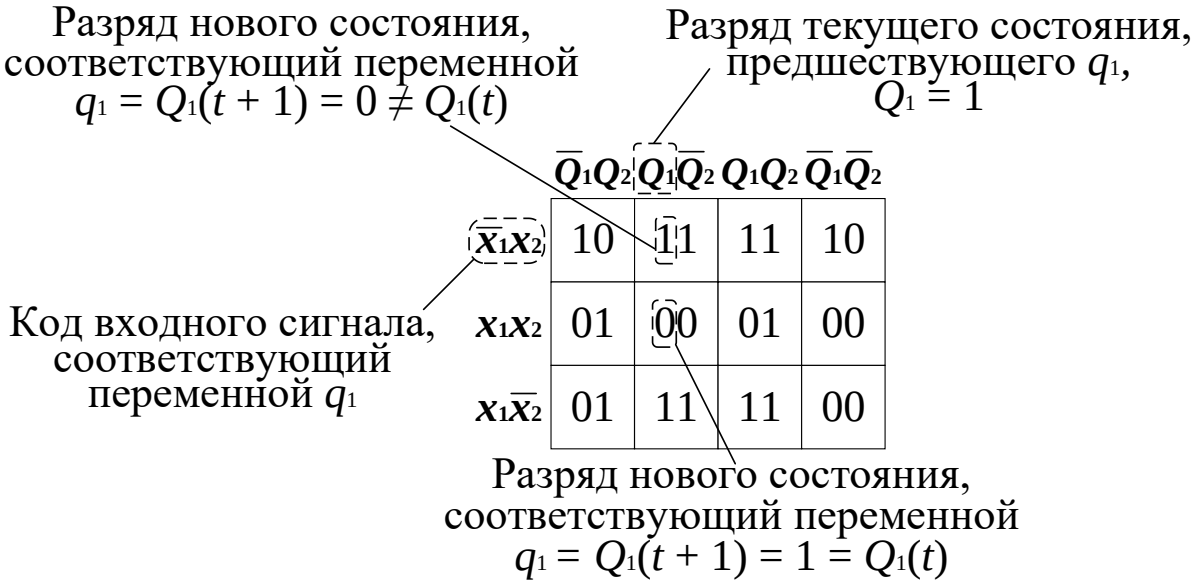


Рисунок 4.10 – Функция формирования кода перехода состояния автомата

Прежде чем перейти к шагу синтеза полученных логических выражений, выполним их минимизацию.

Минимизация функций с помощью метода карт Карно (рисунок 4.11) приведет к следующему результату:

$$y_1 = \bar{x}_1 x_2 + Q_1 x_2;$$
$$y_2 = Q_2 x_1 + Q_1 x_2;$$
$$q_1 = \bar{Q}_2 x_2 + Q_1 x_2;$$
$$q_2 = Q_2 x_1 + x_1 \bar{x}_2.$$

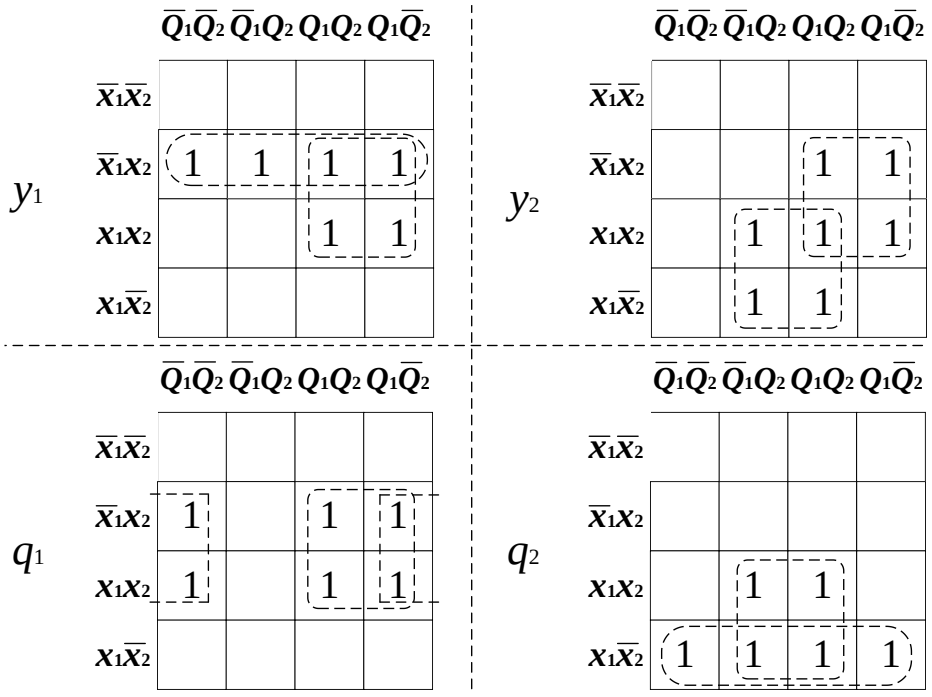


Рисунок 4.11 – Минимизация сигналов переходов и выходных сигналов с использованием карт Карно

Логическая схема, реализующая сформированные логические выражения для выходных сигналов и сигналов управления памятью цифрового автомата, имеет вид, представленный на рисунке 4.12.

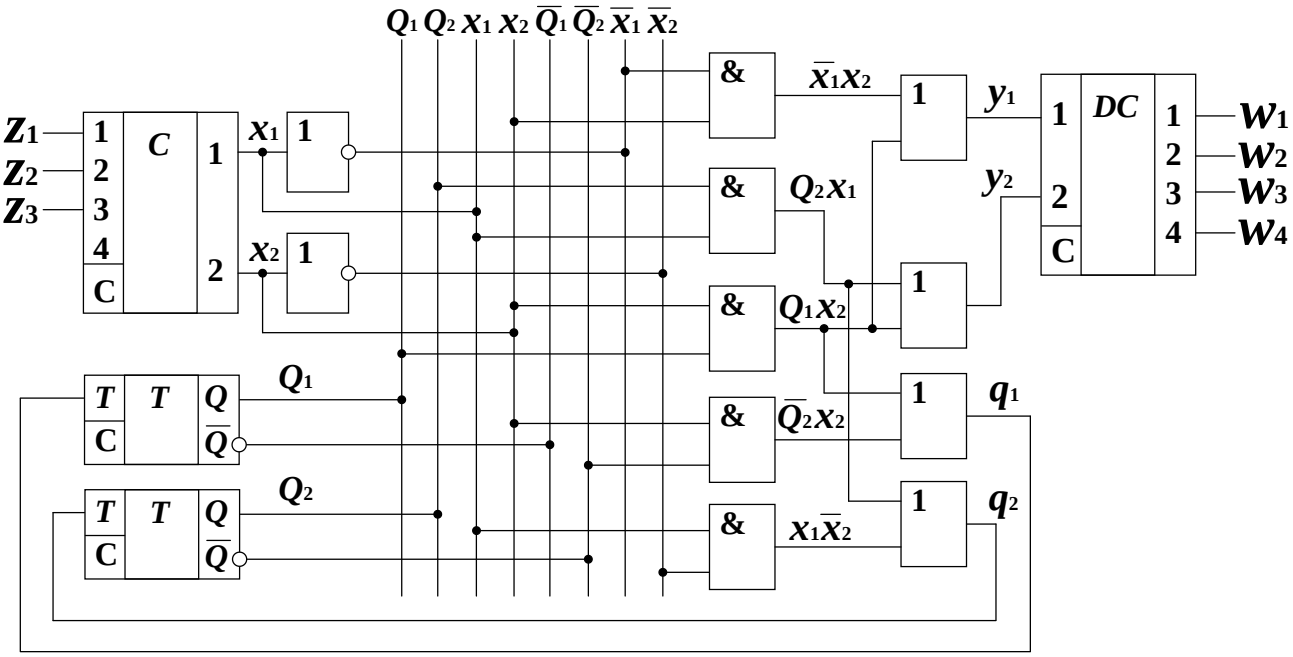


Рисунок 4.12 – Результат синтеза цифрового автомата Мили

4.4 Контрольные вопросы

1 Что такое цифровой автомат?

