CHAPTER III

BOOLEAN ALGEBRA
INTRODUCTION

BOOLEAN ALGEBRA

BOOLEAN VALUES

- Boolean algebra is a form of algebra that deals with single digit binary values and variables.
- Values and variables can indicate some of the following binary pairs of values:
  - ON / OFF
  - TRUE / FALSE
  - HIGH / LOW
  - CLOSED / OPEN
  - 1 / 0
Three fundamental operators in Boolean algebra:

- **NOT**: unary operator that complements represented as \( \overline{A} \), \( A' \), or \( \sim A \)
- **AND**: binary operator which performs logical multiplication
  - i.e. \( A \) ANDed with \( B \) would be represented as \( AB \) or \( A \cdot B \)
- **OR**: binary operator which performs logical addition
  - i.e. \( A \) ORed with \( B \) would be represented as \( A + B \)

<table>
<thead>
<tr>
<th>NOT</th>
<th>AND</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>( \overline{A} )</td>
<td>( A )</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>( 0 )</td>
<td>( 0 )</td>
<td>( 0 )</td>
</tr>
<tr>
<td>( 1 )</td>
<td>( 1 )</td>
<td>( 1 )</td>
</tr>
</tbody>
</table>
Below is a table showing all possible Boolean functions $F_N$ given the two-inputs $A$ and $B$.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>$F_0$</th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_3$</th>
<th>$F_4$</th>
<th>$F_5$</th>
<th>$F_6$</th>
<th>$F_7$</th>
<th>$F_8$</th>
<th>$F_9$</th>
<th>$F_{10}$</th>
<th>$F_{11}$</th>
<th>$F_{12}$</th>
<th>$F_{13}$</th>
<th>$F_{14}$</th>
<th>$F_{15}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<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>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</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>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Null**: $AB$
- **Inhibition**: $A$
- **Implication**: $A + B$
- **Identity**: $AB$
Boolean expressions must be evaluated with the following order of operator precedence:

- parentheses
- NOT
- AND
- OR

Example:

\[
F = (A(C + BD) + BC)\overline{E}
\]
Example 1:
Evaluate the following expression when \( A = 1, B = 0, C = 1 \)
\[
F = C + \overline{C}B + B\overline{A}
\]
- Solution
\[
F = 1 + 1 \cdot 0 + 0 \cdot 1 = 1 + 0 + 0 = 1
\]
Example 2:
Evaluate the following expression when \( A = 0, B = 0, C = 1, D = 1 \)
\[
F = D(\overline{B}\overline{C}A + (\overline{A}B + C) + C)
\]
- Solution
\[
F = 1 \cdot (0 \cdot 1 \cdot 0 + (0 \cdot \overline{0} + 1) + 1) = 1 \cdot (0 + 1 + 1) = 1 \cdot 1 = 1
\]
<table>
<thead>
<tr>
<th>Identity</th>
<th>Identity</th>
</tr>
</thead>
<tbody>
<tr>
<td>X + 0 = X</td>
<td>X · 1 = X</td>
</tr>
<tr>
<td>X + 1 = 1</td>
<td>X · 0 = 0</td>
</tr>
<tr>
<td>X + X = X</td>
<td>X · X = X</td>
</tr>
<tr>
<td>X + X' = 1</td>
<td>X · X' = 0</td>
</tr>
<tr>
<td>(X')' = X</td>
<td>(X' + Y) = X + Y</td>
</tr>
<tr>
<td>X + Y = Y + X</td>
<td>X(Y + Z) = X(YZ)</td>
</tr>
<tr>
<td>X + (Y + Z) = (X + Y) + Z</td>
<td>X(YZ) = (XY)Z</td>
</tr>
<tr>
<td>X(Y + Z) = XY + XZ</td>
<td>X + YZ = (X + Y)(X + Z)</td>
</tr>
<tr>
<td>X + XY = X</td>
<td>X(X + Y) = X</td>
</tr>
<tr>
<td>X + X'Y = X + Y</td>
<td>X(X' + Y) = XY</td>
</tr>
<tr>
<td>(X + Y)' = X'Y'</td>
<td>(XY)' = X' + Y'</td>
</tr>
<tr>
<td>XY + X'Z + YZ = XY + X'Z</td>
<td>(X + Y)(X' + Z)(Y + Z) = (X + Y)(X' + Z)</td>
</tr>
</tbody>
</table>

