Slide Set 3

for ENEL 353 Fall 2019

Steve Norman, PhD, PEng

Electrical & Computer Engineering
Schulich School of Engineering
University of Calgary

Fall Term, 2019

Contents

Outline of Slide Set 3

Ordinary Algebra

Engineering students are very accustomed to seeing things like

$$(a + b)(c + d) = ac + ad + bc + bd$$

and

$$\textit{If } x^2 + x = 0 \textit{ and } x \neq 0, \textit{ then } x + 1 = 0.$$

Symbols such as a, b, c, d, and x are **variables** that **could have any value** taken from a set such as

- the set of all real numbers, or
- the set of all complex numbers.

Outline of Slide Set 3

Boolean Algebra

In *Boolean algebra*, variables have values taken from this set: $\{0, 1\}$.

Ordinary algebra has **many operators**: addition, subtraction, multiplication, division, and others.

Boolean algebra has **three operators**: NOT, AND, and OR.

Boolean algebra is very useful for description, analysis and design of combinational logic systems.

Note: Older textbooks for ENEL 353 use the term "switching algebra" instead of "Boolean algebra". Watch out for that if you are looking at course material from many years ago.

Terminology for Boolean algebra

There are many words that have special meanings in discussion of Boolean algebra.

Here are some of them: *complement*, *literal*, *product*, *sum*, *minterm*, *maxterm*. There are several others.

Make sure you learn **exact meanings** for all of them! Having only a **rough idea** what they mean is **not good enough**—that will lead to **confusion** and **errors**.

Variables, complements and literals

As stated two slides back, *variables* in Boolean algebra are things that may have one of two values: 0 (FALSE), or 1 (TRUE).

The *complement* of a variable is the NOT of that variable. So the complement of A is $\overline{A}$, the complement of B is $\overline{B}$, and so on.

A *literal* is either a variable or the complement of a variable. *So, if A and B are Boolean variables, which of the following expressions are literals?*

- A
- $\overline{A}$
- B
- $\overline{B}$
- AB
- $A + B$
- $A + \overline{A}$

True form and complementary form

These are terms that distinguish the two kinds of literals.

The *true form* of a variable is just the "plain" version of that variable. For example, the true forms of A, B, etc., are simply A, B, etc.

The *complementary form* of a variable is the NOT of that variable. For example, the complementary forms of A, B, and so on, are $\overline{A}$, $\overline{B}$, and so on.

Products

In Boolean algebra, a *product* is defined to be either

- a literal; or
- the AND of two or more literals.

(Our textbook defines a *product* as "the AND of one or more literals", which is correct, but makes you think harder than necessary about what the AND of one literal is.)

Suppose A, B, and C are Boolean variables. *Which of the following expressions are products?*

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

Minterms

Suppose a function has N input variables. A *minterm* is defined to be a **product** that uses **all** N of the input variables—each variable appearing once in either true or complemented form.

For example, if the input variables are A and B, there are 4 minterms: $\bar{A}\bar{B}$, $\bar{A}B$, $A\bar{B}$, and AB.

If the input variables are A, B and C, how many minterms are there, and what are the minterms?

Sums and maxterms

A *sum* is either

- a literal; or
- two or more literals OR-ed together.

Examples of sums: A, $\overline{B}$, $\overline{A} + B$, $A + B + \overline{C}$.

Examples of expressions that are **not** sums: AB, $\overline{A + C}$, $A + \bar{B}\bar{C}$.

A *maxterm* is defined to be a **sum** that uses **all** N of the input variables—each variable appearing once in either true or complementary form. (This definition differs from the *minterm* definition by only one word!)

For input variables A and B, what are all the maxterms?

Outline of Slide Set 3

Order of operations in Boolean algebra

Order of operations is related to the concept of **operator precedence**.

In Boolean algebra:

- NOT has highest precedence.
- AND is next.
- OR has lowest precedence.

Let's use parentheses to show order of operations in the following expressions.

- $AB + BC$
- $\bar{A}B + B\bar{C}$

Clarification about the notation for NOT

NOT is a **unary operator**—that means NOT applies to a **single operand**.

