Design of Digital Circuits
Lab 1 Supplement

Prof. Onur Mutlu
ETH Zurich

Spring 2018

6 March 2018

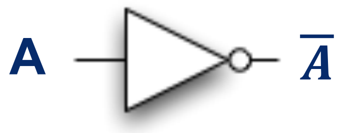
What We Will Learn?

- Boolean Equations
  - Logic operations with binary numbers
- Logic Gates
  - Basic blocks that are interconnected to form larger units that are needed to construct a computer

Boolean Equations and Logic Gates

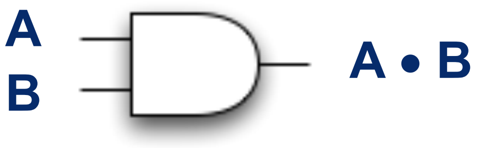
Simple Equations: NOT / AND / OR

$\overline{A}$ *(reads "not A")* is 1 iff A is 0



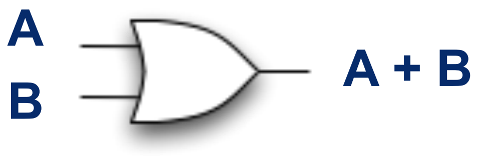
A	$\overline{A}$
0	1
1	0

A • B *(reads "A and B")* is 1 iff A and B are both 1



A	B	$A \bullet B$
0	0	0
0	1	0
1	0	0
1	1	1

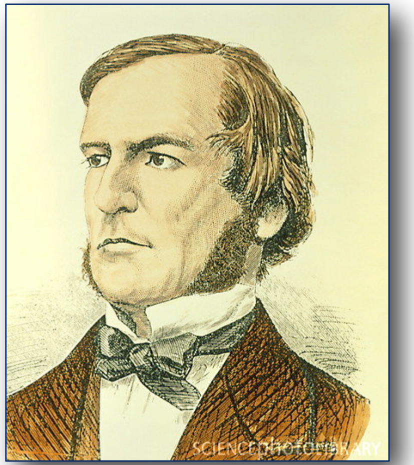
A + B *(reads "A or B")* is 1 iff either A or B is 1



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

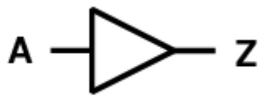
Boolean Algebra: Big Picture

- An algebra on 1's and 0's
  - with AND, OR, NOT operations
- What you start with
  - Axioms: basic stuff about objects and operations you just assume to be true at the start
- What you derive first
  - Laws and theorems: allow you to manipulate Boolean expressions
  - …also allow us to do some simplification on Boolean expressions
- What you derive later
  - More "sophisticated" properties useful for manipulating digital designs represented in the form of Boolean equations



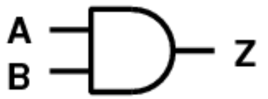
Common Logic Gates

Buffer



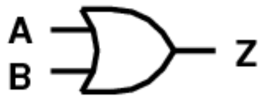
A	Z
0	0
1	1

AND



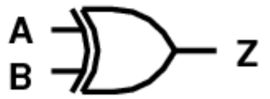
A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

OR



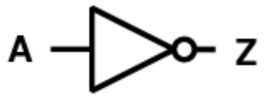
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

XOR



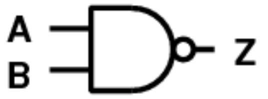
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

Inverter



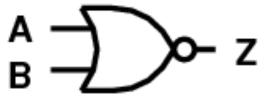
A	Z
0	1
1	0

NAND



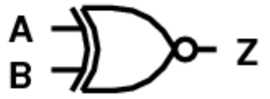
A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

NOR



A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

XNOR



A	B	Z
0	0	1
0	1	0
1	0	0
1	1	1

Boolean Algebra: Axioms

Formal version	*English version*
1. *B* contains at least two elements, *0* and *1*, *such that* $0 \neq 1$	Math formality...
2. *Closure* $a,b \in B$, (i) $a + b \in B$ (ii) $a \bullet b \in B$	Result of AND, OR stays in set you start with
3. *Commutative Laws*: $a,b \in B$, (i) (ii)	For primitive AND, OR of 2 inputs, order doesn't matter
4. *Identities*: $0, 1 \in B$ (i) (ii)	There are identity elements for AND, OR, give you back what you started with
5. *Distributive Laws*: (i) (ii)	• distributes over +, just like algebra …but + distributes over •, also (!!)
6. *Complement*: (i) (ii)	There is a complement element, ANDing, ORing give you an identity

SAFARI

Boolean Algebra: Duality

- Interesting observation
  - All the axioms come in "dual" form
  - Anything true for an expression also true for its dual
  - So any derivation you could make that is true, can be flipped into dual form, and it stays true

- Duality -- More formally
  - A dual of a Boolean expression is derived by replacing
    - Every AND operation with... an OR operation
    - Every OR operation with... an AND
    - Every constant 1 with... a constant 0
    - Every constant 0 with... a constant 1
    - But don't change any of the literals or play with the complements!

