

# CS222: Computer Architecture

Instructor:

Dr Ahmed Shalaby <http://bu.edu.eg/staff/ahmedshalaby14#>

الاحترام - الادب - الاخلاق  
الطالب - المعيد - الدكتور



[Benha University](#)

[Home](#)

[النسخة العربية](#)

[My C.V.](#)

[About](#)

[Courses](#)

[Publications](#)

[Inlinks\(Competition\)](#)

[Theses](#)

[Reports](#)

[Published books](#)

[Workshops / Conferences](#)

[Supervised PhD](#)

[Supervised MSc](#)

[Supervised Projects](#)

[Education](#)

[Language skills](#)

[Academic Positions](#)

[Administrative Positions](#)

[Memberships and awards](#)

[Committees](#)

[Scientific Activities](#)

[Experience](#)

[Outgoing Links](#)

[News](#)

[Photo Gallery](#)

[Staff Statement](#)

You are in: [Home](#)

## Dr. Ahmed Shalaby

**Academic Position:** Asst. Professor

**Current Administrative Position:**

**Ex-Administrative Position:**

**Faculty:** Computers and Artificial Intelligence

**Department:** Computer Science

**Edu-Mail:** ahmed.shalaby@fci.bu.edu.eg

**Alternative Email:** ahmed.shalaby@ejust.edu.eg

**Mobile:**

**Scientific Name:** Ahmed Shalaby

**Publications [ Titles(9) :: Papers(3) :: Abstracts(9) ]**

**Courses Files( 52)**

**Inlinks: (0)**

**External links: (29)**

**News(edit)**

**Great Teams: Course: Logic Design: Projects: Elevator [2019-03-06]**

<https://www.youtube.com/watch?v=s76AcBLlyCc&feature=youtu.be>  
[more](#)

**Research Interests(edit)**

Security, System on Chip, Network on Chip, VLSI, Embedded System, High Efficiency Video Coding (HEVC)

**Selected Publications(edit)**

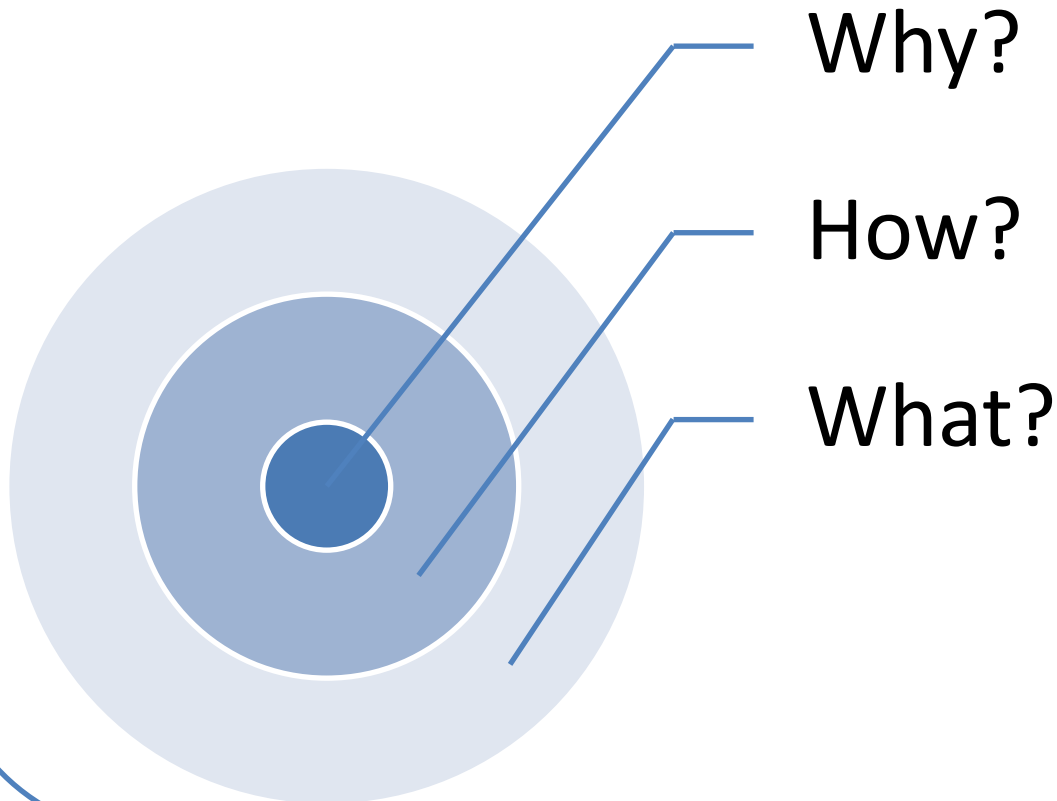
Automatic arrival time detection for earthquakes based on Modified Laplacian of Gaussian filter

Automatic discrimination of earthquakes and quarry blasts using wavelet filter bank and support vector machine

Efficient Low-Power Digital Baseband Transceiver for IEEE 802.15.6 Narrowband Physical Layer



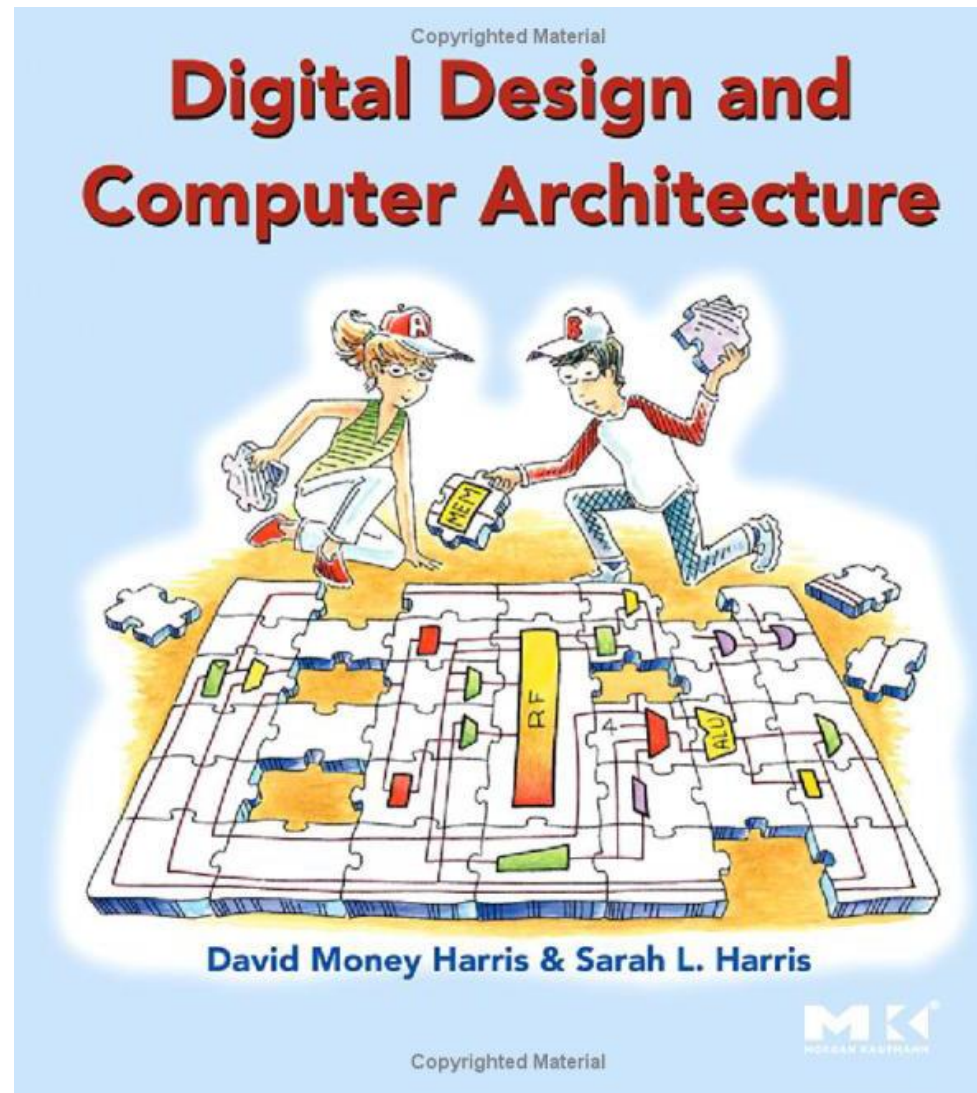
# Study: CS222: Computer Architecture



# What ? Computer Architecture

- **computer architecture** defines **how to command a processor**.
- **computer architecture** is a set of rules and methods that describe **the functionality, organization, and implementation** of computer system.

