Digital Design & Computer Architecture

Sarah Harris & David Harris

Chapter 1: From Zero to One

Modified by Younghwan Yoo, 2023

Chapter 1 :: Topics

- **The Art of Managing Complexity**
- **Number Systems**
  - Binary Numbers
  - Hexadecimal Numbers
  - Bits, Bytes, Nibbles
  - Addition
  - Signed Numbers
  - Extension
- **Logic Gates**
- **Logic Levels**
- **CMOS Transistors**
- **Transistor-Level Gate Design**
- **Power Consumption**

Chapter 1: From Zero to One

The Art of Managing Complexity

The Art of Managing Complexity

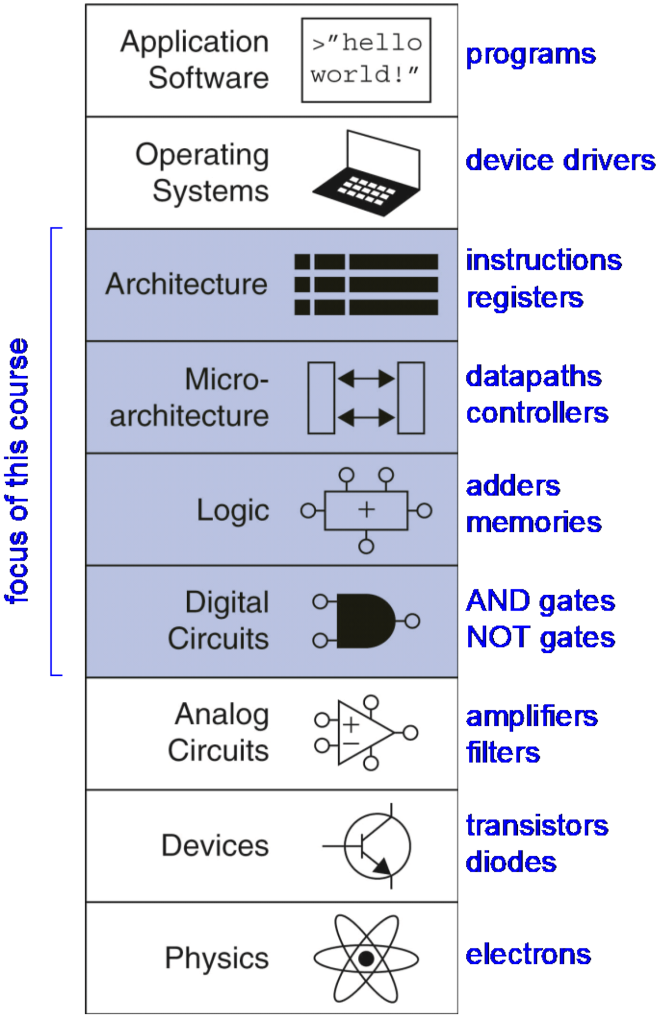
- How do we design things that are too big to fit in one person's head at once?

- Abstraction
- Discipline
- The Three –y's
  - Hierarch**y**
  - Modularit**y**
  - Regularit**y**

Abstraction

Hiding details when they aren't important

- **Digital circuits**: logic gates converting analog voltages to 0 or 1
- **Logic design**: complex structures, e.g., adders and memories
- **Microarchitecture**: combining logic elements to execute instructions defined by *Architecture*
- **Architecture**: a set of instructions and registers that programmers use



Discipline

- Intentionally restrict design choices
- Example: Digital discipline
  - Discrete voltages instead of continuous
  - 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

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
  - 0: low voltage (GND)
  - 1: high voltage (V_{DD})
- ***Bit***: *B*inary dig*it*

Number Systems: Binary Numbers

Number Systems

- Decimal numbers

Decimal numbers in digital systems mean any base 10 numbers, not just those with a decimal point.

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

$$5374_{10} = \underset{\text{five thousands}}{5 \times 10^3} + \underset{\text{three hundreds}}{3 \times 10^2} + \underset{\text{seven tens}}{7 \times 10^1} + \underset{\text{four ones}}{4 \times 10^0}$$

- Binary numbers

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

$$1101_2 = \underset{\text{one eight}}{1 \times 2^3} + \underset{\text{one four}}{1 \times 2^2} + \underset{\text{no two}}{0 \times 2^1} + \underset{\text{one one}}{1 \times 2^0} = 13_{10}$$

Counting in Binary

Binary	Decimal
0	0
1	1
10	2
11	3
100	4
101	5
…	

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

Number Conversion

- Binary to decimal conversion:
  - Convert 10011_2 to decimal
  - $16\times1 + 8\times0 + 4\times0 + 2\times1 + 1\times1 = 19_{10}$
- Decimal to binary conversion:
  - Convert 47_{10} to binary
  - $32\times1 + 16\times0 + 8\times1 + 4\times1 + 2\times1 + 1\times1 = 101111_2$

Decimal to Binary Conversion

- Two methods:
  - **Method 1:** Find the largest power of 2 that fits, subtract and repeat
  - **Method 2:** Repeatedly divide by 2, remainder goes in next most significant bit

Decimal to Binary Conversion

53_{10}

Method 1: Find the largest power of 2 that fits, subtract and repeat

53_{10}	32×1	
53-32 = 21	16×1	
21-16 = 5	4×1	
5-4 = 1	1×1	**= 110101_2**

Method 2: Repeatedly divide by 2, remainder goes in next most significant bit

53_{10} = 53/2 = 26 R1

26/2 = 13 R0

13/2 = 6 R1

6/2 = 3 R0

3/2 = 1 R1

1/2 = 0 R1 **= 110101_2**

Binary Values and Range

- ***N*-digit decimal number**
  - How many values?
  - Range?
  - Example: 3-digit decimal number:
    -
    -

- ***N*-bit binary number**
  - How many values?
  - Range:
  - Example: 3-digit binary number:
    -
    -

Chapter 1: From Zero to One

Number Systems: Hexadecimal Numbers

Hexadecimal Numbers

- Base 16
- Shorthand for binary

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 to Binary Conversion

- Hexadecimal to binary conversion:
  - Convert $4AF_{16}$ (also written 0x4AF) to binary
  -
- Hexadecimal to decimal conversion:
  - Convert $4AF_{16}$ to decimal
  -

Hexadecimal and Binary Prefixes

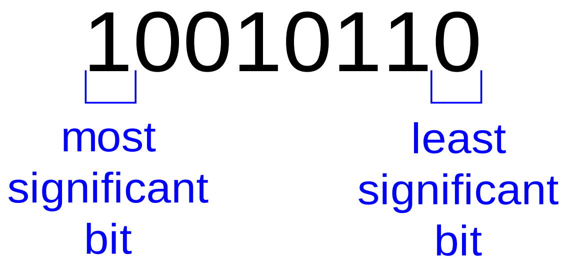
- Hard to write subscripts in text files
- Some programming languages uses prefixes
  - Hex: 0x
    - 0x23AB = $23AB_{16}$
  - Binary: 0b
    - 0b1101 = 1101_2

Chapter 1: From Zero to One

Number Systems: Bytes, Nibbles, & All That Jazz

Bits, Bytes, Nibbles…

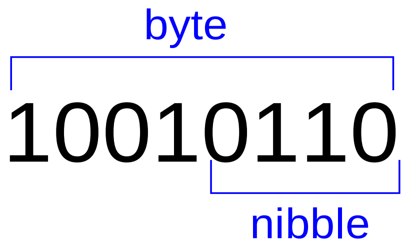
- Byte: 8 bits
  - Represents one of ______ values
  - [__, ___]
- Nibble: 4 bits
  - Represents one of ______ values
  - [__, ___]



10010110

most significant bit

least significant bit



byte

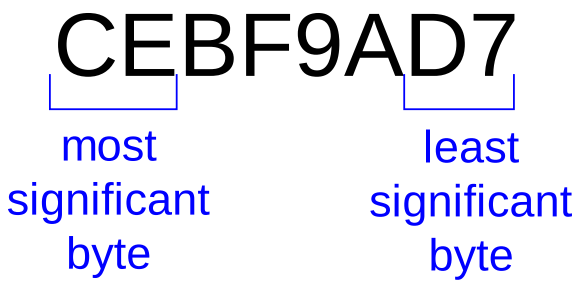
10010110

nibble

One binary digit is __ bit

One hex digit is ___ bits or ___ nibble

Two hex digits make ___ byte



CEBF9AD7

most significant byte

least significant byte

Most significant on left

Least significant on right

Large Powers of Two

- 2^{10} = 1 kilo ≈ 10^{3} (1024)
- 2^{20} = 1 mega ≈ 10^{6} (1,048,576)
- 2^{30} = 1 giga ≈ 10^{9} (1,073,741,824)
- 2^{40} = 1 tera ≈ 10^{12}
- 2^{50} = 1 peta ≈ 10^{15}
- 2^{60} = 1 exa ≈ 10^{18}

Estimating Powers of Two

- What is the value of 2^{24}?