**Boolean Algebra Basic Identities**

- Identity
- Idempotent Law
- Complement
- Involution Law
- Commutativity
- Associativity
- Distributivity
- Absorption Law
- Simplification
- DeMorgan’s Law
- Consensus Theorem
Duality principle:

States that a Boolean equation remains valid if we take the dual of the expressions on both sides of the equals sign.

The dual can be found by interchanging the **AND** and **OR** operators along with also interchanging the 0’s and 1’s.

This is evident with the duals in the basic identities.

- For instance: DeMorgan’s Law can be expressed in two forms
  
  \[(X + Y)' = X'Y'\] \quad \text{as well as} \quad \[(XY)' = X' + Y'\]
Example: Simplify the following expression

\[ F = BC + B\bar{C} + BA \]

Simplification

\[ F = B(C + \bar{C}) + BA \]

\[ F = B \cdot 1 + BA \]

\[ F = B(1 + A) \]

\[ F = B \]
Example: Simplify the following expression

\[ F = A + \bar{A}B + \bar{A}BC + \bar{A}BCD + \bar{A}BCDE \]

Simplification

\[ F = A + \bar{A}(B + BC + BCD + BCDE) \]
\[ F = A + B + BC + BCD + BCDE \]
\[ F = A + B + \bar{B}(C + \bar{C}D + \bar{C}DE) \]
\[ F = A + B + C + \bar{C}D + \bar{C}DE \]
\[ F = A + B + C + \bar{C}(D + \bar{D}E) \]
\[ F = A + B + C + D + \bar{D}E \]
\[ F = A + B + C + D + E \]
• Example: Show that the following equality holds

\[ A(\overline{BC} + BC) = \overline{A} + (B + C)(\overline{B} + \overline{C}) \]

• Simplification

\[ A(\overline{BC} + BC) = \overline{A} + (\overline{BC} + BC) \]

\[ = \overline{A} + (\overline{BC})(BC) \]

\[ = \overline{A} + (B + C)(\overline{B} + \overline{C}) \]
Boolean expressions can be manipulated into many forms.

Some standardized forms are required for Boolean expressions to simplify communication of the expressions.

- **Sum-of-products (SOP)**
  
  - Example:

    \[
    F(A, B, C, D) = AB + \overline{B} \overline{C} \overline{D} + AD
    \]

- **Products-of-sums (POS)**

  - Example:

    \[
    F(A, B, C, D) = (A + B)(\overline{B} + C + \overline{D})(A + D)
    \]
The following table gives the minterms for a *three-input* system:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>$m_0$</th>
<th>$m_1$</th>
<th>$m_2$</th>
<th>$m_3$</th>
<th>$m_4$</th>
<th>$m_5$</th>
<th>$m_6$</th>
<th>$m_7$</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>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</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>0</td>
<td>1</td>
<td>0</td>
<td>0</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>0</td>
<td>1</td>
<td>1</td>
<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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</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>
<td>0</td>
<td>1</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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
**Sum-of-minterms** standard form expresses the Boolean or switching expression in the form of a **sum of products** using **minterms**.

For instance, the following Boolean expression using minterms

\[
F(A, B, C) = \overline{A}BC + \overline{A}BC + ABC + ABC
\]

could instead be expressed as

\[
F(A, B, C) = m_0 + m_1 + m_4 + m_5
\]

or more compactly

\[
F(A, B, C) = \sum m(0, 1, 4, 5) = \text{one-set}(0, 1, 4, 5)
\]
The following table gives the maxterms for a three-input system:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>( M_0 )</th>
<th>( M_1 )</th>
<th>( M_2 )</th>
<th>( M_3 )</th>
<th>( M_4 )</th>
<th>( M_5 )</th>
<th>( M_6 )</th>
<th>( M_7 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<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>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<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>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</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>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
STANDARD FORMS

PRODUCT OF MAXTERMS

- Product-of-maxterms standard form expresses the Boolean or switching expression in the form of **product of sums** using **maxterms**.

