Boolean Algebras

Mongi BLEL

King Saud University

August 30, 2019

Table of contents

Definition

A Boolean algebra is a set B with two binary operations $\vee$ and $\wedge$, elements 0 and 1, and a unary operation $\bar{\ }$ such that these properties hold for all x, y, and z in B:

Identity laws $\begin{cases} x \vee 0 = x \\ x \wedge 1 = x \end{cases}$

Complement laws $\begin{cases} x \vee \bar{x} = 1 \\ x \wedge \bar{x} = 0 \end{cases}$

Associative laws $\begin{cases} (x \vee y) \vee z = x \vee (y \vee z) \\ (x \wedge y) \wedge z = x \wedge (y \wedge z) \end{cases}$

Commutative laws $\begin{cases} x \vee y = y \vee x \\ x \wedge y = y \wedge x \end{cases}$

Distributive laws $\begin{cases} x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z) \\ x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z) \end{cases}$

If we denote $\vee$ by $+$ and $\wedge$ by ., we have the following:

Identity laws $\begin{cases} x + 0 = x \\ x.1 = x \end{cases}$

Complement laws $\begin{cases} x + \bar{x} = 1 \\ x.\bar{x} = 0 \end{cases}$

Associative laws $\begin{cases} (x + y) + z = x + (y + z) \\ (x.y).z = x.(y.z) \end{cases}$

Commutative laws $\begin{cases} x + y = y + x \\ x.y = y.x \end{cases}$

Distributive laws $\begin{cases} x + (y.z) = (x + y).(x + z) \\ x.(y + z) = (x.y) + (x.z) \end{cases}$

From our previous discussion, $B = \{0, 1\}$. The three operations in this Boolean algebra are

- The complement of an element, denoted with a bar, is defined by
$$\overline{0} = 1 \text{ and } \overline{1} = 0$$
- The Boolean sum, denoted by $+$ or by OR, has the following values:
$$1 + 1 = 1, 1 + 0 = 1, 0 + 1 = 1, 0 + 0 = 0.$$
- The Boolean product, denoted by . or by AND, has the following values:
$$1.1 = 1, 1.0 = 0, 0.1 = 0, 0.0 = 0.$$

Example

Using the definitions of complementation, the Boolean sum, and the Boolean product, it follows that
$1.0 + \overline{(0+1)} = 0 + \overline{1} = 0 + 0 = 0$

The complement, Boolean sum, and Boolean product correspond to the logical operators, $\neg$, $\vee$ and $\wedge$, respectively, where 0 corresponds to **F** (false) and 1 corresponds to **T** (true). Equalities in Boolean algebra can be directly translated into equivalences of compound propositions. Conversely, equivalences of compound propositions can be translated into equalities in Boolean algebra.We will see later in this section why these translations yield valid logical equivalences and identities in Boolean algebra. Example 2 illustrates the translation from Boolean algebra to propositional logic.

Example

Translate $1.0 + \overline{(0 + 1)}$, the equality found in Example 1, into a logical equivalence.
We obtain a logical equivalence when we translate each 1 into a T, each 0 into an F, each Boolean sum into a disjunction, each Boolean product into a conjunction, and each complementation into a negation.We obtain $(T \wedge F) \vee \neg(T \vee F) \equiv F$.

Example

The following example illustrates the translation from propositional logic to Boolean algebra.
Translation of the logical equivalence $(T \wedge T) \vee \neg F \equiv T$ into an identity in Boolean algebra.
We obtain an identity in Boolean algebra when we translate each T into a 1, each F into a 0, each disjunction into a Boolean sum, each conjunction into a Boolean product, and each negation into a complementation. We obtain $(1.1) + \overline{0} = 1$.

Identities of Boolean Algebra

There are many identities in Boolean algebra. The most important of these are displayed in Table 5. These identities are particularly useful in simplifying the design of circuits.

