Ollscoil na Gaillimhe
University of Galway

CT101 Computing Systems

Dr. Bharathi Raja Chakravarthi
Lecturer-above-the-bar
Email: bharathi.raja@universityofgalway.ie

University ofGalway.ie

Ollscoil na Gaillimhe
University of Galway

Recap

Complements of Numbers

- Complements are used in digital computers to **simplify the subtraction operation and for logical manipulation**.

- **Two types** of complements for each base-r system:
  - the radix complements (**r's complement**)
  - the diminished radix complements (**(r - 1)'s complement**)

- Value of base r is substituted in the name, then
  - 2's complement and 1's complement
  - 10's complement and 9's complement

Diminished Radix Complement

- **(r - 1)'s** complement of N is **$(r^n - 1) - N$**

Where,
N - number
r - base
n - digits

- **For decimal numbers**,
  r = 10 and **r - 1 = 9**, 9's complement of N is

$$(10^n - 1) - N$$

- **For binary numbers**,
  r = 2 and **r - 1 = 1**, so the 1's complement of N is

$$(2^n - 1) - N$$

Radix Complement

- r's complement of N is

  r^n - N for $N \neq 0$ and **0** for $N = 0$.

  Where,
  N - number
  r - base
  n - digits

- Radix complement is obtained **by adding 1** to the Diminished Radix Complement

$$\mathbf{r^n - N = [(r^n - 1) - N] + 1}$$

Efficient 2's Complement

- **First 1 from right**

01101011100011100000

- **Complement leftmost digits**

<u>**10010100011100**</u>100000

0110100111100

replaced

<u>**1001011000**</u>100

1000000000000

unchanged

1000000000000

Subtraction with Complements

Subtracting two n-digit unsigned numbers, **M-N** in base **r**:

1. Add the **M** to the **r's** complement of **N**.

$$M + (r^n - N) = (M - N) + r^n$$

2. If **M ≥ N**, the sum will produce an end carry, i.e., r^n, which can be discarded to produce M - N

3. If **M < N**, the sum does not produce an end carry. Apply r's complement on the sum & place a –ve sign in front.

$$\left.\begin{array}{c} r^n - (N - M) \\ \text{or} \\ -(r^n + (M-N)) \end{array}\right\} \text{r's complement of } \mathbf{(N - M)}$$

Signed Binary Numbers

- Positive numbers and zero can be represented by unsigned n-digit, radix r numbers.
- We need a representation for negative numbers.
- To represent a sign (+ or -) we need exactly one more bit of information (1 binary digit gives $2^1 = 2$ elements which is exactly what is needed).
- The most significant bit (MSB) is interpreted as a sign bit as shown below:

$$\mathbf{s a_{n-2} \ldots a_2 a_1 a_0}$$

Where:
s = 0 for Positive numbers
s = 1 for Negative numbers
ai are 0 or 1

Binary-Coded Decimal Code

- It is commonly known as **BCD**.

- BCD code is a weighted code, so in this code each digit is assigned a specific Weight according to its position.

- BCD code is also known as 8421 code.

- This is because 8,4,2, and 1 are the weights of the four bits of the BCD code.

Binary-Coded Decimal Code

- To represent 10 decimal digits, it is necessary to use atleast 4 binary bits.
- For each decimal digits (0 to 9) is represented by unique combination of bits
- So, there will be six unused or invalid combination (10 to 15) in BCD code.

Decimal Symbol	BCD Digit
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

BCD Example

$(12)_{10} = (?)_2$

$(12)_{10}$

0001 0011

$(12)_{10} =$ $(00010010)_{BCD}$

$(1100111)_{BCD} = (?)_{10}$

$\underline{0001}\ \underline{0110}$

6 7

$(0110010110)_{BCD} =$ $(67)_{10}$

Binary Storage
Logic Gates
Boolean Algebra

Binary Storage and Registers

Binary Storage:

- When discrete elements of information are represented in binary form, the information storage medium must contain binary storage elements for storing individual bits.
  - ❖ **Binary Cells:** A device that possesses two stable states.
  - ❖ **Cell Input:** Receives data and control signals that set it into one of the states.
  - ❖ **Cell Output:** Physical quantity indicating which state the cell is in.
  - ❖ States are encoded as binary digits[0,1].

