1 Basic Operations

We describe Boolean values as either false or true.

In a system that represents information numerically using only binary digits:

- \( 0 = \text{false} \)
- \( 1 = \text{true} \)

The following three basic Boolean operations represent the only operators we will use when reducing equations into their simplest form.

1.1 NOT

A function whose output is the opposite of its input.

\[
Q = \overline{A}
\] (1)

<table>
<thead>
<tr>
<th>A</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Depending on the context, any of the following forms might be seen used to express the NOT function applied to the variable \( a \):

- \( \neg a \)
- \( 
\overline{a} \)
- \( a' \)
- \( \neg a \) (C bitwise operator)
- \( !a \) (C logical operator)

\(^1\)In non-binary systems it might help to consider a more general definition where zero is considered false and anything that is not false is true.
1.2 AND

A function whose output is true if, and only if, all of its inputs are true.

\[ Q = A \land B \]  

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Depending on the context, any of the following forms might be seen used to express the AND function applied to the variables \( a \) and \( b \)

- \( a \land b \)
- \( a \cdot b \)
- \( ab \)
- \( a \& b \) (C bitwise operator)
- \( a \&\& b \) (C logical operator)

1.3 OR

A function whose output is true if any one, or both, of its inputs is/are true.

\[ Q = A \lor B \]  

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Depending on the context, any of the following forms might be seen used to express the OR function applied to the variables \( a \) and \( b \)

- \( a \lor b \)
• $a + b$
• $a \mid b$ (C bitwise operator)
• $a \| b$ (C logical operator)

2 Composing New Functions

We can combine the above three functions to create new ones.

2.1 NAND

A function whose output is false if, and only if, all of its inputs are true.

$$Q = \overline{A \land B}$$  \hspace{1cm} (4)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Depending on the context, any of the following forms might be seen used to express the NAND function applied to the variables $a$ and $b$

• $\overline{a \land b}$
• $\overline{a \lor b}$
• $\overline{a \cdot b}$
• $\neg(a \& b)$ (C bitwise operators)
• $!(a \&\& b)$ (C logical operators)

2.2 NOR

A function whose output is true if, and only if, all of its inputs are false.

$$Q = \overline{A \lor B}$$  \hspace{1cm} (5)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Depending on the context, any of the following forms might be seen used to express the **NOR** function applied to the variables $a$ and $b$

- $\overline{a \lor b}$
- $a + \overline{b}$
- $\sim (a \mid \lor b)$ (C bitwise operators)
- $\sim (a \mid \mid b)$ (C logical operators)

### 2.3 XOR (Exclusive OR)

A function whose output is **true** when an odd number of its inputs are **true**. We call this **odd parity**. In the special case (when there are only two inputs) the **XOR** function output is **true** when the two inputs are different and **false** when they are the same.

$$Q = A \oplus B$$

$$= \overline{A} \lor B \lor A \lor \overline{B}$$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Depending on the context, any of the following forms might be seen used to express the **XOR** function applied to the variables $a$ and $b$

- $a \oplus b$
- $a \sim b$ (C bitwise operator)

### 3 Material Implication

A function with two inputs $a$ and $b$ whose output is **true** if either $a$ is **false** or $a$ is **true** when $b$ is **true**.

$$Q = A \rightarrow B$$

$$= \overline{A} \lor B$$
Depending on the context, any of the following forms might be seen used to express the \texttt{IMP} function applied to the variables \(a\) and \(b\)

- \(a \rightarrow b\)
- \(a ? b : \text{true} \) (C conditional/ternary operator)

# All Possible Functions With Two Inputs

Consider the following:

- How many ways can we arrange 2 bits? (Answer: \(2^2 = 4\))
- How many ways can we arrange 4 bits? (Answer: \(2^4 = 16\))

Therefore:

1. There are exactly four possible ways to arrange two one-bit values.
2. There are exactly sixteen possible ways to arrange four one-bit values.

