

А. В. ЕРОШЕНКО, Л. Н. ТРОФИМОВА

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ

ЧАСТЬ 3

ЛОГИЧЕСКИЕ ОСНОВЫ ПОСТРОЕНИЯ КОМПЬЮТЕРА

ОМСК 2014

Министерство транспорта Российской Федерации
Федеральное агентство железнодорожного транспорта
Омский государственный университет путей сообщения

А. В. Ерошенко, Л. Н. Трофимова

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ

Часть 3

ЛОГИЧЕСКИЕ ОСНОВЫ ПОСТРОЕНИЯ КОМПЬЮТЕРА

Утверждено редакционно-издательским советом университета
в качестве методических указаний для самостоятельной работы
по дисциплине «Информатика»

Омск 2014

УДК 004.312(075.8)
ББК 22.124я7
Е78

Теоретические основы информатики. Часть 3. Логические основы построения компьютера: Методические указания для самостоятельной работы / А. В. Ерошенко, Л. Н. Трофимова; Омский гос. ун-т путей сообщения. Омск, 2014. 27 с.

Указания содержат описание основных логических элементов. Рассматриваются различные устройства, работу которых описывает алгебра логики: переключатели, триггеры, сумматоры. Приводятся практические рекомендации по составлению логических и переключательных схем и по их упрощению.

Предназначены для студентов первого курса всех специальностей и направлений подготовки очной и заочной форм обучения.

Библиогр.: 4 назв. Табл. 10. Рис. 15.

Рецензенты: доктор техн. наук, доцент А. В. Бубнов;
канд. техн. наук, доцент А. Г. Малютин.

ОГЛАВЛЕНИЕ

Введение.....	5
1. Логические элементы	7
2. Логические схемы и логические функции.....	8
2.1. Переход от логической схемы к формуле логической функции	8
2.2. Построение логической схемы по формуле логической функции	9
2.3. Преобразование логических выражений и схем.....	11
2.4. Построение логической схемы по таблице истинности логической функции.....	12
2.5. Задания.....	15
3. Триггеры. Сумматоры	17
3.1. Триггеры. RS-триггер	18
3.2. Сумматор. Полусумматор	19
3.3. Задания.....	21
4. Переключательные схемы.....	21
4.1. Основные элементы переключательных схем.....	21
4.2. Синтез и анализ переключательных схем.....	23
4.3. Задания.....	24
Библиографический список	26

ВВЕДЕНИЕ

В вычислительных машинах коды нуля и единицы представляются электрическими сигналами, имеющими два различных состояния. Наиболее распространенными способами физического представления информации являются импульсный и потенциальный:

- импульс или его отсутствие;
- высокий или низкий потенциал;
- высокий потенциал или его отсутствие.

При импульсном способе отображения код единицы идентифицируется наличием электрического импульса, код нуля – его отсутствием (может быть и наоборот). Импульс характеризуется амплитудой и длительностью, причем длительность должна быть меньше временного такта машины.

При потенциальном способе отображения код единицы – это высокий уровень напряжения, а код нуля – отсутствие сигнала или низкий его уровень. Уровень напряжения не меняется в течение всего такта работы машины. Форма и амплитуда сигнала при этом во внимание не принимаются, а фиксируется лишь сам факт наличия или отсутствия потенциала.

Изложенное обусловило то, что для анализа и синтеза схем в компьютере при алгоритмизации и программировании решения задач широко используется математический аппарат алгебры логики, оперирующий также с двумя понятиями – «истина» и «ложь».

Кроме того, связь между булевой алгеброй и компьютерами лежит и в используемой в ЭВМ системе счисления. Как известно, она двоичная, поэтому в устройствах компьютера можно хранить и преобразовывать как числа, так и значения логических переменных.

В ЭВМ применяются электрические схемы, состоящие из множества переключателей. Переключатель может находиться только в двух состояниях: замкнутом и разомкнутом. В первом случае ток проходит, во втором – нет. Описывать работу таких схем удобно с помощью алгебры логики. В зависимости от положения переключателей можно получить или не получить сигналы на выходах.

При всей сложности устройства электронных блоков современных компьютеров выполняемые ими действия осуществляются комбинацией относительно небольшого числа типовых логических элементов.

Устройством, способным запоминать, хранить и позволяющим считывать информацию, является триггер, изобретенный в начале XX в. Бонч-Бруевичем. Триггер способен хранить один двоичный разряд за счет того, что может находиться в двух устойчивых состояниях. В основном триггеры используются в регистрах процессора.

Сумматоры широко используются в арифметико-логических устройствах (АЛУ) процессора и выполняют суммирование двоичных разрядов. В современных компьютерах микроскопические транзисторы в кристалле интегральной схемы сгруппированы в системы логических элементов, выполняющих логические операции над двоичными числами. С их помощью построены описанные в данной работе двоичные сумматоры, позволяющие складывать многоразрядные двоичные числа, производить вычитание, умножение, деление и сравнение чисел между собой. Логические элементы, действуя по определенным правилам, управляют движением данных и выполнением инструкций в компьютере.

Построение всех логических схем компьютеров, обрабатывающих информацию в двоичной системе счисления основывается на знании законов булевой алгебры. По этой причине при изучении курса информатики рассматриваются не только математические основы алгебры логики, а также их практическое применение в ЭВМ.

Авторы выражают глубокую благодарность Соколовской Нине Николаевне за оказанную помощь при разработке методических указаний.

