ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)

Кафедра вычислительных машин, комплексов, систем и сетей

Д.А. Затучный

ОСНОВЫ ТЕОРИИ АВТОМАТОВ

Учебное пособие

Утверждено редакционно-издательским советом МГТУ ГА в качестве учебного пособия

Москва
ИД Академии Жуковского
2020

УДК 519.713
ББК 6Ф6.5
З-37

Печатается по решению редакционно-издательского совета
Московского государственного технического университета ГА

Рецензенты:
Надейкина Л.А. (МГТУ ГА) – канд. физ.-мат. наук, доцент;
Акиншин Р.Н. (Секция прикладных проблем при Президиуме РАН) – д-р техн. наук, доцент

Затучный Д.А.

З-37 Основы теории автоматов [Текст] : учебное пособие / Д.А. Затучный. – М. : ИД Академии Жуковского, 2020. – 76 с.

ISBN 978-5-907275-15-7

Учебное пособие издается в соответствии с рабочей программой учебной дисциплины «Теория автоматов» по учебному плану для студентов I и II курсов направления 09.03.01 очной формы обучения.

Данное учебное пособие позволит студентам изучить разделы, связанные с элементами теории алгоритмов, основами алгебры логики, а также методами анализа и синтеза комбинационных схем.

При изучении данного учебного пособия в рамках учебного плана направления подготовки 09.03.01 «Информатика и вычислительная техника» студенты получат теоретические знания, иллюстрированные большим числом практических примеров, по реализации логических функций, а также анализу комбинационных схем. При этом подробно рассмотрены методы минимизации логических функций.

Рассмотрено и одобрено на заседаниях кафедры 19.06.2019 г. и методического совета 19.06.2019 г.

УДК 519.713
ББК 6Ф6.5
Св. тем. план 2020 г.
поз. 26

ЗАТУЧНЫЙ Дмитрий Александрович
ОСНОВЫ ТЕОРИИ АВТОМАТОВ

Учебное пособие

В авторской редакции

Подписано в печать 02.07.2020 г.
Формат 60x84/16 Печ. л. 4,75 Усл. печ. л. 4,42
Заказ № 572/0225-УП01 Тираж 30 экз.

Московский государственный технический университет ГА
125993, Москва, Кронштадтский бульвар, д. 20

Издательский дом Академии имени Н. Е. Жуковского
125167, Москва, 8-го Марта 4-я ул., д. 6А
Тел.: (495) 973-45-68 E-mail: zakaz@itsbook.ru

ISBN 978-5-907275-15-7

Введение

Теория автоматов лежит в основе всех цифровых технологий и программного обеспечения, так, например, компьютер является частным случаем практической реализации конечного автомата. Её изучение является ключевым звеном в учебном процессе направления подготовки 09.03.01, так как является основой для дальнейшего изучения такой дисциплины как Информатика.

Настоящее учебное пособие «Основы теории автоматов» по курсу «Теория автоматов» позволит студентам направления подготовки 090301 изучить разделы, связанные с элементами теории алгоритмов, основами алгебры логики, а также методами анализа и синтеза комбинационных схем.

Развитие алгоритмического мышления является главным залогом в подготовке специалистов направления подготовки 090301. Студенты при изучении данного учебного пособия, предназначенного для изучения дисциплины «Основы теории автоматов» в рамках учебного плана направления подготовки 090301 «Информатика и вычислительная техника» получают теоретические знания, иллюстрированные большим числом практических примеров, по реализации логических функций, а также анализу комбинационных схем. При этом подробно рассмотрены методы минимизации логических функций.

1. Основные понятия теории алгоритмов

1.1. Алгоритм и его свойства

Алгоритмом называется точная инструкция исполнителю в понятной для него форме, определяющая процесс достижения поставленной цели на основе имеющихся исходных данных за конечное число шагов.

Основными свойствами алгоритмов являются:

1. Универсальность (массовость) - применимость алгоритма к различным наборам исходных данных.

2. Дискретность - процесс решения задачи по алгоритму разбит на отдельные действия.

3. Однозначность - правила и порядок выполнения действий алгоритма имеют единственное толкование.

4. Конечность - каждое из действий и весь алгоритм в целом обязательно завершаются.

5. Результативность - по завершении выполнения алгоритма обязательно получается конечный результат.

6. Выполнимость - результат алгоритма достигается за конечное число шагов.

Алгоритм считается правильным, если его выполнение дает правильный результат. Соответственно алгоритм содержит ошибки, если можно указать такие допустимые исходные данные или условия, при которых выполнение алгоритма либо не завершится вообще, либо не будет получено никаких результатов, либо полученные результаты окажутся неправильными.

Выделяют три крупных класса алгоритмов:

- вычислительные алгоритмы, работающие со сравнительно простыми видами данных, такими как числа и матрицы, хотя сам процесс вычисления может быть долгим и сложным;

- информационные алгоритмы, представляющие собой набор сравнительно простых процедур, работающих с большими объемами информации (алгоритмы баз данных);

- управляющие алгоритмы, генерирующие различные управляющие воздействия на основе данных, полученных от внешних процессов, которыми алгоритмы управляют.

1.2. Способы задания алгоритмов

Для описания алгоритма могут быть использованы различные средства.

В зависимости от типа преимущественно используемых средств различают следующие способы описания:

1) словесное описание;
2) табличное описание;
3) графическое описание;
4) аналитическое описание;
5) описание с помощью специальных средств.

Словесное описание представляет собой описание алгоритма на естественном языке и обычно используется для обмена алгоритмами между различными пользователями, особенно на начальной стадии разработки алгоритма. Словесное описание может быть использовано для описания любых алгоритмов, однако оно имеет ряд недостатков, основными из которых являются громоздкость, неоднозначность и отсутствие наглядности. Отметим, что элементы словесного описания используются во всех других способах описания алгоритмов.

Основу табличного описания алгоритмов составляют таблицы различного рода: таблицы работы, таблицы переходов и выходов и т.д. Табличное описание является компактным и довольно наглядным, особенно при описании алгоритмов работы цифровых автоматов с памятью. Однако такое описание не всегда удобно.

При графическом описании могут быть использованы направленные графы, граф-схемы, временные диаграммы и т.д. Главной особенностью такого описания является наглядность. Однако для сложных алгоритмов наглядность может быть недостаточной, особенно при расположении графического описания на нескольких листах. Для обеспечения наглядности в этом случае алгоритм рассматривается на нескольких уровнях детализации.

При аналитическом описании используются математические формулы, логические функции и т.д. Особенно удобен этот способ для описания вычислительных и логических алгоритмов.

Все рассмотренные выше способы описания алгоритмов используют такие средства, которые применяют и в других случаях. В настоящее время для описания алгоритмов часто используют такие средства, которые предназначены специально для этой цели. К таким средствам относятся:

1) алгоритмические языки (языки программирования);
2) машина Тьюринга;
3) рекурсивные функции;
4) нормальные алгоритмы Маркова;
5) операторы Ляпунова и т.д.

Из этих специальных средств машина Тьюринга является объектом, который может использоваться не только для описания алгоритмов, но и как пример цифрового автомата с памятью. Поэтому принцип построения и работы машины Тьюринга далее будет рассмотрен подробно.

1.3. Машина Тьюринга

Машина Тьюринга представляет собой устройство для выполнения алгоритмов преобразования информации. В теории алгоритмов машина Тьюринга используется как средство для описания алгоритмов. Считается, что если алгоритм решения некоторой задачи существует, то этот алгоритм может быть реализован на машине Тьюринга, т.е. для этого алгоритма может быть построена машина Тьюринга. С точки зрения теории цифровых

автоматов машина Тьюринга представляет собой универсальный преобразователь информации.

В состав машины Тьюринга входят:

1) лента, на которой записаны исходные данные и на которую записываются результаты решения задачи;
2) головка записи/чтения информации;
3) блок управления.

Лента состоит из одельных ячеек. В каждую ячейку может быть записан символ из некоторого алфавита. На ленту предварительно записывается исходная информация. В процессе работы машины Тьюринга с ленты с помощью головки считывается символ, находящийся над головкой (текущий символ X_i). При считывании информация в ячейке стирается. После считывания символа X_i в результате работы машины на данном шаге на ленту вместо X_i записывается новый символ Y_i. Лента считается бесконечной в одну или обе стороны и является внешней памятью машины. Головка служит для чтения и записи информации и с помощью привода может перемещаться вдоль ленты вправо или влево на одну ячейку на каждом шаге. В каждый момент времени для записи или чтения доступна только одна ячейка ленты.

Блок управления организует работу машины в целом. Он анализирует считываемую информацию и управляет записью символов Y_i на ленту, а также перемещением головки. Блок управления имеет внутреннюю память. Информация во внутренней памяти представляет собой состояние машины Тьюринга. Реакция машины на считанный символ X_i зависит не только от значения этого символа, но и от состояния машины, которое на каждом шаге её работы может изменяться. Новое состояние машины определяется значением символа X_i и старым состоянием машины.

Перед началом работы на ленту наносится исходная информация. Головка устанавливается под ячейкой ленты, в которой записан первый символ. Машина переводится в начальное состояние.

Процесс преобразования информации в машине Тьюринга состоит из отдельных шагов. На каждом шаге машина выполняет следующие элементарные операции:

1) чтение символа X_i из ячейки, под которой размещена головка;
2) анализ считанного символа в соответствии с алгоритмом решения задачи;
3) запись в ячейку вместо символа X_i нового символа Y_i, который может совпадать с X_i;
4) перемещение головки на одну ячейку влево или вправо;
5) переход машины в новое состояние (запись новой информации во внутреннюю память).

На каждом шаге работы значение символа Y_i, направление перемещения головки и новое состояние машины зависят от значения

символа X_i и текущего состояния машины. Поэтому процесс работы машины Тьюринга может быть описан в виде совокупности команд, каждая из которых имеет следующий вид:

$X_i Q_i \rightarrow Y_i Y_d Q_i$,

где X_i - символ, считанный с ленты;

Q_i - текущее состояние машины;

Y_i - символ, записываемый на ленту;

$Y_d = (П, Л, Ст)$ - сигнал управления движением головки (П – сигнал движения головки вправо; Л – сигнал движения головки влево; Ст – сигнал остановки); Q_j - новое состояние машины.

Алгоритм работы машины Тьюринга может быть описан с помощью ориентированного графа. При этом вершины графа соответствуют состояниям машины, а дуги указывают переходы из одного состояния в другое. На дугах графа отмечаются входные символы X_i и соответствующие им сигналы Y_i и Y_d.

Рассмотрим машину Тьюринга, которая обрабатывает информацию на ленте, составленную из букв А и В.

Массив обрабатываемой информации ограничен слева и справа распределителями. Перед началом работы головка устанавливается справа от левого символа, а машина переводится в начальное состояние.

Суть обработки заключается в том, что машина просматривает информацию на ленте, последовательно перемещая головку вправо, и при обнаружении пары символов АВ заменяет её на пару ВА. Решение задачи заканчивается, если на ленте не останется ни одной комбинации символов вида АВ.

Особенность задачи заключается в том, что в результате такой подстановки на ленте могут появляться новые комбинации, которых ранее на ленте не было. Поэтому принят алгоритм, при котором после каждой подстановки головка возвращается в исходное положение и просмотр ленты повторяется. В этом случае признаком отсутствия на ленте комбинаций типа АВ является достижение головки при её движении вправо ячейки, в которой записан символ. Тогда головка перемещается влево в исходное положение, и машина останавливается, переходя в конечное состояние Q_z.

Каждое состояние имеет свои особенности. Переход машины из одного состояния в другое происходит в случае считывания определённой информации и этот факт необходимо запомнить. В данном случае состояние можно определить следующим образом:

1) состояние Q_1 - исходное состояние. В этом состоянии машина перемещает головку вправо до обнаружения символа А;

2) состояние Q_2 - состояние, в которое машина переходит, если при движении головки вправо последний считанный символ был символом А. В состоянии Q_2 машина «ждёт» символ В;

3) состояние Q_3 - состояние, в которое машина переходит при обнаружении комбинации вида AB. В этом состоянии символ B заменяется на символ A и головка смещается влево на один шаг;

4) состояние Q_4 - состояние, в которое машина переходит после замены всей комбинации AB на комбинацию BA. В этом состоянии машины головка перемещается влево в исходное положение;

5) состояние Q_5 - состояние, в которое машина переходит, если при движении головки вправо из исходного положения на ленте не обнаружено ни одной комбинации вида AB. В этом состоянии машины головка перемещается влево в исходное положение;

6) состояние Q_z - конечное состояние, в которое машина переходит, если закончена обработка информации на ленте и головка установлена в исходное положение.

Алгоритм работы машины Тьюринга может быть описан также в табличной форме. Для этой цели используется таблица переходов и выходов машины. В этой таблице строки соответствуют состояниям Q_i, а столбцы – входным сигналам X_i. На пересечении строки Q_i и столбца X_i записываются выходные символы Y_i, d и Q_i.

Таблица переходов и выходов машины имеет вид таблицы 1.1.

Таблица 1.1

Таблица переходов и выходов машины Тьюринга

МТ Состояния Q_i	Входные символы X_i		
	A	B	*
Q_1	A, П, Q_2	B, П, Q_1	*, Л, Q_5
Q_2	A, П, Q_2	A, Л, Q_3	*, Л, Q_5
Q_3	B, Л, Q_4	_____	_____
Q_4	A, Л, Q_4	B, Л, Q_4	*, П, Q_1
Q_5	A, Л, Q_5	B, Л, Q_5	*, П, Q_z
Q_z	A, Ст, Q_z	B, Ст, Q_z	*, Ст, Q_z

Рассматриваемая машина Тьюринга работает следующим образом. Из исходного положения головка перемещается вправо до обнаружения пары символов AB. При появлении такой комбинации головка перемещается влево и производится последовательная замена символа B на символ A, затем символа A на символ B. После замены головка перемещается влево до символа *, затем делает шаг вправо и устанавливается в исходное положение.

Далее начинается новый цикл поиска комбинации AB при движении головки вправо. Работа машины Тьюринга продолжается до тех пор, пока на ленте не останется ни одной пары символов AB. Признаком этого является то, что головка при движении вправо доходит до символа *. Тогда машина

переходит в состояние Q_5, в котором организуется возвращение головки в исходное положение. Затем машина переходит в состояние Q_z, головка останавливается, и работа машины заканчивается.

1.4. Тезис Тьюринга

Тезис Чёрча-Тьюринга - это гипотеза, постулирующая эквивалентность между интуитивным понятием алгоритмической вычислимости и строго формализованными понятиями частично рекурсивной функции и функции, вычислимой на машине Тьюринга. В связи с интуитивностью исходного понятия алгоритмической вычислимости, данный тезис носит характер суждения об этом понятии и его невозможно строго доказать или опровергнуть. Перед точным определением вычислимой функции математики часто использовали неофициальный термин, «эффективно вычислимый» для описания функций, которые можно вычислить с помощью бумажно-карандашных методов.

Тезис был высказан Алонзо Чёрчем и Аланом Тьюрингом в середине 1930-х годов.

В терминах вычислимости по Тьюрингу тезис гласит, что для любой алгоритмически вычислимой функции существует вычисляющая её значения машина Тьюринга.

Позднее были сформулированы другие практические варианты утверждения:

1) физический тезис Чёрча— Тьюринга: любая функция, которая может быть вычислена физическим устройством, может быть вычислена машиной Тьюринга;

2) сильный тезис Чёрча-Тьюринга: любой конечный физический процесс, не использующий аппарат, связанный с непрерывностью и бесконечностью, может быть вычислен физическим устройством.

1.5. Композиция машин Тьюринга

Сущность композиции машин аналогична известному в математике принципу суперпозиции. Смысл суперпозиции заключается в том, что в качестве аргумента одной функции используется другая функция. Применительно к машинам Тьюринга композиция означает, что результаты решения задачи на одной машине являются исходными данными для другой машины Тьюринга. Общая схема объединения двух машин в композицию показана на рис. 1.1.

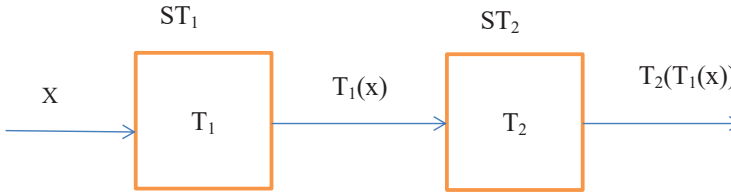


Рис. 1.1. Объединение двух машин Тьюринга в композицию

На рис. 1.1. приняты следующие обозначения:
T_1, T_2 – машины Тьюринга;
ST_1, ST_2 – системы команд машин T_1 и T_2 соответственно;
X – исходная информация для машины T_1;
T_1 (x) – результат работы машины T_1;
$T_2(T_1(x))$ – результат работы машины T_2.

Составив алгоритмы работы машин Тьюринга для решения набора определённых операций (например, арифметических операций), можно составлять композиции машин Тьюринга для решения более сложных задач. При этом разработка общего алгоритма сводится к его составлению из тех операций, для которых уже известны алгоритмы выполнения на машине Тьюринга. Такой подход аналогичен использованию процедур и функций в программировании.

Однако использование композиции машин предполагает, что известны алгоритмы выполнения элементарных операций, из которых составляется общий алгоритм. Поэтому при решении произвольных задач такой подход не исключает необходимости составления новых алгоритмов и соответственно разработки машин Тьюринга с различными блоками управления. Реализовать любой алгоритм без изменения структуры блока управления возможно при использовании универсальной машины Тьюринга.

1.6. Универсальная машина Тьюринга

Если задана некоторая машина Тьюринга (т.е. известны алфавиты входных, выходных сигналов и состояний, исходное положение головки и исходное состояние машины, а также таблица работы машины и лента с исходной информацией), то можно по шагам вручную выполнить все преобразования информации, для которых предназначена эта машина. Фактически в этом случае выполняется моделирование машины Тьюринга.

При анализе операций, которые выполняются при моделировании машины Тьюринга, можно установить, что моделирование сводится к повторению на каждом шаге следующих действий:

1) чтение символа из ячейки ленты, над которой находится головка;
2) поиск команды в таблице работы машины. Поиск выполняется по текущему состоянию машины и значению считанного символа.

3) выбор из команды символа, который должен быть записан на ленту,

и его запись.

4) выбор из команды символа перемещения головки и её перемещение.

5) выбор из команды нового состояния машины и смена текущего состояния на новое. Далее следует переход к следующему шагу и повторение этих действий.

Характер этих элементарных действий таков, что все они могут быть выполнены при помощи некоторой другой машины Тьюринга, которая будет моделировать работу исходной машины. Сущность моделирования поясняется на рис. 1.2.