2 Поясните различия между реальными и абстрактными цифровыми автоматами.

3 Какие объекты входят в математическую модель абстрактного цифрового автомата?

4 Какие бывают абстрактные автоматы по способу формирования функции выходов?

5 Какие бывают абстрактные автоматы по способу задания?

6 Какие бывают способы задания цифровых автоматов?

7 Какие таблицы строятся при матричном способе задания автомата?

8 Как представляется цифровой автомат при графическом способе задания?

9 В чем заключается суть канонического метода синтеза цифрового автомата?

10 Что является исходными данными для канонического метода синтеза цифрового автомата?

11 Опишите алгоритм канонического метода синтеза цифрового автомата.

12 Какие триггеры используются в качестве элементов памяти структурного автомата?

ПРАКТИЧЕСКАЯ ЧАСТЬ

ПРАКТИЧЕСКАЯ РАБОТА № 1. ОСНОВЫ АЛГЕБРЫ ЛОГИКИ

Цели работы

1 Изучить основы алгебры логики.

2 Научиться строить таблицы истинности для логической функции.

3 Научиться приводить выражение к нормальным формам.

4 Изучить логические элементы, реализующие элементарные функции алгебры логики.

Порядок определения индивидуального задания

Для выполнения индивидуального задания описана логическая функция от $k = 4$ переменных $f_1(x_0, x_1, x_2, x_3)$, которая задана выражением вида

$$f_1 = (x_{n1 \bmod k} \cdot x_{n2 \bmod k} + x_{n3 \bmod k} \cdot x_{n4 \bmod k}) + x_{n5 \bmod k} \cdot x_{n6 \bmod k} \times \\ \times x_{n7 \bmod k} \cdot x_{n8 \bmod k} + x_{n9 \bmod k} \cdot x_{n10 \bmod k}, \quad (1.1)$$

где $n1$, $n2$ – цифры дня вашего рождения (в двухзначном представлении);

$n3$, $n4$ – цифры месяца вашего рождения (в двухзначном представлении);

$n5$, $n6$, $n7$, $n8$ – цифры года вашего рождения;

$n9$, $n10$ – две последние цифры вашей группы.

Над цифрами $n1$–$n10$ выполняется операция взятия остатка от целочисленного деления (mod) на $k = 4$ (т. е. если группа № 582571, то $n9 = 7$, $n10 = 1$. Тогда $n9 \bmod k = 7 \bmod 4 = 3$, а $n10 \bmod k = 1 \bmod 4 = 1$).

В том случае, если функция вырождается в константу, при построении над выражением в скобках взять отрицание.

Порядок выполнения индивидуального задания

1 Изучить теоретический материал первого раздела (см. с. 7).

2 Составить таблицу истинности для логической функции f_1.

3 Записать выражения для логической функции f_1 в виде СДНФ и СКНФ.

4 Построить диаграмму Вейча и карту Карно логической функции f_1.

5 Разработать логическую схему полученной функции f_1 с использованием базиса И-ИЛИ-НЕ в нотациях одного из стандартов. При построении логической схемы необходимо использовать форму записи выражения с наименьшим числом операций. Для графического отображения разработанной логической схемы можно использовать также любой графический редактор.

6 По результатам расчетов подготовить отчет, который должен содержать результаты и основные шаги выполнения индивидуального задания, проиллюстрированные таблицами и рисунками.

7 Продемонстрировать и пояснить преподавателю все шаги выполнения индивидуального задания, знать ответы на контрольные вопросы по теме (см. с. 54).

Пример выполнения задания

В качестве исходных данных возьмем студента Джорджа Буля (дата рождения: 02.11.1815), который учится в группе № 001854. Таким образом, исходные значения для определения логической функции индивидуального задания представлены в таблице 1.1.

Таблица 1.1 – Исходные данные для определения логической функции

n_i	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}
Исходные значения	0	2	1	1	1	8	1	5	5	4
Результат взятия остатка (mod 4)	0	2	1	1	1	0	1	1	1	0
Определенные аргументы x_i	x_0	x_2	x_1	x_1	x_1	x_0	x_1	x_1	x_1	x_0

Подставив определенные индексы аргументов в выражение (1.1) и применив аксиомы, получим

$f_1 = (x_0x_2 + x_1x_1) + x_1x_0x_1x_1 + x_1x_0 = (x_0x_2 + x_1) + x_0x_1 = x_0x_2 + x_1.$

Составим таблицу истинности для полученной функции (таблица 1.2).

Таблица 1.2 – Таблица истинности полученной функции

Номер набора	x_0	x_1	x_2	x_3	x_0x_2	x_1	f_1
0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0
2	0	0	1	0	0	0	0
3	0	0	1	1	0	0	0
4	0	1	0	0	0	1	1
5	0	1	0	1	0	1	1
6	0	1	1	0	0	1	1
7	0	1	1	1	0	1	1
8	1	0	0	0	0	0	0
9	1	0	0	1	0	0	0
10	1	0	1	0	1	0	1
11	1	0	1	1	1	0	1
12	1	1	0	0	0	1	1
13	1	1	0	1	0	1	1
14	1	1	1	0	1	1	1
15	1	1	1	1	1	1	1

Запишем выражения для логической функции в виде СДНФ и СКНФ:

$f_{\text{СДНФ}} = \bar{x}_0x_1\bar{x}_2\bar{x}_3 + \bar{x}_0x_1\bar{x}_2x_3 + \bar{x}_0x_1x_2\bar{x}_3 + \bar{x}_0x_1x_2x_3 + x_0\bar{x}_1x_2\bar{x}_3 +$

$+ x_0\bar{x}_1x_2x_3 + x_0x_1\bar{x}_2\bar{x}_3 + x_0x_1\bar{x}_2x_3 + x_0x_1x_2\bar{x}_3 + x_0x_1x_2x_3.$

$f_{\text{СКНФ}} = (x_0 + x_1 + x_2 + x_3)(x_0 + x_1 + x_2 + \bar{x}_3)(x_0 + x_1 + \bar{x}_2 + x_3) \times$

$\times (x_0 + x_1 + \bar{x}_2 + \bar{x}_3)(\bar{x}_0 + x_1 + x_2 + x_3)(\bar{x}_0 + x_1 + x_2 + \bar{x}_3).$

Отобразим логическую функцию на диаграмме Вейча и карте Карно (рисунки 1.1 и 1.2 соответственно).