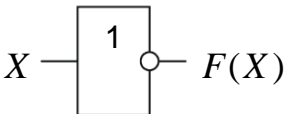
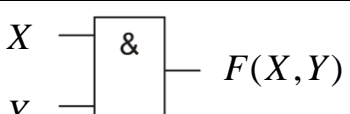
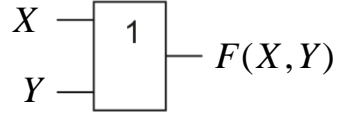
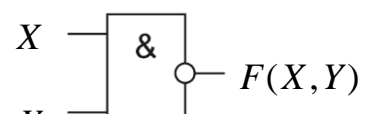
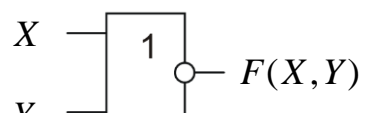
1. ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ

В основе построения компьютеров, а точнее – аппаратного обеспечения, лежат простые логические элементы, которые можно комбинировать между собой, создавая тем самым различные схемы. Одни схемы подходят для осуществления арифметических операций, а на основе других строят различную память ЭВМ.

Базовые логические элементы, их условные обозначения и логические функции, которые они реализуют, приведены в табл. 1.1. Для базового логического элемента НЕ приведены два варианта условного обозначения, которые можно встретить в различных источниках.

Таблица 1.1

Базовые логические элементы

Название логического элемента	Условное обозначение	Логическая функция
НЕ		$F(X) = \overline{X}$
		
И		$F(X,Y) = X \wedge Y$
ИЛИ		$F(X,Y) = X \vee Y$
И-НЕ		$F(X,Y) = \overline{X \wedge Y}$
ИЛИ-НЕ		$F(X,Y) = \overline{X \vee Y}$

Простейший логический элемент представляет собой транзисторный инвертор, который преобразует низкое напряжение в высокое или наоборот (высокое в низкое). Это можно представить как преобразование логического нуля в логическую единицу или наоборот. Таким образом, получаем логический элемент НЕ.

Соединив пару транзисторов различным способом, получают логические элементы ИЛИ-НЕ и И-НЕ, которые принимают уже не один, а два и более входных сигнала. Выходной сигнал всегда один и зависит (выдает высокое или низкое напряжение) от входных сигналов. В случае логического элемента ИЛИ-НЕ получить высокое напряжение (логическую единицу) можно только при условии низкого напряжения на всех входах. В случае элемента И-НЕ все наоборот: логическая единица получается, если все входные сигналы будут нулевыми. Как видно, эти операции обратны таким привычным логическим операциям, как И и ИЛИ. Обычно используются логические элементы И-НЕ и ИЛИ-НЕ, так как их реализация проще: И-НЕ и ИЛИ-НЕ строятся на основе двух транзисторов, тогда как логические И и ИЛИ – на трех.

Транзистору требуется мало времени для переключения из одного состояния в другое (время переключения оценивается в наносекундах), что является одним из существенных преимуществ схем, построенных на их основе.

2. ЛОГИЧЕСКИЕ СХЕМЫ И ЛОГИЧЕСКИЕ ФУНКЦИИ

2.1. Переход от логической схемы к формуле логической функции

Любую сложную логическую функцию можно реализовать, имея простой набор базовых логических операций. Простейшие логические элементы реализованы аппаратно (схемно).

Переход от логической схемы, построенной из логических элементов, требует знания булевых функций, реализуемых логическими элементами. Начиная от входов логической схемы, используя принцип суперпозиции (подстановки функций в качестве аргументов в другие функции), получаем на выходе функцию, реализуемую всей схемой.

Пример 2.1. Рассмотрим переход от логической схемы к формуле логической функции для данной схемы, изображенной на рис. 2.1.

На выходе каждого логического элемента представлена логическая функция, которую он реализует в соответствии с входными значениями.

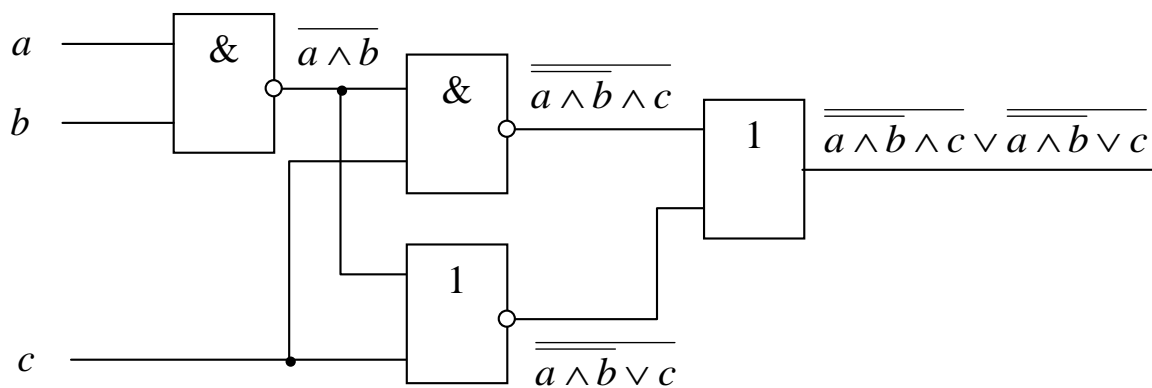


Рис. 2.1. Переход от логической схемы к формуле функции

2.2. Построение логической схемы по формуле логической функции

Построение и расчет любой логической схемы осуществляется с ее выхода.

Пример 2.2. Задана логическая функция $F(A, B, C) = \bar{B} \wedge A \vee B \wedge \bar{A} \vee C \wedge \bar{B}$.

Построение логической схемы по формуле логической функции разобьем на несколько этапов.

Первый этап: в соответствии с приоритетом логических операций в исходной функции $F(A, B, C)$ выполняется логическое сложение, т. е. логическая операция ИЛИ, для которой входными переменными являются $\bar{B} \wedge A, B \wedge \bar{A}, C \wedge \bar{B}$. Логическая схема, реализующая первый этап, представлена на рис. 2.2.

