18-100 Introduction to Electrical and Computer Engineering

Lecture 04

Binary Numbers, Boolean Logic, and Logic Gates

Today's Digital World: "0" and "1" Binary Number

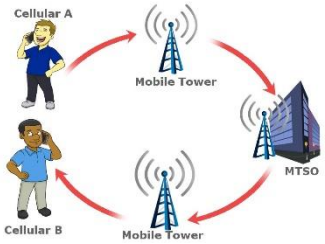
Digital music "10011011110001…1000100010…0100101111001..."

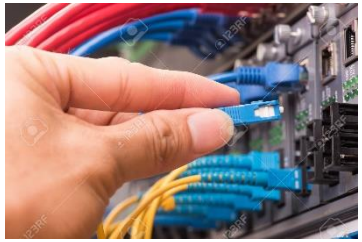
Data Center

Everything is stored as

"100…11011110001…1000100010…01001011110"

Communications



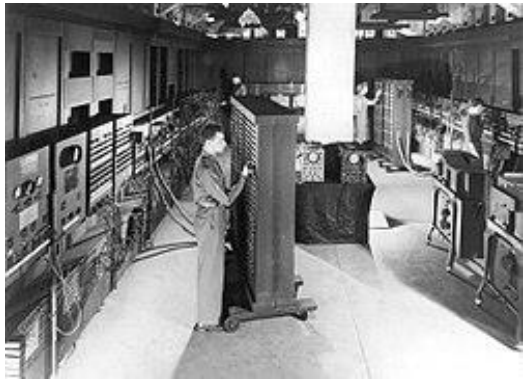


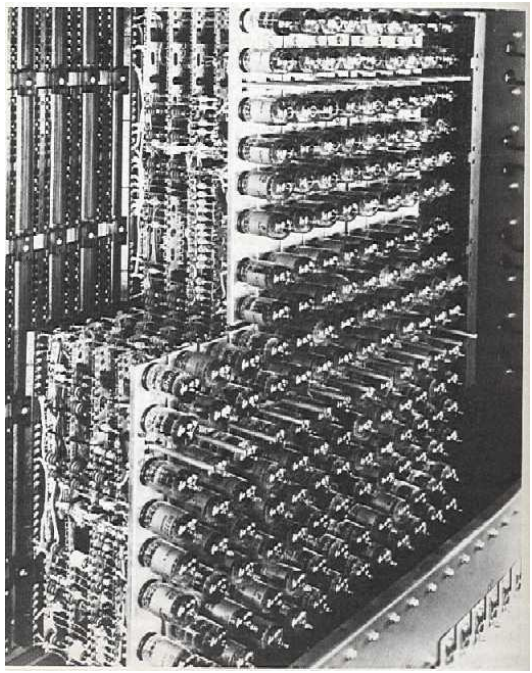
Everything is transmitted as "100…11011110001…1000100010…01001011110"

Quiz

1. The first general purpose computer, ENIAC, was built at Univ. of Pennsylvania in 1946 and was in operation until 1956. It contains 20,000 vacuum tubes. It used 10-digit decimal numbers for computation. Can you guess what is the clock speed of ENIAC?

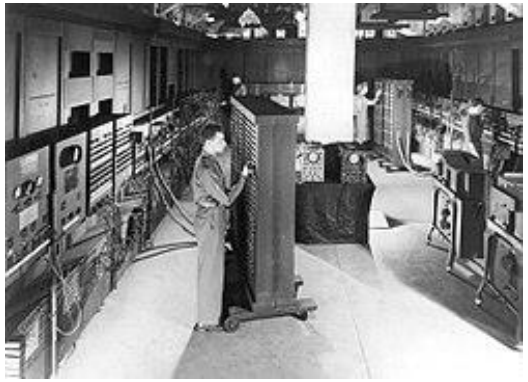
- *3.5 GHz*
- *25 kHz.*
- *50 Hz.*

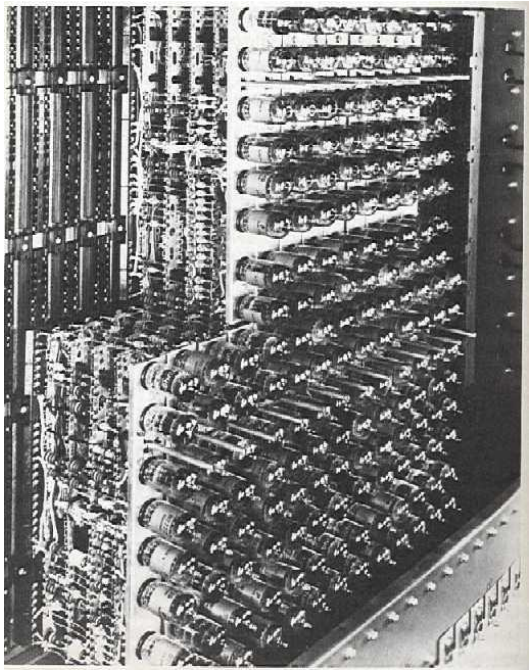




Quiz

1. The first general purpose computer, ENIAC, was built at Univ. of Pennsylvania in 1946 and was in operation until 1956. It contains 20,000 vacuum tubes. It used 10-digit decimal numbers for computation. Can you guess what is the clock speed of ENIAC?



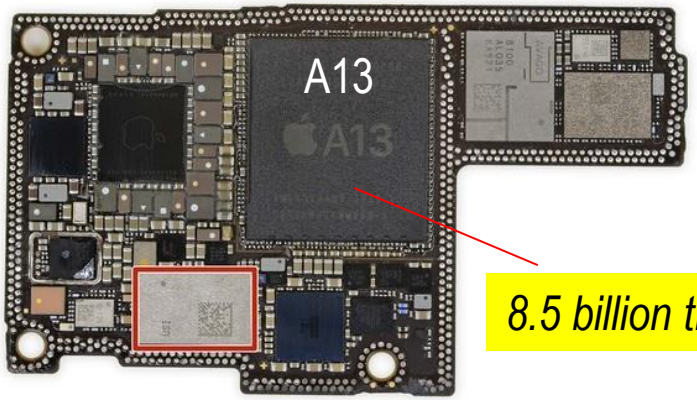


- ❑ *3.5 GHz* *Intel CORE i7 Microprocessor.*
- ☑ *25 kHz.*
- ❑ *50 Hz.* *Frequency of ac power from wall outlet.*

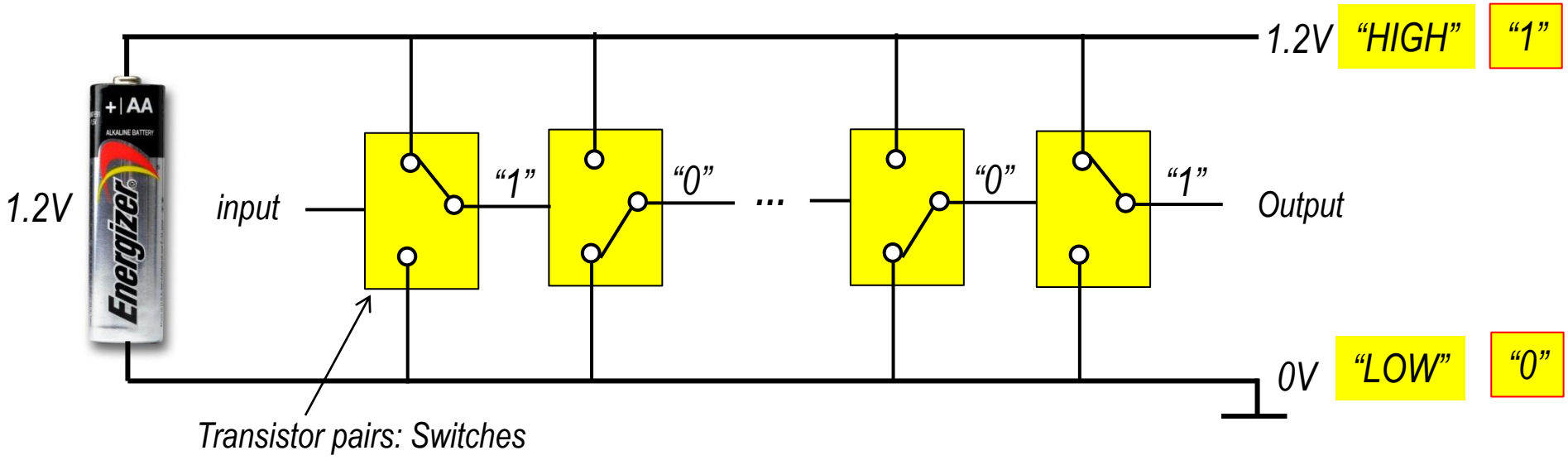
Learning Objectives for This Lecture

- *Why today's computer using binary numbers.*
- *What is Boolean logic and what are logic gates.*
- *How to use logic gates to build logic functions.*
- *How computer uses binary numbers to control "things".*
- *How computer uses logic gates to do "computation".*

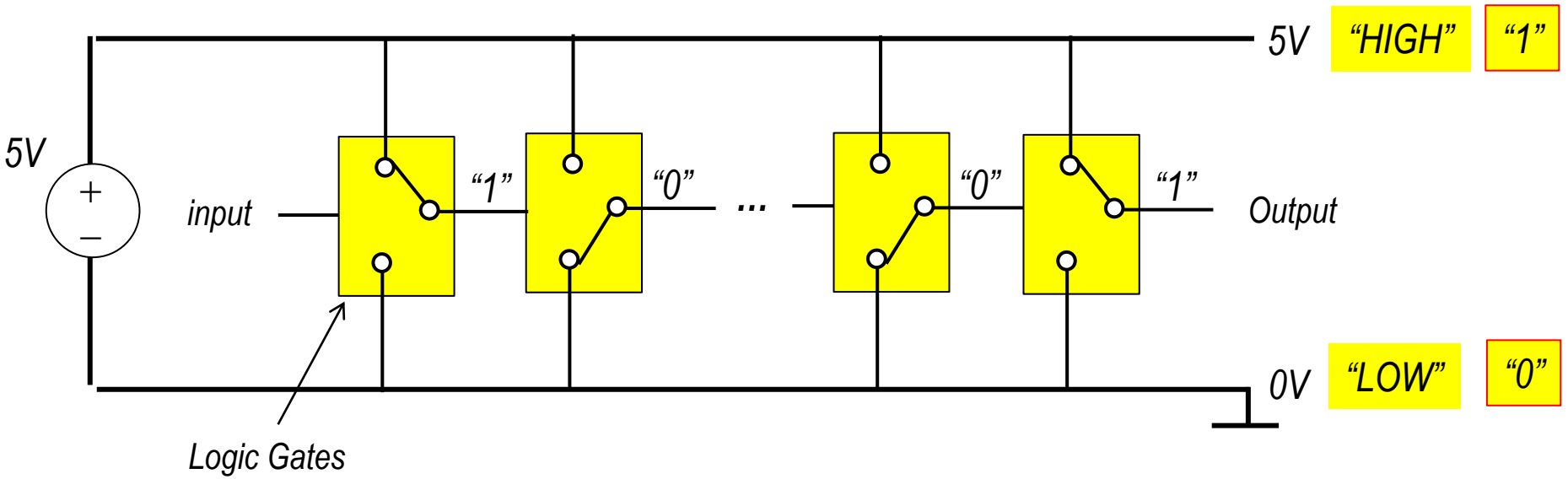
"1" and "0" : Inside a Microprocessor



8.5 billion transistors

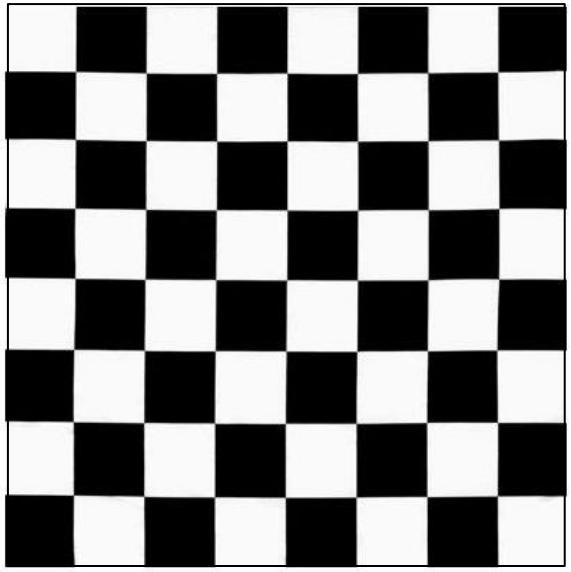


The Logic Levels Used Here



Logic Value	Logic Value	Logic Value	Voltage Value
0	False	LOW	0 (V)
1	True	HIGH	5 (V)

Why "1" and "0"



