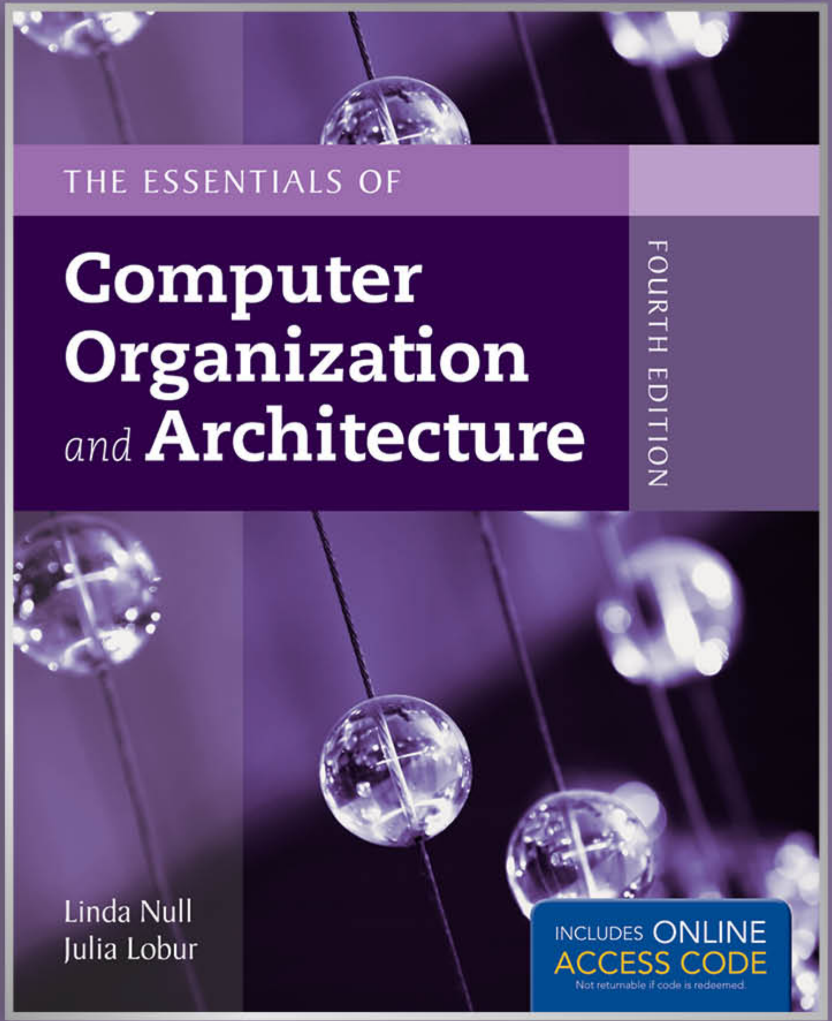
Chapter 3

Boolean Algebra and Digital Logic



Chapter 3 Objectives

- Understand the **relationship** between **Boolean algebra** and **digital circuits**.
- Learn how to **design** simple **circuits**.
- Understand how **digital circuits** work together to **form complex computer systems**.

3.1 Background

- In the latter part of the nineteenth century, **George Boole** incensed philosophers and mathematicians alike when he **suggested that logical thought could be represented through mathematical equations**.
- **Computers**, as we know them today, are **implementations of Boole's *Laws*** *of Thought*.

3.2 Boolean Algebra

- Boolean algebra is a **mathematical system** for the **manipulation of variables** that can have one of two values.
  - In **formal logic**, these values are "**true**" and "**false**"
  - In **digital systems**, these values are "on" and "off," **1** and **0**, or "high" and "low."
- **Boolean expressions (functions)** are created by performing **operations** on Boolean variables.
  - Common Boolean operators include **AND**, **OR**, and **NOT**.

3.2 Boolean Algebra

- **Boolean operators** can be completely described using a **truth table**.
- The truth table for the Boolean operators **AND** and **OR** are shown at the right.
- The **AND** operator is also known as a **Boolean product**. The **OR** operator is the **Boolean sum**.

X AND Y

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

X OR Y

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

3.2 Boolean Algebra

- The truth table for the Boolean **NOT** operator is shown at the right.
- The NOT operation is most often designated by a **prime** (x'). It is sometimes indicated by an **overbar** ($\bar{x}$) or an "**elbow**" ($\neg x$).

NOT X

X	X'
0	1
1	0

3.2 Boolean Algebra

- A **Boolean function** consists of:
  - **Boolean variables**,
  - **Boolean operators**, and
  - **Inputs** from the set {0,1}.
- It produces an **output** that is also a member of the set {0,1}.

3.2 Boolean Algebra

- The truth table for the Boolean function:

  $F(x,y,z) = xz' + y$

  is shown at the right.
- To make evaluation of the Boolean function easier, the truth table contains **extra columns** to hold evaluations of **subparts** of the function.