# How ? Course Book





# How ? Course Content

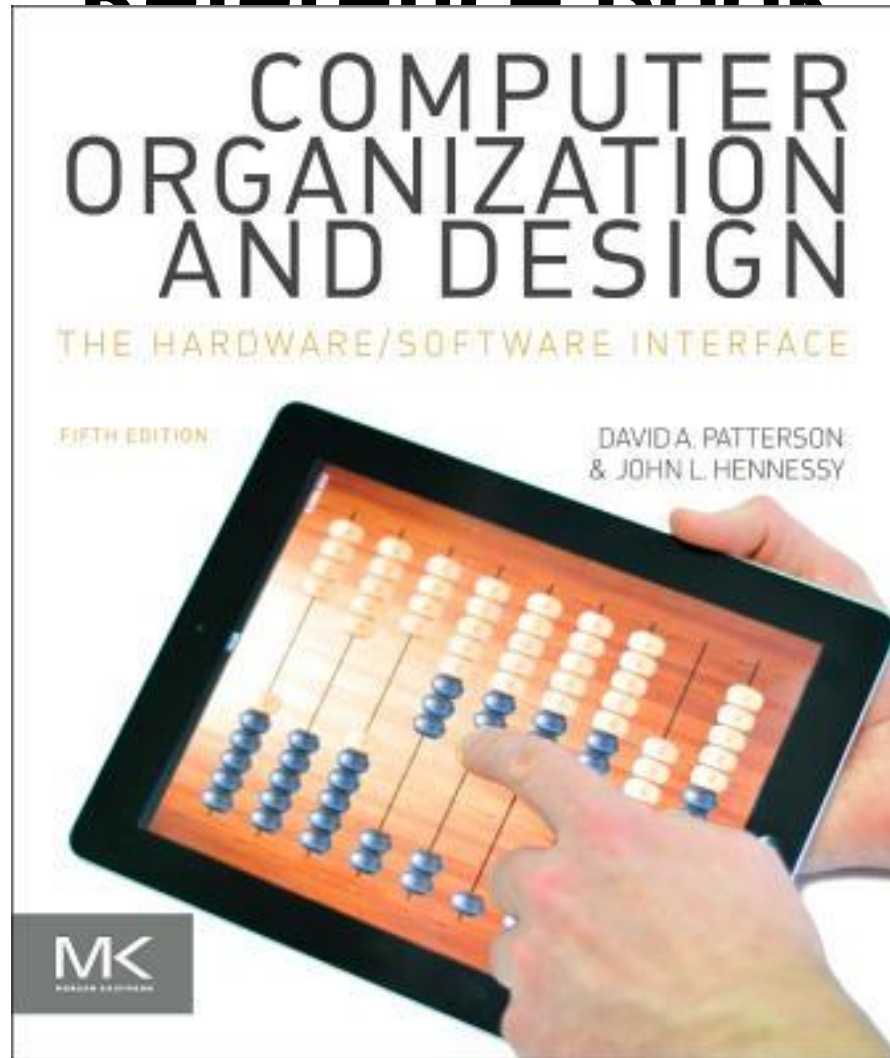
Lec #	Subject	Week #
Lec1	Chapter 1: From Zero to One	Week #1
Lec2	Chapter 2: Combinational Logic Design	Week #2
Lec 3	Chapter 3: Sequential Logic Design	Week #3
Lec 4	Chapter 4: Hardware Description Language	Week #4
Lec 5	Chapter 4 : continue	Week #5
Lec 6	Chapter 5: Digital Building Blocks	Week #6
Lec 7	Chapter 5 continue	Week #7
	Midterm Exam	Week #8
Lec 8	Chapter 6: Computer Architecture	Week #9
Lec 9	Chapter 6 : continue	Week #10
Lec 10	Chapter 7: Microarchitecture	Week #11
Lec 11	Chapter 7 continue	Week #12
Lec 12	Chapter 7 continue	Week #13

# Assessment

Final-Term Exam	50
Mid-Term Exam + lab Exam + Oral Exam + Projects (Verilog – ModelSim + Quartus)	
• logic design Project in Verilog – the week after midterm (Lab)	
• final project -> lab exam	50

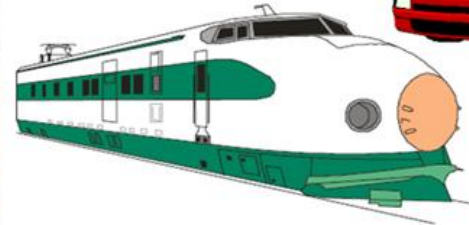
**Chapter 8 is a self study - In Exam.**

# Reference Book

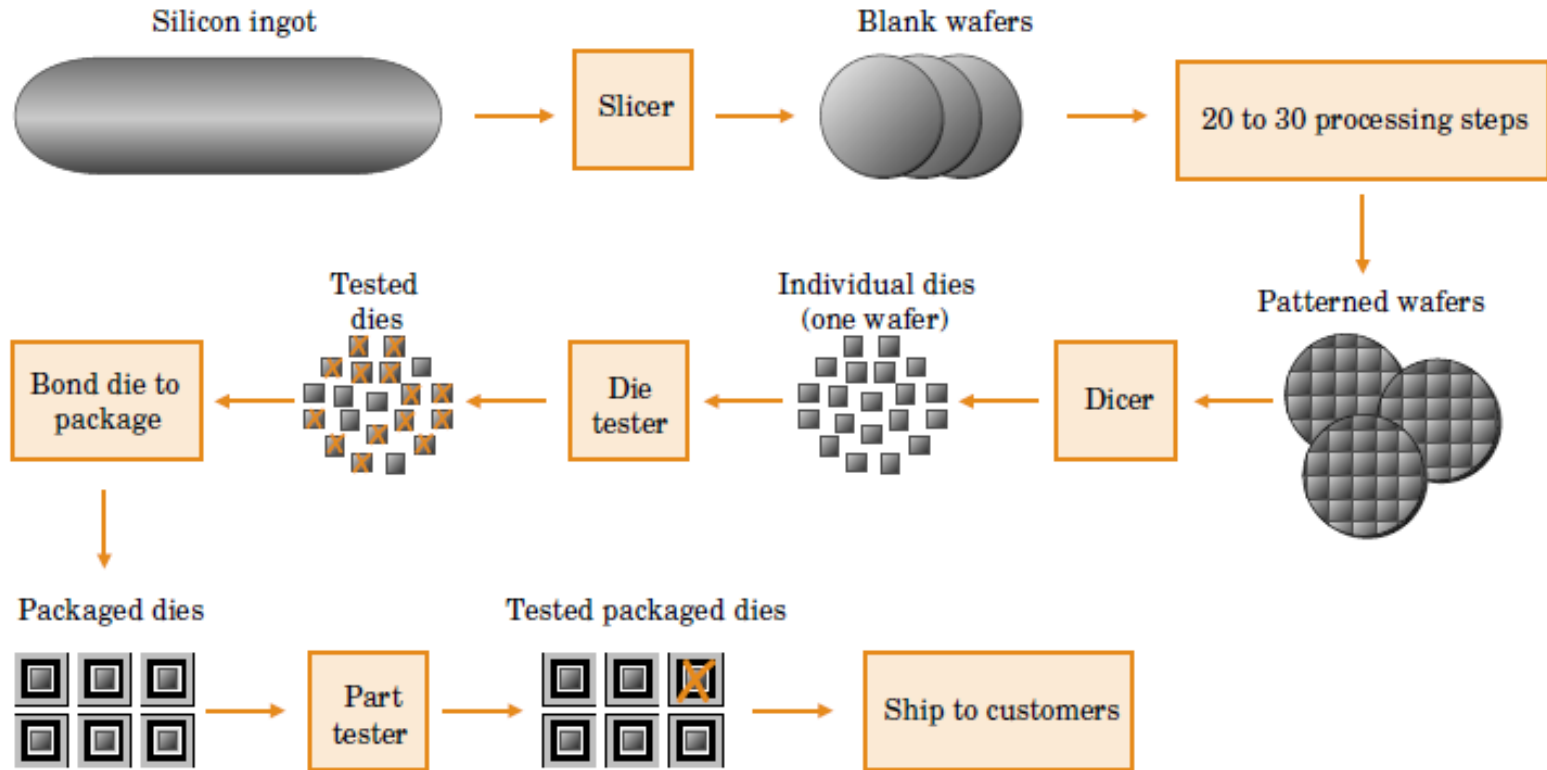




# Why ? Computer Architecture



# How ? Computers Architecture



## Silicon Industry

# Computer Architecture



# Chapter 1

## ***Digital Design and Computer Architecture, 2<sup>nd</sup> Edition***

---

David Money Harris and Sarah L. Harris

FROM ZERO TO ONE  
FROM ZERO TO ONE  
FROM ZERO TO ONE  
FROM ZERO TO ONE  
FROM ZERO TO ONE



# Chapter 1 :: Topics

- **Background**
- **The Game Plan**
- **The Art of Managing Complexity**
- **The Digital Abstraction**
- **Number Systems**
- **Logic Gates**
- **Logic Levels**
- **CMOS Transistors**
- **Power Consumption**

# Background

- Microprocessors have revolutionized our world
  - Cell phones, Internet, rapid advances in medicine, etc.
- The semiconductor industry has grown from \$21 billion in 1985 to \$300 billion in 2011