***Black* ("0") *and White* ("1")**

No Gray

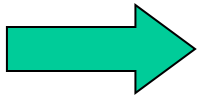
Ensure No Mistakes: It is not so easy to "confuse" black and "white"

Why "1" and "0"

Resilient to Noise

1.0	0.3	0.8	0.0	0.6	0.3
0.1	0.8	0.3	0.8	0.1	1.0
0.8	0.1	1.0	0.2	0.8	0.1
0.0	0.8	0.0	0.8	0.2	0.8
0.8	0.3	1.0	0.2	0.8	0.3
0.1	0.7	0.1	0.8	0.0	0.8

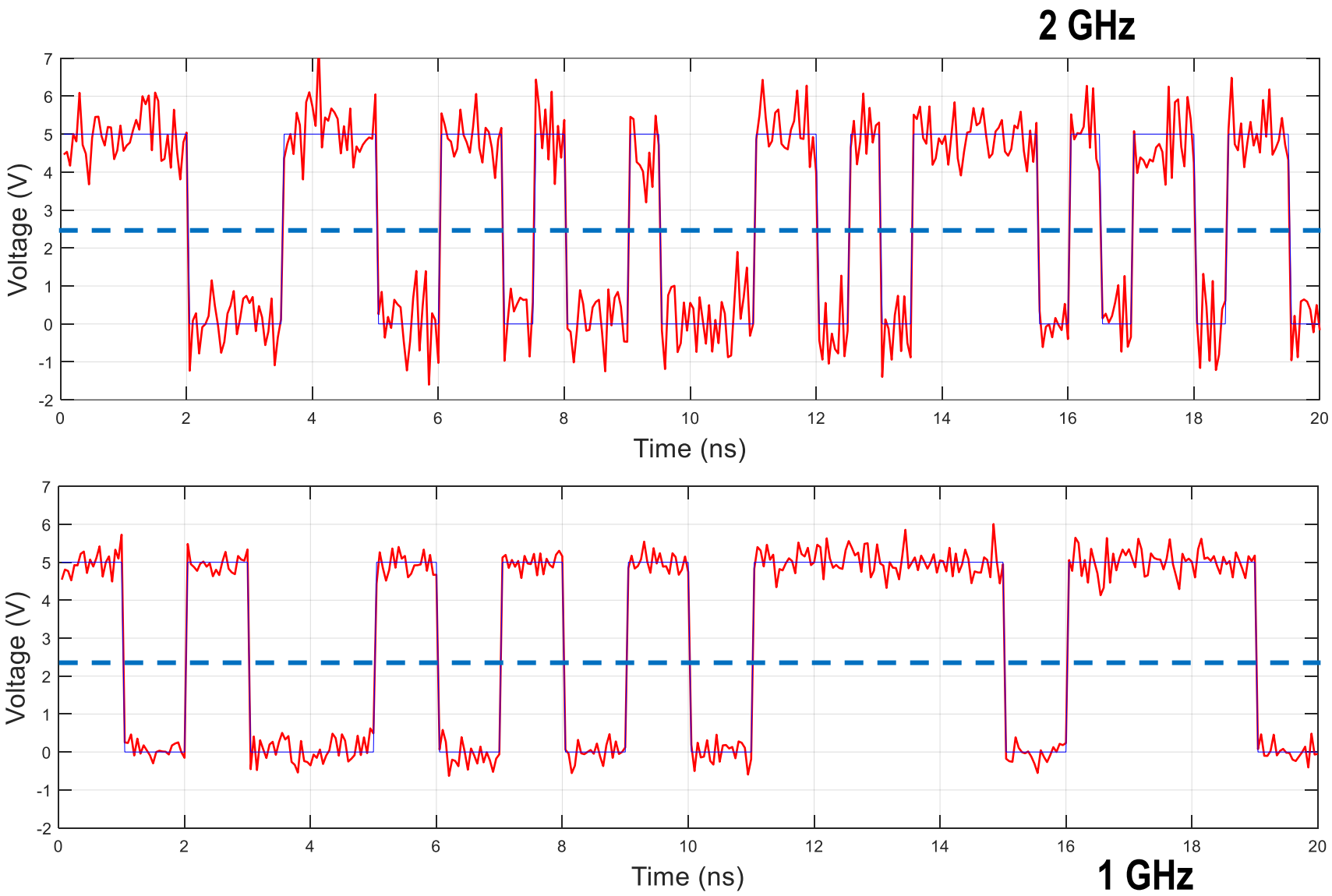
Threshold = 0.5



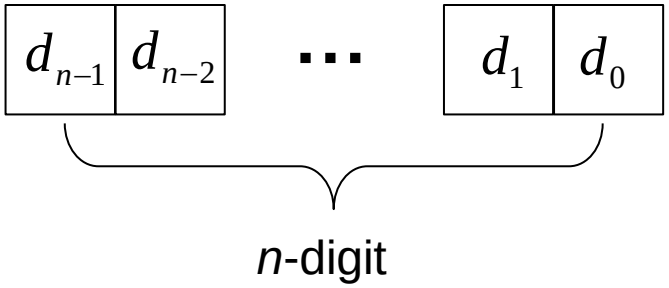
1	0	1	0	1	0
0	1	0	1	0	1
1	0	1	0	1	0
0	1	0	1	0	1
1	0	1	0	1	0
0	1	0	1	0	1

Ensure No Mistakes: It is not so easy to "confuse" black and "white"

Binary Minimize the Impact of Noise



Decimal Number System



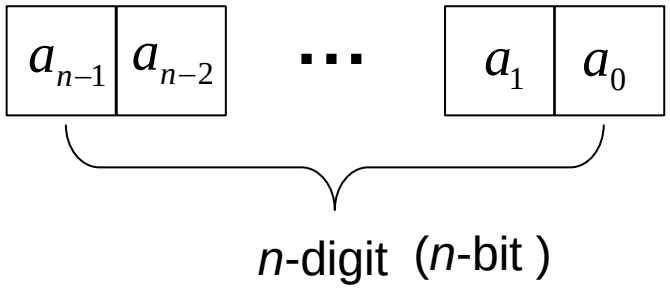
$$d_i = \begin{cases} 9 \\ 8 \\ \vdots \\ \vdots \\ 1 \\ 0 \end{cases} \quad 0 \quad \text{to} \quad 10^n - 1$$

$$d = a_{n-1} \times 10^{n-1} + a_{n-2} \times 10^{n-2} + \ldots + a_1 \times 10^1 + a_0 \times 10^0$$

1982 ⇨ $1 \times 10^3 + 9 \times 10^2 + 8 \times 10^1 + 2 \times 10^0$

$= 1000 + 900 + 80 + 2 = 1982$

Binary Number System



$$a_i = \begin{cases} 0 \\ 1 \end{cases}$$

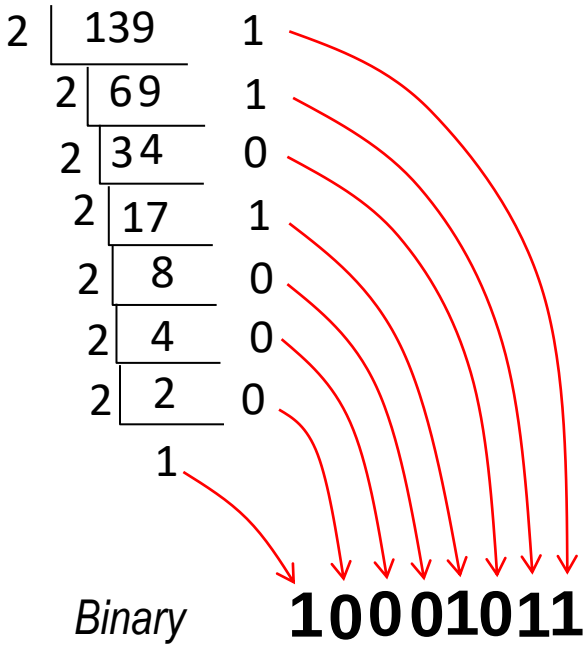
A n-bit binary number has 2^n *number of possible values, ranging from* 0 to $2^n - 1$

Conversion to decimal: $d = a_{n-1} \times 2^{n-1} + a_{n-2} \times 2^{n-2} + \ldots + a_1 \times 2^1 + a_0 \times 2^0$

10101011 ⇨ $1 \times 2^7 + 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^1 + 1 \times 2^0$

$= 128 + 32 + 8 + 2 + 1 = 171$

Decimal to Binary



$$139$$

Decimal

$$139$$

$$= 2 \times 69 + 1$$

$$= 2 \times (2 \times 34 + 1) + 1$$

$$= 2 \times (2 \times 2 \times 17 + 1) + 1$$

$$= 2^3 \times (2 \times 8 + 1) + 2 + 1$$

$$= 2^7 + 2^3 + 2 + 1$$

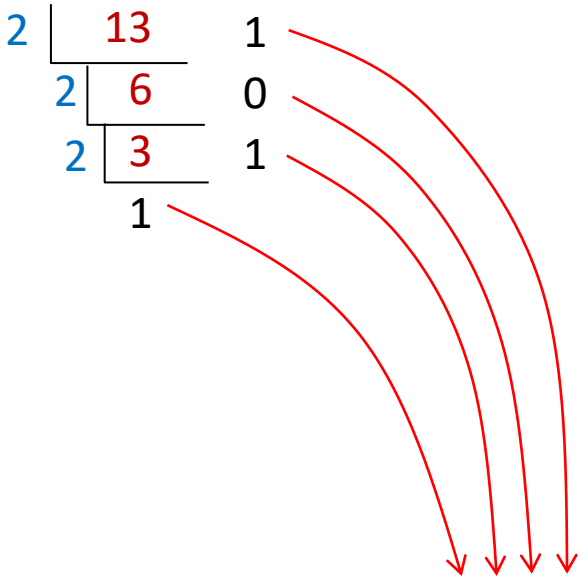
$$= 1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

Each digit will only have two values { 1, 0 } (bit)

Decimal to Binary

13

Decimal



1101

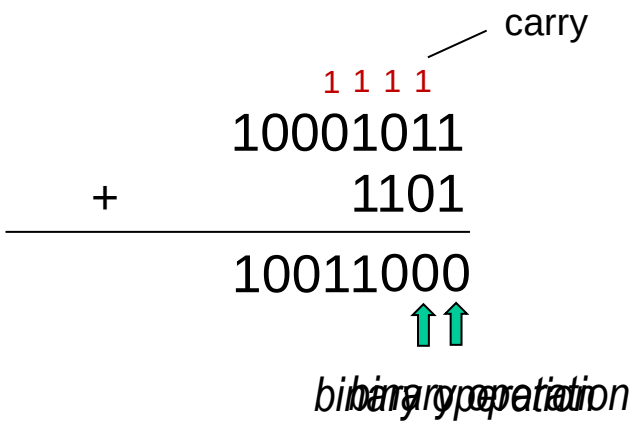
Binary

$$13$$
$$= 2 \times 6 + 1$$
$$= 2 \times 2 \times 3 + 1$$
$$= 2 \times 2 \times (2 + 1) + 1$$
$$= 2^3 + 2^2 + 1$$
$$= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1$$

Addition

Decimal **139 + 13 =**

Binary **10001011 + 1101 = 10011000**



Binary to Decimal

Decimal	**139**	**+**	**13**	**=**	**152**
Binary	**10001011**	**+**	**1101**	**=**	**10011000**

Digit number

7 6 5 4 3 2 1 0

10011000

$$2^7 + 2^4 + 2^3 = 128 + 16 + 8 = 152$$

10011000

Most significant bit (MSB)

Least significant bit (LSB)

Negative Number 8-bit Number Two's Complement

Negative Number: $x + (-x) = 0$

Take any 8-bit binary number: 0 1 1 0 1 0 1 1

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

bit-wise complement: 1 0 0 1 0 1 0 0

Add the above two numbers:

```
  0 1 1 0 1 0 1 1
+ 1 0 0 1 0 1 0 0   bit-wise complement:
-----------------
  1 1 1 1 1 1 1 1
+ 0 0 0 0 0 0 0 1   Add 1:
-----------------
1 0 0 0 0 0 0 0 0   x + (-x) = 0
  \_____8-bit_____/
```

If x = 01101011 $-x$ = 10010101

↑ bit-wise-complement + 1 **(2's Complement)**

For a 8-bit number with both positive and negative integers:

0 1 1 1 1 1 1 1 ——— *Largest positive number* 127_{10}

1 0 0 0 0 0 0 1 ——— *Smallest (negative) number* -127_{10}

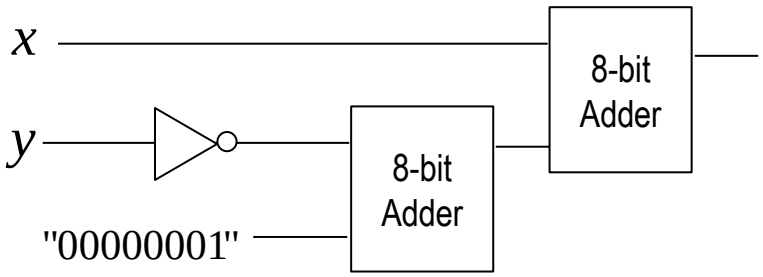
0 0 0 0 0 0 0 1 ——— 1

1 1 1 1 1 1 1 1 ——— -1 *2's complement*

1000001, 10000010, 10000011, ...,11111111,00000000, 00000001, 00000010,...,01111111

-127_{10} -1_{10} 0_{10} 1_{10} 127_{10}

Use adder for subtracting: $x - y = x + (-y) = x + (\bar{y} + 1)$



Binary Number System

2-bit binary number

Binary	Decimal
00	0
01	1
10	2
11	3

$4 = 2^2$ values

3-bit binary number

Binary	Decimal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

$8 = 2^3$ values

4-bit binary number

Binary	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

$16 = 2^4$ values

Hexadecimal Number

A Hexadecimal number is a base 16 number: Each digit has 16 values:

10 11 12 13 14 15

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

16 values

Hexadecimal to Decimal

Hex Decimal

$$2E = 2\times16^1 + 14\times16^0 = 46$$

Hexadecimal to Binary

2 E

0010 1110

Each bit in Hex is 4 bit in Binary!