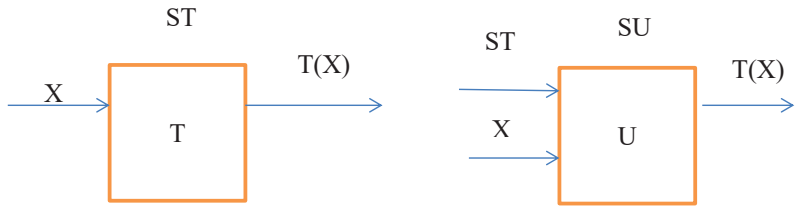


Рис. 1.2. Сущность моделирования исходной машины Тьюринга

Если машина T имеет систему команд ST и обрабатывает ленту с информацией X, то её работа может быть смоделирована другой машиной U, которая имеет собственную систему команд SU. Для моделирования на вход машины U нужно подать не только ленту с информацией X, но и систему команд (таблицу работы) ST. Эта система команд может быть записана на той же ленте, что и исходная информация.

Важной особенностью моделирующей машины является то, что её система команд SU (и соответственно структура блока управления) не зависит от алгоритма работы моделируемой машины T.

Машина Тьюринга, которая может моделировать работу любой другой машины Тьюринга, называется универсальной.

2. Типы автоматов и способы их задания

2.1. Элементарные логические функции

Одним из основных понятий в математической логике является понятие высказывания. Под высказыванием понимается всякое предложение, относительно которого имеет смысл утверждать о его истинности или ложности. Например, 8 - четное число, 2 больше 10, сегодня воскресенье, и т.д.

Из простейших высказываний путем соединения их с помощью логических связок можно составлять новые, более сложные высказывания.

Истинность или ложность новых высказываний определяется истинностью или ложностью составляющих высказываний и тем, какими логическими связками они соединены.

Абстрагируясь от конкретного содержания высказывания можно рассматривать его как некоторую величину, которая может иметь какое-либо одно значение из двух возможных значений: истина или ложь. При этом сложное высказывание, составленное из простых высказываний, в математической логике рассматривается как некоторая функция от высказываний-аргументов. Множество значений, которые может принимать функция, тоже состоит из двух элементов: истина и ложь.

В дальнейшем высказывания будем обозначать прописными буквами латинского алфавита. Значение истинности обозначают через 1, а значение лжи через 0.

Логика, которая оперирует лишь с объектами (высказывания, функции), принимающими одно из двух возможных значений, называется двузначной или булевой. Простые высказывания называются логическими или булевыми переменными, а их функции - логическими или булевыми функциями, либо функциями алгебры логики (ФАЛ).

Функции одной и двух логических переменных называются элементарными.

Логические функции одной переменной.

Для одной переменной число функций равно 4. Все эти функции F_1, F_2, F_3, F_4 представлены в таблице 2.1, которая называется таблицей истинности логических функций. Значение функций F_1 и F_2 не зависят от значения аргумента X . Это константы F_1=1 и F_2=0. Значение функции F_3 повторяют значения аргумента X, F_3 = X. Наконец, значения функции F_4 противоположны значениям аргумента.

Функции F_1, F_2, F_3 являются тривиальными и практического интереса не представляют. Функция F_4 называется отрицанием или инверсией и обозначается $\overline{x}$.

Таблица 2.1

x	F_1	F_2	F_3	F_4
0	0	0	1	1
1	0	1	0	1

Логические функции двух переменных.

Для двух переменных число всех возможных функций равно $2^4 = 16$. Полный набор функций двух переменных, их обозначения и названия приведены в таблице 2.2. Из таблицы видно, что восемь функций могут быть получены из других восьми путем применения операции отрицания.

Таблица 2.2.

Значения логических функций

Значения переменных		Значения логических функций															
x_1	x_2	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}	F_{16}
0	0	0	0	0	0	0	0	0	0	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
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Наиболее часто используемые логические функции имеют специальные названия.

Функция $F_1(x_1, x_2)$ называется генератором 0 ($F_1(x_1, x_2)=0$).

Функция $F_{16}(x_1, x_2)$ называется генератором 1 ($F_{16}(x_1, x_2)=1$).

Функция $F_2(x_1, x_2)$ называется конъюнкцией x_1 и x_2 и обозначается как $x_1 \wedge x_2$ или $x_1 \cdot x_2$. Конъюнкция x_1 и x_2 равна 1, если равны 1 соответственно x_1 и x_2, поэтому её часто называют функцией И. Также её называют логическим умножением, так как её таблица совпадает с таблицей умножения для чисел 0 и 1.

Функцию $F_3(x_1, x_2)$ называют левой коимпликацией и обозначают как $\overline{x_1 \to x_2}$. $F_3(x_1, x_2) = x_1 \wedge \overline{x_2}$.

Функцию $F_5(x_1, x_2)$ называют правой коимпликацией и обозначают как $\overline{x_1 \leftarrow x_2}$. $F_5(x_1, x_2) = x_2 \wedge \overline{x_1}$.

Функцию $F_7(x_1, x_2)$ называют сложением по модулю два (отрицанием эквивалентности, неравнозначностью, исключающим ИЛИ), она принимает значение 1 только при различных значениях переменных. Эта функция обозначается как $x_1 \oplus x_2$.

Функцию $F_8(x_1, x_2)$ называют дизъюнкцией (логическим сложением, функцией ИЛИ). Она обозначается как $x_1 \vee x_2$. Эта функция равна 1, если равна 1 переменная x_1, или переменная x_2 или обе переменные.

Функция $F_9(x_1, x_2)$ является отрицанием функции ИЛИ и называется функцией ИЛИ-НЕ (стрелкой Пирса). Функция ИЛИ-НЕ обозначается как $\overline{x_1 \vee x_2}$. Она принимает значение 1, только если обе переменные равны 0.

Функция $F_{10}(x_1, x_2)$ называется функцией эквивалентности (равнозначности) и обозначается как $x_1 \approx x_2$. Эта функция равна 1, только в случае, если значения всех переменных равны.

Функцию $F_{12}(x_1, x_2)$ называют правой импликацией. Она обозначается как $x_1 \leftarrow x_2$. $F_{12}(x_1, x_2) = x_1 \vee \overline{x_2}$.

Функцию $F_{14}(x_1, x_2)$ называют импликацией (левой). Она обозначается как $x_1 \to x_2$. Эту функцию можно выразить также, как $F_{14}(x_1, x_2) = \overline{x_1} \vee x_2$.

Функция $F_{15}(x_1, x_2)$ является отрицанием конъюнкции и называется функцией И-НЕ (функцией Шеффера). Она принимает значение 1, только если хотя бы одна переменная равна 0. $F_{15}(x_1, x_2) = \overline{x_1 \wedge x_2}$.

Функции $F_4(x_1, x_2)$, $F_6(x_1, x_2)$, $F_{11}(x_1, x_2)$ и $F_{13}(x_1, x_2)$ являются вырожденными и фактически являются функциями одной переменной:

$F_4(x_1, x_2) = x_1$, $F_6(x_1, x_2) = x_2$, $F_{11}(x_1, x_2) = \overline{x_2}$, $F_{13}(x_1, x_2) = \overline{x_1}$.

Из рассмотренных выше элементарных функций при синтезе цифровых схем наиболее часто используются функции И, ИЛИ, НЕ, И-НЕ и ИЛИ-НЕ.

2.2. Формы логических функций

Используя принцип суперпозиции, можно из элементарных функций построить логическую функцию любого числа переменных. Логическая функция в общем случае может быть записана в различной форме. Наиболее часто используют дизъюнктивные и конъюнктивные нормальные формы.

Дизъюнктивная нормальная форма (ДНФ) логической функции представляет собой дизъюнкцию элементарных конъюнкций. ***Элементарная конъюнкция*** – это конъюнкция переменных или их отрицаний. Например,

$F(x_1, x_2, x_3) = x_1 x_2 \vee x_1 x_3 \vee x_2 x_3$ – функция, записанная в ДНФ.

Аналогично определяется конъюнктивная нормальная форма. ***Конъюнктивная нормальная форма (КНФ)*** логической функции представляет собой конъюнкцию элементарных дизъюнкций. ***Элементарная дизъюнкция*** – это дизъюнкция переменных или их отрицаний.

Например, $F(x_1, x_2, x_3) = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_3}) \wedge (x_1 \vee x_2 \vee x_3)$.

Совершенная дизъюнктивная нормальная форма (СДНФ) – это ДНФ, каждая конъюнкция которой содержит все переменные в прямом или инверсном виде.

Например, $F(x_1, x_2, x_3) = x_1 \overline{x_2} x_3 \vee x_1 x_2 x_3 \vee x_1 x_2 \overline{x_3}$.

Аналогично определяется совершенная конъюнктивная нормальная форма.

Совершенная конъюнктивная нормальная форма(СКНФ) – это КНФ, каждая дизъюнкция которой содержит все переменные в прямом или инверсном виде.

Например, $F(x_1, x_2, x_3) = (\overline{x_1} \wedge x_2 \vee \overline{x_3})(x_1 \vee \overline{x_2} \vee x_3)(x_1 \vee x_2 \vee x_3)$.

Любая логическая функция имеет только одну совершенную ДНФ или КНФ. Совершенные формы логической функции могут быть записаны по таблице истинности.

При записи СДНФ по таблице истинности каждому набору значений переменных, на котором функция равна 1, соответствует конъюнкция всех переменных функции. Если значение переменной в наборе равно 1, то

переменная входит в конъюнкцию в прямом виде, иначе – в инверсном. Дизъюнкция всех полученных конъюнкций образует СДНФ.

При записи СКНФ каждому набору значений переменных, на котором функция равна 0, соответствует дизъюнкция всех переменных функции. Если значение переменной в наборе равно 0, то переменная входит в дизъюнкцию в прямом виде, иначе – в инверсном. Конъюнкция всех полученных дизъюнкций образует СКНФ.

Пример. Пусть логическая функция трех переменных описана следующей таблицей истинности (таблица 2.3).

Таблица 2.3.

Таблица истинности

Значения переменных			Значения функции
a	b	c	F(abc)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Тогда функция будет записана в СДНФ в следующем виде:

$F(abc) = \bar{a}b\bar{c} \vee a\bar{b}\bar{c} \vee a\bar{b}c \vee ab\bar{c} \vee abc$.

СКНФ этой функции будет иметь следующий вид:

$F(abc) = (a \vee b \vee c)(a \vee b \vee \bar{c})(a \vee \bar{b} \vee \bar{c})$.

Обычно при разработке цифровых схем используются дизъюнктивные нормальные формы логических функций.

Кроме нормальных (дизъюнктивных и конъюнктивных) форм логических функций могут использоваться и другие, например, скобочные формы. Скобочные формы записываются с использованием скобок, уточняющих порядок выполнения логических операций. Скобочные формы в некоторых случаях позволяют уменьшить сложность схемы.

2.3. Основные законы алгебры логики

Реализация логических функций связана с выполнением эквивалентных преобразований для получения минимальных форм используемых функций. Эквивалентные преобразования изменяют только форму логической функции, не изменяя её таблицу истинности. Такие преобразования выполняются с использованием законов алгебры логики.

Основные законы алгебры логики имеют следующий вид:

1. Закон двойного отрицания:

$\overline{\overline{a}} = a$.

Проверим этот закон путём составления таблицы истинности.

a	$\overline{a}$	$\overline{\overline{a}}$
0	1	0
1	0	1

Как следует из таблицы истинности, значения функций a и $\overline{\overline{a}}$ совпадают при всех значениях переменных, т.е. эти функции эквивалентны.

2. Законы одинарных элементов:

$a \vee 1 = 1$,

$a \wedge 1 = a$,

$a \vee 0 = a$,

$a \wedge 0 = 0$.

Составим таблицы истинности:

a	$a \vee 1$
0	1
1	1

Как следует из таблицы истинности, значение функции $a \vee 1$ равно 1 при всех значениях переменной a.

a	$a \wedge 1$
0	0
1	1

Как следует из таблицы истинности, значения функций a и $(a \wedge 1)$ совпадают при всех значениях переменных, т.е. эти функции эквивалентны.

a	$a \vee 0$
0	0
1	1

Как следует из таблицы истинности, значения функций a и $(a \vee 0)$ совпадают при всех значениях переменных, т.е. эти функции эквивалентны.

a	$a \wedge 0$
0	0
1	0

Как следует из таблицы истинности, значение функции $a \wedge 0$ равно 0 при всех значениях переменной a.

3. Закон тавтологии (идемпотентности):

$a \vee a = a$,

$a \wedge a = a$.

a	$a \vee a$
0	0
1	1

a	$a \wedge a$
0	0
1	1

Как следует из таблиц истинности, значения функций, находящихся в левой и правой частях равенств, совпадают при всех значениях переменных, т.е. эти функции эквивалентны.

4. Закон исключённого третьего (противоречия):

$a \vee \bar{a} = 1$,

$a \wedge \bar{a} = 0$.

Составим таблицы истинности:

a	$\bar{a}$	$a \vee \bar{a}$
0	1	1
1	0	1

a	$\bar{a}$	$a \wedge \bar{a}$
0	1	0
1	0	0

Как следует из таблицы истинности, значение функции $a \vee \bar{a}$ равно 1, а значение функции $a \wedge \bar{a}$ равно 0 при всех значениях переменной a.

5. Коммуникативный закон:

$a \vee b = b \vee a$,

$a \wedge b = b \wedge a$.

a	b	$a \vee b$	$b \vee a$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

a	b	$a \wedge b$	$b \wedge a$
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

Как следует из таблиц истинности, значения функций, находящихся в левой и правой частях равенств, совпадают при всех значениях переменных, т.е. эти функции эквивалентны.

6. Ассоциативный закон:

$a \vee (b \vee c) = (a \vee b) \vee c$.

$a \wedge (b \wedge c) = (a \wedge b) \wedge c$.

Составим таблицы истинности:

a	b	c	$b \vee c$	$a \vee (b \vee c)$	$a \vee b$	$(a \vee b) \vee c$
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

a	b	c	$b \wedge c$	$a \wedge (b \wedge c)$	$a \wedge b$	$(a \wedge b) \wedge c$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	0	1	0
1	1	1	1	1	1	1

Эквивалентность функций, находящихся в левой и правой частях равенств, следует из приведённых выше таблиц.

7. Дистрибутивный закон:

$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$.

$a \vee b \wedge c = \big(a \vee b) \wedge (a \vee c)\big)$.

a	b	c	$b \vee c$	$a \wedge (b \vee c)$	$(a \wedge b) \vee (a \wedge c)$
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

a	b	c	$b \wedge c$	$a \vee (b \wedge c)$	$a \vee b$	$a \vee c$	$(a \vee b) \wedge (a \vee c)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Эквивалентность функций, находящихся в левой и правой частях равенств, следует из приведённых выше таблиц.

8. Закон отрицания отрицания (Закон де Моргана):

$\overline{a \wedge b} = \overline{a} \vee \overline{b}$.

$\overline{a \vee b} = \overline{a} \wedge \overline{b}$.

Составим таблицы истинности:

a	b	$a \wedge b$	$\overline{a \wedge b}$	$\overline{a}$	$\overline{b}$	$\overline{a} \vee \overline{b}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

a	b	$\overline{a \vee b}$	$\overline{a}$	$\overline{b}$	$\overline{a} \wedge \overline{b}$
0	0	1	1	1	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	0

Эквивалентность функций, находящихся в левой и правой частях равенств, следует из приведённых выше таблиц.

9. Закон поглощения:

$a \vee a \wedge b = a$.

$a \wedge (a \vee b) = a$.

a	b	$a \vee (a \wedge b)$
0	0	0
0	1	0
1	0	1
1	1	1

a	b	$a \wedge (a \vee b)$
0	0	0
0	1	0
1	0	1
1	1	1

Так как значения функций a и $a \vee (a \wedge b)$ в первом случае и значения функций a и $a \wedge (a \vee b)$ во втором случае совпадают при всех значениях переменных, то можно сделать вывод об истинности закона поглощения.

10. Закон склеивания:

$a \wedge b \vee a \wedge \bar{b} = a$.

$(a \vee b) \wedge (a \vee \bar{b}) = a$.

Составим таблицы истинности:

a	b	$a \wedge b$	$a \wedge \bar{b}$	$(a \wedge b) \vee (a \wedge \bar{b})$
0	0	0	0	0
0	1	0	0	0
1	0	0	1	1
1	1	1	0	1

a	b	$a \vee b$	$a \vee \bar{b}$	$(a \vee b) \wedge (a \vee \bar{b})$
0	0	0	1	0
0	1	1	0	0
1	0	1	1	1
1	1	1	1	1

Эквивалентность левых и правых частей равенств и, соответственно, истинность законов склеивания следует из приведённых выше таблиц.

11. Закон Порецкого:

$a \vee \bar{a} \wedge b = a \vee b$.

$a \wedge (\bar{a} \vee b) = a \wedge b$.

Таблицы истинности имеют следующий вид:

a	b	$\bar{a}$	$\bar{a} \wedge b$	$a \vee (\bar{a} \wedge b)$	$a \vee b$
0	0	1	0	0	0
0	1	1	1	1	1
1	0	0	0	1	1
1	1	0	0	1	1

a	b	$\bar{a} \vee b$	$a \wedge (\bar{a} \vee b)$	$a \wedge b$
0	0	0	0	0
0	1	1	0	0
1	0	0	0	0
1	1	1	1	1

Истинность закона Порецкого следует из сравнения 5 и 6 столбцов приведённых таблиц.

Следует отметить, что в соответствии с принципом суперпозиции законы алгебры логики применимы не только к простым переменным, но и к функциям.

ПРИМЕР.

По закону поглощения

$F(abc) = ac \vee ac(b \vee c) = ac$.

Законы алгебры логики используются также для вычисления значений логических функций по известным значениям всех переменных. При этом следует учитывать порядок выполнения логических операций с учётом их старшинства:

1. Действия в скобках.
2. Одноместные операции (отрицание или инверсия).
3. Конъюнкция (И).
4. Дизъюнкция (ИЛИ).
5. Сложение по модулю 2.

2.4. Техническая реализация логических функций

Вычисления значений логических функций для любых наборов значений переменных выполняется с помощью логических схем. Логические схемы называют также комбинационными схемами, так как значение сигнала на выходе такой схемы зависит только от комбинации значений сигнала на входе схемы. Комбинационная схема строится из логических элементов. Логические элементы выполняют (реализуют) элементарные логические функции.

При создании комбинационных схем необходимо стремиться не только к минимальному количеству элементов, но и к их минимальному разнообразию (минимальному числу типов элементов). Набор типов элементов, при использовании которого можно построить любую комбинационную схему, называется ***функционально полной системой логических элементов.***

Аналогично можно поставить вопрос о минимальном наборе элементов логических функций, с помощью которого можно записать любую логическую функцию. Такой набор называется ***функционально полной системой логических функций.*** Существует несколько функционально полных систем. Например, известно, что любую логическую функцию можно записать в СДНФ. В такой форме используются только элементарные логические функции И, ИЛИ и НЕ. Поэтому логические функции И, ИЛИ и НЕ составляют функционально полную систему. Эта система представляется вполне естественной, но она избыточна. Доказано, что функционально полную систему могут образовать функции И и НЕ, а также функции ИЛИ и НЕ. Кроме того, функция И-НЕ, как и функция ИЛИ-НЕ, также представляет собой функционально полную систему.

