notes w/ annotations

CpE111
Introduction to Computer Engineering

Dr. Minsu Choi

CH 2. Boolean Algebra & Logic Gates

UMR UNIVERSITY OF MISSOURI-ROLLA
The Name. The Degree. The Difference.

Date representation & processing

- In the binary # system, information (date) is represented entirely by using the binary digits (bits) 0 and 1.
- Map 0 and 1 to F (false) and T (true) -> logical operations can be done.
- Ex) Logic cell w/ three input ports & one output port.

One way to define a binary function => Truth tables. (Function)

Input A	Input B	Input C	output F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

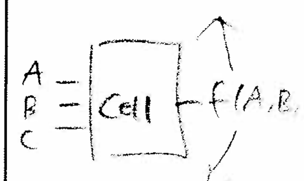
must be 8, same

$2^3 = 8$.

In general, bit patterns listed in ascending order from all zeros to all ones.

Determined by cell's functionality.

binary function of A, B & C.



called a binary function.

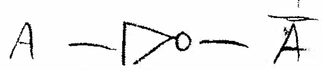
Basic logic operations

- 1. NOT - changes the value of a variable from 0 to 1 or vice versa.
- Ex) truth table

A	$\bar{A}$
0	1
1	0

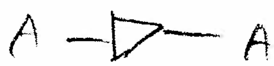
→ Simplified notation called the complement of A.

Another way to represent logic functions. ⇒ logic diagrams (≈ pictorial descriptions).



graphical symbol of NOT function called an inverter (operation itself is called inversion)

⇒ based on.



Buffer symbol (input = output)

Continued,

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

→ OR operation. (A OR B = A+B = A ∨ B)

→ If one of two input is 1, then output is 1.

- OR gate

OR gate symbol

A+B+C+D ⇒ OR4 gate (4-input OR gate)

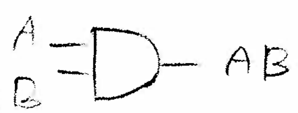
Continued,

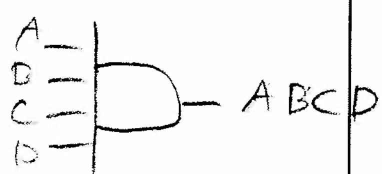
- AND gate → AND operation (= A AND B = AB = A ∧ B)

A	B	A · B
0	0	0
0	1	0
1	0	0
1	1	1

→ If both A and B are 1, then 1.

AND gate symbol





Basic identities

- Boolean algebra describes the behavior of binary variables that are subjected to the multiple NOT, OR and AND operations. -> Some identities can be used to simplify complex logic expressions.
- NOT identity

$$\overline{(\overline{A})} = A$$

$\approx$ NOT [NOT(A)] = A

⇒ involution theorem.

Continued,

- OR identities

$A+1=1$
$A+0=A$
$A+A=A$ → idempotent theorem
$A+\bar{A}=1$ → complementary property

- AND identities

$A\cdot 0=0$
$A\cdot 1=A$
$A\cdot A=A$ → idempotent theorem
$A\cdot\bar{A}=0$ → complementary property

Algebraic laws

- Commutative laws: allows us to arrange variables in any other without changing the result.

$A+B=B+A$
$A\cdot B=B\cdot A$

Continued,

- Associative laws

$A + B + C = (A+B) + C = A + (B+C)$

$A \cdot B \cdot C = (A \cdot B) + C = A \cdot (B \cdot C)$

⇒ order of evaluation does not matter.

- Ex) Combination of AND and OR?

$A \cdot (B+C) \neq A \cdot B + C = (A \cdot B) + C$

Since AND operation has higher priority.

So, parentheses are necessary.

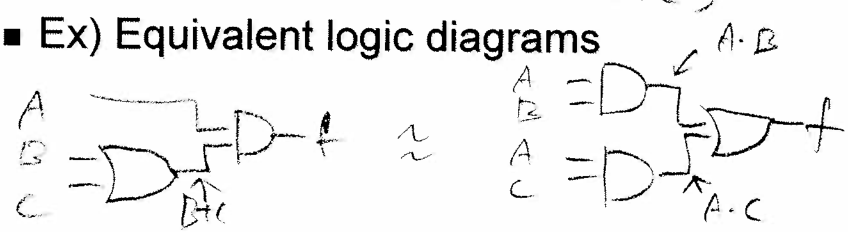
Continued,

- Distributive laws (both AND and OR)

$A \cdot (B+C) = (A \cdot B) + (A \cdot C)$

$A + (B \cdot C) = (A+B) \cdot (A+C)$

- Ex) Equivalent logic diagrams



output of OR gate → input of AND gate

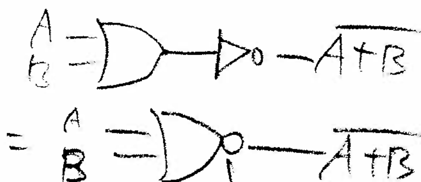
⇒ this kind of wiring scheme is "logic cascade".

