УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2021

УНИВЕРСИТЕТ ИТМО

Элементы цифровых и аналогово-цифровых устройств. Алгебра логики

Санкт-Петербург, 2021

Темы, освещенные в презентации

- Элементы алгебры логики
- Логические функции

Базовые логические элементы

Цифровые устройства обработки информации, в том числе электронно-вычислительные машины, широко используют электронные схемы, называемые базовыми логическими элементами (ЛЭ).

Особенностью ЛЭ является то, что напряжение на их выводах и входах принято характеризовать не числовым значением, а одним из двух возможных уровней – высоким или низким.

Условно одному из этих уровней присваивается значение логической единицы, а другому – логического нуля.

Таким образом, сигналы на входе и выходе ЛЭ являются двоичными или бинарными.

Преобразование информации, представленной в цифровой форме, осуществляется путем выполнения определенной последовательности арифметических и логических операций в сложных цифровых устройствах, построенных на базе простых логических элементов.

Элементы алгебры логики

Основой для проектирования логических устройств является алгебра логики или булева алгебра, в которой логическая переменная X принимает только два значения: X=1 (логическая 1) и X=0 (логический 0). В алгебре логики определены три основные операции: ***конъюнкция*** (логическое ***умножение***), ***дизъюнкция*** (логическое ***сложение***), ***инверсия*** (логическое ***отрицание***).

Результат ***конъюнкции***, называемой также операцией ***И***, для переменных X1 и X2 записывается в виде Y=X1*X2, или, что тоже, Y=X1 ∧ X2. Логическая функция Y также является бинарной, ее сочетание со всеми сочетаниями логических переменных определяются так называемой таблицей истинности.

Результат ***дизъюнкции***, называемой по-другому операцией ***ИЛИ*** и записываемой в виде Y=X1+X2 или Y=X1vX2, определяется таблицей истинности, приведенной на втором рисунке.

Элементы алгебры логики

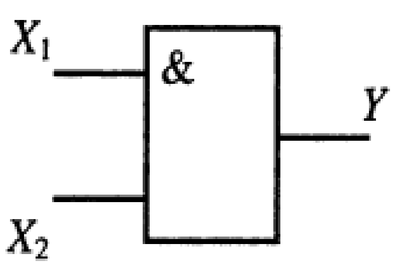
Операции И и ИЛИ могут производиться над большим, чем два, числом переменных, а соответствующие ЛЭ имеют более двух входных выводов.

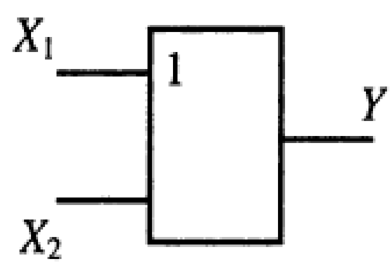
Операция ***инверсии***, называемая также операцией ***НЕ***, записывается в виде $Y=\overline{X}$ (читается «не икс»), имеет таблицу истинности и условное графическое обозначение (третий рисунок).

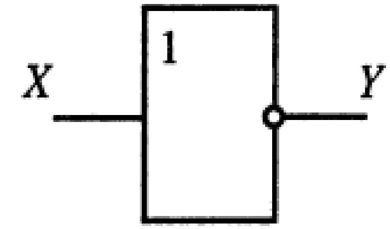
X_1	X_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

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

X	Y
1	0
0	1







Законы и правила

Коммутативный закон:

$$X1 + X2 = X2 + X1 \qquad X1 \cdot X2 = X2 \cdot X1$$

Ассоциативный закон:

$$X1 + (X2 + X3) = (X1 + X2) + X3 \qquad X1 \cdot (X2 \cdot X3) = (X1 \cdot X2) \cdot X3$$

Дистрибутивный закон:

$$X1 + X2 \cdot X3 = (X1 + X2) \cdot (X1 + X3) \qquad X1 \cdot (X2 + X3) = X1 \cdot X2 + X1 \cdot X3$$

Закон поглощения

$$X1 + X1 \cdot X2 = X1 \qquad X1 \cdot (X1 + X2) = X1$$

Правило повторения

$$X + X = X \qquad X \cdot X = X$$

Правило отрицания

$$X + \bar{X} = 1 \qquad X \cdot \bar{X} = 0$$

Правило двойного отрицания $X = \overline{(\bar{X})}$

Правило склеивания

$$(X1 + X2) \cdot (\overline{X1} + X2) = X2 \qquad X1 \cdot X2 + \overline{X1} \cdot X2 = X2$$

Теорема де Моргана

$$\overline{X1 + X2} = \overline{X1} \cdot \overline{X2}$$

Кроме того, определены следующие операции с нулем и единицей:

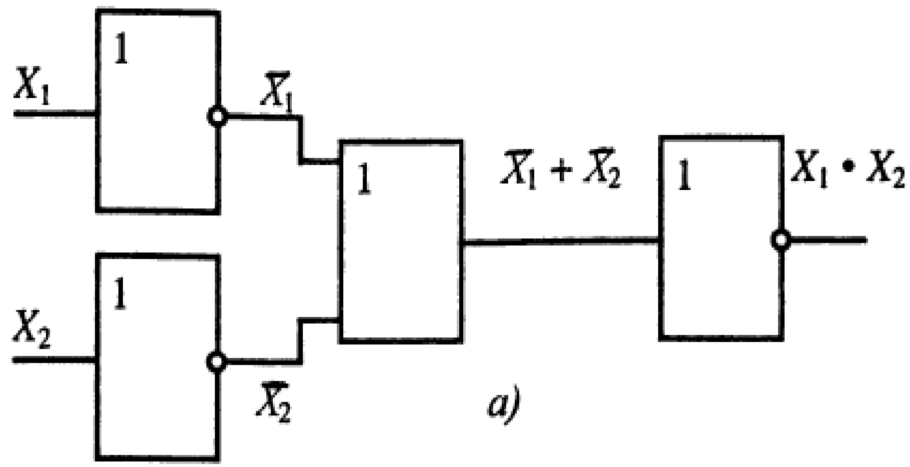
$X + 0 = X, \quad X \cdot 1 = X,$

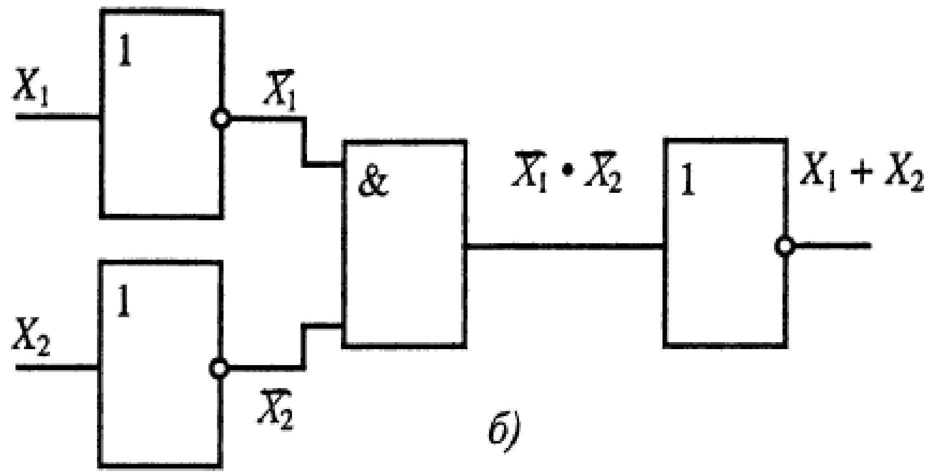
$X + 1 = 1, X \cdot 0 = 0,$

$\bar{1} = 0, \bar{0} = 1.$

Элементы алгебры логики

Операцию логического умножения можно заменить операцией логического сложения над инверсными значениями переменных с последующей инверсией результата: $X1 \cdot X2 = \overline{\overline{X1} + \overline{X2}}$. Вид логической схемы, реализующей операцию И на элементах ИЛИ и НЕ, показан на рисунке а. Аналогично можно исключить элемент ИЛИ: $X1 + X2 = \overline{\overline{X1} \cdot \overline{X2}}$ (рисунок б).

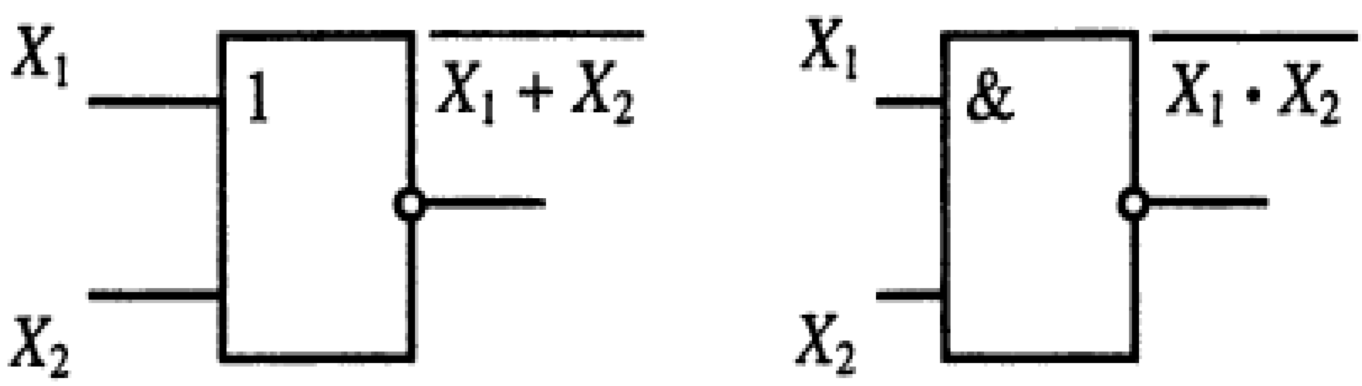




Элементы алгебры логики

Системы из логических элементов И, НЕ, либо ИЛИ, НЕ также являются функционально полными. По этой причине схемы, выполняющие операции И-НЕ, ИЛИ-НЕ, называются универсальными логическими элементами. Их условное графическое обозначение показано на рисунке а) и б), соответственно. Схема ***ИЛИ-НЕ*** на рисунке а) осуществляя логическую операцию $Y = \overline{X1 + X2}$, называемую также ***стрелкой Пирса*** и обозначаемую $Y = X1 \downarrow X2$. Схема ***И-НЕ*** на рисунке б) выполняет операцию $Y = \overline{X1 \cdot X2} = X1 \mid X2$, называемую ***штрихом Шеффера***.