- How large of a value can a 32-bit integer variable represent?

Chapter 1: From Zero to One

Number Systems: Addition

Addition

- Decimal

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

- Binary

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

Binary Addition Examples

- Add the following 4-bit binary numbers

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

- Add the following 4-bit binary numbers

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

Overflow

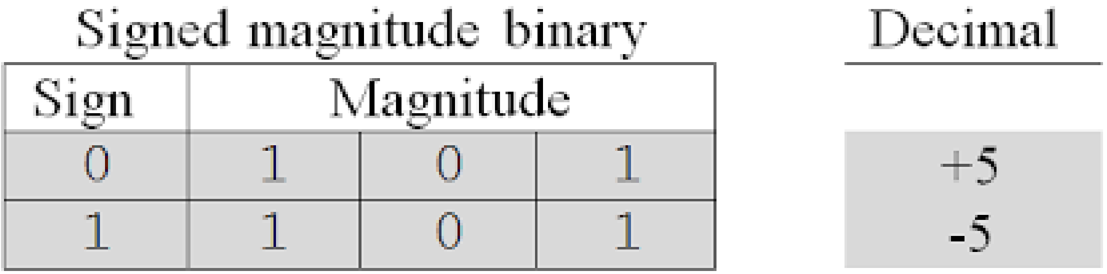
- Digital systems operate on a **fixed number of bits**
- Overflow: when result is too big to fit in the available number of bits
- See previous example of 11 + 6

Chapter 1: From Zero to One

Number Systems: Signed Numbers

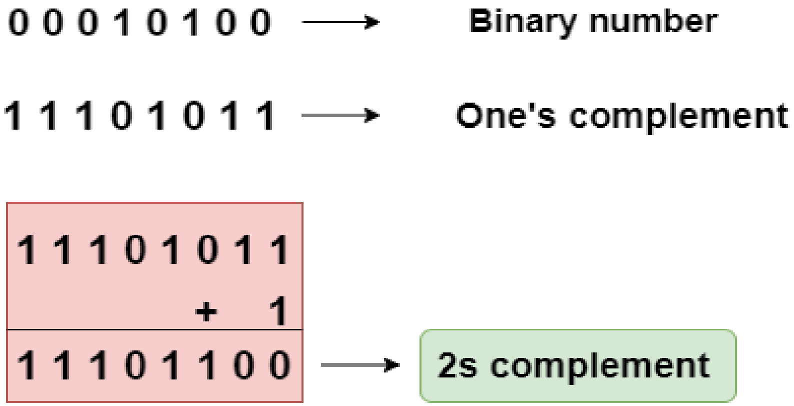
Signed Binary Numbers

- Sign/Magnitude Numbers



Signed magnitude binary				Decimal
Sign	Magnitude			
0	1	0	1	+5
1	1	0	1	-5

- Two’s Complement Numbers



0 0 0 1 0 1 0 0 → Binary number

1 1 1 0 1 0 1 1 → One's complement

1 1 1 0 1 0 1 1
+ 1
1 1 1 0 1 1 0 0 → 2s complement

(+20)

(-20)

Sign/Magnitude Numbers

- 1 sign bit, *N*-1 magnitude bits
- Sign bit is the most significant (left-most) bit
  - Positive number: sign bit = 0
  - Negative number: sign bit = 1

$$A: \{a_{N-1}, a_{N-2}, \dots a_2, a_1, a_0\}$$

$$A = (-1)^{a_{N-1}} \sum_{i=0}^{N-2} a_i \, 2^i$$

- Example, 4-bit sign/mag representations of ± 6:

  +6 =

  - 6 =

- Range of an *N*-bit sign/magnitude number:

Sign/Magnitude Numbers

Problems:

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

$$\begin{array}{r} 1110 \\ +\,0110 \\ \hline 10100 \end{array}$$

(wrong!)

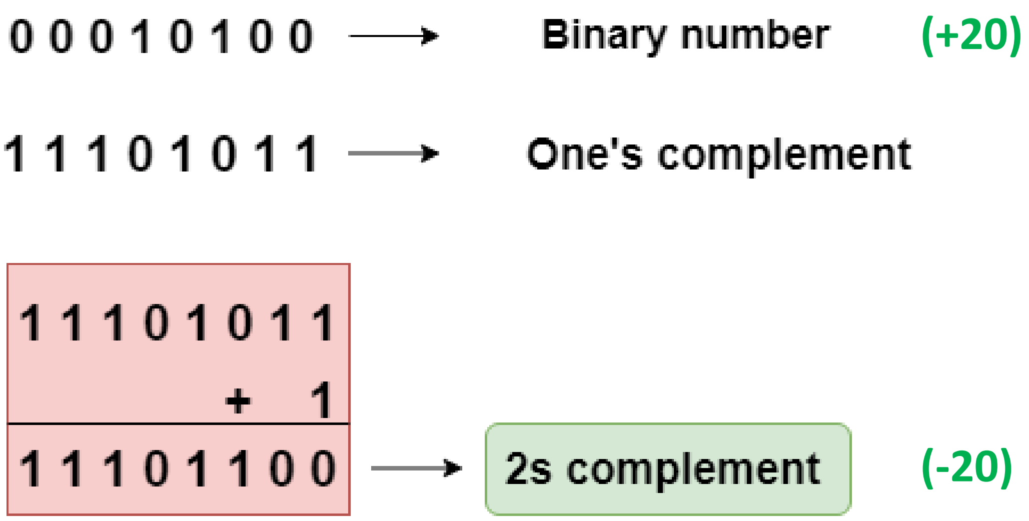
- Two representations of 0 (± 0):

1000

0000

Two's Complement Numbers

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



0 0 0 1 0 1 0 0 → Binary number (+20)

1 1 1 0 1 0 1 1 → One's complement

1 1 1 0 1 0 1 1
+ 1

1 1 1 0 1 1 0 0 → 2s complement (-20)

Two's Complement Numbers

- msb has weight of -2^{N-1}

$$A = a_{N-1}(-2^{N-1}) + \sum_{i=0}^{N-2} a_i\, 2^i$$

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

Reversing the Sign

- **How to reverse the sign** of a two's complement number
    1. Invert the bits
    2. Add 1
- **Example:** Reverse the sign of $3_{10} = 0011_2$
    1.
    2.

Historically, this reversing the sign method has been called: "Taking the Two's complement". But this terminology can be confusing, so we instead we call it "reversing the sign".

Two's Complement Examples

- Reverse the sign of $6_{10} = 0110_2$
  1.
  2.

- What is the decimal value of the two's complement number 1001_2?
  1.
  2.

