# 5 BOOLEAN EXPRESSIONS

**What to Expect**

A Boolean expression uses the various Boolean functions to create a mathematical model of a digital logic circuit. That model can then be used to build a circuit in a simulator like *Logisim-Evolution*. The following topics are included in this chapter.

- Creating a Boolean expression from a description
- Analyzing a circuit's properties to develop a minterm or maxterm expression
- Determining if a Boolean expression is in canonical form
- Converting a Boolean expression with missing terms to its canonical equivalent
- Simplifying a complex Boolean expression using algebraic methods

## 5.1 INTRODUCTION

Electronic circuits that do not require any memory devices (like flip-flops or registers) are created using what is called "Combinational Logic." These systems can be quite complex, but all outputs are determined solely by input signals that are processed through a series of logic gates. Combinational circuits can be reduced to a Boolean Algebra expression, though it may be quite complex; and that expression can be simplified using methods developed in this chapter and Chapter 6, Practice Problems, page 117, and Chapter 7, One In First Cell, page 145. Combinational circuits are covered in Chapter 8, 32x8, page 173.

In contrast, electronic circuits that require memory devices (like flip-flops or registers) use what is called "Sequential Logic." Those circuits often include feedback loops, so the final output is determined by input signals plus the feedback loops that are processed through a series of logic gates. This makes sequential logic circuits much more complex than combinational and the simplification of those circuits is covered in Chapter 9, Comparators, page 189.

Finally, most complex circuits include both combinational and sequential sub-circuits. In that case, the various sub-circuits would be independently simplified using appropriate methods. Several examples of these types of circuits are analyzed in Chapter 10, Sample Problems, page 211.

## 5.2 CREATING BOOLEAN EXPRESSIONS

A circuit designer is often only given a written (or oral) description of a circuit and then asked to build that device. Too often, the designer may receive notes scribbled on the back of a dinner napkin, along with some verbal description of the desired output, and be expected to build a circuit to accomplish that task. Regardless of the form of the request, the process that the designer follows, in general, is:

1. WRITE THE PROBLEM STATEMENT. The problem to be solved is written in a clear, concise statement. The better this statement is written the easier each of the following steps will be, so time spent polishing the problem statement is worthwhile.
2. CONSTRUCT A TRUTH TABLE. Once the problem is clearly defined, the circuit designer constructs a truth table where all inputs/outputs are included. It is essential that all possible input combinations that lead to a *True* output are identified.
3. WRITE A BOOLEAN EXPRESSION. When the truth table is completed, it is easy to create a Boolean expression from that table as covered in Example 5.2.1.
4. SIMPLIFY THE BOOLEAN EXPRESSION. The expression should be simplified as much as possible, and that process is covered in this chapter, Chapter 6, Practice Problems, page 117, and Chapter 7, One In First Cell, page 145.
5. DRAW THE LOGIC DIAGRAM. The logic diagram for a circuit is constructed from the simplified Boolean expression.
6. BUILD THE CIRCUIT. If desired, a physical circuit can be built using the logic diagram.

### 5.2.1 *Example*

A machine is to be programmed to help pack shipping boxes for the ABC Novelty Company. They are running a promotion so if a customer purchases any two of the following items, but not all three, a free poster will be added to the purchase: joy buzzer, fake blood, itching powder. Design the logic needed to add the poster to appropriate orders.

1. PROBLEM STATEMENT. The problem is already fairly well stated. A circuit is needed that will activate the "drop poster" machine when any two of three inputs (joy buzzer, fake blood, itching powder), but not all three, are *True*.

2. TRUTH TABLE. Let J be the Joy Buzzer, B be the Fake Blood, and P be the Itching Powder; and let the truth table inputs be *True* (or 1) when any of those items are present in the shipping box. Let the output D be for "Drop Poster" and when it is *True* (or 1) then a poster will be dropped into the shipping box. The Truth table is illustrated in Table 5.1.

| Inputs | | | Output |
|---|---|---|---|
| J | B | P | D |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Table 5.1: Truth Table for Example

3. WRITE BOOLEAN EXPRESSION. According to the Truth Table, the poster will be dropped into the shipping box in only three cases (when output D is *True*). Equation 5.1 was generated from the truth table.

$$BP + JP + JB = D \tag{5.1}$$

4. SIMPLIFY BOOLEAN EXPRESSION. The Boolean expression for this problem is already as simple as possible so no further simplification is needed.

5. DRAW LOGIC DIAGRAM. Figure 5.1 was drawn from the switching equation.

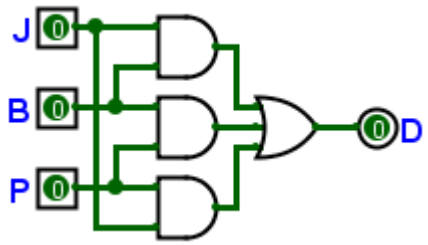


Figure 5.1: Logic Diagram From Switching Equation

6. BUILD THE CIRCUIT. This circuit could be built (or "realized") with three AND gates and one 3-input OR gate.

## 5.3 MINTERMS AND MAXTERMS

### 5.3.1 *Introduction*

The solution to a Boolean equation is normally expressed in one of two formats: Sum of Products (SOP) or Product of Sums (POS).