Binary Storage and Registers

Registers:

- Flip-flop is a 1-bit memory cell that can be used for storing digital data.
- To increase the storage capacity in terms of the number of bits, we have to use a group of flip flops. Such a group of flip-flops is known as a **register**.
- The n-bit register will consist of n number of flip-flops, and it can store an n-bit word.
- A register with n cell can be in one of the 2n states.
- The register state (or content) can be interpreted as value, ASCII, etc.

Register Transfer

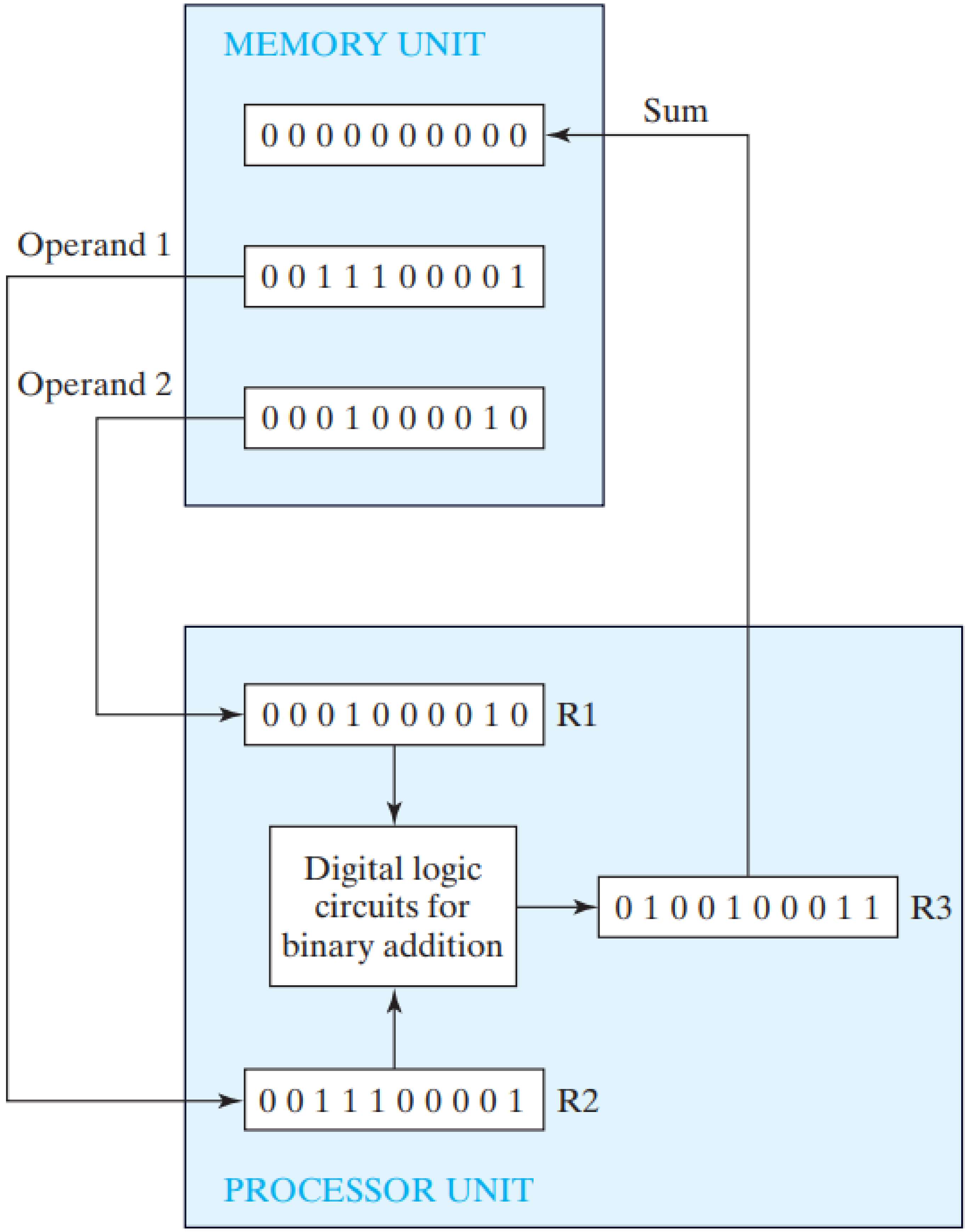
- It is a very general way to describe a digital circuit (including a computer).
- Registers are interconnected to each other.
- At a given time content of one register is transferred to another.
- The transforming circuit is a data processing or data path element/circuit.