Найдем минимальную форму логической функции (МДНФ):

$f_{\text{МДНФ}} = x_1 + x_0x_2.$

Построим логическую схему полученной функции с использованием базиса И-ИЛИ-НЕ по стандарту MilSpec 806B и ГОСТ 2.743–91 (рисунки 1.3, 1.4 соответственно).

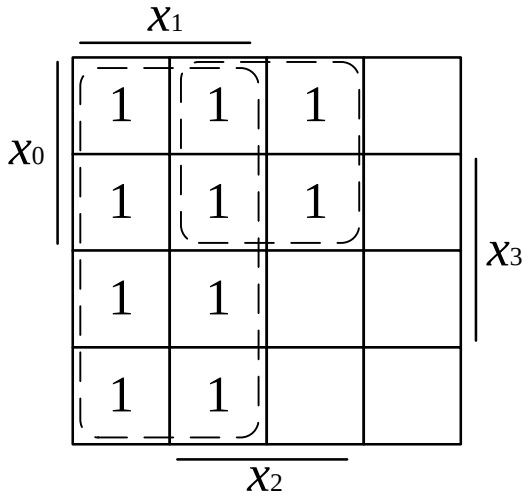


Рисунок 1.1 – Диаграмма Вейча логической функции

x_0x_1 \ x_2x_3	00	01	11	10
00				
01	1	1	1	1
11	1	1	1	1
10			1	1

Рисунок 1.2 – Карта Карно логической функции

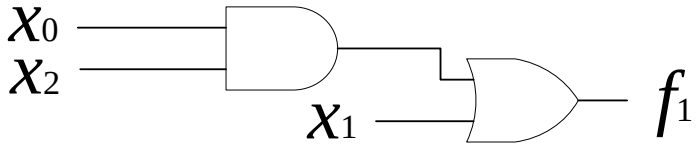


Рисунок 1.3 – Логическая схема полученной логической функции (MilSpec 806B)

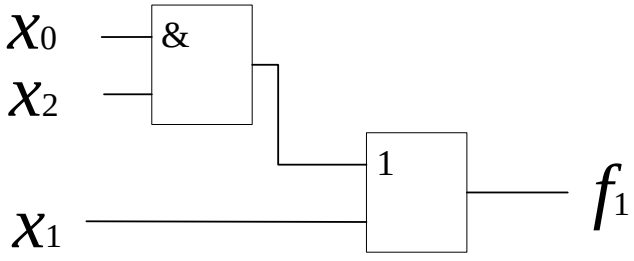


Рисунок 1.4 – Логическая схема полученной логической функции (ГОСТ 2.743–91)

ПРАКТИЧЕСКАЯ РАБОТА № 2. МИНИМИЗАЦИЯ ЛОГИЧЕСКИХ ФУНКЦИЙ МЕТОДОМ КВАЙНА

Цели работы

1 Изучить методы минимизации логических функций.

2 Научиться приводить выражение к нормальным формам.

3 Изучить основные принципы работы метода минимизации Квайна.

Порядок выполнения индивидуального задания:

1 Изучить теоретический материал второго раздела (см. с. 55).

2 Составить карту Карно для функции f_2 индивидуального варианта.

3 Построить минимальную ДНФ (МДНФ) для булевой функции f_2 с помощью метода Квайна с описанием всех шагов метода.

4 Проверить корректность полученной МДНФ путем сопоставления ее с исходными данными на диаграмме Вейча индивидуального варианта задания.

5 По результатам расчетов подготовить отчет, который должен содержать результаты и основные шаги выполнения индивидуального задания, проиллюстрированные таблицами и рисунками.

6 Продемонстрировать и пояснить преподавателю все шаги выполнения индивидуального задания, знать ответы на контрольные вопросы по теме (см. с. 69).

Порядок определения индивидуального задания

Номер варианта индивидуального задания (В) определяется по формуле

В = (Порядковый номер по списку группы) `mod` 31,

где `mod` – операция получения остатка от целочисленного деления.

Индивидуальное задание по вариантам приведено в таблице 2.1 и представляет собой диаграмму Вейча для функции f_2 четырех переменных (*A*, *B*, *C*, *D*).

Таблица 2.1 – Варианты индивидуальных заданий для практической работы № 2

В1

B \ *A*				
	1	1		1
		1	1	1
	1	1	1	
		1		1

C — *D*

В2

B \ *A*				
	1		1	1
	1	1	1	
			1	1
			1	1

C — *D*

В3

B \ *A*				
	1	1	1	1
	1	1	1	
		1		1
	1	1		1

C — *D*

В4

B \ *A*				
		1	1	1
	1		1	
		1	1	1
	1		1	1

C — *D*

В5

B \ *A*				
	1			
	1	1	1	
	1	1		1
	1		1	1

C — *D*

В6

B \ *A*				
	1		1	1
		1	1	
	1	1	1	1
	1		1	1

C — *D*

Продолжение таблицы 2.1

B7

B \ A				
	1	1		1
	1	1		1
	1	1	1	
		1		1

D; C

B8

B \ A				
		1	1	
		1	1	1
	1	1		1
	1			1

D; C

B9

B \ A				
	1		1	1
		1	1	1
	1	1		1
	1	1		1

D; C

B10

B \ A				
	1	1		1
		1	1	1
	1	1	1	
		1	1	

D; C

B11

B \ A				
	1		1	1
	1	1	1	
				1
		1	1	1

D; C

B12

B \ A				
	1	1		1
		1	1	
	1	1	1	1
	1		1	1

D; C

B13

B \ A				
		1	1	1
	1		1	
	1	1	1	1
	1			1

D; C

B14

B \ A				
	1			
	1	1	1	
		1	1	1
	1		1	1

D; C

B15

B \ A				
	1	1	1	1
	1	1	1	
		1		1
	1		1	1

D; C

B16

B \ A				
	1	1		1
		1		1
	1	1	1	
		1	1	1

D; C

B17

B \ A				
		1	1	
			1	1
	1	1	1	1
	1			1

D; C

B18

B \ A				
		1	1	1
		1	1	1
	1		1	1
	1	1		1

D; C

B19

B \ A				
	1	1		1
		1	1	
	1	1	1	
	1	1	1	

D; C

B20

B \ A				
		1		1
			1	1
	1	1	1	
		1	1	1

D; C

B21

B \ A				
	1	1		1
	1	1	1	
	1	1		1
	1		1	1

D; C

Продолжение таблицы 2.1

B22

B \ A		1	1	1
		1	1	
		1		1
	1	1	1	1

C, D

B23

B \ A	1	1	1	1
	1		1	
		1		1
	1			1

C, D

B24

B \ A	1		1	1
	1	1	1	1
		1		1
	1		1	1

C, D

B25

B \ A	1	1		1
	1		1	1
	1	1	1	
			1	1

C, D

B26

B \ A	1	1		1
	1	1		1
	1		1	1
		1		

C, D

B27

B \ A		1	1	1
	1		1	1
	1		1	1
	1	1		1

C, D

B28

B \ A	1	1		1
	1		1	1
	1		1	1
		1	1	

C, D

B29

B \ A		1	1	1
		1	1	1
	1	1	1	1

C, D

B30

B \ A	1	1		1
			1	1
	1	1	1	
	1	1	1	1

C, D

Пример выполнения задания

Логическая функция $f_2(A, B, C, D)$ задана диаграммой Вейча (рисунок 2.1).