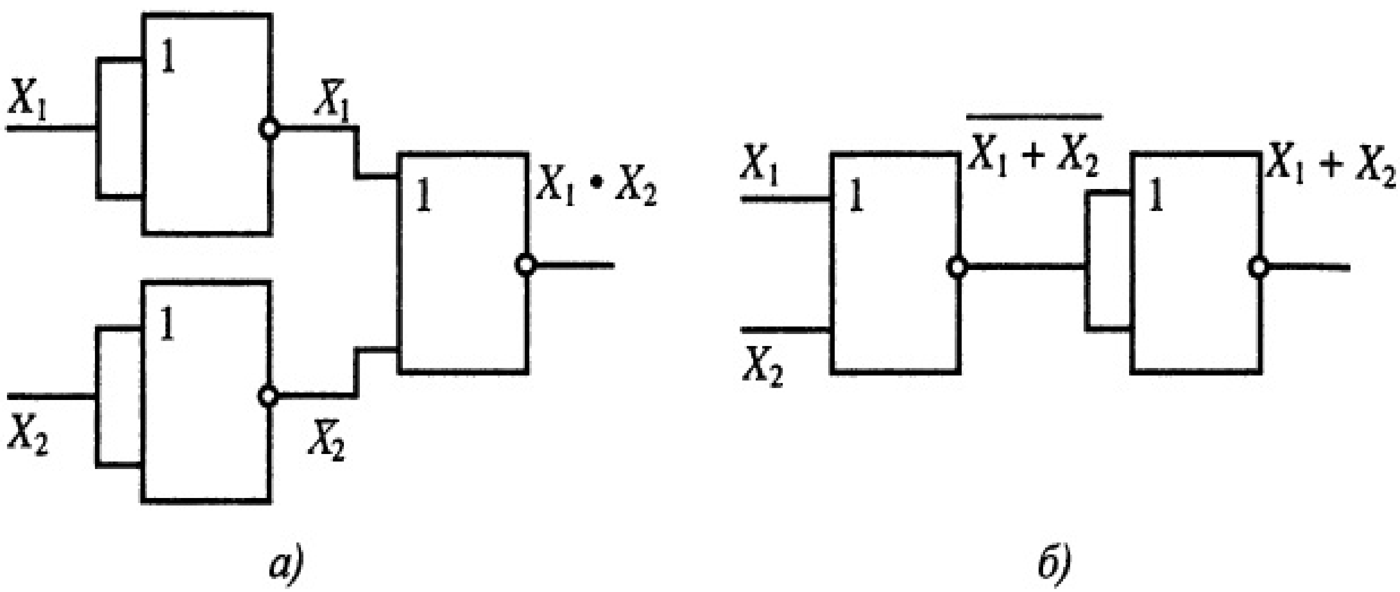
Функционально полная систем –набор элементов, используя который можно реализовать произвольную логическую функцию любой сложности.



а) б)

Элементы алгебры логики

Таким образом, реализация операции И на элементах ИЛИ-НЕ выглядит как показано на рисунке а). Нетрудно видеть, что схема на ЛЭ ИЛИ-НЕ, показанная на рисунке б), реализует операцию ИЛИ. Если по такой же схеме включить элементы И-НЕ, то будет реализована операция И.



Логические функции

Задача реализации заданной логической функции Y обычно оформляется в виде ***таблицы истинности***, называемой также ***таблицей переключений***, только уже не для двух, а для n переменных.

Ориентируясь на таблицу, ищут логическую функцию, которая бы соответствовала данной таблице. В общем случае одна и та же функция может быть реализована различными сочетаниями логических элементов. По этой причине прежде чем реализовать логическую функцию ее преобразуют в простую форму.

При переходе от таблично заданной функции к записи в виде логической суммы часто используют совершенную дизъюнктивную нормальную форму (СДНФ).

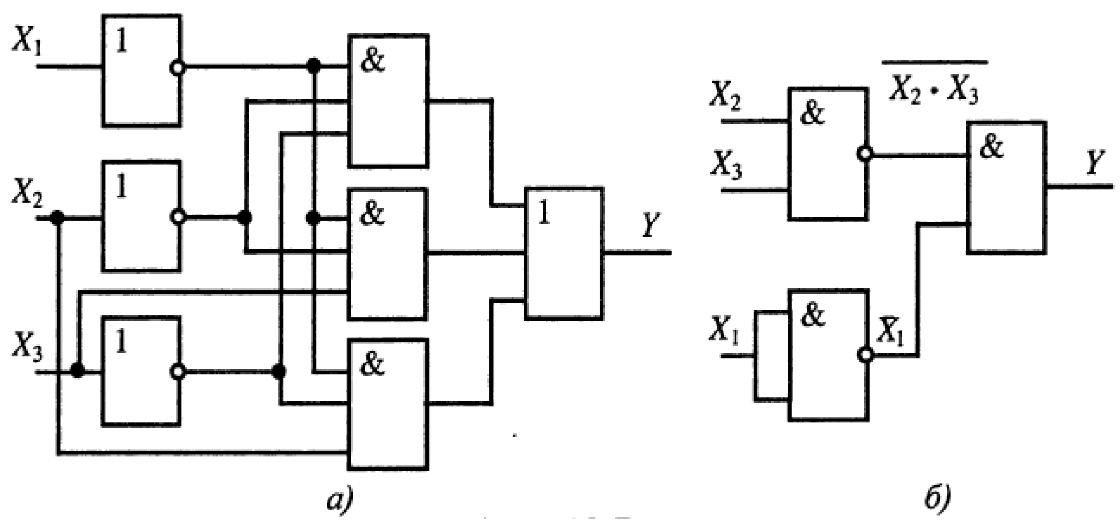
Пример

Пусть имеется таблица истинности для трех переменных (верхний рисунок). В соответствии со сделанными выше пояснениями выражение для функции Y в СДНФ будет иметь следующий вид:

$Y = \overline{X1} \cdot \overline{X2} \cdot \overline{X3} + \overline{X1} \cdot \overline{X2} \cdot X3 + \overline{X1} \cdot X2 \cdot \overline{X3}$.