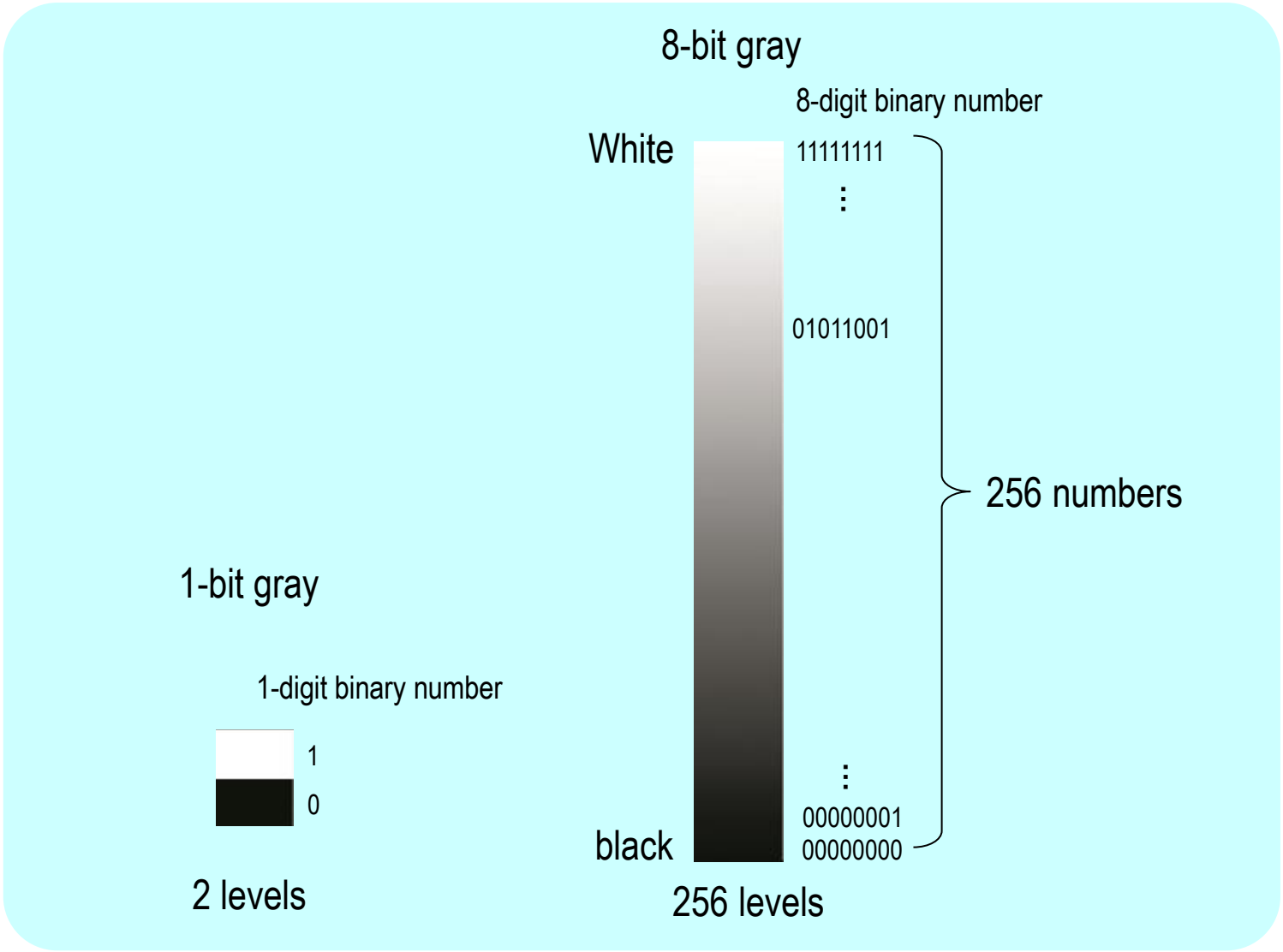
0x — *The number to follow is hexadecimal.*

0x2E = 46

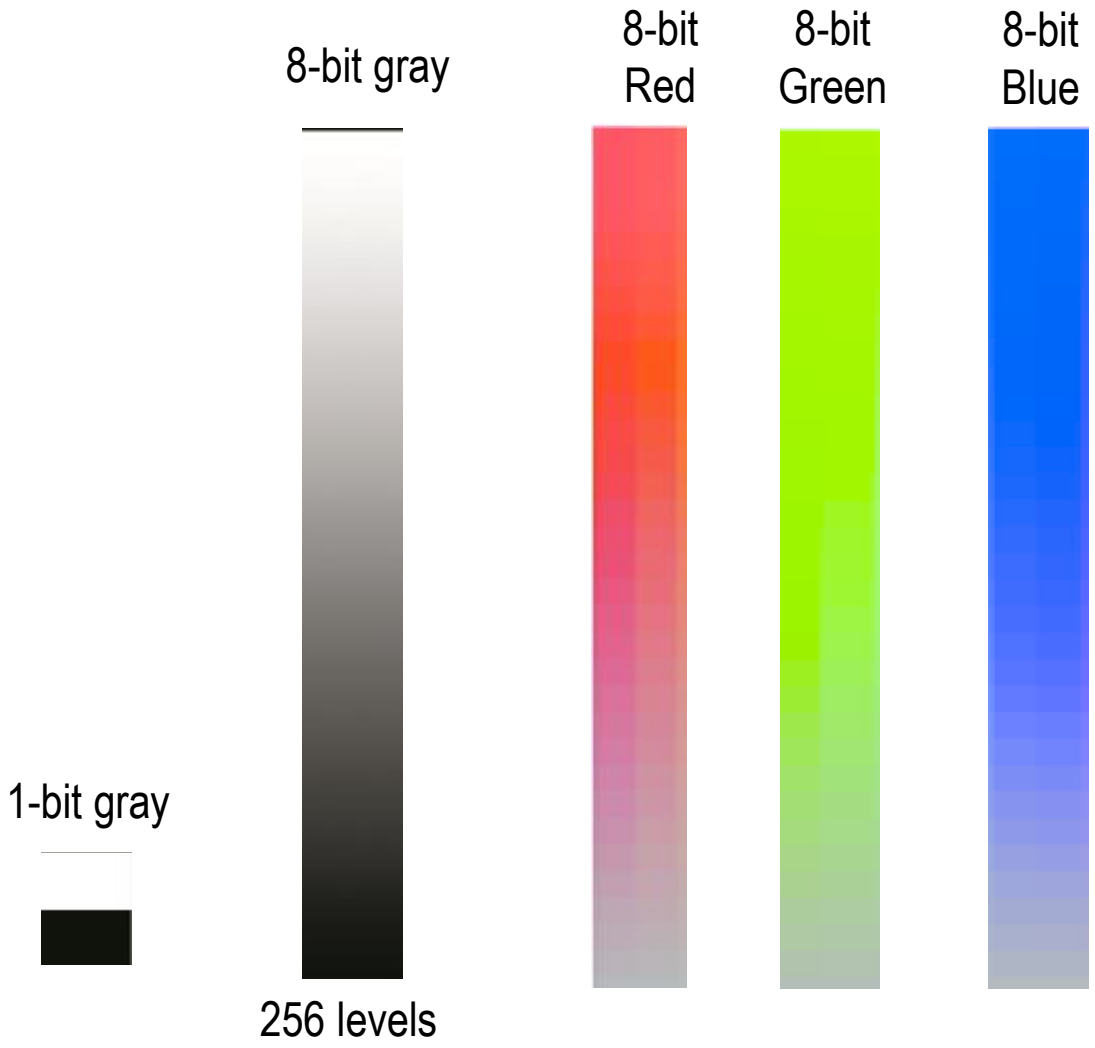
0x21 = 33

Hex Decimal

Taking More Digits



Binary to Represent Analog



24-bit Color

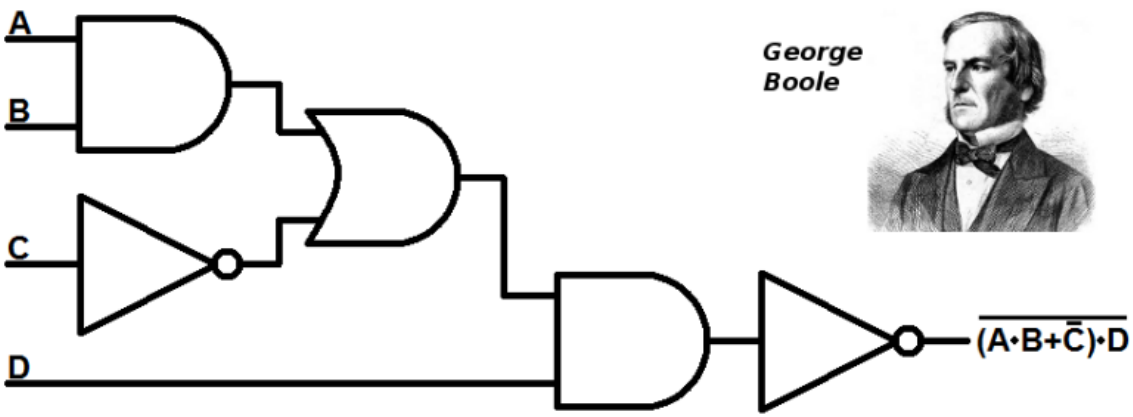
16,777,216 color variations

A Simple Truth of "1" and "0"

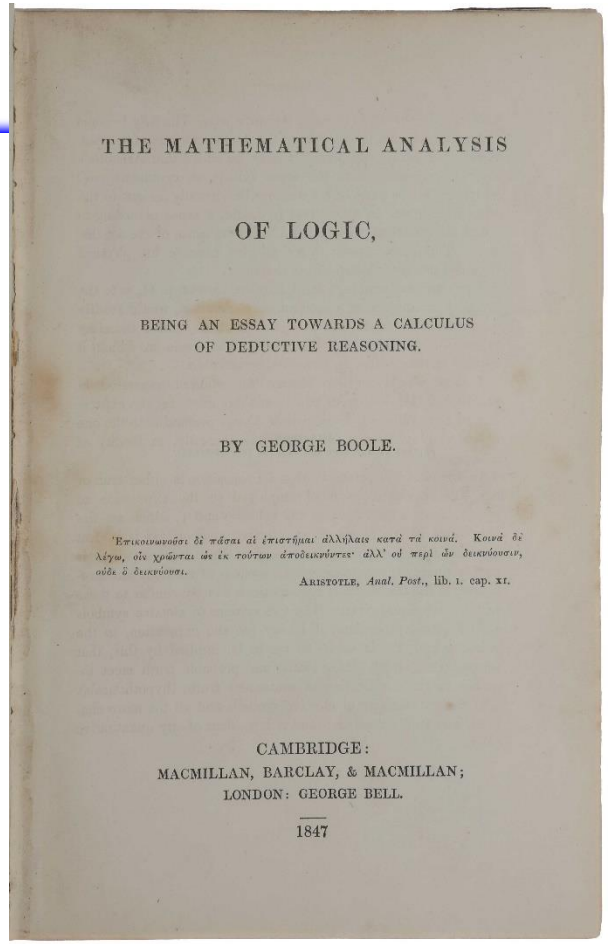
Each transistor only controls a "1" and a "0", however,

8,500,000,000 Transistors can carry out very complex functions!

Binary Logic:



George Boole



THE MATHEMATICAL ANALYSIS

OF LOGIC,

BEING AN ESSAY TOWARDS A CALCULUS
OF DEDUCTIVE REASONING.

BY GEORGE BOOLE.

