CHAPTER SIX

Standard Boolean Expression Formats

The development of standards is important to the computer industry for a number of reasons. It allows independently developed systems and subsystems to be connected, it provides reliability through well-tested design methods, and it shortens the design time by providing off-the-shelf tools and components for quick development.

In the design of digital systems, there are some standards that are regularly applied to combinational logic. Over time, design tools and programmable hardware have been developed to support these standards allowing for quick implementation of digital logic.

This chapter outlines two standard representations of combinational logic: Sum-of-Products and Product-of-Sums. Both of these formats represent the fastest possible digital circuitry since, aside from a possible inverter, all of the signals pass through exactly two layers of logic gates. This also opens the door for the development of programmable hardware where a single computer chip can be programmed to handle any logic circuit.

6.1 Sum-of-Products

A sum-of-products (SOP) expression is a boolean expression in a specific format. The term sum-of-products comes from the expression's form: a sum (OR) of one or more products (AND). As a digital circuit, an SOP expression takes the output of one or more AND gates and OR's them together to create the final output.

The inputs to the AND gates are either inverted or non-inverted input signals. This limits the number of gates that any input signal passes through before reaching the output to an inverter, an AND gate, and an OR gate. Since each gate causes a delay in the transition from input to output, and since the SOP format forces all signals to go through exactly two gates (not counting the inverters), an SOP expression gives us predictable performance regardless of which input in a combinational logic circuit changes.

Below is an example of an SOP expression:

$$\overline{A}BC\overline{D} + A\overline{B}D + C\overline{D} + A\overline{D}$$

There are no parentheses in an SOP expression since they would necessitate additional levels of logic. This also means that an SOP expression cannot have more than one variable combined in a term with an inversion bar. The following is *not* an SOP expression:

$$(\overline{AB})C\overline{D} + A\overline{B}D + C\overline{D} + A\overline{D}$$

This is because the first term has A and B passing through a NAND gate before being AND'ed with C and D thereby creating a third level of logic. To fix this problem, we need to break up the NAND using DeMorgan's Theorem.

$$(\overline{AB})C\overline{D} + A\overline{B}D + C\overline{D} + A\overline{D}$$

$$(\overline{A} + \overline{B})C\overline{D} + A\overline{B}D + C\overline{D} + A\overline{D}$$

$$\overline{A}C\overline{D} + \overline{B}C\overline{D} + A\overline{B}D + C\overline{D} + A\overline{D}$$

This expression is now considered to be in SOP format.

As far as the implementation of an SOP expression is concerned, combinations of non-inverted and inverted signals are input to one or more AND gates. The outputs from these gates are all input to a single OR gate. Figure 6-1 shows a sample SOP binary circuit.

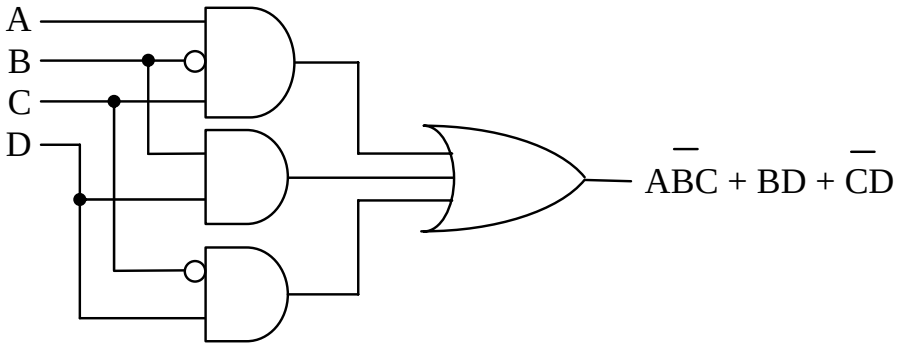


Figure 6-1 Sample Sum-of-Products Binary Circuit

6.2 Converting an SOP Expression to a Truth Table

Examining the truth table for an AND gate reveals that exactly one row has a one for its output. All of the other rows have a zero output. If we invert one of the inputs, this simply moves the row with the one

output to another position. There is still only one row outputting a one. Figure 6-2 shows some examples of this behavior.