Transfer of Information

- Circuit elements to manipulate individual bits of information

- Load-store machine

```
LD    R1;
LD    R2;
ADD   R3, R2, R1;
SD    R3;
```



Binary Logic

- We use binary logic to have an abstract representation of logic gates.
- A digital logic circuit has a no. of input lines A, B, C, ..., and a no. of output lines. We will consider one output line called Z at the moment.
- We write Z = f(A, B, C,...) to mean that the value of Z is determined by the values of the inputs A, B, C... Z is said to be a function of its inputs.
- The two values of binary logic can be called by different names (yes or no, true or false).
- In our case, we can assign the values 1 and 0.

Binary Logic

There are three basic logical operations: AND, OR, and NOT. Each operation produces a binary result, denoted by z.

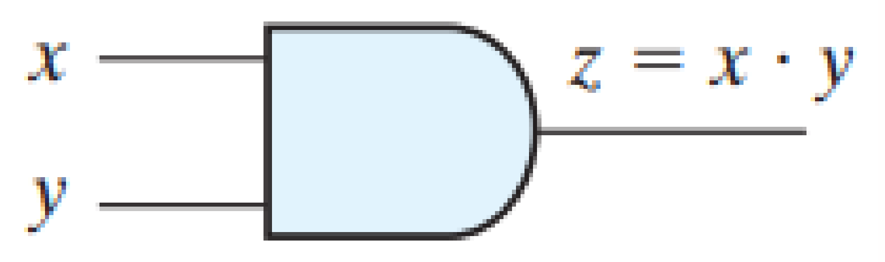
- AND – represented by a dot or absence of an operator. **E.g.**, $x \cdot y = z$ or $xy = z$
- OR – represented by a plus sign. **E.g.,** $x + y = z$
- NOT – represented by a prime (sometimes by an overbar). **E.g.**, $x' = z$ or $\bar{x} = z$

Logic Gates

A logic gate is a simple switching circuit that determines whether an input pulse can pass through to the output in digital circuits.