'Επιστωνυοῦσι δὲ πᾶσαι αἱ ἐπιστῆμαι ἀλλήλαις κατὰ τὰ κοινά. Κοινὰ δὲ λέγω, οἷς χρῶνται ὡς ἐκ τούτων ἀποδεικνύντες· ἀλλ' οὐ περὶ ὧν δεικνύουσιν, οὐδὲ ἃ δεικνύουσι.

ARISTOTLE, *Anal. Post.*, lib. i. cap. xi.

CAMBRIDGE:
MACMILLAN, BARCLAY, & MACMILLAN;
LONDON: GEORGE BELL.

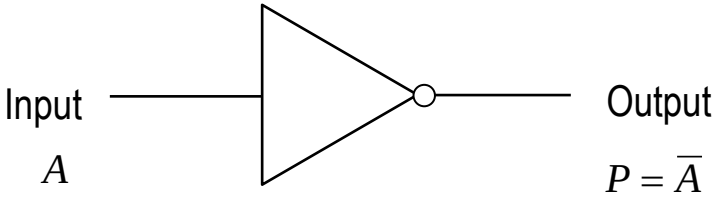
1847

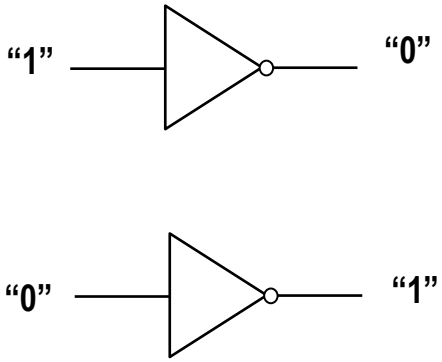
"True" = "1"

"False" = "0"

Binary bit operation: Simplest decision making

NOT Gate: Inverter

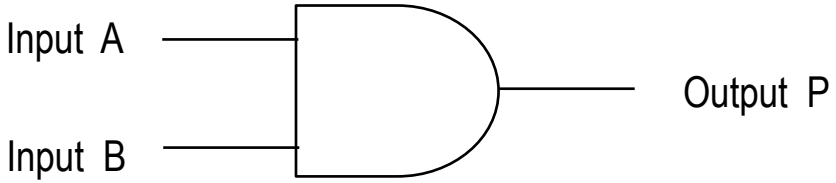


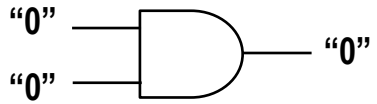


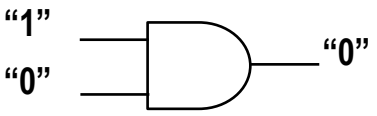
Truth Table $P = \overline{A}$

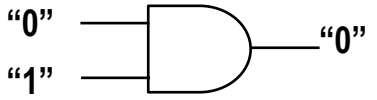
Input	Output
1	0
0	1

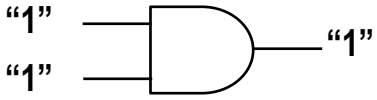
AND Gate









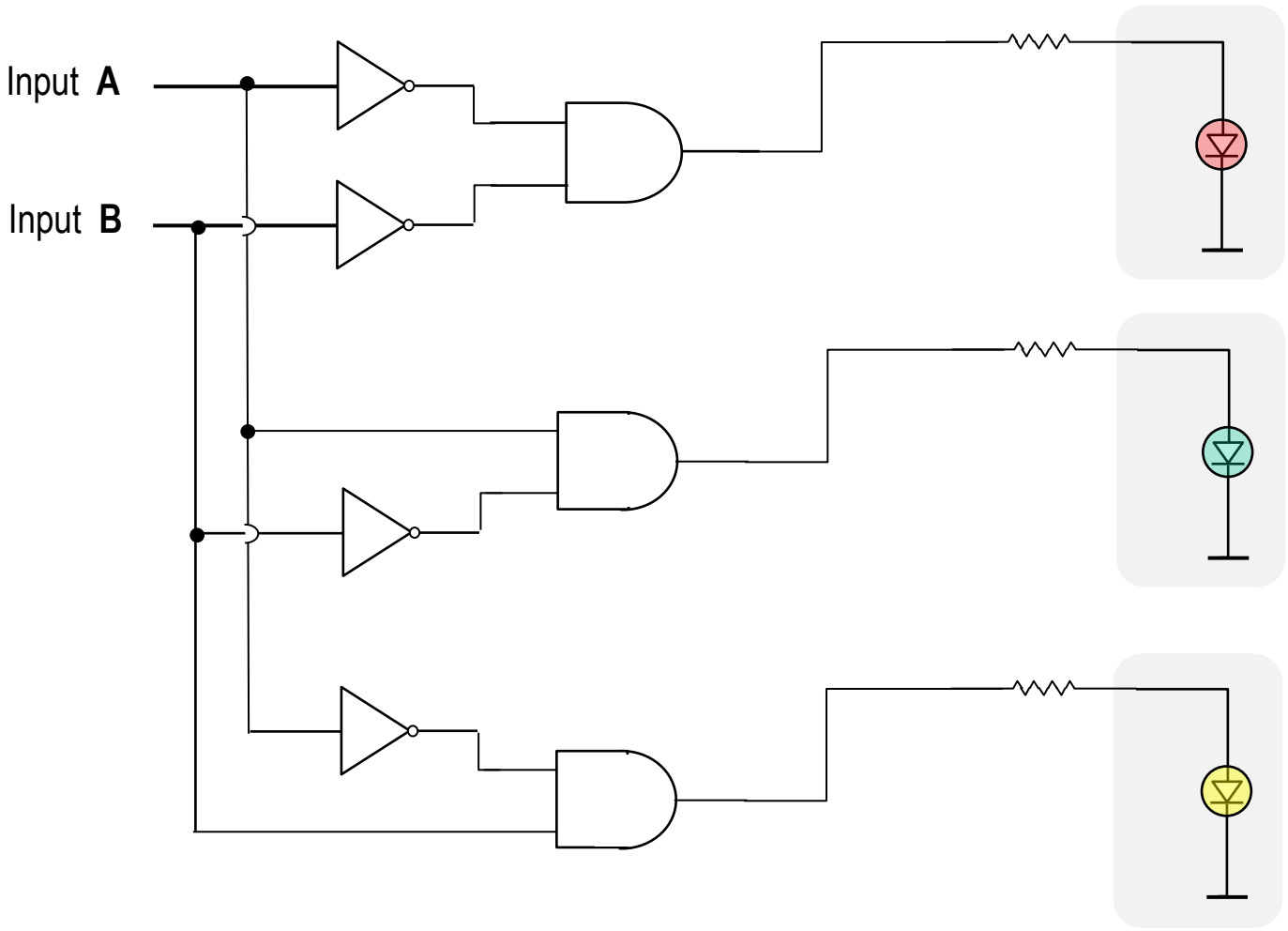


Truth Table $P = A \cdot B$

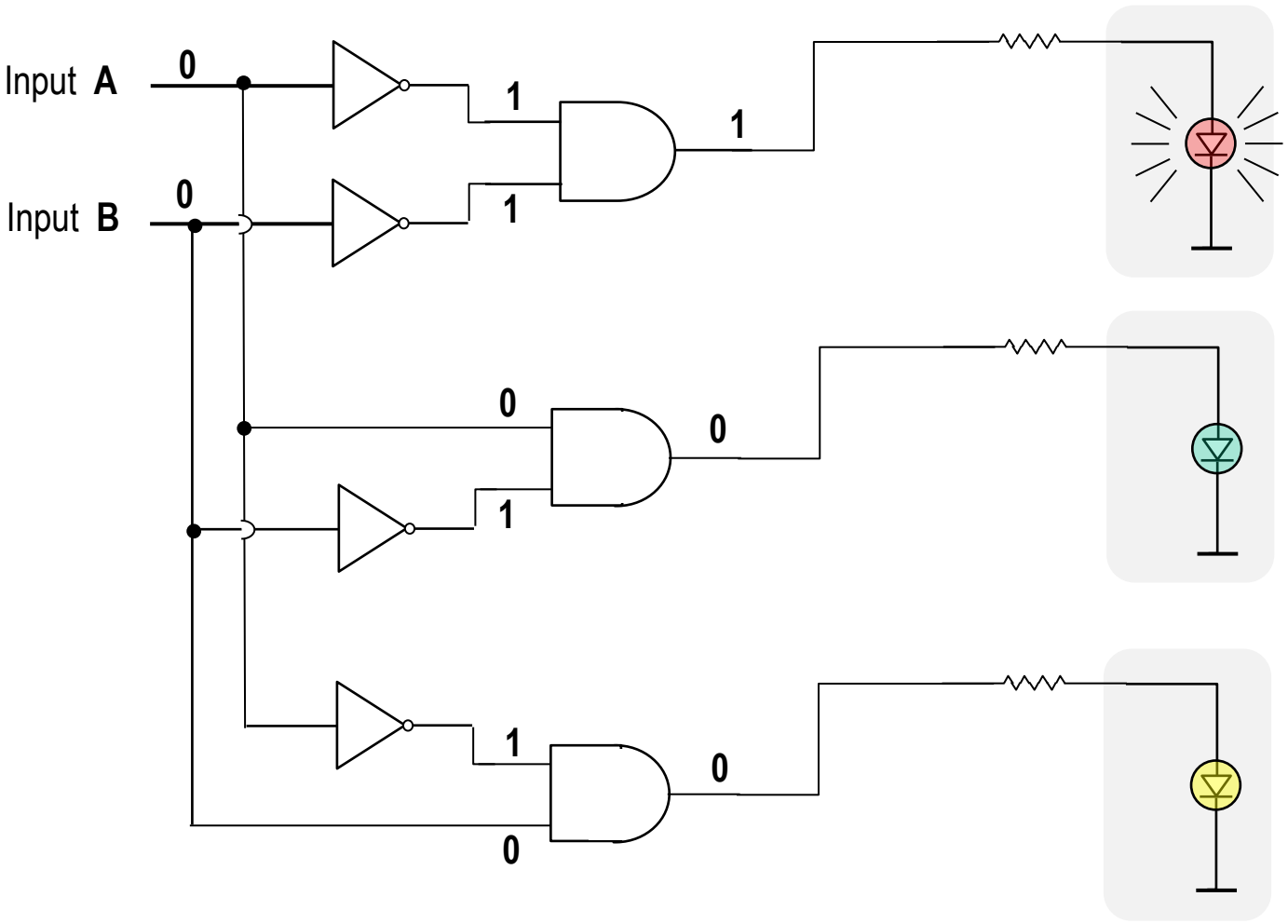
Input A	Input B	Output P
0	0	0
1	0	0
0	1	0
1	1	1

$$\begin{cases} 0 \cdot 0 = 0 \\ 1 \cdot 0 = 0 \\ 0 \cdot 1 = 0 \\ 1 \cdot 1 = 1 \end{cases}$$

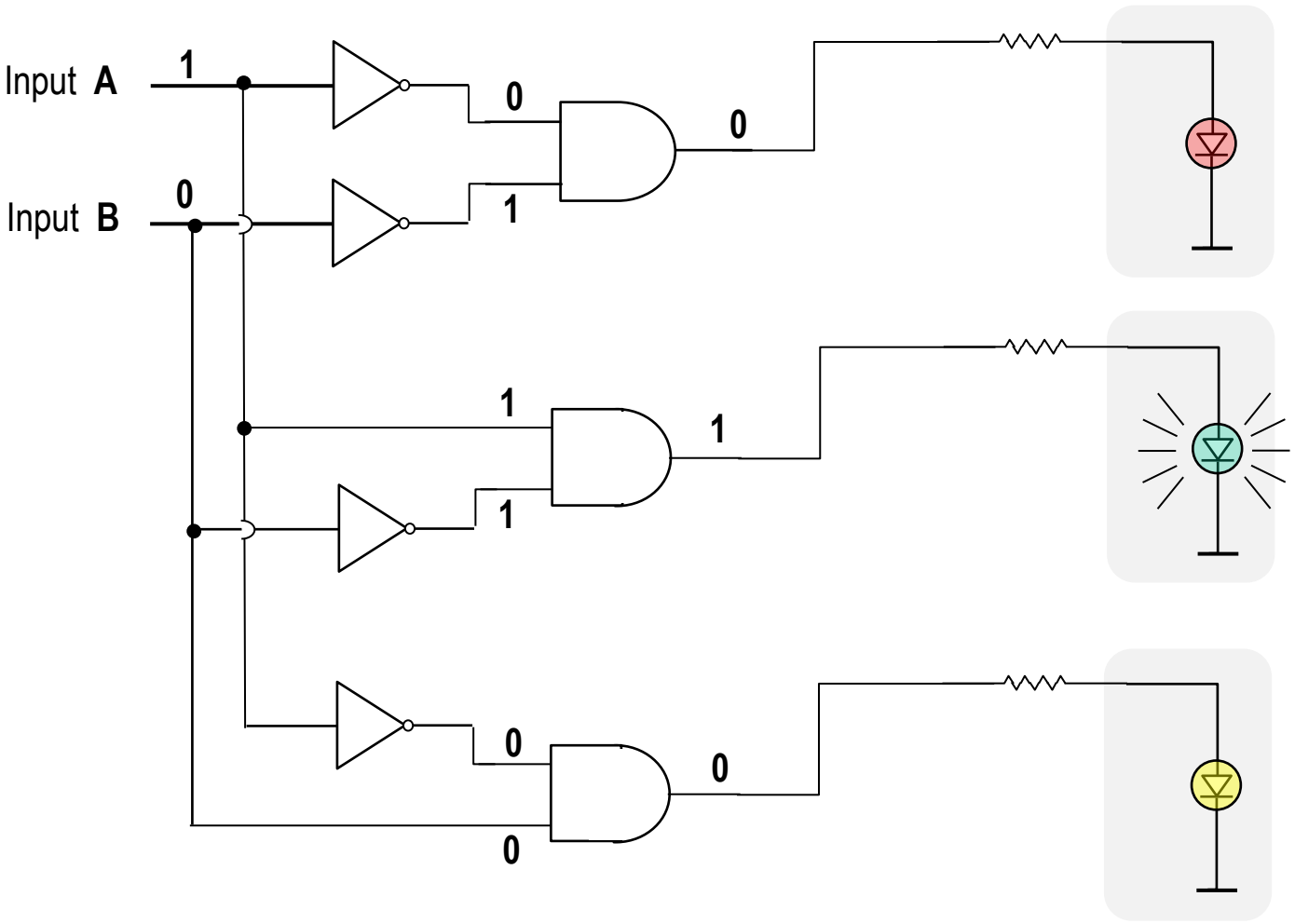
Using Logic Gates to Control Action: The Use of "1" and "0"