### 5.3.2 *Sum Of Products (SOP) Defined*

Equation 5.2 is an example of a SOP expression. Notice that the expression describes four inputs (A, B, C, D) that are combined through two AND gates and then the output of those AND gates are combined through an OR gate.

$$(A'BC'D) + (AB'CD) = Y \tag{5.2}$$

Each of the two terms in this expression is a *minterm*. Minterms can be identified in a Boolean expression as a group of inputs joined by an AND gate and then two or more minterms are combined with an OR gate. The circuit illustrated in Figure 5.2 would realize Equation 5.2.

*Notice the inverting bubble on three of the AND gate inputs.*

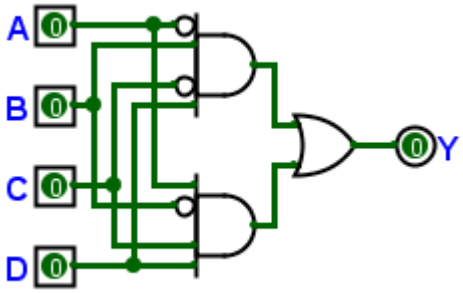


Figure 5.2: Logic Diagram For SOP Example

### 5.3.3 *Product of Sums (POS) Defined*

Equation 5.3 is an example of a POS expression. Notice that the expression describes four inputs (A, B, C, D) that are combined through two OR gates and then the output of those OR gates are combined through an AND gate.

$$(A' + B + C +' D)(A + B' + C + D) = Y \tag{5.3}$$

Each term in this expression is called a *maxterm*. Maxterms can be identified in a Boolean expression as a group of inputs joined by an OR gate; and then two or more maxterms are combined with an AND gate. The circuit illustrated in Figure 5.3 would realize Equation 5.3.

*Notice the inverting bubble on three of the OR gate inputs.*

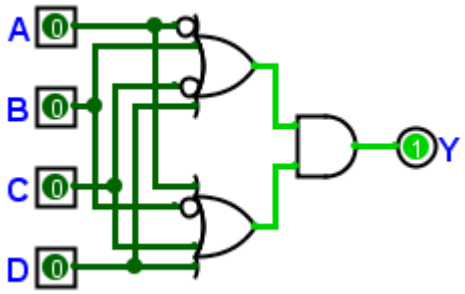


Figure 5.3: Logic Diagram For POS Example

### 5.3.4 *About Minterms*

A term that contains all of the input variables in one row of a truth table joined with an AND gate is called a minterm. Consider truth table 5.2 which is for a circuit with three inputs ( A, B, and C) and one output ( Q).

| Inputs | | | Outputs | |
|---|---|---|---|---|
| A | B | C | Q | m |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 2 |
| 0 | 1 | 1 | 1 | 3 |
| 1 | 0 | 0 | 0 | 4 |
| 1 | 0 | 1 | 1 | 5 |
| 1 | 1 | 0 | 0 | 6 |
| 1 | 1 | 1 | 0 | 7 |

Table 5.2: Truth Table for First Minterm Example

The circuit that is represented by this truth table would output a *True* in only two cases, when the inputs are A′BC or AB′C. Equation 5.4 describes this circuit.

$$(A'BC) + (AB'C) = Q \tag{5.4}$$

The terms A′BC and AB′C are called *minterms* and they contain every combination of input variables that outputs a *True* when the three inputs are joined by an AND gate. Minterms are most often used to describe circuits that have fewer *True* outputs than *False* (that is, there are fewer 1's than 0's in the output column). In the example above, there are only two *True* outputs with six *False* outputs, so minterms describe the circuit most efficiently.

Minterms are frequently abbreviated with a lower-case m along with a subscript that indicates the decimal value of the variables. For example, A′BC, the first of the *True* outputs in the truth table above, has a binary value of 011, which is a decimal value of 3; thus, the minterm is $m_3$. The other minterm in this equation is $m_5$ since its binary value is 101, which equals decimal 5. It is possible to verbally describe the entire circuit as: $m_3$ OR $m_5$. For convenience, each of the minterm numbers are indicated in the last column of the truth table.