(a) **AND Gate:**

Two-input AND gate

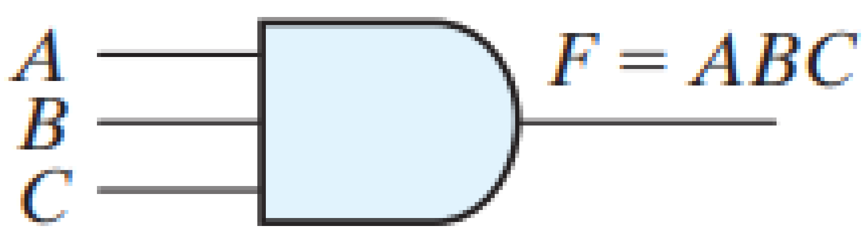


Truth Table

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

Logic Gates

Three-input AND gate



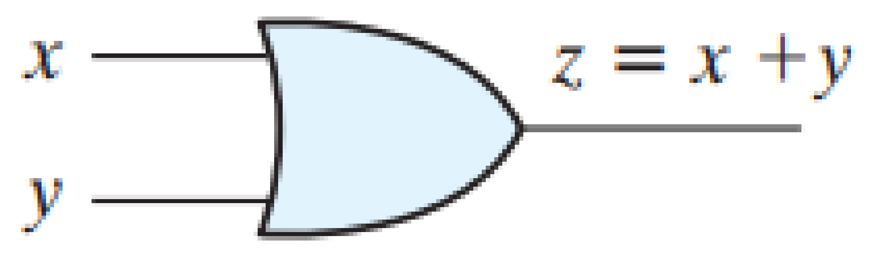
Truth Table

A	B	C	F
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

Logic Gates

(b) **OR Gate:**

Two-input OR gate

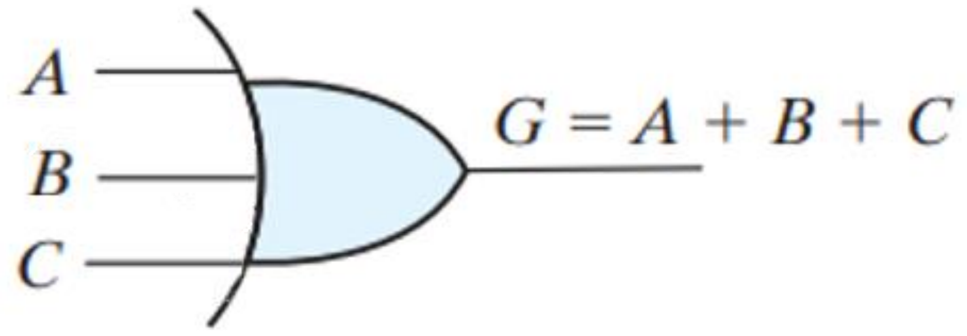


Truth Table

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

Logic Gates

Four-input OR gate



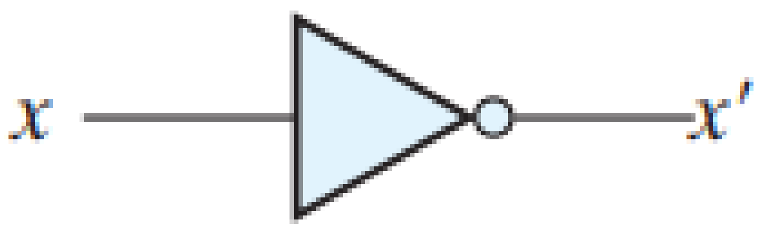
Truth Table

A	B	C	G
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

Logic Gates

(c) **NOT Gate:**

NOT gate or inverter



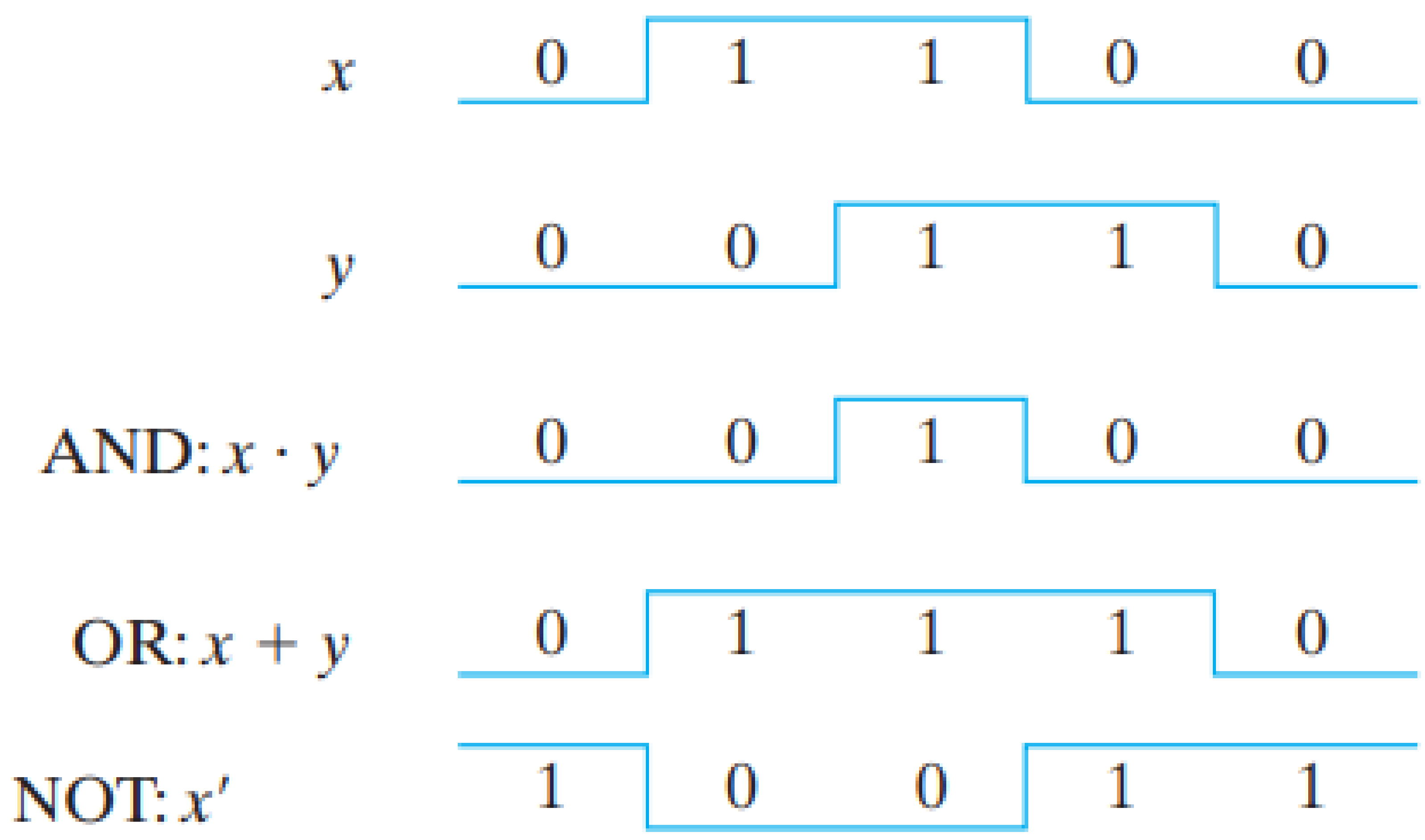
Truth Table

NOT

x	x'
0	1
1	0

Logic Gates

Input – Output signals for gates



Boolean Algebra and Logic Gates

- Finding simpler and cheaper, but equivalent, realizations of a circuit can reduce the overall cost of the design.
- Mathematical methods that simplify circuits rely primarily on Boolean algebra.
- Boolean algebra enable you to optimize simple circuits.
- It enable you to understand the purpose of algorithms used by software tools to optimize complex circuits involving millions of logic gates.

Basic Definitions

- Boolean algebra defined with a set of elements, a set of operators, and a number of unproved axioms or postulates.

- A set of elements is any collection of objects, usually having a common property.

  If **S** → Set

  x,y → objects

  $\mathbf{x} \in \mathbf{S}$ → x is an element of S

  $\mathbf{y} \in \mathbf{S}$ → y is not an element of S

- A = {1, 2, 3, 4} → the elements of set A are the numbers 1, 2, 3, and 4.

Basic Definitions

A **binary operator** defined on a set S of elements is a rule that assigns, to each pair of elements from S, a unique element from S.

E.g., $a \boxed{*} b = c$ ⟶ a rule for **finding c from the pair (a, b) if a, b, c ∈ S**

↓

binary operator

Note: * is not a binary operator **if a, b ∈ S, and if c ∉ S.**

Boolean Algebra - Postulates

- ❖ Postulates of a mathematical system form the basic assumptions to deduce the rules, theorems, and properties of the system.