Boolean Identities	
Identity	Name
$\bar{\bar{x}} = x$	Low of double complement
$x + x = x$ $x.x = x$	Idempotent lows
$x + 0 = x$ $x.1 = x$	Identity lows
$x + 1 = 1$ $x.0 = 0$	Domination lows

Boolean Identities	
Identity	Name
$x+y=y+x$ $x.y=y.x$	Commutative lows
$x+(y+z)=(x+y)+z$ $x.(y.z)=(x.y).z$	Associative lows
$x+(y.z)=(x+y).(x+z)$ $x.(y+z)=x.y+x.z$	Distributive lows
$\overline{(x+y)}=\bar{x}.\bar{y}$ $\overline{(x.y)}=\bar{x}+\bar{y}$	De Morgan's lows
$x+x.y=x$ $x.(x+y)=x$	Absorption lows
$x+\bar{x}=1$	Unit property
$x.\bar{x}=0$	Zero property

Example

Show that the distributive law $x(y+z) = xy + xz$ is valid.
The verification of this identity is shown in Table ??. The identity holds because the last two columns of the table agree.

Verification of one of the Distributive Laws

x	y	z	$y+z$	xy	xz	$x(y+z)$	$xy+xz$
1	1	1	1	1	1	1	1
1	1	0	1	1	0	1	1
1	0	1	1	0	1	1	1
1	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0

Boolean Expressions and Boolean Functions

Let B be a boolean algebra and $B_n = \{(x_1, \ldots, x_n);\ x_1, \ldots, x_n \in B\}$. The variable x is called a **Boolean variable** if it assumes values only from B. A function from B_n to B is called a **Boolean function of degree** n.

Example

The function $F(x, y) = x\bar{y}$ from the set of ordered pairs of Boolean variables to the set $\{0, 1\}$ is a Boolean function of degree 2 with $F(1, 1) = 0, F(1, 0) = 1, F(0, 1) = 0$, and $F(0, 0) = 0$. We display these values of F in the following Table.

x	y	F(x,y)
1	1	0
1	0	1
0	1	0
0	0	0

1. Two different Boolean expressions that represent the same function are called **equivalent**.
2. For instance, the Boolean expressions xy, $xy + 0$, and $xy.1$ are equivalent.
3. The **complement** of the Boolean function F is the function $\overline{F}$, where $\overline{F}(x_1, \cdots, x_n) = \overline{F(x_1, \cdots, x_n)}$.
4. Let F and G be Boolean functions of degree n. The **Boolean sum** $F + G$ and the **Boolean product** FG are defined by $(F + G)(x_1, \cdots, x_n) = F(x_1, \cdots, x_n) + G(x_1, \cdots, x_n)$, $(FG)(x_1, \cdots, x_n) = F(x_1, \cdots, x_n)G(x_1, \cdots, x_n)$.

Representation of Boolean Functions

Two important problems of Boolean algebra will be studied in this section. The first problem is: Given the values of a Boolean function, how can a Boolean expression that represents this function be found? This problem will be solved by showing that any Boolean function can be represented by a **Boolean sum of Boolean products** of the variables and their complements. The solution of this problem shows that every Boolean function can be represented using the three Boolean operators $., +,$ and $\_$. The second problem is: Is there a smaller set of operators that can be used to represent all Boolean functions? We will answer this question by showing that all Boolean functions can be represented using only one operator. Both of these problems have practical importance in circuit design.

Sum-of-Products Expansions

Let $F(x, y, z)$ and $G(x, y, z)$ be the Boolean functions defined by the following table.

x	y	z	F(x,y,z)	G(x,y,z)
1	1	1	0	0
1	1	0	0	1
1	0	1	1	0
1	0	0	0	1
0	1	1	1	0
0	1	0	0	1
0	0	1	1	0
0	0	0	0	1

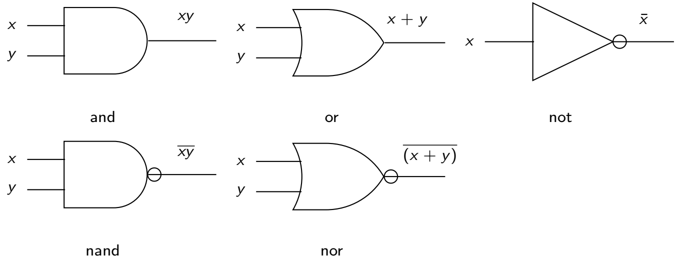
The expressions of these functions are:

$F(x, y, z) = x\bar{y}z + \bar{x}yz + \bar{x}\bar{y}z,$

$G(x, y, z) = xy\bar{z} + x\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}\bar{y}\bar{z}.$

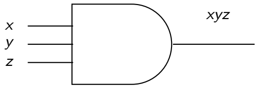
Logic Gates

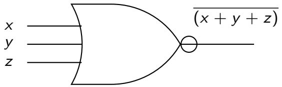
A logic gate is an electronic circuit or device which makes the logical decisions. To arrive at this decisions, the most common logic gates used are OR, AND, NOT, NAND, and NOR gates. The NAND and NOR gates are called universal gates. The exclusive-OR gate is another logic gate which can be constructed using AND, OR and NOT gate. Logic gates are also called switches.



Remark

The associative laws show that there is no ambiguity about a term such as $x + y + z$ or xyz, so we can introduce multiple-input primitive gates:





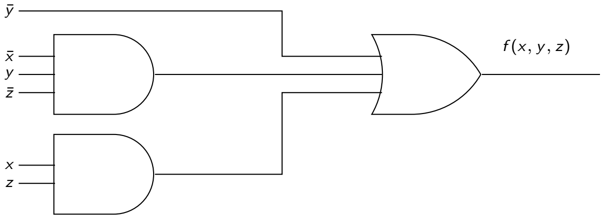
Any logic function can be implemented using NAND gates. To achieve this, first the logic function has to be written in Sum of Product (SOP) form. Once logic function is converted to SOP, then is very easy to implement using NAND gate. In other words any logic circuit with AND gates in first level and OR gates in second level can be converted into a NAND-NAND gate circuit. Any logic function can be implemented using NOR gates. To achieve this, first the logic function has to be written in Product of Sum (POS) form. Once it is converted to POS, then it's very easy to implement using NOR gate. In other words any logic circuit with OR gates in first level and AND gates in second level can be converted into a NOR-NOR gate circuit.

Example

For example, let f be the following Boolean function:

$$f(x,y,z) = \bar{y} + \bar{x}y\bar{z} + xz.$$

This Boolean function can be implemented with the following circuit.



Remarks

1. We can write expressions in many ways, but some ways are more useful than others.
2. A sum of products (SOP) expression is characterized by:
   - There are only OR (sum) operations at the "outermost" level.
   - Each term in the sum must be a product of literals.
3. The advantage is that a sum of products expression can be implemented using a fairly simple two-level circuit:
   Literals are at the "0th" level.
   AND gates are at the first level.
   A single OR gate is at the second level.

Definition

A **literal** is a Boolean variable or its complement.
A **minterm** of the Boolean variables $x_1, x_2, \cdots, x_n$ is a Boolean product $y_1 y_2 \cdots y_n$, where $y_i = x_i$ or $y_i = \overline{x_i}$. Hence, a minterm is a product of n literals, with one literal for each variable.

For example, a three-variable function like $f(x, y, z)$ has up to 8 minterms:

$$xyz,\ xy\bar{z},\ x\bar{y}z,\ x\bar{y}\bar{z},\ \bar{x}yz,\ \bar{x}y\bar{z},\ \bar{x}\bar{y}z,\ \bar{x}\bar{y}\bar{z}$$

Every function can be written as a sum of minterms, which is a special kind of sum of products form.
The sum of minterms form for any function is unique. This sum is called the complete sum of product (CSP) of the function.
If you have a truth table for a function, you can write a sum of minterms expression just by picking out the rows of the table where the function output is 1.

Definition

- We denote by **CSP** form of Boolean function its (**Complete sum-of-product**).
- We denote by **CPS** form of Boolean function its (**Complete product-of-sum**). This form is obtained by giving the CSP for the complement of the function, and we take the complement of the CSP give the CPS.

• Every Boolean function can be written in many different ways, so it's sometimes useful to use standard representations like sums of products or sums of minterms.

• Every Boolean expression can be converted to a circuit.

Now we look at a graphical technique for simplifying an expression into a minimal sum of products (MSP) form:

•There are a minimal number of product terms in the expression.

• Each term has a minimal number of literals.