B \ A	1	1	1	1
		1		
	1	1		1
	1			1

C, D

Рисунок 2.1 – Диаграмма Вейча заданной логической функции четырех аргументов

Составим карту Карно для заданной функции. Чтобы не допустить ошибок в переносе функции на карту Карно, лучше записать заданную функцию в одной из совершенных форм, например в СДНФ. Получим

$$f_{2\,(\text{СДНФ})}(A,B,C,D) = AB\bar{C}\bar{D} + ABC\bar{D} + \bar{A}BC\bar{D} + \bar{A}B\bar{C}\bar{D} + ABCD +$$
$$+ A\bar{B}\bar{C}D + A\bar{B}CD + \bar{A}\bar{B}\bar{C}D + A\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}\bar{D}.$$

По записанной СДНФ составим карту Карно для исходной функции (рисунок 2.2).

CD / AB	00	01	11	10
00	1	1		
01	1			1
11	1		1	1
10	1	1	1	

Рисунок 2.2 – Карта Карно заданной логической функции четырех аргументов

Минимизируем функцию методом Квайна:

$$f_2 = \overset{1}{A\bar{B}\bar{C}\bar{D}} + \overset{2}{A\bar{B}C\bar{D}} + \overset{3}{\bar{A}\bar{B}C\bar{D}} + \overset{4}{\bar{A}\bar{B}\bar{C}\bar{D}} + \overset{5}{ABCD} +$$

$$+ \overset{6}{A\bar{B}\bar{C}D} + \overset{7}{A\bar{B}CD} + \overset{8}{\bar{A}\bar{B}\bar{C}D} + \overset{9}{A\bar{B}\bar{C}\bar{D}} + \overset{10}{\bar{A}\bar{B}\bar{C}\bar{D}}.$$

Выполним всевозможные операции склеивания. Для удобства пронумеруем конституенты, а результат склеиваний сведем в таблицу 2.2.

Таблица 2.2 – Результаты выполнения операции склеивания

Склеиваемые конституенты	Переменная	Результат склеивания	Номер новой импликанты
1–2	C	$AB\bar{D}$	11
1–4	A	$B\bar{C}\bar{D}$	12
1–9	B	$A\bar{C}\bar{D}$	13
2–3	A	$BC\bar{D}$	14
2–5	D	ABC	15
3–4	C	$\bar{A}B\bar{D}$	16
4–10	B	$\bar{A}\bar{C}\bar{D}$	17
5–7	B	ACD	18
6–7	C	$A\bar{B}D$	19
6–8	A	$\bar{B}\bar{C}D$	20
6–9	D	$A\bar{B}\bar{C}$	21
8–10	D	$\bar{A}\bar{B}\bar{C}$	22
9-10	A	$\bar{B}\bar{C}\bar{D}$	23
11–16	A	$B\bar{D}$	24
12–14	C	$B\bar{D}$	24
12–23	B	$\bar{C}\bar{D}$	25
13–17	A	$\bar{C}\bar{D}$	25
20–23	D	$\bar{B}\bar{C}$	26
21–22	A	$\bar{B}\bar{C}$	26

В результате всех склеиваний и поглощений получаем сокращенную ДНФ:

$f_{2\ (\text{ДНФ сокр.})} = ABC + ACD + A\bar{B}D + B\bar{D} + \bar{C}\bar{D} + \bar{B}\bar{C}.$

Для выявления обязательных импликант и формирования на их основе минимальной ДНФ построим таблицу импликант (таблица 2.3).

Таблица 2.3 – Таблица импликант функции $f_2(A, B, C, D)$

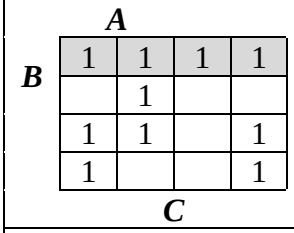
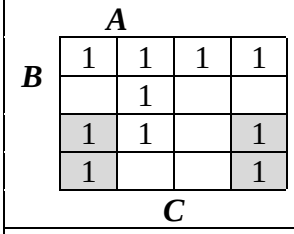
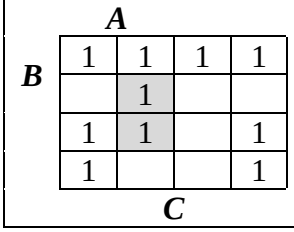
Импликанты	**Конституенты единицы**										**Приоритет**
	$\boldsymbol{AB\bar{C}\bar{D}}$	$\boldsymbol{ABC\bar{D}}$	$\boldsymbol{\bar{A}BC\bar{D}}$	$\boldsymbol{\bar{A}B\bar{C}\bar{D}}$	$\boldsymbol{ABCD}$	$\boldsymbol{A\bar{B}C\bar{D}}$	$\boldsymbol{A\bar{B}CD}$	$\boldsymbol{\bar{A}\bar{B}C\bar{D}}$	$\boldsymbol{A\bar{B}\bar{C}\bar{D}}$	$\boldsymbol{\bar{A}\bar{B}\bar{C}\bar{D}}$	
ABC		*			*						3
ACD					*		*				2
$A\bar{B}D$						*	*				3
$B\bar{D}$	*	*	*	*							1
$\bar{C}\bar{D}$	*			*					*	*	3
$\bar{B}\bar{C}$						*		*	*	*	1

Из таблицы 2.3 следует, что импликанта $B\bar{D}$ является обязательной, только она перекрывает конституенту $\bar{A}BC\bar{D}$, присвоим ей приоритет «1», аналогичным образом импликанта $\bar{B}\bar{C}$ также является обязательной, только она соответствует конституенте $\bar{A}\bar{B}\bar{C}D$ – ставим приоритет «1». Импликанты $B\bar{D}$ и $\bar{B}\bar{C}$ в совокупности перекрывают все конституенты, кроме $ABCD$ и $A\bar{B}CD$, но их перекрывает импликанта ACD, отметим ее приоритетом «2». Найденные импликанты перекрывают все конституенты, поэтому остальные импликанты будут лишними, отметим их приоритетом «3». В результате найдена одна тупиковая ДНФ, которая соответствует минимальной ДНФ и принимает вид

$$f_{2\ (\text{ДНФ сокр.})} = B\bar{D} + \bar{B}\bar{C} + ACD.$$

Проверим корректность полученной МДНФ путем сопоставления ее с исходными данными на диаграмме Вейча (таблица 2.4).

Таблица 2.4 – Сопоставление исходных диаграмм Вейча с полученной МДНФ

Диаграмма Вейча	**Пояснение**
	$B\bar{D}$ соответствует закрашенной области на диаграмме
	$\bar{B}\bar{C}$ соответствует закрашенной области на диаграмме
	ACD соответствует закрашенной области на диаграмме

Таким образом, полученная МДНФ полностью соответствует исходной функции, заданной диаграммой Вейча.

ПРАКТИЧЕСКАЯ РАБОТА № 3. МИНИМИЗАЦИЯ ЛОГИЧЕСКИХ ФУНКЦИЙ МЕТОДОМ КАРТ КАРНО

Цели работы

1 Изучить методы минимизации логических функций.

2 Научиться составлять каноническую форму записи логической функции, заданной в числовой форме.

3 Изучить основные принципы работы метода минимизации с использованием карт Карно.

Порядок выполнения индивидуального задания:

1 Изучить теоретический материал второго раздела (см. с. 55).