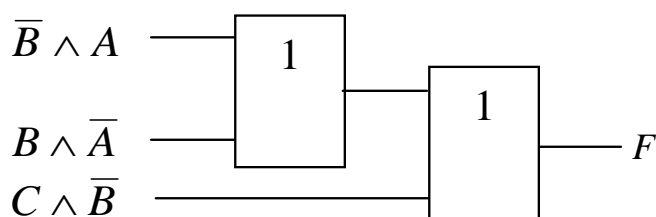


Рис. 2.2. Логическая схема операции ИЛИ для реализации логической функции $F(A, B, C) = \bar{B} \wedge A \vee B \wedge \bar{A} \vee C \wedge \bar{B}$

Второй этап: ко входам каждого из элементов ИЛИ подключаются логические элементы И, входными переменными которых являются уже A, B, C и их инверсии. Логическая схема, реализующая второй этап, представлена на рис. 2.3.

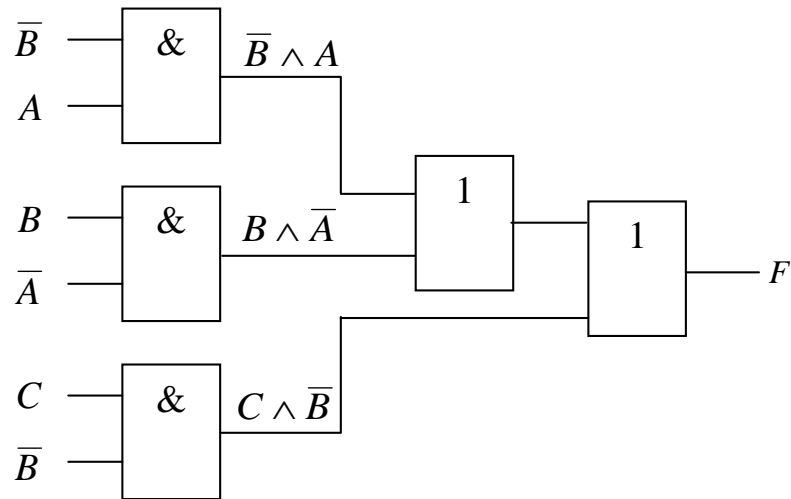


Рис. 2.3. Логическая схема операций И, ИЛИ для реализации логической функции $F(A, B, C) = \overline{B} \wedge A \vee B \wedge \overline{A} \vee C \wedge \overline{B}$

Третий этап: для получения инверсий \overline{A} и \overline{B} на соответствующих входах ставят логические элементы НЕ. Логическая схема, реализующая третий этап, представлена на рис. 2.4.

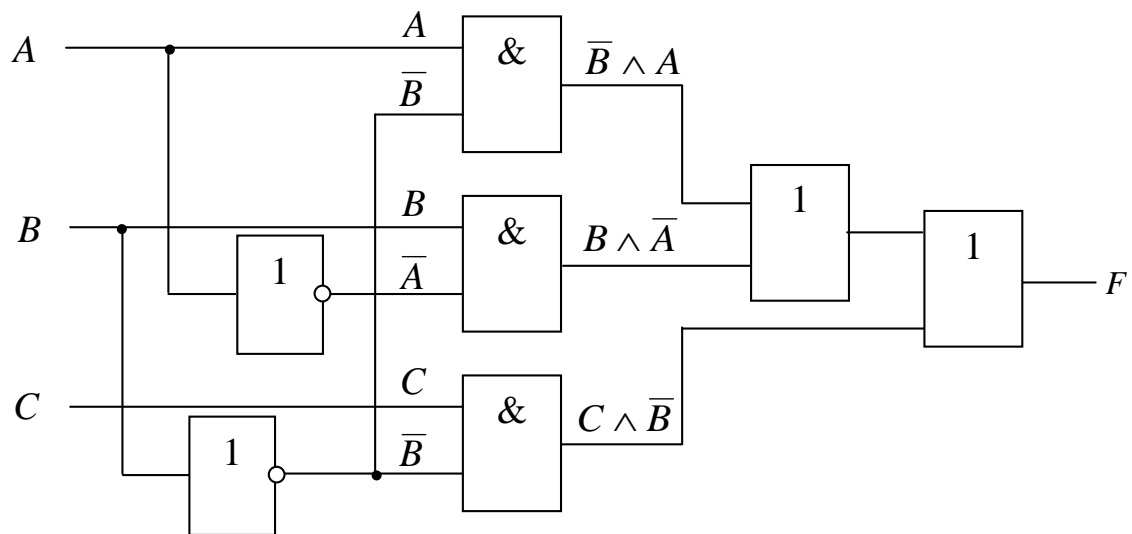


Рис. 2.4. Логическая схема операций НЕ, И, ИЛИ для реализации логической функции $F(A, B, C) = \overline{B} \wedge A \vee B \wedge \overline{A} \vee C \wedge \overline{B}$

Такой способ построения логической схемы основан также на принципе суперпозиции. В связи с тем, что значениями логических функций могут быть только нули и единицы, любые логические функции могут быть представлены как аргументы других более сложных функций.

2.3. Преобразование логических выражений и схем

Преобразование логических выражений часто сводится к их упрощению. При этом надо использовать законы алгебры логики, обратив особое внимание на приемы замены отдельной переменной или константы формулой.

Преобразование логических схем можно выполнить следующим образом. Записать логическую функцию, реализуемую логической схемой, затем упростить полученное выражение и составить новую логическую схему, реализующую его.

Пример 2.3. Упростить логическое выражение: $x \wedge \bar{y} \vee \bar{x} \wedge y \wedge z \vee x \wedge z$.

Введем вспомогательный логический множитель $(y \vee \bar{y})$:

$$x \wedge \bar{y} \vee \bar{x} \wedge y \wedge z \vee x \wedge z = x \wedge \bar{y} \vee \bar{x} \wedge y \wedge z \vee x \wedge z \wedge (y \vee \bar{y}).$$

На основании дистрибутивного закона раскрываем скобки и комбинируем два крайних и два средних логических слагаемых:

$$x \wedge \bar{y} \vee \bar{x} \wedge y \wedge z \vee x \wedge z \wedge (y \vee \bar{y}) = (x \wedge \bar{y} \vee x \wedge \bar{y} \wedge z) \vee (\bar{x} \wedge y \wedge z \vee x \wedge y \wedge z).$$

Используя закон поглощения и склеивания, получим:

$$(x \wedge \bar{y} \vee x \wedge \bar{y} \wedge z) \vee (\bar{x} \wedge y \wedge z \vee x \wedge y \wedge z) = x \wedge \bar{y} \vee y \wedge z.$$

Тогда $F(x, y, z) = x \wedge \bar{y} \vee y \wedge z$. Логическая схема для полученной логической функции представлена на рис. 2.5.