The Karnaugh map

- To reduce the number of terms in a Boolean expression representing a circuit, it is necessary to find terms to combine. There is a graphical method, called a **Karnaugh map** or **K-map**, for finding terms to combine for Boolean functions involving a relatively small number of variables.
- We will first illustrate how K-maps are used to simplify expansions of Boolean functions in two variables.
- We will continue by showing how K-maps can be used to minimize Boolean functions in three variables and then in four variables.

There are four possible minterms in the sum-of-products expansion of a Boolean function in the two variables x and y. A K-map for a Boolean function in these two variables consists of four cells, where a 1 is placed in the cell representing a minterm if this minterm is present in the expansion. Cells are said to be **adjacent** if the minterms that they represent differ in exactly one literal. For instance, the cell representing $\overline{x}y$ is adjacent to the cells representing xy and $\overline{xy}$. The four cells and the terms that they represent are shown in Figure ??.

	y	$\bar{y}$
x	xy	$x\bar{y}$
$\bar{x}$	$\bar{x}y$	$\bar{x}\bar{y}$

K-maps in Two Variables.

A K-map in three variables is a rectangle divided into eight cells. The cells represent the eight possible minterms in three variables. Two cells are said to be adjacent if the minterms that they represent differ in exactly one literal. One of the ways to form a K-map in three variables is shown in Figure ??.

	yz	$y\bar{z}$	$\bar{y}\bar{z}$	$\bar{y}z$
x	xyz	$xy\bar{z}$	$x\bar{y}\bar{z}$	$x\bar{y}z$
$\bar{x}$	$\bar{x}yz$	$\bar{x}y\bar{z}$	$\bar{x}\bar{y}\bar{z}$	$\bar{x}\bar{y}z$

K-maps in Three Variables.

A K-map in four variables is a rectangle divided into sixteen cells. The cells represent the sixteen possible minterms in three variables. Two cells are said to be adjacent if the minterms that they represent differ in exactly one literal. One of the ways to form a K-map in three variables is shown in Figure ??.

	zw	$z\bar{w}$	$\bar{z}\bar{w}$	$\bar{z}w$
xy	$xyzw$	$xyz\bar{w}$	$xy\bar{z}\bar{w}$	$xy\bar{z}w$
$x\bar{y}$	$x\bar{y}zw$	$x\bar{y}z\bar{w}$	$x\bar{y}\bar{z}\bar{w}$	$x\bar{y}\bar{z}w$
$\bar{x}\bar{y}$	$\bar{x}\bar{y}zw$	$\bar{x}\bar{y}z\bar{w}$	$\bar{x}\bar{y}\bar{z}\bar{w}$	$\bar{x}\bar{y}\bar{z}w$
$\bar{x}y$	$\bar{x}yzw$	$\bar{x}yz\bar{w}$	$\bar{x}y\bar{z}\bar{w}$	$\bar{x}y\bar{z}w$

K-maps in Four Variables.

Definition

- We denote by **MSP** form (**Minimal sum-of-product**), is obtained using K-maps method.
- We denote by **MPS** form (**Minimal product-of-sum**), is obtained by giving the CSP for the complement of the function, and we take the complement of the CSP give the CPS.

Grouping terms in the K-map: The power of K-maps is in minimizing the terms, K-maps can be minimized with the help of grouping the terms to form single terms.
When forming groups of squares, consider the following:

1. Every square containing 1 must be considered at least once.
2. A square containing 1 can be included in as many groups as desired.
3. A group must be as large as possible.
4. If a square containing 1 cannot be placed in a group, then leave it out to include in final expression.
5. The number of squares in a group must be equal to 2 .i.e. $2, 4, 8, .$

6. The map is considered to be folded or spherical, therefore squares at the end of a row or column are treated as adjacent squares.
7. The simplified logic expression obtained from a K-map is not always unique. Groupings can be made in different ways.
8. Before drawing a K-map the logic expression must be in canonical form.