2 Представить функцию f_3 индивидуального варианта в СДНФ и СКНФ.

3 Составить карту Карно для СДНФ функции f_3 индивидуального варианта.

4 Выполнить минимизацию функции f_3 с помощью метода карт Карно с описанием всех шагов метода.

5 По результатам расчетов подготовить отчет, который должен содержать результаты и основные шаги выполнения индивидуального задания, проиллюстрированные таблицами и рисунками.

6 Продемонстрировать и пояснить преподавателю все шаги выполнения индивидуального задания, знать ответы на контрольные вопросы по теме (см. с. 69).

Порядок определения индивидуального задания

Номер варианта индивидуального задания (В) определяется по формуле

В = (Порядковый номер по списку группы) mod 16.

Индивидуальное задание по вариантам представлено в таблице 3.1.

Таблица 3.1 – Варианты индивидуальных заданий для практической работы № 3

Номер варианта (В)	**Функция $f_3(x_1, x_2, x_3, x_4)$, заданная на единичных наборах**
1	$f_3 = \{0, 1, 4, 5, 7, 9, 13, 15\}$
2	$f_3 = \{0, 1, 4, 5, 6, 7, 13, 15\}$
3	$f_3 = \{0, 2, 3, 4, 6, 8, 10, 11\}$
4	$f_3 = \{0, 1, 2, 4, 8, 9, 10, 14\}$
5	$f_3 = \{0, 1, 8, 9, 10, 11, 14, 15\}$
6	$f_3 = \{0, 1, 2, 3, 6, 7, 11, 15\}$
7	$f_3 = \{2, 4, 5, 9, 11, 12, 13, 15\}$
8	$f_3 = \{1, 2, 3, 5, 6, 7, 13, 14, 15\}$
9	$f_3 = \{4, 5, 7, 8, 9, 11, 12, 13, 15\}$
10	$f_3 = \{0, 1, 2, 4, 5, 6, 8, 9\}$
11	$f_3 = \{0, 1, 3, 4, 5, 7, 9, 10, 11, 13, 14, 15\}$
12	$f_3 = \{0, 1, 2, 3, 4, 6, 8, 9, 10, 11, 12, 14\}$
13	$f_3 = \{0, 1, 2, 3, 4, 5, 8, 12\}$
14	$f_3 = \{2, 3, 6, 7, 9, 11, 13, 15\}$
15	$f_3 = \{0, 3, 5, 6, 9, 10, 12, 15\}$

Пример выполнения задания представлен в подразделе 2.2 (см. с. 62).

ПРАКТИЧЕСКАЯ РАБОТА № 4. СИНТЕЗ ЦИФРОВЫХ АВТОМАТОВ

Цели работы

1 Изучить основы теории цифровых автоматов.

2 Изучить канонический алгоритм синтеза цифровых автоматов.

3 Научиться строить автоматы Мили и Мура на различных триггерах.

Порядок выполнения индивидуального задания

1 Изучить теоретический материал третьего и четвертого разделов (см. с. 70, 79).

2 Выполнить синтез структурного автомата в соответствии с индивидуальным вариантом. В качестве элементов памяти использовать триггер в соответствии с индивидуальным вариантом.

3 Построить логическую схему цифрового автомата в базисе И-ИЛИ-НЕ.

4 По результатам расчетов подготовить отчет, который должен содержать результаты и основные шаги выполнения индивидуального задания, проиллюстрированные таблицами и рисунками.

5 Продемонстрировать и пояснить преподавателю все шаги выполнения индивидуального задания, знать ответы на контрольные вопросы по теме (см. с. 78, 96).

Порядок определения индивидуального задания

Номер варианта индивидуального задания (В) определяется по формуле

В = (Порядковый номер по списку группы) mod 17.

Индивидуальные задания по вариантам представлены в таблице 4.1.

Таблица 4.1 – Варианты индивидуальных заданий для практической работы № 4

<table>
<tr><th>Номер варианта (В)</th><th>Автомат</th><th>Тип триггера</th><th>Таблица, задающая автомат</th></tr>
<tr><td>1</td><td rowspan="4">Мили</td><td>T</td><td rowspan="4">
<table>
<tr><th>Входные сигналы</th><th colspan="4">Состояния</th></tr>
<tr><th></th><th>a_1</th><th>a_2</th><th>a_3</th><th>a_4</th></tr>
<tr><td>z_1</td><td>w_2/a_1</td><td>w_2/a_1</td><td>w_2/a_1</td><td>w_2/a_2</td></tr>
<tr><td>z_2</td><td>w_1/a_1</td><td>w_3/a_4</td><td>w_1/a_1</td><td>w_2/a_4</td></tr>
<tr><td>z_3</td><td>w_4/a_3</td><td>w_1/a_1</td><td>w_3/a_2</td><td>w_1/a_1</td></tr>
</table>
</td></tr>
<tr><td>2</td><td>D</td></tr>
<tr><td>3</td><td>RS</td></tr>
<tr><td>4</td><td>JK</td></tr>
<tr><td>5</td><td rowspan="4">Мура</td><td>T</td><td rowspan="4">
<table>
<tr><th colspan="2">Выходные сигналы</th><td>w_2</td><td>w_3</td><td>w_1</td><td>w_2</td></tr>
<tr><th colspan="2">Состояния</th><td>a_1</td><td>a_2</td><td>a_3</td><td>a_4</td></tr>
<tr><th rowspan="3">Входные сигналы</th><td>z_1</td><td>a_1</td><td>a_2</td><td>–</td><td>a_2</td></tr>
<tr><td>z_2</td><td>–</td><td>a_4</td><td>a_1</td><td>a_4</td></tr>
<tr><td>z_3</td><td>a_3</td><td>a_1</td><td>a_2</td><td>a_1</td></tr>
</table>
</td></tr>
<tr><td>6</td><td>D</td></tr>
<tr><td>7</td><td>RS</td></tr>
<tr><td>8</td><td>JK</td></tr>
</table>

Продолжение таблицы 4.1

<table>
<tr><th>Номер варианта (В)</th><th>Автомат</th><th>Тип триггера</th><th>Таблица, задающая автомат</th></tr>
<tr><td>9</td><td rowspan="4">Мили</td><td>T</td><td rowspan="4">
<table>
<tr><th>Входные сигналы</th><th colspan="4">Состояния</th></tr>
<tr><td></td><td>a_1</td><td>a_2</td><td>a_3</td><td>a_4</td></tr>
<tr><td>z_1</td><td>w_4/a_2</td><td>w_3/a_2</td><td>—</td><td>—</td></tr>
<tr><td>z_2</td><td>w_1/a_3</td><td>—</td><td>w_3/a_3</td><td>w_3/a_4</td></tr>
<tr><td>z_3</td><td>—</td><td>w_3/a_1</td><td>w_4/a_4</td><td>w_2/a_2</td></tr>
</table>
</td></tr>
<tr><td>10</td><td>D</td></tr>
<tr><td>11</td><td>RS</td></tr>
<tr><td>12</td><td>JK</td></tr>
<tr><td>13</td><td rowspan="4">Мура</td><td>T</td><td rowspan="4">
<table>
<tr><th colspan="2">Выходные сигналы</th><td>w_1</td><td>w_2</td><td>w_3</td></tr>
<tr><th colspan="2">Состояния</th><td>a_1</td><td>a_2</td><td>a_3</td></tr>
<tr><th rowspan="4">Входные сигналы</th><td>z_1</td><td>a_2</td><td>a_2</td><td>–</td></tr>
<tr><td>z_2</td><td>a_1</td><td>a_3</td><td>a_3</td></tr>
<tr><td>z_3</td><td>–</td><td>a_2</td><td>–</td></tr>
<tr><td>z_4</td><td>–</td><td>a_1</td><td>a_2</td></tr>
</table>
</td></tr>
<tr><td>14</td><td>D</td></tr>
<tr><td>15</td><td>RS</td></tr>
<tr><td>16</td><td>JK</td></tr>
</table>

Пример выполнения задания

Синтезировать структурный автомат Мили, заданный таблицей 4.2. В качестве элементов памяти использовать *RS*-триггер. Построить логическую схему в базисе ИЛИ-НЕ.