A	B	C	A·B·C
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

A	B	C	$A \cdot B \cdot \overline{C}$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

A	B	C	$A \cdot \overline{B} \cdot \overline{C}$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Figure 6-2 Samples of Single Product (AND) Truth Tables

The output of an OR gate is a one if any of the inputs is a one. Therefore, when the products are OR'ed together, a one appears in the output column for each of the products. For example, if we OR'ed together each of the products from Figure 6-2, a one would be output in the rows corresponding to A=1, B=1, and C=1; A=1, B=0, and C=1; and A=0, B=1, and C=0. This is shown in Figure 6-3.

A	B	C	A·B·C	$A \cdot B \cdot \overline{C}$	$A \cdot \overline{B} \cdot \overline{C}$	$ABC + A\overline{B}C + \overline{A}\overline{B}\overline{C}$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	1	1
0	1	1	0	0	0	0
1	0	0	0	0	0	0
1	0	1	0	1	0	1
1	1	0	0	0	0	0
1	1	1	1	0	0	1

Figure 6-3 Sample of a Sum-of-Products Truth Table

Therefore, to convert an SOP expression to a truth table, examine each product to determine when it is equal to a one. Where that product is a one, a one will also be outputted from the OR gate.

Each of the products in the above example contains all of the input variables for the SOP expression. What if one of the products doesn't do this? For example, what if an SOP expression has inputs A, B, and C, but one of its products only depends on A and B?

This is not a problem if we remember that we are looking to see when that product is equal to a one. For a product containing only A and B in an SOP expression containing inputs A, B, and C, the product has ones in ***two rows***, one for C=0 and one for C=1. As an example, let's convert the following SOP expression to a truth table.

$$\overline{A}\,\overline{B}C + A\overline{B} + ABC$$

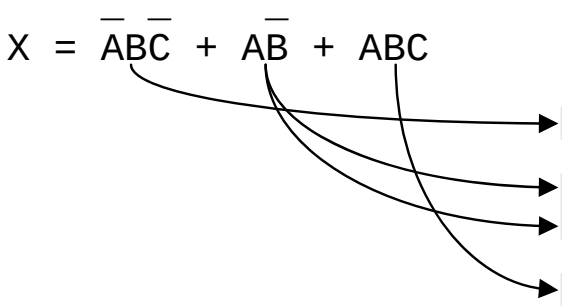
The first step is to determine where each product equals a one. Beginning with the first term, the different input conditions resulting in the output of a logic one from the AND gates are listed below.

$\overline{A}\,\overline{B}C = 1$ when $\overline{A}=1$, $B=1$, and $\overline{C}=1$,
which means when $A=0$, $B=1$, and $C=0$.

$A\overline{B} = 1$ when $A=1$, $\overline{B}=1$, and $C=1$ or 0,
which means when $A=1$, $B=0$, and $C=0$ or 1, i.e., two rows will have a one output due to this term.

$ABC = 1$ when $A=1$, $B=1$, and $C=1$.

Placing a one in each row identified above should result in the truth table for the corresponding SOP expression. Remember to set the remaining row outputs to zero to complete the table.



A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Figure 6-4 Conversion of an SOP Expression to a Truth Table

6.3 Converting a Truth Table to an SOP Expression

Any truth table can be converted to an SOP expression. The conversion process goes like this: identify the rows with ones as the output, and then come up with the unique product to put a one in that row. Note that this will give us an SOP expression where all of the

products use all of the variables for inputs. This usually gives us an expression that can be simplified using the tools from Chapter 5.

Example

Derive the SOP expression for the following truth table.

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Solution

First, identify each row that contains a one as the output.

A	B	C	X	
0	0	0	0	
0	0	1	1	A = 0, B = 0, and C = 1
0	1	0	0	
0	1	1	1	A = 0, B = 1, and C = 1
1	0	0	1	A = 1, B = 0, and C = 0
1	0	1	0	
1	1	0	1	A = 1, B = 1, and C = 0
1	1	1	0	

Now we need to make a product for each of these rows. The product that outputs a one for the row where A=0, B=0, and C=1 must invert A and B in order to have a product of $1 \cdot 1 \cdot 1 = 1$. Therefore, our product is:

$$\overline{A} \cdot \overline{B} \cdot C$$

The product that outputs a one for the row where A=0, B=1, and C=1 must invert A in order to have a product of $1 \cdot 1 \cdot 1 = 1$. This gives us our second product:

$$\overline{A} \cdot B \cdot C$$

The third product outputs a one for the row where A=1, B=0, and C=0. Therefore, we must invert B and C in order to have a product of 1·1·1 = 1.

$$A \cdot \overline{B} \cdot \overline{C}$$

The final product outputs a one for the row where A=1, B=1, and C=0. This time only C must be inverted.

$$A \cdot B \cdot \overline{C}$$

OR'ing all of these products together gives us our SOP expression.

