# Digital Design & Computer Arch.

## Lab 1 Supplement:
## Drawing Basic Circuits

Prof. Onur Mutlu
ETH Zurich
Spring 2021
9 March 2021

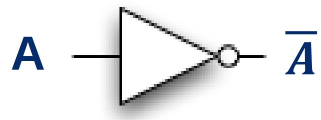
# What We Will Learn?

- In Lab 1, you will design simple combinatorial circuits
- We will cover a tutorial about:
  - 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

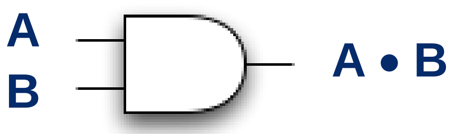
# Simple Equations: NOT / AND / OR

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



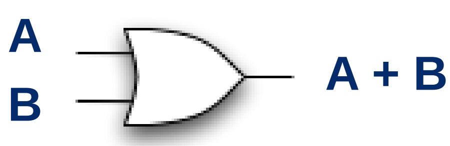

| $A$ | $\overline{A}$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

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



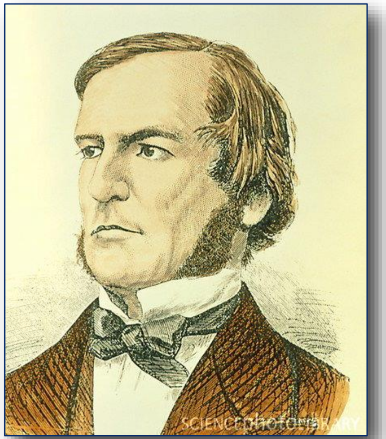

| $A$ | $B$ | $A \bullet B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

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 |

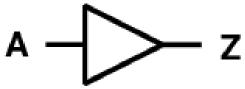
# Boolean Algebra: Big Picture



- 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

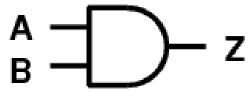
# Common Logic Gates

**Buffer**




| A | Z |
|---|---|
| 0 | 0 |
| 1 | 1 |

**AND**




| A | B | Z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

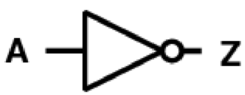
**OR**


| A | B | Z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**XOR**

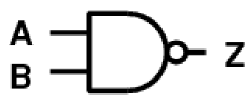

| A | B | Z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Inverter**



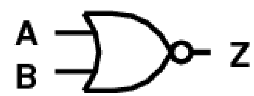

| A | Z |
|---|---|
| 0 | 1 |
| 1 | 0 |

**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* $a,b \in B$,<br>(i) $a + b \in B$<br>(ii) $a \bullet b \in B$ | Result of AND, OR stays in set you start with |
| 3. *Commutative Laws*: $a,b \in B$,<br>(i) $a + b = b + a$<br>(ii) $a \bullet b = b \bullet a$ | For primitive AND, OR of 2 inputs, order doesn't matter |
| 4. *Identities*: $0, 1 \in B$<br>(i) $a + 0 = a$<br>(ii) $a \bullet 1 = a$ | There are identity elements for AND, OR, give you back what you started with |
| 5. *Distributive Laws*:<br>(i) $a + (b \bullet c) = (a + b) \bullet (a + c)$<br>(ii) $a \bullet (b + c) = a \bullet b + a \bullet c$ | • distributes over +, just like algebra<br>…but + distributes over •, also (!!) |
| 6. *Complement*:<br>(i) $a + a' = 1$<br>(ii) $a \bullet a' = 0$ | There is a complement element, ANDing, ORing give you an identity |

# 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**

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

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

# Boolean Algebra: Useful Laws

Dual

*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. $(X \bullet Y) \bullet Z = X \bullet (Y \bullet Z)$
$= X \bullet Y \bullet Z$

Parenthesis order doesn't matter

*Distributive Laws:*

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

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

Axiom

*Simplification Theorems:*

9. $X \bullet Y + X \bullet \overline{Y} = X$

9D. $(X + Y) \bullet (X + \overline{Y}) = X$

10. $X + X \bullet Y = X$

10D. $X \bullet (X + Y) = X$

11. $(X + \overline{Y}) \bullet Y = X \bullet Y$

11D. $(X \bullet \overline{Y}) + Y = X + Y$

Useful for simplifying expressions

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

# DeMorgan's Law

***DeMorgan's Law:***

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

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

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

$$F = A + B + C$$

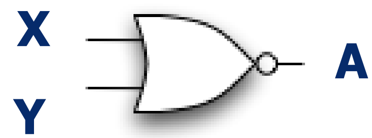
  - Applying DeMorgan's Law (12), gives us:

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

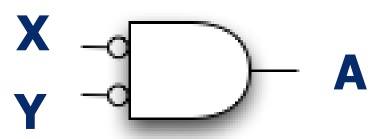
# DeMorgan’s Law (cont.)

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

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

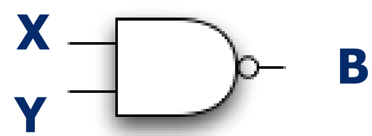



**NOR is equivalent to AND with inputs complemented**

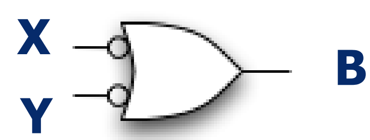



| $X$ | $Y$ | $\overline{X+Y}$ | $\overline{X}$ | $\overline{Y}$ | $\overline{X}\,\overline{Y}$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |

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




**NAND is equivalent to OR with inputs complemented**




| X | Y | $\overline{XY}$ | $\overline{X}$ | $\overline{Y}$ | $\overline{X}+\overline{Y}$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |

# Part 1: A Comparator Circuit

- Design a comparator that receives two 4-bit numbers A and B, and sets the output bit EQ to logic-1 if A and B are equal


- Hints:
  - First compare A and B bit by bit
  - Then combine the results of the previous steps to set EQ to logic-1 if all A and B are equal

# Part 2: A More General Comparator

- Design a circuit that receives two 1-bit inputs A and B, and:
  - sets its first output (O1) to 1 if *A>B*,
  - sets the second output (O2) to 1 if *A=B*,
  - sets the third output (O3) to 1 if *A<B*.

# Part 3: Circuits with Only NAND Gates

- Design the circuit of Part 2 using only NAND gates

- Logical Completeness:
  - The set of gates {AND, OR, NOT} is logically complete because we can build a circuit to carry out the specification of any combinatorial logic we wish, without any other kind of gate

  - NAND and NOR are also logically complete

# Last Words

- In this lab, you will draw the schematics of some simple operations
- Part 1: A comparator circuit
- Part 2: A more general comparator circuit
- Part 3: Designing circuits using only NAND gates
- You will find more exercises in the lab report

# Report Deadline

23:59, 26 March 2021

# Digital Design & Computer Arch.

## Lab 1 Supplement:
## Drawing Basic Circuits

Prof. Onur Mutlu
ETH Zurich

Spring 2021

9 March 2021


SAFARI