As another example, consider truth table 5.3.

| Inputs | | | | Outputs | |
|---|---|---|---|---|---|
| A | B | C | D | Q | m |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 2 |
| 0 | 0 | 1 | 1 | 0 | 3 |
| 0 | 1 | 0 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 1 | 5 |
| 0 | 1 | 1 | 0 | 1 | 6 |
| 0 | 1 | 1 | 1 | 0 | 7 |
| 1 | 0 | 0 | 0 | 0 | 8 |
| 1 | 0 | 0 | 1 | 0 | 9 |
| 1 | 0 | 1 | 0 | 0 | 10 |
| 1 | 0 | 1 | 1 | 0 | 11 |
| 1 | 1 | 0 | 0 | 0 | 12 |
| 1 | 1 | 0 | 1 | 0 | 13 |
| 1 | 1 | 1 | 0 | 1 | 14 |
| 1 | 1 | 1 | 1 | 0 | 15 |

Table 5.3: Truth Table for Second Minterm Example

Equation 5.5 describes this circuit because these are the rows where the output is *True*.

$$(A'B'C'D') + (A'BC'D) + (A'BCD') + (ABCD') = Q \tag{5.5}$$

These would be minterms $m_0$, $m_5$, $m_6$, and $m_{14}$. Equation 5.6 shows a commonly used, more compact way to express this result.

$$\int(A, B, C, D) = \sum(0, 5, 6, 14) \tag{5.6}$$

Equation 5.6 would read: "For the function of inputs A, B, C, and D, the output is *True* for minterms 0, 5, 6, and 14 when they are combined with an OR gate." This format is called *Sigma Notation*, and it is easy to derive the full Boolean equation from it by remembering that $m_0$ is 0000, or $A'B'C'D'$; $m_5$ is 0101, or $A'BC'D$; $m_6$ is 0110, or $A'BCD'$; and $m_{14}$ is 1110, or $ABCD'$. Therefore, the Boolean equation can be quickly created from the Sigma Notation.

A logic equation that is created using minterms is often called the *Sum of Products* (or *SOP*) since each term is composed of inputs ANDed together ("products") and the terms are then joined by OR gates ("sums").

### 5.3.5 *About Maxterms*

A term that contains all of the input variables joined with an OR gate ("added together") for a *False* output is called a *maxterm*. Consider Truth Table 5.4, which has three inputs ( A, B, and C) and one output ( Q):

| Inputs | | | Outputs | |
|---|---|---|---|---|
| A | B | C | Q | M |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 2 |
| 0 | 1 | 1 | 1 | 3 |
| 1 | 0 | 0 | 1 | 4 |
| 1 | 0 | 1 | 1 | 5 |
| 1 | 1 | 0 | 0 | 6 |
| 1 | 1 | 1 | 1 | 7 |

Table 5.4: Truth Table for First Maxterm Example

The circuit that is represented by this truth table would output a *False* in only two cases. Since there are fewer *False* outputs than *True*, it is easier to create a Boolean equation that would generate the *False* outputs. Because the equation describes the *False* outputs, each term is built by *complementing* the inputs for each of the *False* output lines. After the output groups are defined, they are joined with an AND gate. Equation 5.7 is the Boolean equation for Truth Table 5.4.

$$(A + B' + C)(A' + B' + C) = Q \tag{5.7}$$

The terms $A + B' + C$ and $A' + B' + C$ are called $maxterms$, and they contain the complement of the input variables for each of the *False* output lines. Maxterms are most often used to describe circuits that have fewer *False* outputs than *True*. In Truth Table 5.4, there are only two *False* outputs with six *True* outputs, so maxterms describe the circuit most efficiently.

Maxterms are frequently abbreviated with an upper-case M along with a subscript that indicates the decimal value of the complements of the variables. For example, the complement of $A + B' + C$, the first of the *False* outputs in Truth Table 5.4, is 010, which is a decimal value of 2; thus, the maxterm would be $M_2$. This can be confusing, but remember that the *complements* of the inputs are used to form the expression. Thus, $A + B' + C$ is 010, not 101. The other maxterm in this equation is $M_6$ since the binary value of its complement is 110,

which equals decimal 6. It is possible to describe the entire circuit as a the product of two groups of maxterms: $M_2$ and $M_6$.

As another example, consider Truth Table 5.5.

| Inputs | | | | Outputs | |
|---|---|---|---|---|---|
| A | B | C | D | Q | M |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 2 |
| 0 | 0 | 1 | 1 | 0 | 3 |
| 0 | 1 | 0 | 0 | 1 | 4 |
| 0 | 1 | 0 | 1 | 1 | 5 |
| 0 | 1 | 1 | 0 | 1 | 6 |
| 0 | 1 | 1 | 1 | 1 | 7 |
| 1 | 0 | 0 | 0 | 1 | 8 |
| 1 | 0 | 0 | 1 | 0 | 9 |
| 1 | 0 | 1 | 0 | 1 | 10 |
| 1 | 0 | 1 | 1 | 1 | 11 |
| 1 | 1 | 0 | 0 | 0 | 12 |
| 1 | 1 | 0 | 1 | 1 | 13 |
| 1 | 1 | 1 | 0 | 1 | 14 |
| 1 | 1 | 1 | 1 | 1 | 15 |

Table 5.5: Truth Table for Second Maxterm Example

Equation 5.8 describes this circuit.

$$(A + B + C' + D')(A' + B + C + D')(A' + B' + C + D) = Q \tag{5.8}$$

These would be maxterms $M_3$, $M_9$, and $M_{12}$. Equation 5.9 shows a commonly used, more compact way to express this result.

$$\int(A, B, C, D) = \prod(3, 9, 12) \tag{5.9}$$

Equation 5.9 would read: "For the function of A, B, C, D, the output is *False* for maxterms 3, 9, and 12 when they are combined with an AND gate." This format is called *Pi Notation*, and it is easy to derive the Boolean equation from it. Remember that $M_3$ is 0011, or $A + B + C' + D'$ (the complement of the inputs), $M_9$ is 1001, or $A' + B + C + D'$, and $M_{12}$ is 1100, or $A' + B' + C + D$. The original equation can be quickly created from the Pi Notation.