- ❖ The most common postulates used to formulate various algebraic structures are as follows:

Postulate 1: Closure

- A set S is closed with respect to a binary operator if, for every pair of elements of S, the binary operator specifies a rule for obtaining a unique element of S.

- **For example**, the set of natural numbers **N = {1, 2, 3, 4, …}** is closed with respect to the binary operator + by the rules of arithmetic addition, since, for any **a, b** ∈ **N**, there is a unique **c** ∈ **N** such that **a + b = c**.

Boolean Algebra - Postulates

Postulate 2: Associative law

- A binary operator * on a set S is said to be associative whenever

$$(x * y) * z = x * (y * z) \text{ for all } x, y, z, \in S$$

Postulate 3: Commutative law

- A binary operator * on a set S is said to be commutative whenever

$$x * y = y * x \text{ for all } x, y \in S$$

Boolean Algebra - Postulates

Postulate 4: Identity element

- A binary operation * on S if there exists an element $e \in S$ with the property that

  $$\mathbf{e * x = x * e = x} \text{ for every } x \in S$$

- E.g., $x + 0 = 0 + x = x$ for any $x \in I$ where $I = \{ c, -3, -2, -1, 0, 1, 2, 3, c\}$,

Postulate 5: Inverse

- a binary operator * is said to have an inverse whenever, for every $x \in S$, there exists an element $y \in S$ such that

  $$\mathbf{x * y = e}$$

- E.g., $x + x' = 1$ and $x \cdot x' = 0$

Boolean Algebra - Postulates

Postulate 6: Distributive law

- If * and · are two binary operators on a set S, * is said to be distributive over . whenever

$$x * (y \cdot z) = (x * y) \cdot (x * z)$$

- A field is a set of elements, together with two binary operators, each having properties 1 through 5 and both operators combining to give property 6.
- The field of real numbers is the basis for arithmetic and ordinary algebra.

Boolean Algebra - Postulates

- The operators and postulates have the following meanings:
  - The binary operator **+ defines addition**.
  - The additive identity is 0.
  - The additive inverse defines subtraction.