Two’s Complement Addition

- Add 6 + (-6) using two’s complement numbers

$$\begin{array}{rr} & 0110 \\ + & 1010 \\ \hline \end{array}$$

- Add -2 + 3 using two’s complement numbers

$$\begin{array}{rr} & 1110 \\ + & 0011 \\ \hline \end{array}$$

Subtraction

- Subtract a 2’s complement number by reversing the sign and adding.
- Reverse sign by taking 2’s complement

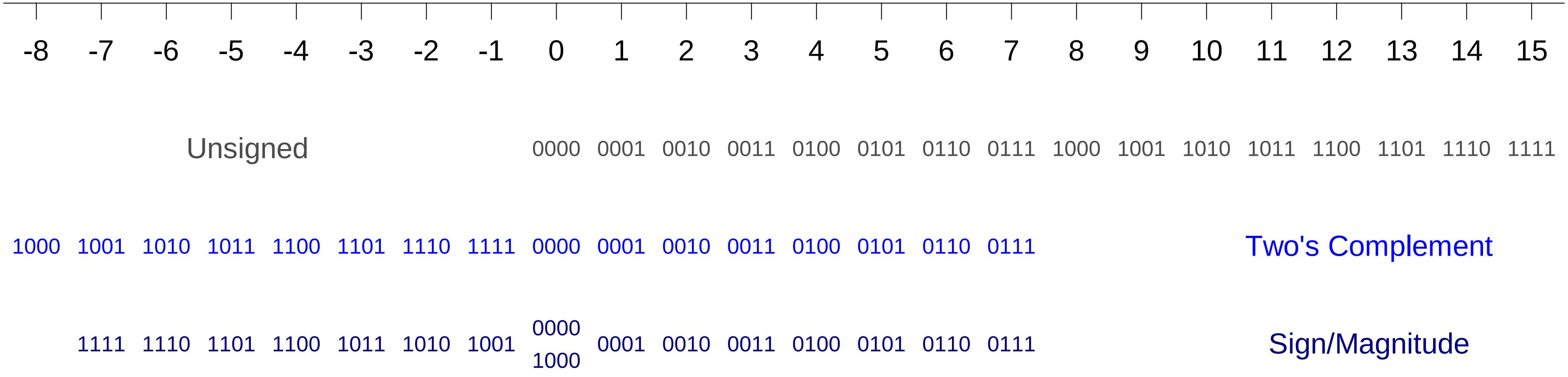
- Ex: 3 – 5 = 3 + (-5)

$$\begin{array}{rr@{\qquad}r} & 0011 & 3 \\ + & 1011 & -5 \\ \hline & 1110 & -2 \end{array}$$

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:



	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Unsigned									0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Two's Complement	1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111								
Sign/Magnitude		1111	1110	1101	1100	1011	1010	1001	0000 1000	0001	0010	0011	0100	0101	0110	0111								

Chapter 1: From Zero to One

Number Systems: Extension

Increasing Bit Width

Extend number from *N* to *M* bits (*M* > *N*) :

- **Sign-extension** for 2’s complement numbers
- **Zero-extension** for unsigned numbers

Sign-Extension

- Sign bit copied to msb's
- Number value is same
- **Example 1:**
  - 4-bit representation of 3 = **0**011
  - 8-bit sign-extended value: **0000**0011
- **Example 2:**
  - 4-bit representation of -5 = **1**011
  - 8-bit sign-extended value: **1111**1011

Zero-Extension

- Zeros copied to msb's
- Value changes for negative numbers
- **Example 1:**
  - 4-bit value = 0011 = 3_{10}
  - 8-bit zero-extended value: **0000**0011 = 3_{10}
- **Example 2:**
  - 4-bit value = 1011 = -5_{10}
  - 8-bit zero-extended value: **0000**1011 = 11_{10}

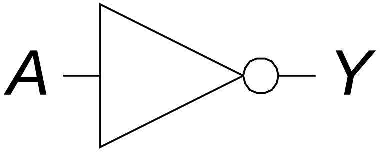
Logic Gates

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

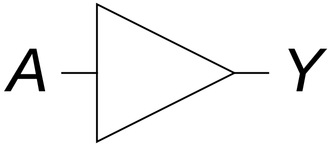
NOT



$Y = \overline{A}$

A	Y
0	
1	

BUF

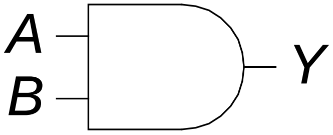


$Y = A$

A	Y
0	
1	

Two-Input Logic Gates

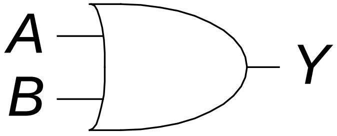
AND



$Y = AB$

A	B	Y
0	0	
0	1	
1	0	
1	1	

OR

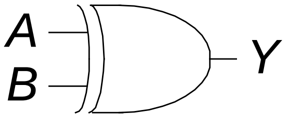


$Y = A + B$

A	B	Y
0	0	
0	1	
1	0	
1	1	

More Two-Input Logic Gates

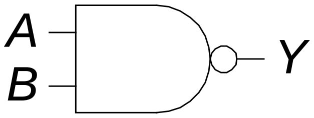
XOR



$$Y = A \oplus B$$

A	B	Y
0	0	
0	1	
1	0	
1	1	

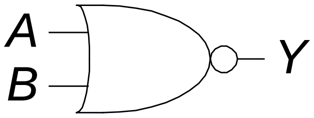
NAND



$$Y = \overline{AB}$$

A	B	Y
0	0	
0	1	
1	0	
1	1	

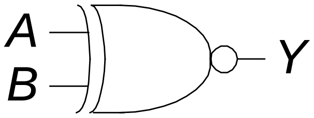
NOR



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

A	B	Y
0	0	
0	1	
1	0	
1	1	

XNOR



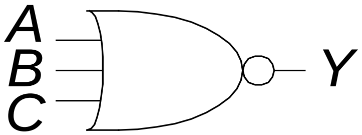
$$Y = \overline{A \oplus B}$$

A	B	Y
0	0	
0	1	
1	0	
1	1	

... called *equality gate* because it is TRUE when inputs are equal

Multiple-Input Logic Gates

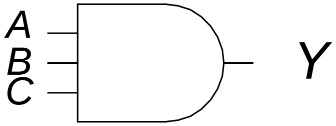
NOR3



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

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

AND3



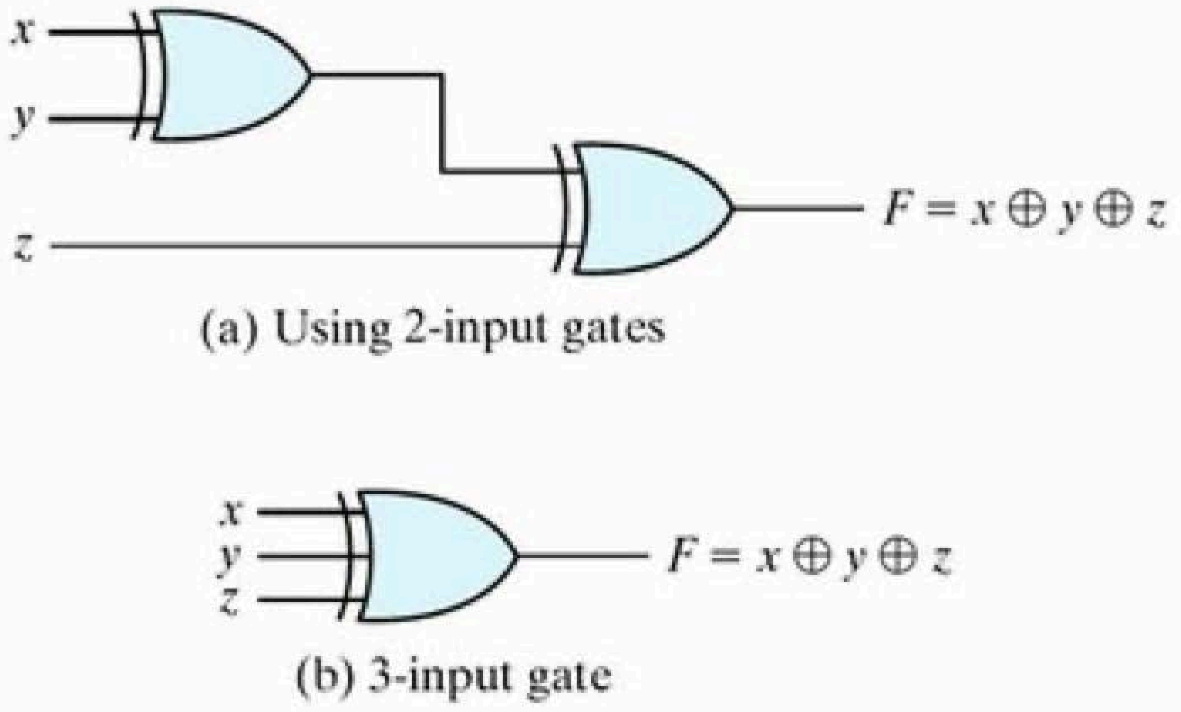
$$Y = ABC$$

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

Truth table rows are listed in binary order.