X_1	X_2	X_3	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Данное выражение позволяет определить структуру логического устройства, которое реализует таблицу истинности. Оно должно содержать инвертор на каждую переменную, три схемы И на три входа и схему или также с тремя входами (рисунок а) и б)).



Синтез комбинационных логических устройств

К комбинационным схемам относятся схемы в которых отсутствуют элементы памяти и элементы задержки времени.

Разработка данных схем ведется в несколько этапов:

1. Задается словесное описание работы схемы.
2. Составляется таблица истинности.
3. Записывается исходная логическая функция и осуществляется ее минимизация.
4. Выполняется реализация минимизированной логической функции.

Основные параметры логических элементов

1. Каждый элемент реализует какую-то логическую операцию, поэтому к первому пункту отнесем конкретный набор логических функций данного элемента.

2. Конкретные значения напряжений U^1, U^0, отвечающие логической единице и логическому нулю, а также логический перепад.

3. Передаточная характеристика, т.е. зависимость Uвых от Uвх.

4. Помехоустойчивость.

5. Коэффициент объединения по входу Коб.

Коэффициент объединения по входу – это число входных сигналов, которые могут быть поданы на вход усилителя.

6. Коэффициент разветвления по выходу Кразв.

Коэффициент разветвления – это максимальное число таких же логических элементов, которые можно подключить параллельно к выходу данного логического элемента.

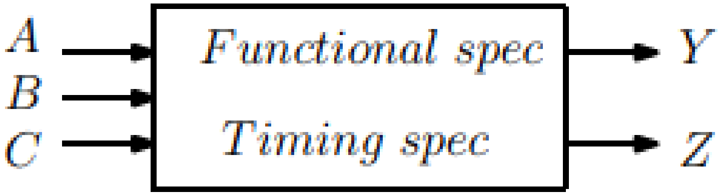
7. Быстродействие.

Быстродействие логического элемента характеризуется временем задержки выходного сигнала по отношению к входному при переключении элемента.

8. Средняя потребляемая мощность и работа переключения схемы.

Introduction

In digital electronics, a *circuit* is a network that processes discrete valued variables.



- A, B, C are inputs;
- Y, Z are outputs;
- a *functional specification* describing the relationship between inputs and outputs;
- a *timing specification* describing the delay between inputs changing and outputs responding.

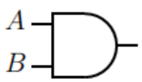
Digital circuits are classified as *combinational* or *sequential:*

- a combinational circuit's outputs depend only on the current values of the inputs;
- a sequential circuit's outputs depend on both current and previous values of the inputs (sequential circuits have memory).

Introduction

Examples of combinational (logic) blocks and sequential blocks.

Combinational (logic) blocks

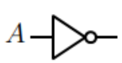


$Y = A \cdot B$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$Y = A + B$

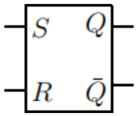
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1



$Y = \bar{A}$

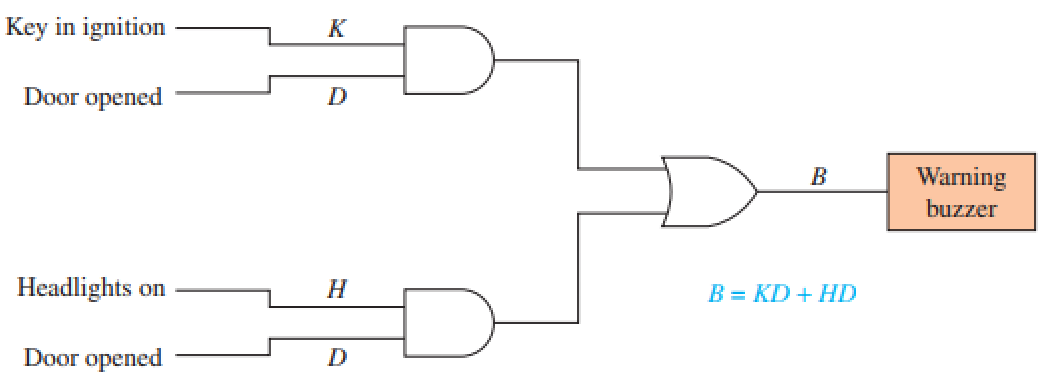
A	Y
0	0
1	1

Sequential blocks

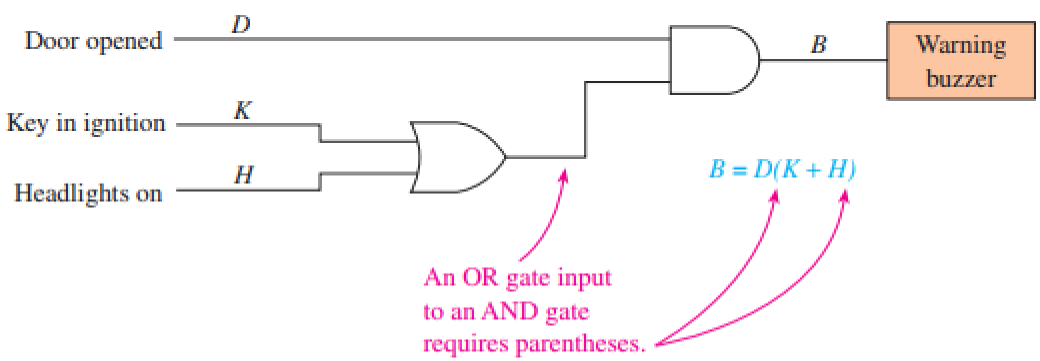


Introduction

Combinational logic employs the use of two or more of the basic logic gates to form a more useful, complex function.