Conclusion: There are 16 possible Boolean functions that have two one-bit inputs \(a\) and \(b\):

<table>
<thead>
<tr>
<th>(a)</th>
<th>(b)</th>
<th>(a \land b)</th>
<th>(a \rightarrow b)</th>
<th>(a \lor b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

5. DeMorgan’s Morgan’s Laws

De Morgan observed the following relationships:

\[
\begin{align*}
\overline{a \land b} &= \overline{a} \lor \overline{b} \\
\overline{a \lor b} &= \overline{a} \land \overline{b}
\end{align*}
\]  \(10\) \(11\)

Proof by truth table:

<table>
<thead>
<tr>
<th>(a)</th>
<th>(b)</th>
<th>(\overline{a})</th>
<th>(\overline{b})</th>
<th>(a \land b)</th>
<th>(\overline{a \land b})</th>
<th>(\overline{a} \lor \overline{b})</th>
<th>(a \lor b)</th>
<th>(\overline{a \lor b})</th>
<th>(\overline{a} \land \overline{b})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
6 Completeness

Completeness refers to the fact that the basic operations provide a sufficient basis to describe all other possible Boolean operations.

https://en.wikipedia.org/wiki/Boolean_algebra#Completeness

A proof showing that NAND is complete by showing a truth table for a NOT, AND and OR function that are constructed only using the NAND function.

$$\overline{a} = a \land a$$  (12)
$$a \land b = a \land \overline{b} \land \overline{a} \land b$$  (13)
$$a \lor b = a \land \overline{a} \land b \land \overline{b}$$  (14)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>$\overline{a}$</th>
<th>$a \land b$</th>
<th>$a \land \overline{a} \land b \land \overline{a}$</th>
<th>$a \lor b$</th>
<th>$a \land \overline{a}$</th>
<th>$a \land \overline{a} \land b \land \overline{a}$</th>
<th>$a \land b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

7 Operator Precedence & Parenthesis

- The NOT operator has higher precedence than AND.
- The AND operator has higher precedence than OR.
- The OR operator has higher precedence than IMPLICATION.

Note that XOR is a function of one or more of these operators. Therefore its precedence is a matter of its implementation. Languages like C that include an explicit XOR operator put its precedence between that of the AND and OR operators.

Parenthesis can be used to manage the order of operations when the precedence rules would otherwise cause an undesirable result:

$$a \lor b \land c = a \lor (b \land c)$$  (15)
$$\neq (a \lor b) \land c$$  (16)

8 Laws of Boolean Algebra

Summarized from https://en.wikipedia.org/wiki/Boolean_algebra

---

2This along with DeMorgan’s laws and the above discussion on the composition of new functions demonstrates that the NAND function alone can be used to perform all 16 possible Boolean functions.
\( a \land (b \land c) = (a \land b) \land c \)
\( \text{Associativity of } \land \quad (17) \)
\( a \lor (b \lor c) = (a \lor b) \lor c \)
\( \text{Associativity of } \lor \quad (18) \)
\( a \land b = b \land a \)
\( \text{Commutativity of } \land \quad (19) \)
\( a \lor b = b \lor a \)
\( \text{Commutativity of } \lor \quad (20) \)
\( a \land (b \lor c) = (a \land b) \lor (a \land c) \)
\( \text{Distributivity of } \land \text{ over } \lor \quad (21) \)
\( a \lor (b \land c) = (a \lor b) \land (a \lor c) \)
\( \text{Distributivity of } \lor \text{ over } \land \quad (22) \)
\( a \land 1 = a \)
\( \text{Identity for } \land \quad (23) \)
\( a \lor 0 = a \)
\( \text{Identity for } \lor \quad (24) \)
\( a \land 0 = 0 \)
\( \text{Annihilator for } \land \quad (25) \)
\( a \lor 1 = 1 \)
\( \text{Annihilator for } \lor \quad (26) \)
\( a \land a = a \)
\( \text{Idempotence } \land \quad (27) \)
\( a \lor a = a \)
\( \text{Idempotence } \lor \quad (28) \)
\( a \land (b \lor c) = (a \land b) \lor (a \land c) \)
\( \text{Distributivity of } \land \text{ over } \lor \quad (29) \)
\( a \lor (b \land c) = (a \lor b) \land (a \lor c) \)
\( \text{Distributivity of } \lor \text{ over } \land \quad (30) \)
\( a \land b = a \lor b \)
\( \text{DeMorgan’s 1} \quad (34) \)
\( a \lor b = a \land b \)
\( \text{DeMorgan’s 2} \quad (35) \)

### 8.1 An Exercise

You should be able to show that each of the following equations are true by applying the above Boolean laws and by showing the truth tables for each case.

\[
\begin{align*}
a \land b &= \overline{a} \lor \overline{b} \\
\overline{a} \land \overline{b} &= a \lor \overline{b} \\
\overline{a} \lor \overline{b} &= \overline{a} \land \overline{b} \\
(a \lor b) \land (c \land d) &= a \land c \land a \land d \lor b \land c \land b \land d
\end{align*}
\]