FROM ZERO TO ONE  
FROM ZERO TO ONE  
FROM ZERO TO ONE



# The Game Plan

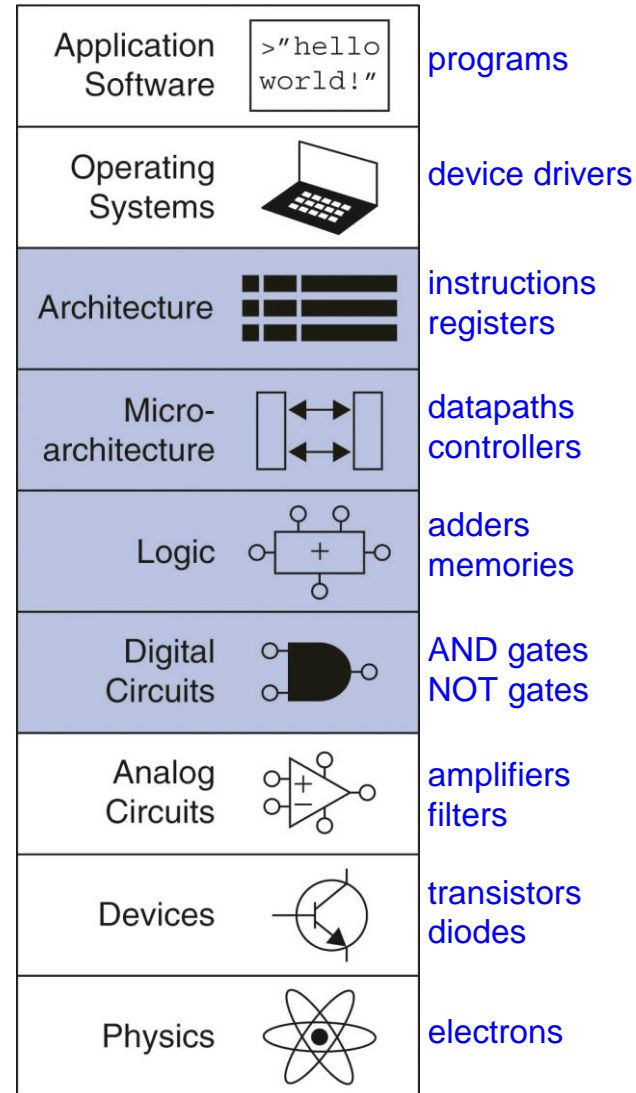
- Purpose of the course:
  - Understand what's under the hood of a computer
  - Learn the principles of digital design
  - Design and build a microprocessor

# The Art of Managing Complexity

- Abstraction
- Discipline
- The Three –y's
  - Hierarchy
  - Modularity
  - Regularity

# Abstraction

- Hiding details when they aren't important



# Discipline

- Intentionally restrict design choices
- Example: Digital discipline
  - Discrete voltages instead of continuous (Analog to Digital Converters)
  - Simpler to design than analog circuits – can build more sophisticated systems
  - Digital systems replacing analog predecessors:
    - i.e., digital cameras, digital television, cell phones, CDs

# The Three -y's

- **Hierarchy**
  - A system divided into modules and submodules
- **Modularity**
  - Having well-defined functions and interfaces
- **Regularity**
  - Encouraging uniformity, so modules can be easily reused

# The Digital Abstraction

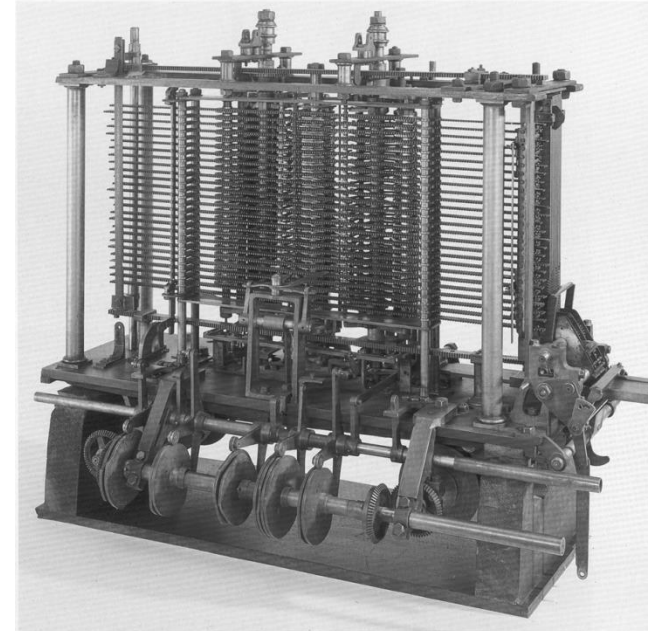
- Most physical variables are **continuous**
  - Voltage on a wire
  - Frequency of an oscillation
  - Position of a mass
- Digital abstraction considers **discrete subset** of values



# The Analytical Engine

- Designed by Charles Babbage from 1834 – 1871
- Considered to be the first digital computer
- Built from mechanical gears, where each gear represented a discrete value (0-9)

Quantum Computing

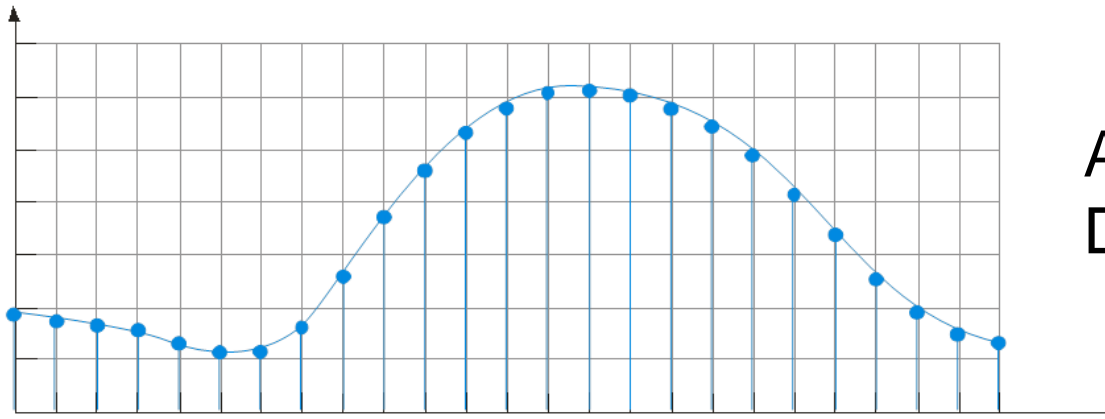


# Digital Discipline: Binary Values

- **Two discrete values:**
  - 1's and 0's
  - 1, TRUE, HIGH
  - 0, FALSE, LOW
- **1 and 0:** voltage levels, rotating gears, fluid levels, etc.
- Digital circuits use **voltage** levels to represent 1 and 0
- ***Bit:*** Binary digit

# Digital Discipline

Most natural quantities (such as temperature, pressure, light intensity, ...) are **analog** quantities that vary continuously.

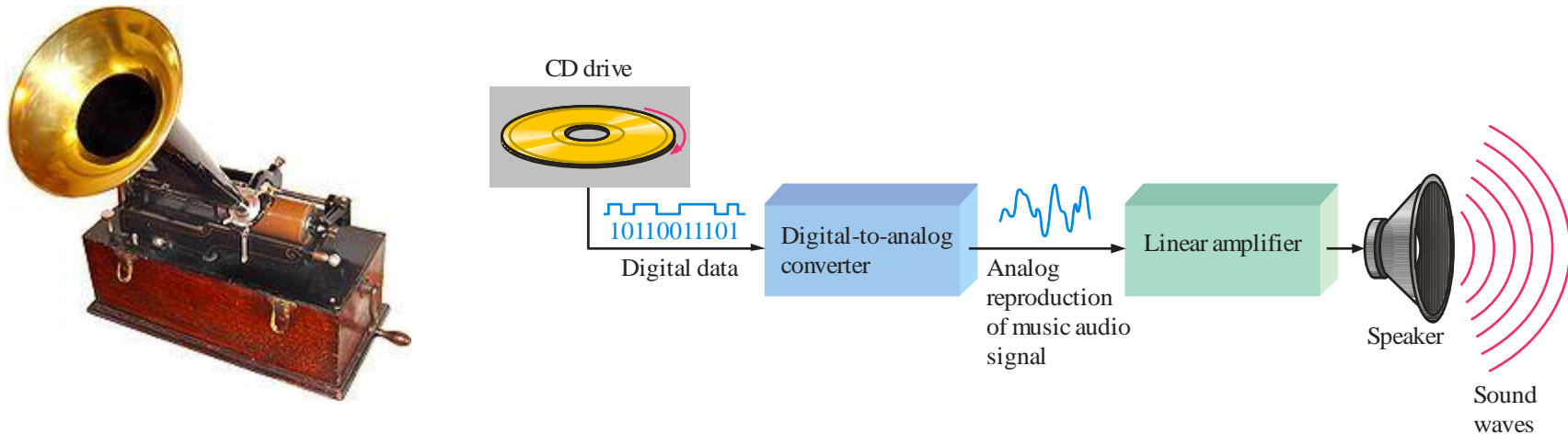


Analog = continuous  
Digital = discrete

**Digital systems** can process, store, and transmit data more efficiently but can only assign discrete values to each point.

