
Digital Design & Computer Arch.

Lab 1 Supplement: Drawing Basic Circuits

Prof. Onur Mutlu

ETH Zurich

Spring 2022

8 March 2022

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

\bar{A} (reads “not A”) is 1 iff A is 0



A	\bar{A}
0	1
1	0

$A \cdot B$ (reads “A and B”) is 1 iff A and B are both 1



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

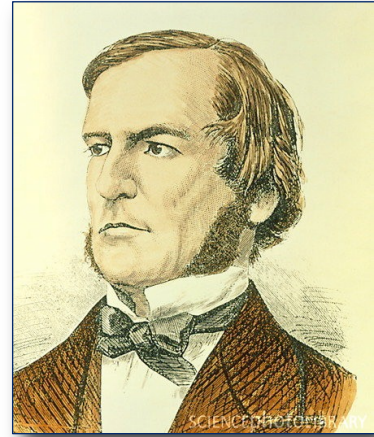
$A + B$ (reads “A or B”) is 1 iff either A or B is 1



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

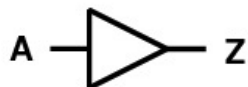
Boolean Algebra: Big Picture

- 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



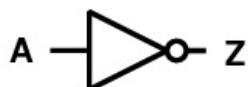
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

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

1. B contains at least two elements,
 0 and 1 , such that $0 \neq 1$

2. *Closure* $a, b \in B$,
(i) $a + b \in B$
(ii) $a \cdot b \in B$

3. *Commutative Laws*: $a, b \in B$,
(i) $a + b = b + a$
(ii) $a \cdot b = b \cdot a$

4. *Identities*: $0, 1 \in B$
(i) $a + 0 = a$
(ii) $a \cdot 1 = a$

5. *Distributive Laws*:
(i) $a + (b \cdot c) = (a + b) \cdot (a + c)$
(ii) $a \cdot (b + c) = a \cdot b + a \cdot c$

6. *Complement*:
(i) $a + a' = 1$
(ii) $a \cdot a' = 0$

English version

Math formality...

Result of AND, OR stays
in set you start with

For primitive AND, OR of
2 inputs, order doesn't matter

There are identity elements
for AND, OR, give you back
what you started with

- distributes over $+$, just like algebra
...but $+$ distributes over \cdot , also (!!)

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

$$\begin{aligned} a \cdot (b + c) &= (a \cdot b) + (a \cdot c) \\ \rightarrow a + (b \cdot c) &= (a + b) \cdot (a + c) \end{aligned}$$

Boolean Algebra: Useful Laws

Operations with 0 and 1:

1. $X + 0 = X$

2. $X + 1 = 1$

Dual



1D. $X \cdot 1 = X$

2D. $X \cdot 0 = 0$

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

Idempotent Law:

3. $X + X = X$

3D. $X \cdot X = X$

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. $X \cdot \overline{X} = 0$

AND, OR with complement
gives you an identity

Commutative Law:

6. $X + Y = Y + X$

6D. $X \cdot Y = Y \cdot X$

Just an axiom...

Useful Laws (cont.)

Associative Laws:

$$\begin{aligned} 7. (X + Y) + Z &= X + (Y + Z) \\ &= X + Y + Z \end{aligned}$$

$$\begin{aligned} 7D. (X \cdot Y) \cdot Z &= X \cdot (Y \cdot Z) \\ &= X \cdot Y \cdot Z \end{aligned}$$

Parenthesis order
doesn't matter

Distributive Laws:

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

$$8D. X + (Y \cdot Z) = (X + Y) \cdot (X + Z) \quad \text{Axiom}$$

Simplification Theorems:

$$9. X \cdot Y + X \cdot \bar{Y} = X$$

$$9D. (X + Y) \cdot (X + \bar{Y}) = X$$

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

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

$$11. (X + \bar{Y}) \cdot Y = X \cdot Y$$

$$11D. (X \cdot \bar{Y}) + Y = X + Y$$

Useful for
simplifying
expressions

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

DeMorgan's Law

DeMorgan's Law:

$$12. \overline{(X + Y + Z + \dots)} = \bar{X} \cdot \bar{Y} \cdot \bar{Z} \cdot \dots$$

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

■ Think of this as a transformation

- Let's say we have:

$$F = A + B + C$$

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

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

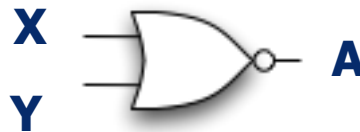
DeMorgan's Law (cont.)

Interesting — these are conversions between **different types of logic**

That's useful given you don't always have **every type of gate**

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

NOR is equivalent to AND with inputs complemented



X	Y	$\overline{X + Y}$	\bar{X}	\bar{Y}	$\bar{X}\bar{Y}$
0	0	1	1	1	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	0

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

NAND is equivalent to OR with inputs complemented



X	Y	\overline{XY}	\bar{X}	\bar{Y}	$\bar{X} + \bar{Y}$
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, 25 March 2022

Digital Design & Computer Arch.

Lab 1 Supplement: Drawing Basic Circuits

Prof. Onur Mutlu

ETH Zurich

Spring 2022

8 March 2022