Применительно к логическим элементам это означает, что функционально полную систему элементов могут составлять элементы И, ИЛИ, НЕ или И-НЕ, а также элемент ИЛИ-НЕ. В настоящее время для построения цифровых схем чаще всего используют элементы И-НЕ. Таким образом, все электронные схемы компьютера можно построить на элементах одного типа, И-НЕ (ИЛИ-НЕ).

Ниже приведены графические графические изображения некоторых логических элементов.

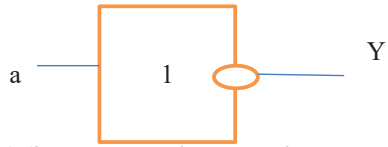
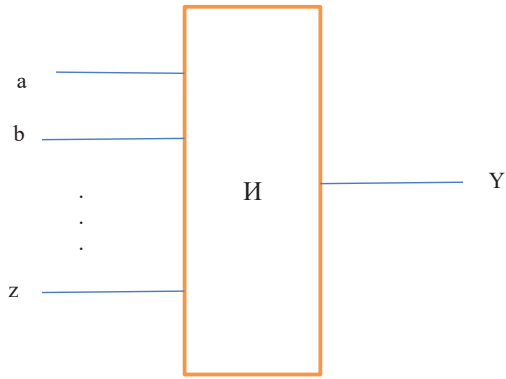


Рис. 2.1. Элемент НЕ (инвертор)

Этот элемент реализует функцию $Y = \bar{a}$.



.

Рис. 2.2. Элемент И (конъюнктор)

Этот элемент реализует функцию $Y = a \wedge b \wedge \ldots \wedge z$.

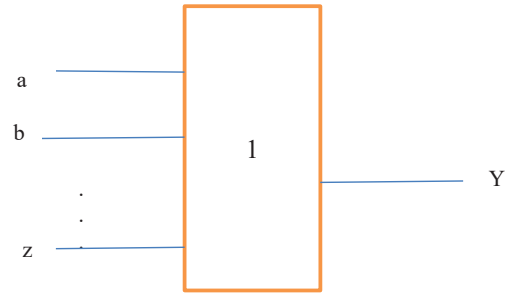
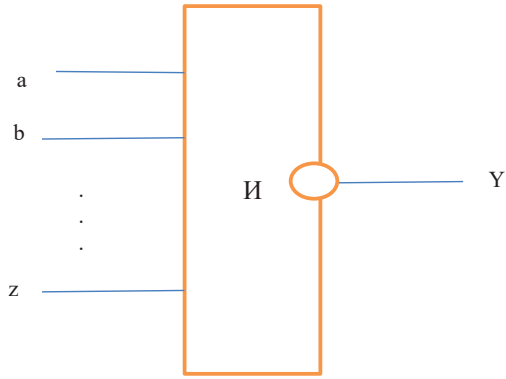


Рис. 2.3. Элемент ИЛИ (дизъюнктор)

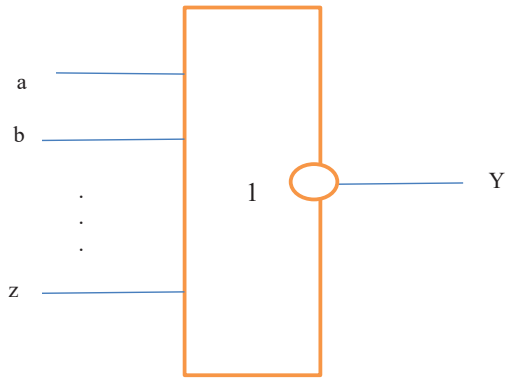
Этот элемент реализует функцию $Y = a \vee b \vee \ldots \vee z$.



.

Рис. 2.4. Элемент И-НЕ

Этот элемент реализует функцию $Y = \overline{a \wedge b \wedge \ldots \wedge z}$.



.

Рис. 2.5. Элемент ИЛИ-НЕ

Этот элемент реализует функцию $Y = \overline{a \vee b \vee \ldots \vee z}$.

Вычерчивание электронных схем, реализующих заданные логические функции, выполняется путём замены элементарных логических функций соответствующими логическими элементами. Затем элементы объединяют с учётом старшинства логических функций. Если реализуемая функция записана в СДНФ, то логическая схема состоит из трёх ярусов. В первом ярусе располагаются инверторы, во втором – элементы И. Выходные сигналы элементов И поступают на входы элемента ИЛИ, представляющего третий ярус схемы. Число входов элемента ИЛИ равно числу конъюнкций в СДНФ.

Пример. Построить функциональную схему, реализующую логическую функцию $F(abc) = abc \vee \bar{a}bc \vee ab\bar{c}$. (рис. 2.6).

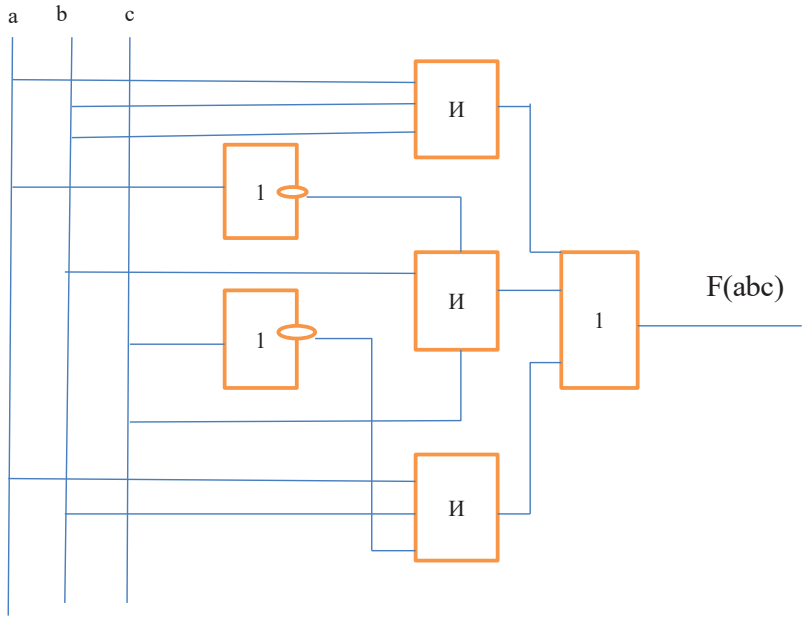


Рис. 2.6. Функциональная схема, реализующая логическую функцию

2.5. Типы цифровых автоматов

В зависимости от типа элементов, из которых построен автомат, различают два основных типа цифровых автоматов:

1) комбинационные схемы;
2) цифровые автоматы с памятью.

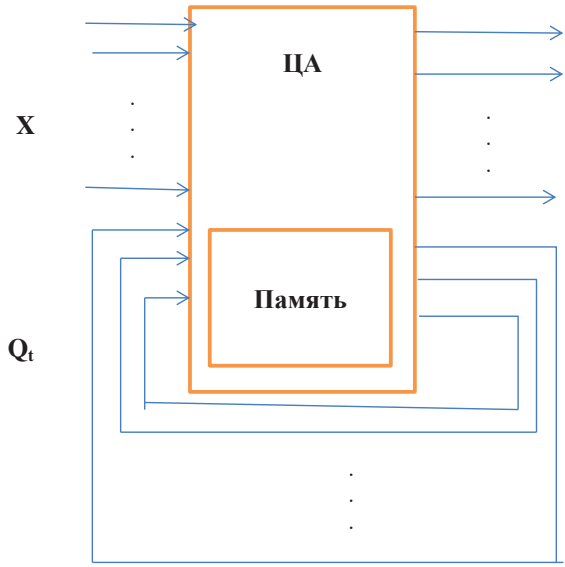
Комбинационные схемы состоят только из логических элементов (И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ и т.д.). Комбинационные схемы могут иметь несколько входов и несколько выходов.

На рис. 2.7. представлен вид такой комбинационной схемы.

Рис. 2.7. Вид комбинационной схемы

Главной особенностью комбинационных схем является то, что сигнал на выходе комбинационных схем зависит только от комбинации сигналов на входах схемы и не зависит ни от времени, ни от предыстории входных сигналов. Таким образом, при многократном поступлении одного и того же входного сигнала на выходах комбинационной схемы будет формироваться один и тот же входной сигнал.

Цифровые автоматы с памятью состоят из логических элементов и элементов памяти, изображённых на рисунке 2.8. Информация, записанная в элементах памяти такого автомата, называется состоянием цифрового автомата. Основная особенность цифровых автоматов с памятью состоит в том, что сигнал на выходе автомата зависит как от входного сигнала, так и от состояния автомата.



$$Y = Y(X_t, Q_t)$$

Рис. 2.8. Вид цифрового автомата с памятью

В свою очередь, состояние автомата зависит от того, какие сигналы поступали на его входы до настоящего момента времени, т.е. от предыстории работы автомата. Поэтому при многократном поступлении одного и того же входного сигнала на выходах автомата с памятью могут формироваться различные сигналы.

При сопоставлении комбинационных схем и цифровых автоматов с памятью можно сделать вывод, что комбинационные схемы имеют единственное состояние. Комбинационные схемы используются для технической реализации логических функций. При помощи комбинационной схемы определяется значение логической функции для заданных значений

логических переменных. В теории автоматов обычно считается, что при изменении сигналов на входах схемы выходной сигнал мгновенно принимает соответствующее значение, т.е. задержка сигналов логическими элементами не учитывается.

В цифровых автоматах с памятью формирование выходного сигнала занимает определённое время, которое обычно разбивается на такты. В каждом такте в зависимости от входного сигнала и состояния автомата формируется выходной сигнал и новое состояние. Таким образом, можно сказать, что выходной сигнал автомата с памятью зависит от последовательности входных сигналов, поэтому цифровые автоматы с памятью называют также последовательностными схемами.

2.6. Задачи анализа и синтеза комбинационных схем

Структурно комбинационная схема может быть представлена как совокупность элементарных логических схем – логических элементов. Логические элементы выполняют над входными переменными элементарные логические операции типа И-НЕ, И, ИЛИ, ИЛИ-НЕ и т.д. Число входов логического элемента соответствует числу аргументов воспроизводимой им булевой функции. Графическое изображение комбинационной схемы, при котором показаны связи между различными элементами, а сами элементы представлены условными обозначениями, называется ***функциональной*** схемой.

Основными характеристиками комбинационных схем являются сложность и быстродействие.

Сложность комбинационной схемы оценивается количеством оборудования, составляющего схему. При разработке схем на основе конкретной элементной базы количество оборудования обычно измеряется числом корпусов (модулей) интегральных микросхем, используемых в схеме. В теоретических разработках ориентируются на произвольную элементную базу и поэтому для оценки затрат оборудования используется оценка сложности схем по Квайну.

Сложность (цена) по Квайну определяется суммарным числом входов логических элементов в составе схемы. При такой оценке единица сложности– один вход логического элемента. Цена инверсного входа обычно принимается равной двум (если на вход схемы сигналы поступают только в прямой форме). Такой подход к оценке сложности оправдан по следующим причинам:

1) сложность схемы легко вычисляется по логическим функциям, на основе которых строится схема: для ДНФ сложность схемы равна сумме количества букв (букве со знаком отрицания соответствует цена 2) и количества знаков дизъюнкции, увеличенного на 1 для каждого дизъюнктивного выражения;

2) все классические методы минимизации логических функций обеспечивают минимальность схемы именно в смысле цены по Квайну.

Практика показывает, что схема с минимальной ценой по Квайну обычно реализуется наименьшим числом конструктивных элементов – корпусов интегральных микросхем.

Быстродействие комбинационной схемы оценивается максимальной задержкой сигнала при прохождении его от входа схемы к выходу, т.е. определяется промежутком времени от момента поступления входных сигналов до момента установления соответствующих значений выходных сигналов. Задержка сигнала кратна числу элементов, через которые проходит сигнал от входа к выходу схемы. Поэтому быстродействие схемы характеризуется значением $r\tau$, где τ - задержка сигнала на одном элементе. Значение r определяется количеством уровней комбинационной схемы, которое рассчитывается следующим образом. Входам КС приписывается уровень ***нулевой***. Логические элементы, связанные только с входами схемы относятся к уровню ***первому.*** Элемент относится к уровню ***k*,** если он связан по входам с элементами уровней ***k*-1, *k*-2,** и т.д. Максимальный уровень элементов ***r*** определяет количество уровней КС, называемое ***рангом схемы***.

Как известно, любая логическая функция может быть представлена в ДНФ, которой соответствует двухуровневая комбинационная схема. Следовательно, быстродействие любой КС в принципе можно довести до 2τ.

Минимизация булевой функции с целью уменьшения сложности схем обычно приводит к необходимости представления функций в скобочной форме, которой соответствуют схемы с $r > 2$. Таким образом, уменьшение затрат оборудования в общем случае приводит к снижению быстродействия схем.

В ходе разработки комбинационных схем приходится решать задачи анализа и синтеза.

Задача анализа состоит в определении статических и динамических свойств комбинационной схемы. В статике определяются логические функции, реализуемые комбинационной схемой по известной ей структуре. В динамике рассматривается способность надёжного функционирования схемы в переходных процессах при смене значений переменных на входах схемы, т.е. определяется наличие на выходах схемы возможных нежелательных импульсных сигналов, которые не следуют непосредственно из выражений для логических функций, реализуемых схемой.

Задача синтеза заключается в построении из заданного набора логических элементов комбинационной схемы, реализующей заданную систему логических функций.

Решение задачи синтеза не является однозначным, можно предложить различные варианты комбинационных схем, реализующих одну и ту же систему логических функций, но отличающихся по тем или иным параметрам. Разработчик комбинационных схем из этого множества вариантов выбирает один, исходя из дополнительных критериев: минимального количества логических элементов, необходимых для реализации схемы, максимального быстродействия и т.д. Существуют

различные методы синтеза комбинационных схем, среди которых наиболее разработан канонический метод.

2.7. Способы задания цифровых автоматов

Задать автомат - значит описать алгоритм его работы. Автомат считается заданным, если известны:

- алфавит входных сигналов $X = \{X_1, X_2, \ldots, X_i, \ldots, X_n\}$;
- алфавит выходных сигналов $Y = \{Y_1, Y_2, \ldots, Yj, \ldots, Y_m\}$;
- алфавит состояний $Q = \{Q_1, Q_2, \ldots, Q_r, \ldots, Q_k\}$;
- начальное состояние Q_0;
- функция переходов $Q_{t+1} = Q(X_t, Q_t)$;
- функция выходов $Y_t = Y(X_t, Q_t)$.

В приведенном перечне объектов основными являются функции переходов и выходов. Для задания автоматов используются следующие способы:

- словесное описание;
- графическое описание;
- табличное описание;
- аналитическое описание.

Словесное описание используется на начальном этапе задания автомата, а его элементы находят применение в других способах. Недостатком словесного описания является его громоздкость и возможная неоднозначность.

При графическом задании используются направленные графы, схемы алгоритмов, временные диаграммы и функциональные схемы. Графическое задание предполагает использование специальных обозначений и выполняется по специальным правилам (стандартам). Графическое задание обладает наглядностью, но для сложных алгоритмов наглядность снижается. При задании автомата при помощи направленных графов каждой вершине графа соответствует состояние автомата, дуги графа указывают переходы автомата в новое состояние при определенных входных сигналах. Для каждого входного сигнала (или состояния) указывается также выходной сигнал. Способ обозначения выходного сигнала на графе зависит от типа автомата.

Табличное описание используется обычно для перехода от графической формы к аналитической. При этом алгоритм работы автомата задается с помощью таблиц переходов и выходов. Обычно таблицы переходов и выходов совмещаются в одну таблицу. В этой таблице для каждого состояния и каждого входного сигнала указываются следующее состояние и выходной сигнал.

Аналитическое описание сводится к заданию конкретного вида функций переходов и выходов. Обычно эти функции записываются в дизъюнктивной нормальной форме как функции, в которых аргументами являются сигналы на отдельных входах автомата, а также сигналы на

выходах отдельных элементах памяти.

3. Минимизация логических функций

3.1. Общая характеристика методов минимизации логических функций

Любая логическая функция может быть записана в различной форме. Различные формы логической функции могут иметь различную сложность, т.е. для их реализации в виде комбинационной схемы может требоваться различное количество логических элементов. Поэтому перед реализацией логической функции должна быть предпринята попытка минимизировать (упростить) эту функцию. Для минимизации логических функций могут быть использованы различные методы.

По форме представления исходной логической функции различают следующие методы минимизации:

- аналитические методы;
- графические методы.

В аналитических методах минимизации логическая функция представляется обычно в СДНФ. Из этих методов чаще всего используются метод непосредственных преобразований и метод Квайна.

В графических методах логическая функция задается в виде графа или в виде специальной диаграммы. Наиболее известным из графических методов является метод Карно.

По степени приближения результата минимизации к минимальной форме логической функции различают точные и приближенные методы минимизации. Точные методы позволяют получить строго минимальную форму логической функции. К таким методам относятся метод Квайна и метод интервалов. Точные методы требуют много времени для их реализации, поэтому часто применяют приближенные методы. Эти методы не гарантируют получения минимальной формы. Результатом реализации таких алгоритмов является форма функции, близкая к минимальной (может быть и минимальная, но не всегда). Достоинством приближенных методов минимизации является приемлемое время их выполнения. Приближенным методом является, например, метод непосредственных преобразований.

3.2. Общая последовательность минимизации

Основной операцией, которая обеспечивает упрощение формы логической функции, является операция склеивания, сущность которой может быть записана следующим образом:

$$ab \vee a\bar{b} = a \qquad (3.1)$$

или в общем виде:

$$aF \vee a\bar{F} = a \qquad (3.2)$$

где F - произвольная логическая функция.

Одна и та же конъюнкция может склеиваться с несколькими другими конъюнкциями. В этом случае используется правило неполного склеивания:

$$aF \vee a\bar{F} = a \vee aF \vee a\bar{F}. \quad (3.3.)$$

При неполном склеивании результат склеивания дополняется исходными конъюнкциями, которые участвовали в склеивании. Неполное склеивание не приводит к усложнению функции, так как исходные конъюнкции могут быть при необходимости исключены из логической функции путем использования правила поглощения:

$$a \vee aF = a. \quad (3.4)$$

Введём некоторые определения.

Импликантой данной логической функции называется другая логическая функция, которая принимает значение 1 только на части входных комбинаций, на которых данная логическая функция равна 1. Рассмотрим пример (таблица 3.1).

Таблица 3.1

Значения функции и её импликант

Переменные			Функции		
a	b	c	F	F1	F2
0	0	0	1	1	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	1	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	0	0	0
1	1	1	1	1	0

В таблице 3.1. заданы значения некоторой функции F. При этом функции F1 и F2 являются импликантами функции F.

Логические функции часто задаются в СДНФ. Для краткости иногда вместо записи СДНФ функция задается перечислением десятичных номеров наборов значений переменных, на которых функция принимает значение «1».