# Digital Discipline Example – Audio

- Analog to Digital Converters ...  
Sampling and Quantization

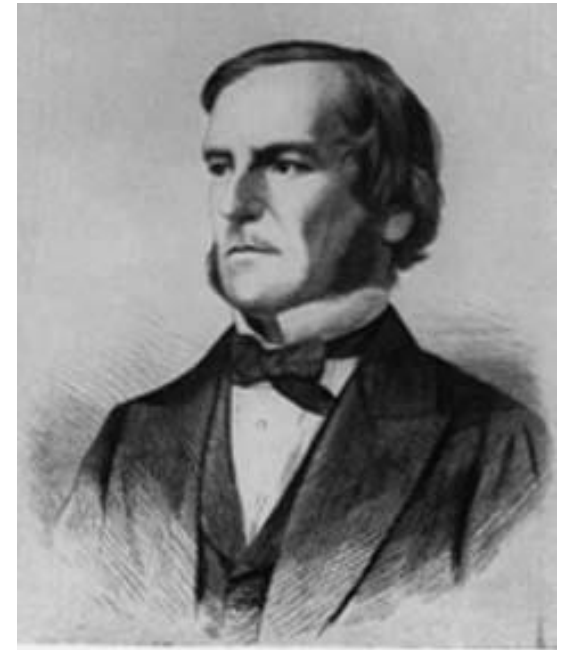


Types of electronic devices or instruments:

- Analog
- Digital
- Combination analog and digital

# George Boole, 1815-1864

- Born to working class parents
- Taught himself mathematics and joined the faculty of Queen's College in Ireland
- Wrote *An Investigation of the Laws of Thought* (1854)
- Introduced binary variables
- Introduced the three fundamental logic operations: AND, OR, and NOT



GEORGE BOOLE

Scanned at the American  
Institute of Physics

# Number Systems

- Decimal numbers

1's column  
10's column  
100's column  
1000's column

$$5374_{10} = 5 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$$

five                      three                      seven                      four  
thousands              hundreds              tens                      ones

- Binary numbers

1's column  
2's column  
4's column  
8's column

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$$

one                      one                      no                      one  
eight                      four                      two                      one



# Powers of Two

- $2^0 = 1$
- $2^1 = 2$
- $2^2 = 4$
- $2^3 = 8$
- $2^4 = 16$
- $2^5 = 32$
- $2^6 = 64$
- $2^7 = 128$
- $2^8 = 256$
- $2^9 = 512$
- $2^{10} = 1024$
- $2^{11} = 2048$
- $2^{12} = 4096$
- $2^{13} = 8192$
- $2^{14} = 16384$
- $2^{15} = 32768$
- Handy to memorize up to  $2^9$

# Number Conversion

- Decimal to binary conversion:

- Convert  $10011_2$  to decimal
- $16 \times 1 + 8 \times 0 + 4 \times 0 + 2 \times 1 + 1 \times 1 = 19_{10}$

- Decimal to binary conversion:

- Convert  $47_{10}$  to binary
- $32 \times 1 + 16 \times 0 + 8 \times 1 + 4 \times 1 + 2 \times 1 + 1 \times 1 = 101111_2$

# Binary Values and Range

- $N$ -digit decimal number
  - How many values?  $10^N$
  - Range?  $[0, 10^N - 1]$
  - Example: 3-digit decimal number:
    - $10^3 = 1000$  possible values
    - Range:  $[0, 999]$
- $N$ -bit binary number
  - How many values?  $2^N$
  - Range:  $[0, 2^N - 1]$
  - Example: 3-digit binary number:
    - $2^3 = 8$  possible values
    - Range:  $[0, 7] = [000_2 \text{ to } 111_2]$

# Hexadecimal Numbers

FROM ZERO TO ONE

Hex Digit	Decimal Equivalent	Binary Equivalent
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111



# Hexadecimal Numbers

- Base 16
- Shorthand for binary

# Hexadecimal to Binary Conversion

- Hexadecimal to binary conversion:
  - Convert  $4AF_{16}$  (also written  $0x4AF$ ) to binary
  - $0100\ 1010\ 1111_2$
- Hexadecimal to decimal conversion:
  - Convert  $4AF_{16}$  to decimal
  - $16^2 \times 4 + 16^1 \times 10 + 16^0 \times 15 = 1199_{10}$



# Bits, Bytes, Nibbles...

- Bits

10010110  
└─┬─┘ └─┬─┘  
most      least  
significant      significant  
bit                      bit

- Bytes & Nibbles

byte  
┌───────────┐  
10010110  
└─────────┘  
nibble

- Bytes

CEBF9AD7  
└─┬─┘ └─┬─┘  
most      least  
significant      significant  
byte                      byte

# Large Powers of Two

- $2^{10} = 1 \text{ kilo} \approx 1000 (1024)$
- $2^{20} = 1 \text{ mega} \approx 1 \text{ million } (1,048,576)$
- $2^{30} = 1 \text{ giga} \approx 1 \text{ billion } (1,073,741,824)$

# Addition

- Decimal

$$\begin{array}{r} 11 \leftarrow \text{carries} \\ 3734 \\ + 5168 \\ \hline 8902 \end{array}$$

- Binary

$$\begin{array}{r} 11 \leftarrow \text{carries} \\ 1011 \\ + 0011 \\ \hline 1110 \end{array}$$

Problems : Big Numbers / Negative Numbers  
\\ Engineer Life ... develop and learn .. All times

# Binary Addition Examples

- Add the following 4-bit binary numbers

$$\begin{array}{r} 1 \\ 1001 \\ + 0101 \\ \hline 1110 \end{array}$$

- Add the following 4-bit binary numbers

$$\begin{array}{r} 111 \\ 1011 \\ + 0110 \\ \hline 10001 \end{array}$$

Overflow!

# Overflow

- Digital systems operate on a **fixed number of bits**
- Overflow: when the result is too big to fit in the available number of bits
- See the previous example of  $11 + 6$

[Ariane-5 Rocket Explosion \(2002\)](#)

$$\begin{array}{r} 111 \\ 1011 \\ + 0110 \\ \hline 10001 \end{array}$$

Overflow!

# Signed Binary Numbers

- Sign/Magnitude Numbers
- Two's Complement Numbers

# Sign/Magnitude Numbers

- 1 sign bit,  $N-1$  magnitude bits
  - Sign bit is the most significant (left-most) bit
    - Positive number: sign bit = 0     $A : \{a_{N-1}, a_{N-2}, \dots, a_2, a_1, a_0\}$
    - Negative number: sign bit = 1     $A = (-1)^{a_{n-1}} \sum_{i=0}^{n-2} a_i 2^i$
- Sign    Magnitude
- Example, 4-bit sign/mag representations of  $\pm 6$ :
    - +6 = **0110**
    - 6 = **1110**
  - Range of an  $N$ -bit sign/magnitude number:  
 **$[-(2^{N-1}-1), 2^{N-1}-1]$**



# Sign/Magnitude Numbers

- Problems:
  - Addition doesn't work, for example  $-6 + 6$ :

$$\begin{array}{r} 1110 \\ + 0110 \\ \hline 10100 \text{ (wrong!)} \end{array}$$

- Two representations of 0 ( $\pm 0$ ):

1000

0000

# Two's Complement Numbers

- Don't have same problems as sign/magnitude numbers:
  - Addition works
  - Single representation for 0

# Two's Complement Numbers

- Msb has value of  $-2^{N-1}$       Magnitude

$$A = a_{n-1} \left( \underset{\text{Sign}}{-2^{n-1}} \right) + \sum_{i=0}^{n-2} a_i 2^i$$

- Most positive 4-bit number: **0111**
- Most negative 4-bit number: **1000**
- **The most significant bit still indicates the sign**  
(1 = negative, 0 = positive)
- Range of an  $N$ -bit two's comp number:

$$[-(2^{N-1}), 2^{N-1}-1]$$

# “Taking the Two’s Complement”

- Flip the sign of a two’s complement number
- Method:
  1. Invert the bits
  2. Add 1
- Example: Flip the sign of  $3_{10} = 0011_2$

1. **1100**

2. **+ 1**

**$1101 = -8 + 4 + 0 + 1 = -3_{10}$**

# Two's Complement Examples

- Take the two's complement of  $6_{10} = 0110_2$

1. 1001

2. + 1

$1010_2 = -6_{10}$

- What is the decimal value of the two's complement number  $1001_2$ ?