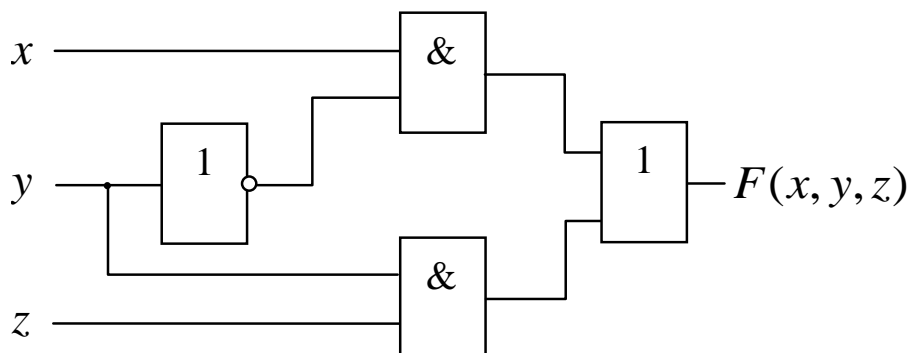


Рис. 2.5. Логическая схема для примера 2.3

Пример 2.4. Упростить логическую схему из примера 2.1 (см. рис. 2.1).

Логическое выражение, описывающее логическую схему, имеет вид:

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

Применим к первому и второму слагаемым закон де Моргана:

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

Применим закон двойного отрицания:

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

Применим закон поглощения, сгруппировав первое и третье слагаемые логического выражения:

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

Логическая схема для полученной логической функции $F(a, b, c) = a \wedge b \vee \overline{c}$ представлена на рис. 2.6.

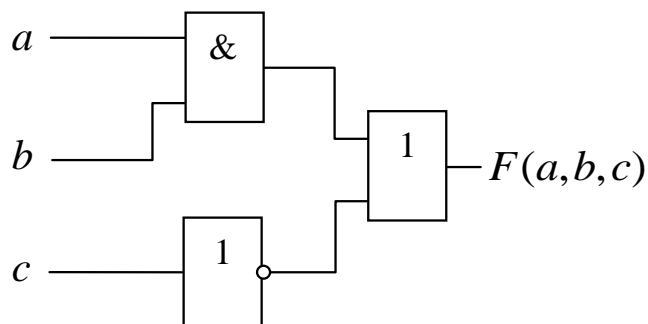


Рис. 2.6. Логическая схема для примера 2.4

2.4. Построение логической схемы по таблице истинности логической функции

Любую логическую функцию, заданную таблицей истинности, можно представить в виде как совершенной дизъюнктивной нормальной формы (СДНФ), так и совершенной конъюнктивной нормальной формы (СКНФ), то есть выразить с помощью трех логических операций: конъюнкции, дизъюнкции и отрицания.

Если булева функция не равна тождественному нулю, то ее можно представить в виде СДНФ по ее таблице истинности следующим образом: берем только те наборы переменных (x_1, x_2, \dots, x_n) , для которых $f(x_1, x_2, \dots, x_n) = 1$, и составляем простую конъюнкцию для этого набора так: если $x_i = 0$, то берем в этой конъюнкции $\overline{x_i}$, если $x_i = 1$, то берем x_i . Составляя дизъюнкцию этих простых конъюнкций, приходем к СДНФ.

Иными словами, алгоритм получения СДНФ по таблице истинности можно описать следующими действиями:

отметить те строки таблицы истинности, в столбце функции которых стоят единицы;

выписать для каждой отмеченной строки конъюнкцию всех переменных следующим образом: если значение некоторой переменной в данной строке равно единице, то в конъюнкцию включают саму эту переменную, если равно нулю, то ее отрицание;

все полученные конъюнкции связать операцией дизъюнкции.

По аналогии с представлением любой функции (не равной тождественному нулю) в виде СДНФ можно функцию (не равную тождественной единице) представить в виде СКНФ: простая дизъюнкция составляется для тех наборов переменных (x_1, x_2, \dots, x_n) , для которых $f(x_1, x_2, \dots, x_n) = 0$, причем если $x_i = 1$, то в этой дизъюнкции берем $\overline{x_i}$, если же $x_i = 0$, то берем x_i .

Алгоритм получения СКНФ по таблице истинности:

отметить те строки таблицы истинности, в столбце функции которых стоят нули;

выписать для каждой отмеченной строки дизъюнкцию всех переменных следующим образом: если значение некоторой переменной в данной строке равно нулю, то в дизъюнкцию включают саму эту переменную, если равно единице, то ее отрицание;

все полученные дизъюнкции связать операцией конъюнкции.

Пример 2.5. Составить для логической функции импликации и сложения по модулю два (неравнозначности) – СДНФ и СКНФ.

Таблица истинности заданных логических функций представлена в табл. 2.1.

Таблица 2.1

Таблица истинности импликации и
неравнозначности

x	y	$x \rightarrow y$	$x \oplus y$
0	0	1	0
0	1	1	1
1	0	0	1
1	1	1	0

Тогда СДНФ для этих функций запишется в виде:

$$x \rightarrow y = \bar{x} \wedge \bar{y} \vee \bar{x} \wedge y \vee x \wedge y; \quad x \oplus y = x \wedge \bar{y} \vee \bar{x} \wedge y;$$

СКНФ:

$$x \rightarrow y = \bar{x} \vee y; \quad x \oplus y = (x \vee y) \wedge (\bar{x} \vee \bar{y}).$$

Пример 2.6. Составить логическую схему, реализующую логическую функцию $f(x, y, z)$, заданную таблицей истинности (табл. 2.2).