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

The next three sections parallel the first three for a second standard boolean expression format: the product-of-sums.

6.4 Product-of-Sums

The product-of-sums (POS) format of a boolean expression is much like the SOP format with its two levels of logic (not counting inverters). The difference is that the outputs of multiple OR gates are combined with a single AND gate which outputs the final result. The expression below adheres to the format of a POS expression.

$$(\overline{A}+B+C+\overline{D})(A+\overline{B}+D)(C+\overline{D})(A+\overline{D})$$

As with SOP expressions, a POS expression cannot have more than one variable combined in a term with an inversion bar. For example, the following is not a POS expression:

$$(\overline{A+B+C+D})(A+\overline{B}+D)(C+\overline{D})(A+\overline{D})$$

In this example, the sum (OR) of A, B, and C is inverted thereby adding a third level of logic: A, B, and C are OR'ed together then inverted and then OR'ed with D before going to the AND gate. Getting this expression to adhere to the proper POS format where the NOT is distributed to the individual terms is not as easy as it was with the SOP. Often times it is easier to determine the truth table for the function and then convert that truth table to the correct POS format. This will be shown in a later section in this chapter.

As far as hardware is concerned, POS expressions take the output of OR gates and connect them to the inputs of a single AND gate. The sample circuit shown in Figure 6-5 adheres to this format.

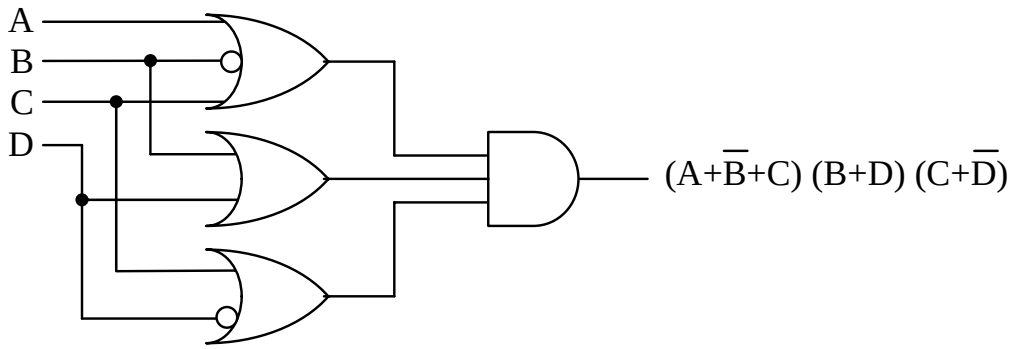


Figure 6-5 Sample Product-of-Sums Binary Circuit

6.5 Converting POS to Truth Table

Converting a POS expression to a truth table follows a similar process as the one used to convert an SOP expression to a truth table. The difference is this: where the SOP conversion focuses on rows with a one output, the POS conversion focuses on rows with a zero output.

We do this because the OR gate has an output of zero on exactly one row while all of the other rows have an output of one. If we invert one of the inputs, this moves the row with the zero output to another position. There is still only one row outputting a zero.

The row with the zero output is the row where all of the inputs equal zero. If one of the inputs is inverted, then its non-inverted value must be one for the OR gate to output a zero. Figure 6-6 shows a few examples of this behavior.

A	B	C	A+B+C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

A	B	C	A+$\overline{B}$+C
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

A	B	C	$\overline{A}$+B+$\overline{C}$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Figure 6-6 Samples of Single Sum (OR) Truth Tables

By AND'ing the output from these OR gates together, then the final output will be zero anytime one of the OR gates outputs a zero. Therefore, AND'ing the three sums in Figure 6-6 together will produce a zero output on the following conditions:

A=0, B=0. and C=0
A=0, B=1. and C=0
A=1, B=0. and C=1

This is shown in Figure 6-7.