So when you see an "overline" over a complex expression, that expression has to be evaluated before NOT can be applied.

So, for example,

- $\overline{AB}$ really means $\overline{(AB)}$;
- $\overline{A+B}$ really means $\overline{(A+B)}$;
- $\overline{\overline{A}B + A\overline{B}}$ really means $\overline{\left(\overline{A}B + A\overline{B}\right)}$.

When "overline" or "bar" is used for NOT, it's fairly obvious that something like $\overline{A + B}$ means, "Do the OR first, then the NOT."

When $'$ ("prime") is used for NOT, then it's important to know the operator precedence. For example, $A + B'$ means $A + (B')$, not $(A + B)'$.

We won't use $'$ for NOT this term in ENEL 353, but the point made on this slide is helpful when you read material about Boolean algebra that does use $'$ for NOT.

Associativity of AND and OR operators

We've just seen expressions such as

- $\bar{A}\bar{B}\bar{C}$;
- $A + B + \bar{C}$.

These might raise questions, such as

- Does the first example mean $\left(\bar{A}\bar{B}\right)\bar{C}$ or $\bar{A}\left(\bar{B}\bar{C}\right)$, or does it matter?
- Does the second example mean $(A + B) + \bar{C}$ or $A + \left(B + \bar{C}\right)$, or does it matter?

In fact, AND and OR are both **associative**, which means that

- $(XY)Z = X(YZ)$—it doesn't matter which AND is done first. So we usually just write XYZ.
- $(X + Y) + Z = X + (Y + Z)$, so we usually just write $X + Y + Z$.

Let's **prove** *that* $(XY)Z = X(YZ)$.

$(X + Y) + Z = X + (Y + Z)$ can be proved in a similar manner.

Recall that the N-input AND function was defined as

$$\text{AND}\,(A_1, A_2, \ldots, A_N) = \begin{cases} 1 & \text{if } \textbf{all} \text{ of } A_1, A_2, \ldots, A_N \text{ are } 1 \\ 0 & \text{otherwise} \end{cases}$$

Because the AND operator is associative, we can write

$$\text{AND}\,(A_1, A_2, \ldots, A_N) = A_1 A_2 \cdots A_N.$$

Similarly for N-input OR, we can write

$$\text{OR}\,(A_1, A_2, \ldots, A_N) = A_1 + A_2 + \cdots + A_N.$$

Outline of Slide Set 3

Sum-of-products expressions

A *sum-of-products*—often abbreviated as SOP—is defined to be either

- a single product, or
- two or more products, ORed together.

With variables A, B and C, which of the following are SOP expressions?

1. $\bar{A}C$
2. $A + B + C$
3. $A + \bar{A}BC$
4. $\bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$
5. $A(\bar{B}\bar{C} + BC)$
6. $\overline{AB + \bar{B}C}$

Sum-of-products canonical form

Sum-of-products canonical form is defined to be an SOP expression for a Boolean function in which **all the products are minterms**.

With variables A, B and C, which of the following expressions are in SOP canonical form?

1. $\bar{A}C$
2. $A + B + C$
3. $A + \bar{A}BC$
4. $\bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$
5. $A(\bar{B}\bar{C} + BC)$
6. $\overline{AB + \bar{B}C}$

Using SOP canonical form to get a Boolean equation from a truth table

Suppose you have a truth table for some function F, which is a function of some number of Boolean variables.

Then there is a straightforward procedure for finding the SOP canonical form equation for F.

- **Step 1:** For each row in the table that has $F = 1$, find which **minterm** is **true** for that row.
- **Step 2:** OR together all the minterms found in Step 1.

Let's use this procedure to find SOP canonical form equations for

- $F = \overline{A \oplus B}$ *(the XNOR function), and*
- *the sum function of a 1-bit full adder.*

Minterm numbering

Consider a truth table involving N variables. We can number the rows of the table starting at 0 and ending with $2^N - 1$.

The number of a row will be the value of the unsigned binary number made up of the 0's and 1's in that row on the input side of the table.

m_k, read as "minterm k", is the minterm that is true for the input values in row k of the table.