Выберем строки таблицы, где значения функции равны единице. Таких строк три, т. е. функция принимает истинное значение только для этих трех наборов переменных. Отсюда СДНФ для функции можно записать следующим образом:

$$f(x, y, z) = \bar{x} \wedge y \wedge \bar{z} \vee \bar{x} \wedge y \wedge z \vee x \wedge y \wedge \bar{z}.$$

Полученное выражение можно упростить. Для этого сгруппируем первые два слагаемых и вынесем множитель $\bar{x} \wedge y$ за скобки:

$$\bar{x} \wedge y \wedge \bar{z} \vee \bar{x} \wedge y \wedge z \vee x \wedge y \wedge \bar{z} = \bar{x} \wedge y \wedge (\bar{z} \vee z) \vee x \wedge y \wedge \bar{z}.$$

Применяя законы исключенного третьего и исключения констант, имеем:

$$\bar{x} \wedge y \wedge (\bar{z} \vee z) \vee x \wedge y \wedge \bar{z} = \bar{x} \wedge y \vee x \wedge y \wedge \bar{z}.$$

Вынесем логический множитель y за скобки, а к скобочному выражению применим дистрибутивный закон:

$$\bar{x} \wedge y \vee x \wedge y \wedge \bar{z} = y \wedge (\bar{x} \vee \bar{z}).$$

Применяя закон де Моргана, имеем:

$$y \wedge (\bar{x} \vee \bar{z}) = y \wedge \overline{x \wedge z}.$$

Получилась очень простая логическая схема (рис. 2.7).

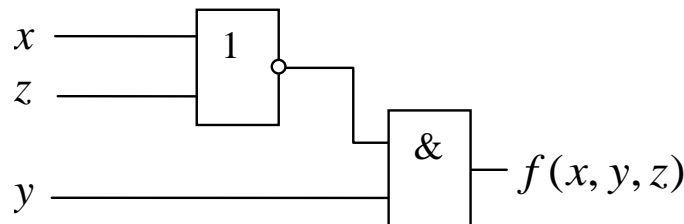


Рис. 2.7. Логическая схема, реализующая функцию $f(x, y, z) = y \wedge \overline{x \wedge z}$

Таблица 2.2

Таблица $f(x, y, z)$

x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

2.5. Задания

Задание 1. Решить задачу.

Три преподавателя отбирают задачи для олимпиады. На выбор предлагается несколько задач. По каждой из задач каждый из преподавателей высказывает свое мнение: легкая задача – (0) или трудная – (1). Задача включается в олимпиадное задание, если не менее двух преподавателей отметят ее как трудную, но если все три преподавателя считают ее трудной, то такая задача не включается в олимпиадное задание как слишком сложная.

Составьте структурную формулу и функциональную схему устройства, которое на выходе будет выдавать 1, если задача включена в олимпиадное задание, и 0, если не включена.

Задание 2. В соответствии с номером варианта (табл. 2.3):

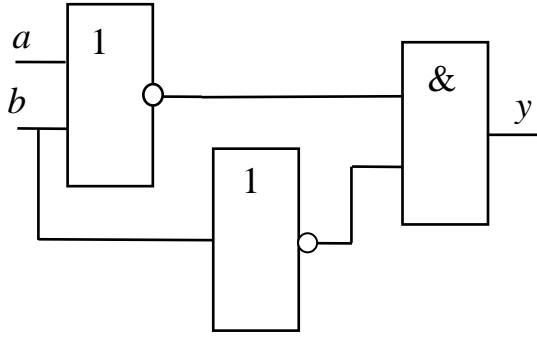
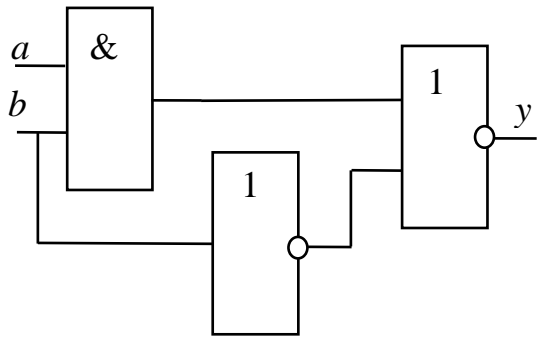
по логической схеме составить логическое выражение и таблицу истинности для него. Затем упростить логическое выражение, построить по нему упрощенную логическую схему и таблицу истинности. Полученные таблицы сравнить между собой;

по заданному логическому выражению составить таблицу истинности и логическую схему.

Таблица 2.3

Исходные данные для задания 2

Номер варианта	Логическая схема	Логическое выражение (a , b и c – логические переменные)
1	2	3
1		$\bar{a} \oplus b \vee \bar{c}$
2		$a \vee b \oplus \bar{c}$
3		$a \vee (b \oplus \bar{c})$
4		$a \vee \bar{b} \wedge \bar{c}$
5		$a \oplus \bar{b} \oplus \bar{c}$

1	2	3
6		$\overline{a \wedge b \vee c}$
7		$\overline{a \oplus b \oplus c}$
8		$\overline{a \wedge b \vee c}$
9		$\overline{a \oplus b \oplus c}$
10		$\overline{a \wedge b \vee c}$
11		$a \oplus b \vee c$
12		$\overline{a} \wedge (b \vee \overline{c})$
13		$a \vee (b \oplus c)$
14		$\overline{a \vee b \vee c}$
15		$\overline{a \vee b} \oplus c$

Задание 3. Для логической функции равнозначности (эквивалентности) составить СКНФ и СДНФ. В качестве образца использовать пример 2.5. Для полученных логических функций построить логические схемы.

Задание 4. В соответствии с номером варианта составить логическую схему, реализующую логическую функцию $f(x, y, z)$, заданную таблицей истинности (табл. 2.4). В качестве образца использовать пример 2.6.