1. 0110

2. + 1

$0111_2 = 7_{10}$ , so  $1001_2 = -7_{10}$

# Two's Complement Addition

- Add  $6 + (-6)$  using two's complement numbers

$$\begin{array}{r} 111 \\ 0110 \\ + 1010 \\ \hline 10000 \end{array}$$

- Add  $-2 + 3$  using two's complement numbers

$$\begin{array}{r} 111 \\ 1110 \\ + 0011 \\ \hline 10001 \end{array}$$

# Increasing Bit Width

- **Extend number from  $N$  to  $M$  bits ( $M > N$ ) :**
  - Sign-extension
  - Zero-extension



# Sign-Extension

- Sign bit copied to msb's
- Number value is same

## • Example 1:

- 4-bit representation of 3 = 0011
- 8-bit sign-extended value: 00000011

## • Example 2:

- 4-bit representation of -5 = 1011
- 8-bit sign-extended value: 11111011

# Zero-Extension

- Zeros copied to msb's
- Value changes for negative numbers

## • Example 1:

- 4-bit value =  $0011_2 = 3_{10}$
- 8-bit zero-extended value:  $00000011 = 3_{10}$

## • Example 2:

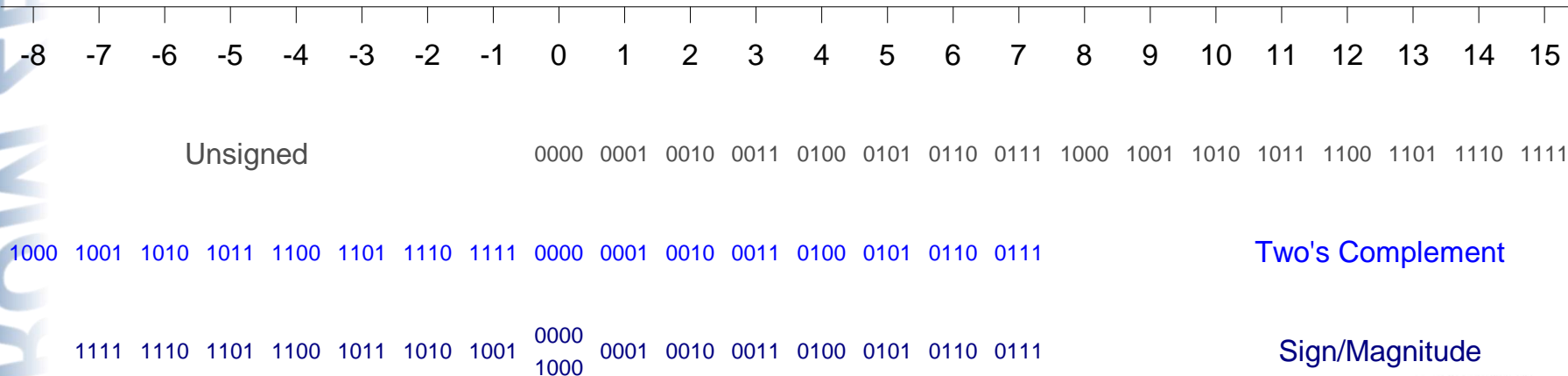
- 4-bit value =  $1011 = -5_{10}$
- 8-bit zero-extended value:  $00001011 = 11_{10}$



# Number System Comparison

Number System	Range
Unsigned	$[0, 2^N-1]$
Sign/Magnitude	$[-(2^{N-1}-1), 2^{N-1}-1]$
Two's Complement	$[-2^{N-1}, 2^{N-1}-1]$

For example, 4-bit representation:



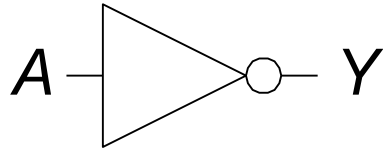
FROM ZERO TO ONE

# Logic Gates

- **Perform logic functions:**
  - inversion (NOT), AND, OR, NAND, NOR, etc.
- **Single-input:**
  - NOT gate, buffer
- **Two-input:**
  - AND, OR, XOR, NAND, NOR, XNOR
- **Multiple-input**

# Single-Input Logic Gates

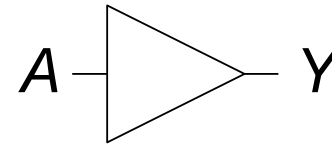
## NOT



$$Y = \bar{A}$$

A	Y
0	1
1	0

## BUF

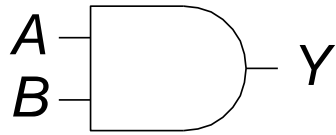


$$Y = A$$

A	Y
0	0
1	1

# Two-Input Logic Gates

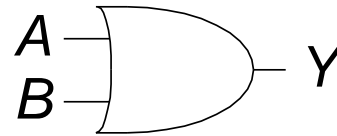
## AND



$$Y = AB$$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

## OR

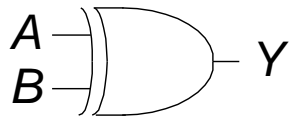


$$Y = A + B$$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

# More Two-Input Logic Gates

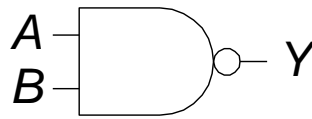
## XOR



$$Y = A \oplus B$$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

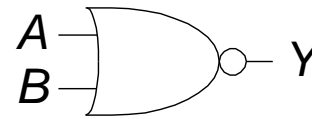
## NAND



$$Y = \overline{AB}$$

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

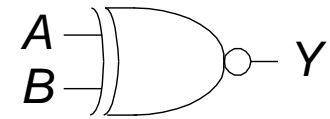
## NOR



$$Y = \overline{A + B}$$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

## XNOR



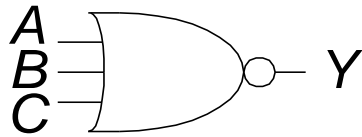
$$Y = \overline{A \oplus B}$$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1



# Multiple-Input Logic Gates

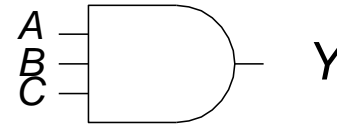
## NOR3



$$Y = \overline{A+B+C}$$

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

## AND3



$$Y = ABC$$

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

- Multi-input XOR: Odd parity

FROM ZERO TO ONE

# Logic Levels

- Discrete voltages represent 1 and 0
- For example:
  - 0 = *ground* (GND) or 0 volts
  - 1 =  $V_{DD}$  or 5 volts
- What about 4.99 volts? Is that a 0 or a 1?
- What about 3.2 volts?

# Logic Levels

- *Range* of voltages for 1 and 0
- Different ranges for inputs and outputs to allow for *noise*

# V<sub>DD</sub> Scaling

- In 1970's and 1980's, V<sub>DD</sub> = 5 V
- V<sub>DD</sub> has dropped
  - Avoid frying tiny transistors
  - Save power
- 3.3 V, 2.5 V, 1.8 V, 1.5 V, 1.2 V, 1.0 V, ...
- Be careful connecting chips with different supply voltages

Chips operate because they contain magic smoke

Proof:

- if the magic smoke is let out, the chip stops working



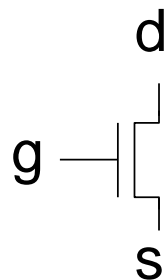
# Logic Family Examples

Logic Family	$V_{DD}$	$V_{IL}$	$V_{IH}$	$V_{OL}$	$V_{OH}$
TTL	5 (4.75 - 5.25)	0.8	2.0	0.4	2.4
CMOS	5 (4.5 - 6)	1.35	3.15	0.33	3.84
LVTTL	3.3 (3 - 3.6)	0.8	2.0	0.4	2.4
LVC MOS	3.3 (3 - 3.6)	0.9	1.8	0.36	2.7

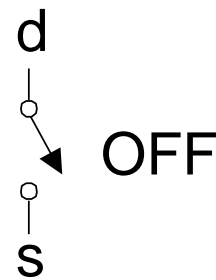


# Transistors

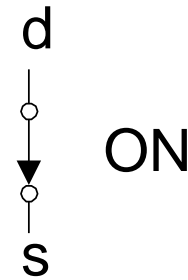
- Logic gates built from transistors
- 3-ported voltage-controlled switch
  - 2 ports connected depending on voltage of 3rd
  - d and s are connected (ON) when g is 1