Multiple-Input XOR

- Odd parity



(a) Using 2-input gates

(b) 3-input gate

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(c) Truth table

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

Scanned at the American Institute of Physics

Chapter 1: From Zero to One

Logic Levels

Logic Levels

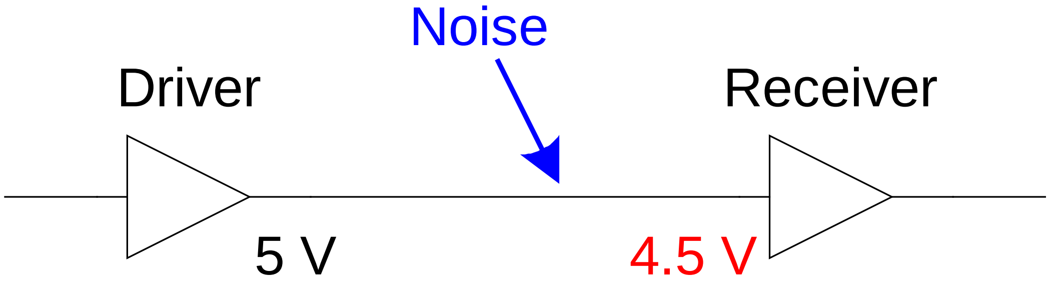
- Discrete voltages represent 1 and 0
- For example:
  - 0 = *ground* (GND) or 0 volts
  - 1 = V_{DD} or 5 volts
- What about 4.99 volts? Is that a 0 or a 1?
- What about 3.2 volts?

Logic Levels

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

What is Noise?

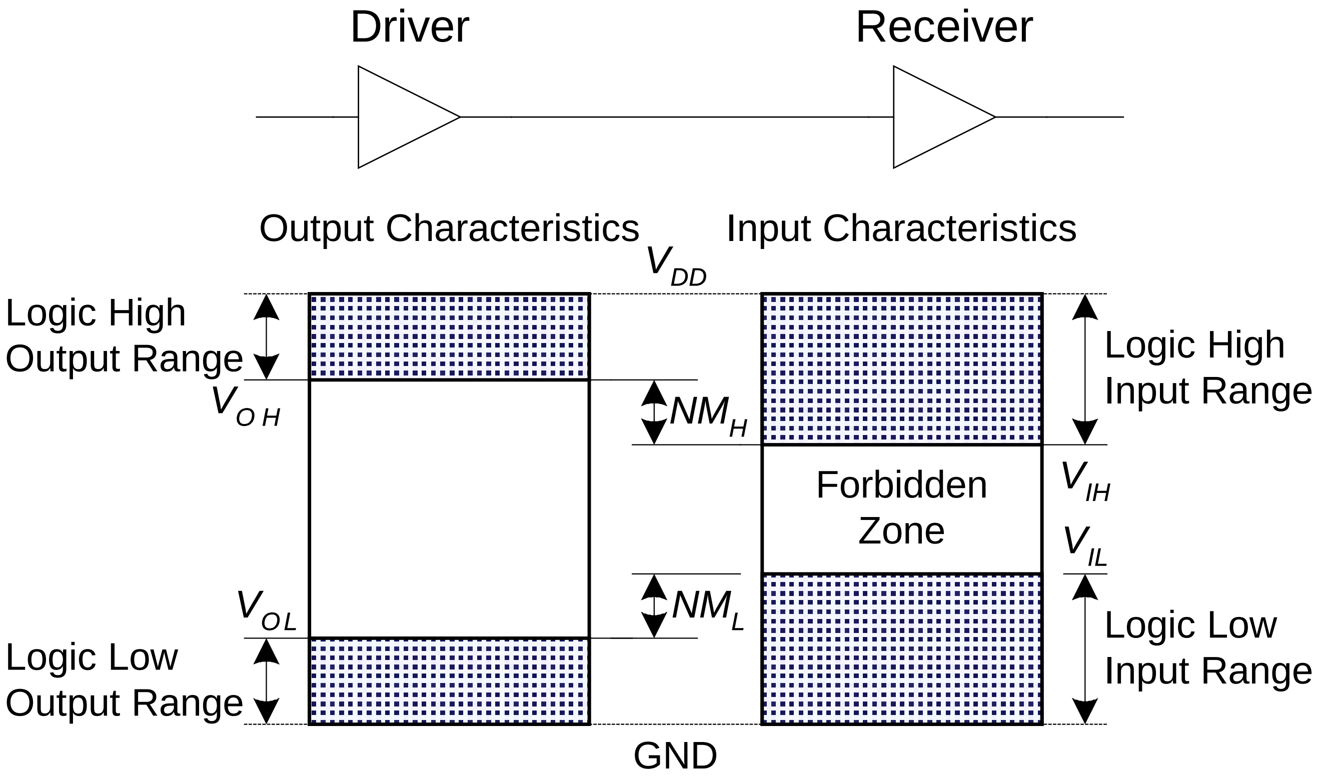
- **Anything that degrades the signal**
  - E.g., resistance, power supply noise, coupling to neighboring wires, etc.
- **Example:** a gate (driver) outputs 5 V but, because of resistance in a long wire, receiver gets 4.5 V



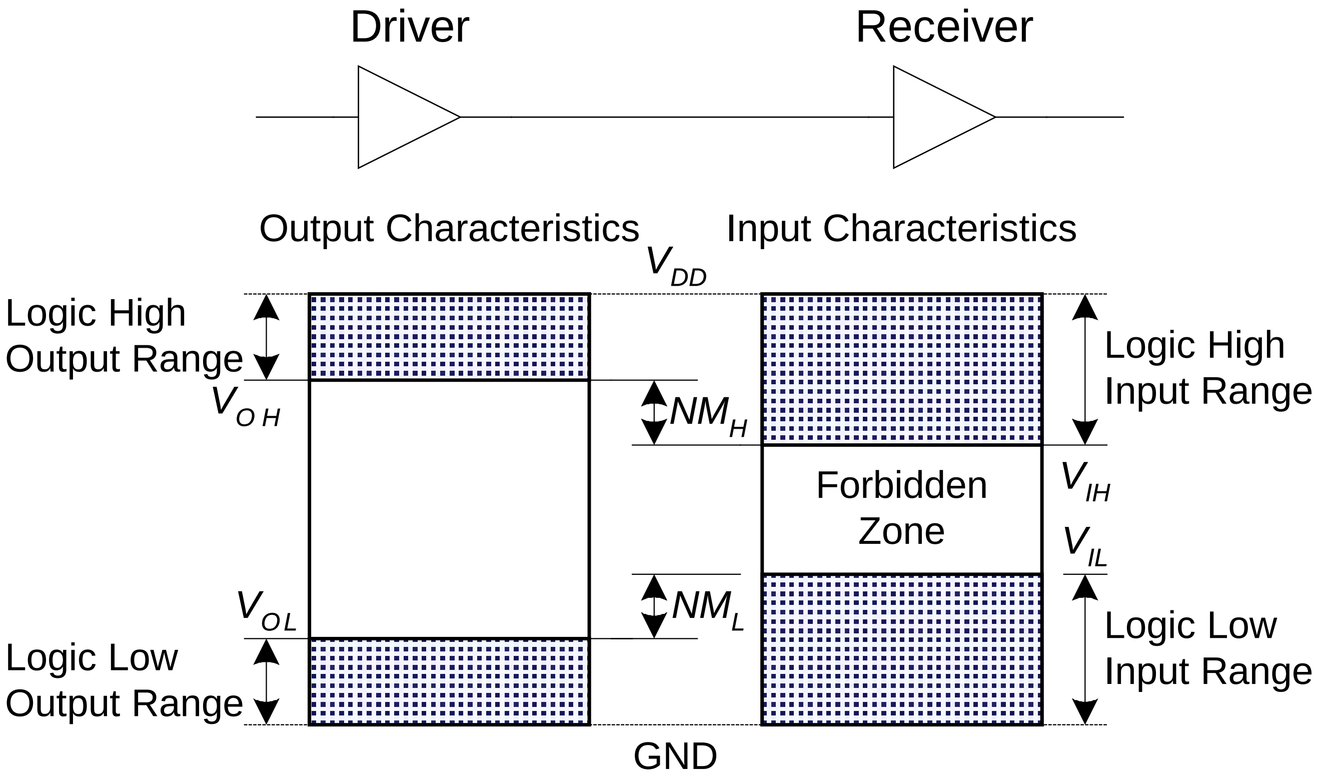
The Static Discipline

- With logically valid inputs, every circuit element must produce logically valid outputs
- Use limited ranges of voltages to represent discrete values