До построения логической схемы необходимо упростить первоначально полученное логическое выражение.

Таблица 2.4

Таблица логической функции $f(x, y, z)$

Переменные			Номер варианта														
x	y	z	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0
0	0	1	1	0	1	1	0	0	1	0	0	0	0	0	1	0	0
0	1	0	0	1	0	1	0	0	0	1	0	1	1	1	0	0	1
0	1	1	0	0	1	0	1	1	0	1	1	0	1	0	1	0	0
1	0	0	0	1	0	1	0	0	1	0	1	0	0	0	1	0	1
1	0	1	0	0	1	0	0	0	1	0	0	1	1	1	0	1	1
1	1	0	1	0	0	0	0	1	0	1	0	1	0	0	0	1	0
1	1	1	0	0	0	0	1	1	0	0	0	0	0	1	0	1	0

3. ТРИГГЕРЫ. СУММАТОРЫ

Память (устройство, предназначенное для хранения данных и команд) является важной частью компьютера. Можно сказать, что память и определяет сам компьютер: если вычислительное устройство не имеет памяти, то оно уже не компьютер.

Элементарной единицей компьютерной памяти является бит, поэтому требуется устройство, способное находиться в двух состояниях, т. е. хранить единицу или ноль. Также это устройство должно уметь быстро переключаться из одного состояния в другое под внешним воздействием, что дает возможность изменять информацию. И, наконец, устройство должно позволять определять его состояние, т. е. должно предоставлять информацию о своем состоянии.

Устройством, способным запоминать, хранить и позволяющим считывать информацию, является триггер, изобретенный в начале XX в. М. А. Бонч-Бруевичем.

Разнообразие триггеров весьма велико. Наиболее простой из них так называемый RS-триггер, который собирается из двух логических элементов.

3.1. Триггеры. RS-триггер

Триггер – это электронная схема, широко применяемая в регистрах компьютера для надежного запоминания одного разряда двоичного кода. Триггер имеет два устойчивых состояния, одно из которых соответствует двоичной единице, а другое – двоичному нулю.

Термин «триггер» происходит от английского слова *trigger* – защёлка, спусковой крючок. Для обозначения этой схемы в английском языке чаще употребляется термин «flip-flop», что в переводе означает «хлопанье». Это звукоподражательное название электронной схемы указывает на ее способность почти мгновенно переходить из одного электрического состояния в другое и наоборот.

Самый распространенный тип триггера – так называемый RS-триггер (S и R , соответственно, от английских *set* – установка и *reset* – сброс). Условное графическое обозначение триггера приведено на рис. 3.1.

RS-триггер «запоминает», на какой его вход подавался сигнал, соответствующий единице, в последний раз. Если сигнал был подан на S -вход, то триггер на выходе постоянно «сообщает», что хранит единицу. Если сигнал, соответствующий единице, подан на R -вход, то триггер на выходе Q имеет ноль. Триггер имеет два выхода Q и \bar{Q} , в том случае если не указано название выхода, то имеется в виду выход Q (\bar{Q} всегда имеет противоположное Q значение).

Другими словами, вход S отвечает за установку триггера в единицу, а вход R – за установку триггера в ноль. Установка производится сигналом с высоким напряжением (соответствует единице), все зависит от того, на какой вход он подается сигнал.

Большую часть времени на входы подается сигнал, равный нулю (низкое напряжение). При этом триггер сохраняет свое прежнее состояние.

Реализация триггера с помощью вентилях ИЛИ-НЕ показана на рис. 3.2, а таблица истинности триггера представлена в табл. 3.1.

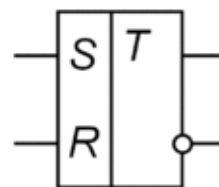


Рис. 3.1. Условное графическое обозначение RS-триггера

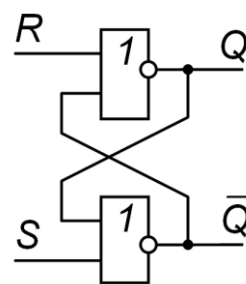


Рис. 3.2. Реализация RS-триггера на логических элементах ИЛИ-НЕ

Проанализируем возможные комбинации значений входов R и S триггера, используя его схему (см. рис. 3.2) и таблицу истинности схемы ИЛИ-НЕ (см. табл. 3.1).

Таблица 3.1

Таблица истинности RS-триггера

S	R	Q	\bar{Q}
0	0	Запрещенное состояние	
0	1	1	0
1	0	0	1
1	1	Хранение бита	

Если на входы триггера подать $S = 1$, $R = 0$, то (независимо от состояния) на выходе Q верхнего вентиля появится ноль. После этого на входах нижнего вентиля окажется $R = 0$, $Q = 0$ и выход \bar{Q} станет равным единице.

Точно так же при подаче нуля на вход S и единицы на вход R на выходе \bar{Q} появится ноль, а на Q – единица.

Если на входы R и S подана логическая единица, то состояние Q и \bar{Q} не меняется.

Подача на оба входа R и S логического нуля может привести к неоднозначному результату, поэтому эта комбинация входных сигналов запрещена.

Поскольку один триггер может запомнить только один разряд двоичного кода, то для запоминания байта нужно восемь триггеров, для запоминания килобайта, соответственно, $8 \cdot 2^{10} = 8192$ триггеров. Современные микросхемы памяти содержат миллионы триггеров.

3.2. Сумматор. Полусумматор

Арифметико-логическое устройство процессора обязательно содержит в своем составе такие элементы, как сумматоры. Эти схемы позволяют складывать двоичные числа.

Пример 3.1. Построить логическую схему полусумматора. Эта схема должна суммировать два одноразрядных двоичных числа и вырабатывать их сумму s и перенос в следующий разряд p . Следует отметить, что в двоичной системе счисления

$0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1, 1 + 1 = 10$. Перенос в старший разряд возникает только в последнем случае. Алгоритм сложения записан таблицей истинности – табл. 3.2.