Boolean algebra



Boolean algebra is a mathematical system with logic notation used to describe different interconnections of digital circuits.

Basic Boolean Identities - basic Identities in Boolean algebra

№	Identity
1	X+0=X
2	X+1=1
3	X+X=X
4	X+$\bar{X}$=1
5	X·0=0
6	X·1=X
7	X·X=X
8	X·$\bar{X}$=0
9	$\bar{\bar{X}} = X$

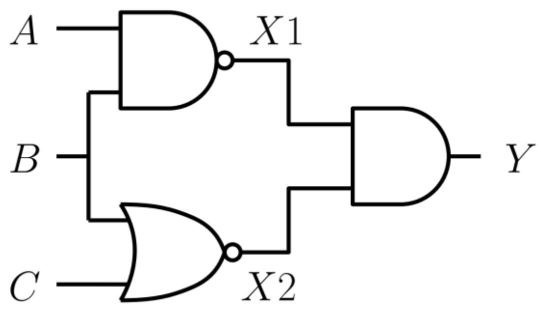
Boolean algebra

Basic Boolean Identities

№	Identity	Comments
1	X+Y=Y+X	Commutative
2	X·Y=Y·X	Commutative
3	X+(Y+Z)=(X+Y)+Z	Associative
4	X·(Y·Z)=(X·Y)·Z	Associative
5	X·(Y+Z)=X·Y+X·Z	Distributive
6	X+Y·Z=(X+Y)·(X+Z)	Distributive
7	X+X·Y=X	Absorption
8	X·(X+Y)=X	Absorption
9	$X \cdot Y + \bar{X} \cdot Z + Y \cdot Z = X \cdot Y + \bar{X} \cdot Z$	Consensus
10	$\overline{X+Y+Z} = \bar{X} \cdot \bar{Y} \cdot \bar{Z}$	DeMorgan
	$\overline{X \cdot Y \cdot Z} = \bar{X} + \bar{Y} + \bar{Z}$	DeMorgan

Example

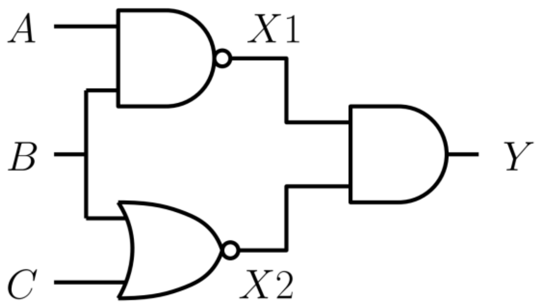
Derive the Boolean function for the combinational network



$Y = ???$

Example

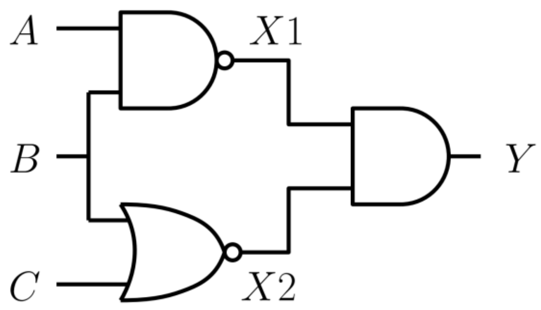
Derive the Boolean function for the combinational network



