Digital Arithmetic

Computer Systems 2016

Oleks

DIKU

September 14, 2016

0

1

Truth Tables

NOT

A	$\overline{A}$
0	1
1	0

Truth Tables

NOT

A	$\overline{A}$
0	1
1	0

AND

A	B	$A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

OR

A	B	$A + B$
0	0	0
0	1	1
1	0	1
1	1	1

An **algebra** is a collection of functions adhering to **laws**.

The functions **NOT**, **AND**, and **OR** form a **boolean algebra**:

$$\begin{aligned} \textbf{NOT} &: \mathbb{B}^1 \to \mathbb{B} \\ \textbf{AND} &: \mathbb{B}^2 \to \mathbb{B} \\ \textbf{OR} &: \mathbb{B}^2 \to \mathbb{B} \end{aligned}$$

Boolean Algebra Laws - AND form

Name	AND Form
Identity Law	$1 \cdot A = A$
Null Law	$0 \cdot A = 0$
Idempotent Law	$A \cdot A = A$
Inverse Law	$A \cdot \overline{A} = 0$
Commutative Law	$A \cdot B = B \cdot A$
Associative Law	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$
Distributive Law	$A + B \cdot C = (A + B) \cdot (A \cdot C)$
Absorption Law	$A \cdot (A + B) = A$
De Morgan's Law	$\overline{A \cdot B} = \overline{A} + \overline{B}$

Boolean Algebra Laws - OR form

Name	OR Form
Identity Law	$0 + A = A$
Null Law	$1 + A = 1$
Idempotent Law	$A + A = A$
Inverse Law	$A + \overline{A} = 1$
Commutative Law	$A + B = B + A$
Associative Law	$(A + B) + C = A + (B + C)$
Distributive Law	$A \cdot (B + C) = A \cdot B + A \cdot C$
Absorption Law	$A + A \cdot B = A$
De Morgan's Law	$\overline{A + B} = \overline{A} \cdot \overline{B}$

Truth Tables

NOT

A	$\overline{A}$
0	1
1	0

AND

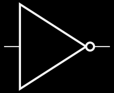
A	B	$A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

OR

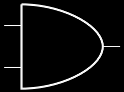
A	B	$A + B$
0	0	0
0	1	1
1	0	1
1	1	1

Gates

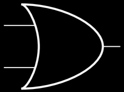
NOT



AND



OR



More Truth Tables

XOR

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

NAND

A	B	$A \uparrow B$
0	0	1
0	1	1
1	0	1
1	1	0

NOR

A	B	$A \downarrow B$
0	0	1
0	1	0
1	0	0
1	1	0

More Truth Tables

XOR

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

NAND

A	B	$A \uparrow B$
0	0	1
0	1	1
1	0	1
1	1	0

NOR

A	B	$A \downarrow B$
0	0	1
0	1	0
1	0	0
1	1	0

Do we add these as basic functions to our algebra?

More Truth Tables

XOR

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

NAND

A	B	$A \uparrow B$
0	0	1
0	1	1
1	0	1
1	1	0

NOR

A	B	$A \downarrow B$
0	0	1
0	1	0
1	0	0
1	1	0

Do we add these as basic functions to our algebra?

Do we need fancy new hardware?

Admissibility

A function is **admissible** if

it can be derived from other functions

Admissibility

A function is **admissible** if

it can be derived from other functions

$\equiv$

it does not contradict the laws of the algebra

XOR is admissible

— Proof by reading the truth table.

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

$$\begin{aligned} A \oplus B &\equiv \left(\overline{A} \cdot B\right) + \left(A \cdot \overline{B}\right) && \text{(read 1s)} \\ &\equiv \overline{\left(\overline{A} \cdot \overline{B}\right) + \left(A \cdot B\right)} && \text{(read 0s)} \end{aligned}$$

"*A* or *B*, but not both *A* and *B*"

NAND is admissible

— Proof by reading the truth table.

A	B	$A \uparrow B$
0	0	1
0	1	1
1	0	1
1	1	0

$$A \uparrow B \equiv \left(\overline{A} \cdot \overline{B}\right) + \left(A \cdot \overline{B}\right) + \left(\overline{A} \cdot B\right) \qquad \text{(read 1s)}$$

$$\equiv \overline{\left(A \cdot B\right)} \qquad \text{(read 0s)}$$

"not both A and B"

NOR is admissible

— Proof by reading the truth table.