Анализ таблицы истинности полусумматора показывает, что логическая функция s двух аргументов – a и b – это неравнозначность или «исключающее ИЛИ», а p – это конъюнкция. Имеем формулы: $s = a \oplus b, p = a \wedge b$.

Логическая схема полусумматора изображена на рис. 3.3 (логический элемент с обозначением «=1» реализует операцию «неравнозначность»), а ее условное обозначение – на рис. 3.4.

Следует обратить внимание на то, что в качестве булевых переменных выступают цифровые разряды двоичного числа. Этот пример показывает применение алгебры логики для построения всех логических схем компьютера, преобразующего информацию, представленную в двоичной системе счисления.

Таблица 3.2

Таблица истинности полусумматора

a	b	s	p
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

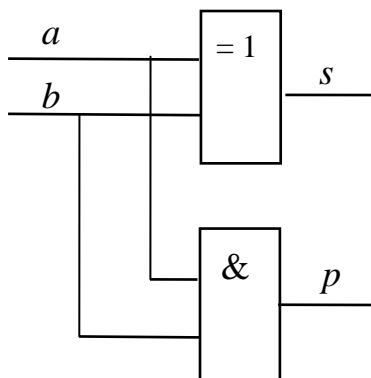


Рис. 3.3. Логическая схема полусумматора

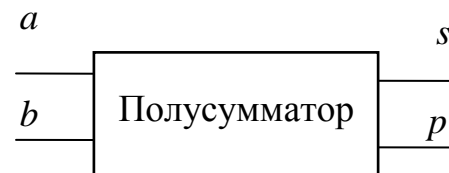


Рис. 3.4. Условное обозначение полусумматора

В отличие от полусумматора сумматор учитывает перенос из предыдущего разряда, поэтому имеет не два, а три входа.

Чтобы учесть перенос, приходится усложнять схему. По сути она получается состоящей из двух полусумматоров.

Пример 3.2. Построить логическую схему одноразрядного сумматора на три входа: a и b – значения одноименных разрядов двух двоичных чисел, p – перенос из предыдущего разряда. Сумматор должен выработать значение суммы s и переноса q в следующий разряд.

Схема одноразрядного сумматора изображена на рис. 3.5. В правильности его работы можно убедиться составлением таблицы истинности.

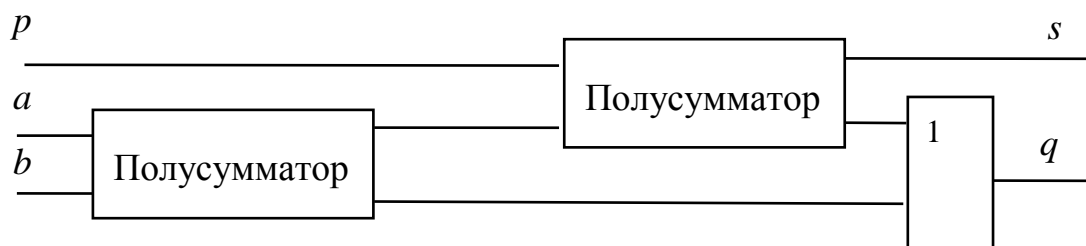


Рис. 3.5. Логическая схема одноразрядного сумматора

3.3. Задания

Задание 1. Составить логическую схему полусумматора, не используя логический элемент, реализующий операцию неравнозначности.

Задание 2. Не используя логический элемент «неравнозначность», составить логическую схему и таблицу истинности одноразрядного сумматора.

Задание 3. Используя один полусумматор и два сумматора, составить логическую схему суммирования двух трехразрядных двоичных чисел.

4. ПЕРЕКЛЮЧАТЕЛЬНЫЕ СХЕМЫ

Переключательная схема – это схематическое изображение некоторого устройства, состоящего из переключателей и соединяющих их проводников, а также из входов и выходов, на которые подается и с которых снимается электрический сигнал.

В компьютерах и других автоматических устройствах широко применяются электрические схемы, содержащие сотни и тысячи переключательных элементов: реле, выключателей и т. п. При разработке этих схем также используется аппарат алгебры логики.

4.1. Основные элементы переключательных схем

Каждый переключатель имеет только два состояния: замкнутое и разомкнутое. Переключателю X поставим в соответствие логическую переменную x , которая принимает значение единицы в том и только в том случае, когда


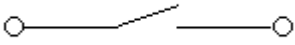
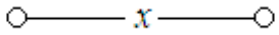
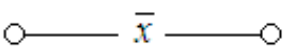
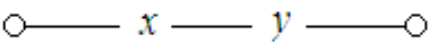
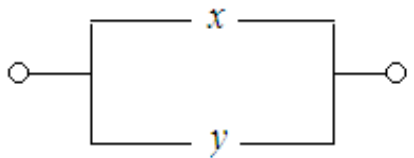
переключатель X замкнут и схема проводит ток; если же переключатель разомкнут, то x равен нулю.

Будем считать, что два переключателя – X и \bar{X} – связаны таким образом, что когда X замкнут, то \bar{X} разомкнут, и наоборот. Следовательно, если переключателю X поставлена в соответствие логическая переменная x , то переключателю \bar{X} должна соответствовать переменная \bar{x} .

Всей переключательной схеме также можно поставить в соответствие логическую переменную, равную единице, если схема проводит ток, и равную нулю – если не проводит. Эта переменная является функцией от переменных, соответствующих всем переключателям схемы, и называется функцией проводимости.

Таблица 4.1

Функции проводимости F некоторых переключательных схем

Изображение переключательной схемы	Функция проводимости переключательной схемы
1	2
	Схема не содержит переключателей и проводит ток всегда, следовательно, $F = 1$
	Схема содержит один постоянно разомкнутый контакт, следовательно, $F = 0$
	Схема проводит ток, когда переключатель x замкнут, и не проводит, когда x разомкнут, следовательно, $F(x) = x$
	Схема проводит ток, когда переключатель x разомкнут, и не проводит, когда x замкнут, следовательно, $F(x) = \bar{x}$
	Схема проводит ток, когда оба переключателя замкнуты, следовательно, $F(x, y) = x \wedge y$
	Схема проводит ток, когда хотя бы один из переключателей замкнут, следовательно, $F(x, y) = x \vee y$

Любая сложная схема может быть преобразована в отдельные группы и представлена в виде логических функций нескольких переменных.