A logic equation that is created using maxterms is often called the *Product of Sums* (or *POS*) since each term is composed of inputs OR ed together ("sums") and the terms are then joined by AND gates ("products").

### 5.3.6 *Minterm and Maxterm Relationships*

The minterms and maxterms of a circuit have three interesting relationships: equivalence, duality, and inverse. To define and understand these terms, consider Truth Table 5.6 for some unspecified "black box" circuit:

| Inputs | | | Outputs | | Terms | |
|---|---|---|---|---|---|---|
| A | B | C | Q | Q' | minterm | Maxterm |
| 0 | 0 | 0 | 0 | 1 | $A'B'C'$ $(m_0)$ | $A+B+C$ $(M_0)$ |
| 0 | 0 | 1 | 1 | 0 | $A'B'C$ $(m_1)$ | $A+B+C'$ $(M_1)$ |
| 0 | 1 | 0 | 0 | 1 | $A'BC'$ $(m_2)$ | $A+B'+C$ $(M_2)$ |
| 0 | 1 | 1 | 0 | 1 | $A'BC$ $(m_3)$ | $A+B'+C'$ $(M_3)$ |
| 1 | 0 | 0 | 0 | 1 | $AB'C'$ $(m_4)$ | $A'+B+C$ $(M_4)$ |
| 1 | 0 | 1 | 1 | 0 | $AB'C$ $(m_5)$ | $A'+B+C'$ $(M_5)$ |
| 1 | 1 | 0 | 0 | 1 | $ABC'$ $(m_6)$ | $A'+B'+C$ $(M_6)$ |
| 1 | 1 | 1 | 1 | 0 | $ABC$ $(m_7)$ | $A'+B'+C'$ $(M_7)$ |

Table 5.6: Minterm and Maxterm Relationships

#### 5.3.6.1 *Equivalence*

The minterms and the maxterms for a given circuit are considered equivalent ways to describe that circuit. For example, the circuit described by Truth Table 5.6 could be defined using minterms (Equation 5.10).

$$\int(A,B,C) = \sum(1,5,7) \tag{5.10}$$

However, that same circuit could also be defined using maxterms (Equation 5.11).

$$\int(A,B,C) = \prod(0,2,3,4,6) \tag{5.11}$$

These two functions describe the same circuit and are, consequently, equivalent. The *Sigma Function* includes the terms 1, 5, and 7 while the *Pi Function* includes all other terms in the truth table (0, 2, 3, 4, and 6). To put it a slightly different way, the *Sigma Function* describes

the truth table rows where Q = 1 (minterms) while the *Pi Function* describes the rows in the same truth table where Q = 0 (maxterms). Therefore, Equation 5.12 can be derived.

$$\sum(1,5,7) \equiv \prod(0,2,3,4,6) \tag{5.12}$$

#### 5.3.6.2 *Duality*

Each row in Truth Table 5.6 describes two terms that are considered duals. For example, minterm $m_5$ ($AB'C$) and maxterm $M_5$ ($A' + B + C'$) are duals. Terms that are duals are complements of each other (Q vs. Q′) and the input variables are also complements of each other; moreover, the inputs for the three minterms are combined with an AND while the maxterms are combined with an OR. The output of the circuit described by Truth Table 5.6 could be defined using minterms (Equation 5.13).

$$Q = \sum(1,5,7) \tag{5.13}$$

The dual of the circuit would be defined by using the maxterms for the same output rows. However, those rows are the *complement* of the circuit (Equation 5.14).

$$Q' = \prod(1,5,7) \tag{5.14}$$

This leads to the conclusion that the complement of a *Sigma Function* is the *Pi Function* with the same inputs, as in Equation 5.15 (the overline was used to emphasize the the fact that the *PI Function* is complemented).

$$\sum(1,5,7) = \overline{\prod(1,5,7)} \tag{5.15}$$

#### 5.3.6.3 *Inverse*

The complement of a function yields the opposite output. For example the following functions are inverses because one defines Q while the other defines Q′ using only minterms of the same circuit (or truth table).

$$Q = \sum(1,5,7) \tag{5.16}$$

$$Q' = \sum(0,2,3,4,6) \tag{5.17}$$

#### 5.3.6.4 *Summary*

These three relationships are summarized in the following table. Imagine a circuit with two or more inputs and an output of Q. Table 5.7 summarizes the various relationships in the Truth Table for that circuit.

| | |
|---|---|
| Minterms where Q is 1 | Minterms where Q' is 1 |
| Maxterms where Q is 1 | Maxterms where Q' is 1 |

Table 5.7: Minterm-Maxterm Relationships

The adjacent items in a single column are equivalent (that is, Q Minterms are equivalent to Q Maxterms), items that are diagonal are duals ( Q Minterms and $Q'$ Maxterms are duals), and items that are adjacent in a single row are inverses ( Q Minterms and $Q'$ Minterms are inverses).