$g = 0$

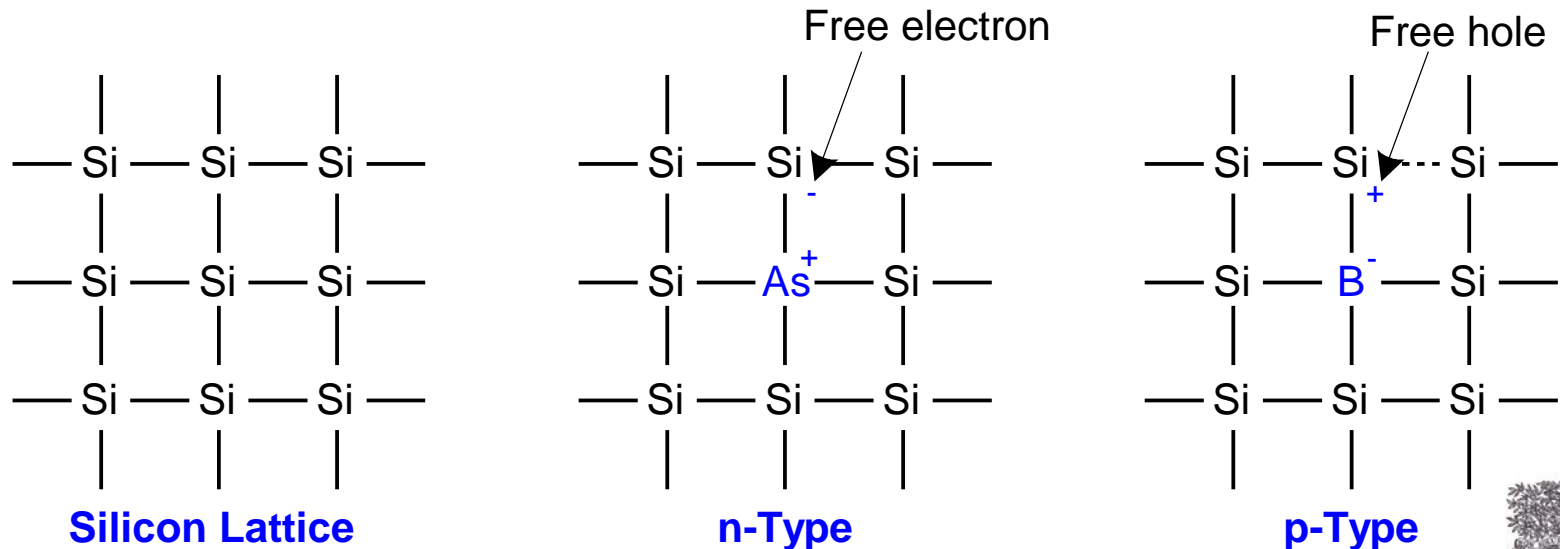


$g = 1$



# Silicon

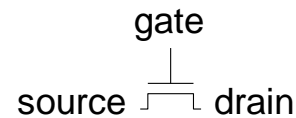
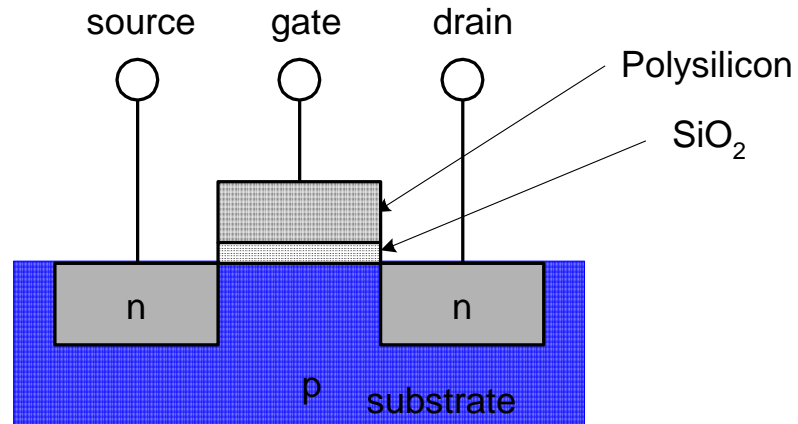
- Transistors built from silicon, a semiconductor
- Pure silicon is a poor conductor (no free charges)
- Doped silicon is a good conductor (free charges)
  - n-type (free *negative* charges, electrons)
  - p-type (free *positive* charges, holes)





# MOS Transistors

- **Metal oxide silicon (MOS) transistors:**
  - Polysilicon (used to be **metal**) gate
  - **Oxide** (silicon dioxide) insulator
  - Doped **silicon**

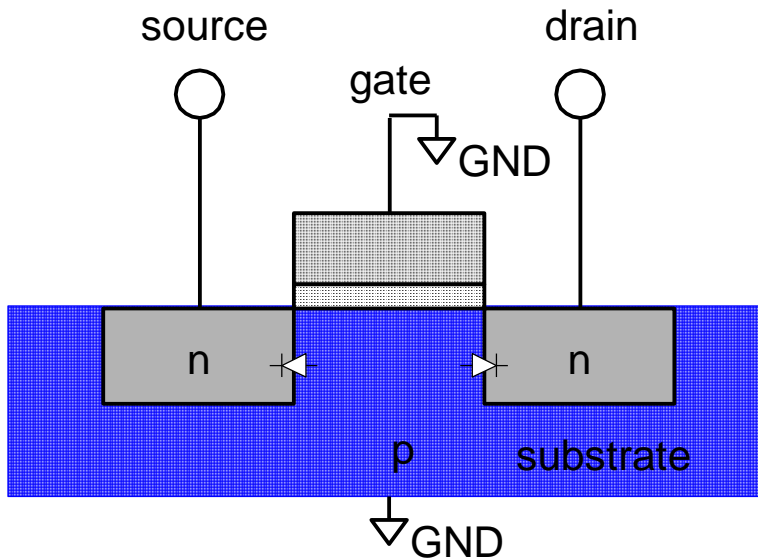


nMOS

# Transistors: nMOS

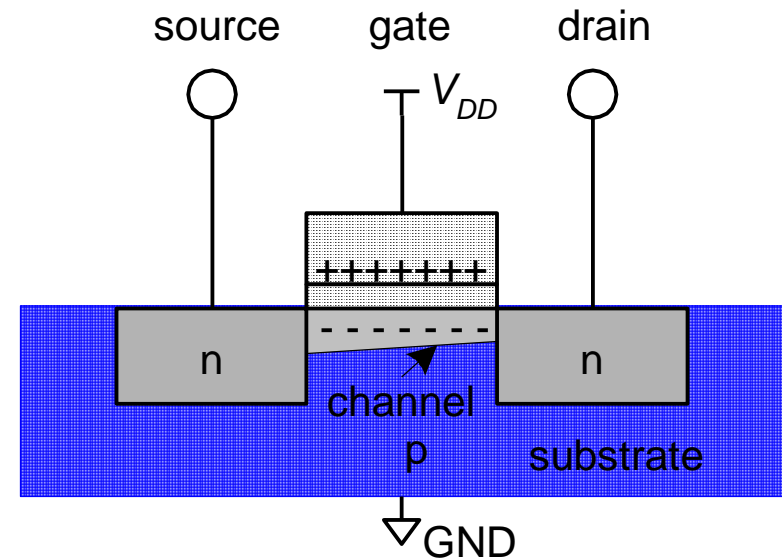
Gate = 0

OFF (no connection between source and drain)



Gate = 1

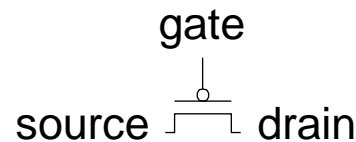
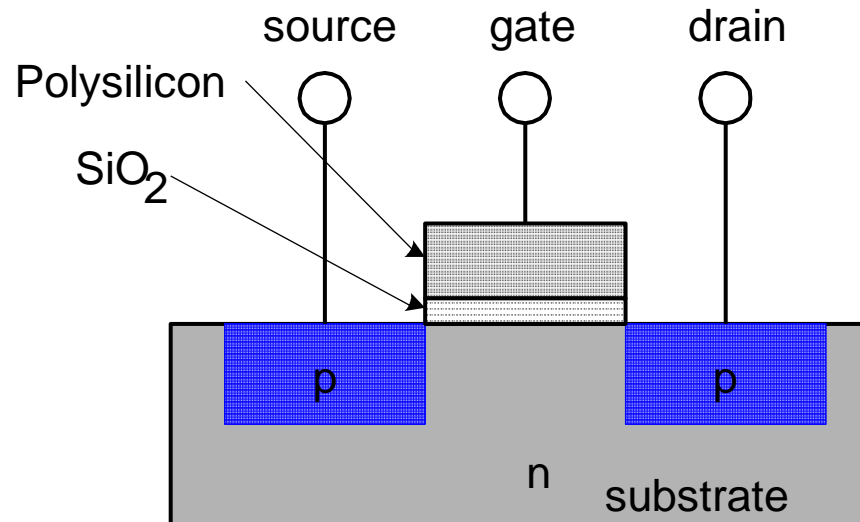
ON (channel between source and drain)



FROM ZERO TO ONE

# Transistors: pMOS

- pMOS transistor is opposite
  - ON when Gate = 0
  - OFF when Gate = 1

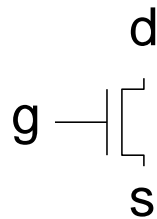


FROM ZERO TO ONE

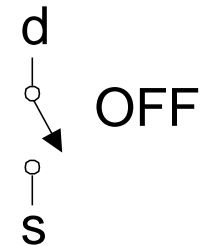
# Transistor Function

FROM ZERO TO ONE

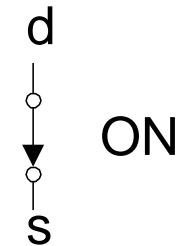
nMOS



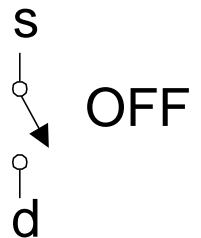
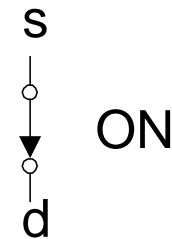
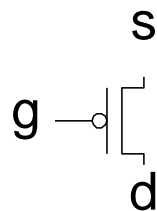
$g = 0$