A	B	C	A+B+C	$A+\overline{B}+C$	$\overline{A}+\overline{B}+\overline{C}$	$(A+B+C)(A+\overline{B}+C)(\overline{A}+\overline{B}+\overline{C})$
0	0	0	0	1	1	0
0	0	1	1	1	1	1
0	1	0	1	0	1	0
0	1	1	1	1	1	1
1	0	0	1	1	1	1
1	0	1	1	1	0	0
1	1	0	1	1	1	1
1	1	1	1	1	1	1

Figure 6-7 Sample of a Product-of-Sums Truth Table

Therefore, to convert a POS expression to a truth table, examine each of the sums to determine where the sum is equal to zero. When that sum is equal to a zero, a zero will also be present at the final output of the circuit.

When a sum does not contain all of the circuit's inputs, then more than one row will get a zero output from the OR gate. Every time an input drops out of a sum, the number of rows with a zero output from that OR gate is doubled.

For example, if a POS expression uses as its inputs A, B, C, and D, then a sum within that expression that uses only B, C, and D as inputs will have two rows with zero outputs and a sum using only A and C as inputs will have four rows with zero outputs.

The output of the first sum is equal to zero only when all of the inputs, A, B and the inverse of D, are equal to zero. This occurs in two places, once for C=0 and once for C=1. Therefore, the output of a product-of-sums circuit with this OR expression in it will have a zero in the rows where A=0, B=0, C=0, and D=1 and where A=0, B=0, C=1, and D=1.

The next sum uses only B and C from the four inputs. Therefore, there must be four rows with outputs of zero. This is because A and D have no effect on this sum and can have any of the four states: A=0 and D=0; A=0 and D=1; A=1 and D=0; or A=1 and D=1.

A single variable sum as shown in the last column of the truth table in Figure 6-8 will force zeros to be output for half of the input conditions. In the case of this truth table, the inverse of A equals zero when A equals 1.

A	B	C	D	$A+B+\overline{D}$	$B+\overline{C}$	$\overline{A}$
0	0	0	0	1	1	1
0	0	0	1	0	1	1
0	0	1	0	1	0	1
0	0	1	1	0	0	1
0	1	0	0	1	1	1
0	1	0	1	1	1	1
0	1	1	0	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	1	0
1	0	0	1	1	1	0
1	0	1	0	1	0	0
1	0	1	1	1	0	0
1	1	0	0	1	1	0
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	1	0

Figure 6-8 Sample Sums With Multiple Zero Outputs

Example

Convert the following POS expression to a truth table.

$$(\overline{A}+B+\overline{C})(A+\overline{B})(A+B+C)$$

Solution

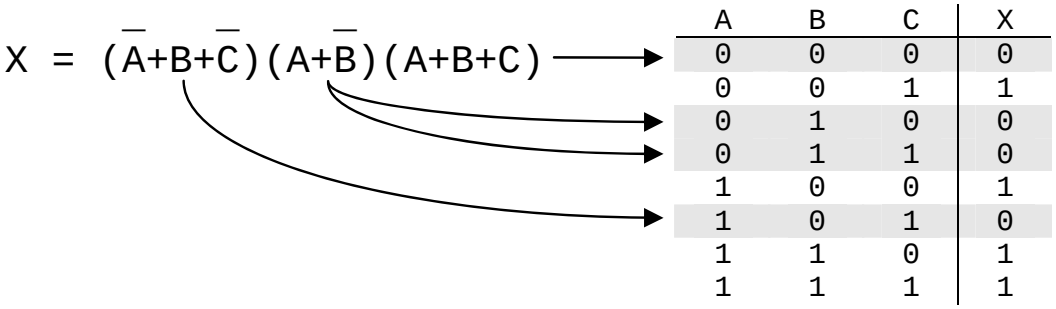
The first step is to determine where each sum equals zero. Beginning with the first term, the three different conditions for a zero output are listed below.

$\overline{A}+B+\overline{C} = 0$ when $\overline{A}=0$, $B=0$, and $\overline{C}=0$,
which means when $A=1$, $B=0$, and $C=1$.

$A+\overline{B} = 0$ when $A=0$, $\overline{B}=0$, and $C=1$ or 0,
which means when $A=0$, $B=1$, and $C=0$ or 1, i.e.,
two rows will have a zero output due to this term.

$A+B+C = 0$ when $A=0$, $B=0$, and $C=0$.