$F(x,y,z) = xz' + y$

x	y	z	z'	xz'	xz' + y
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

3.2 Boolean Algebra

- As with common arithmetic, Boolean operations have **precedence rules**.
- The **NOT** operator has **highest priority**, **followed by AND** and **then OR**.
- This is how we chose the function subparts in our table.

$$F(x,y,z) = xz' + y$$

x	y	z	z'	xz'	xz'+ y
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

3.2 Boolean Algebra

- Digital computers contain **circuits** that **implement Boolean functions**.
- **The simpler** that we can make a Boolean **function, the smaller the circuit** that will result.
  - **Simpler circuits are cheaper to build, consume less power, and run faster** than complex circuits.
- With this in mind, we always want to **reduce our Boolean functions to their simplest form.**
- There are a number of **Boolean identities** that help us to do this.

3.2 Boolean Algebra

- Most Boolean identities have an AND (product) form as well as an OR (sum) form. We give our identities using both forms. Our first group is rather intuitive:

Identity Name	AND Form	OR Form
Identity Law	$1x = x$	$0 + x = x$
Null Law	$0x = 0$	$1 + x = 1$
Idempotent Law	$xx = x$	$x + x = x$
Inverse Law	$xx' = 0$	$x + x' = 1$

3.2 Boolean Algebra

- Our second group of Boolean identities should be familiar to you from your study of algebra:

Identity Name	AND Form	OR Form
Commutative Law	$xy = yx$	$x+y = y+x$
Associative Law	$(xy)z = x(yz)$	$(x+y)+z = x + (y+z)$
Distributive Law	$x+yz = (x+y)(x+z)$	$x(y+z) = xy+xz$

3.2 Boolean Algebra

- Our last group of Boolean identities are perhaps the most useful.
- If you have studied set theory or formal logic, these laws are also familiar to you.

Identity Name	AND Form	OR Form
Absorption Law	$x(x+y) = x$	$x + xy = x$
DeMorgan's Law	$(xy)' = x' + y'$	$(x+y)' = x'y'$
Double Complement Law	$(x)'' = x$	

3.2 Boolean Algebra

- DeMorgan’s law provides an easy way of finding the complement of a Boolean function.
- DeMorgan’s law can be extended to any number of variables.
- Replace each variable by its complement and change all ANDs to ORs and all ORs to ANDs.
- Thus, we find the the complement of:

$$F(x,y,z) = (xy)+(x'y)+(xz')$$

is:

3.2 Boolean Algebra

- DeMorgan's law provides an easy way of finding the complement of a Boolean function.
- DeMorgan's law can be extended to any number of variables.
- Replace each variable by its complement and change all ANDs to ORs and all ORs to ANDs.
- Thus, we find the the complement of:

$$F(x,y,z) = (xy)+(x'y)+(xz')$$

is:

$$\begin{aligned} F'(x,y,z) &= ((xy)+(x'y)+(xz'))' \\ &= (xy)'(x'y)'(xz')' \\ &= (x'+y')(x+y')(x'+z) \end{aligned}$$

3.2 Boolean Algebra

- Through our exercises in simplifying Boolean expressions, we see that **there are numerous ways of stating the same Boolean expression.**
  - These “synonymous” forms are ***logically equivalent***.
  - Logically **equivalent expressions have identical truth tables**.
- In order to eliminate as much confusion as possible, **designers express Boolean functions in *standardized* or *canonical* form**.

3.2 Boolean Algebra

- There are two canonical forms for Boolean expressions: **sum-of-products** and **product-of-sums**.
  - Recall the Boolean product is the AND operation and the Boolean sum is the OR operation.
- In the **sum-of-products** form, AND-ed variables are OR-ed together.
  - For example: $F(x,y,z) = xy + xz + yz$
- In the **product-of-sums form**, OR-ed variables are AND-ed together:
  - For example: $F(x,y,z) = (x+y)(x+z)(y+z)$

3.2 Boolean Algebra

- It is easy to **convert** a function **to sum-of-products** form using its **truth table**.
- We are interested in the values of the variables that make the function **true** (=1).
- Using the truth table, we **list the values of the variables that result in a true function value**.
- **Each group** of variables is then **OR-ed together**.

$$F(x,y,z) = xz' + y$$

x	y	z	xz'+ y
0	0	0	0
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	1

3.2 Boolean Algebra

- The sum-of-products form for our function is:

$$F(x,y,z) = xz' + y$$

x	y	z	xz'+ y
0	0	0	0
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	1

$$F(x,y,z) = (x'yz') + (x'yz) + (xy'z') + (xyz') + (xyz)$$

We note that this function is not in simplest terms. Our aim is only to rewrite our function in canonical sum-of-products form.

Exercise

- Provide the truth table of a function F that outputs the **majority** of three inputs x, y and z
- i.e., if at least two inputs are 0 then the output is 0, and if at least two inputs are 1 then the output is 1
- Convert the function into a **sum-of-products** and a **product-of sums**
- **Simplify** the function