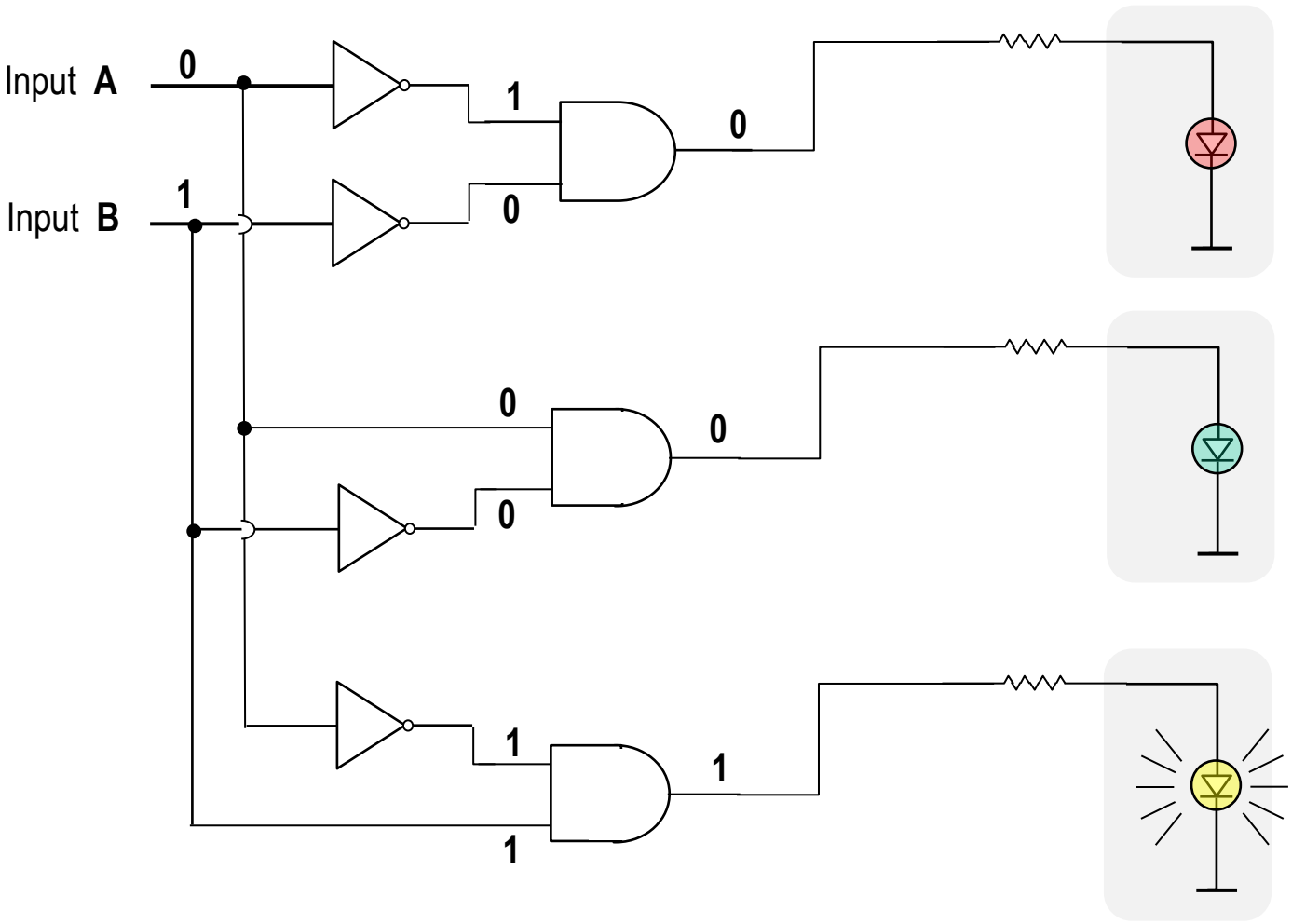
Using Logic Gates to Control Action: The Use of "1" and "0"



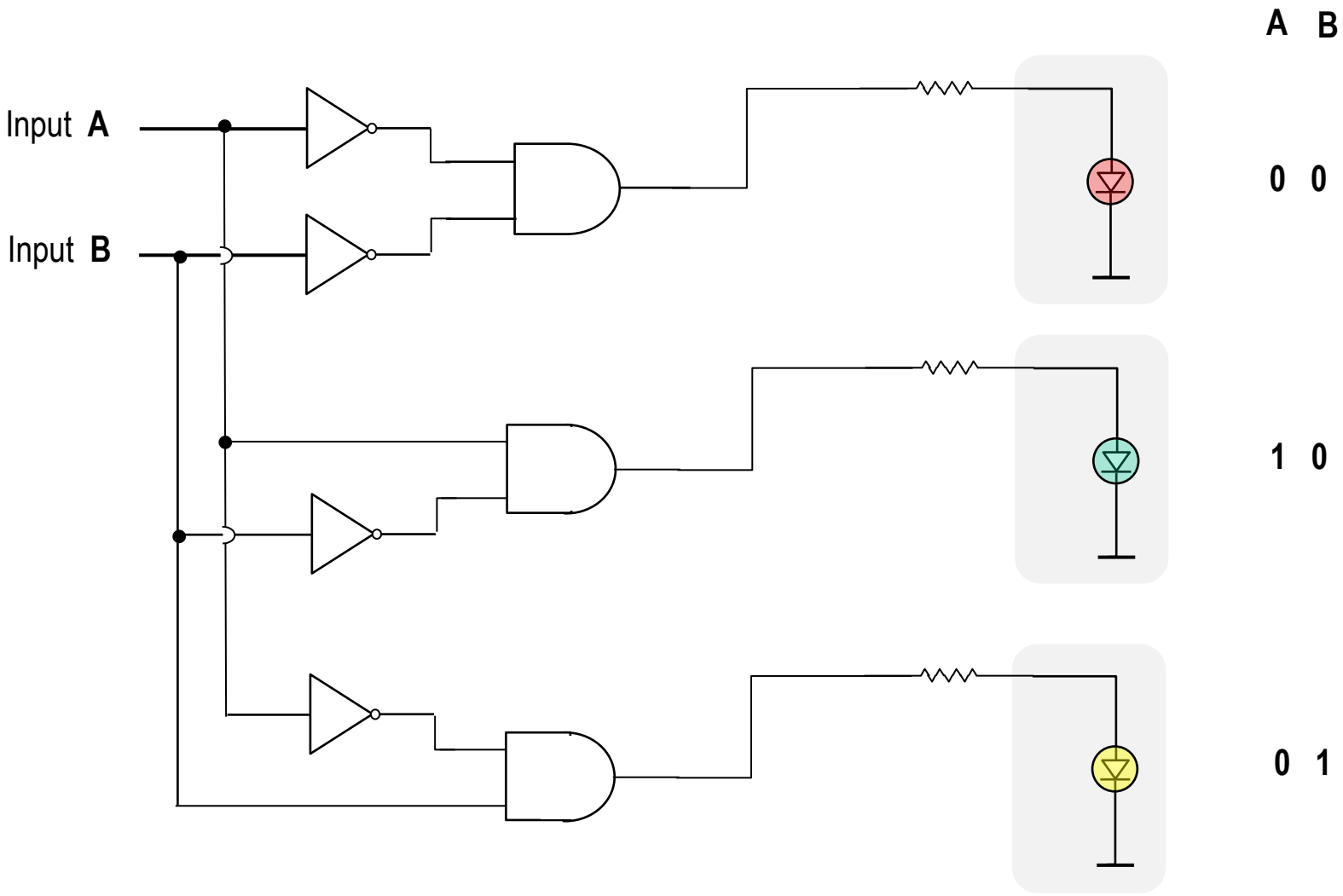
Using Logic Gates to Control Action: The Use of "1" and "0"



Using Logic Gates to Control Action: The Use of "1" and "0"



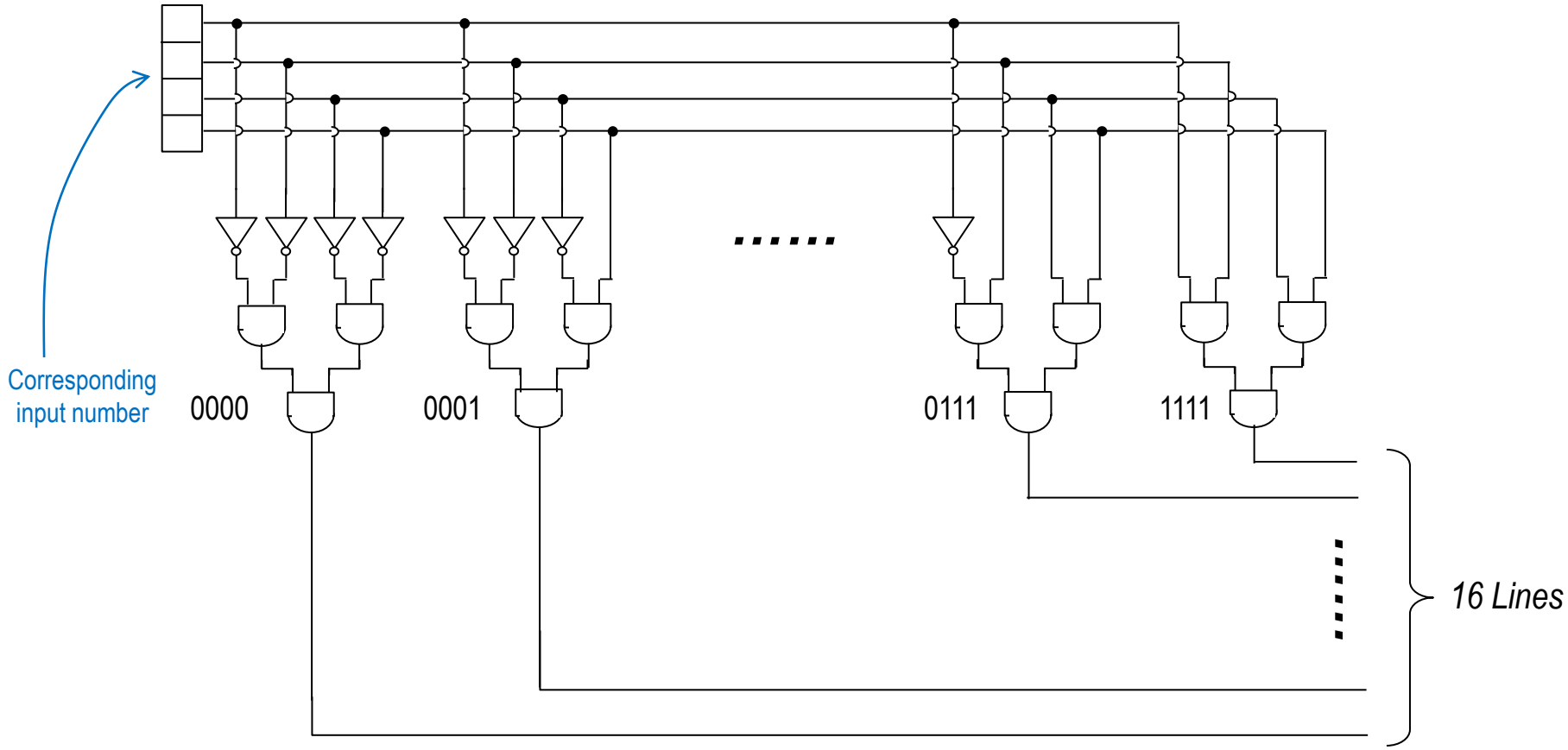
Using Logic Gates to Control Action: The Use of "1" and "0"



n-bit Input Controlling/Selecting 2^n Different Actions

A n-bit number has 2^n different values: Can be used to control 2^n different actions!