Placing a zero in each row identified above should result in the truth table for the corresponding POS expression. Remember to set the remaining row outputs to zero to complete the table.



A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Figure 6-9 Conversion of a POS Expression to a Truth Table

6.6 Converting a Truth Table to a POS Expression

Just as with SOP expressions, any truth table can be converted to a POS expression. The conversion process goes like this: identify the rows with zeros as the output, and then come up with the unique sum to put a zero in that row. The final group of sums can then be AND'ed together producing the POS expression.

Let's go through the process by deriving the POS expression for the following truth table.

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

First, identify each row that contains a zero as the output.

A	B	C	X	
0	0	0	0	A = 0, B = 0, and C = 0
0	0	1	1	
0	1	0	0	A = 0, B = 1, and C = 0
0	1	1	1	
1	0	0	1	
1	0	1	0	A = 1, B = 0, and C = 1
1	1	0	1	
1	1	1	0	A = 1, B = 1, and C = 1

Next, make a sum for each of these rows. Remember that a sum outputs a zero when all of its inputs equal zero. Therefore, to make a sum for a row equal to zero, the inputs equal to one must be inverted. For the first row, a zero is output when A=0, B=0, and C=0. Since for this case, all of the inputs are already zero, simply OR the non-inverted inputs together.

$$A+B+C$$

The sum that outputs a zero for the row where A=0, B=1, and C=0 must invert B in order to have a sum of 0+0+0=0. This gives us our second sum:

$$A+\overline{B}+C$$

The third sum outputs a zero for the row where A=1, B=0, and C=1. Therefore, we must invert A and C in order to have a sum of 0+0+0=0.

$$\overline{A}+B+\overline{C}$$

The final sum outputs a zero for the row where A=1, B=1, and C=1. In this case, all of the inputs must be inverted to get the sum 0+0+0=0.

$$\overline{A}+\overline{B}+\overline{C}$$

AND'ing all of these sums together gives us our POS expression.

$$(A+B+C)(A+\overline{B}+C)(\overline{A}+B+\overline{C})(\overline{A}+\overline{B}+\overline{C})$$

6.7 NAND-NAND Logic

Chapter 5 presented DeMorgan's Theorem and its application to Boolean expressions. The theorem is repeated below for convenience.

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

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

Figure 6-10 depicts DeMorgan's Theorem with circuit diagrams.

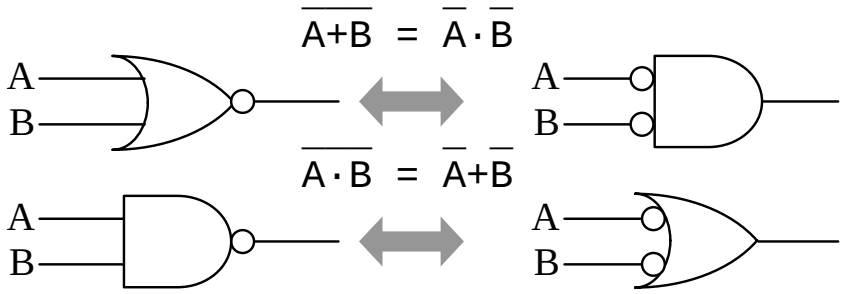


Figure 6-10 Circuit Depiction of DeMorgan's Theorem

If we invert both sides of the first expression where the inverse of the sum is equal to the product of the inverses, then we get the new circuit equivalents shown in Figure 6-11. Note that the double inverse over the OR cancels.

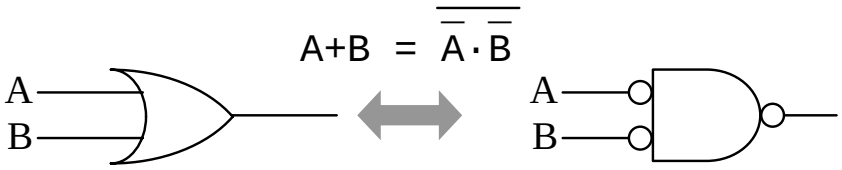


Figure 6-11 OR Gate Equals a NAND Gate With Inverted Inputs

This means that an OR gate operates identically to a NAND gate with inverted inputs. This is also true for OR gates with three or more inputs.