Let's determine the minterm numbering for the cases of (a) two input variables and (b) three input variables.

Minterm numbering and shorthand for SOP canonical form

There are a couple of compact ways to use minterm numbers to write SOP canonical forms. These avoid writing out all the literals involved, and are especially handy for functions of 3 or more variables with lots of minterms.

For example, if SOP canonical form for $F(A,B,C)$ is the OR of minterms m_0, m_1, m_3, m_4, and m_6, we can write

$$F = \Sigma(m_0, m_1, m_3, m_4, m_6)$$

or just

$$F = \Sigma(0,1,3,4,6).$$

Minterm numbering and shorthand for SOP canonical form: Examples

Let's write the SOP canonical form for $F(A,B) = \overline{A \oplus B}$ *using minterm numbers.*

Let's write the SOP canonical form for the sum function of a one-bit full adder using minterm numbers.

For the function $F = \Sigma(m_0, m_1, m_3, m_4, m_6)$, *let's determine the truth table, assuming that the input variables are* A, B, *and* C.

Outline of Slide Set 3

Products of sums and product-of-sums canonical form

A *product-of-sums*—often abbreviated as POS—is defined to be either

- a single sum, or
- two or more sums, ANDed together.

With variables A, B and C, which of the following are POS expressions?

- $A + \bar{B}$
- $(A + \bar{B})(A + C)$
- $A(B + \bar{C})$
- $A\bar{B}C$

An expression in *product-of-sums canonical form* is defined to be a POS expression in which **all the sums are maxterms.**

From truth table to SOP or POS canonical form

The table below summarizes

- a method to find SOP canonical form from a truth table—this is review of material on slide 22;
- a method to find POS canonical form from a truth table.

Goal	**Step 1**	**Step 2**
SOP canonical form	For truth table rows with $F = 1$, find **true** minterms.	OR together minterms found in Step 1.
POS canonical form	For truth table rows with $F = 0$, find **false** maxterms.	AND together maxterms found in Step 1.

From truth table to POS canonical form

Goal	**Step 1**	**Step 2**
POS canonical form	For truth table rows with $F = 0$, find **false** maxterms.	AND together maxterms found in Step 1.

Let's apply this method to find POS canonical form for $F = \overline{A \oplus B}$.

Let's apply this method to find POS canonical form for the sum function of a 1-bit full adder.

Remark about methods for finding SOP and POS canonical forms

There is some **simple intuition** behind the method for **SOP canonical form**: Really, you're just making a list of all of the different combinations of input bits that make the output function true.

Your instructor's opinion about the method for **POS canonical form**: Maybe there's an intuitive way to understand it, but if so, it's **not very simple!**

Maxterm numbering and shorthand for POS canonical forms

The number of a maxterm is the row number of the truth table row for which that maxterm is **false**.

Let's find maxterm numbers for a function of 3 variables, then use those to write POS canonical form for the sum-output-of-1-bit-full-adder example.

M_k is read as "maxterm k". **Attention:** Use lowercase m for minterm, and uppercase M for maxterm.

Π notation (Π is the uppercase of π) is sometimes used to represent a product of maxterms. *Let's do an example.*

Outline of Slide Set 3

Axioms of Boolean algebra

The *axioms* of a mathematical system are a minimal set of basic definitions we assume to be true. Everything true about the system can be derived from its axioms.

For Boolean algebra, the axioms are:

(A1)	If $B \neq 1$ then $B = 0$.	(A1′)	If $B \neq 0$ then $B = 1$.
(A2)	$\overline{0} = 1$	(A2′)	$\overline{1} = 0$
(A3)	$0 \cdot 0 = 0$	(A3′)	$1 + 1 = 1$
(A4)	$1 \cdot 1 = 1$	(A4′)	$0 + 0 = 0$
(A5)	$0 \cdot 1 = 1 \cdot 0 = 0$	(A5′)	$1 + 0 = 0 + 1 = 1$

Question: Does the above table convey any new information, information that we have not already seen in ENEL 353?