Каждая конъюнкция СДНФ задает один набор значений переменных, на которых логическая функция принимает значение 1 (единичный набор). Поэтому единичный набор является импликантой. Импликантой является также результат склеивания импликант между собой. Импликанты, полученные при склеивании, задают уже не один, а несколько единичных наборов, т.е. интервал единичных значений или единичный интервал. Единичный интервал может включать 1, 2, 4, 8 и т.д. единичных наборов. Таким образом, понятия импликанты и единичного интервала являются синонимами и отличаются лишь по форме. Импликанты записываются в буквенной форме, а единичные интервалы - в цифровой форме (в виде двоичных или десятичных чисел).

Импликанты могут склеиваться друг с другом и порождать новые, более простые, импликанты.

Если импликанта не склеивается с другими импликантами, она называется ***простой импликантой***, а соответствующий ей единичный

интервал называется ***максимальным единичным интервалом.***

Каждая логическая функция имеет определенный набор простых импликант. Дизъюнкция всех простых импликант данной логической функции называется ***сокращенной ДНФ*** (сокращенной дизъюнктивной нормальной формой) данной логической функции.

ПРИМЕР.

Пусть $F(abcd) = \overset{2}{\bar{a}\bar{b}c\bar{d}} \vee \overset{4}{\bar{a}b\bar{c}\bar{d}} \vee \overset{6}{\bar{a}bc\bar{d}} \vee \overset{12}{ab\bar{c}\bar{d}} \vee \overset{14}{abc\bar{d}}$

В данном примере над каждой конъюнкцией указан ее десятичный номер. После склеивания получим сокращенную ДНФ исходной функции:

$F(abcd) = b\bar{d} \vee \bar{a}c\bar{d} \vee a\bar{b}cd$.

Сокращенная ДНФ данной функции включает три простых импликанты (максимальных единичных интервала). Для записи интервалов в цифровой форме следует записать последовательность символов «0», «1» и «-», длина которой равна числу переменных. Если импликанта содержит переменную в прямой форме, вместо переменной в последовательности записывается символ «1». Вместо переменной в инверсной форме записывается символ «0», а вместо отсутствующих переменных записывается символ «-». Например, импликанте $b\bar{d}$ соответствует интервал ***-1-0.*** Так интервалы записываются в двоичной форме.

Для записи десятичной формы интервала следует вместо символов «-» записать все возможные комбинации символов «0» и «1». При этом образуются двоичные номера всех единичных наборов, входящих в данный интервал. Далее записываются десятичные номера наборов, соответствующих нижней и верхней границам интервала так, как это показано ниже:

$b\bar{d} \rightarrow$ ***-1-0*** $\rightarrow 0100 \rightarrow (4,14)$
0110
1100
1110

$\bar{a}c\bar{d} \rightarrow$ ***0-10*** $\rightarrow 0010 \rightarrow (2,6)$
0110

$a\bar{b}cd \rightarrow$ ***1001*** $\rightarrow (9,9)$

В данном примере символы «0» и «1» исходных интервалов выделены полужирным курсивом.

В сокращенной ДНФ могут быть лишние импликанты, т.е. такие, при отбрасывании которых значение логической функции не меняется. При отбрасывании лишних импликант из сокращенной ДНФ получается ***тупиковая*** форма исходной логической функции. Особенность тупиковой формы заключается в том, что эта форма не может быть упрощена.

Логическая функция может иметь несколько тупиковых форм.

Различные тупиковые формы одной и той же логической формы могут иметь различную сложность. Для сравнительной оценки сложности логических функций используют такие показатели, как суммарное число букв в логической функции, суммарное число букв плюс число конъюнкций в логической функции, суммарное количество входов элементов, необходимых для реализации логической функции (сложность по Квайну) и др.

При наличии нескольких тупиковых форм логической функции одна из тупиковых форм, имеющая наименьшую сложность, называется ***минимальной*** формой данной логической функции.

В общем случае последовательность минимизации логических функций может иметь вид, показанный на рис. 3.1.

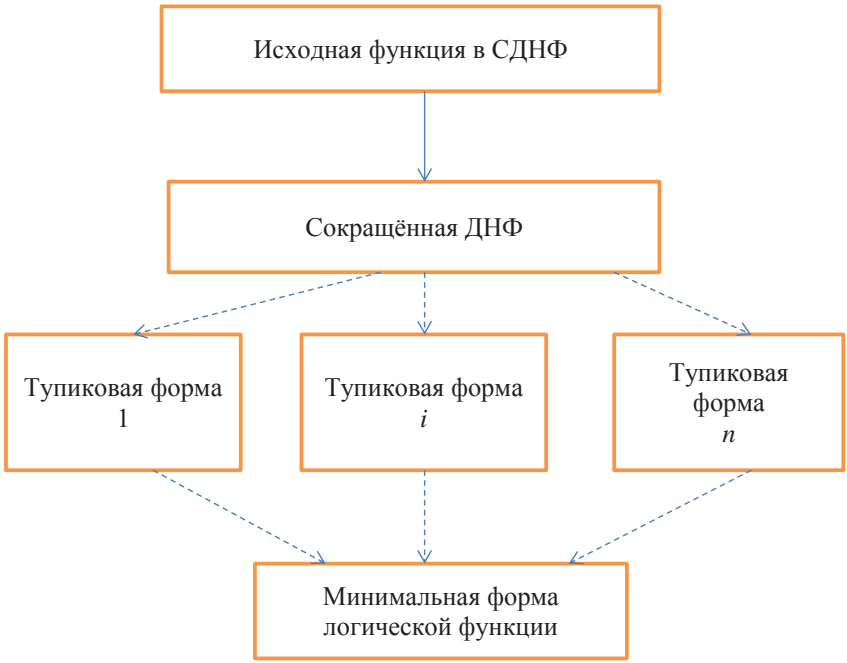


Рис. 3.1. Вид последовательности минимизации логических функций

Минимизируемая логическая функция представляется обычно в совершенной дизъюнктивной нормальной форме (СДНФ). Путем применения логических операций неполного склеивания и поглощения из СДНФ получают сокращенную ДНФ исходной логической функции. После отбрасывания лишних импликант из сокращенной ДНФ получают несколько

тупиковых форм логической функции. Далее из тупиковых форм по критерию сложности выбирается минимальная форма исходной логической функции.

Приведенная последовательность реализуется в точных методах минимизации. В приближенных методах минимизации находится лишь одна из тупиковых форм, которая и принимается за минимальную форму логической функции.

Конкретный набор операций и их последовательность при минимизации зависят от используемого метода.

3.3. Метод непосредственных преобразований

Метод непосредственных преобразований является аналитическим приближенным методом минимизации. Результат минимизации существенно зависит от квалификации исполнителя. Поэтому метод не является строго алгоритмическим и не гарантирует получения не только минимальной, но даже и тупиковой формы логической функции. Однако этот метод широко используется в инженерной практике для минимизации простых логических функций.

Сущность метода заключается в последовательном применении к исходной логической функции приведенных выше операций склеивания, неполного склеивания и поглощения до тех пор, пока эти операции применимы к исходной форме логической функции или к промежуточным формам исходной функции.

В процессе минимизации могут быть полезными также следующие известные соотношения и правила алгебры логики:

$$\begin{array}{l} x \wedge x \wedge x ... \wedge x = x;\ x \vee x \vee x ... \vee x = x; \\ x \wedge \bar{x} = 0;\ x \vee \bar{x} = 1; \\ x \wedge 1 = x;\ x \vee 1 = 1; \\ x \wedge 0 = 0;\ x \vee 0 = x; \\ \bar{\bar{x}} = x;\ \overline{\overline{x}} = x. \end{array} \qquad (3.5)$$

В некоторых случаях полезно использовать также правило свертки

$$x \vee \bar{x}y = x \vee y \qquad (3.6)$$

или правило Порецкого

$$xy \vee \bar{x}z \vee yz = xy \vee \bar{x}z \qquad (3.7)$$

Рассмотрим примеры.

ПРИМЕР. Минимизировать логическую функцию F(abc) методом непосредственных преобразований.

$$F(abc) = \bar{a}\bar{b}c \vee \bar{a}b\bar{c} \vee a\bar{b}\bar{c} \vee abc. \qquad (3.8)$$

Так как первая конъюнкция может склеиваться со второй и с третьей, то в результате применения операции склеивания получим:

$$F(abc) = (\bar{a}\bar{b}c \vee \bar{a}b\bar{c}) \vee (\bar{a}\bar{b}c \vee a\bar{b}\bar{c}) \vee abc = \bar{a}\bar{c} \vee \bar{b}\bar{c} \vee abc. \qquad (3.9)$$

В данном примере одна из конъюнкций дважды участвует в

склеивании, а последняя конъюнкция не склеивается с другими.

ПРИМЕР. Минимизировать логическую функцию F(abcd) методом непосредственных преобразований.

$$F(abcd) = abcd \vee abc\bar{d} \vee ab\bar{c}d \vee ab\bar{c}\bar{d} \vee \bar{a}bcd \vee \bar{a}bc\bar{d}. \qquad (3.10)$$

Последовательно применяя операцию склеивания, получим:

$$F(abcd) = (abcd \vee abc\bar{d}) \vee (ab\bar{c}d \vee ab\bar{c}\bar{d}) \vee (abcd \vee \bar{a}bcd) \vee \bar{a}bc\bar{d} = \\ = (abc \vee ab\bar{c}) \vee bcd \vee \bar{a}bc\bar{d} = ab \vee bcd \vee \bar{a}bc\bar{d} \qquad (3.11)$$

В данном примере конъюнкции, полученные в результате склеивания, склеиваются затем между собой. При каждом склеивании число букв в конъюнкциях (ранг конъюнкции) уменьшается на единицу. Поэтому при повторном склеивании вместо конъюнкций из четырех букв получена одна конъюнкция из двух букв.

При большом количестве логических переменных применение метода непосредственных преобразований становится затруднительным, но принципиально возможным. Но при этом трудно гарантировать даже получение хотя бы одной из тупиковых форм.

Приведем пример минимизации логической функции пяти переменных:

$$F(abcdf) = \bar{a}bc\bar{d}\bar{f} \vee abcdf \vee \bar{a}bcd\bar{f} \vee ab\bar{c}d\bar{f} \vee \bar{a}bc\bar{d}f \vee \bar{a}b\bar{c}df \vee abcd\bar{f} \vee \bar{a}bcdf \vee ab\bar{c}df. \qquad (3.12)$$

$$F(abcdf) = (\bar{a}bc\bar{d}\bar{f} \vee \bar{a}b\bar{c}df) \vee (\bar{a}bcd\bar{f} \vee \bar{a}bcdf) \vee (ab\bar{c}d\bar{f} \vee ab\bar{c}df) \vee (abcd\bar{f} \vee abcdf) \vee \\ \vee (\bar{a}bc\bar{d}\bar{f} \vee \bar{a}bcdf) = (\bar{a}b\bar{c}d \vee \bar{a}bcd) \vee (ab\bar{c}d \vee abcd) \vee \bar{a}bcf = (\bar{a}bd \vee abd) \vee \bar{a}bcf = \\ = bd \vee \bar{a}bcf. \qquad (3.13)$$

Из данного примера следует, что даже для такой сравнительно несложной функции поиск конъюнкций, которые могут быть склеены, представляет значительные трудности. Так как при минимизации выполняется большое количество сравнительно несложных операций, для упрощения реальных логических функций, которые встречаются в инженерной практике, целесообразно использовать машинные программы. Для минимизации же логических функций с числом переменных не более четырех широкое применение нашли графические методы, в частности метод Карно.

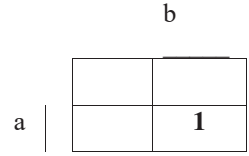
3.4. Метод Карно

Метод Карно представляет собой приближенный метод минимизации логических функций, который при определенных навыках позволяет получить минимальную форму исходной функции. Исходная логическая функция представляется в СДНФ. Минимизированная логическая функция представлена в дизъюнктивной нормальной форме.

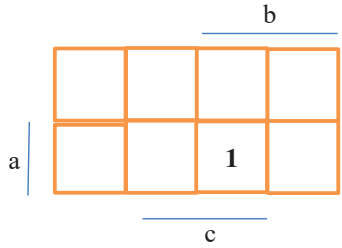
Сущность метода Карно заключается в построении специальных диаграмм, представляющих собой прямоугольные таблицы. Каждой конъюнкции исходной логической функции соответствует одна из клеток таблицы. Главная особенность диаграмм Карно заключается в том, что

конъюнкциям, которые могут быть склеены, соответствуют соседние по строке или по столбцу клетки.

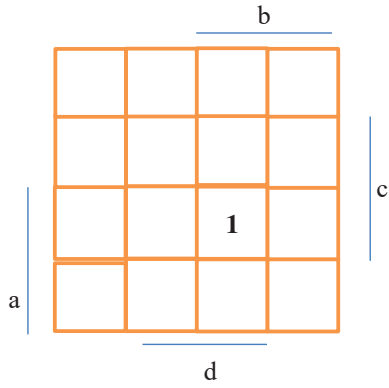
Примеры диаграмм для логических функций двух, трех и четырех переменных представлены на рис. 3.2.



а)



б)



в)

Рис. 3.2. Диаграммы Карно для логических функций: а) двух; б)трёх; в) четырёх переменных

Диаграмма Карно по каждой переменной делится на две половины. Половина диаграммы, отмеченная соответствующей буквой, содержит все конъюнкции, в которых эта буква записана в прямой форме (без отрицания). На рис. 3.2 символом 1 отмечены клетки, которые соответствуют следующим конъюнкциям:

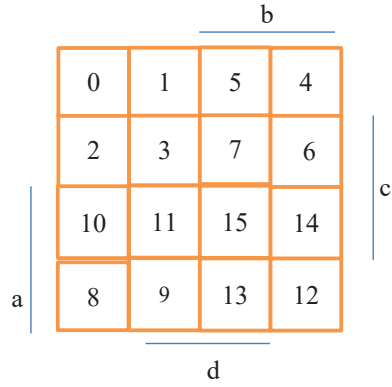
F(ab) = ab; F(abc)=abc; F(abcd) = abcd.

В дальнейшем будем рассматривать применение диаграмм Карно на примере функций четырех переменных. Заполнение диаграмм Карно упрощается при использовании эталонных диаграмм. На эталонной диаграмме клетки помечены номерами соответствующих конъюнкций исходной логической функции. Номера конъюнкций соответствуют двоичным комбинациям входных сигналов в таблице истинности логической функции. Эталонная диаграмма для функции четырех переменных, а также пример ее заполнения по таблице истинности (таблица 3.2.) приведены на рис. 3.3.

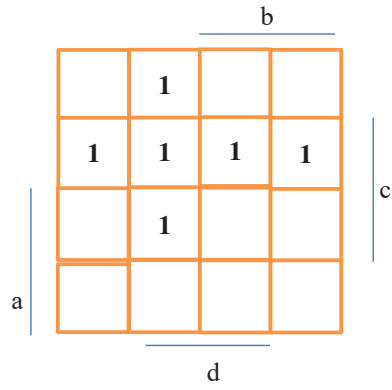
Таблица 3.2.

Таблица истинности функций 4 переменных

Номера наборов	Значения переменных				Значение функции
	a	b	c	d	
0	0	0	0	0	0
1	0	0	0	1	**1**
2	0	0	1	0	**1**
3	0	0	1	1	**1**
4	0	1	0	0	0
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	0
10	1	0	1	0	0
11	1	0	1	1	**1**
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0



а)



б)

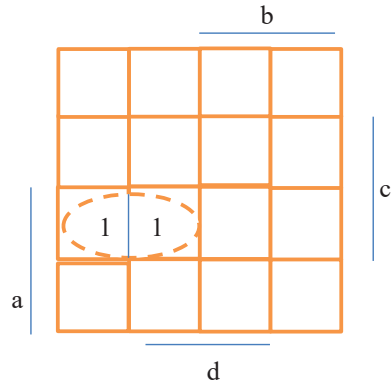
Рис. 3.3. Диаграммы функции 4 переменных: а) эталонная диаграмма; б) отмеченная диаграмма

Следует отметить, что вид эталонной диаграммы зависит от способа её разметки по переменным a, b, c, d.

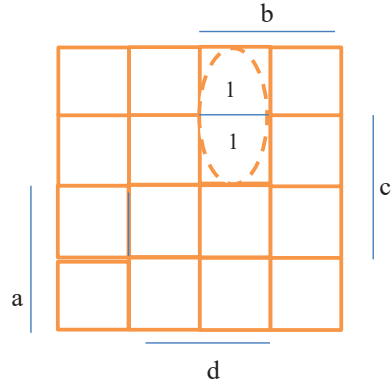
При использовании метода Карно общая идея заключается в том, что отмеченные клетки диаграммы объединяются в группы по 2, 4 или 8 клеток. Каждая отмеченная клетка должна входить, по крайней мере, в одну группу.

Каждая группа должна содержать только отмеченные клетки. Группы клеток должны образовывать квадраты или прямоугольники. При этом число групп должно быть минимальным, а количество квадратов в группах - максимальным .

При объединении двух клеток в одну группу получается одна конъюнкция, ранг которой на единицу меньше ранга исходных конъюнкций. При этом полученная конъюнкция не содержит той логической переменной, которая различным образом входит в исходные конъюнкции (без инверсии и с инверсией). На рис. 3.4 показаны примеры групп из двух конъюнкций (клеток) и результаты склеивания.



а)

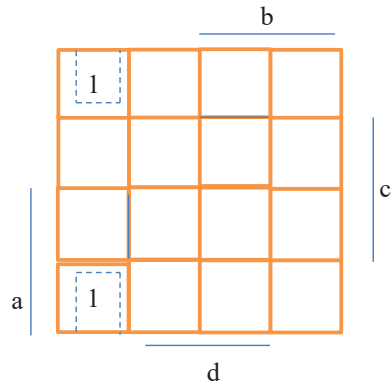


б)

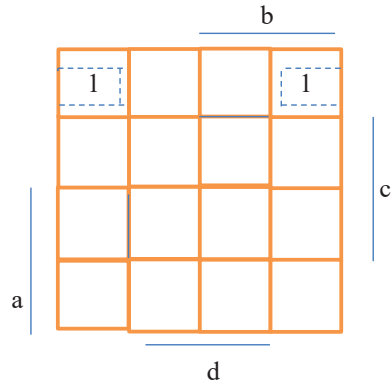
Рис. 3.4. Примеры групп из двух конъюнкций (клеток) и результаты склеивания:

а) $F(abcd) = a\bar{b}c\bar{d} \vee a\bar{b}cd = a\bar{b}c$; $F(abcd) = \bar{a}b\bar{c}d \vee \bar{a}bcd = \bar{a}bd$

Диаграммы Карно строятся таким образом, что соседними являются крайние в столбце или в строке клетки. Примеры объединения таких клеток в группы приведены на рис. 3.5.