Example

Find the K-maps for $f(x,y) = xy + \overline{x}y$, $g(x,y) = x\overline{y} + \overline{x}y$, and $h(x,y) = x\overline{y} + \overline{x}y + \overline{x}\ \overline{y}$.
The three K-maps of these functions is as follows respectively:

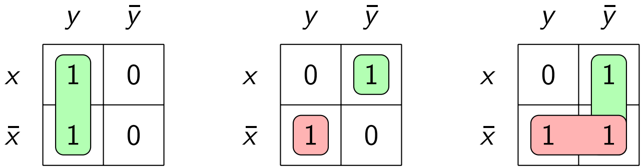


figure 1: K-maps for the Sum-of-Products Expansions

Using the K-maps, the minimal expansions for these Boolean functions are $f(x, y) = y$, $g(x, y) = x\overline{y} + \overline{x}y$, and $h(x, y) = \overline{x} + \overline{y}$.

Example

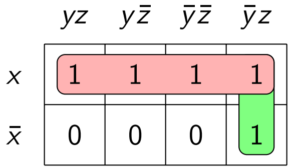
Let $f(x, y, z) = xy + \bar{y}z + xz$.

• First, we look for the (CSP) form of f.

Here is the truth table and sum of minterms for f

x	y	z	f(x,y,z)
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	0
0	1	1	0
0	1	0	0
0	0	1	1
0	0	0	0

The K-map of f is:



	yz	$y\bar{z}$	$\bar{y}\bar{z}$	$\bar{y}z$
x	1	1	1	1
$\bar{x}$	0	0	0	1

The (MSP) of f is $f(x, y, z) = x + \bar{y}z$.

Example

Let f be the following Boolean function: $f(x,y,z) = x + \overline{y}(\overline{x} + z)$

	yz	$y\bar{z}$	$\bar{y}\bar{z}$	$\bar{y}z$
x	1	1	1	1
$\bar{x}$	0	0	1	1

The (CSP) of f is $xyz + xy\bar{z} + x\bar{y}\bar{z} + x\bar{y}z + \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z$.
The (CSP) of $\bar{f}$ is $\bar{x}yz + \bar{x}y\bar{z}$.
The (MSP) of f is $\bar{x}\bar{y} + x$.
The (MPS) of f is $(x + \bar{y} + \bar{z}).(x + \bar{y} + z)$.

Example

Let g be the following Boolean function:
$g(x, y, z) = \overline{x}(x + \overline{y} + y\overline{z})$.

	yz	$y\bar{z}$	$\bar{y}\bar{z}$	$\bar{y}z$
x	0	0	0	0
$\bar{x}$	0	1	1	1

The (CSP) of g is $\bar{x}y\bar{z} + \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z$.
The (CSP) of $\bar{g}$ is $\bar{x}yz + xyz + xy\bar{z} + x\bar{y}\bar{z} + x\bar{y}z$.
The (MSP) of g is $\bar{x}y\bar{z} + \bar{x}\bar{y}$.
The (MPS) of g is
$(x + \bar{y} + z).(\bar{x} + \bar{y} + \bar{z}).(\bar{x} + \bar{y} + z).(\bar{x} + y + z).(\bar{x} + y + \bar{z})$.

Example

We consider the Boolean function of degree 4:
$f(x, y, z, w) = x\bar{y}zw + xy\bar{z}w + \bar{x}yz\bar{w}$.

	zw	$z\bar{w}$	$\bar{z}\bar{w}$	$\bar{z}w$
xy	0	$\boxed{1}$	0	0
$x\bar{y}$	0	1	1	$\boxed{1}$
$\bar{x}\bar{y}$	0	0	0	0
$\bar{x}y$	$\boxed{1}$	0	0	0

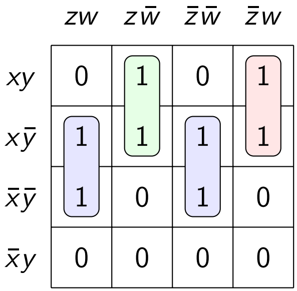
The (CSP) of f is $x\bar{y}zw + xy\bar{z}w + \bar{x}yz\bar{w}$.
The (CSP) of $\bar{f}$ is $\bar{x}yzw + \bar{x}\bar{y}zw + \bar{x}yzw + xyz\bar{w} + x\bar{y}z\bar{w} + \bar{x}\bar{y}z\bar{w} + xy\bar{z}\bar{w} + x\bar{y}\bar{z}\bar{w} + \bar{x}\bar{y}\bar{z}\bar{w} + \bar{x}y\bar{z}\bar{w} + x\bar{y}\bar{z}w + \bar{x}\bar{y}\bar{z}w + \bar{x}y\bar{z}w$.
The (MSP) of f is $x\bar{y}zw + \bar{x}yz\bar{w} + xy\bar{z}w$.
The (MPS) of f is $(\bar{x} + \bar{y} + \bar{z}).(z + w).(x + \bar{w}).(y + w).(y + z)$.