NOR and NAND gates

- NOR = NOT-OR, NAND = NOT-AND

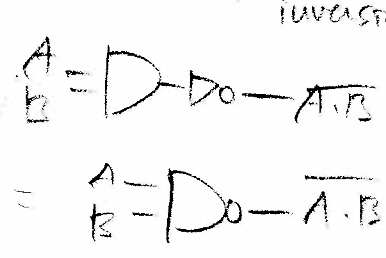
$\overline{A+B}$

AB	$\overline{A+B}$
00	1
01	0
10	0
11	0



$\overline{A\cdot B}$

AB	$\overline{A\cdot B}$
00	1
01	1
10	1
11	0



AB	$\overline{A+B}$	$\bar{A}\cdot\bar{B}$
00	1	1
01	0	0
10	0	0
11	0	0

Same!

DeMorgan's Theorems

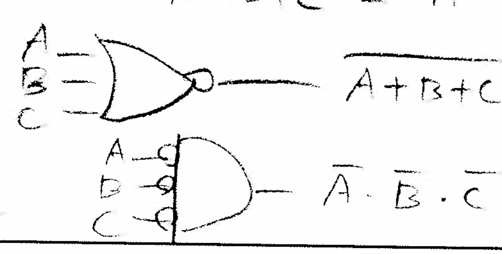
- Provides alternative expressions that relate the NOR and NAND operations to each other.
- Ex) NOR2 gate with input A & B

$$\overline{A+B} = \bar{A}\cdot\bar{B}$$

⇒ NOR is equivalent to ANDing the complements of the inputs

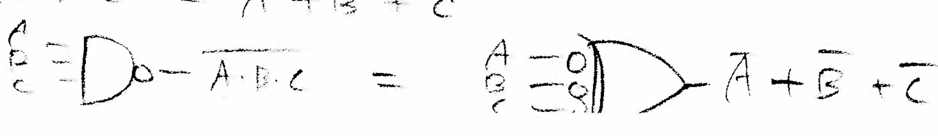
NOR3 : $\overline{A+B+C} = \bar{A}\cdot\bar{B}\cdot\bar{C}$

⇒



NAND2 $\overline{A\cdot B} = \bar{A}+\bar{B}$

$\overline{A\cdot B\cdot C} = \bar{A}+\bar{B}+\bar{C}$



AB	$\overline{A\cdot B}$	$\bar{A}+\bar{B}$
00	1	1
01	1	1
10	1	1
11	0	6

Useful Boolean Identities

$A + AB = A$

proof) using distributive law.

$A + AB = A \cdot 1 + A \cdot B$

$= A(1 + B)$

$= A \cdot 1$

$= A$

$A + \bar{A} \cdot B = A + B$

proof) $A + \bar{A}B = \boxed{A + AB} + \bar{A} \cdot B$

$= A + (A + \bar{A})B$

$= A + 1 \cdot B$

$= A + B.$

Algebraic Reductions

- Reduction of a logic expression to the "simplest" form to implement the function using the smallest # of gates.
- The basic reduction rules are summarized in Table 3.1 (page 70).
- Ex1) $f = A \cdot B + A \cdot \bar{B}$ ↓ distributive law

  $= A(B + \bar{B})$ ↓ complementary property.

  $= A \cdot 1$ ↓ AND identity.

  $= A$

Continued,

■ Ex2)

$F = A\cdot B\cdot C + B\cdot C$ ⇒ B·C term is common

$= A\cdot(B\cdot C) + (B\cdot C)$

$= (A+1)\cdot(B\cdot C)$ ↓ distributive law

$= 1\cdot(B\cdot C)$ ↓ OR identity.

$= B\cdot C$ ↓ AND identity.

■ Ex3)

$g = \overline{(a+\bar{b}+c) + (b+\bar{c})}$ ↓ DeMorgan's theorem

$= \overline{(a+\bar{b}+c)}\cdot\overline{(b+\bar{c})}$ ↓ once more

$= (\bar{a}\cdot b\cdot\bar{c})\cdot(\bar{b}\cdot c)$

$= \bar{a}\cdot b\cdot\bar{c}\cdot\bar{b}\cdot c$

$= \bar{a}\cdot(b\cdot\bar{b})\cdot(c\cdot\bar{c})$

$= \bar{a}\cdot 0\cdot 0$

$= 0.$ ⇒ g is constant 0.