Две схемы называются равносильными, если через одну из них проходит ток тогда и только тогда, когда он проходит через другую (при одном и том же входном сигнале).

Из двух равносильных схем более простой считается та схема, функция проводимости которой содержит меньшее число логических операций или переключателей.

Задача нахождения среди равносильных схем наиболее простых является очень важной. Большой вклад в ее решение внесли российские ученые Ю. И. Журавлев, С. В. Яблонский и др.

4.2. Синтез и анализ переключательных схем

При рассмотрении переключательных схем возникают две основные задачи: синтез и анализ схемы.

Синтез схемы по заданным условиям ее работы сводится к следующим трем этапам:

составлению функции проводимости по таблице истинности, отражающей эти условия;

упрощению этой функции;

построению соответствующей схемы.

Анализ схемы сводится к следующим двум действиям:

определению значений ее функции проводимости при всех возможных наборах входящих в эту функцию переменных;

получению упрощенной формулы.

Пример 4.1. Определить и проанализировать функцию проводимости переключательной схемы, представленной на рис. 4.1.

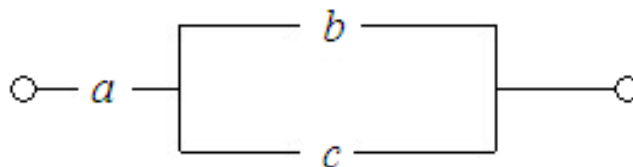


Рис. 4.1. Переключательная схема для примера 4.1

Функция проводимости для данной переключательной схемы имеет вид:
 $F(a, b, c) = a \wedge (b \vee c)$. Таблица истинности приведена в табл. 4.2.

Таблица 4.2

Таблица истинности переключательной схемы примера 4.1

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

Анализируя таблицу истинности, можно сделать логический вывод о том, что для прохождения тока необходимо и достаточно, чтобы были замкнуты переключатели a и b или a и c , или все три – a, b, c .

Пример 4.2. Построить схему, содержащую четыре переключателя – x, y, z, t , такую, чтобы она проводила ток тогда и только тогда, когда замкнут контакт переключателя t и какой-нибудь из остальных трех контактов.

В этом случае можно обойтись без построения таблицы истинности. Очевидно, что функция проводимости имеет вид: $F(x, y, z, t) = t \wedge (x \vee y \vee z)$. Схема представлена на рис. 4.2.

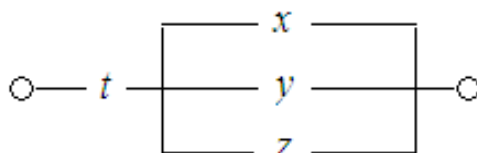


Рис. 4.2. Переключательная схема для примера 4.2

4.3. Задания

Задание 1. Определить и проанализировать функции проводимости переключательных схем, представленных на рис. 4.3.

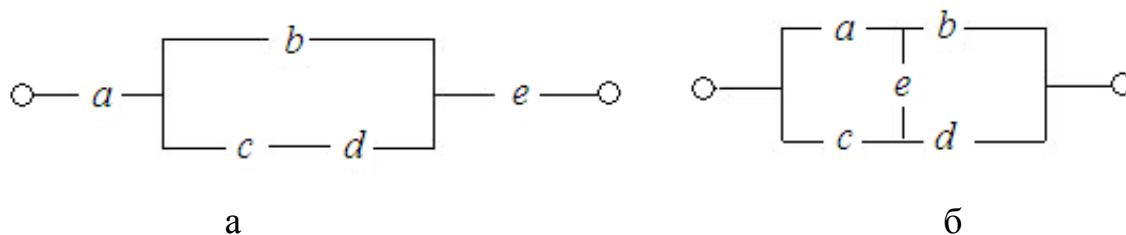


Рис. 4.3. Переключательные схемы для задания 1

Задание 2. В соответствии с номером варианта проверьте равносильность переключательных схем, представленных в табл. 4.3.

Таблица 4.3

Переключательные схемы для задания 2

Номер варианта	Переключательные схемы
1, 6, 11	
2, 7, 12	
3, 8, 13	
4, 9, 14	
5, 10, 15	

Библиографический список

1. Могилев А. В. Информатика: Учебное пособие / А. В. Могилев, Н. И. Пак, Е. К. Хеннер. М.: Академия, 2008. 848 с.
2. Могилев А. В. Практикум по информатике: Учебное пособие / А. В. Могилев, Н. И. Пак, Е. К. Хеннер. М.: Академия, 2008. 608 с.
3. Лихтарников Л. М. Математическая логика. Курс лекций. Задачник, практикум и решения: Учебное пособие / Л. М. Лихтарников, Т. Г. Сукачева. СПб: Лань, 2008. 288 с.
4. Кузнецов О. П. Дискретная математика для инженера / О. П. Кузнецов. СПб: Лань, 2004. 400 с.

Учебное издание

ЕРОШЕНКО Александра Викторовна,
ТРОФИМОВА Людмила Николаевна

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ
Часть 3
ЛОГИЧЕСКИЕ ОСНОВЫ ПОСТРОЕНИЯ КОМПЬЮТЕРА

Редактор Н. А. Майорова
Корректор И. А. Сенеджук

Подписано в печать . . . 2014. Формат $60 \times 84 \frac{1}{16}$.
Офсетная печать. Бумага офсетная. Усл. печ. л. 1,7. Уч.-изд. л. 1,9.
Тираж 500 экз. Заказ .

**

Редакционно-издательский отдел ОмГУПС
Типография ОмГУПС

*

644046, г. Омск, пр. Маркса, 35