- For instance, the following Boolean expression using maxterms

\[
F(A, B, C) = (A + B + \overline{C})(\overline{A} + B + C)(\overline{A} + \overline{B} + \overline{C})
\]

could instead be expressed as

\[
F(A, B, C) = M_1 \cdot M_4 \cdot M_7
\]

or more compactly as

\[
F(A, B, C) = \prod M(1, 4, 7) = \text{zero-set}(1, 4, 7)
\]
Given an arbitrary Boolean function, such as

$$F(A, B, C) = AB + \overline{B}(\overline{A} + \overline{C})$$

how do we form the canonical form for:

- **sum-of-minterms**
  - Expand the Boolean function into a sum of products. Then take each term with a missing variable $X$ and **AND** it with $X + \overline{X}$.

- **product-of-maxterms**
  - Expand the Boolean function into a product of sums. Then take each factor with a missing variable $X$ and **OR** it with $X\overline{X}$. 
STANDARD FORMS
FORMING SUM OF MINTERMS

- Example

\[
F(A, B, C) = AB + B(\overline{A} + \overline{C}) = AB + \overline{AB} + \overline{BC}
\]

\[
= AB(\overline{C} + \overline{C}) + \overline{AB}(\overline{C} + \overline{C}) + (A + \overline{A})\overline{BC}
\]

\[
= \overline{ABC} + \overline{ABC} + \overline{ABC} + \overline{ABC} + ABC
\]

\[
= \sum m(0, 1, 4, 6, 7)
\]

\[
\begin{array}{ccc|c}
A & B & C & F \\
0 & 0 & 0 & 1 \quad 0 \\
0 & 0 & 1 & 1 \quad 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 \quad 4 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 \quad 6 \\
1 & 1 & 1 & 1 \quad 7 \\
\end{array}
\]

Minterms listed as 1s in Truth Table
STANDARD FORMS
FORMING PROD OF MAXTERMS

- Example

\[ F(A, B, C) = AB + \overline{B}(\overline{A} + \overline{C}) = AB + \overline{A}B + \overline{B}C \]

\[ = (A + \overline{B})(A + B + \overline{C})(\overline{A} + B + \overline{C}) \]

\[ = (A + \overline{B} + CC)(A + B + \overline{C})(\overline{A} + B + \overline{C}) \]

\[ = (A + \overline{B} + C)(A + \overline{B} + \overline{C})(\overline{A} + B + \overline{C}) \]

\[ = \prod M(2, 3, 5) \]

<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>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</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>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Maxterms are sums, so this is called a "product of sums" (POS)

Maxterms listed as 0s in Truth Table
Full Solution of Maxterm Example

F(A,B,C) = AB + B'(A'+C')

Distributivity (OR) X(Y+Z) = XY+XZ

AB + A'B' + B'C'

Distributivity (AND) Identity, X + YZ = (X+Y)(X+Z)

(AB + A') * (AB + B') + B'C'

Simplification Identity, X + X'Y = X + Y

[ (B + A') * (A + B') ] + B'C'

Distributivity (AND) Identity, X = [ (B + A') * (A + B') ], Y = B', Z = C'

( [ (B + A') * (A + B') ] + B' ) * ( [ (B + A') * (A + B') ] + C' )

Distributivity (AND) Identity on each ANDed term:
X = B' and C' (respectively) Y = (B + A'), Z = (A + B'))

( (A' + B + B’) * (A + B’ + B’) ) * ( (A’ + B + C’) * (A + B’ + C’) )

Idempotent and Identity – drop (A’ + B + B’), is always 1, X+X'=1, X+1=1, X*1=X also X+X=X

(A + B') * (A' + B + C') * (A + B' + C')

Expand (A + B') = (A + B' +CC') since CC'=0, X+0=X

(A + B') = (A + B' + C)*(A + B' + C')

F(A,B,C) = (A + B’+ C) (A + B’+ C’) (A’ + B + C’) where (A + B’ + C’) was duplicated, X+X=X.