### 5.3.7 *Sum of Products Example*

#### 5.3.7.1 *Given*

A "vote-counter" machine is designed to turn on a light if any two or more of three inputs are *True*. Create a circuit to realize this machine.

#### 5.3.7.2 *Truth Table*

When realizing a circuit from a verbal description, the best place to start is constructing a truth table. This will make the Boolean expression easy to write and then make the circuit easy to realize. For the "vote-counter" problem, start by defining variables for the truth table: Inputs A, B, C and Output Q.

Next, construct the truth table by identifying columns for each of the three input variables and then indicate the output for every possible input condition (Table 5.8).

| Inputs | | | Outputs | |
|---|---|---|---|---|
| A | B | C | Q | m |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 2 |
| 0 | 1 | 1 | 1 | 3 |
| 1 | 0 | 0 | 0 | 4 |
| 1 | 0 | 1 | 1 | 5 |
| 1 | 1 | 0 | 1 | 6 |
| 1 | 1 | 1 | 1 | 7 |

Table 5.8: Truth Table for SOP Example

#### 5.3.7.3 *Boolean Equation*

In a truth table, if there are fewer *True* outputs than *False*, then it is easiest to construct a Sum-of-Products equation. In that case, a Boolean expression can be derived by creating the minterms for all *True* outputs and combining those minterms with OR gates. Equation 5.18 is the *Sigma Function* of this circuit.

$$\int(A, B, C) = \sum(3, 5, 6, 7) \tag{5.18}$$

At this point, a circuit could be created with four 3-input AND gates combined into one 4-input OR gate.

*It may be possible to simplify the Boolean expression, but that process is covered elsewhere in this book.*

### 5.3.8 *Product of Sums Example*

#### 5.3.8.1 *Given*

A local supply company is designing a machine to sort packages for shipping. All packages go to the post office *except* packages going to the local ZIP code containing chemicals (they are shipped by courier) and packages going to a distant ZIP code containing only perishables (they are shipped via air freight).

#### 5.3.8.2 *Truth Table*

When realizing a circuit from a verbal description, the best place to start is constructing a truth table. This will make the Boolean expression easy to write and then make the circuit easy to realize. For the sorting machine problem, start by defining variables for the truth table:

- Ship via post office (the Output): O is *True* if ship by Post Office
- Zip Code: Z is *True* if the zip code is local
- Chemicals: C is *True* if the package contains chemicals
- Perishable: P is *True* if the package contains perishables

Next, construct the truth table by identifying columns for each of the three input variables and then indicate the output for every possible input condition (Table 5.8).

| Inputs | | | Outputs | |
|---|---|---|---|---|
| Z | C | P | O | M |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 2 |
| 0 | 1 | 1 | 0 | 3 |
| 1 | 0 | 0 | 1 | 4 |
| 1 | 0 | 1 | 1 | 5 |
| 1 | 1 | 0 | 0 | 6 |
| 1 | 1 | 1 | 0 | 7 |

Table 5.9: Truth Table for POS Example

#### 5.3.8.3 *Boolean Equation*

In the truth table, if there are fewer *False* outputs than *True* then it is easiest to construct a Products-of-Sums equation. In that case, a Boolean expression can be derived by creating the maxterms for all *False* outputs and combining the complement those maxterms with AND gates. Equation 5.19 is the *Pi Expression* of this circuit.

$$\int(Z, C, P) = \prod(1, 3, 6, 7) \tag{5.19}$$

*It may be possible to simplify the Boolean expression, but that process is covered elsewhere in this book.*

At this point, a circuit could be created with four 3-input OR gates combined into one 4-input AND gate.

### 5.3.9 *Summary*

SOP Boolean expressions may be generated from truth tables quite easily, by determining which rows of the table have an output of *True*, writing one minterm for each of those rows, and then summing all of the minterms. The resulting expression will lend itself well to

implementation as a set of AND gates (products) feeding into a single OR gate (sum).

POS Boolean expressions may be generated from truth tables quite easily, by determining which rows of the table have an output of *False*, writing one maxterm for each of those rows, and then multiplying all of the maxterms. The resulting expression will lend itself well to implementation as a set of OR gates (sums) feeding into a single AND gate (product).