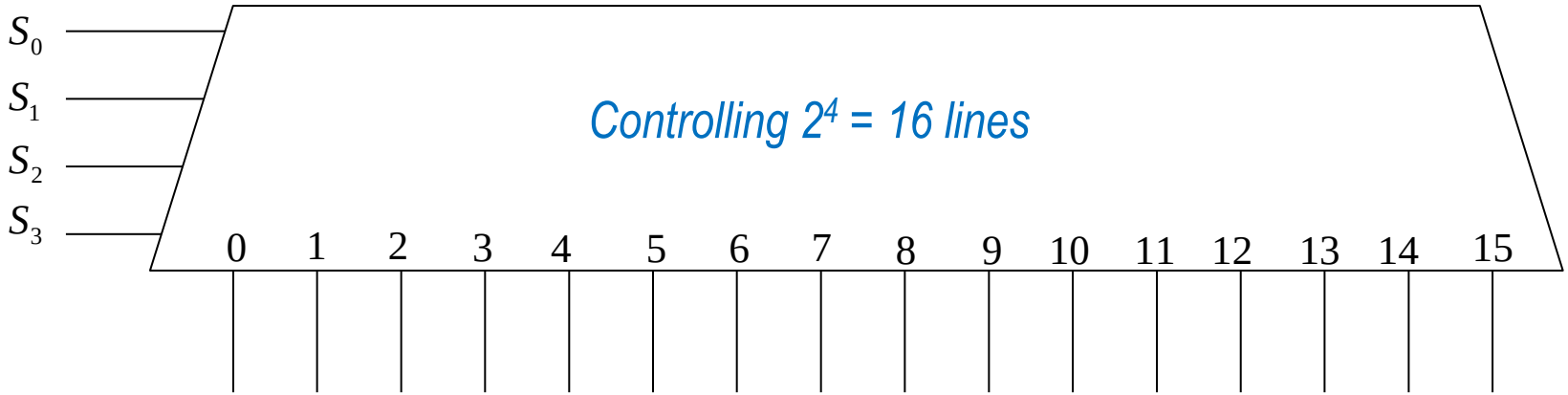
Example: 4-bit number controlling/selecting 16 lines



n-bit Input Controlling/Selecting 2^n Different Actions

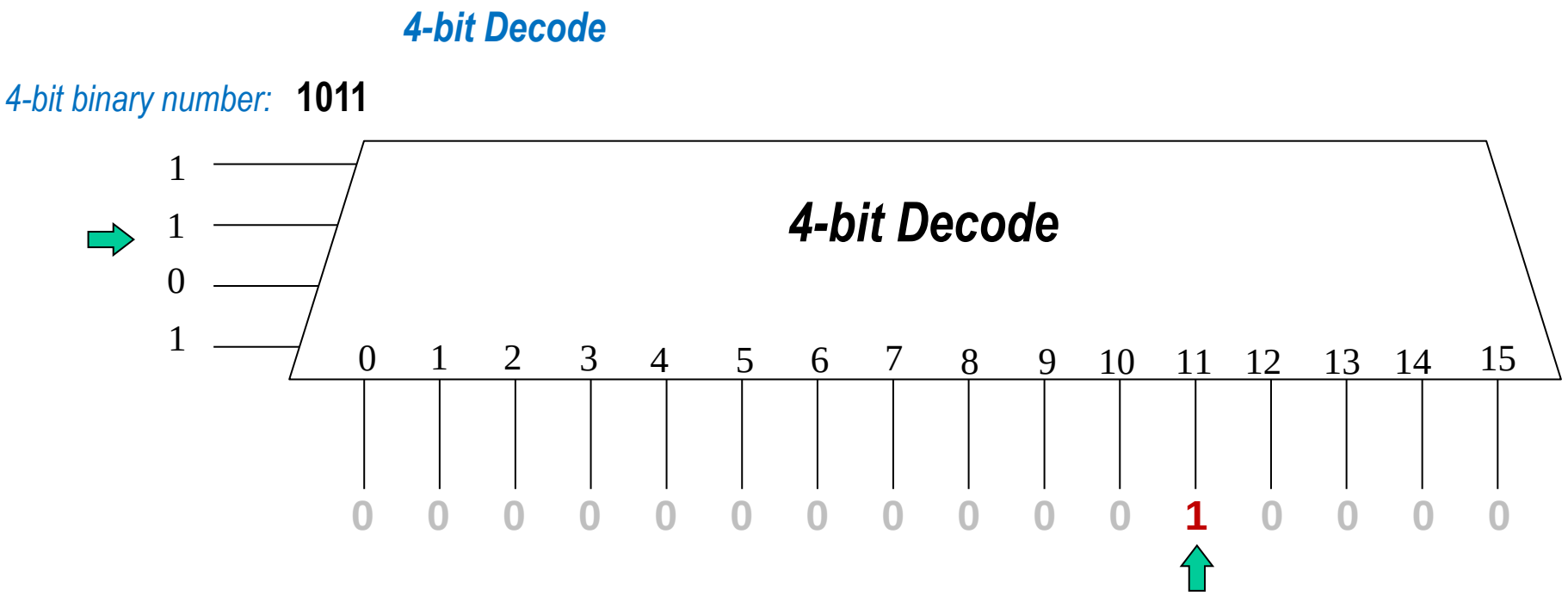
4-bit Decode

4-bit binary number



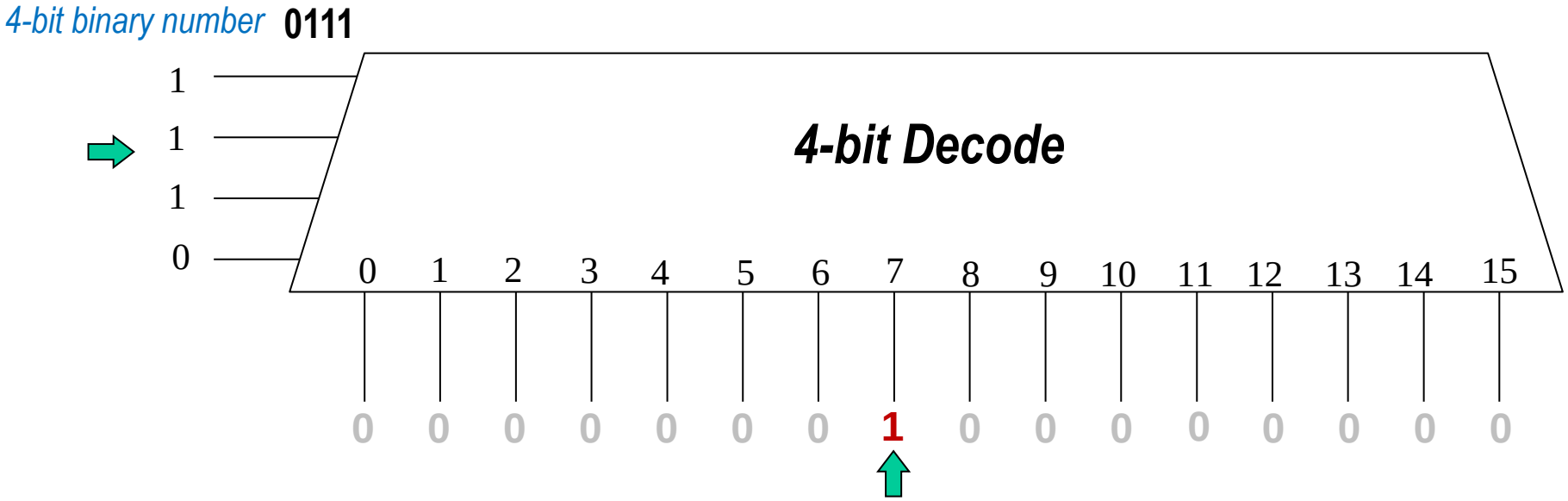
This is how machine(computer) uses binary numbers to perform different actions!

n-bit Input Controlling/Selecting 2^n Different Actions



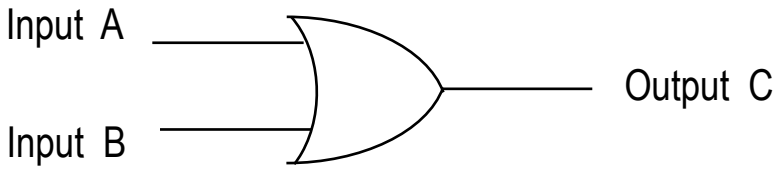
A 4-bit binary number selects a single line (out of the 16 lines) to be "HIGH".
(The rest all remain "LOW")

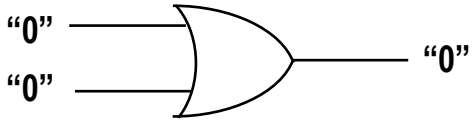
4-bit Input Controlling/Selecting 2^4 Different Actions

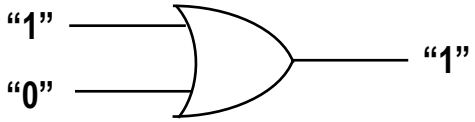


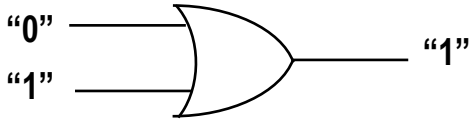
A 4-bit binary number selects a single line (out of the 16 lines) to be "HIGH".
(The rest all remain "LOW")

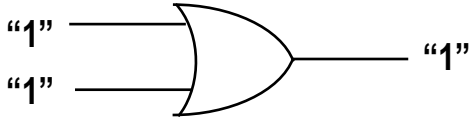
OR Gate









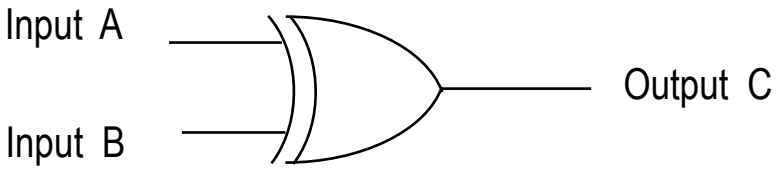


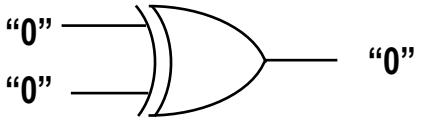
Truth Table $C = A + B$

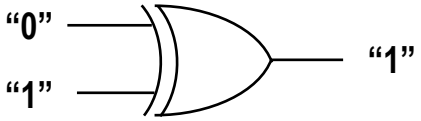
Input A	Input B	Output C
0	0	0
1	0	1
0	1	1
1	1	1

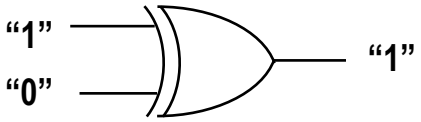
$$\begin{cases} 0+0=0 \\ 1+0=1 \\ 0+1=1 \\ 1+1=1 \end{cases}$$

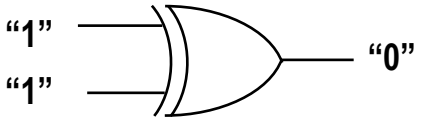
Exclusive OR Gate: XOR











Truth Table $C = A \oplus B$

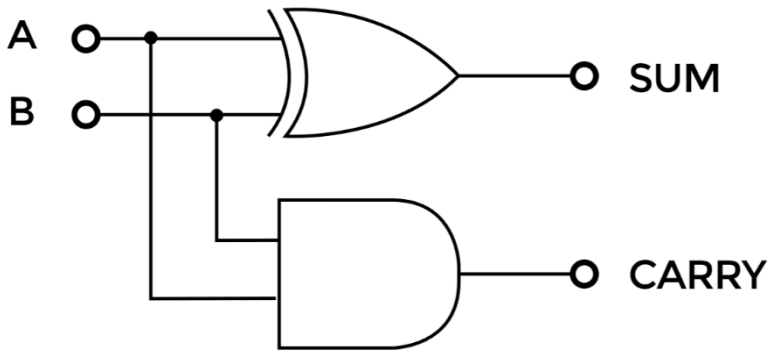
Input A	Input B	Output C	
0	0	0	$0 \oplus 0 = 0$
0	1	1	$1 \oplus 0 = 1$
1	0	1	$0 \oplus 1 = 1$
1	1	0	$1 \oplus 1 = 0$