$g = 1$

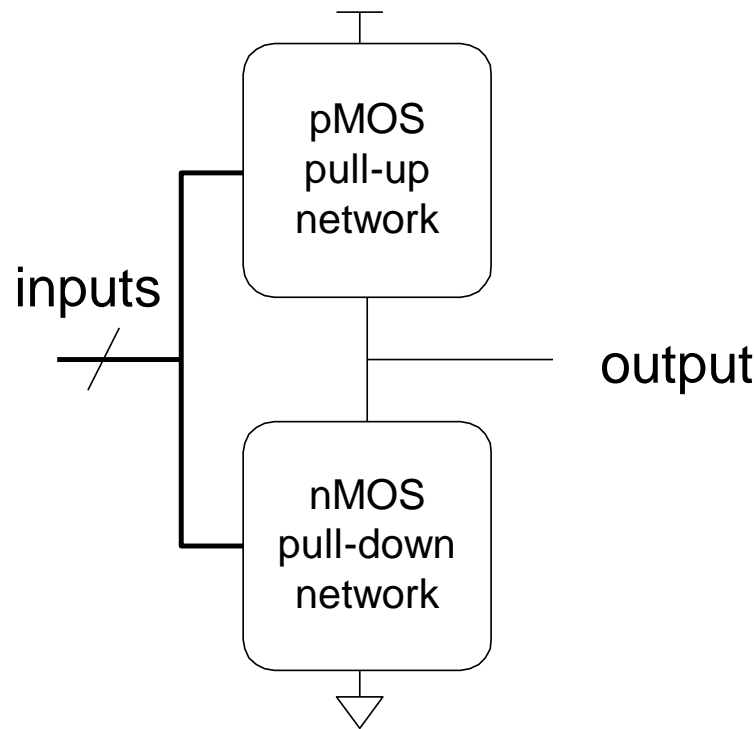


pMOS



# Transistor Function

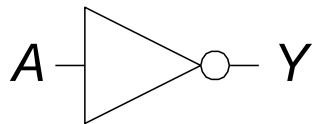
- **nMOS:** pass good 0's, so connect source to GND
- **pMOS:** pass good 1's, so connect source to  $V_{DD}$



FROM ZERO TO ONE

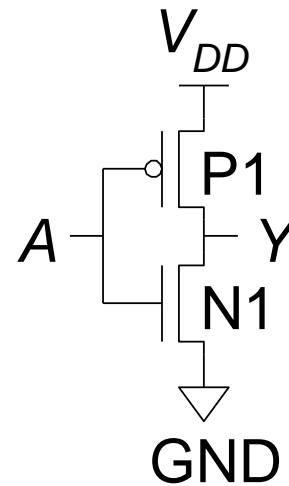
# CMOS Gates: NOT Gate

## NOT



$$Y = \bar{A}$$

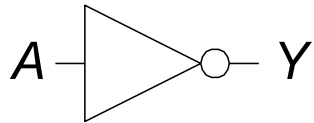
A	Y
0	1
1	0



A	P1	N1	Y
0			
1			

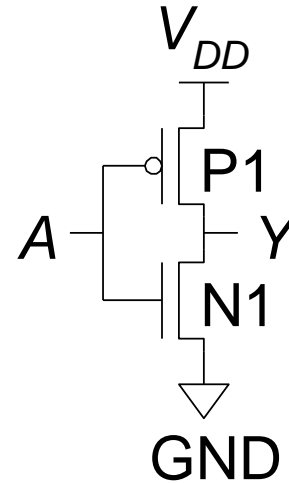
# CMOS Gates: NOT Gate

## NOT



$$Y = \bar{A}$$

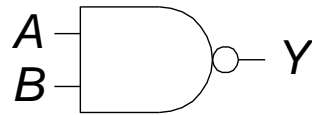
A	Y
0	1
1	0



A	P1	N1	Y
0	ON	OFF	1
1	OFF	ON	0

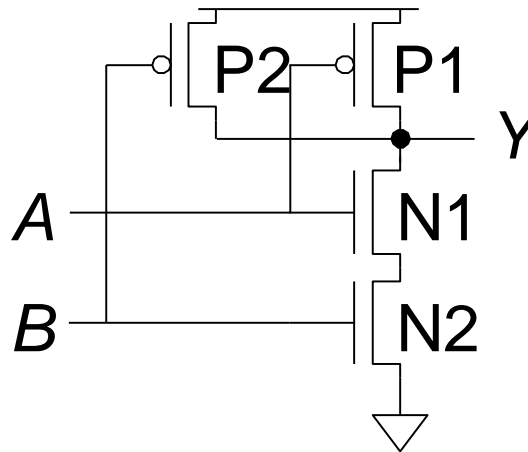
# CMOS Gates: NAND Gate

## NAND



$$Y = \overline{AB}$$

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

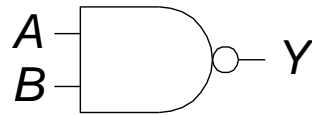


A	B	P1	P2	N1	N2	Y
0	0					
0	1					
1	0					
1	1					



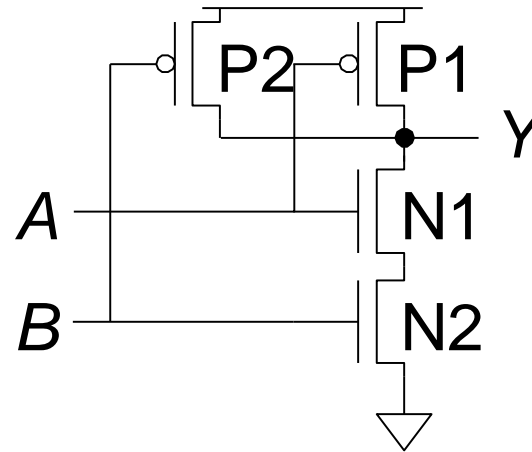
# CMOS Gates: NAND Gate

## NAND



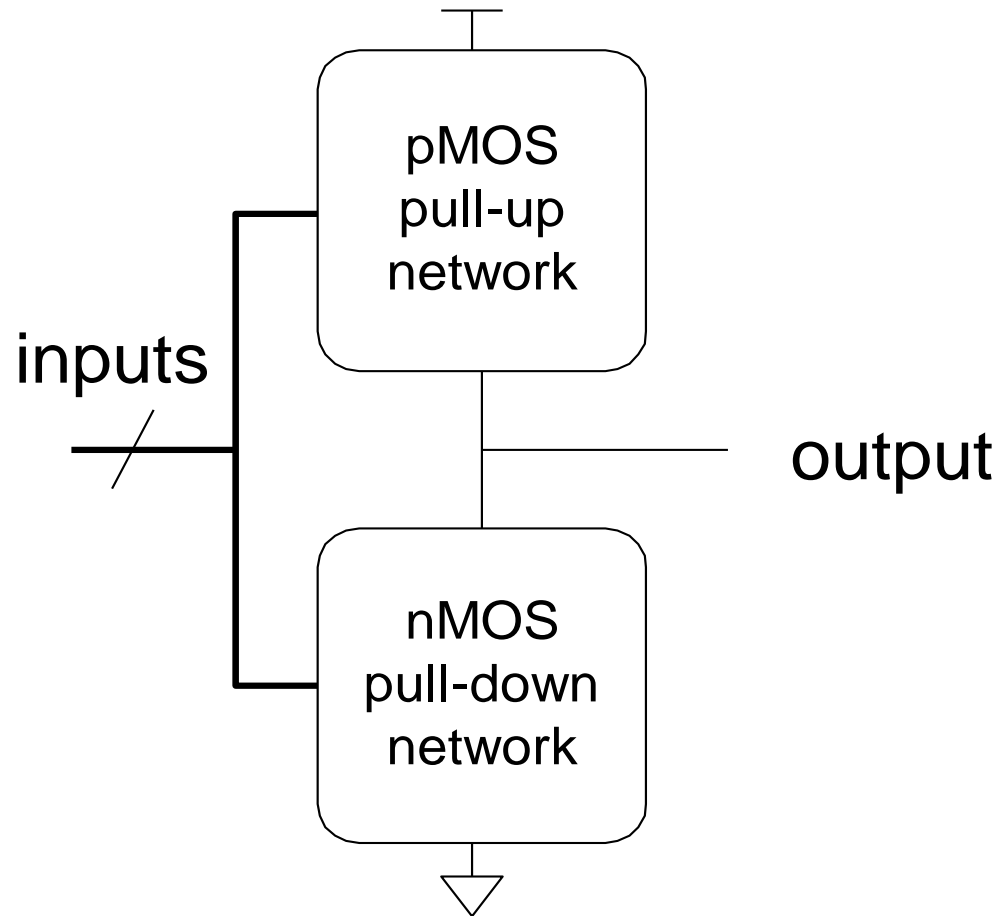
$$Y = \overline{AB}$$

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0



A	B	P1	P2	N1	N2	Y
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	1
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0