Example

Consider the following Boolean function:

$$f(x,y,z) = xyz\bar{w} + xy\bar{z}w + x\bar{y} + \bar{x}\bar{y}zw + \bar{x}\bar{y}\bar{z}\bar{w}.$$

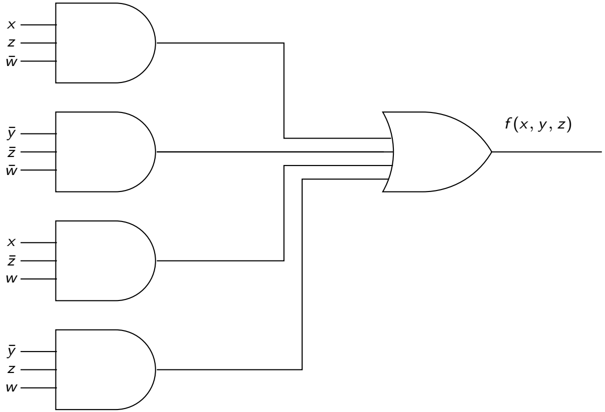
The Karnaugh -map for $f(x,y,z,w)$ is:



	zw	$z\bar{w}$	$\bar{z}\bar{w}$	$\bar{z}w$
xy	0	1	0	1
$x\bar{y}$	1	1	1	1
$\bar{x}\bar{y}$	1	0	1	0
$\bar{x}y$	0	0	0	0

MSP(f) $= xz\bar{w} + \bar{y}\bar{z}\bar{w} + x\bar{z}w + \bar{y}zw.$
$\bar{f} = yzw + \bar{x}z\bar{w} + y\bar{z}\bar{w} + \bar{x}\bar{z}w.$
MPS(f) $= (\bar{y} + \bar{z} + \bar{w})(x + \bar{z} + w)(\bar{y} + z + w)(x + z + \bar{w})$

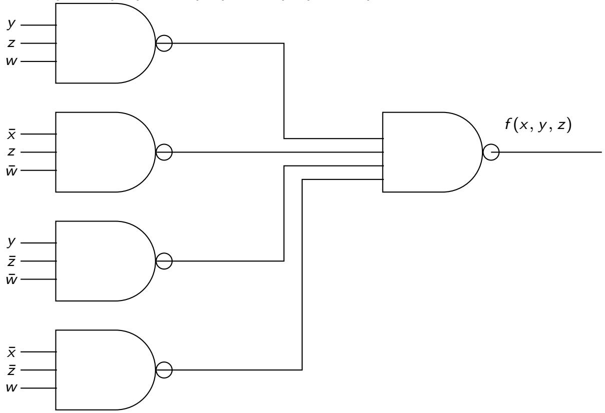
Construction of a minimal circuit using (AND-OR) gates, with $f(x, y, z, w)$ output.



A minimal circuit using (AND-OR) gates, with $f(x, y, z, w)$ output.

Construction of circuits with $f(x, y, z, w)$ output using **NAND** gates.

$\bar{f} = yzw + \bar{x}z\bar{w} + y\bar{z}\bar{w} + \bar{x}\bar{z}w$, then
$f = \overline{(yzw)}.\overline{(\bar{x}z\bar{w})}.\overline{(y\bar{z}\bar{w})}.\overline{(\bar{x}\bar{z}w)}$.