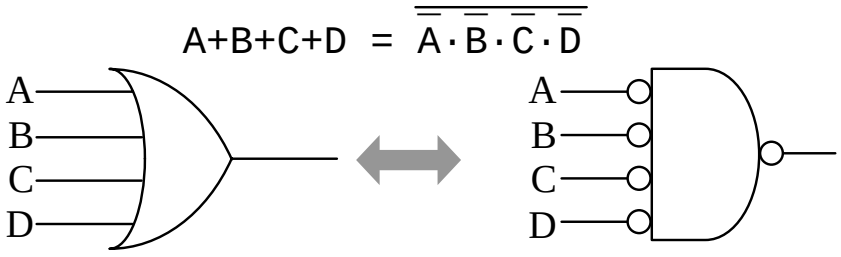


Figure 6-12 OR-to-NAND Equivalency Expanded to Four Inputs

Now let's turn our attention to SOP expressions. Assume we have an equation like the one below.

$$X = (\overline{A} \cdot B \cdot \overline{C}) + (\overline{A} \cdot \overline{B} \cdot C) + (A \cdot B \cdot \overline{C})$$

Figure 6-13 shows the logic circuit equivalent of this expression.

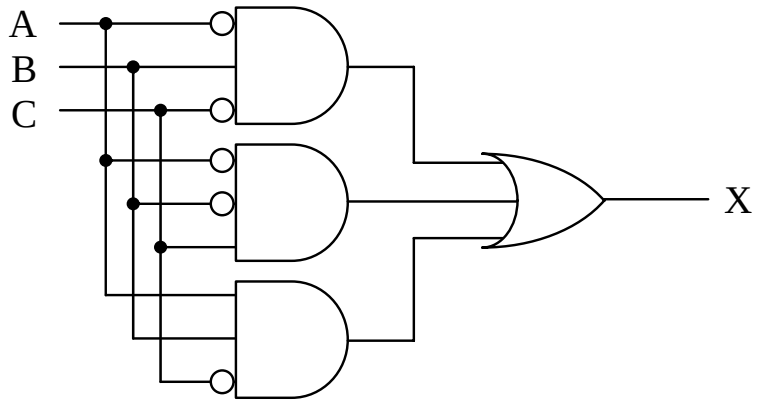


Figure 6-13 Sample SOP Circuit

If we substitute the OR gate with a NAND gate with inverted inputs, we get the circuit shown in Figure 6-14.

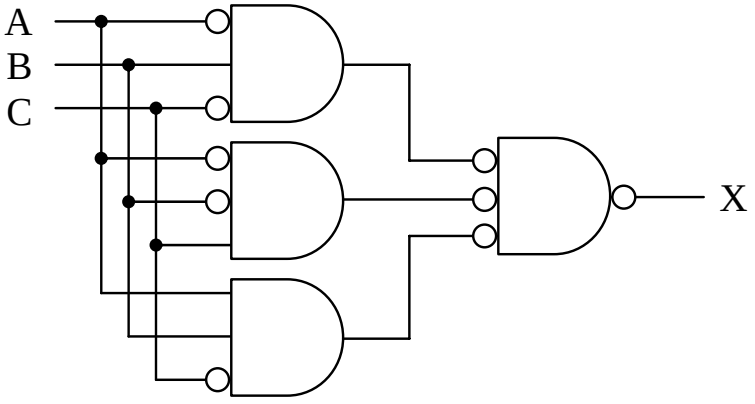


Figure 6-14 Sample SOP Circuit with Output OR Gate Replaced

If we take each of the inverter circles at the inputs to the rightmost gate (the NAND gate that replaced the OR gate), and move them to the outputs of the AND gates, we get the circuit shown in Figure 6-15.

This example shows that all of the gates of a sum-of-products expression, both the AND gates and the OR gate, can be replaced with

NAND gates. Though this doesn't appear at first to be significant, it is important to the implementation of digital logic.