  - The binary operator **· defines multiplication**.
  - The multiplicative identity is 1.
  - For $a \neq 0$, the multiplicative inverse of $a = 1/a$ defines division (i.e., $a \cdot 1/a = 1$).
  - The only distributive law applicable is that of · over +:

$$\mathbf{a \cdot (b + c) = (a \cdot b) + (a \cdot c)}$$

Huntington postulates

1. (a) The structure is closed with respect to the operator +.
   (b) The structure is closed with respect to the operator $\cdot$.

2. (a) The element 0 is an identity element with respect to +; that is, $\mathbf{x + 0 = 0 + x = x}$.
   (b) The element 1 is an identity element with respect to $\cdot$; that is, $\mathbf{x \cdot 1 = 1 \cdot x = x}$.

3. (a) The structure is commutative with respect to +; that is, $\mathbf{x + y = y + x}$.
   (b) The structure is commutative with respect to $\cdot$; that is, $\mathbf{x \cdot y = y \cdot x}$.

Huntington postulates

4. (a) The operator $\cdot$ is distributive over $+$; that is, $\mathbf{x \cdot (y + z) = (x \cdot y) + (x \cdot z)}$.
   (b) The operator $+$ is distributive over $\cdot$; that is, $\mathbf{x + (y \cdot z) = (x + y) \cdot (x + z)}$.

5. For every element $x \in B$, there exists an element $x' \in B$ (called the complement of x)
   (a) $\mathbf{x + x' = 1}$
   (b) $\mathbf{x \cdot x' = 0}$.

6. There exist at least two elements $\mathbf{x, y \in B}$ such that $\mathbf{x \neq y}$.

Two-Valued Boolean Algebra

- **Two-valued Boolean algebra** is defined on a set of two elements, **B = {0, 1}** with rules for the two binary operators **+** and **·** as shown in the following operator tables.

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

x	x'
0	1
1	0

- These rules are exactly the same as the AND, OR, and NOT operations

Two-Valued Boolean Algebra

Huntington postulates are valid for the set B = {0, 1} and the two binary operators + and ·

1. The structure is closed with respect to the two operators, since the result of each operation is either **1 or 0** and **1, 0 ∈ B**.

2. Establishes the two identity elements, 0 for + and 1 for ·
   (a) $\mathbf{0 + 0 = 0}$ $\quad\quad$ $\mathbf{0 + 1 = 1 + 0 = 1}$;
   (b) $\mathbf{1 \cdot 1 = 1}$ $\quad\quad$ $\mathbf{1 \cdot 0 = 0 \cdot 1 = 0}$.

Two-Valued Boolean Algebra

3. Commutative laws are obvious from the symmetry of the binary operator tables

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

x	x'
0	1
1	0

Two-Valued Boolean Algebra

4. **Distributive law:** For each combination, we derive **$x \cdot (y + z)$** and show that the value is the same as the value of **$(x \cdot y) + (x \cdot z)$**:

Consider

x	y	z
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Two-Valued Boolean Algebra

$x \cdot (y + z)$

x	y	z	$y + z$	$x \cdot (y + z)$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Two-Valued Boolean Algebra

$(x \cdot y) + (x \cdot z)$:

x	y	z	$x \cdot y$	$x \cdot z$	$(x \cdot y) + (x \cdot z)$
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

Shows that the value of **$x \cdot (y + z)$** is same as the value of **$(x \cdot y) + (x \cdot z)$**

Two-Valued Boolean Algebra

5. From the complement table, it is easily shown that

    (a) $x + x' = 1$, since $0 + 0' = 0 + 1 = 1$ and $1 + 1' = 1 + 0 = 1$.

    (b) $x \cdot x' = 0$, since $0 \cdot 0' = 0 \cdot 1 = 0$ and $1 \cdot 1' = 1 \cdot 0 = 0$.

Thus, postulate 5 is satisfied and postulate 1 is verified.

6. Postulate 6 is satisfied because the two-valued Boolean algebra has two elements, **1 and 0, with $1 \neq 0$**.

Two-Valued Boolean Algebra

- The two-valued Boolean algebra defined in this section is also called "**switching algebra**" by engineers.

- To emphasize the similarities between two-valued Boolean algebra and other binary systems, that algebra was called "**binary logic**".

References

- Computer Organization and Architecture Designing for Performance Tenth Edition by William Stallings
- Digital Design With an Introduction to the Verilog HDL FIFTH EDITION by M Morris, M. and Michael, D., 2013.

Ollscoil na Gaillimhe
University of Galway

Thank *you*