## 5.4 CANONICAL FORM

### 5.4.1 *Introduction*

The word "canonical" simply means "standard" and it is used throughout mathematics and science to denote some standard form for equations. In digital electronics, Boolean equations are considered to be in canonical form when each of the terms in the equation includes all of the possible inputs and those terms appear in the same order as in the truth table. Using the canonical form is important when simplifying a Boolean equation. For example, imagine the solution to a given problem generated table 5.10.

| Inputs | | | Outputs | |
|---|---|---|---|---|
| A | B | C | Q | m |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 2 |
| 0 | 1 | 1 | 1 | 3 |
| 1 | 0 | 0 | 0 | 4 |
| 1 | 0 | 1 | 0 | 5 |
| 1 | 1 | 0 | 0 | 6 |
| 1 | 1 | 1 | 1 | 7 |

Table 5.10: Canonical Example Truth Table

Minterm equation 5.20 is derived from the truth table and is presented in canonical form. Notice that each term includes all possible inputs ( A, B, and C), and that the terms are in the same order as they appear in the truth table.

$$(A'B'C) + (A'BC) + (ABC) = Q \tag{5.20}$$

Frequently, though, a Boolean equation is expressed in standard form, which is not the same as canonical form. Standard form means

that some of the terms have been simplified and not all of the inputs will appear in all of the terms. For example, consider Equation 5.21, which is the solution for a 4-input circuit.

$$(A'C) + (B'CD) = Q \tag{5.21}$$

This equation is in standard form so the first term, $A'C$, does not include inputs B or D and the second term, $B'CD$, does not include input A. However, all inputs must be present in every term for an equation to be in canonical form.

Building a truth table for an equation in standard form raises an important question. Consider the truth table for Equation 5.21.

| Inputs | | | | Outputs | |
|---|---|---|---|---|---|
| A | B | C | D | Q | m |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 2 |
| 0 | 0 | 1 | 1 | ? | 3 |
| 0 | 1 | 0 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 0 | 5 |
| 0 | 1 | 1 | 0 | 0 | 6 |
| 0 | 1 | 1 | 1 | 0 | 7 |
| 1 | 0 | 0 | 0 | 0 | 8 |
| 1 | 0 | 0 | 1 | 0 | 9 |
| 1 | 0 | 1 | 0 | 0 | 10 |
| 1 | 0 | 1 | 1 | ? | 11 |
| 1 | 1 | 0 | 0 | 0 | 12 |
| 1 | 1 | 0 | 1 | 0 | 13 |
| 1 | 1 | 1 | 0 | 0 | 14 |
| 1 | 1 | 1 | 1 | 0 | 15 |

Table 5.11: Truth Table for Standard Form Equation

In what row would $B'CD$, the second term in Equation 5.21, be placed? $B'CD$ is 011 (in binary), but since the A term is missing would it be a 0 or 1; in other words, would $B'CD$ generate an output of 1 for row 0011 ($m_3$) or 1011 ($m_{11}$)? (The output for these two rows are marked with a question mark in Table 5.11.) In fact, the output for *both* of these rows must be considered *True* in order to ensure that all possible combinations of input are covered. Thus, the final equation for this circuit must include at least these two terms:

$(A'B'CD) + (AB'CD)$. In the same way, the term $A'C$ means that the output is *True* for $m_2$, $m_3$, $m_6$, and $m_7$ since input $A'C$ is *True* and any minterm that contains those two value is also considered *True*. Thus, the final equation for this circuit must include at least these four terms: $(A'B'CD') + (A'B'CD) + (A'BCD') + (A'BCD)$.

### 5.4.2 *Converting Terms Missing One Variable*

To change a standard Boolean expression that is missing one input term into a canonical Boolean expression, insert both *True* and *False* for the missing term into the original standard expression. As an example, consider the term $B'CD$. Since term $A$ is missing, both $A$ and $A'$ must be included in the converted canonical expression. Equation 5.22 proves that $B'CD$ can be expanded to include both possible values for $A$ by using the Adjacency Property (page 84).

$$(B'CD) \rightarrow (AB'CD) + (A'B'CD) \tag{5.22}$$

A term that is missing one input variable will expand into two terms that include all variables. For example, in a system with four input variables (as above), any standard term with only three variables will expand to a canonical expression containing two groups of four variables.

Expanding a standard term that is missing one variable can also be done with a truth table. To do that, fill in an output of 1 for every line where the *True* inputs are found while ignoring all missing variables. As an example, consider Truth Table 5.11 where the outputs for $m_3$ and $m_11$ are marked with a question mark. However, the output for both of these lines should be marked as *True* because $B'CD$ is *True* (input $A$ is ignored). Then, those two minterms lead to the Boolean expression $AB'CD + A'B'CD$.