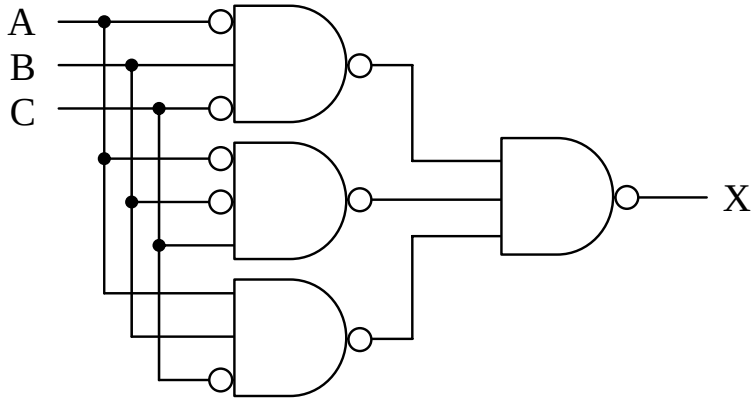


Figure 6-15 Sample SOP Circuit Implemented With NAND Gates

Since an SOP expression can be created for any truth table, then any truth table can be implemented entirely with NAND gates. This allows a designer to create an entire digital system from a single type of gate resulting in a more efficient use of the hardware.

There is an additional benefit to this observation. For most of the technologies used to implement digital logic, the NAND gate is the fastest available gate. Therefore, if a circuit can maintain the same structure while using the fastest possible gate, the overall circuit will be faster.

Another way to make digital circuits faster is to reduce the number of gates in the circuit or to reduce the number of inputs to each of the circuit's gates. This will not only benefit hardware, but also any software that uses logic processes in its operation. Chapter 7 presents a graphical method for generating optimally reduced SOP expressions without using the laws and rules of boolean algebra.

6.8 What's Next?

At this point, we should be able to convert any truth table to a boolean expression and finally to digital circuitry. It should also be clear that no truth table is represented by a unique circuit.

Chapter 7 introduces a simple graphical tool that uses the distributive law to generate the most reduced form of the SOP circuit for a given truth table. As long as the user can follow a set of rules used to generate the products for the circuit, it is a fail-safe tool to make simplified hardware.

Problems

1. Which of the following boolean expressions are in proper sum-of-products form?

   a.) $\overline{A} \cdot B \cdot \overline{D} \cdot \overline{E} + \overline{B} \cdot \overline{C} \cdot D$

   b.) $A \cdot B \cdot C + A \cdot \overline{B} \cdot \overline{C} + \overline{A}(\overline{B} \cdot \overline{C} + B \cdot C)$

   c.) $A + \overline{B} \cdot C \cdot E + D \cdot \overline{E}$

   d.) $A \cdot D + A \cdot \overline{C} + \overline{A \cdot B} \cdot C \cdot D$

   e.) $\overline{B} \cdot C + B \cdot D + \overline{B} \cdot (E + \overline{F})$

2. If a POS expression uses five input variables and has a sum within it that uses only three variables, how many rows in the POS expression's truth table have zeros as a result of that sum?
3. Draw the digital circuit corresponding to the following SOP expressions.

   a.) $\overline{A} \cdot B \cdot \overline{D} \cdot \overline{E} + A \cdot \overline{B} \cdot \overline{C} \cdot D$

   b.) $A \cdot B \cdot C + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot C$

   c.) $A \cdot \overline{C} + \overline{B} \cdot C \cdot E + D \cdot \overline{E}$

4. Draw the NAND-NAND digital logic circuit for each of the SOP expressions shown in problem 3.
5. List the two reasons why the NAND-NAND implementation of an SOP expression is preferred over an AND-OR implementation.
6. Put the following boolean expression into the proper SOP format.

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

7. Which form of boolean expression, SOP or POS, would best be used to implement the following truth table?

A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

8. Derive the SOP and POS expressions for each of the truth tables shown below.

a.)

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

b.)

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

c.)

A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

9. Derive the truth table for each of the following SOP expressions.

a.) $\overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot C + \overline{A} \cdot \overline{B} \cdot C$

b.) $\overline{A} + A \cdot \overline{B} \cdot \overline{C} + A \cdot B \cdot \overline{C}$

c.) $A \cdot \overline{C} + \overline{A} \cdot B \cdot C + \overline{A} \cdot \overline{B} \cdot \overline{C}$

10. Derive the truth table for each of the following POS expressions.

a.) $(\overline{A}+B+\overline{C}) \cdot (A+\overline{B}+C) \cdot (\overline{A}+\overline{B}+C)$

b.) $(A + B) \cdot (\overline{A} + \overline{C})$

c.) $(A+\overline{C}) \cdot (\overline{A}+B+C) \cdot (\overline{A}+\overline{B}+\overline{C})$