Noise Margins



Noise Margins

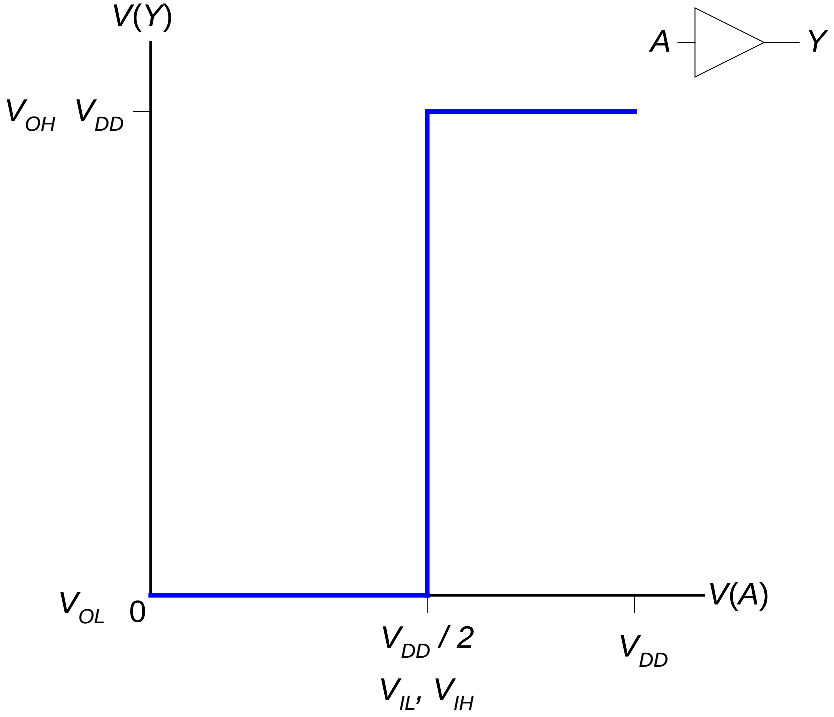


High Noise Margin: $NM_H =$

Low Noise Margin: $NM_L =$

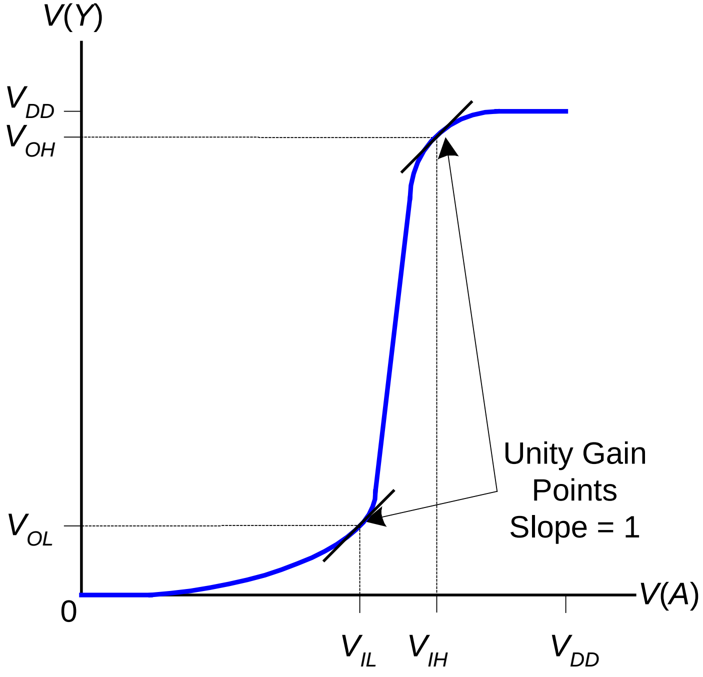
DC Transfer Characteristics

Ideal Buffer:



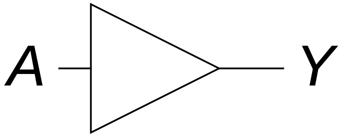
$$NM_H = NM_L = V_{DD}/2$$

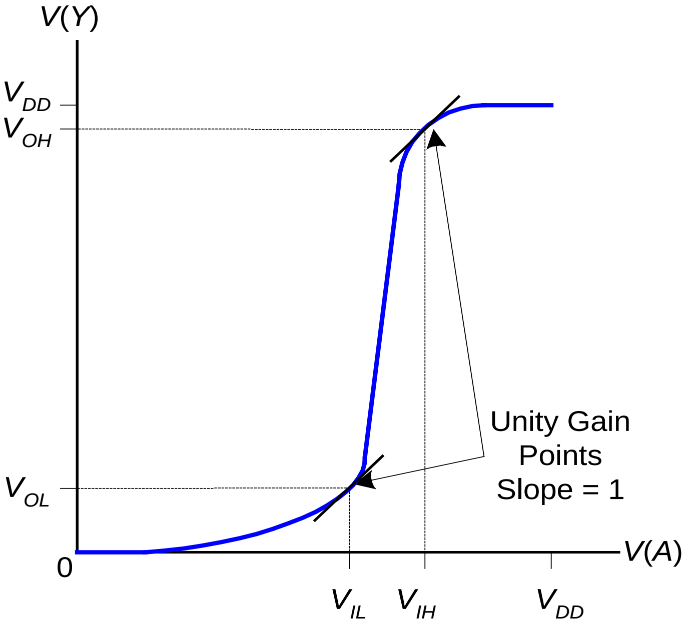
Real Buffer:

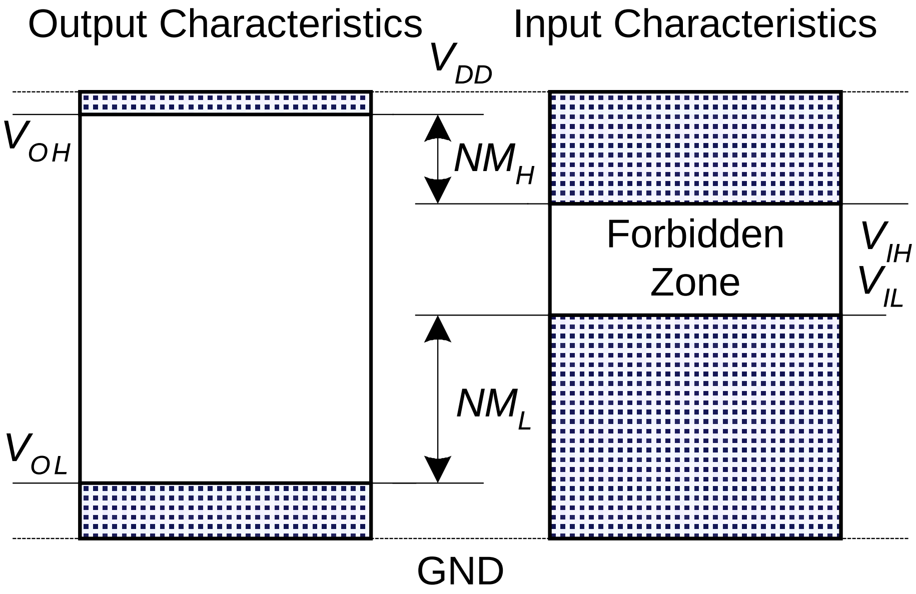


$$NM_H, NM_L < V_{DD}/2$$

DC Transfer Characteristics







V_{DD} Scaling

- In 1970's and 1980's, V_{DD} = 5 V
- V_{DD} 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

V_{DD} Scaling

- In 1970's and 1980's, V_{DD} = 5 V
- V_{DD} 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

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
LVCMOS	3.3 (3 - 3.6)	0.9	1.8	0.36	2.7

- Transistor-Transistor Logic (TTL)
- Complementary Metal-Oxide-Semiconductor (CMOS)

