Sum of Products (SOP) Expressions

The Sum of Products (SOP) form of Boolean expressions and equations contains a list of terms (called minterms) in which all variables are ANDed (products). These minterms are then ORed (summed) together. Expressions in this form are particularly well-suited for minimization using the most common methods and for FPGA (Field Programmable Gate Arrays) programming languages, such as VHDL (used in this classes).

Conventional Form

Consider the following logical expression:

\[ W = A \overline{B} C + \overline{A} B \overline{C} + \overline{A} B C + \overline{A} \overline{B} C \]

We will refer to this as the conventional form of a Sum of Products (SOP) expression. Note that the structure of the expression shows four groups of products (minterms). These minterms are ORed together. The AND logic has first operation precedence, followed by the OR operator.

The list of the variables in the entire expression is referred to as its domain, in this case A, B, and C. This is stated as, "Y is a function of A, B, and C", with the definition of the function following the equal sign. Since there are three variables in the domain, the expression has a domain of three.

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

Summation Forms

A summation form of a logical expression can be obtained by referring to each minterm in the expression by its numerical equivalent in the domain, rather than by the list of variables. In this case we substitute a "1" for all positive variables and a "0" for all inverted (i.e.; those with a bar over them). All missing variables (i.e.; AC is missing B), you have to expand to include all variables. So, AC would be ABC + A!BC (The exclamation point is sometimes used as a "NOT" notation, as is the "/")

Example:

\[ f(A, B, C) = A \overline{B} C + \overline{A} B \overline{C} + \overline{A} B C + \overline{A} \overline{B} C \]
\[ f(A, B, C) = 1 0 1 + 0 0 0 + 0 1 0 + 0 0 1 \]
\[ f(A, B, C) = 5 + 0 + 2 + 1 \]
\[ f(A, B, C) = \Sigma(5, 0, 2, 1) \]

8/23/2008
\[ f(A, B, C) = \Sigma(0, 1, 2, 5) \]

The conversion process took three steps.

1. Each minterm is converted to its binary equivalent. This is done by treating each inverted variable as a 0 digit, and each non-inverted variable as a 1 digit.
2. Each minterm is converted from binary to decimal
3. Finally the minterms are placed in ascending order.

**Typographic Forms**

Typographic form (and ascending typographic form) was discussed in the previous topic. Here, the Boolean equation shown above, is converted to ascending typographic form so that these three formats can be observed and compared.

\[
\begin{align*}
  f(A, B, C) & = AB'C + AB'C + AB'C + AB'C & \text{Original} \\
  f(A, B, C) & = 101 + 000 + 010 + 001 & \text{Temporary Binary Substitution} \\
  f(A, B, C) & = 000 + 001 + 010 + 101 & \text{Numerical Ascending Order} \\
  f(A, B, C) & = \overline{A} \overline{B} \overline{C} + \overline{A} \overline{B} C + \overline{A} B \overline{C} + A \overline{B} C & \text{Symbolic Ascending Order} \\
  f(A, B, C) & = \text{not A and not B and not C or } \text{not A and not B and C or } \text{not A and B and not C or } A \text{ and not B and C} & \text{VHDL Format}
\end{align*}
\]

**Advantages of the Different Forms**

We have discussed three forms of logical expressions: conventional, typographic, and summation. Each form has advantages in specific situations, all of which will be used in state machine design.

The conventional form is most useful for implementation with discrete logic gates.

The VHDL form is most useful for implementation with programmable logic when the summation form is not used.

The summation form is also useful for implementation in programmable logic. One advantage of the summation format of a logical expression is...
that very long and complex expressions can be easily described. Another advantage is the ease of conversion between this and other forms. Finally, the most important advantage of this form concerns its implementation with hardware devices.