Таблица 4.2 – Таблица переходов для автомата Мили

Входные сигналы	**Состояния**			
	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$	$\boldsymbol{a_3}$	$\boldsymbol{a_4}$
z_1	w_4/a_2	w_3/a_2	–	–
z_2	w_1/a_3	–	w_3/a_3	w_3/a_4
z_3	–	w_3/a_1	w_4/a_4	w_2/a_2

Выполним кодирование входных, выходных сигналов и сигналов состояний заданного автомата Мили (рисунок 4.1).

	x_1	x_2
z_1	0	1
z_2	1	0
z_3	1	1

	y_1	y_2
w_1	0	1
w_2	1	0
w_3	1	1
w_4	0	0

	Q_1	Q_2
a_1	0	1
a_2	1	0
a_3	1	1
a_4	0	0

Рисунок 4.1 – Кодирование входных, выходных сигналов и сигналов состояний автомата

В результате кодирования таблица исходного автомата Мили примет вид, представленный в таблицах 4.3, 4.4.

Таблица 4.3 – Кодированная таблица выходов автомата Мили

Входы	**Состояния**			
	$\bar{Q}_1 Q_2$	$Q_1 \bar{Q}_2$	$Q_1 Q_2$	$\bar{Q}_1 \bar{Q}_2$
$\bar{x}_1 x_2$	00	11	–	–
$x_1 \bar{x}_2$	01	–	11	11
$x_1 x_2$	–	11	00	10

Таблица 4.4 – Кодированная таблица переходов состояний автомата Мили

Входы	**Состояния**			
	$\bar{Q}_1 Q_2$	$Q_1 \bar{Q}_2$	$Q_1 Q_2$	$\bar{Q}_1 \bar{Q}_2$
$\bar{x}_1 x_2$	10	10	–	–
$x_1 \bar{x}_2$	11	–	11	00
$x_1 x_2$	–	01	00	10

Запишем функцию выходного сигнала:

$y_1 = Q_1 \bar{Q}_2 \bar{x}_1 x_2 + Q_1 \bar{Q}_2 x_1 x_2 + Q_1 Q_2 x_1 \bar{x}_2 + \bar{Q}_1 \bar{Q}_2 x_1 \bar{x}_2 + \bar{Q}_1 \bar{Q}_2 x_1 x_2;$

$y_2 = \bar{Q}_1 Q_2 x_1 \bar{x}_2 + Q_1 \bar{Q}_2 \bar{x}_1 x_2 + Q_1 \bar{Q}_2 x_1 x_2 + Q_1 Q_2 x_1 \bar{x}_2 + \bar{Q}_1 \bar{Q}_2 x_1 \bar{x}_2.$

На конституентах $\bar{Q}_1 Q_2 x_1 x_2$, $Q_1 \bar{Q}_2 x_1 \bar{x}_2$, $Q_1 Q_2 \bar{x}_1 x_2$, $\bar{Q}_1 \bar{Q}_2 \bar{x}_1 x_2$ значения выходного сигнала не определены, поэтому на этапе минимизации можем учитывать их так, как это будет удобно.

На следующем шаге проанализируем таблицу переходов *RS*-триггера (таблица 4.5).

Таблица 4.5 – Переходы *RS*-триггера

Состояние *RS*-триггера	**Входные сигналы (*R*, *S*)**		
	00	**01**	**10**
0	0	1	0
1	1	1	0

Сигналы, необходимые для переключения триггера из одного состояния в другое, представлены в таблице 4.6.

Таблица 4.6 – Переключение состояний *RS*-триггера

Переход состояния		**Сигнал на входах *RS*-триггера**	
из	**в**	***R***	***S***
0	0	–	0
0	1	0	1
1	0	1	0
1	1	0	–

Перепишем таблицу переходов состояний автомата Мили с учетом сигналов *RS*-триггера (таблица 4.7).

Таблица 4.7 – Автомат Мили, кодированная таблица переходов состояний

Входы	**Состояния**			
	$\bar{Q}_1 Q_2$	$Q_1 \bar{Q}_2$	$Q_1 Q_2$	$\bar{Q}_1 \bar{Q}_2$
$\bar{x}_1 x_2$	01 10	0– –0	–	–
$x_1 \bar{x}_2$	01 0–	–	0– 0–	–0 –0
$x_1 x_2$	–	10 01	10 10	01 –0

На основании таблицы 4.7 запишем функции для $R_1 S_1 R_2 S_2$.

$R_1 = Q_1 \bar{Q}_2 x_1 x_2 + Q_1 Q_2 x_1 x_2$.

На конституентах $\bar{Q}_1 Q_2 x_1 x_2, Q_1 \bar{Q}_2 x_1 \bar{x}_2, Q_1 Q_2 \bar{x}_1 x_2, \bar{Q}_1 \bar{Q}_2 \bar{x}_1 x_2, \bar{Q}_1 \bar{Q}_2 x_1 \bar{x}_2$ значение сигнала R_1 не определено.

$S_1 = \bar{Q}_1 Q_2 \bar{x}_1 x_2 + \bar{Q}_1 Q_2 x_1 \bar{x}_2 + \bar{Q}_1 \bar{Q}_2 x_1 x_2$.

На конституентах $\bar{Q}_1 Q_2 x_1 x_2$, $Q_1 \bar{Q}_2 x_1 \bar{x}_2$, $Q_1 Q_2 \bar{x}_1 x_2$, $\bar{Q}_1 \bar{Q}_2 \bar{x}_1 x_2$, $Q_1 \bar{Q}_2 \bar{x}_1 x_2$, $Q_1 Q_2 x_1 \bar{x}_2$ значение сигнала S_1 не определено.

$R_2 = \bar{Q}_1 Q_2 \bar{x}_1 x_2 + Q_1 Q_2 x_1 x_2$.

На конституентах $\bar{Q}_1 Q_2 x_1 x_2$, $Q_1 \bar{Q}_2 x_1 \bar{x}_2$, $Q_1 Q_2 \bar{x}_1 x_2$, $\bar{Q}_1 \bar{Q}_2 \bar{x}_1 x_2$, $Q_1 \bar{Q}_2 \bar{x}_1 x_2$, $\bar{Q}_1 \bar{Q}_2 x_1 \bar{x}_2, \bar{Q}_1 \bar{Q}_2 x_1 x_2$ значение сигнала R_2 не определено.

$S_2 = Q_1 \bar{Q}_2 x_1 x_2$.