Theorems of Boolean algebra

Theorems of Boolean algebra are relations between expressions in Boolean algebra; the theorems can be derived from the axioms.

Theorems can be used to transform an expression into a different-looking but equivalent expression.

Often (but not always) the goal of transformation is to **simplify** an expression for a Boolean function—in some sense to go from a complicated description of the function to a simpler one.

(Note: Theorems of Boolean algebra have practical value, but they aren't as deep and interesting, as, say, the important theorems in differential and integral calculus!)

Theorems of One Variable

These are all fairly obvious consequences of how the AND, OR, and NOT operations are defined ...

Theorem	Dual Theorem	Name
(T1) $B \cdot 1 = B$	(T1′) $B + 0 = B$	Identity
(T2) $B \cdot 0 = 0$	(T2′) $B + 1 = 1$	Null Element
(T3) $B \cdot B = B$	(T3′) $B + B = B$	Idempotency
(T4) $\overline{\overline{B}} = B$		Involution
(T5) $B \cdot \overline{B} = 0$	(T5′) $B + \overline{B} = 1$	Complements

Every one of these nine theorems is sometimes handy in algebraic manipulation of Boolean expressions.

Dual Axioms and Theorems

Every axiom or theorem in which 0, 1, AND or OR is present has a *dual*—an axiom or theorem obtained by making **all** for these substitutions: 1 for 0; 0 for 1; OR for AND; AND for OR.

You can see this principle at work in the table of theorems of one variable . . .

Theorem	**Dual Theorem**	**Name**
(T1) $B \cdot 1 = B$	(T1′) $B + 0 = B$	Identity
(T2) $B \cdot 0 = 0$	(T2′) $B + 1 = 1$	Null Element
(T3) $B \cdot B = B$	(T3′) $B + B = B$	Idempotency
(T4) $\overline{\overline{B}} = B$		Involution
(T5) $B \cdot \overline{B} = 0$	(T5′) $B + \overline{B} = 1$	Complements

Some theorems of several variables

The textbook assigns numbers to several-variable theorems from T6/T6′ to T12/T12′. Here is a partial table:

Theorem and Dual Theorem		Name
T6	$B \cdot C = C \cdot B$	Commutativity
T6′	$B + C = C + B$	
T7	$(B \cdot C) \cdot D = B \cdot (C \cdot D)$	Associativity
T7′	$(B + C) + D = B + (C + D)$	
T8	$(B \cdot C) + (B \cdot D) = B \cdot (C + D)$	Distributivity
T8′	$(B + C) \cdot (B + D) = B + (C \cdot D)$	

T6 and T6′ are obvious from the truth tables for AND and OR. T7 and T7′ are the associativity properties of AND and OR mentioned previously. (We actually proved T7 by "perfect induction".)

Distributivity properties

Let's review the table from the previous slide ...

Theorem and Dual Theorem		**Name**
T6	$B \cdot C = C \cdot B$	Commutativity
T6′	$B + C = C + B$	
T7	$(B \cdot C) \cdot D = B \cdot (C \cdot D)$	Associativity
T7′	$(B + C) + D = B + (C + D)$	
T8	$(B \cdot C) + (B \cdot D) = B \cdot (C + D)$	Distributivity
T8′	$(B + C) \cdot (B + D) = B + (C \cdot D)$	

Let's look at Theorems T8 and T8′, and compare them to similar-looking equations in ordinary algebra.

More several-variable theorems

Theorem and Dual Theorem		Name
T9	$B \cdot (B + C) = B$	Covering
T9′	$B + (B \cdot C) = B$	
T10	$(B \cdot C) + (B \cdot \overline{C}) = B$	Combining
T10′	$(B + C) \cdot (B + \overline{C}) = B$	

Let's prove T10 two ways:

- *by perfect induction;*
- *by deriving the equation using other theorems.*

T11 and T11′: Consensus Theorems

T11:

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

T11′, the **dual** of T11:

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

These allow dropping a redundant product from an SOP expression (T11) or redundant sum from an POS expression (T11′). Opportunities to use them are not easy to spot when doing algebraic manipulation, but it turns out that T11 justifies simplifications done using **Karnaugh maps**, as we'll see a little later in the course.