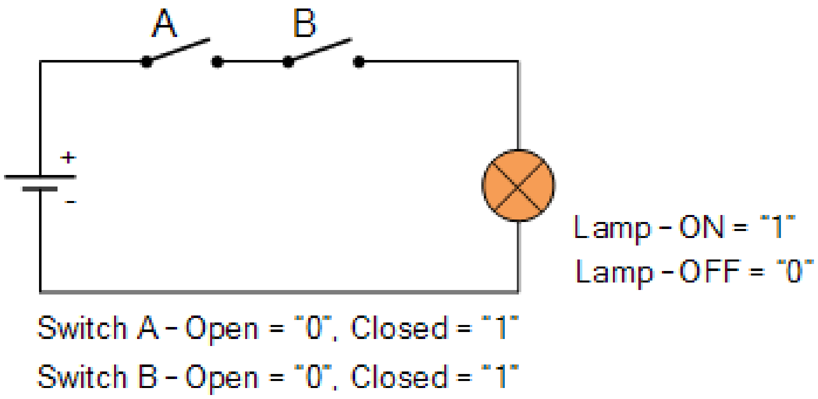
Chapter 1: From Zero to One

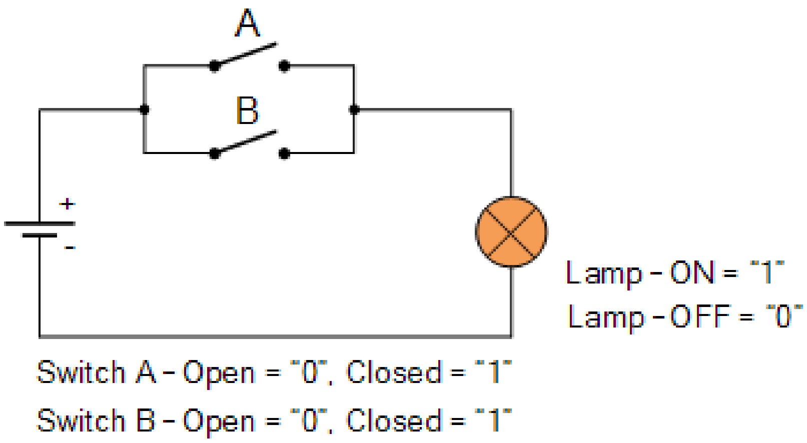
CMOS Transistors

Switch

- AND operation:
  1 X 1 = 1

- OR operation:
  0 + 0 = 0

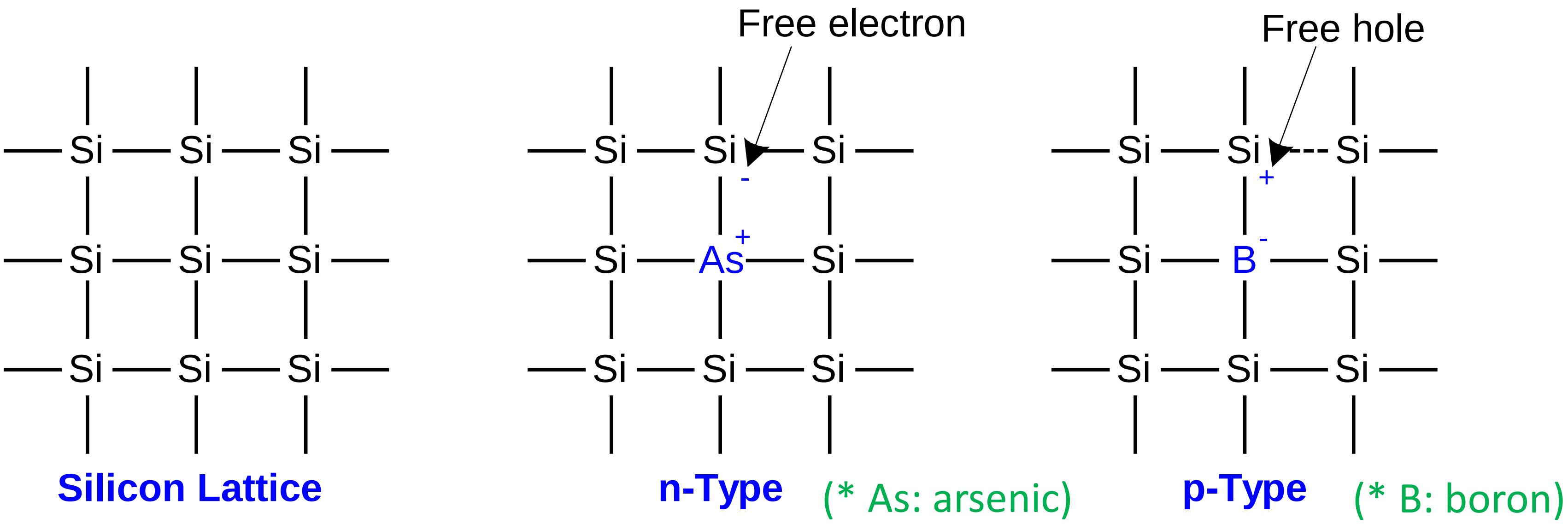

A
B
+
-
Lamp - ON = "1"
Lamp - OFF = "0"
Switch A - Open = "0", Closed = "1"
Switch B - Open = "0", Closed = "1"


A
B
+
-
Lamp - ON = "1"
Lamp - OFF = "0"
Switch A - Open = "0", Closed = "1"
Switch B - Open = "0", Closed = "1"

How can we make electronically controlled switches?

Silicon

- Pure silicon is a poor conductor (no free charges)
- Doped silicon is a good conductor (free charges)
  - n-type (free ***n***egative charges, electrons)
  - p-type (free ***p***ositive charges, holes)



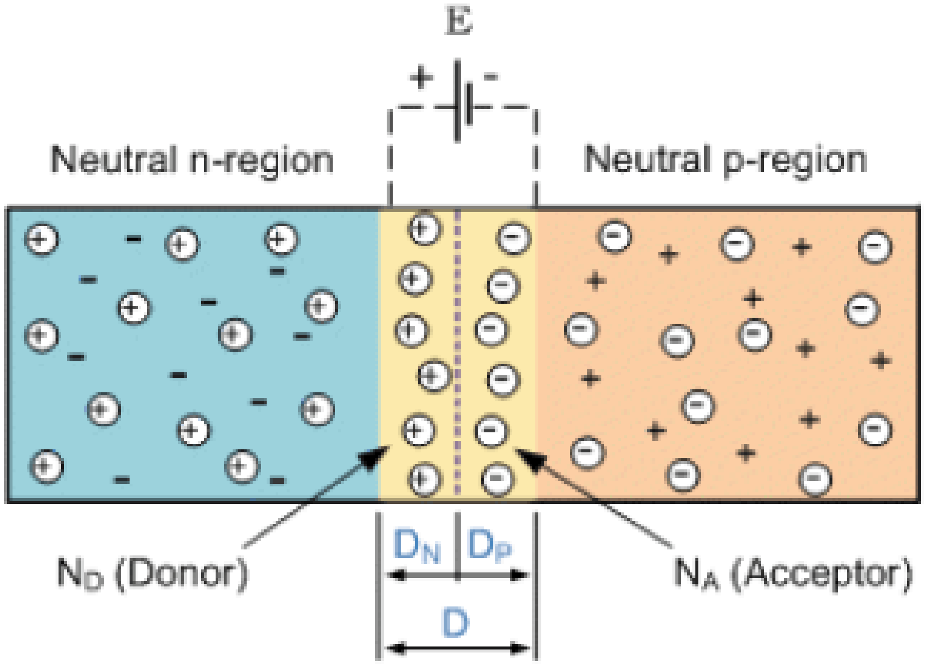
Silicon Lattice

n-Type (* As: arsenic)