а)



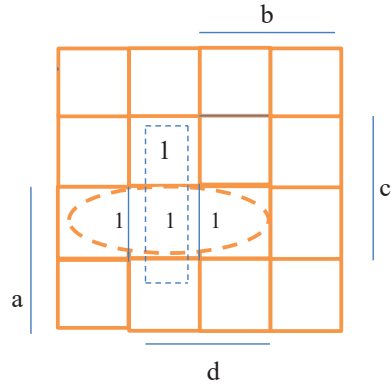
б)

Рис. 3.5.: а) $F(abcd) = \bar{a}\bar{b}\bar{c}\bar{d} \vee a\bar{b}\bar{c}\bar{d} = \bar{b}\bar{c}\bar{d}$;

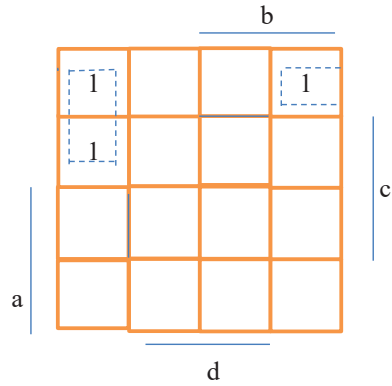
б) $F(abcd) = \bar{a}\bar{b}\bar{c}\bar{d} \vee \bar{a}b\bar{c}\bar{d} = \bar{a}\bar{c}\bar{d}$.

В соответствии с правилами неполного склеивания и поглощения одна и та же конъюнкция может склеиваться с несколькими другими конъюнкциями.

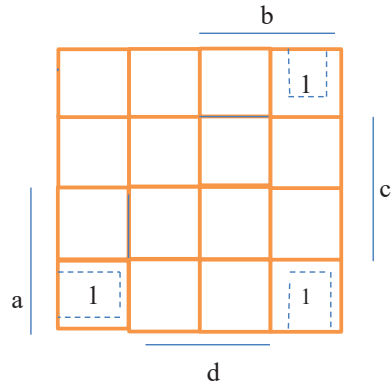
Примеры такого рода приведены на рис. 3.6.



а)



б)



в)

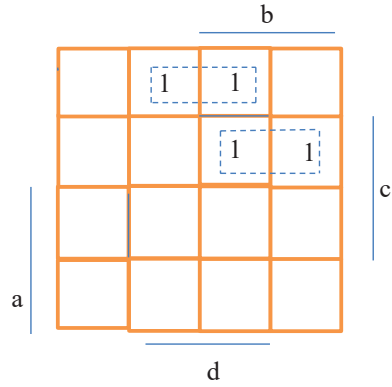
Рис. 3.6. Примеры склеивания одной и той же конъюнкции с другими конъюнкциями:

а) $F(abcd) = a\bar{b}cd \vee a\bar{b}\bar{c}d \vee a\bar{b}\bar{c}\bar{d} \vee abcd = a\bar{b}c \vee \bar{b}cd \vee acd$;

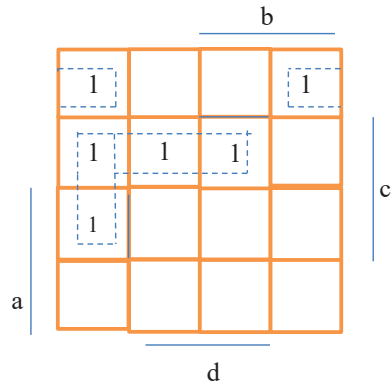
б) $F(abcd) = a\bar{b}\bar{c}d \vee ab\bar{c}d \vee a\bar{b}cd = a\bar{c}d \vee a\bar{b}d$;

в) $F(abcd) = a\bar{b}\bar{c}d \vee a\bar{b}\bar{c}\bar{d} \vee ab\bar{c}\bar{d} = a\bar{c}\bar{d} \vee b\bar{c}\bar{d}$

На одной диаграмме может быть несколько групп с одинаковым числом клеток, как это показано на рис. 3.7.



а)



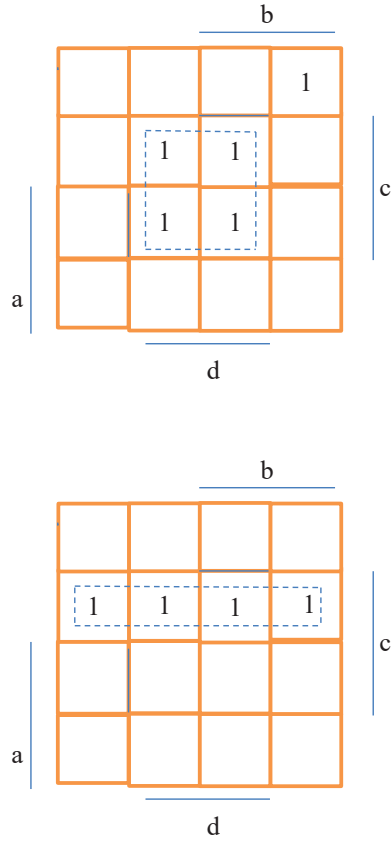
б)

Рис. 3.7. Примеры нескольких групп с одинаковым числом клеток:

а) $F(abcd) = \bar{a}\bar{b}\bar{c}d \vee \bar{a}b\bar{c}d \vee \bar{a}bc\bar{d} \vee ab\bar{c}\bar{d} = \bar{a}\bar{c}d \vee \bar{a}bc$;

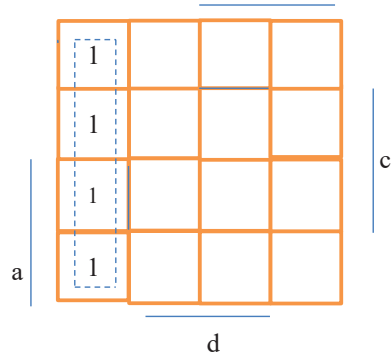
б) $F(abcd) = \bar{a}\bar{b}\bar{c}\bar{d} \vee \bar{a}b\bar{c}\bar{d} \vee \bar{a}\bar{b}cd \vee \bar{a}bcd \vee a\bar{b}cd \vee a\bar{b}\bar{c}\bar{d} = \bar{a}cd \vee \bar{b}cd \vee a\bar{c}d$.

При объединении четырех конъюнкций в группу получается конъюнкция, ранг которой на две единицы меньше ранга исходных конъюнкций, при этом полученная конъюнкция не содержит тех переменных, которые по разному входят в исходные конъюнкции. Примеры групп из четырех конъюнкций показаны на рисунке 3.8.



б)

Рис. 3.8. Примеры групп из четырёх конъюнкций:
а) $F(abcd) = cd$; б) $F(abcd) = \bar{a}c$;

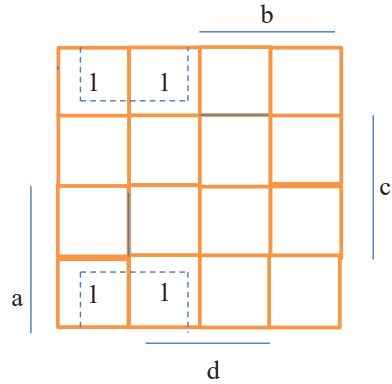


в)

Продолжение рис. 3.8. Примеры групп из четырёх конъюнкций:
в) $F(abcd) = \bar{b}\bar{d}$.

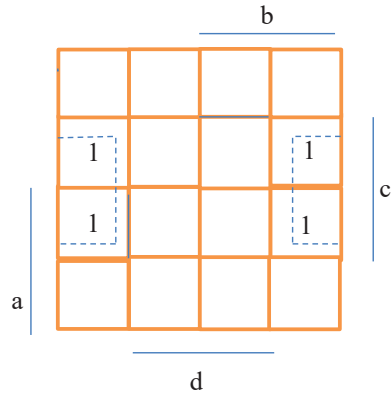
Так как соседними являются также верхняя и нижняя строки, а также левый и правый столбцы, это может быть использовано при объединении конъюнкций в группы так, как это показано на рисунке 3.9.

На одной диаграмме может быть несколько групп из четырех конъюнкций, как это показано на рисунке 3.10.

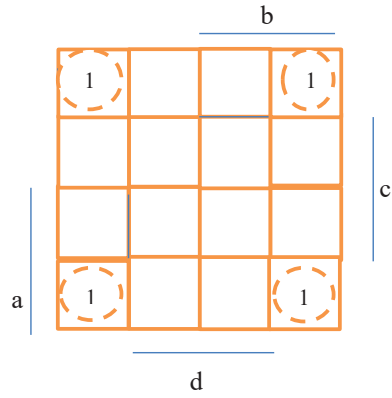


а)

Рис. 3.9. Примеры объединения конъюнкций в группы с учётом соседства верхней и нижней строки, а также левого и правого столбца:
а) $F(abcd) = \bar{b}\bar{c}$;



б)

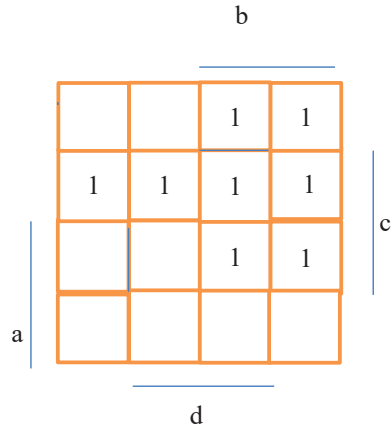


в)

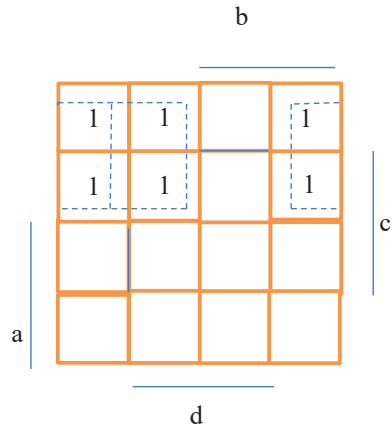
Продолжение рис. 3.9. Примеры объединения конъюнкций в группы с учётом соседства верхней и нижней строки, а также левого и правого столбца:

б) $F(abcd) = c\bar{d}$;

в) $F(abcd) = \bar{c}\bar{d}$.



а)

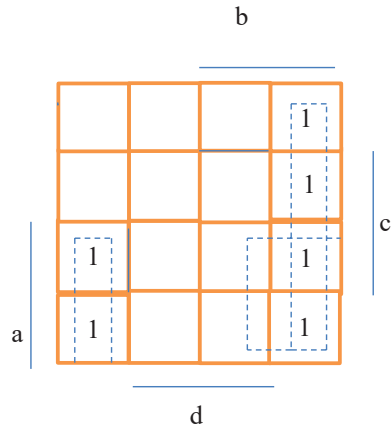


б)

Рис. 3.10. Примеры нескольких групп из четырёх конъюнкций на одной диаграмме:

а) $F(abcd) = \bar{a}c \vee \bar{a}b \vee bc$;

б) $F(abcd) = \bar{a}\bar{b} \vee \bar{a}\bar{d}$;

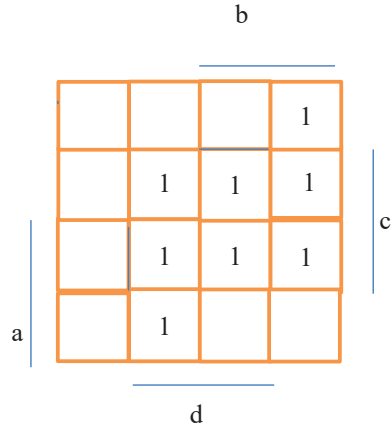


в)

Продолжение рис. 3.10. Примеры нескольких групп из четырёх конъюнкций на одной диаграмме:

в) $F(abcd) = a\bar{d} \vee b\bar{d}$.

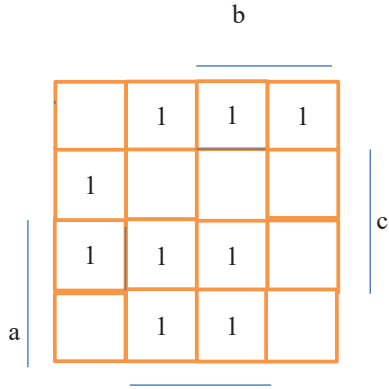
Следует обратить внимание на то, что в группу не может входить шесть конъюнкций, даже если соответствующие клетки образуют прямоугольник. На рисунке 3.11 показаны примеры минимизации для таких случаев.



а)

Рис. 3.11. Примеры минимизации для шести конъюнкций:

а) $F(abcd) = cd \vee bc \vee a\bar{b}d \vee \bar{a}\bar{b}\bar{d}$;



б)

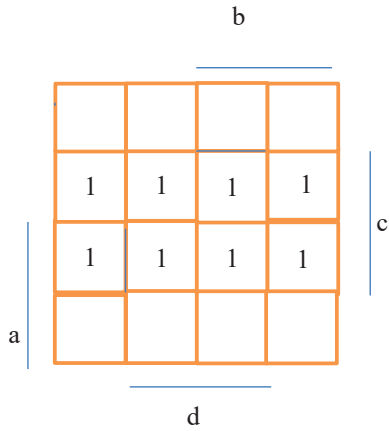
Продолжение рис. 3.11. Примеры минимизации для шести конъюнкций:

а) $F(abcd) = cd \vee bc \vee a\bar{b}d \vee \bar{a}\bar{b}\bar{d}$;

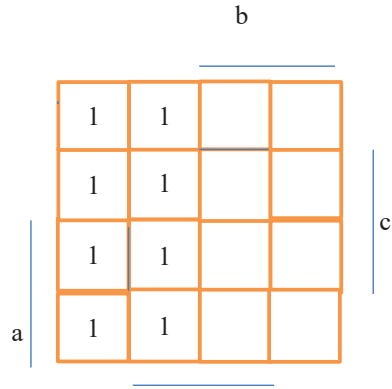
.

При объединении восьми конъюнкций в группу результат минимизации представляет собой одну букву (с инверсией или без нее). Некоторые варианты таких групп приведены на рис. 3.12.

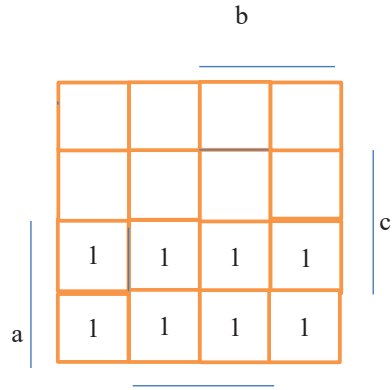
При минимизации на диаграмме могут быть получены различные конфигурации, которые включают группы из различного числа конъюнкций. На рис. 3.13 показаны некоторые примеры логических функций и результаты их минимизации.



а)



б)



в)

Рис. 3.12. Некоторые варианты групп из 8 конъюнкций:

а) $F(abcd) = c$;

б) $F(abcd) = \bar{b}$;

в) $F(abcd) = a$.

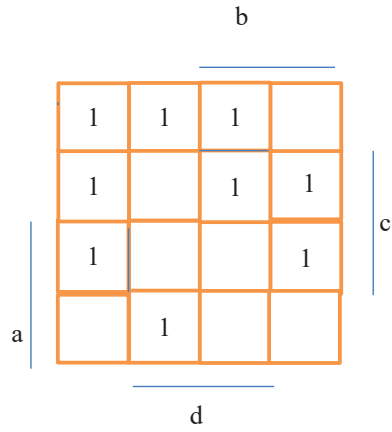


Рис. 3.13. $F(abcd) = c\bar{d} \vee a\bar{b}\bar{c} \vee \bar{a}bd$

В заключение следует отметить, что метод Карно практически можно использовать лишь для минимизации логических функций не более чем четырех переменных. Для минимизации более сложных функций используют метод Квайна или его модификации.

3.5. Метод Квайна.

Метод Квайна является аналитическим точным методом минимизации логических функций. Метод Квайна является базовым методом, который служит основой для других, в частности приближенных, методов минимизации. Общая последовательность минимизации при помощи метода Квайна приведена ранее на рисунке 3.1.

Минимизируемая логическая функция представляется в СДНФ. Из совершенной ДНФ последовательно получают сокращенную ДНФ, а затем все тупиковые формы логической функции. Далее из тупиковых форм выбирается минимальная форма исходной логической функции. Более подробно последовательность действий при использовании метода Квайна может быть описана по шагам.

Шаг 1. В исходной логической функции (или в функции, полученной на предыдущем шаге) выполняются все возможные операции неполного склеивания. Для этого перебираются все пары конъюнкций и проверяются на возможность склеивания. При неполном склеивании конъюнкции, которые участвуют в склеивании, отмечаются.

Шаг 2. Полученная на предыдущем шаге функция проверяется на возможность применения операции неполного склеивания. Если в ней можно выполнить неполное склеивание, то повторяется шаг 1, иначе выполняется шаг 3.

Шаг 3. Выполняется операция поглощения. Так как отмечены все конъюнкции, которые участвовали в склеивании, то операция поглощения сводится к отбрасыванию всех отмеченных конъюнкций. Каждая из оставшихся конъюнкций представляет собой простую импликанту. Поэтому дизъюнкция всех оставшихся на шаге 3 конъюнкций образует сокращенную ДНФ.

Шаг 4. Из сокращенной ДНФ получают все тупиковые формы. Для этого используется так называемая импликантная матрица (метод Петрика).

Шаг 5. Из тупиковых форм по некоторому критерию выбирается минимальная форма логической функции.

Из перечисленных операций наиболее сложными являются определение сокращенной ДНФ исходной функции и нахождение тупиковых форм. Рассмотрим технику выполнения этих операций более подробно.

Для определения сокращенной ДНФ по совершенной ДНФ логической функции необходимо выполнить сначала все возможные операции неполного склеивания, затем - операции поглощения. Техника выполнения этих операций при минимизации функций без использования ЭВМ может быть различной.

Представляется удобным следующий порядок определения сокращенной ДНФ.

Конъюнкции исходной логической функции записываются в столбец. Далее последовательно проверяется возможность склеивания первой конъюнкции со всеми остальными, второй со всеми остальными и т. д. Каждая конъюнкция может участвовать в склеивании неоднократно. Результаты склеивания записываются в новый столбец. При записи результата склеивания целесообразно вместо выпавшей логической переменной записывать символ «-» (тире). Конъюнкции, которые хотя бы один раз участвовали в склеивании, отмечаются, например, символом (*). После склеивания всех конъюнкций первого столбца аналогичные операции выполняются со вторым столбцом. Операции склеивания выполняются до тех пор, пока в очередном столбце их будет нельзя применить. Если при склеивании в столбце получено несколько одинаковых конъюнкций, из них для дальнейшего использования оставляется только одна. Сокращенная ДНФ получается как дизъюнкция всех неотмеченных конъюнкций (простых импликант).