$$Y = X1 \cdot X2$$

Example

Derive the Boolean function for the combinational network



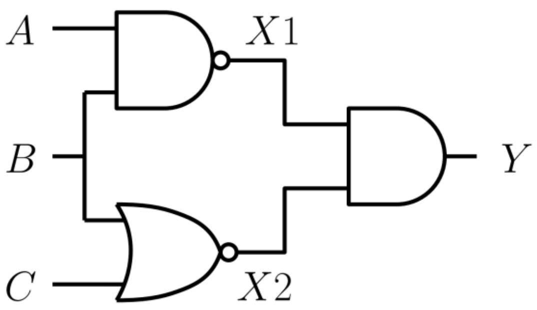
$$Y = X1 \cdot X2$$

$$X1 = \overline{A \cdot B}$$

$$X2 = \overline{B + C}$$

Example

Derive the Boolean function for the combinational network



$$Y = X1 \cdot X2$$

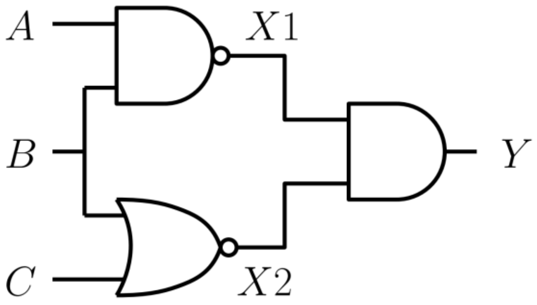
$$X1 = \overline{A \cdot B}$$

$$X2 = \overline{B + C}$$

$$Y = \overline{AB} \cdot \overline{B + C}$$

Example

Derive the Boolean function for the combinational network



$$Y = X1 \cdot X2$$

$$X1 = \overline{A \cdot B}$$

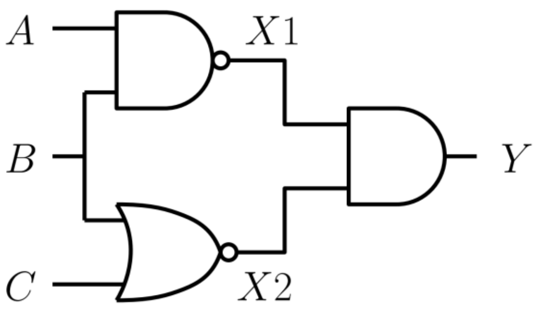
$$X2 = \overline{B + C}$$

$$Y = \overline{AB} \cdot \overline{B + C}$$

Is it the final result?

Example

Derive the Boolean function for the combinational network



$$Y = X1 \cdot X2$$

$$X1 = \overline{A \cdot B}$$

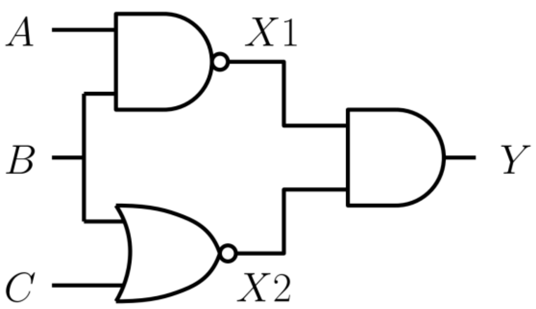
$$X2 = \overline{B + C}$$

$$Y = \overline{AB} \cdot \overline{B + C}$$

Is it the final result?

NO, we can make some transformations…

Example



$$Y = \overline{AB} \cdot \overline{B + C}$$

We are going to apply De Morgan’s theorem.

In the form of an equation, **De Morgan’s theorem** is stated as follows:

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

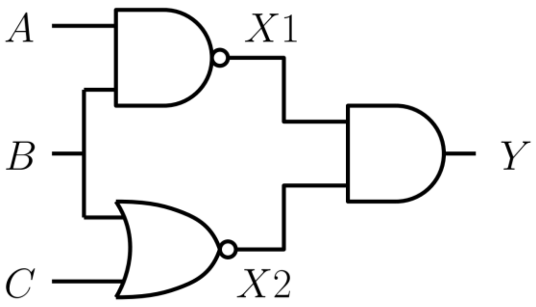
$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

We would like to have the final equation in a form called the **sum-of-products (SOP) form**.

Also we can obtain the final equation in a form called the **Product-of-sums (POS) form.**

We will find more simple equation for our case in the SOP form, this form is very useful form for building truth tables and Karnaugh maps

Example

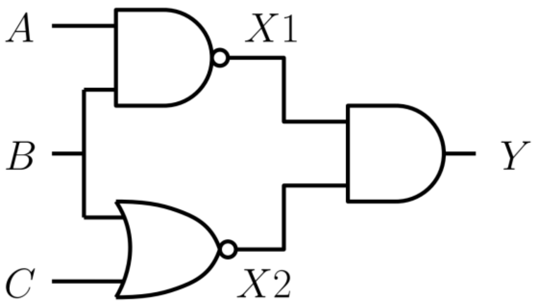


$$Y = \overline{AB} \cdot \overline{B + C}$$

First step – We apply De Morgan’s theorem

$$Y = \overline{AB} \cdot \overline{B + C} = (\bar{A} + \bar{B}) \cdot (\bar{B} \cdot \bar{C})$$

Example



$$Y = \overline{AB} \cdot \overline{B + C}$$

First step – We apply De Morgan's theorem

$$Y = \overline{AB} \cdot \overline{B + C} = (\bar{A} + \bar{B}) \cdot (\bar{B} \cdot \bar{C})$$

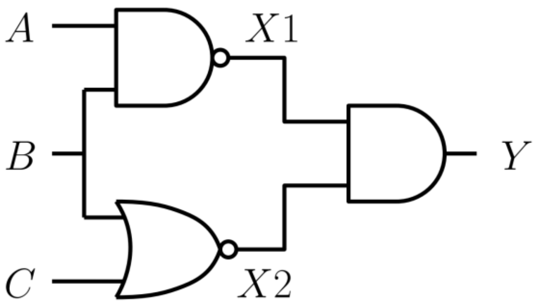
Second step – We use Boolean algebra rules

1) X·(Y+Z)=X·Y+X·Z

5	X·(Y+Z)=X·Y+X·Z	Distributive

$$Y = \bar{A} \cdot \bar{B} \cdot \bar{C} + \bar{B} \cdot \bar{B} \cdot \bar{C}$$

Example



$$Y = \overline{AB} \cdot \overline{B + C}$$

First step – We apply De Morgan's theorem

$$Y = \overline{AB} \cdot \overline{B + C} = (\bar{A} + \bar{B}) \cdot (\bar{B} \cdot \bar{C})$$

Second step – We use Boolean algebra rules

1) X·(Y+Z)=X·Y+X·Z

$$Y = \bar{A} \cdot \bar{B} \cdot \bar{C} + \bar{B} \cdot \bar{B} \cdot \bar{C}$$