На конституентах $\bar{Q}_1 Q_2 x_1 x_2$, $Q_1 \bar{Q}_2 x_1 \bar{x}_2$, $Q_1 Q_2 \bar{x}_1 x_2$, $\bar{Q}_1 \bar{Q}_2 \bar{x}_1 x_2$, $\bar{Q}_1 Q_2 x_1 \bar{x}_2$, $Q_1 Q_2 x_1 \bar{x}_2$ значение сигнала S_2 не определено.

Конституенты, значение сигнала на которых не определено, могут быть приняты за единицу при формировании контура в карте Карно (при выполнении минимизации), т. к. в определенной ситуации (в зависимости от расположения других единиц на карте Карно) это позволит достигнуть лучшего результата минимизации.

Минимизируем полученные функции (рисунки 4.2–4.4).

$x_1 x_2 \backslash Q_1 Q_2$	00	01	11	10
00				
01	*		*	1
11	1	*		1
10			1	*

$$y_1 = \bar{Q}_2 x_2 + Q_1 x_1 \bar{x}_2$$

$x_1 x_2 \backslash Q_1 Q_2$	00	01	11	10
00				
01	*		*	1
11		*		1
10	1	1	1	*

$$y_2 = x_1 \bar{x}_2 + Q_1 \bar{Q}_2 x_2$$

Рисунок 4.2 – Минимизация полученных функций y_1 и y_2

$x_1 x_2 \backslash Q_1 Q_2$	00	01	11	10
00				
01	*		*	
11		*	1	1
10	*			*

$$R_1 = Q_1 x_1 x_2$$

$x_1 x_2 \backslash Q_1 Q_2$	00	01	11	10
00				
01	*	1	*	*
11	1	*		
10		1	*	*

$$S_1 = \bar{Q}_1 x_2 + Q_2 x_1 \bar{x}_2$$

Рисунок 4.3 – Минимизация полученных функций R_1 и S_1

$x_1x_2\backslash Q_1Q_2$	00	01	11	10
00				
01	*	1	*	*
11	*	*	1	
10			*	*

$R_2 = Q_2x_2$

$x_1x_2\backslash Q_1Q_2$	00	01	11	10
00				
01	*		*	
11		*		1
10		*	*	*

$S_2 = Q_1\bar{Q}_2x_1$

Рисунок 4.4 – Минимизация полученных функций R_2 и S_2

Приведем полученные функции к базису ИЛИ-НЕ:

$y_1 = \bar{Q}_2x_2 + Q_1x_1\bar{x}_2 = \overline{\overline{Q_2 + \bar{x}_2} + \overline{\overline{Q_1} + \overline{x_1} + x_2}};$

$y_2 = x_1\bar{x}_2 + Q_1\bar{Q}_2x_2 = \overline{\overline{x_1 + x_2} + \overline{\bar{Q}_1 + Q_2 + \bar{x}_2}};$

$R_1 = Q_1x_1x_2 = \overline{\bar{Q}_1 + \bar{x}_1 + \bar{x}_2};$

$S_1 = \bar{Q}_1x_2 + Q_2x_1\bar{x}_2 = \overline{\overline{Q_1 + \bar{x}_2} + \overline{\bar{Q}_2 + \bar{x}_1 + x_2}};$

$R_2 = Q_2x_2 = \overline{\bar{Q}_2 + \bar{x}_2};$

$S_2 = Q_1\bar{Q}_2x_1 = \overline{\bar{Q}_1 + Q_2 + \bar{x}_1}.$

Построим логическую схему цифрового автомата в базисе ИЛИ-НЕ (рисунок 4.5).

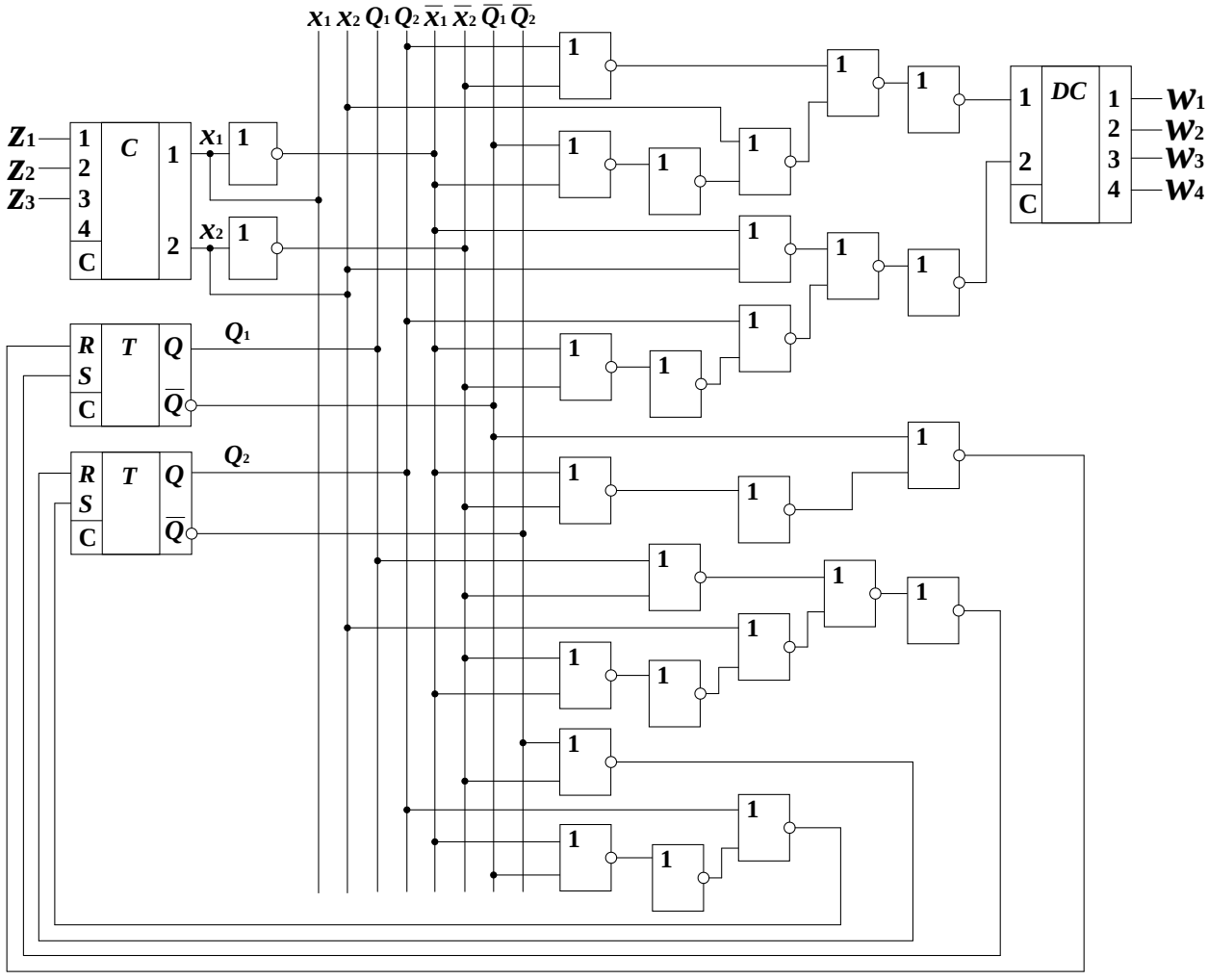


Рисунок 4.5 – Логическая схема цифрового автомата в базисе ИЛИ-НЕ

КОНТРОЛЬНАЯ РАБОТА. ЛОГИЧЕСКИЕ ОСНОВЫ ЦИФРОВЫХ УСТРОЙСТВ

Задание

1 Индивидуальный вариант задания контрольной работы (КР) соответствует порядковому номеру в списке группы. Значение *X* соответствует первой цифре номера группы студента и необходимо для определения варианта задания из практических работ (ПР) № 1–4 по таблице К.1. Например, студент 3-й в списке гр. 582571 ($X = 5$), тогда он выполняет варианты: 24-й (4 + 19) из ПР № 1, 14-й (4 + 9) из ПР № 2, 8-й (13 – 5) из ПР № 3, 7-й (5 + 2) из ПР № 4.