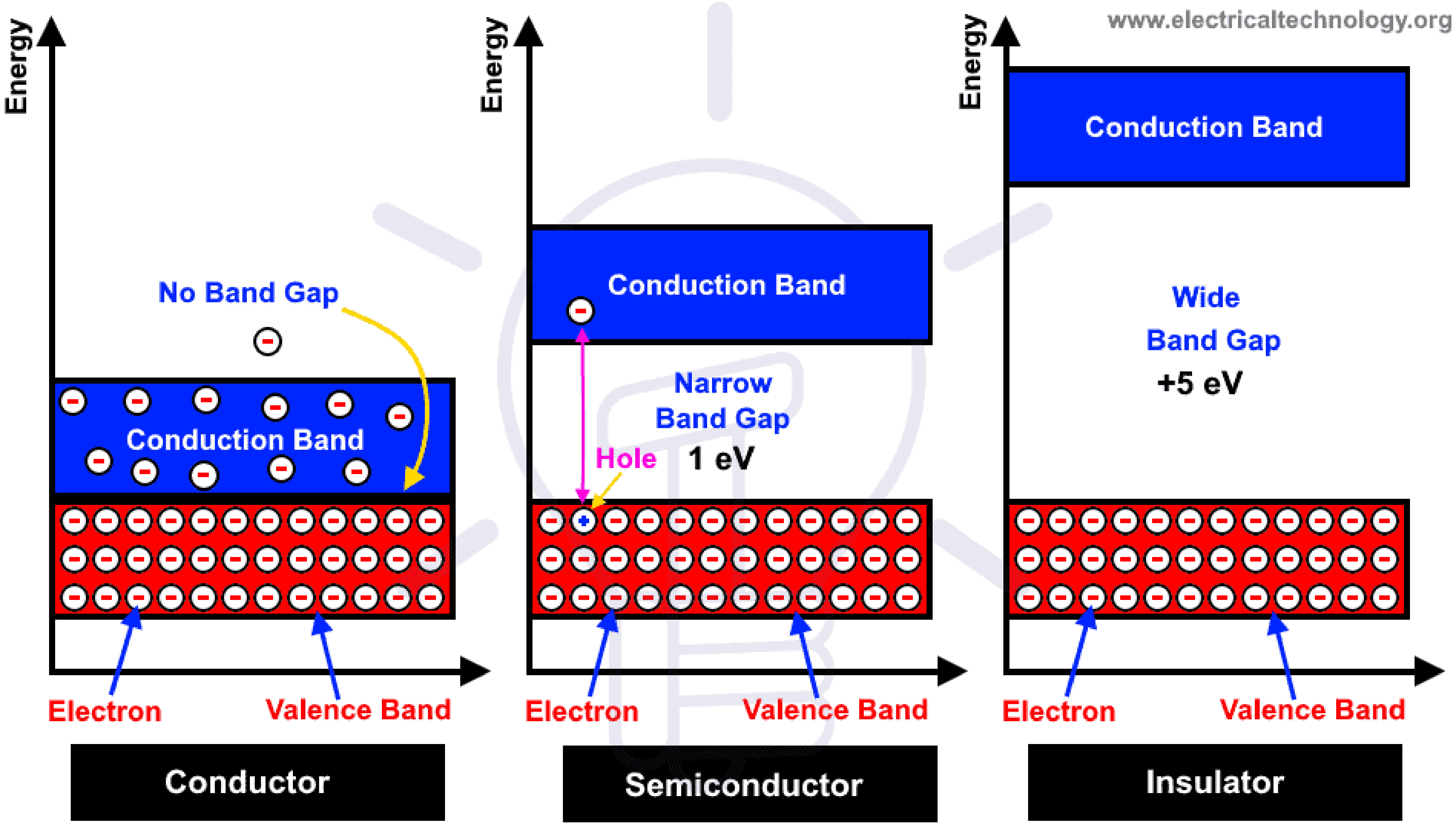
p-Type (* B: boron)

Semiconductor

- **Feature**
  - Insulator at low temperature
  - Conductor if energy is given to make electrons jump from valence band to conduction band

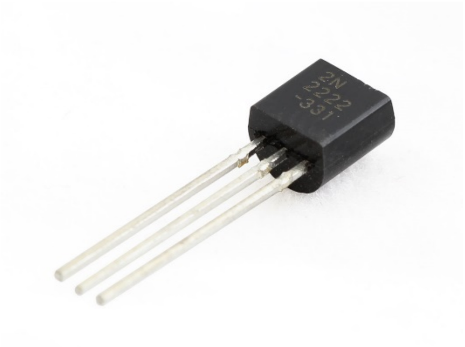


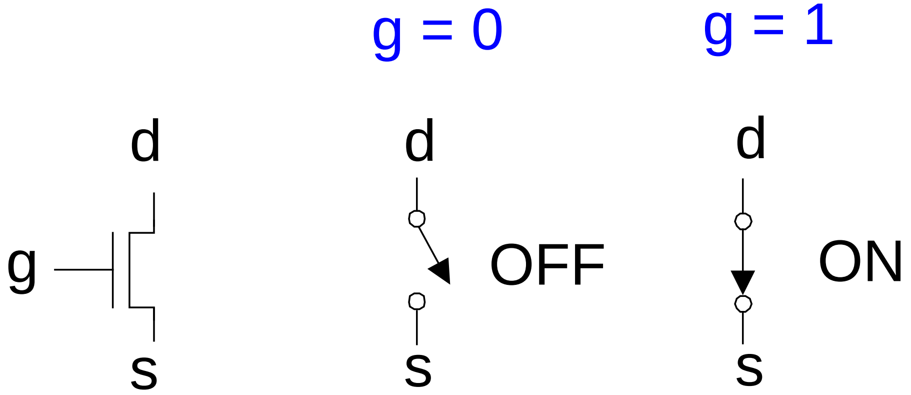
Semiconductor



Transistors

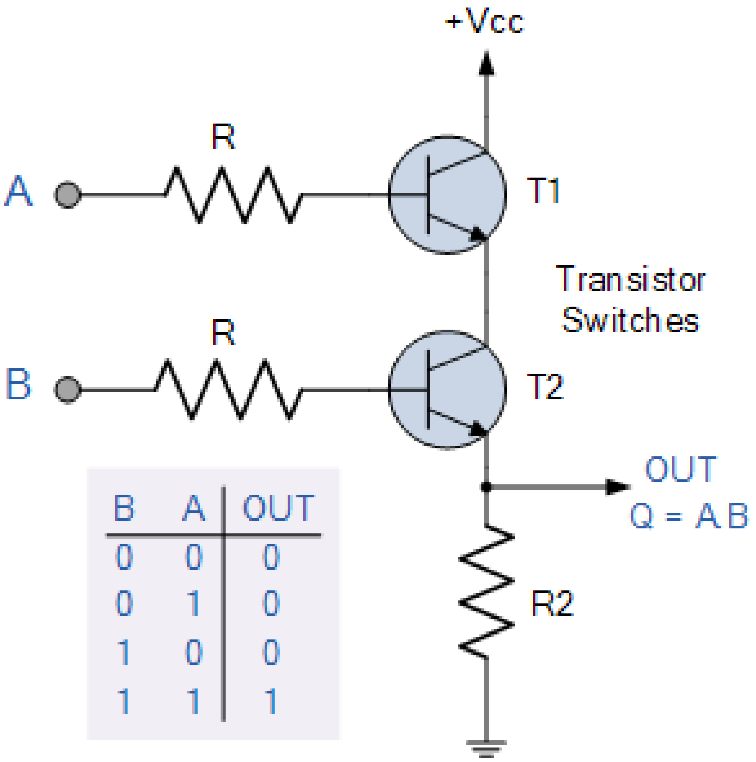
- Transistors built from silicon, a semiconductor
- Logic gates built from transistors
- 3-ported voltage-controlled switch
  - 2 ports connected depending on voltage of 3rd port
  - d and s are connected (ON) when *g* is 1