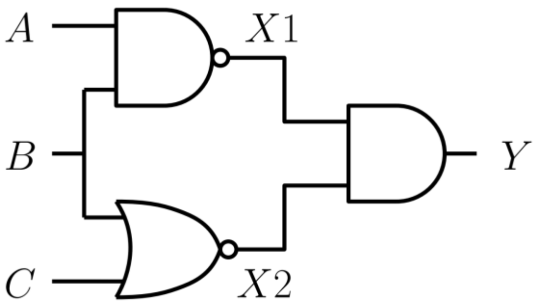
2) X·X=X

7	X·X=X

$$Y = \bar{A} \cdot \bar{B} \cdot \bar{C} + \bar{B} \cdot \bar{C}$$

$$Y = \bar{B} \cdot \bar{C} + (\bar{A} + 1)$$

Example



$$Y = \overline{AB} \cdot \overline{B + C}$$

First step – We apply De Morgan's theorem

$$Y = \overline{AB} \cdot \overline{B + C} = (\bar{A} + \bar{B}) \cdot (\bar{B} \cdot \bar{C})$$

Second step – We use Boolean algebra rules

1) X·(Y+Z)=X·Y+X·Z

$$Y = \bar{A} \cdot \bar{B} \cdot \bar{C} + \bar{B} \cdot \bar{B} \cdot \bar{C}$$

2) X·X=X

$$Y = \bar{A} \cdot \bar{B} \cdot \bar{C} + \bar{B} \cdot \bar{C}$$

$$Y = \bar{B} \cdot \bar{C} + (\bar{A} + 1)$$

3) X+1=1

$$Y = \bar{B} \cdot \bar{C}$$

This is the final result!!!

Example

True table for example

A	B	C	$Y = \overline{AB} \cdot \overline{B + C}$	$Y = \bar{B} \cdot \bar{C}$
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Karnaugh mapping

Boolean algebra and De Morgan’s theorem, let us to minimize Boolean functions, but we need have a lot of practice for make more simple solution.

Karnaugh mapping is a *systematic approach*, which will always produce the simplest configuration possible for the logic circuit.

Karnaugh mapping

Boolean algebra and De Morgan’s theorem, let us to minimize Boolean functions, but we need have a lot of practice for make more simple solution.

Karnaugh mapping is a *systematic approach*, which will always produce the simplest configuration possible for the logic circuit.

In 1953 Maurice Karnaug puplished an article about system papping and minimizing Boolean expressions.

Karnaugh, Maurice (November 1953). "The Map Method for Synthesis of Combinational Logic Circuits". Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics. 72 (5): 593–599. doi:10.1109/TCE.1953.6371932

Karnaugh mapping

Boolean algebra and De Morgan’s theorem, let us to minimize Boolean functions, but we need have a lot of practice for make more simple solution.

Karnaugh mapping is a *systematic approach*, which will always produce the simplest configuration possible for the logic circuit.

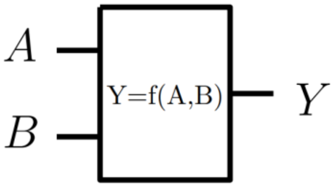
In 1953 Maurice Karnaug puplished an article about system papping and minimizing Boolean expressions.

Karnaugh, Maurice (November 1953). "The Map Method for Synthesis of Combinational Logic Circuits". Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics. 72 (5): 593–599. doi:10.1109/TCE.1953.6371932

That is the Karnaugh map and how can we use it???

Karnaugh mapping

Karnaugh map for two input parameters.

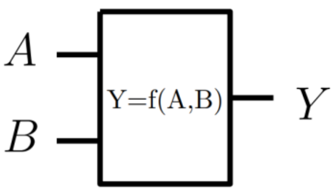


A	B	Y
0	0	$\bar{A}\bar{B}$
0	1	$\bar{A}B$
1	0	$A\bar{B}$
1	1	AB

	$\bar{B}$	B
$\bar{A}$	1	2
A	3	4

Karnaugh mapping

Karnaugh map for two input parameters.



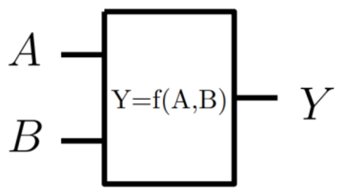
A	B	Y
0	0	$\bar{A}\bar{B}$
0	1	$\bar{A}B$
1	0	$A\bar{B}$
1	1	AB

	$\bar{B}$	B
$\bar{A}$	1	2
A	3	4

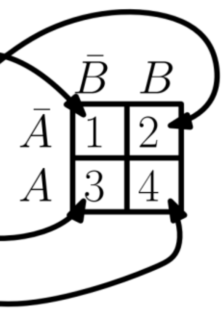
And how can we use it???

Karnaugh mapping

Karnaugh map for two input parameters.



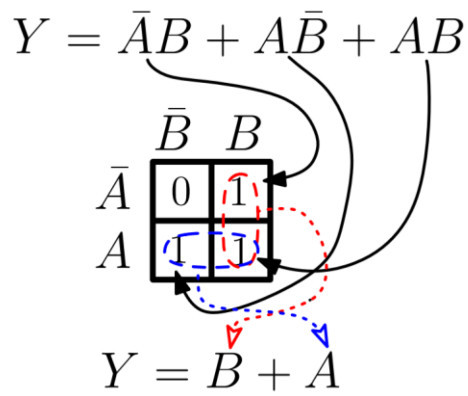
A	B	Y
0	0	$\bar{A}\bar{B}$
0	1	$\bar{A}B$
1	0	$A\bar{B}$
1	1	AB



And how can we use it???