Computing Using Logic Gates: Addition

Adding Binary Numbers for the Rightest Digit (No Input Carry)

A + B	0 + 0 = 0	1 + 0 = 1	0 + 1 = 1	1 + 1 = 0 (carry 1)

Half Adder (no input carry)



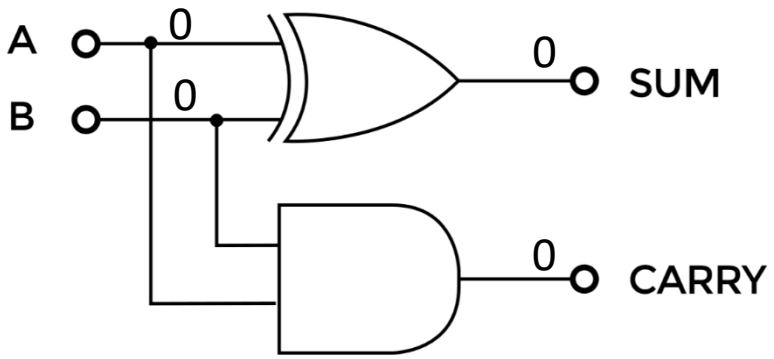
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Computing Using Logic Gates: Addition

Adding Binary Numbers for the Rightest Digit (No Input Carry)

A + B	0 + 0 = 0	1 + 0 = 1	0 + 1 = 1	1 + 1 = 0 (carry 1)

Half Adder (no input carry)



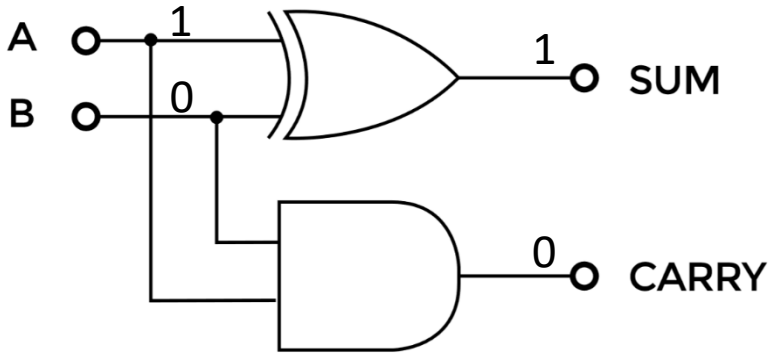
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Computing Using Logic Gates: Addition

Adding Binary Numbers for the Rightest Digit (No Input Carry)

A + B	0 + 0 = 0	1 + 0 = 1	0 + 1 = 1	1 + 1 = 0 (carry 1)

Half Adder (no input carry)



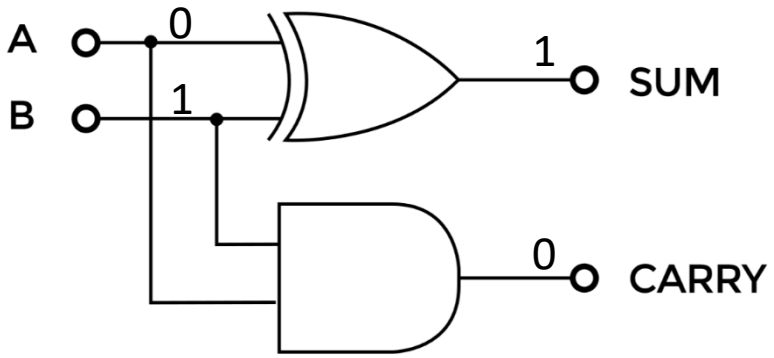
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Computing Using Logic Gates: Addition

Adding Binary Numbers for the Rightest Digit (No Input Carry)

A + B	0 + 0 = 0	1 + 0 = 1	0 + 1 = 1	1 + 1 = 0 (carry 1)

Half Adder (no input carry)



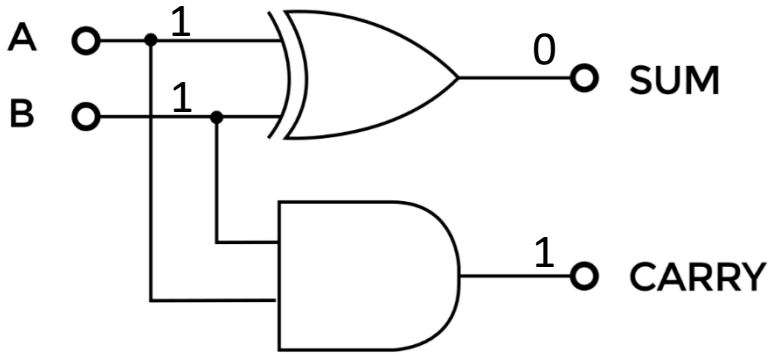
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Computing Using Logic Gates: Addition

Adding Binary Numbers for the Rightest Digit (No Input Carry)

A + B	0 + 0 = 0	1 + 0 = 1	0 + 1 = 1	1 + 1 = 0 (carry 1)

Half Adder (no input carry)



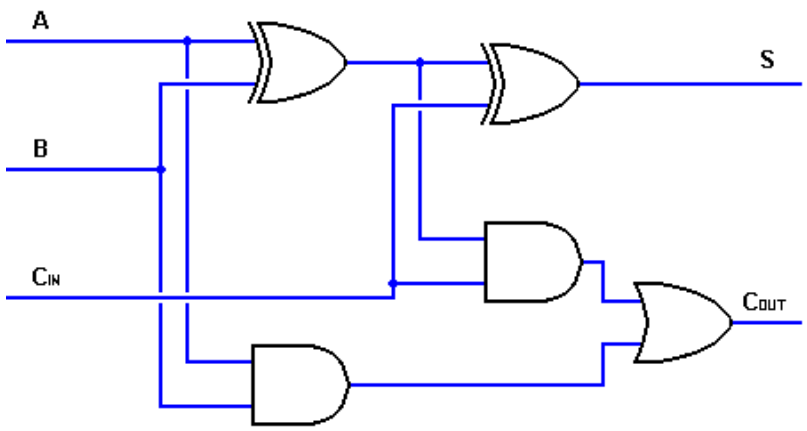
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Addition

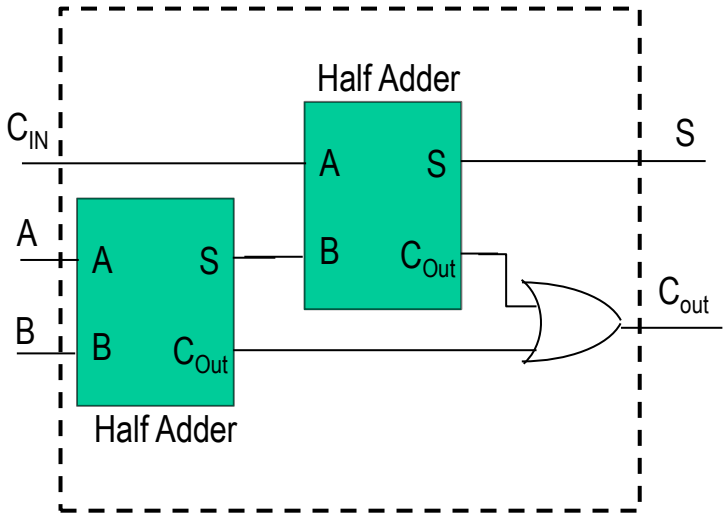
Addition with input carry:

$$\begin{array}{r} 0 \\ +\ 0\ 1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\ 1\ 0\ 1 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ +\ 1\ 1\ 1 \\ \hline 0 \end{array} \qquad \begin{array}{r} 1 \\ +\ 1\ 1\ 1 \\ \hline 1 \end{array}$$

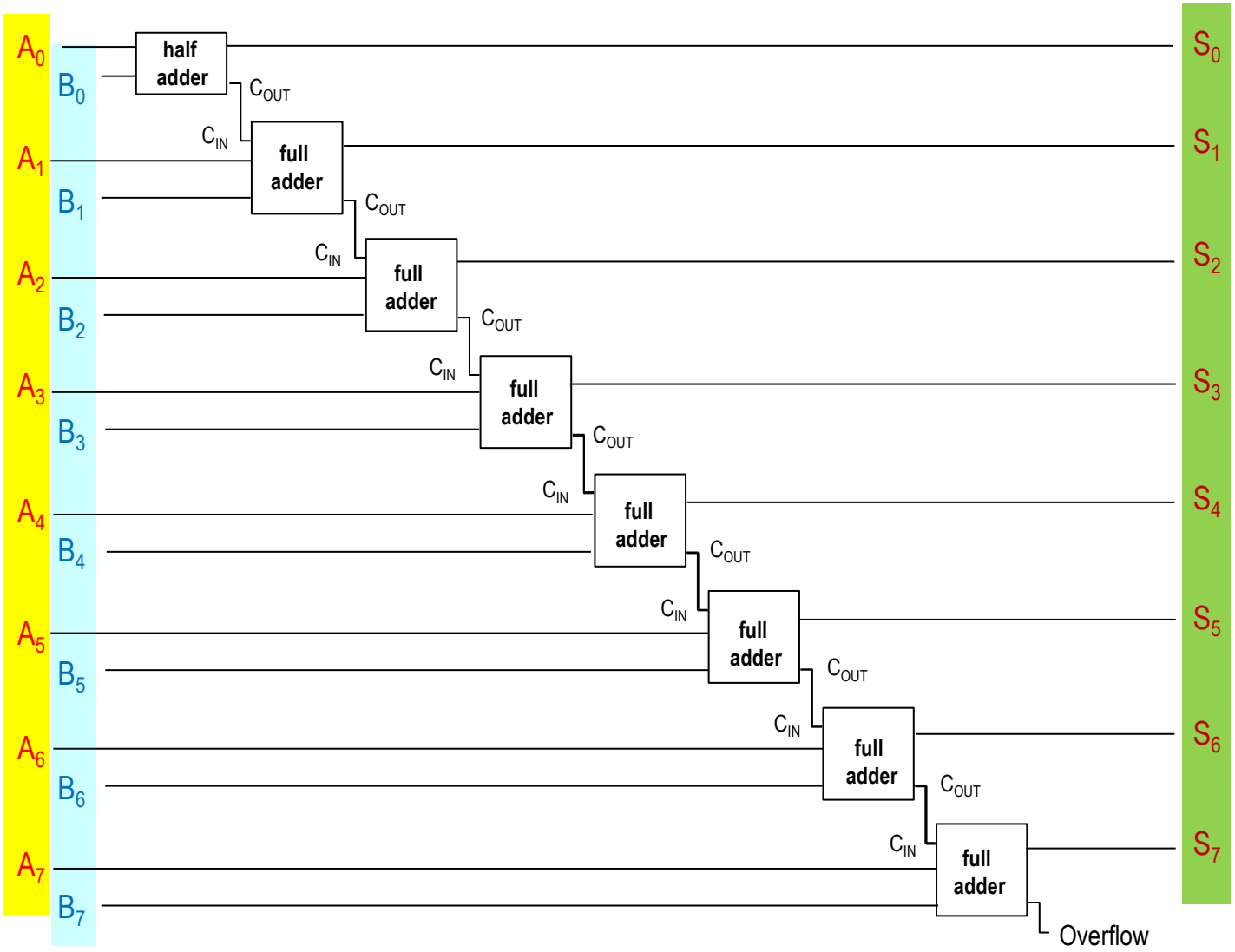
Full Adder (with input carry)