### 5.4.3 *Converting Terms Missing Two Variables*

It is easiest to expand a standard expression that is missing two terms by first inserting one of the missing variables and then inserting the other missing variable in two distinct steps. The process for inserting a single missing variable is found in Section 5.4.2. Consider the term $A'C$ in a four-variable system. It is missing both the $B$ and $D$ variables. To expand that term to its canonical form, start by inserting either of the two missing variables. For example, Equation 5.23 illustrates entering $B$ and $B'$ into the expression.

$$(A'C) \rightarrow (A'BC) + (A'B'C) \tag{5.23}$$

Then, Equation 5.24 illustrates inserting D and D′ into the expression.

$$\begin{aligned}(A'BC) &\rightarrow (A'BCD) + (A'BCD') \\ (A'B'C) &\rightarrow (A'B'CD) + (A'B'CD')\end{aligned} \tag{5.24}$$

In the end, A′C expands to Equation 5.25:

$$\begin{aligned}(A'C) \rightarrow &(A'BCD) + (A'BCD') \\ &+ (A'B'CD) + (A'B'CD')\end{aligned} \tag{5.25}$$

Thus, in a four-variable system, any standard term with only two variables will expand to a canonical expression with four groups of four variables.

Expanding a standard term that is missing two variables can also be done with a truth table. To do that, fill in an output of 1 for every line where the *True* inputs are found while ignoring all missing variables. As an example, consider a Table 5.12, where A′C is marked as *True*:

| Inputs | | | | Outputs | |
|---|---|---|---|---|---|
| A | B | C | D | Q | m |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 2 |
| 0 | 0 | 1 | 1 | 1 | 3 |
| 0 | 1 | 0 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 0 | 5 |
| 0 | 1 | 1 | 0 | 1 | 6 |
| 0 | 1 | 1 | 1 | 1 | 7 |
| 1 | 0 | 0 | 0 | 0 | 8 |
| 1 | 0 | 0 | 1 | 0 | 9 |
| 1 | 0 | 1 | 0 | 0 | 10 |
| 1 | 0 | 1 | 1 | 0 | 11 |
| 1 | 1 | 0 | 0 | 0 | 12 |
| 1 | 1 | 0 | 1 | 0 | 13 |
| 1 | 1 | 1 | 0 | 0 | 14 |
| 1 | 1 | 1 | 1 | 0 | 15 |

Table 5.12: Truth Table for Standard Form Equation

Notice that outputs for $m_2$, $m_3$, $m_6$, and $m_7$ are *True* because for each of those minterms A′C is *True* (inputs B and D are ignored).

Then, those four minterms lead to the Boolean expression $A'B'CD' + A'B'CD + A'BCD' + A'BCD$.

### 5.4.4 *Summary*

This discussion started with Equation 5.26, which is in standard form.

$$(A'C) + (B'CD) = Q \tag{5.26}$$

After expanding both terms, Equation 5.27 is generated.

$$\begin{aligned}&(A'BCD) + (A'BCD') + (A'B'CD) \\ &+(A'B'CD') + (AB'CD) + (A'B'CD) = Q\end{aligned} \tag{5.27}$$

Notice, though, that the term $A'B'CD$ appears two times, so one of those can be eliminated by the Idempotence property (page 77), leaving Equation 5.28.

$$\begin{aligned}&(A'BCD) + (A'BCD') \\ &+(A'B'CD') + (AB'CD) + (A'B'CD) = Q\end{aligned} \tag{5.28}$$

To put the equation in canonical form, which is important for simplification; all that remains is to rearrange the terms so they are in the same order as they would appear in a truth table, which results in Equation 5.29.

$$\begin{aligned}&(A'B'CD') + (A'B'CD) \\ &+(A'BCD') + (A'BCD) + (AB'CD) = Q\end{aligned} \tag{5.29}$$

### 5.4.5 *Practice Problems*

*Parenthesis were not used in order to save space; however, the variable groups are evident.*

| | | |
|---|---|---|
| **1** | Standard (A,B,C) | $A'B + C + AB'$ |
| | Cannonical | $A'B'C + A'BC' + A'BC + AB'C' + AB'C + ABC$ |
| **2** | Standard (A,B,C,D) | $A'BC + B'D$ |
| | Cannonical | $A'B'C'D' + A'B'CD' + A'BCD' + A'BCD + AB'C'D' + AB'CD'$ |
| **3** | Standard (A,B,C,D) | $A' + D$ |
| | Cannonical | $A'B'C'D' + A'B'C'D + A'B'CD' + A'B'CD + A'BC'D' + A'BC'D + A'BCD' + A'BCD + AB'C'D + AB'CD + ABC'D + ABCD$ |
| **4** | Standard (A,B,C) | $A(B' + C)$ |
| | Cannonical | $AB'C' + AB'C + ABC$ |

Table 5.13: Canonical Form Practice Problems

## 5.5 SIMPLIFICATION USING ALGEBRAIC METHODS

### 5.5.1 *Introduction*

One method of simplifying a Boolean equation is to use common algebraic processes. It is possible to reduce an equation step-by-step using the various properties of Boolean algebra in the same way that real-number equations can be simplified.

### 5.5.2 *Starting From a Circuit*

Occasionally, the circuit designer is faced with an existing circuit and must attempt to simplify it. In that case, the first step is to find the Boolean equation for the circuit and then simplify that equation.

#### 5.5.2.1 *Generate a Boolean Equation*

In the circuit illustrated in Figure 5.4, the A, B, and C input signals are assumed to be provided from switches, sensors, or perhaps other sub-circuits. Where these signals originate is of no concern in the task of gate reduction.