# CMOS Gate Structure

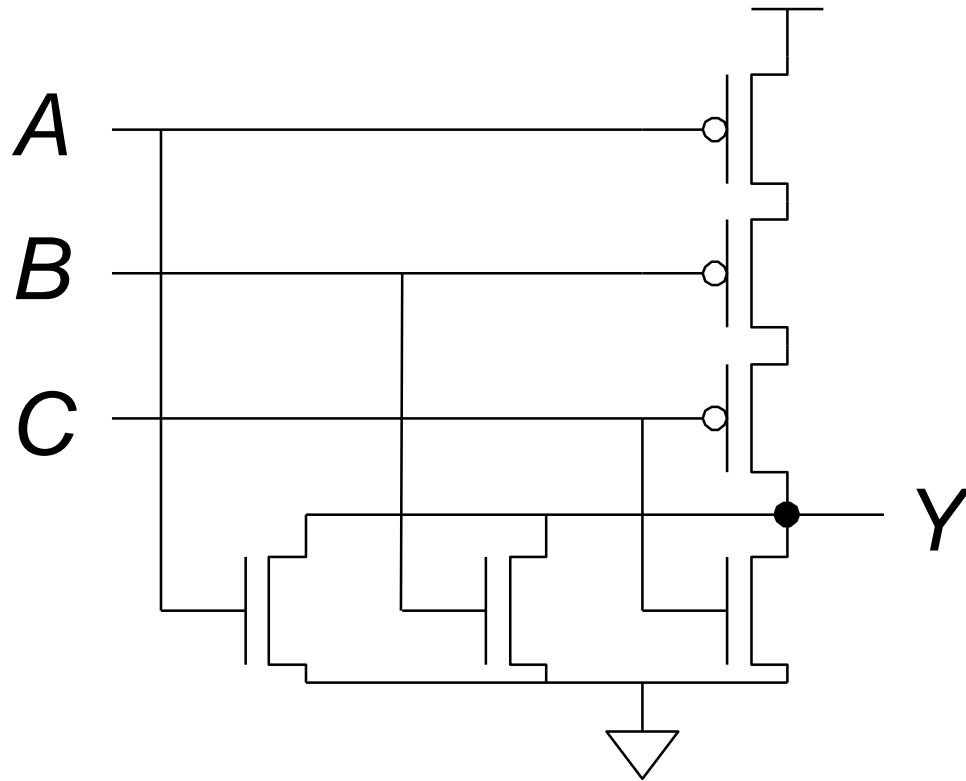


Silicon Run

# NOR Gate

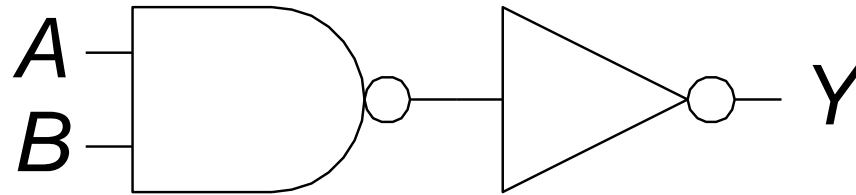
How do you build a three-input NOR gate?

# NOR3 Gate



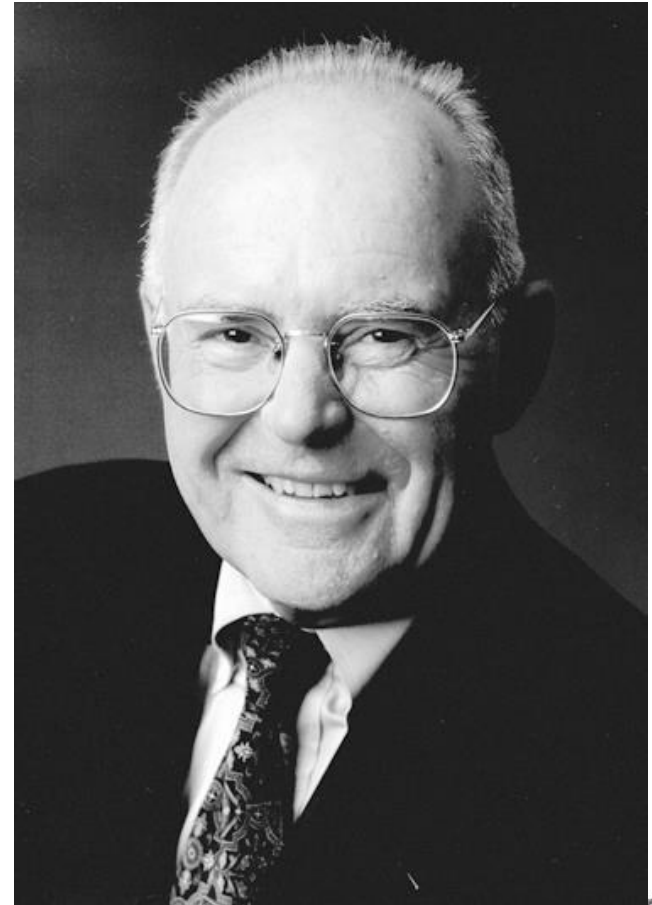
# Other CMOS Gates

How do you build a two-input AND gate?

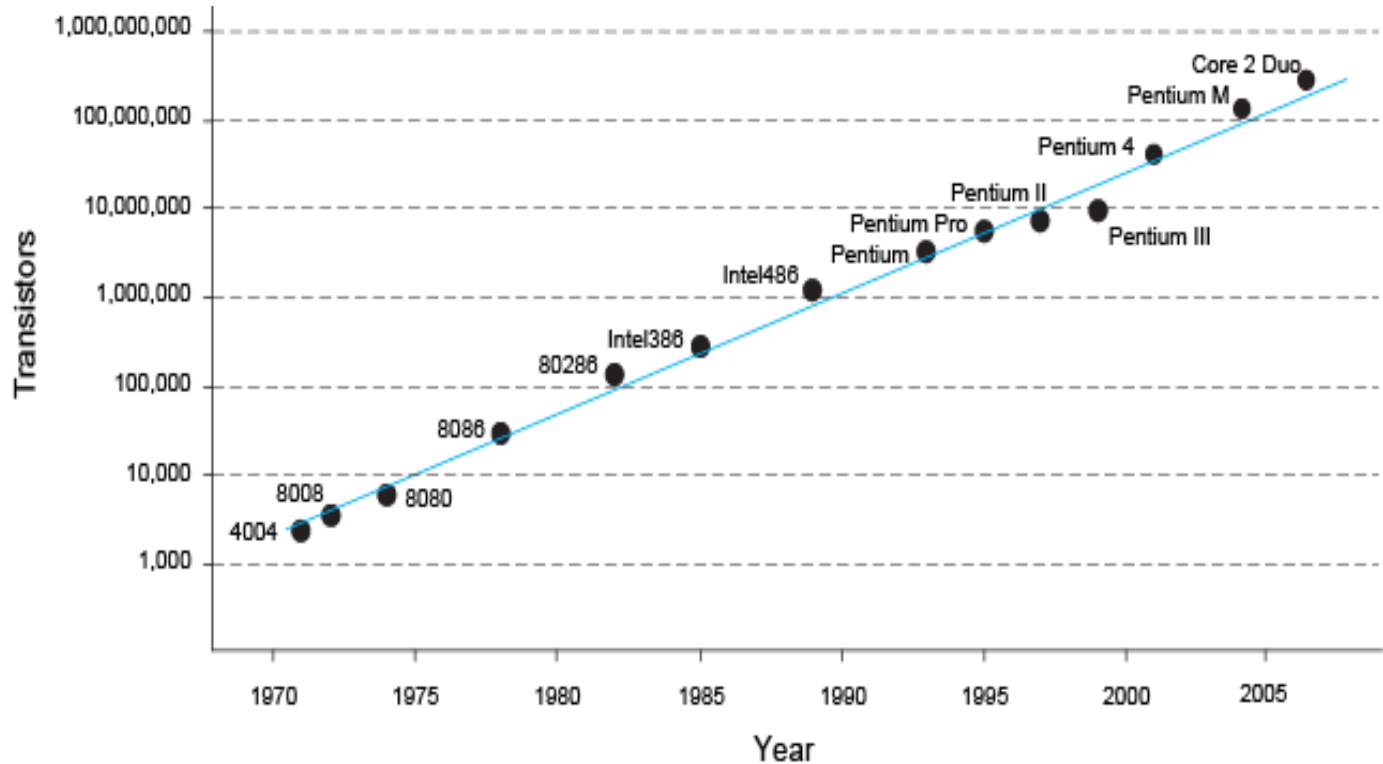


# Gordon Moore, 1929-

- Cofounded Intel in 1968 with Robert Noyce.
- **Moore's Law:** number of transistors on a computer chip doubles every year (observed in 1965)
- Since 1975, transistor counts have doubled every two years.



# Moore's Law



- *“If the automobile had followed the same development cycle as the computer, a Rolls-Royce would today cost \$100, get one million miles to the gallon, and explode once a year . . .”*

– Robert Cringley