Рассмотрим пример определения сокращенной ДНФ. Пусть задана логическая функция в СДНФ, для которой необходимо найти сокращенную ДНФ:

$$F(abcd) = \bar{a}\bar{b}\bar{c}\bar{d} \vee \bar{a}\bar{b}\bar{c}d \vee \bar{a}\bar{b}c\bar{d} \vee \bar{a}\bar{b}cd \vee \bar{a}b\bar{c}\bar{d} \vee \bar{a}bc\bar{d} \vee \bar{a}bcd \vee ab\bar{c}\bar{d} \vee abc\bar{d} \quad (3.14)$$

Порядок нахождения сокращённой ДНФ приведён ниже:

$\bar{a}\bar{b}\bar{c}\bar{d}$	*	$\bar{a}\bar{b}\bar{c}-$	$--\bar{c}\bar{d}$
$\bar{a}\bar{b}\bar{c}d$	*	$\bar{a}\bar{b}-\bar{d}$	$--\bar{c}\bar{d}$
$\bar{a}\bar{b}c\bar{d}$	*	$\bar{a}-c\bar{d}$ *	
$\bar{a}\bar{b}cd$	*	$\bar{a}-\bar{c}d$	
$\bar{a}bc\bar{d}$	*	$-\bar{b}c\bar{d}$ *	
$\bar{a}bcd$	*	$\bar{a}b-d$	
$a\bar{b}c\bar{d}$	*	$\bar{a}bc-$	
$abc\bar{d}$	*	$-bc\bar{d}$ *	
		$a-c\bar{d}$ *	

В последнем столбце получены две одинаковые конъюнкции, одну из которых отбрасываем.

Неотмеченные конъюнкции составляют сокращённую ДНФ исходной логической функции:

$$F(abcd)=\bar{a}\bar{b}\bar{c}\vee\bar{a}\bar{b}\bar{d}\vee\bar{a}\bar{c}d\vee\bar{a}bd\vee\bar{a}bc\vee c\bar{d} \qquad (3.15)$$

Для большей наглядности при определении соседних конъюнкций представляется целесообразным записывать вместо исходных конъюнкций их двоичные коды. Например, последовательность получения сокращенной ДНФ логической функции (3.16), представленной в СДНФ, может быть записана в следующем виде:

$$F(abcd)=\overset{0}{\bar{a}\bar{b}\bar{c}\bar{d}}\vee\overset{1}{\bar{a}\bar{b}\bar{c}d}\vee\overset{2}{\bar{a}\bar{b}c\bar{d}}\vee\overset{5}{\bar{a}b\bar{c}d}\vee\overset{6}{\bar{a}bc\bar{d}}\vee\overset{7}{\bar{a}bcd}\vee\overset{9}{a\bar{b}\bar{c}d}\vee\overset{12}{ab\bar{c}\bar{d}}\vee\overset{13}{ab\bar{c}d}\vee\overset{14}{abc\bar{d}} \qquad (3.16)$$

0000*	000-	--01
0001*	00-0	--01
0010*	0-01*	
0101*	-001*	
0110*	0-10	
0111*	01-1	
1001*	-101*	

1100* 011-

1101* -110

1110* 1-01*

110-

11-0

В результате получим сокращённую ДНФ логической функции (3.17):

$$F(abcd) = \bar{a}\bar{b}\bar{c} \vee \bar{a}\bar{b}\bar{d} \vee \bar{a}\bar{c}\bar{d} \vee abd \vee \bar{a}bc \vee b\bar{c}\bar{d} \vee ab\bar{c} \vee ab\bar{d} \vee c\bar{d}. \qquad (3.17)$$

Для определения тупиковых форм логической функции по сокращенной ДНФ этой функции используются импликантные матрицы. Импликантная матрица представляет собой таблицу, столбцы которой отмечены конъюнкциями исходной логической функции, а строки - простыми импликантами, Пример импликантной матрицы для сокращенной ДНФ (3.17) приведен в виде таблицы 3.2. В строках матрицы помечаются клетки, находящиеся в столбцах, отмеченных исходными конъюнкциями, из которых образована простая импликанта строки. Так как склеиваться могут 2, 4, 8 и т.д. конъюнкций, то в строке может быть 2, 4, 8 и т.д. отметок.

Тупиковые формы находятся при отбрасывании лишних импликант из сокращенной ДНФ. Для этого необходимо решить задачу покрытия всех столбцов матрицы отметками некоторой совокупности строк, являющейся подмножеством всех строк матрицы. Такая совокупность строк, не содержащая лишних импликант, является тупиковой формой исходной логической функции. Тупиковые формы можно определить методом подбора, однако при этом не гарантируется нахождение всех тупиковых форм, а, следовательно, и минимальной формы логической функции

Таблица 3.2

Пример импликантной матрицы для сокращённой ДНФ

Простые импликанты		Исходные конъюнкции									
		0	1	2	5	6	7	9	12	13	14
		$\bar{a}\bar{b}\bar{c}\bar{d}$	$\bar{a}\bar{b}\bar{c}d$	$\bar{a}\bar{b}c\bar{d}$	$\bar{a}b\bar{c}d$	$\bar{a}bc\bar{d}$	$\bar{a}bcd$	$a\bar{b}\bar{c}d$	$ab\bar{c}\bar{d}$	$ab\bar{c}d$	$abc\bar{d}$
A	$\bar{a}\bar{b}\bar{c}$	*	*								
B	$\bar{a}\bar{b}\bar{d}$	*		*							
C	$\bar{a}c\bar{d}$			*		*					

D	$\bar{a}bd$				*		*				
E	$\bar{a}bc$					*	*				
F	$bc\bar{d}$					*					*
H	$ab\bar{c}$								*	*	
K	$ab\bar{d}$								*		*
L	$\bar{c}d$		*		*			*		*	

Для решения задачи покрытия и определения всех тупиковых форм исходной функции используется метод Петрика. Сущность метода Петрика заключается в следующем. Строки матрицы отмечаются, например, прописными буквами латинского алфавита. Далее составляется логическая функция, описывающая условие покрытия всех столбцов отметками. Для сокращенной ДНФ (3.17) условие покрытия всех столбцов будет записано в следующем виде:

$$P = (A \vee B)(A \vee L)(B \vee C)(D \vee L)(C \vee E \vee F)(D \vee E)L(H \vee K)(H \vee L)(F \vee K) \quad (3.18)$$

Из выражения (3.18) следует, что если в каком-либо столбце имеется единственная отметка, то соответствующая этой отметке импликанта должна обязательно входить в каждую тупиковую форму. Для данной функции такой импликантой является импликанта L.

После выполнения операции поглощения (например, $L(H \vee L) = L$) уравнение (3.18) принимает следующий вид:

$$P = (A \vee B)(B \vee C)(C \vee E \vee F)(D \vee E)(H \vee K)(F \vee K)L. \quad (3.19)$$

Затем в уравнении (3.19) необходимо раскрыть скобки и выполнить все операции поглощения. При этом предварительно следует попарно сгруппировать скобки так, чтобы в соседних скобках было по одной одинаковой букве. Это позволит применить операцию поглощения при раскрытии каждой пары скобок и упростить дальнейшие действия. Например:

$$P = (A \vee B)(B \vee C)(C \vee E \vee F)(D \vee E)(H \vee K)(F \vee K)L.$$

При последовательном раскрытии скобок в выражении (3.19) получим:

$$P = (B \vee AC)(E \vee CD \vee DF)(K \vee FH) = (BE \vee BCD \vee BDF \vee ACE \vee ACD)(K \vee FH)L =$$

$$= BEKL \vee BCDKL \vee BDFKL \vee ACEKL \vee ACDKL \vee BEFHL \vee BDFHL \vee$$

$$\vee ACEFHL \vee ACDFHL. \quad (3.20)$$

Каждая из конъюнкций выражения (3.20) соответствует одной из тупиковых форм исходной логической функции. Так, например, конъюнкция BEKL задает тупиковую форму:

$$F(abcd)_{\text{туп1}} = \bar{a}\bar{b}\bar{d} \vee \bar{a}bc \vee ab\bar{d} \vee \bar{c}d, \qquad (3.21)$$

а конъюнкция ACDFHL – тупиковую форму

$$F(abcd)_{\text{туп9}} = \bar{a}\bar{b}\bar{c} \vee \bar{a}c\bar{d} \vee \bar{a}bd \vee b\bar{c}\bar{d} \vee ab\bar{c} \vee \bar{c}\bar{d}. \qquad (3.22)$$

Из выражения (3.19) следует, что логическая функция (3.17) имеет 9 тупиковых форм различной сложности (тупиковые формы включают от 4 до 6 простых импликант разного ранга).

При выборе минимальной формы из тупиковых форм могут использоваться различные критерии. Чаще всего используют критерий Квайна. Для тупиковых форм (3.21) и (3.22) значение критерия составляет 15 и 23 соответственно. В данном случае минимальной является тупиковая форма (3.21):

$$F(abcd)_{\text{мин}} = F(abcd)_{\text{туп1}} = \bar{a}\bar{b}\bar{d} \vee \bar{a}bc \vee ab\bar{d} \vee \bar{c}d.$$

По минимальной форме логической функции можно построить комбинационную схему из логических элементов заданного типа.

4. Частные случаи комбинационных схем

4.1. Особенности синтеза схем на интегральных элементах

Комбинационная схема является ***частично определенной***, если значение выходного сигнала задано не для всех комбинаций входных сигналов. Такие схемы, например, используются при операциях с двоично-десятичными кодами.

Двоично-десятичные коды предназначены для кодирования десятичных цифр. Максимальное значение десятичной цифры равно 9, поэтому для кодирования любой десятичной цифры достаточно четырех двоичных разрядов (9_{10} —> 1001_2). При естественном кодировании каждой десятичной цифре соответствует двоичное число, причем значения десятичной цифры и двоичной тетрады совпадают (0_{10} -> 0000_2, 1_{10} -> 0001_2 и т.д.).

Количество различных комбинаций четырехразрядного двоичного кода равно 16, из которых для кодирования десятичных цифр используются только 10, а остальные 6 являются запрещенными. При естественном кодировании запрещенными являются комбинации 1010,1011,..., 1111.

Недостатком естественного кодирования десятичных цифр является то, что при таком кодировании арифметические операции над двоично-десятичными числами выполняются довольно сложно.

При синтезе комбинационных схем на интегральных элементах возникает необходимость перехода от системы элементов базиса И, ИЛИ, НЕ к другому базису. Это обусловлено тем, что интегральные элементы выполняются обычно с использованием логики И-НЕ или ИЛИ-НЕ. В то же время после минимизации логические функции записаны в дизъюнктивной форме, требующей для реализации элементов И, ИЛИ, НЕ.

Для перехода от одного базиса к другому используется правило инверсии, которое может быть записано в следующем виде:

$$\overline{a \vee b} = \bar{a}\bar{b};\ \overline{a \wedge b} = \bar{a} \vee \bar{b} \qquad (4.1)$$

Можно показать, что из элементов И-НЕ (ИЛИ-НЕ) можно построить все элементы базиса И, ИЛИ, НЕ.

Так при использовании элементов И-НЕ для этого можно использовать следующие преобразования:

$$\bar{a} = \overline{aa};\ ab = \overline{\overline{ab}};\ a \vee b = \overline{\overline{a \vee b}} = \overline{\bar{a}\bar{b}}. \qquad (4.2)$$

Способы реализации функций НЕ, И, ИЛИ на элементах И-НЕ показаны на рис. 4.1.

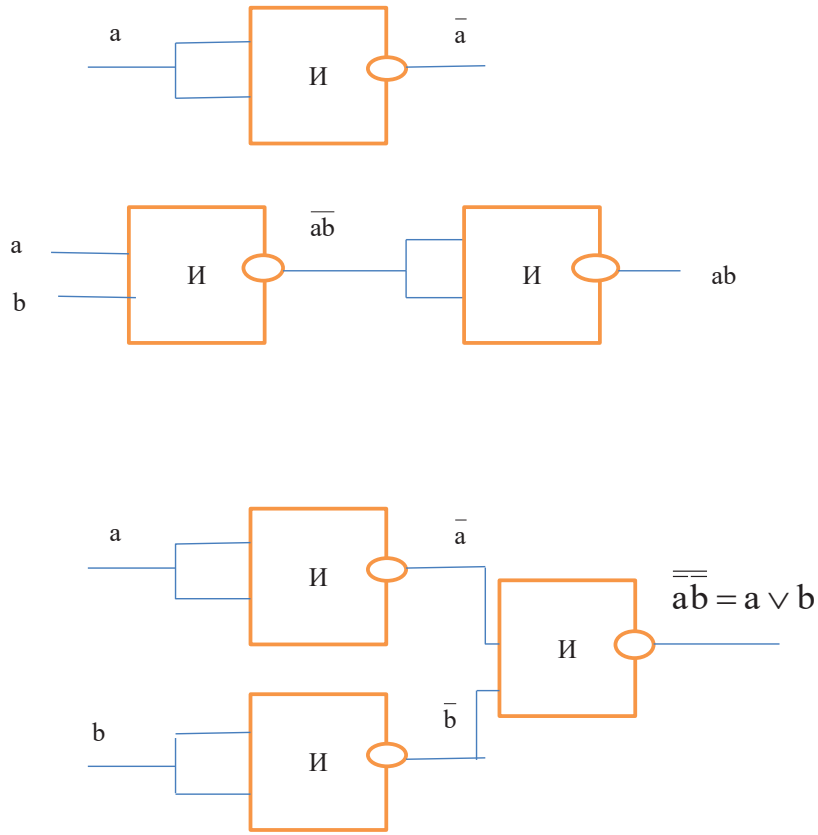


Рис. 4.1. Способы реализации функций НЕ, И, ИЛИ на элементах И-НЕ

Для элементов ИЛИ-НЕ аналогичные преобразования имеют следующий вид:

$$\bar{a} = \overline{a \vee a};\ ab = \overline{\overline{ab}} = \overline{\bar{a} \vee \bar{b}};\ a \vee b = \overline{\overline{a \vee b}} \qquad (4.2)$$

Способы реализации функций НЕ, И, ИЛИ на элементах ИЛИ-НЕ показаны на рис. 4.2.

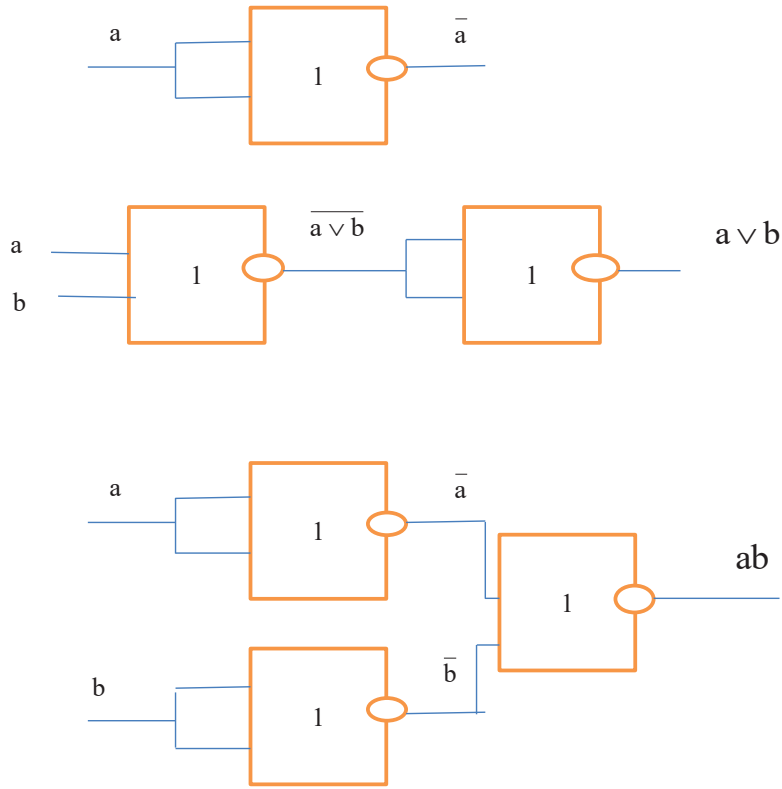


Рис. 4.2. Способы реализации функций НЕ, И, ИЛИ на элементах ИЛИ-НЕ

Если исходная логическая функция представлена в дизъюнктивной форме (в виде дизъюнкции элементарных конъюнкций), то для перехода к базису И-НЕ над этой функцией выполняется операция двойной инверсии с последующим применением правила инверсии. Например:

$$F(abcd) = a\bar{b}c \vee \bar{a}bc \vee b\bar{c} \vee \bar{d} = \overline{\overline{a\bar{b}c \vee \bar{a}bc \vee b\bar{c} \vee \bar{d}}} = \overline{\overline{a\bar{b}c} \wedge \overline{\bar{a}bc} \wedge \overline{b\bar{c}} \wedge d} \qquad (4.3)$$

Логической функции (4.3) соответствует функциональная схема, представленная на рисунке 4.3.

Из рисунка 4.3 видно, что схема, выполненная на элементах И-НЕ, состоит из двух ярусов. Первый ярус включает элементы И-НЕ, реализующие конъюнкции исходной логической функции. Число этих

элементов равно числу конъюнкций. Число входов каждого элемента равно рангу соответствующей конъюнкции. Второй ярус состоит из одного элемента И-НЕ с числом входов, равным сумме числа конъюнкций и одиночных букв в исходной форме логической функции

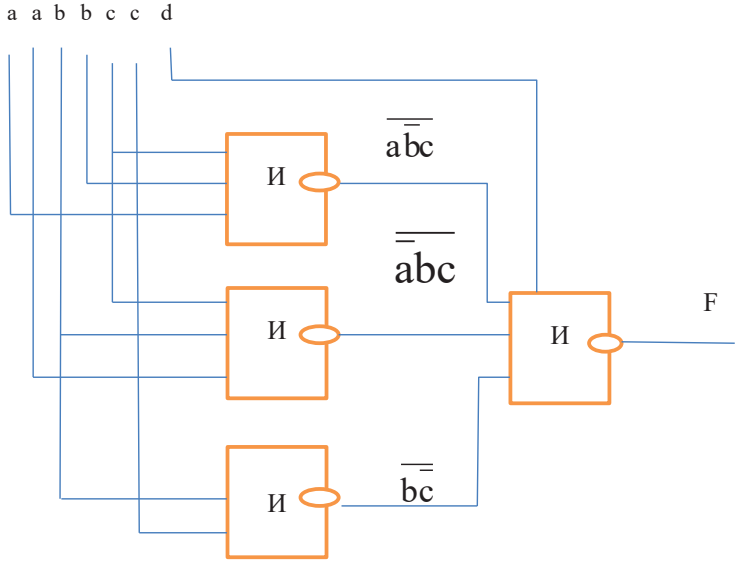


Рис. 4.3. Функциональная схема логической функции F(abcd)

Для перехода к базису ИЛИ-НЕ применим операцию двойного отрицания ко всей функции и к её отдельным конъюнкциям:

$$F(abcd) = a\bar{b}c \vee \bar{a}bc \vee b\bar{c} \vee \bar{d} = \overline{\overline{a\bar{b}c} \vee \overline{\bar{a}bc} \vee \overline{b\bar{c}} \vee \overline{\bar{d}}} = \overline{\overline{(\bar{a} \vee b \vee \bar{c})} \vee \overline{(a \vee \bar{b} \vee c)} \vee \overline{(\bar{b} \vee c)} \vee \overline{\bar{d}}} \quad (4.4)$$

Логической функции (4.4) соответствует функциональная схема, состоящая из трёх ярусов. Элемент третьего яруса представляет собой инвертор, выполненный на элементе ИЛИ-НЕ.