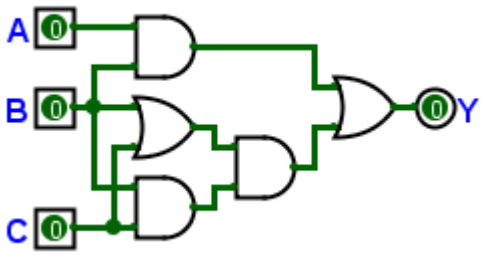


Figure 5.4: Example Circuit

To generate the Boolean equation for a circuit, write the output of each gate as determined by the input signals and type of gate, working from the inputs to the final output. Figure 5.5 illustrates the result of this process.

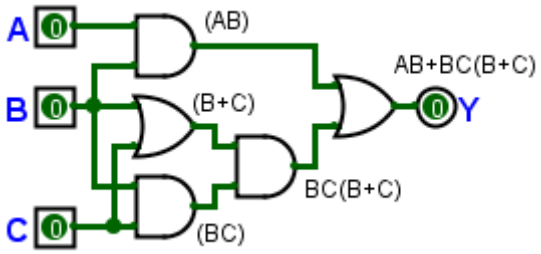


Figure 5.5: Example Circuit With Gate Outputs

This process leads to Equation 5.30.

$$AB + BC(B + C) = Y \tag{5.30}$$

### 5.5.3 *Starting From a Boolean Equation*

If a logic circuit's function is expressed as a Boolean equation, then algebraic methods can be applied to reduce the number of logic gates, resulting in a circuit that performs the same function with fewer components. As an example, Equation 5.31 simplifies the circuit found in Figure **??**.

$$\begin{aligned}
AB + BC(B + C) &\quad \text{Original Expression} \\
AB + BBC + BCC &\quad \text{Distribute BC} \\
AB + BC + BC &\quad \text{Idempotence: BB=B and CC=C} \\
AB + BC &\quad \text{Idempotence: BC+BC=BC} \\
B(A + C) &\quad \text{Factor}
\end{aligned} \tag{5.31}$$

The final expression, $B(A + C)$, requires only two gates, and is much simpler than the original, yet performs the same function. Such component reduction results in higher operating speed (less gate propagation delay), less power consumption, less cost to manufacture, and greater reliability.

As a second example, consider Equation 5.32.

$$A + AB = Y \tag{5.32}$$

This is simplified below.

$$\begin{aligned} A + AB &\quad \text{Original Expression} \\ A(1 + B) &\quad \text{Factor} \\ A(1) &\quad \text{Annihilation (1+B=1)} \\ A &\quad \text{Identity A1=A} \end{aligned} \tag{5.33}$$

The original expression, $A + AB$ has been reduced to $A$ so the original circuit could be replaced by a wire directly from input $A$ to output $Y$. Equation 5.34 looks similar to Equation 5.32, but is quite different and requires a more clever simplification.

$$A + A'B = Y \tag{5.34}$$

This is simplified below.

$$\begin{aligned} A + A'B &\quad \text{Original Expression} \\ A + AB + A'B &\quad \text{Expand A to A+AB (Absorption)} \\ A + B(A + A') &\quad \text{Factor B out of the last two terms} \\ A + B(1) &\quad \text{Complement Property} \\ A + B &\quad \text{Identityt: B(1)=B} \end{aligned} \tag{5.35}$$

Note how the Absorption Property ($A + AB = A$) is used to "un-simplify" the first $A$ term, changing $A$ into $A + AB$. While this may seem like a backward step, it ultimately helped to reduce the expression to something simpler. Sometimes "backward" steps must be taken to achieve the most elegant solution. Knowing when to take such a step is part of the art of algebra.

As another example, simplify this POS expression equation:

$$(A + B)(A + C) = Y \tag{5.36}$$

This is simplified below.

$$
\begin{aligned}
(A+B)(A+C) &\quad \text{Original Expression} \\
AA+AC+AB+BC &\quad \text{Distribute A+B} \\
A+AC+AB+BC &\quad \text{Idempotence: AA=A} \\
A+AB+BC &\quad \text{Absorption: A+AC=A} \\
A+BC &\quad \text{Absorption: A+AB=A}
\end{aligned} \tag{5.37}
$$

In each of the examples in this section, a Boolean expression was simplified using algebraic methods, which led to a reduction in the number of gates needed and made the final circuit more economical to construct and reliable to operate.

### 5.5.4 *Practice Problems*

Table 5.14 shows a Boolean expression on the left and its simplified version on the right. This is provided for practice in simplifying expressions using algebraic methods.

| Original Expression | Simplified |
|---|---|
| $A(A'+B)$ | $AB$ |
| $A+A'B$ | $A+B$ |
| $(A+B)(A+B')$ | $A$ |
| $AB+A'C+BC$ | $AB+A'C$ |

Table 5.14: Simplifying Boolean Expressions