2 Для установленного варианта КР выполнить необходимые расчеты из указанных в таблице К.1 практических работ (см. с. 99, 102, 107, 108) в соответствии с определенными вариантами их заданий; знать ответы на контрольные вопросы (см. с. 54, 69, 78, 96); уметь объяснить каждый шаг процесса решения своего варианта индивидуального задания КР.

Таблица К.1 – Варианты индивидуальных заданий контрольной работы

Вариант задания КР	**Номер варианта задания из практической работы**			
	ПР № 1 (с. 99)	**ПР № 2 (с. 102)**	**ПР № 3 (с. 107)**	**ПР № 4 (с. 108)**
1	$X + 21$	$X + 11$	$15 - X$	$X + 0$
2	$X + 20$	$X + 10$	$14 - X$	$X + 1$
3	$X + 19$	$X + 9$	$13 - X$	$X + 2$
4	$X + 18$	$X + 8$	$12 - X$	$X + 3$
5	$X + 17$	$X + 7$	$11 - X$	$X + 4$
6	$X + 16$	$X + 6$	$10 - X$	$X + 5$
7	$X + 15$	$X + 5$	$X + 0$	$X + 6$
8	$X + 14$	$X + 4$	$X + 1$	$X + 7$
9	$X + 13$	$X + 3$	$X + 2$	$16 - X$
10	$X + 12$	$X + 2$	$X + 3$	$15 - X$
11	$X + 11$	$X + 1$	$X + 4$	$14 - X$
12	$X + 10$	$X + 0$	$X + 5$	$13 - X$
13	$X + 9$	$X + 12$	$X + 6$	$12 - X$
14	$X + 8$	$X + 13$	$10 - X$	$11 - X$
15	$X + 7$	$X + 14$	$11 - X$	$10 - X$
16	$X + 6$	$X + 15$	$12 - X$	$X + 7$
17	$X + 5$	$X + 16$	$13 - X$	$X + 6$
18	$X + 4$	$X + 17$	$14 - X$	$X + 5$
19	$X + 3$	$X + 18$	$15 - X$	$X + 4$
20	$X + 2$	$X + 19$	$X + 6$	$X + 3$
21	$X + 1$	$X + 20$	$X + 5$	$X + 2$
22	$X + 0$	$X + 21$	$X + 4$	$X + 1$
23	$30 - X$	$21 - X$	$X + 3$	$X + 0$
24	$29 - X$	$22 - X$	$X + 2$	$11 - X$
25	$28 - X$	$23 - X$	$X + 1$	$13 - X$
26	$27 - X$	$24 - X$	$X + 0$	$15 - X$
27	$26 - X$	$25 - X$	$15 - X$	$X + 1$
28	$25 - X$	$26 - X$	$13 - X$	$X + 3$
29	$24 - X$	$27 - X$	$11 - X$	$X + 5$
30	$23 - X$	$28 - X$	$X + 1$	$X + 7$
31	$22 - X$	$29 - X$	$X + 3$	$X + 0$
32	$21 - X$	$30 - X$	$X + 5$	$16 - X$

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И РЕКОМЕНДОВАННОЙ ЛИТЕРАТУРЫ

1 Савенко, А. Г. Основы компьютерной техники. Практические занятия : учеб.-метод. пособие / А. Г. Савенко, А. В. Матвеев. – Минск : БГУИР, 2020. – 82 с.

2 Савельев, А. Я. Прикладная теория цифровых автоматов : учеб. для вузов по спец. ЭВМ / А. Я. Савельев. – М. : Высш. шк., 1987. – 272 с.

3 Савельев, А. Я. Основы информатики : учеб. для вузов / А. Я. Савельев. – М. : МГТУ им. Н. Э. Баумана, 2001. – 328 с.

4 Лысиков, Б. Г. Цифровая и вычислительная техника : учеб. / Б. Г. Лысиков. – Минск : Экоперспектива, 2002. – 264 с.

5 Лысиков, Б. Г. Арифметические и логические основы цифровых автоматов : учеб. для вузов по специальности ЭВМ / Б. Г. Лысиков. – 2-е изд. – Минск : Выш. шк., 1980. – 336 с.

6 Ожиганов, А. А. Теория автоматов : учеб. пособие / А. А. Ожиганов. – СПб. : НИУ ИТМО, 2013. – 84 с.

7 Карпов, Ю. Г. Теория автоматов / Ю. Г. Карпов. – СПб. : Питер, 2003. – 208 с.

8 Луцик, Ю. А. Арифметические и логические основы вычислительной техники : учеб. пособие / Ю. А. Луцик, И. В. Лукьянова. – Минск : БГУИР, 2014. – 174 с.

9 Степанов, А. Н. Курс информатики для студентов информационно-математических специальностей / А. Н. Степанов. – СПб. : Питер, 2018. – 1088 с.

10 Искра, Н. А. Арифметические и логические основы вычислительной техники : пособие / Н. А. Искра, И. В. Лукьянова, Ю. А. Луцик. – Минск : БГУИР, 2016. – 75 с.

11 Куприянова, Д. В. Арифметические и логические основы вычислительной техники : пособие / Д. В. Куприянова, И. В. Лукьянова, Ю. А. Луцик. – Минск : БГУИР, 2021. – 72 с.

12 Прикладная теория цифровых автоматов / К. Г. Самофалов, А. М. Ромлинкевич, В. Н. Валуйский [и др.]. – Киев : Вища школа, 1987. – 375 с.

13 Сергеев, Н. П. Основы вычислительной техники : учеб. пособие для вузов / Н. П. Сергеев, Н. П. Вашкевич. – 2-е изд., перераб. и доп. – М. : Высш. шк., 1988. – 311 с.

14 Фомин, Д. В. Основы компьютерной электроники : учеб. пособие / Д. В. Фомин. – М. ; Берлин : ДиректМедиа, 2014. – 108 с.

Св. план 2024, поз. 38

Учебное издание

Савенко Андрей Геннадьевич
Матвеев Андрей Владимирович

ЛОГИЧЕСКИЕ И СТРУКТУРНЫЕ ОСНОВЫ ЦИФРОВЫХ УСТРОЙСТВ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редактор *А. Ю. Шурко*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *Е. Г. Бабичева*

Подписано в печать 05.09.2025. Формат 60×84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 6,86. Уч.-изд. л. 6,2. Тираж 40 экз. Заказ 204.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
Ул. П. Бровки, 6, 220013, г. Минск