Examples Converting SOP Expression to VHDL Format

Example #1: Convert the following SOP Expression to typographic form:

\[ W = \overline{A} \overline{B} C + A \overline{B} \overline{C} + A B C \]

\[ W = f(A, B, C) = \text{not } A \text{ and } \text{not } B \text{ and } C \text{ or } A \text{ and } \text{not } B \text{ and } \text{not } C \text{ or } A \text{ and } B \text{ and } C \]

Example #2: Convert the following SOP Expression to VHDL form:

\[ X = \overline{A} C D + A B \overline{C} + \overline{C} D \]

\[ X = f(A, B, C, D) = \text{not } A \text{ and } C \text{ and } D \text{ or } A \text{ and } B \text{ and } \text{not } C \text{ or } \text{not } C \text{ and } \text{not } D \]

Example #3: Convert the following SOP Expression to typographic form:

\[ Y = \overline{A} + B \overline{C} + \overline{A} C \]

\[ Y = f(A, B, C) = \text{not } A \text{ or } B \text{ and } \text{not } C \text{ or } \text{not } A \text{ and } C \]

Conversion from conventional to summation form:

The conversion from conventional to the summation format is not quite as straightforward as the conversion to the typographic format. The summation form lists all true minterms of the expression without regard to any minimization of the expression. Therefore, if the expression has been minimized in any way it must be expanded to include all of the individual minterms. In examples 2 and 3 above, variables are missing from the minterms, so these expressions must be first expanded before they can be converted to summation format.

In example 1 above all the variables are present. So, another method (Karnaugh maps) will be used to see if all terms are represented.

Example #1: Convert the equation SOP expression to its summation format (this is the same expression as example #2 above)

Solution:
\[ X = f(A,B,C,D) = \overline{A} \overline{C} \overline{D} + A \overline{B} \overline{C} + \overline{C} \overline{D} \quad \text{Conventional Format} \]

Obviously, some of the variables are missing, so this expression must be expanded to include redundancies.

\[
\begin{align*}
X &= f(A,B,C,D) = \overline{A} \overline{C} \overline{D} + A \overline{B} \overline{C} + \overline{C} \overline{D} \quad \text{Original} \\
X &= f(A,B,C,D) = \overline{A} \overline{C} \overline{D} (B + \overline{B}) + A \overline{B} \overline{C} (D + \overline{D}) + \overline{C} \overline{D} (A + \overline{A})(B + \overline{B}) \quad \text{Expand terms} \\
X &= f(A,B,C,D) = \overline{A} B \overline{C} \overline{D} + \overline{A} \overline{B} C \overline{D} + A B \overline{C} \overline{D} + A B \overline{C} D + \overline{A} B C \overline{D} + \overline{A} B C D + \overline{A} B C D + A B C \overline{D} + A B C \overline{D} + A B C D + A B C D \\
X &= f(A,B,C,D) = \Sigma (7,3,13,12,0,8,4,12) \\
X &= f(A,B,C,D) = \Sigma (0,3,4,7,8,12,13)
\end{align*}
\]

Example #2: Convert the equation SOP expression to its summation format using a Karnaugh map (this is the same expression as example #1 above)

Solution:

\[
\begin{align*}
W &= f(A,B,C) = \overline{A} \overline{B} C + A \overline{B} \overline{C} + A B C \quad \text{Original} \\
\end{align*}
\]

To begin, we create a three variable Karnaugh Map (K-Map) as shown below. The binary numbers across the top and left side represent the values of the variables.

```
\[ \begin{array}{cccc}
A & \bar{A} & 0 & 1 \\
\hline
0 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 \\
\end{array} \]
```

Place the terms from the original expression in the K-Map (because all variables are represented there should be no expansion necessary). The small decimal numbers in each box represent the decimal equivalent of the binary numbers across the top and sides of the K-Map. The decimal numbers in the boxes that

8/23/2008
Example #3: Convert the following SOP Expression to the summation format. This is the same expression as example 3 above

Solution: We will use K-Maps again to solve this example.

\[ Y = \overline{A} + B \overline{C} + \overline{A} C \]  \hspace{2cm} \text{Original}

To begin, we create a three variable Karnaugh Map (K-Map) as shown below. The binary numbers across the top and left side represent the values of the variables.
Place the terms from the original expression in the K-Map. This first diagram shows the placement for the first term (!A)

Place the terms from the original expression in the K-Map. This first diagram shows the placement for the first term (B&!C). Note that some of the 1s are redundant (exactly what we are trying to find!).

Place the terms from the original expression in the K-Map. This first diagram shows the placement for the first term (not A and C). Note that some of the 1s are redundant again.

8/23/2008
Now that we have all the 1s, we can use the K-Map to write the summation expression:

\[ W = f(A, B, C) = \Sigma(0, 1, 2, 3, 6) \]