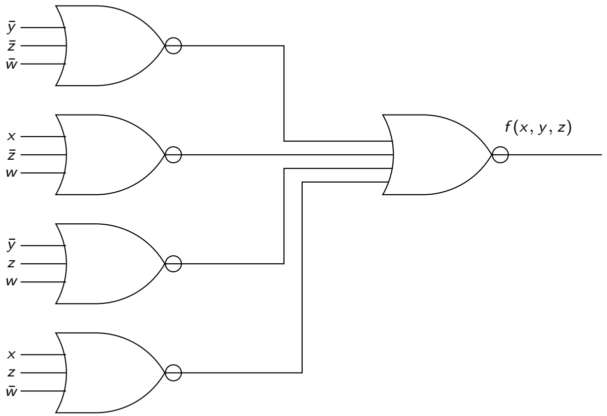


A circuit using (NAND) gates, with $f(x, y, z, w)$ output.

Construction of circuits with $f(x,y,z,w)$ output using **NOR** gates.

$f = (\bar{y}+\bar{z}+\bar{w})(x+\bar{z}+w)(\bar{y}+z+w)(x+z+\bar{w})$, then

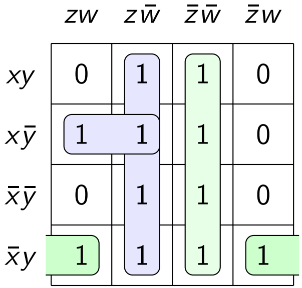
$\bar{f} = \overline{(\bar{y}+\bar{z}+\bar{w})} + \overline{(x+\bar{z}+w)} + \overline{(\bar{y}+z+w)} + \overline{(x+z+\bar{w})}$



A circuit using (NOR) gates, with $f(x,y,z,w)$ output.

Example

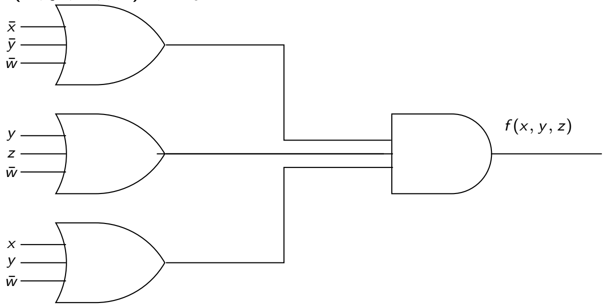
Let $f(x, y, z) = z\bar{w} + \bar{z}\bar{w} + x\bar{y}z + \bar{x}yzw + \bar{x}y\bar{z}w$. The Karnaugh -map for f is



	zw	$z\bar{w}$	$\bar{z}\bar{w}$	$\bar{z}w$
xy	0	1	1	0
$x\bar{y}$	1	1	1	0
$\bar{x}\bar{y}$	0	1	1	0
$\bar{x}y$	1	1	1	1

$\mathbf{MSP(f)} = z\bar{w} + \bar{z}\bar{w} + \bar{x}yw + x\bar{y}z,$
$\bar{f} = xyw + \bar{x}\bar{y}w + \bar{y}\bar{z}w.$
$\mathbf{MPS(f)} = (\bar{x} + \bar{y} + \bar{w})(x + y + \bar{w})(y + z + \bar{w}).$

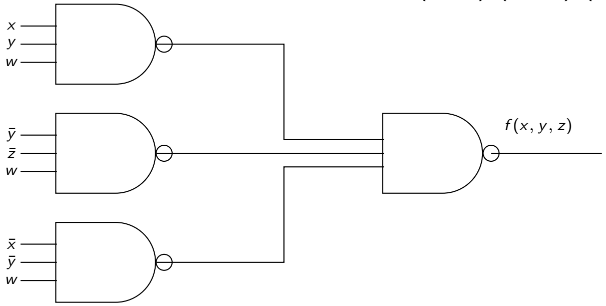
Construction of a minimal circuit using (AND-OR) gates, with $f(x, y, z, w)$ output.



A minimal circuit using (AND-OR) gates, with $f(x, y, z, w)$ output.

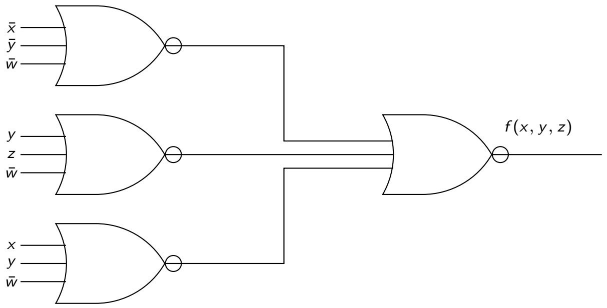
Construction of circuits with $f(x, y, z, w)$ output using **NAND** gates.