4.2. Особенности синтеза схем с несколькими выходами

Комбинационную схему с несколькими выходами можно рассматривать как несколько отдельных схем с одним выходом. В этом случае каждая такая схема синтезируется отдельно. Общая схема составляется путем простого объединения отдельных схем. При этом количество элементов общей комбинационной схемы равно сумме всех

элементов отдельных схем. Для упрощения комбинационных схем с несколькими выходами используются два приема.

Первый прием является вполне очевидным и заключается в выделении одинаковых частей логических функций, описывающих логику работы схем с одним выходом. Элементы, реализующие такие одинаковые части нескольких функций, являются общими для соответствующих схем с одним выходом. Пусть, например, при синтезе схемы с двумя выходами получены следующие логические функции:

$$F_1(abcd) = a\bar{b}d \vee bcd \vee a\bar{b}\bar{c}; \qquad (4.5)$$

$$F_2(abcd) = abc \vee acd \vee a\bar{b}d.$$

Комбинационные схемы могут быть выполнены на элементах И, ИЛИ, НЕ так, как это показано на рис. 4.4.

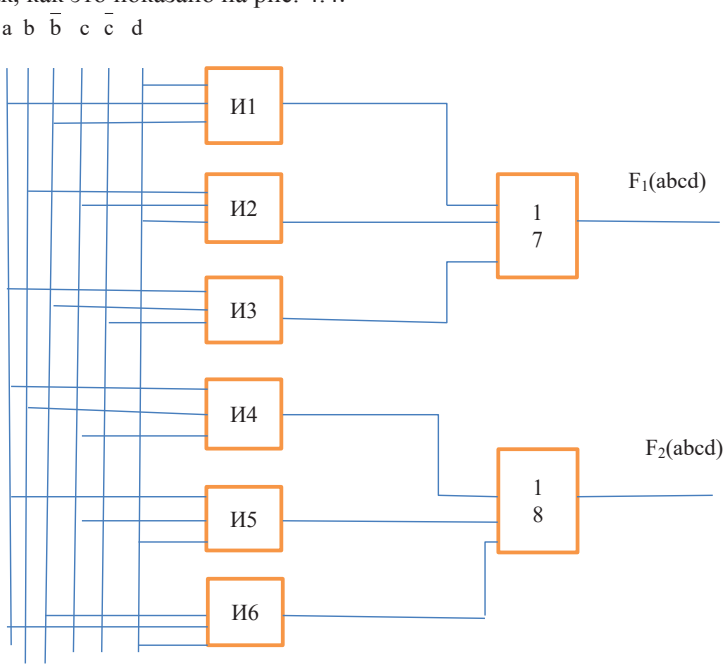


Рис. 4.4.. Выполнение комбинационной схемы на элементах И, ИЛИ, НЕ.

Если учесть, что элементы 1 и 6 на схеме рис. 4.4. реализуют одну и туже логическую функцию, то схему можно упростить так, как это показано на рис. 4.5.

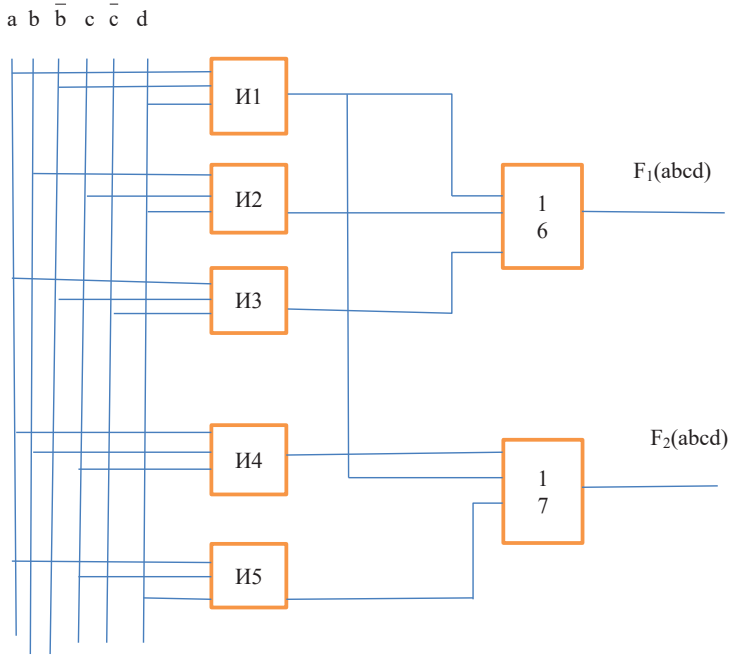
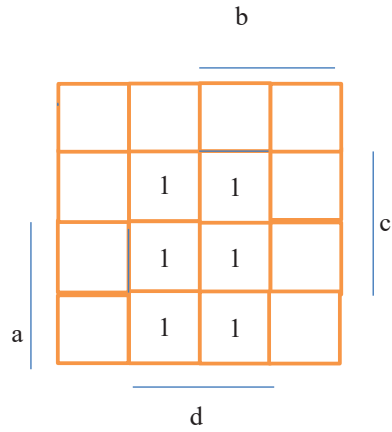


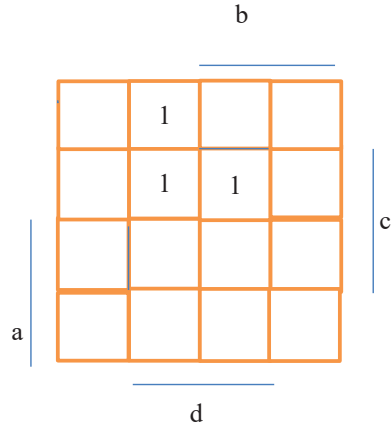
Рис. 4.5. Упрощённый вид комбинационной схемы

Второй прием упрощения схем с несколькими выходами заключается в том, что минимизация отдельных логических функций проводится так, чтобы искусственно образовать общие части в нескольких функциях. При этом иногда приходится не упрощать логическую функцию, а останавливаться на некоторой промежуточной, не минимальной форме логической функции. Понятно, что это следует делать лишь тогда, когда за счет искусственного образования общих частей нескольких функций удается упростить комбинационную схему в целом.

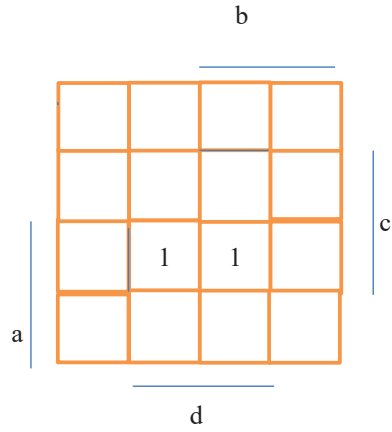
Пусть, например, при синтезе схемы с четырьмя выходами получены диаграммы Карно, показанные на рисунке 4.6.



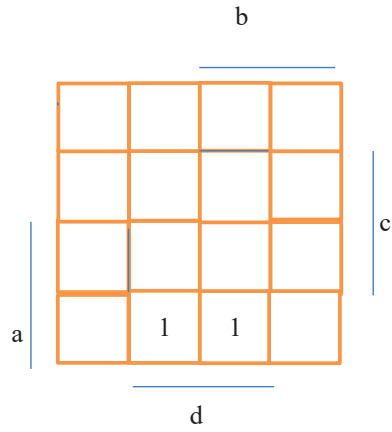
$F_1(abcd)$



$F_2(abcd)$



$F_3(abcd)$



$F_4(abcd)$

Рис. 4.6. Диаграммы Карно, полученные для схемы с четырьмя выходами

При раздельной минимизации будут получены следующие уравнения:

$$F_1(abcd) = ad \vee cd;\ F_2(abcd) = \bar{a}cd \vee a\bar{b}\bar{d}; \qquad (4.6)$$

$F_3(abcd) = acd;\ F_4(abcd) = a\bar{c}d.$

По уравнениям (4.6) можно составить комбинационную схему, показанную на рис. 4.7.

a $\bar{a}$ $\bar{b}$ c $\bar{c}$ d

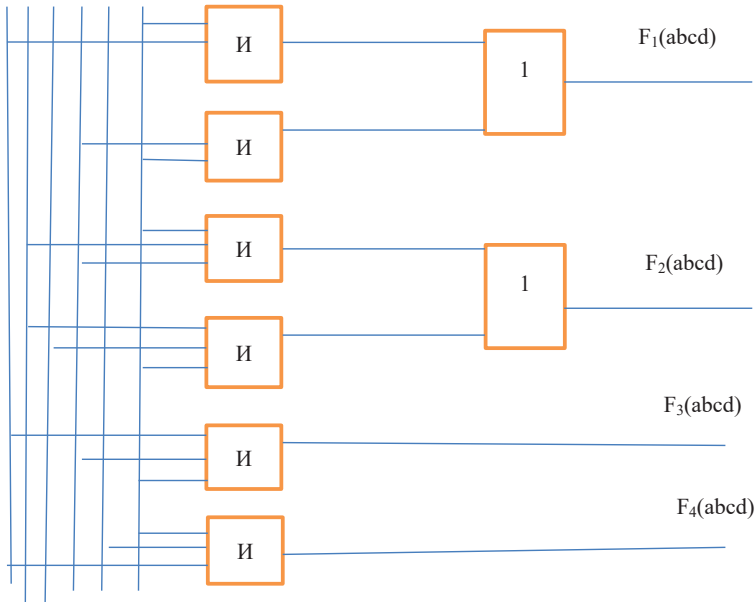


Рис. 4.7. Комбинационная схема, составленная по уравнениям (4.6.)

Если учесть, что логические функции (4.6) образуют систему функций с одинаковыми переменными, и выполнить минимизацию для функции F_1 иначе (рис. 4.8), то результат минимизации можно записать в следующем виде:

$F_1(abcd) = \bar{a}cd \vee a\bar{c}d \vee acd;$ $F_2(abcd) = \bar{a}cd \vee \bar{a}\bar{b}d;$ $F_3(abcd) = acd;$

$F_4(abcd) = a\bar{c}d.$ (4.7)

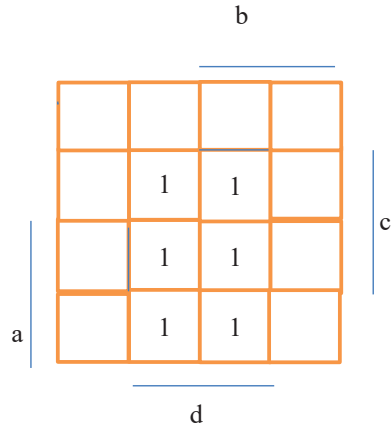


Рис. 4.8. Результат минимизации в виде диаграммы Карно

Логическим функциям (4.7) соответствует комбинационная схема, приведённая на рис. 4.9.

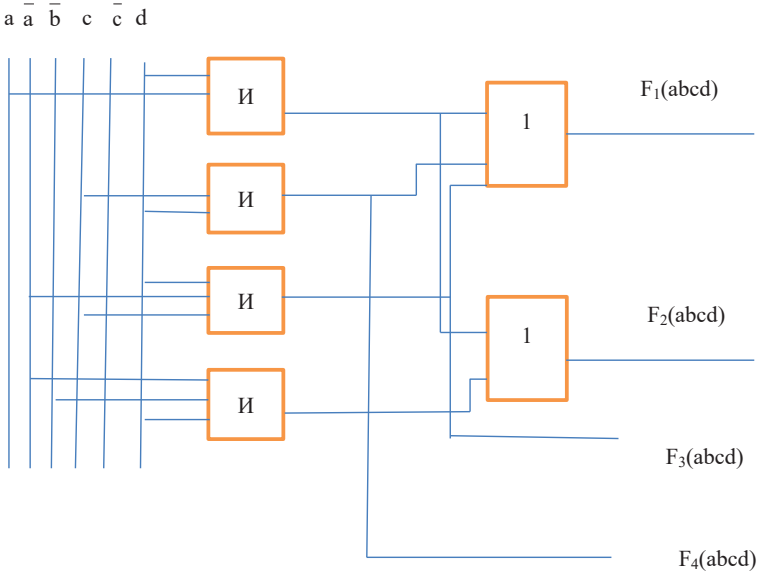


Рис. 4.9. Комбинационная схема, соответствующая (4.7)

Сравнение схем на рисунках 4.7 и 4.9 показывает, что несмотря на усложнение схемы реализации функции $F_i(abcd)$ удалось упростить общую комбинационную схему с четырьмя выходами.

Отметим, что упрощение схем с несколькими выходами требует применения специальных методов минимизации системы логических функций.

4.3. Скобочные формы логических функций

Перед составлением схемы целесообразно проверить возможность ее упрощения за счет использования скобочных форм. Скобочные формы в некоторых случаях позволяют уменьшить сложность схемы. Например, если задана функция $F(abc) = abc \vee \bar{a}\bar{b}\bar{c}$, то её можно представить в следующей форме:

$$F(abc) = b(ac \vee \overline{\bar{a}\bar{c}}).$$

Варианты реализации этих форм функции представлены на рис. 4.10

.

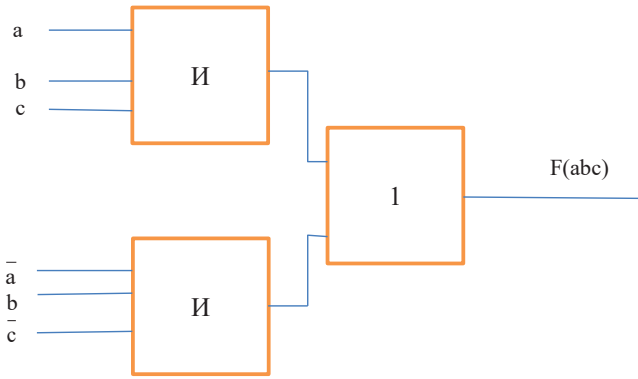


Рис. 4.10. Варианты реализации форм функции F(abc).

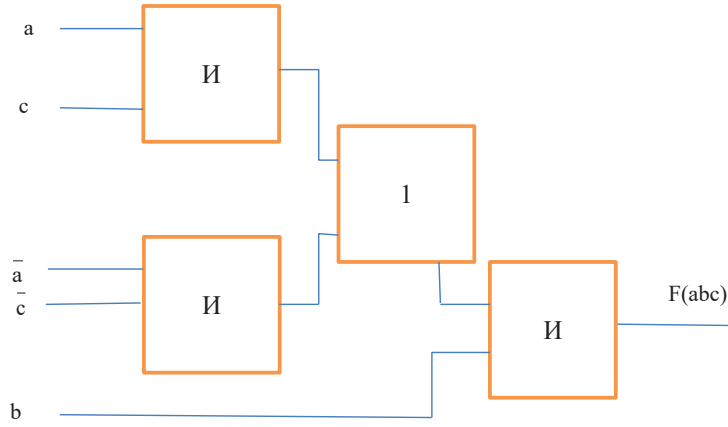


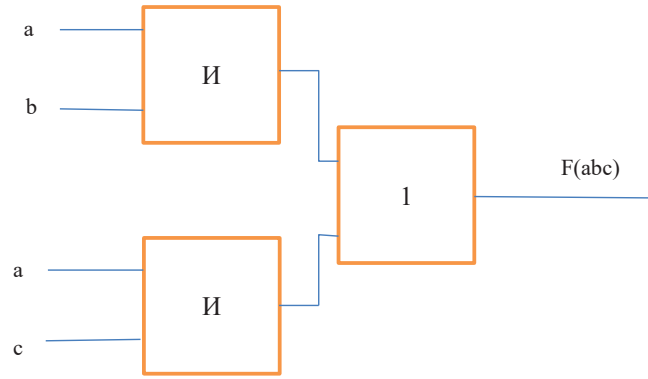
Рис. 4.10. Варианты реализации форм функции F(abc).

Как видно на рисунке 4.10, в данном случае переход к скобочной форме не уменьшает общее количество входов элементов. Кроме того, при этом увеличивается количество элементов в цепи прохождения входных сигналов, что приводит к увеличению времени их задержки.

Если задана функция $F(abc) = ab \vee ac$, то её можно представить в следующей форме:

$$F(abc) = a(b \vee c).$$

Две формы этой функции могут быть реализованы так, как показано на рис. 4.11.



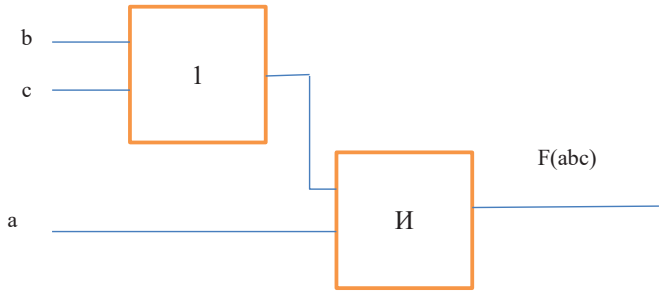


Рис. 4.11. Две формы функции F(abc)

Для заданной функции, как видно на рисунке 4.11, переход к скобочной форме позволил уменьшить общее количество входов элементов и число элементов без увеличения времени задержки сигналов.

Переход к скобочной форме можно выполнить с помощью ***факторного алгоритма,*** суть которого рассмотрим на примере.

Пусть задана некоторая логическая функция в виде:

$$Y = X_1X_2X_3\overline{X_4} \vee X_1X_2X_3X_5X_6 \vee X_1\overline{X_3}\,\overline{X_5}\,\overline{X_6} \vee X_1\overline{X_2}X_3X_4 .$$

Для реализации этой функции по приведенному выражению необходимо использовать 3 логических элемента 4И, один логический элемент 5И, один логический элемент 4ИЛИ. При этом суммарное число входов элементов составляет 21.

С помощью факторного алгоритма получим скобочную форму для заданной функции. Для этого обозначим все конъюнкции буквами:

$$A = X_1X_2X_3\overline{X_4},\ B = X_1X_2X_3X_5X_6,\ C = X_1\overline{X_3}\,\overline{X_5}\,\overline{X_6},\ D = X_1\overline{X_2}X_3X_4,$$

и будем рассматривать их как некоторые множества. Находим попарные пересечения множеств:

$$A \cap B = X_1X_2X_3,\quad A \cap C = X_1,\quad A \cap D = X_1X_3,\quad B \cap C = X_1,\quad B \cap D = X_1X_3,$$
$$C \cap D = X_1.$$

Полученные пересечения показывают общие части отдельных конъюнкций. Выбираем пересечение, которое имеет наибольшую длину (если такое отсутствует, то выбирают то, которое чаще всего встречается). В данном случае это $A \cap B = X_1X_2X_3$. Поэтому из конъюнкций A и B выносим общую часть $X_1X_2X_3$. Тогда имеем:

$$Y = X_1X_2X_3(\overline{X_4} \vee X_5X_6) \vee X_1\overline{X_3}\,\overline{X_5}\,\overline{X_6} \vee X_1\overline{X_2}X_3X_4 .$$

Обозначим $F = X_1X_2X_3(\overline{X_4} \vee X_5X_6)$ и находим пересечения:

$$F \cap C = X_1,\ F \cap D = X_1X_3,\ C \cap D = X_1.$$

Следовательно, для исходной функции имеем:

$Y = X_1X_3[X_2(\overline{X_4} \vee X_5X_6) \vee \overline{X_2}X_4] \vee X_1\overline{X_3}\,\overline{X_5}\,\overline{X_6}$.