A	B	$A \downarrow B$
0	0	1
0	1	0
1	0	0
1	1	0

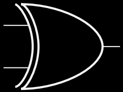
$$A \downarrow B \equiv \left(\overline{A} \cdot \overline{B}\right) \qquad \text{(read 1s)}$$

$$\equiv \overline{\left(A \cdot \overline{B}\right) + \left(\overline{A} \cdot B\right) + (A \cdot B)} \qquad \text{(read 0s)}$$

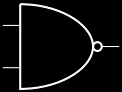
"neither *A* nor *B*"

More Gates

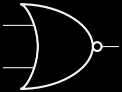
XOR



NAND



NOR



We like fancy new hardware!

{ **AND, OR, NOT** }

is not the **only** possible set of basic functions for a boolean algebra.

AND is admissible in $\{$ OR, NOT $\}$

$$A \cdot B \equiv \overline{\overline{A} + \overline{B}}$$

— Proof <u>by truth table</u>.

AND is admissible in $\{$ OR, NOT $\}$

$$A \cdot B \equiv \overline{\overline{A} + \overline{B}}$$

— Proof by truth table.

A	B	$A \cdot B$	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$	$\overline{\overline{A} + \overline{B}}$
0	0	0	1	1	1	0
0	1	0	1	0	1	0
1	0	0	0	1	1	0
1	1	1	0	0	0	1

OR is admissible in $\{\ \text{AND}, \text{NOT}\ \}$

$$A + B \equiv \overline{\overline{A} \cdot \overline{B}}$$

— Proof by truth table.

OR is admissible in $\{$ AND, NOT $\}$

$$A + B \equiv \overline{\overline{A} \cdot \overline{B}}$$

— Proof by truth table.

A	B	$A + B$	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$	$\overline{\overline{A} \cdot \overline{B}}$
0	0	0	1	1	1	0
0	1	1	1	0	0	1
1	0	1	0	1	0	1
1	1	1	0	0	0	1

Prove At Your Own Risk

NOT is admissible in { **NAND** }
AND is admissible in { **NAND** }
OR is admissible in { **NAND** }

Prove At Your Own Risk

NOT is admissible in { **NAND** }
AND is admissible in { **NAND** }
OR is admissible in { **NAND** }

NOT is admissible in { **NOR** }
AND is admissible in { **NOR** }
OR is admissible in { **NOR** }

Prove At Your Own Risk

NOT is admissible in { **NAND** }
AND is admissible in { **NAND** }
OR is admissible in { **NAND** }

NOT is admissible in { **NOR** }
AND is admissible in { **NOR** }
OR is admissible in { **NOR** }

NAND is admissible in { **NOR** }
NOR is admissible in { **NAND** }

Prove At Your Own Risk

NOT is admissible in { **NAND** }
AND is admissible in { **NAND** }
OR is admissible in { **NAND** }

NOT is admissible in { **NOR** }
AND is admissible in { **NOR** }
OR is admissible in { **NOR** }

NAND is admissible in { **NOR** }
NOR is admissible in { **NAND** }

Conclusion: We can make due with different **basic** gates.

Addition

1-bit Addition

A	B	Carry	Result
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

n-bit Addition

Given the bit-strings $x_{n-1}x_{n-2}\cdots x_0$ and $y_{n-1}y_{n-2}\cdots y_0$.

n-bit Addition

Given the bit-strings $x_{n-1}x_{n-2}\cdots x_0$ and $y_{n-1}y_{n-2}\cdots y_0$.

Show how to produce a bit-string $z_{n-1}z_{n-2}\cdots z_0$,
and a carry- (overflow) bit c_n, such that

n-bit Addition

Given the bit-strings $x_{n-1}x_{n-2}\cdots x_0$ and $y_{n-1}y_{n-2}\cdots y_0$.

Show how to produce a bit-string $z_{n-1}z_{n-2}\cdots z_0$, and a carry- (overflow) bit c_n, such that

$$\sum_{0}^{n-1} z_i 2^i + c_n = \sum_{0}^{n-1} x_i 2^i + \sum_{0}^{n-1} y_i 2^i$$

Carry-Ripple Addition

Let $c_0 = 0$, and let z_i be given by the following truth table:

x_i	y_i	c_i	c_{i+1}	z_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Carry-Ripple Addition

Let $c_0 = 0$, and let z_i be given by the following truth table:

x_i	y_i	c_i	c_{i+1}	z_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

“Full Adder”

Carry-Ripple Addition

Let $c_0 = 0$, and let z_i be given by the following truth table:

x_i	y_i	c_i	c_{i+1}	z_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

"Full Adder"

Area: $O(n)$. Delay: $O(n)$.