Simple example – consider Boolean equation

$Y = \bar{A}B + A\bar{B} + AB$



Karnaugh mapping

Consider the Karnaugh map for three input parameters.

$$Y = A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + AB\bar{C}$$

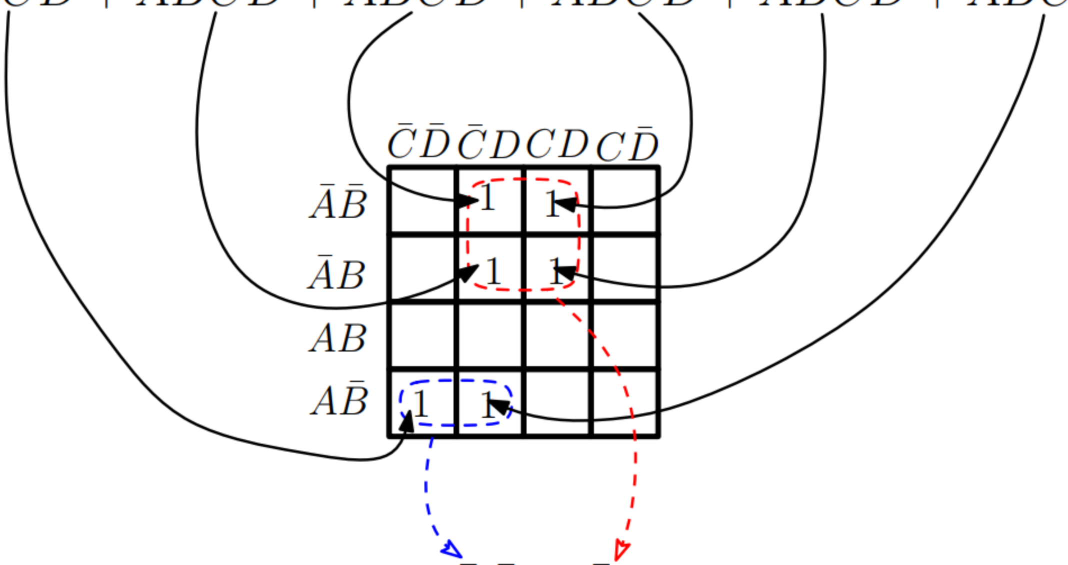
	$\bar{C}$	C
$\bar{A}\bar{B}$	1	1
$\bar{A}B$		
AB	1	
$A\bar{B}$	1	

$$Y = A\bar{C} + \bar{A}\bar{B}$$

Karnaugh mapping

Consider the Karnaugh map for four input parameters.

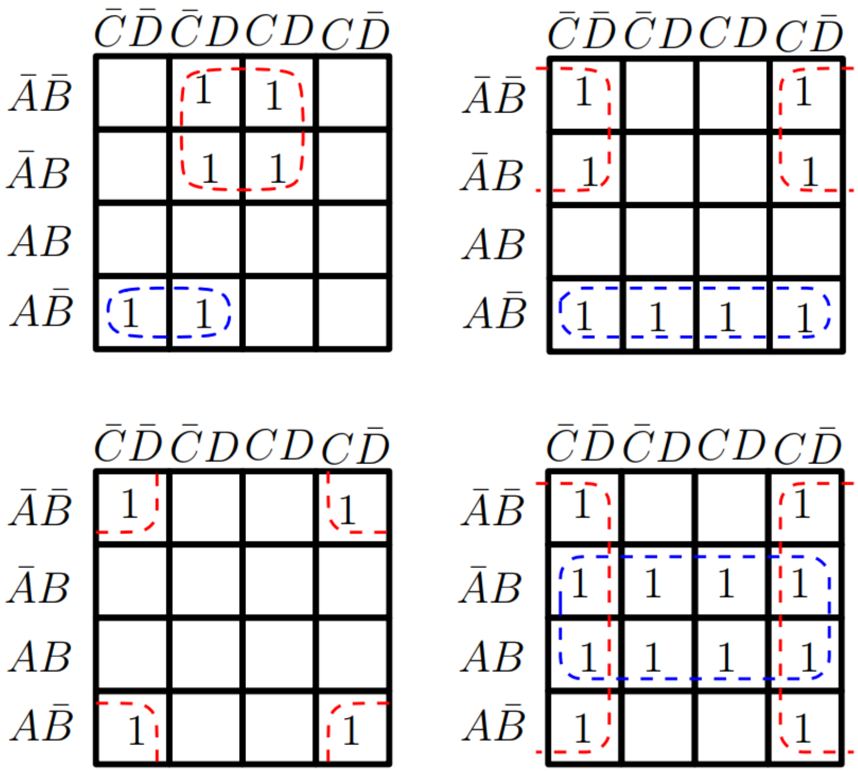
$$Y = A\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}CD + \bar{A}BCD + A\bar{B}\bar{C}D$$



$$Y = A\bar{B}\bar{C} + \bar{A}D$$

Karnaugh mapping

Which looping technics can we use in the Karnaugh map?



Example

$$Y = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C} + ABC$$

	$\bar{C}$	C
$\bar{A}\bar{B}$		
$\bar{A}B$	1	1
AB	1	1
$A\bar{B}$	1	1

$$Y = A + B$$

Example

$$Y = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C} + ABC$$

A	B	C	Y

$$Y = A + B$$

A	B	Y

УНИВЕРСИТЕТ ИТМО

Спасибо за внимание!

Санкт-Петербург, 2021