$\bar{f} = xyw + \bar{y}\bar{z}w + \bar{x}\bar{y}w$, then $f = \overline{(xyw)}.\overline{(\bar{y}\bar{z}w)}.\overline{(\bar{x}\bar{y}w)}$.



A circuit using (NAND) gates, with $f(x, y, z, w)$ output.

Construction of circuits with $f(x, y, z, w)$ output using **NOR** gates.
$f = (\bar{x} + \bar{y} + \bar{w})(x + y + \bar{w})(y + z + \bar{w})$, then
$\bar{f} = \overline{(\bar{x} + \bar{y} + \bar{w})} + \overline{(x + y + \bar{w})} + \overline{(y + z + \bar{w})}$.



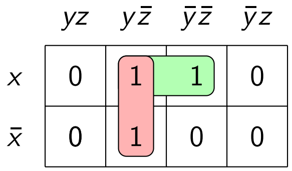
A circuit using (NOR) gates, with $f(x, y, z, w)$ output.

Example

Use K-maps to minimize these sum-of-products expansions.

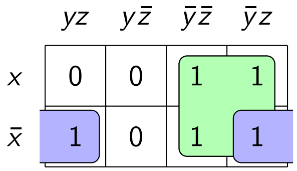
a) $xy\overline{z} + x\overline{y}\,\overline{z} + \overline{x}yz + \overline{x}\,\overline{y}\,\overline{z}$
b) $x\overline{y}z + x\overline{y}\,\overline{z} + \overline{x}\,\overline{y}z + \overline{x}yz + \overline{x}\,\overline{y}\,\overline{z}$
c) $xyz + xy\overline{z} + x\overline{y}z + x\overline{y}\,\overline{z} + \overline{x}yz + \overline{x}\,\overline{y}z + \overline{x}\,\overline{y}\,\overline{z}$
d) $xy\overline{z} + x\overline{y}\,\overline{z} + \overline{x}\,\overline{y}z + \overline{x}\,\overline{y}\,\overline{z}$

(a) $xy\bar{z} + x\bar{y}\;\bar{z} + \bar{x}yz + \bar{x}\;\bar{y}\;\bar{z} = y\bar{z} + x\bar{z}.$



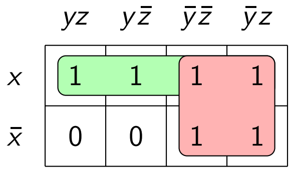
	yz	$y\bar{z}$	$\bar{y}\bar{z}$	$\bar{y}z$
x	0	1	1	0
$\bar{x}$	0	1	0	0

(b) $x\bar{y}z + x\bar{y}\;\bar{z} + \bar{x}\;\bar{y}z + \bar{x}yz + \bar{x}\;\bar{y}\;\bar{z} = \bar{y} + \bar{x}\bar{y}.$



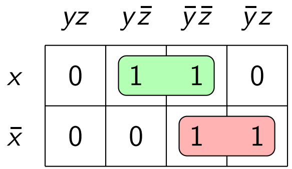
	yz	$y\bar{z}$	$\bar{y}\bar{z}$	$\bar{y}z$
x	0	0	1	1
$\bar{x}$	1	0	1	1

(c) $xyz + xy\bar{z} + x\bar{y}z + x\bar{y}\,\bar{z} + \bar{x}yz + \bar{x}\,\bar{y}z + \bar{x}\,\bar{y}\,\bar{z} = x + \bar{y}.$



	yz	$y\bar{z}$	$\bar{y}\bar{z}$	$\bar{y}z$
x	1	1	1	1
$\bar{x}$	0	0	1	1

(d) $xy\bar{z} + x\bar{y}\,\bar{z} + \bar{x}\,\bar{y}z + \bar{x}\,\bar{y}\,\bar{z} = x\bar{z} + \bar{x}\bar{y}.$



	yz	$y\bar{z}$	$\bar{y}\bar{z}$	$\bar{y}z$
x	0	1	1	0
$\bar{x}$	0	0	1	1