Maxterms (1 where A,B, or C is complemented in a term): 010, 011, 101 = 2, 3, 5
General Rule for Expanding SOP or POS to Maxterms or Minterms

If a term is missing one variable, it expands to two minterms or maxterms.

If a term is missing two variables, it expands to four minterms or maxterms.

If a term is missing N variables, it expands to $2^N$ minterms or maxterms.

Examples for $F(A,B,C,D)$ - replicate incomplete term with every possible combination of complemented and non-complemented missing variables:

**Minterms**

$ABC = ABCD + ABCD'$

$AB = ABCD + ABCD' + ABC'D + ABC'D'$

$A = ABCD + ABCD' + ABC'D + ABC'D' + AB'CD + A B'CD' + A B'C'D + A B'C'D'$

**Maxterms**

$A+B+C = (A+B+C+D) (A+B+C+D')$

$A+B = (A+B+C+D) (A+B+C+D') (A+B+C'+D) (A+B+C'+D')$

Converting between sum-of-minterms and product-of-maxterms

- The two are complementary, as seen by the truth tables.
- To convert interchange the $\Sigma$ and $\Pi$, then use missing terms.
- Example: The example from the previous slides

$$ F(A, B, C) = \sum m(0, 1, 4, 6, 7) $$

\[
\begin{array}{c}
F'(A,B,C) = \text{SUM } m(2,3,5)
\end{array}
\]

is re-expressed as

$$ F(A, B, C) = \prod M(2, 3, 5) $$

\[
\begin{array}{c}
F'(A,B,C) = \text{PI } M(0,1,4,6,7)
\end{array}
\]

where the numbers 2, 3, and 5 were missing from the minterm representation.
Often it is desired to simplify a Boolean function. A quick graphical approach is to use Karnaugh maps.

2-variable Karnaugh map

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

F = AB

3-variable Karnaugh map

<table>
<thead>
<tr>
<th></th>
<th>A=00</th>
<th>A=01</th>
<th>A=11</th>
<th>A=10</th>
</tr>
</thead>
<tbody>
<tr>
<td>B=01</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B=10</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

F = AB + C

4-variable Karnaugh map

<table>
<thead>
<tr>
<th></th>
<th>A=000</th>
<th>A=001</th>
<th>A=011</th>
<th>A=010</th>
<th>A=111</th>
<th>A=110</th>
<th>A=100</th>
<th>A=101</th>
</tr>
</thead>
<tbody>
<tr>
<td>B=00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B=01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B=10</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B=11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

F = AB + CD
Notice that the ordering of cells in the map are such that moving from one cell to an adjacent cell only changes one variable.

This ordering allows for grouping of minterms/maxterms for simplification.
**Implicant**
- Bubble covering only 1's (size of bubble must be a power of 2).

**Prime implicant**
- Bubble that is expanded as big as possible (but increases in size by powers of 2).

**Essential prime implicant**
- Bubble that contains a 1 covered only by itself and no other prime implicant bubble.

**Non-essential prime implicant**
- All contained 1's can be covered by another prime implicant bubble.
• Procedure for finding the SOP from a Karnaugh map
  • Step 1: Form the 2-, 3-, or 4-variable Karnaugh map as appropriate for the Boolean function.
  • Step 2: Identify all essential prime implicants for 1's in the Karnaugh map.
  • Step 3: Identify non-essential prime implicants to cover remaining 1's in the Karnaugh map (not covered in step 2).
  • Step 4: For each essential implicant and selected non-essential prime implicants (that cover all 1's), determine the corresponding product term.
  • Step 5: Form a sum-of-products with all product terms from previous step.
Simplify the following Boolean function

\[ F(A, B, C) = \sum m(0, 1, 4, 5) = \overline{A}BC + \overline{A}BC + ABC + \overline{A}BC \]

Solution:

The essential prime implicants are \( \overline{B} \).

There are no non-essential prime implicants.