Обозначим $E = X_1X_3[X_2(\overline{X_4} \vee X_5X_6) \vee \overline{X_2}X_4]$.

Пересечение $E \cap C = X_1$. Следовательно, окончательно имеем:

$Y = X_1\{X_3[X_2(\overline{X_4} \vee X_5X_6) \vee \overline{X_2}X_4] \vee \overline{X_3}\,\overline{X_5}\,\overline{X_6}\}$.

Для реализации функции по последнему выражению необходимо 5 элементов 2И, 1 элемент 3И, 3 элемента 2ИЛИ (рисунок 4.12). При этом суммарное число входов элементов составляет 19, т.е. сложность схемы уменьшилась по сравнению со схемой, реализующей исходную форму функции.

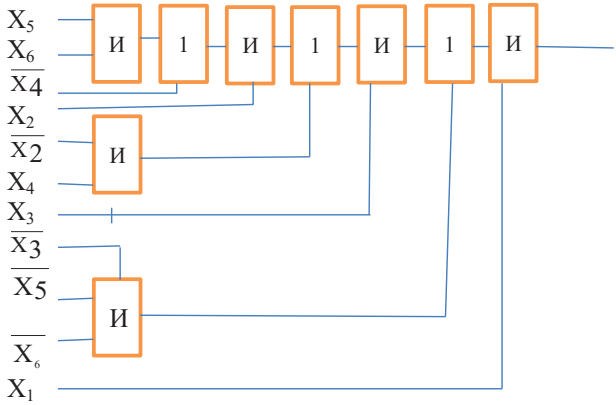


Рис. 4.12. Функциональная схема реализации функции

4.4. Синтез схем на элементах с ограниченных числом входов

При большом количестве переменных для реализации логических функций необходимо использовать элементы с соответствующим числом входов. Реально логические элементы могут иметь не более 8-10 входов. Поэтому необходимо уметь строить комбинационные схемы при наличии логических элементов с ограниченным числом входов. Для учета ограничений на число входов элементов необходимо преобразовать форму логической функции, применяя сочетательный закон алгебры логики.

Например, пусть задана функция

$F(abcd) = \bar{a}\bar{b}cd \vee \bar{b}c\bar{d} \vee ac$.

Она может быть реализована в виде схемы 1, приведённой на рис. 4.13.

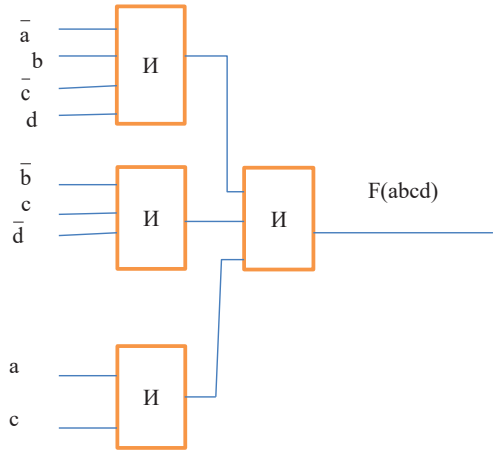


Рис. 4.13. Схема 1.

В схеме используются элементы на 2, 3 и 4 входа. Если в наличии имеются элементы только на 2 входа, исходная функция преобразуется следующим образом:

$$F\ (abcd) = \bar{a}b\bar{c}d \vee \bar{b}c\bar{d} \vee ac = ((\bar{a}b)(\bar{c}d) \vee (\bar{b}c)\bar{d}) \vee ac\,.$$

После преобразования логическая функция реализуется в виде схемы 2, приведённой на рис. 4.14.

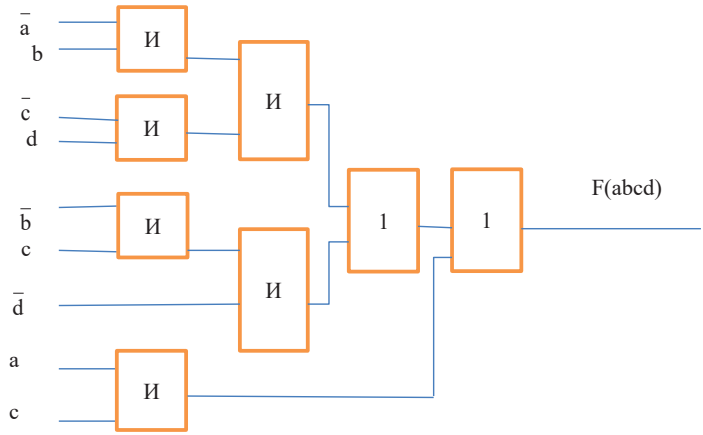


Рис. 4.14. Схема 2.

Во второй схеме используются только элементы на 2 входа. При этом, по сравнению с первой схемой, увеличивается как сложность схемы, так и время задержки сигнала в схеме.

При построении схем на элементах И-НЕ с двумя входами преобразования логических функций проводятся с применением закона двойной инверсии и имеют следующий вид:

$$F(abcd) = \bar{a}\bar{b}cd \vee \bar{b}c\bar{d} \vee ac = \overline{\overline{(\overline{ab})(cd)} \vee \overline{(\overline{bc})\overline{d}}} \vee ac = \overline{\overline{(\overline{ab})(cd)} \vee \overline{(\overline{bc})\overline{d}}} \wedge \overline{ac} =$$

$$= \overline{\overline{(\overline{ab})} \wedge \overline{(cd)} \wedge \overline{(\overline{bc})\overline{d}}} \wedge ac = \overline{\overline{(\overline{ab})(cd)} \wedge \overline{(\overline{bc})d} \wedge \overline{ac}} = \overline{\overline{(\overline{ab})cd} \wedge \overline{(\overline{bc})d} \wedge \overline{ac}}.$$

Преобразования сводятся к тому, что над каждой конъюнкцией должен быть, по крайней мере, один знак инверсии. При реализации логической функции на схеме появляются дополнительные инверторы, выполненные на элементах И-НЕ. Реализация заданной функции на элементах И-НЕ с двумя входами представлена в виде схемы на рис. 4.15.

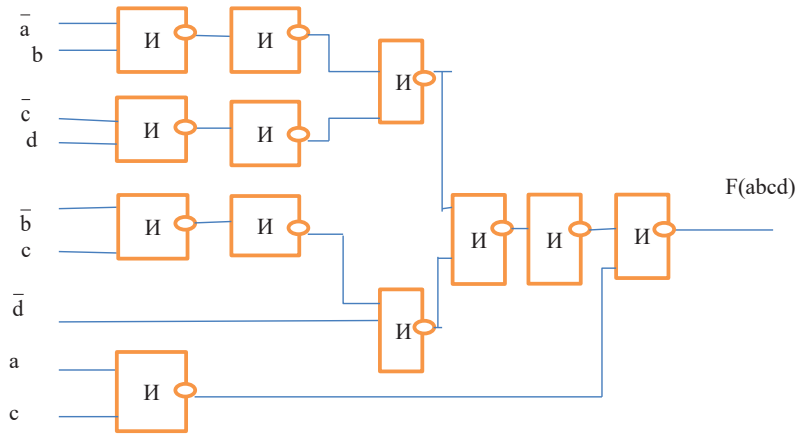


Рис. 4.15. Реализация функции F(abcd) на элементах И-НЕ

Эта схема сохраняет структуру предыдущей схемы, но по сравнению с ней содержит дополнительные инверторы.

5. Анализ комбинационных схем

5.1. Методы анализа комбинационных схем

Задачи анализа комбинационных схем (КС) возникают при необходимости проверить правильность синтеза (на этапе проектирования) или определить логическую функцию, реализуемую КС (при анализе или ремонте схем). Все существующие методы анализа делятся на ***прямые*** и ***косвенные.***

В результате анализа КС ***прямым*** методом получается множество

наборов входных переменных, обеспечивающих заданное значение на выходе, что позволяет записать в алгебраическом виде логическую функцию, реализуемую схемой. К прямым методам относится метод π- алгоритма.

Применение ***косвенных*** методов даёт возможность определить реакцию схемы на заданный набор входных переменных в статике или проанализировать переходный процесс смены одного входного набора на другой. Примерами косвенных методов анализа являются методы синхронного и асинхронного моделирования.

Все упомянутые методы анализа являются машиноориентированными, что позволяет выполнить анализ схемы на ЭВМ.

Для всех методов анализа необходимо описать схему в виде схемного списка, в который включаются в общем случае следующие данные:

- номер логического элемента в схеме;
- логическая функция, реализуемая логическим элементом;
- входные переменные для данного логического элемента.

Например, на рис. 5.1. представлена схема, а в таблице 5.1. список, описывающий схему.

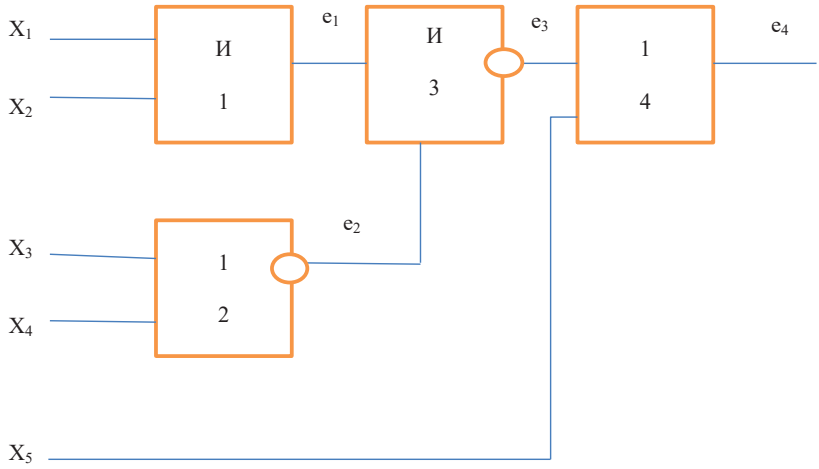


Рис. 5.1. Схема для метода анализа

Таблица 5.1

Схемный список

Номер элемента	Функция	Входы
1	2И	X_1 X_2
2	2ИЛИ-НЕ	X_3 X_4
3	2И-НЕ	e_1 e_2
4	2 ИЛИ	e_3 X_5

5.2. Метод π- алгоритма. Нулевые и единичные покрытия логических функций.

При анализе комбинационных схем методом л-алгоритма определяются наборы входных переменных, обеспечивающих заданное значение выходного сигнала КС. Наборы, обеспечивающие на выходе КС логическую 1, образуют так называемое ***единичное покрытие*** C^1 . Аналогично входные наборы, обеспечивающие на выходе КС логический 0, образуют ***нулевое покрытие*** C°. Рассмотрим покрытия C° и C^1 для простейшего логического элемента 2И, выполняющего функцию $Y=X_1X_2$. Таблица истинности для этой функции и условное обозначение элемента 2И приведены на рисунке 5.2.

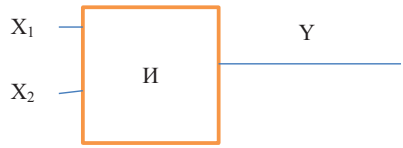


Рис. 5.2. Условное обозначение элемента 2И

Таблица 5.2

Таблица истинности элемента 2И

X_1	X_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

Как видно из приведённой таблицы только при единственном наборе $X_1=1$ и $X_2=1$ на выходе логического элемента будет 1, т.е. единичное покрытие включает только один набор $C^1= \{1\ 1\}$. На выходе логического элемента будет 0 при трёх наборах, образующих нулевое покрытие:

$$C^0 = \begin{bmatrix} 00 \\ 01 \\ 10 \end{bmatrix}.$$

Это покрытие можно упростить, заметив, что первый набор склеивается со вторым и третьим, т.е.

$$C^0 = \begin{bmatrix} 0X \\ X0 \end{bmatrix}.$$

Таким образом, для логического элемента 2И можно сказать, что 1 на его выходе будет только при обеих единицах на входах, а для обеспечения 0 на выходе достаточно подать хотя бы на один вход 0.

При анализе схемы методом π- алгоритма, задавшись определённым значением сигнала на выходе, заменяют его соответствующим покрытием элемента, формирующего выходной сигнал. В результате этого определяется, какие сигналы должны быть на выходах элементов, подключённых к выходному логическому элементу. В свою очередь, сигналы на выходах этих элементов можно заменить соответствующими покрытиями, т.е. определить значения выходных сигналов для других логических элементов и т.д. Этот процесс продолжается до тех пор, пока не получатся покрытия, состоящие только из входных переменных, называемых опорными. Совокупность таких покрытий и даёт соответствующее покрытие схемы.

5.3. Метод синхронного моделирования

При данном методе считается, что все логические элементы переключаются одновременно, без задержки. В результате применения метода определяется установившееся значение сигнала на выходе схемы.

Рассмотрим метод синхронного моделирования на примере схемы на рис. 5.1. На первом этапе схему разбиваем на уровни и записываем в порядке возрастания уровня уравнения, описывающие функционирование логического элемента (таблица 5.3).

Таблица 5.3

Запись в порядке возрастания уровня уравнений, описывающих функционирование логического элемента

Уровень	№ элемента	Уравнение
1	1 2	$e_1 = X_1 \wedge X_2$ $e_2 = \overline{X_3 \vee X_4}$
2	3	$e_3 = \overline{e_1 \wedge e_2}$
3	4	$Y = e_4 = e_3 \vee X_5$

Проанализируем схему при подаче на вход набора X_1=0, X_2=0, X_3=0, X_4=1, X_5=1. Для этого решаем записанные уравнения в порядке возрастания уровня. Имеем:

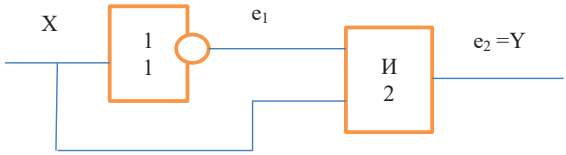
$$e_1 = X_1 \wedge X_2 = 0 \wedge 0 = 0;$$

$$e_2 = \overline{X_3 \vee X_4} = \overline{0 \vee 1} = 0;$$
$$e_3 = \overline{e_1 \wedge e_2} = \overline{0 \wedge 0} = 1;$$
$$Y = e_4 = e_3 \vee X_5 = 1 \vee 1 = 1.$$

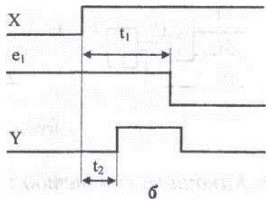
Следовательно, при подаче на вход набора (00011), на выходе будет Y=1. Аналогично можно промоделировать работу схемы при подаче на вход любого другого набора.

5.4. Метод асинхронного моделирования

Реальный логический элемент переключается за какое-то конечное время, зависящее от технологии изготовления, условий эксплуатации, емкостей нагрузки и т.д. Прохождение сигнала последовательно через несколько логических элементов будет приводить к накоплению времени задержки и возникновению сдвига во времени выходного сигнала по отношению ко входному. Наличие задержки и порождаемого ею временного сдвига сигналов может приводить к появлению на выходе отдельных логических элементов и всей схемы в целом кратковременных сигналов, не предусмотренных функцией, реализуемой схемой. Как иллюстрацию рассмотрим схему на рис. 5.3 .



а)



б

t_1 – время задержки инвертора;
t_2 – время задержки элемента 2И
Рис. 5.3. Риск сбоя

Данная схема реализует функцию $Y = X \wedge \overline{X} = 0$, т.е. константу 0 независимо от входного сигнала X. Однако в переходном процессе в результате задержки срабатывания логического элемента возможна ситуация, когда на обоих входах элемента 2И будут логические единицы, что может привести к появлению на выходе схемы логической 1 (рис. 5.3 б). Рассмотренный случай возможен при задержке срабатывания второго элемента больше, чем первого. Такое явление называется ***риском сбоя***. Различают статический и динамический риски сбоя.

При ***статическом*** риске сбоя до и после переходного процесса состояние выходного сигнала одно и то же, а во время переходного процесса возможно кратковременное появление противоположного сигнала.

При ***динамическом*** риске сбоя до и после переходного процесса состояния выходного сигнала противоположные, но в переходном процессе выходной сигнал несколько раз меняет свое значение. Динамический риск сбоя возможен в схеме (рис. 5.4).

В данном примере динамический риск сбоя на выходе КС сопровождается статическим риском на выходе элемента 1. Как видно из временных диаграмм, риск сбоя имеет место при наличии определенного временного сдвига между сигналами, поступающими на вход логического элемента. Нежелательные сигналы на выходе могут и отсутствовать при другом соотношении временных сигналов, однако принципиальная возможность их появления является фактором, снижающим надежность работы схемы. Поэтому очень важно уметь обнаруживать и устранять такие явления.

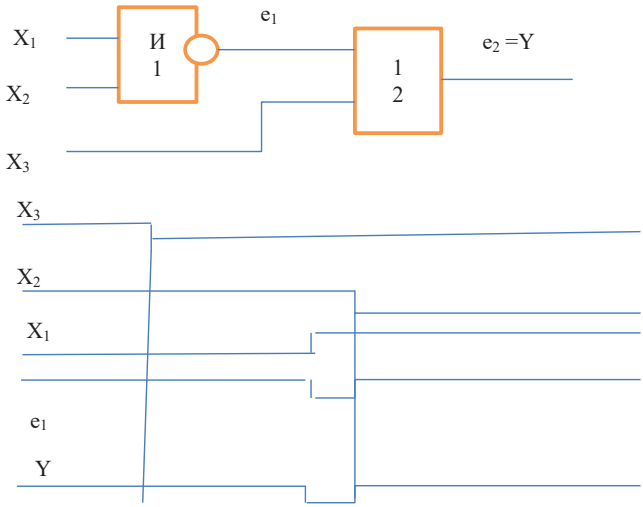


Рис. 5.4. Схема динамического риска сбоя

Литература

1. Р.М.Половов, А.Г.Рощин. Теория автоматов. Учебное пособие, часть 1.– М.:МГТУГА,2007.
2. Н.Н.Горнец, А.Г.Рощин, В.В.Соломенцев. Организация ЭВМ и систем. Учебное пособие для ВУЗов. –М.: Издательский центр «Академия», 2008.
3. Микушин А.В., Сажнев А.М., Синин В.И. Цифровые устройства и микропроцессоры. СПб, БХВ-Петербург, 2010.
4. Угрюмов Е. П. Цифровая схемотехника. СПб, БХВ-Петербург, 2010.

Содержание