Do students need to memorize all the numbers and names for theorems?

For the most part, the answer is **No**. You will be tested on algebraic manipulation, but that is much more a matter of finding allowable steps in manipulations than it is a matter of knowing names of theorems.

Example: You don't need to know the names and numbers for theorems that allow you to do this ...

$$A = A \cdot 1 = A(B + \overline{B}) = AB + A\overline{B}$$

... but you must know that it is a valid and often useful step.

A big exception: It is good to know exactly what is meant by **De Morgan's Theorem**!

Outline of Slide Set 3

T12 and T12′: De Morgan's Theorem

Here are T12 and its dual, for N input variables:

$$\text{T12: } \overline{B_1 \cdot B_2 \cdot \ldots \cdot B_N} = \overline{B_1} + \overline{B_2} + \ldots + \overline{B_N}$$

$$\text{T12}': \overline{B_1 + B_2 + \ldots + B_N} = \overline{B_1} \cdot \overline{B_2} \cdot \ldots \cdot \overline{B_N}$$

For the cases of $N = 2$, $N = 3$, and $N = 4$, let's write out De Morgan's Theorem, using input variables A, B, C and D, as necessary.

Mistakes you must avoid!

If you are doing algebraic manipulation just a little faster than you are thinking, you might do things like

$$\overline{AB} = \bar{A}\bar{B} \quad \text{(Incorrect!)}$$

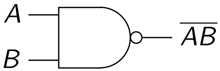
or

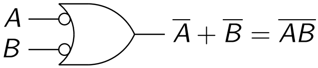
$$\overline{A + \overline{B} + C} = \overline{A} + B + \overline{C} \quad \text{(Also incorrect!)}$$

What are correct transformations for $\overline{AB}$ *and* $\overline{A + \overline{B} + C}$*?*

De Morgan's Theorem, NAND gates and NOR gates

A NAND gate performs a "NOT-of-the-AND" function. De Morgan's Theorem says that we can also view that as an "OR-of-the-NOTs" function. So the following two symbols both describe the behaviour of a single circuit element . . .





Let's write a similar description of a 3-input NOR gate and draw two different symbols for the gate.

De Morgan's Theorem and POS Canonical Form

Review . . .

Goal	**Step 1**	**Step 2**
SOP canonical form	For truth table rows with $F = 1$, find **true** minterms.	OR together minterms found in Step 1.
POS canonical form	For truth table rows with $F = 0$, find **false** maxterms.	AND together maxterms found in Step 1.

Suppose that the method for finding SOP canonical form makes sense to you, but the method for POS canonical form is a little mysterious.

Let's demonstrate how POS canonical form for F can be found starting with SOP canonical form for $\overline{F}$.

Outline of Slide Set 3

Using theorems to simplify expressions

A practical application of Boolean algebra theorems is to find an economical expression for a logic function.

Usually, using fewer gates with smaller numbers of inputs per gate leads to circuits that

- use less chip area;
- consume less power;
- switch output faster when inputs change from 0 to 1 or 1 to 0.

Example: Canonical SOP for the C_{OUT} function in a 1-bit full adder is $\overline{A}BC_{\text{IN}} + A\overline{B}C_{\text{IN}} + AB\overline{C_{\text{IN}}} + ABC_{\text{IN}}$. *Let's use algebraic manipulation to find a simpler SOP expression for* $C_{OUT}(A,B,C_{IN})$.

Using theorems to simplify expressions often works well, but it raises a few questions, such as . . .

- What is a good choice of theorem to use to make my expression simpler?
- What is a bad choice, to be avoided, because it would make my expression more complicated, rather than simpler?
- It's so hard to avoid making mistakes that result in incorrect expressions! How the **** do I make sure things don't go wrong?!?!
- How do I know when I'm done—how do I know there are no more simplifications to be made?

Unfortunately, the questions on the previous slide don't always have simple answers!

We'll see later in the course that **Karnaugh maps** help with answers to these questions, especially in the cases of 3–5 input variables.