The sum-of-products solution is \( F = \overline{B} \).
**SIMPLIFICATION**

**EXAMPLE FOR SOP (2)**

**SOP: SUM OF PRODUCTS**

- **Simplify the following Boolean function**

\[
F(A, B, C) = \sum m(0, 1, 4, 6, 7) = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}BC + AB\overline{C} + ABC
\]

- **Solution:**

\[
\begin{array}{c|cccc}
A & 0 & 01 & 11 & 10 \\
\hline
0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 \\
\end{array}
\]

- The essential prime implicants are \(\overline{A}\overline{B}\) and \(AB\).
- The non-essential prime implicants are \(\overline{B}\overline{C}\) or \(AC\).
- The sum-of-products solution is

\[
F = AB + \overline{A}\overline{B} + \overline{B}C \quad \text{or} \quad F = AB + \overline{A}\overline{B} + AC.
\]
• **Procedure for finding the SOP from a Karnaugh map**

  • Step 1: Form the 2-, 3-, or 4-variable Karnaugh map as appropriate for the Boolean function.
  • Step 2: Identify all essential prime implicants for 0's in the Karnaugh map.
  • Step 3: Identify non-essential prime implicants to cover remaining 0's in the Karnaugh map.
  • Step 4: For each essential and the selected non-essential prime implicants from each set, determine the corresponding sum term.
  • Step 5: Form a product-of-sums with all sum terms from previous step.
Simplify the following Boolean function

\[ F(A, B, C) = \prod M(2, 3, 5) = (A + \bar{B} + C)(A + \bar{B} + \bar{C})(\bar{A} + B + \bar{C}) \]

Solution:

The essential prime implicants are \( \bar{A} + B + \bar{C} \) and \( A + \bar{B} \).

There are no non-essential prime implicants.

The product-of-sums solution is \( F = (A + \bar{B})(\bar{A} + B + \bar{C}) \).
• Simplify the following Boolean function

\[ F(A, B, C) = \prod M(0, 1, 5, 7, 8, 9, 15) \]

• Solution:

• The essential prime implicants are \( B + C \) and \( \bar{B} + \bar{C} + \bar{D} \).

• The non-essential prime implicants can be \( A + \bar{B} + \bar{D} \) or \( A + C + \bar{D} \).

• The product-of-sums solution can be either

\[ F = (B + C)(\bar{B} + \bar{C} + \bar{D})(A + \bar{B} + \bar{D}) \]

or

\[ F = (B + C)(\bar{B} + \bar{C} + \bar{D})(A + C + \bar{D}) \]
Switching expressions are sometimes given as incomplete, or with don’t-care conditions.

- Having don’t-care conditions can simplify Boolean expressions and hence simplify the circuit implementation.
- Along with the zero-set( ) and one-set( ), we will also have dc( ).
- Don’t-cares conditions in Karnaugh maps
  - Don’t-cares will be expressed as an “X” or “-” in Karnaugh maps.
  - Don’t-cares can be bubbled along with the 1s or 0s depending on what is more convenient and help simplify the resulting expressions.
Find the SOP simplification for the following Karnaugh map:

- The essential prime implicants are $BD$ and $\overline{B}C$.
- There are no non-essential prime implicants.
- The sum-of-products solution is $F = \overline{B}C + BD$. 

\[
\begin{array}{cccc}
\text{AB} & 00 & 01 & 11 & 10 \\
00 & 0 & 0 & 1 & 1 \\
01 & 1 & 0 & 0 & 1 \\
11 & 1 & X & 0 & X \\
10 & 0 & 0 & 1 & X \\
\end{array}
\]

zero-set(0, 1, 5, 7, 8, 9, 15) 
one-set(2, 3, 4, 6, 11, 12) 
dc(10, 13, 14)
• Find the POS simplification for the following Karnaugh map

<table>
<thead>
<tr>
<th>AB</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>X</td>
</tr>
</tbody>
</table>

zero-set(0, 1, 5, 7, 8, 9, 15)
one-set(2, 3, 4, 6, 11, 12)
dc(10, 13, 14)

• Solution:
  • The essential prime implicants are $B + C$ and $\overline{B} + \overline{D}$.
  • There are no non-essential prime implicants.
  • The product-of-sums solution is $F = (B + C)(\overline{B} + \overline{D})$. 