Example

$$a \bullet (b + c) = (a \bullet b) + (a \bullet c)$$

$$\rightarrow \; a + (b \bullet c) = (a + b) \bullet (a + c)$$

Boolean Algebra: Useful Laws

Dual $\downarrow$

Operations with 0 and 1:

1. $X + 0 = X$ | 1D. $X \bullet 1 = X$
2. $X + 1 = 1$ | 2D. $X \bullet 0 = 0$

AND, OR with identities gives you back the original variable or the identity

Idempotent Law:

3. $X + X = X$ | 3D. $X \bullet X = X$

AND, OR with self = self

Involution Law:

4. $\overline{(\overline{X})} = X$

double complement = no complement

Laws of Complementarity:

5. $X + \overline{X} = 1$ | 5D. $X \bullet \overline{X} = 0$

AND, OR with complement gives you an identity

Commutative Law:

6. $X + Y = Y + X$ | 6D. $X \bullet Y = Y \bullet X$

Just an axiom…

Useful Laws (cont)

Associative Laws:

7. $(X + Y) + Z = X + (Y + Z) = X + Y + Z$

7D. $(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z) = X \cdot Y \cdot Z$

Parenthesis order doesn't matter

Distributive Laws:

8. $X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$

8D. $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$

Axiom

Simplification Theorems:

9.

9D.

10.

10D.

11.

11D.

Useful for simplifying expressions

Actually worth remembering — they show up a lot in real designs…

DeMorgan's Law

DeMorgan's Law:

12. $\overline{(\boldsymbol{X} + \boldsymbol{Y} + \boldsymbol{Z} + \cdots)} = \overline{\boldsymbol{X}}.\overline{\boldsymbol{Y}}.\overline{\boldsymbol{Z}}.\ldots$

12D. $\overline{(\boldsymbol{X}.\boldsymbol{Y}.\boldsymbol{Z}.\ldots)} = \overline{\boldsymbol{X}} + \overline{\boldsymbol{Y}} + \overline{\boldsymbol{Z}} + \ldots$

- **Think of this as a transformation**
  - Let's say we have:

$$F = A + B + C$$

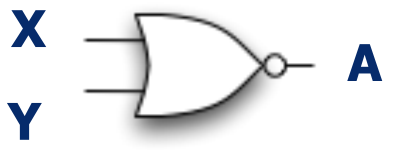
  - Applying DeMorgan's Law (12), gives us:

$$\boldsymbol{F} = \overline{\overline{(\boldsymbol{A} + \boldsymbol{B} + \boldsymbol{C})}} = \overline{(\overline{\boldsymbol{A}}.\overline{\boldsymbol{B}}.\overline{\boldsymbol{C}})}$$

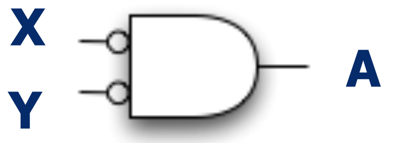
DeMorgan's Law (cont.)

Interesting — these are conversions between different types of logic
That's useful given you don't always have every type of gate

$$A = \overline{(X+Y)} = \bar{X}\bar{Y}$$

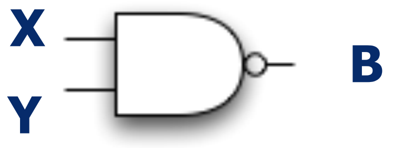


NOR is equivalent to AND with inputs complemented

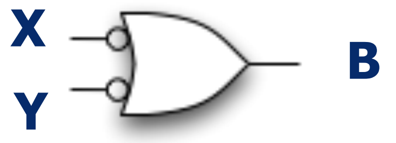


X	Y	$X+Y$	X	Y	XY
0	0	1	1	1	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	0

$$B = \overline{(XY)} = \bar{X} + \bar{Y}$$



NAND is equivalent to OR with inputs complemented



X	Y	XY	X	Y	$X+Y$
0	0	1	1	1	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0

Design of Digital Circuits
Lab 1 Supplement

Prof. Onur Mutlu
ETH Zurich

Spring 2018

6 March 2018