For example, a proof of Equation 39:

\[
\begin{align*}
f &= \overline{a} \lor a \land b \\
&= \overline{a} \lor (a \land b) \\
&= (\overline{a} \lor a) \land (\overline{a} \lor b) \\
&= (a \lor \overline{a}) \land (\overline{a} \lor b) \\
&= 1 \land (\overline{a} \lor b) \\
&= (\overline{a} \lor b) \land 1 \\
&= \overline{a} \lor b \\
\end{align*}
\]
Given
Show operator precedence
Distributivity of \lor over \land
Commutativity of \lor
Complement of \lor
Identity for \land
9 Minterm (lower-case m) (SOP form)

See also: https://en.wikipedia.org/wiki/Canonical_normal_form

Any Boolean function can be expressed in SOP form.\[3\]

In a minterm, each variable may appear only once as $a$ or $\overline{a}$. For a three-input function, there are $2^3 = 8$ minterms:

\[
\begin{align*}
m0 &= \overline{a} \land \overline{b} \land \overline{c} \\
m1 &= \overline{a} \land \overline{b} \land c \\
m2 &= \overline{a} \land b \land \overline{c} \\
m3 &= \overline{a} \land b \land c \\
m4 &= a \land \overline{b} \land \overline{c} \\
m5 &= a \land \overline{b} \land c \\
m6 &= a \land b \land \overline{c} \\
m7 &= a \land b \land c
\end{align*}
\]

Note that the appearance of the NOT-bars match the pattern of zeros when counting from zero to seven in binary.

Use of minterms in an equation allows a shorter notation as shown below:

\[
f = m1 \lor m2 \lor m4 \lor m7
\]

\[
= (\overline{a} \land \overline{b} \land c) \lor (\overline{a} \land b \land \overline{c}) \lor (a \land \overline{b} \land \overline{c}) \lor (a \land b \land c)
\]

*Parenthesis added for readability.*

The long form shown in equation \[57\] suggests a rationale for why it is called SOP (Sum Of Products.)

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$(\overline{a} \land \overline{b} \land c)$</th>
<th>$(\overline{a} \land b \land \overline{c})$</th>
<th>$(a \land \overline{b} \land \overline{c})$</th>
<th>$(a \land b \land c)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

As can be seen in the truth table, the “mechanics” driving the SOP equation is based on identifying those specific function input patterns where the output is \textit{true}. By using an OR gate to drive the output, it will set its output \textit{true} any time one of the recognized input terms is \textit{true}.

10 Maxterm (upper-case M) (POS form)

Same idea as the minterm, but inverted and use the $\lor$ instead of the $\land$ function:

Any Boolean function can be expressed in POS form.

\[ M_0 = a \lor b \lor c \] (58)
\[ M_1 = a \lor b \lor \overline{c} \] (59)
\[ M_2 = a \lor \overline{b} \lor c \] (60)
\[ M_3 = a \lor \overline{b} \lor \overline{c} \] (61)
\[ M_4 = \overline{a} \lor b \lor c \] (62)
\[ M_5 = \overline{a} \lor b \lor \overline{c} \] (63)
\[ M_6 = \overline{a} \lor \overline{b} \lor c \] (64)
\[ M_7 = \overline{a} \lor \overline{b} \lor \overline{c} \] (65)

Note that the appearance of the NOT-bars match the pattern of ones when counting from zero to seven in binary.

\[ f = M_0 \land M_1 \land M_2 \land M_4 \] (66)
\[ = (a \lor b \lor c) \land (a \lor b \lor \overline{c}) \land (a \lor \overline{b} \lor c) \land (\overline{a} \lor b \lor c) \] (67)

The long form shown in equation 67 is often called a POS (Product Of Sums.)

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
& a & b & c & (a \land b \land c) & (a \land b \land \overline{c}) & (a \land \overline{b} \land c) & f \\
\hline
M_0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
M_1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
M_2 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
M_3 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
M_4 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
M_5 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
M_6 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
M_7 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
\end{array}
\]

As can be seen in the truth table, the “mechanics” driving the POS equation is based on identifying those specific function input patterns where the output is \textit{false}. By using an \textit{AND} to drive the output, it will set its output \textit{false} any time one of the recognized input terms is \textit{false}.

\subsection{Relationship Between Minterms and Maxterms}

The complement of a maxterm, such as $M_5$, is the respective minterm $m_5$.

This can be verified with DeMorgan’s as in:

\[ M_5 = \overline{a} \lor \overline{b} \lor \overline{c} \] Given (68)
\[ m_5 = a \land \overline{b} \land \overline{c} \] Given (69)
\[ \overline{M_5} = \overline{a} \lor b \lor c \] Complement both sides (70)
\[ \overline{M_5} = \overline{a} \land \overline{b} \land \overline{c} \] DeMorgan 1 (71)
\[ \overline{M_5} = a \land \overline{b} \land c \] Double negation (72)
\[ \overline{M_5} = m_5 \] (73)
Note that DeMorgan’s also works with more than two inputs.