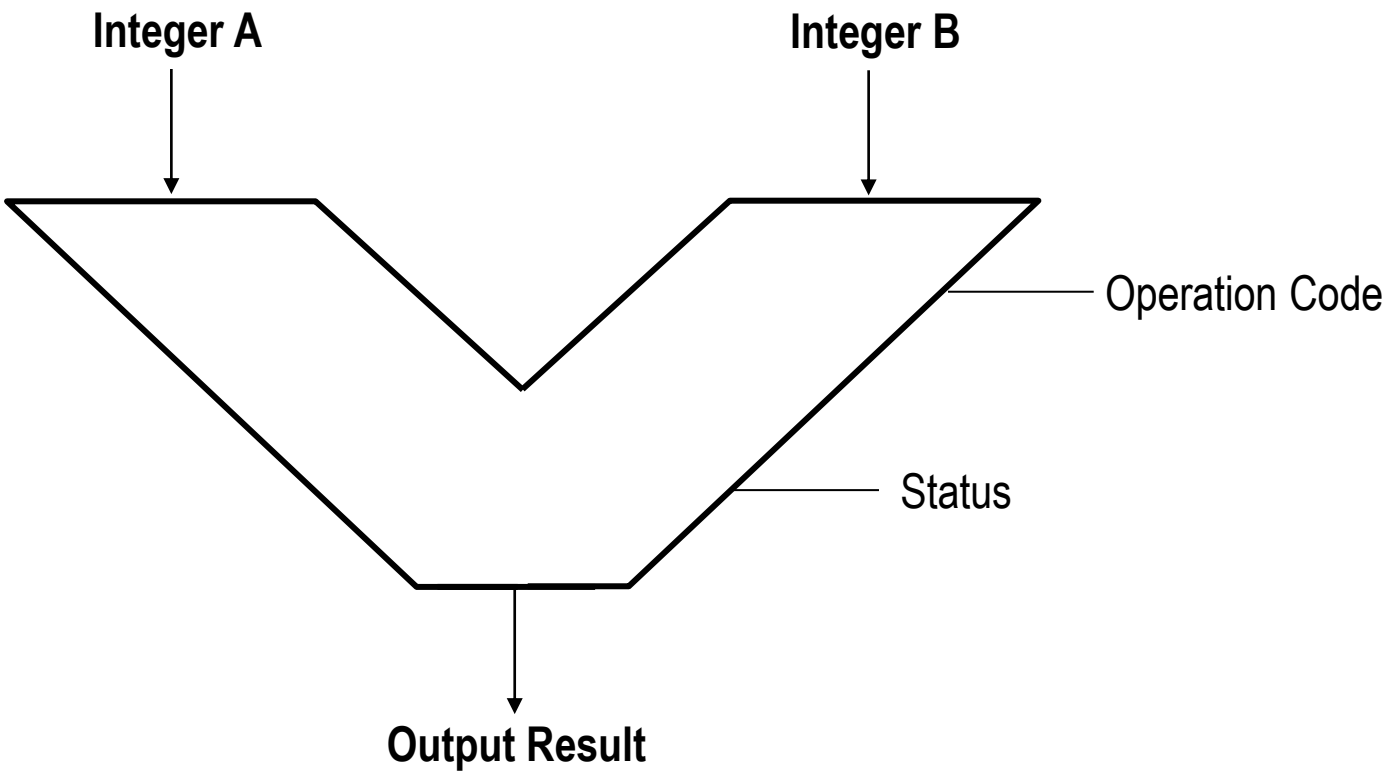
Full Adder



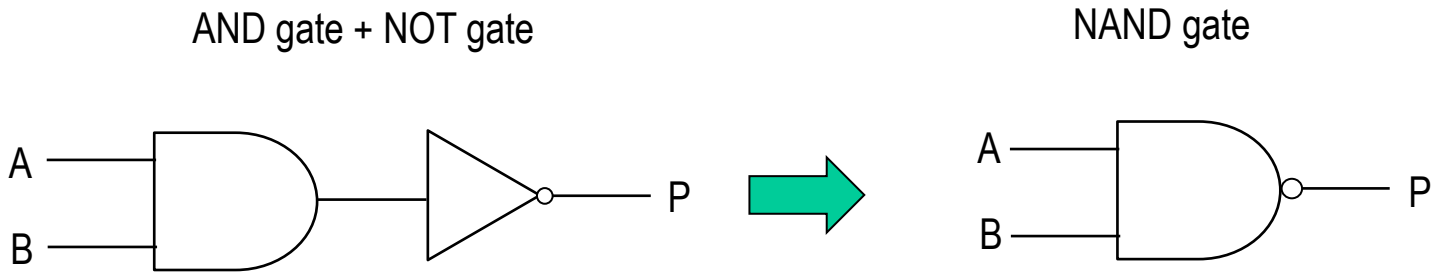
8-bit Adder

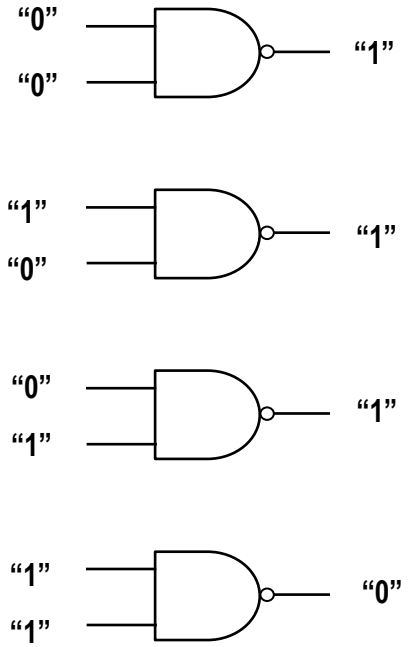

A0
B0
A1
B1
A2
B2
A3
B3
A4
B4
A5
B5
A6
B6
A7
B7
half adder
full adder
full adder
full adder
full adder
full adder
full adder
full adder
COUT
CIN
S0
S1
S2
S3
S4
S5
S6
S7
Overflow

Arithmetic Logic Unit (ALU)



NAND Gate :



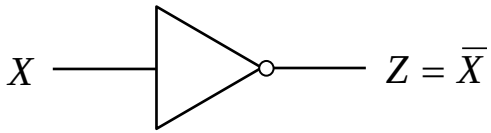


Truth Table $P = \overline{A \cdot B}$

Input A	Input B	Output P
0	0	1
1	0	1
0	1	1
1	1	0

$$\overline{0 \cdot 0} = \overline{0} = 1$$
$$\overline{1 \cdot 0} = \overline{0} = 1$$
$$\overline{0 \cdot 1} = \overline{0} = 1$$
$$\overline{1 \cdot 1} = \overline{1} = 0$$

Boolean Algebra

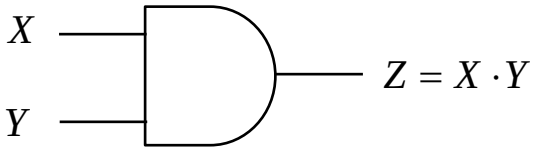


$$\begin{cases} X = 1 \Rightarrow Z = 0 \\ X = 0 \Rightarrow Z = 1 \end{cases}$$

Rules

X and $\overline{X}$ are complements

$$\overline{\overline{X}} = X$$

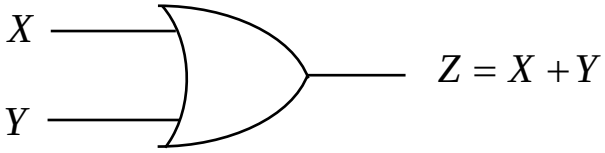


$$\begin{cases} 0 \cdot 0 = 0 \\ 1 \cdot 0 = 0 \\ 0 \cdot 1 = 0 \\ 1 \cdot 1 = 1 \end{cases}$$

$$0 \cdot X = 0$$

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

$$1 \cdot X = X$$



$$\begin{cases} 0 + 0 = 0 \\ 1 + 0 = 1 \\ 0 + 1 = 1 \\ 1 + 1 = 1 \end{cases}$$

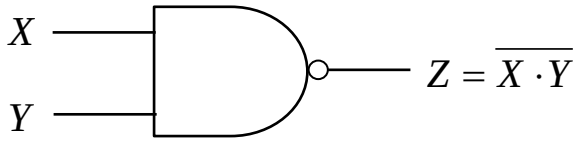
$$0 + X = X$$

$$1 + X = 1$$

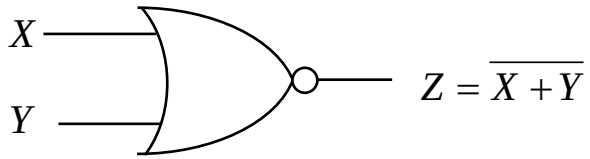
$$X + \overline{X} = 1$$

$$X + Y = Y + X$$

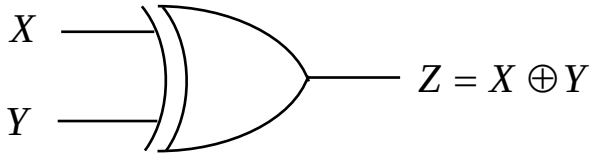
Boolean Algebra



$$\begin{cases} \overline{0 \cdot 0} = \bar{0} = 1 \\ \overline{1 \cdot 0} = \bar{0} = 1 \\ \overline{0 \cdot 1} = \bar{0} = 1 \\ \overline{1 \cdot 1} = \bar{1} = 0 \end{cases}$$



$$\begin{cases} \overline{0 + 0} = \bar{0} = 1 \\ \overline{1 + 0} = \bar{1} = 0 \\ \overline{0 + 1} = \bar{1} = 0 \\ \overline{1 + 1} = \bar{1} = 0 \end{cases}$$



$$\begin{cases} 0 \oplus 0 = 0 \\ 1 \oplus 0 = 1 \\ 0 \oplus 1 = 1 \\ 1 \oplus 1 = 0 \end{cases}$$

Boolean Algebra Example — Fail-Safe Autopilot Logic

Prior to takeoff or landing maneuver, a commercial aircraft requires the following check:

2** of the possible **3 pilots** must be available**: the pilot, the copilot and the autopilot

Set Logic Variables:

X – State of the pilot: 1= Present, 0=Absent

Y – State of the copilot: 1= Present, 0=Absent

Z – State of the autopilot: 1= Functioning, 0=Not Functioning

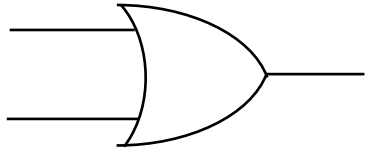
Logic function corresponding to "System Ready" is

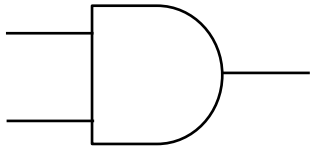
$$f = X \cdot Y + X \cdot Z + Y \cdot Z$$

$f = 1$: *System Ready ;* $f = 0$: *System Not Ready*

Rules for Boolean Algebra

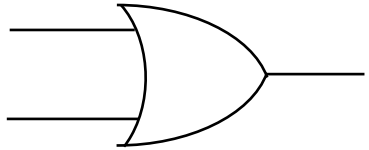
1. $0 + X = X$
2. $1 + X = 1$
3. $X + X = X$
4. $X + \overline{X} = 1$
5. $0 \cdot X = 0$
6. $1 \cdot X = X$
7. $X \cdot X = X$
8. $X \cdot \overline{X} = 0$
9. $\overline{\overline{X}} = X$
10. $X + Y = Y + X$ Commutative Law
11. $X \cdot Y = Y \cdot X$ Commutative Law
12. $X + (Y + Z) = (X + Y) + Z$
13. $X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$
14. $X \cdot (Y + Z) = X \cdot Y + X \cdot Z$ Distributive Law
15. $X + X \cdot Z = X$ Absorption Law

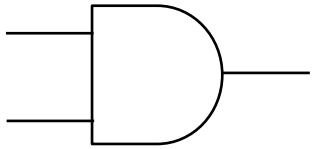




Rules for Boolean Algebra

1. $0 + X = X$
2. $1 + X = 1$
3. $X + X = X$
4. $X + \overline{X} = 1$
5. $0 \cdot X = 0$
6. $1 \cdot X = X$
7. $X \cdot X = X$
8. $X \cdot \overline{X} = 0$
9. $\overline{\overline{X}} = X$
10. $X + Y = Y + X$ Commutative Law
11. $X \cdot Y = Y \cdot X$ Commutative Law
12. $X + (Y + Z) = (X + Y) + Z$
13. $X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$
14. $X \cdot (Y + Z) = X \cdot Y + X \cdot Z$ Distributive Law
15. $X + X \cdot Z = X$ Absorption Law





De Morgan's Laws

$$\overline{(X + Y)} = \overline{X} \cdot \overline{Y}$$

$$\overline{(X \cdot Y)} = \overline{X} + \overline{Y}$$

- Any logic function can be implemented using only **OR** and **NOT** gates.
- Any logic function can be implemented using only **AND** and **NOT** gates.

Boolean Algebra Example — Fail-Safe Autopilot Logic

Positive check

$$f = X \cdot Y \;+\; X \cdot Z \;+\; Y \cdot Z$$ *sum-of-product*

System Not Ready Condition:

$$\overline{f} = \overline{X \cdot Y + X \cdot Z + Y \cdot Z}$$

$$= \overline{(X \cdot Y)} \cdot \overline{(X \cdot Z)} \cdot \overline{(Y \cdot Z)} \qquad \overline{(X+Y)} = \overline{X} \cdot \overline{Y}$$

$$= (\overline{X} + \overline{Y}) \cdot (\overline{X} + \overline{Z}) \cdot (\overline{Y} + \overline{Z}) \qquad \overline{(X \cdot Y)} = \overline{X} + \overline{Y}$$

$$\overline{f} = (\overline{X} + \overline{Y}) \cdot (\overline{X} + \overline{Z}) \cdot (\overline{Y} + \overline{Z})$$