# 【CA】Day1

## 【Ch1】Boolean Logic

### 1.1 Boolean Algebra

#### 1.1.1 Boolean Algebra

Boolean algebra deals with Boolean(also called binary) values that are typically labelled true/false, 1/0, yes/no.

A Boolean function is a function that operates on binary inputs and returns binary outputs.

Truth Table Representation: The simplest way to specify a Boolean function is to enumerate all the possible values of the function's input variables, along with the function's output for each set of inputs.

| $x$ | $y$ | $z$ | $f(x, y, z)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

**Figure 1.1** Truth table representation of a Boolean function (example).

Boolean Expressions: In addition to the truth table specification, a Boolean function can also be specified using Boolean operations over its input variables.

The basic Boolean operators that are typically used are

- And
- Or
- Not

We will use a common arithmetic-like notation for those operations:

- $x \cdot y$ means x And y
- $x + y$ means x Or y
- $\bar{x}$ means Not x

To illustrate the function defined in the figure above, the equivalent function given by the Boolean expression is $f(x, y, z) = (x + y) \cdot \bar{z}$

Boolean expression called the canonical representation. Starting with the function's truth table, we focus on all the rows in which the function has value 1. For each such row, we construct a term created by Anding together literals that fix the values of all the row's inputs.

$f(x, y, z) = \bar{x}y\bar{z} + x\bar{y}\bar{z} + xy\bar{z}$

This construction leads to an important conclusion: Every Boolean function, no matter how complex, can be expressed using three Boolean operators only: And, Or, and Not.

Two-Input Boolean Functions: An inspection of the figure above reveals that the number of Boolean functions that can be defined over n binary variables is $2^{2^n}$.

The sixteen Boolean functions spanned by two variables are listed below.

| Function | $x$ | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|
| | $y$ | 0 | 1 | 0 | 1 |
| Constant 0 | 0 | 0 | 0 | 0 | 0 |
| And | $x \cdot y$ | 0 | 0 | 0 | 1 |
| $x$ And Not $y$ | $x \cdot \bar{y}$ | 0 | 0 | 1 | 0 |
| $x$ | $x$ | 0 | 0 | 1 | 1 |
| Not $x$ And $y$ | $\bar{x} \cdot y$ | 0 | 1 | 0 | 0 |
| $y$ | $y$ | 0 | 1 | 0 | 1 |
| Xor | $x \cdot \bar{y} + \bar{x} \cdot y$ | 0 | 1 | 1 | 0 |
| Or | $x + y$ | 0 | 1 | 1 | 1 |
| Nor | $\overline{x + y}$ | 1 | 0 | 0 | 0 |
| Equivalence | $x \cdot y + \bar{x} \cdot \bar{y}$ | 1 | 0 | 0 | 1 |
| Not $y$ | $\bar{y}$ | 1 | 0 | 1 | 0 |
| If $y$ then $x$ | $x + \bar{y}$ | 1 | 0 | 1 | 1 |
| Not $x$ | $\bar{x}$ | 1 | 1 | 0 | 0 |
| If $x$ then $y$ | $\bar{x} + y$ | 1 | 1 | 0 | 1 |
| Nand | $\overline{x \cdot y}$ | 1 | 1 | 1 | 0 |
| Constant 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 1.2** All the Boolean functions of two variables.

### 1.1.2 Gate Logic

A gate is a physical device that implements a Boolean function.

If a Boolean function f operates on n variables and returns m binary results, the gate that implements f will have n input pins and m output pins.

The simplest gates of all are made from tiny switching devices, called transistors.

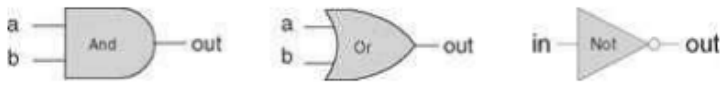


**Figure 1.3** Standard symbolic notation of some elementary logic gates.

Primitive and Composite Gates: Since all logic gates have the same input and output semantics(0's and 1's), they can be chained together, creating composite gates of arbitrary complexity.

For example, if we want to construct a 3-way Boolean function And(a, b, c), we can build the logic gates as following:

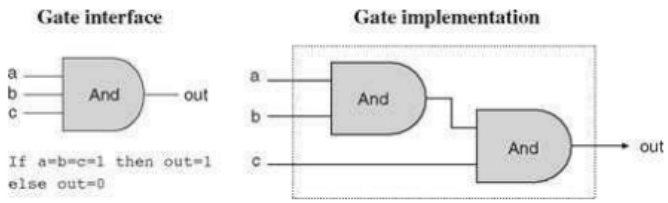


**Figure 1.4** Composite implementation of a three-way And gate. The rectangle on the right defines the

Now we want to design Xor(a, b) = Or(And(a, Not(b)), And(Not(a), b))

The following figure shows the logic design of Xor(a, b)

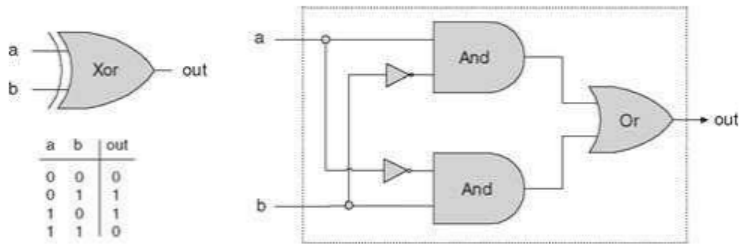


**Figure 1.5** Xor gate, along with a possible implementation.