It turns out that converting a minterm to a maxterm works the same way. For example:

\[
\begin{align*}
M5 &= \overline{a} \lor b \lor \overline{c} & \text{Given} \\
m5 &= a \land \overline{b} \land c & \text{Given} \\
\overline{m5} &= a \land \overline{b} \land c & \text{Complement both sides} \\
\overline{m5} &= \overline{a} \lor b \lor \overline{c} & \text{DeMorgan 2} \\
\overline{m5} &= \overline{a} \lor b \lor \overline{c} & \text{Double negation} \\
\overline{m5} &= M5 & \text{}(79)
\end{align*}
\]

### 10.2 When to Apply the POS or SOP Form

If SOP and POS can be easily converted back and forth, it begs the question “Why concern ourselves with both forms?”

The answer lies in which of the two forms is easier to apply to a given situation. For example, the following truth table is best expressed in SOP form simply because there are fewer cases when the output is true.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Therefore there are fewer terms expressed in the SOP form describing it:

\[
f = m3 \lor m5 \lor m7
\]

\[
f = (\overline{a} \land b \land c) \lor (a \land \overline{b} \land c) \lor (a \land b \land c)
\]

...than there are in the POS form:

\[
f = M0 \land M1 \land M2 \land M4 \land M6
\]

\[
f = (a \lor b \lor c) \land (a \lor b \lor \overline{c}) \land (a \lor \overline{b} \lor c) \land (\overline{a} \lor b \lor c) \land (\overline{a} \lor \overline{b} \lor c)
\]

The shorter function may be more desirable to work with.

### 11 Simplification of SOP and POS Forms

While it is easy to express any Boolean function in SOP or POS form, it is desirable to be able to then reduce them to their simplest form.
To accomplish this, we apply the Boolean laws until the function becomes irreducible and has the fewest number of operators.

As an example we demonstrate a reduction of the following truth table using both the SOP and POS functions:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### SOP:

\[
f = m_2 \lor m_4 \lor m_5 \lor m_6 \lor m_7
\]

\[
= (\overline{a} \land b \land \overline{c}) \lor (a \land \overline{b} \land \overline{c}) \lor (a \land b \land c) \lor (a \land b \land \overline{c}) \lor (a \land b \land c)
\]

\[
= (\overline{a} \land b \land \overline{c}) \lor (a \land \overline{b} \land (\overline{c} \lor c)) \lor (a \land b \land (\overline{c} \lor c))
\]

\[
= (\overline{a} \land b \land \overline{c}) \lor (a \land \overline{b} \land 1) \lor (a \land b \land 1)
\]

\[
= (\overline{a} \land b \land \overline{c}) \lor (a \land \overline{b}) \lor (a \land b)
\]

\[
= (\overline{a} \land b \land \overline{c}) \lor (a \land (\overline{b} \lor b))
\]

\[
= (\overline{a} \land b \land \overline{c}) \lor (a \land 1)
\]

\[
= (\overline{a} \land b \land \overline{c}) \lor a
\]

\[
= a \lor (\overline{a} \land b \land \overline{c})
\]

\[
= a \lor (\overline{a} \land (b \lor \overline{c}))
\]

\[
= (a \lor \overline{a}) \land (a \lor (b \land \overline{c}))
\]

\[
= 1 \land (a \lor (b \land \overline{c}))
\]

\[
= a \lor (b \land \overline{c})
\]

### POS:

\[
f = M_0 \land M_1 \land M_3
\]

\[
= (a \lor b \lor c) \land (a \lor b \lor \overline{c}) \land (a \lor \overline{b} \lor \overline{c})
\]

\[
= ((a \lor b) \lor (c \land \overline{c})) \land (a \lor \overline{b} \lor \overline{c})
\]

\[
= ((a \lor b) \lor 0) \land (a \lor \overline{b} \lor \overline{c})
\]

\[
= (a \lor b) \land (a \lor \overline{b} \lor \overline{c})
\]

\[
= a \lor (b \land (\overline{b} \lor \overline{c}))
\]

\[
= a \lor ((b \land \overline{b}) \lor (b \land \overline{c}))
\]

\[
= a \lor (0 \lor (b \land \overline{c}))
\]

\[
= a \lor (b \land \overline{c})
\]

Ultimately, we see that both methods arrived at the same conclusion!