Continued,

OR3 AND2 OR2

■ Ex4)

$h = (A+B+C)\cdot(A+B)$

$= (A+B+C)\cdot A + (A+B+C)\cdot B$ ↓ distributive law

$= A\cdot A + A\cdot B + AC + A\cdot B + B\cdot B + BC$

$= A + A\cdot B + B + A\cdot C + B\cdot C$

$= A\cdot(1+B) + B + (A+B)\cdot C$

$= (A+B) + (A+B)\cdot C$

$= (A+B) + (1\cdot C)$

$= \underline{A+B+C}$

↳ one OR3 gate.

※ Other logic sets that are equivalent to {NOT, AND, OR} are also complete.

Complete Logic Sets

- A complete logic set of logic operation is one that allows us to create every possible logic functions using only those in the set.
  - {NOT, AND, OR} → Any logic function can be implemented by these ops.
  - {NOT, OR} → AND can be implemented by DeMorgan's law. $\overline{\bar{A}+\bar{B}} = A \cdot B$

Continued,

- {NOT, AND} ex) $\overline{\bar{A} \cdot \bar{B}} = A + B$.
- Note that {AND, OR} is not a complete logic set, since NOT cannot be produced by them.
- {NAND}
- {NOR}

} are also complete sets.

NAND-based logic

NAND

- Ex) $\overline{A \cdot A} = \overline{A}$ → NOT

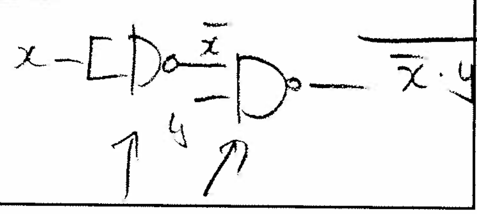
$\overline{\overline{A}\cdot\overline{B}} = A+B$ → OR

involution theorem.

ex) $f = x+\overline{y} = \overline{\overline{x+\overline{y}}} = \overline{\overline{x}\cdot y}$

NAND

$\overline{\overline{x}\cdot y}$ → logic diagram



$\overline{\overline{x}\cdot y} = (x+\overline{y})$

NOT

NAND gates only.

NOR-based logic

$\overline{A+A} = \overline{A}$ (NOT)

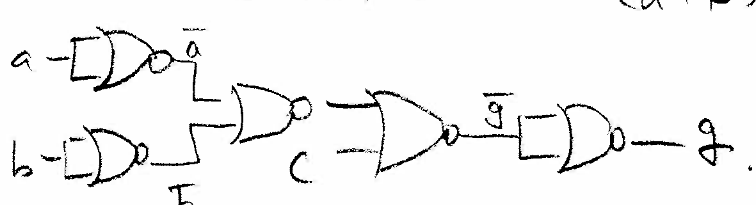
$\overline{\overline{A}+\overline{B}} = A\cdot B$ (AND)

- Ex) $g = a\cdot b + c$

DeMorgan reduction

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

$= \overline{(\overline{a}+\overline{b})} + c = \overline{\overline{(\overline{a}+\overline{b})+c}}$



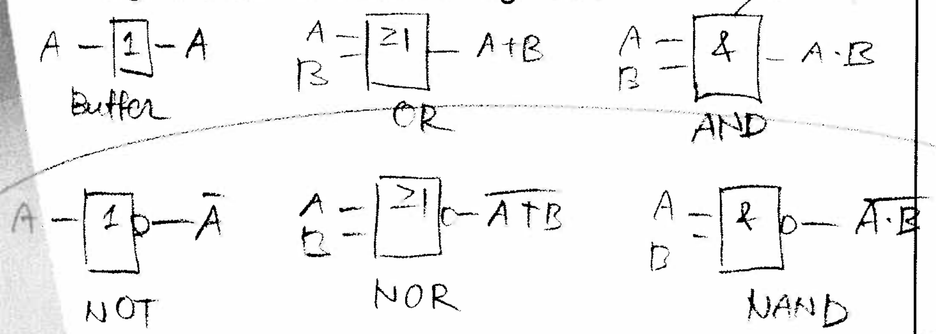
NOR only.

Shape-specific symbols have been introduced so far

IEEE Logic Gate Symbols

- Very common in practice (alternative way).
- IEEE is an acronym that stands for the Institute of Electrical & Electronics Engineers, a professional organization for electrical engineers.

ampersand



~~Continued,~~

Program Completed

University of Missouri-Rolla

UMR UNIVERSITY OF MISSOURI-ROLLA
The Name. The Degree. The Difference.