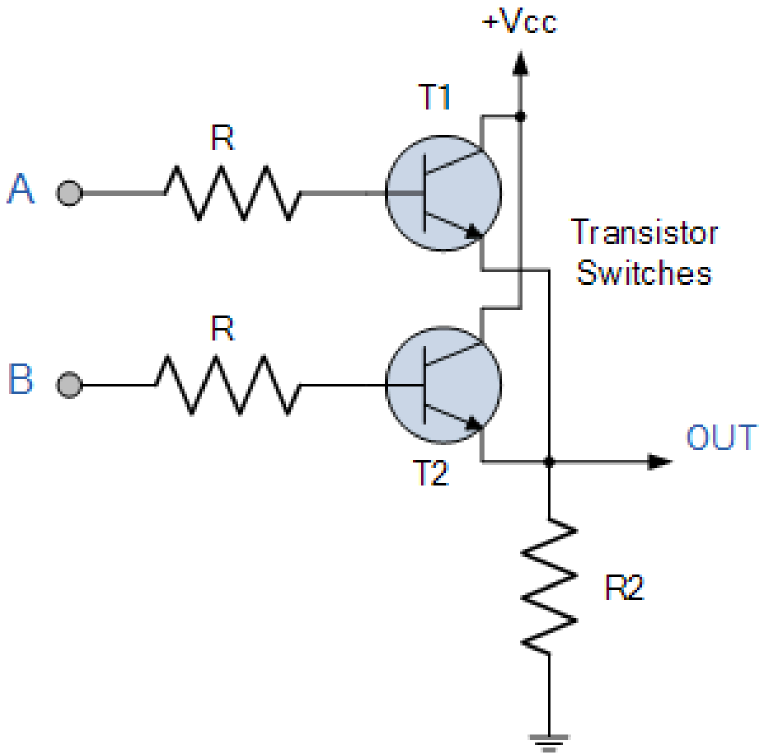
Logic Gates with Transistors

- AND gate

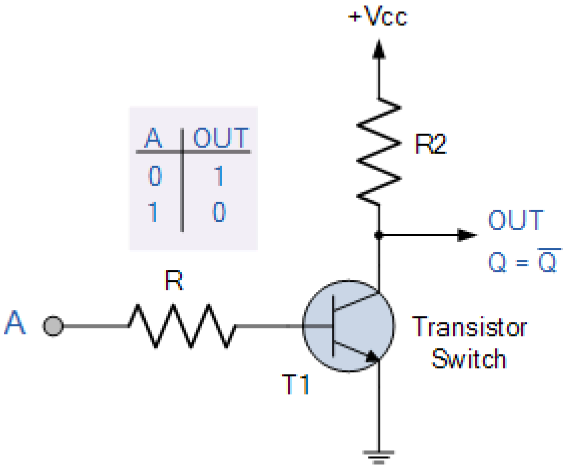


B	A	OUT
0	0	0
0	1	0
1	0	0
1	1	1

- OR gate

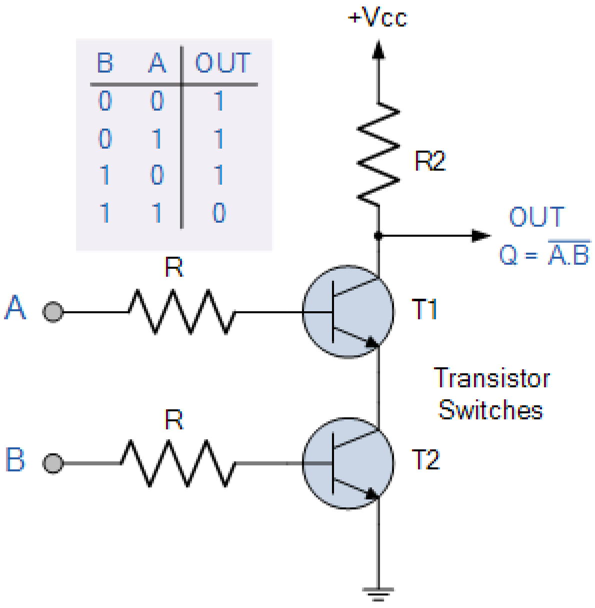


Logic Gates with Transistors



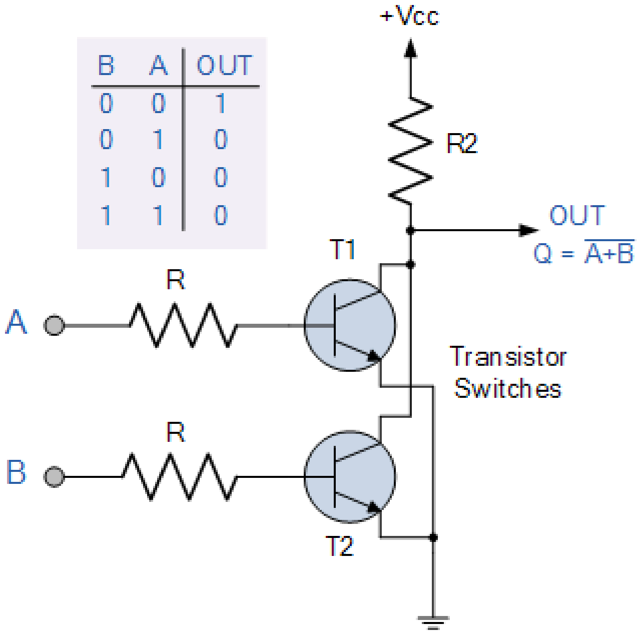
A	OUT
0	1
1	0

NOT



B	A	OUT
0	0	1
0	1	1
1	0	1
1	1	0

NAND

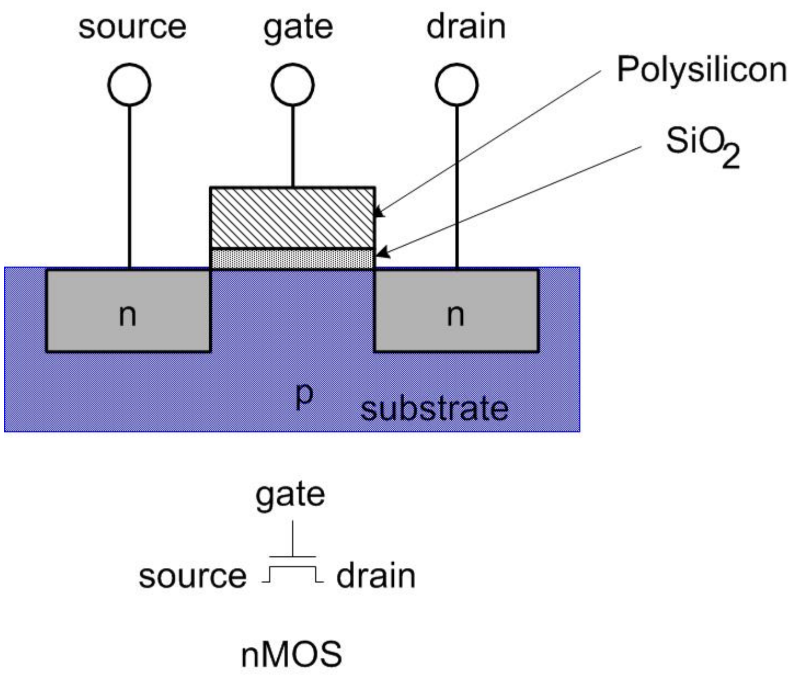


B	A	OUT
0	0	1
0	1	0
1	0	0
1	1	0

NOR

MOS Transistors

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



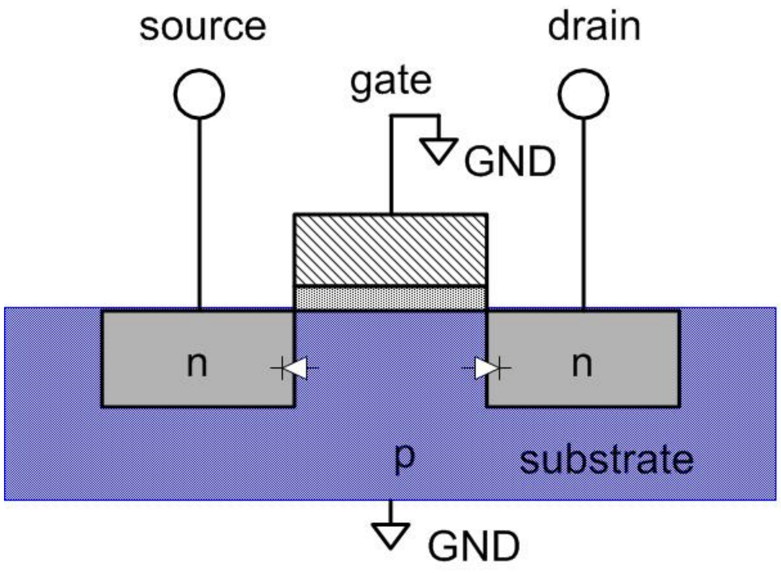
Transistors: nMOS

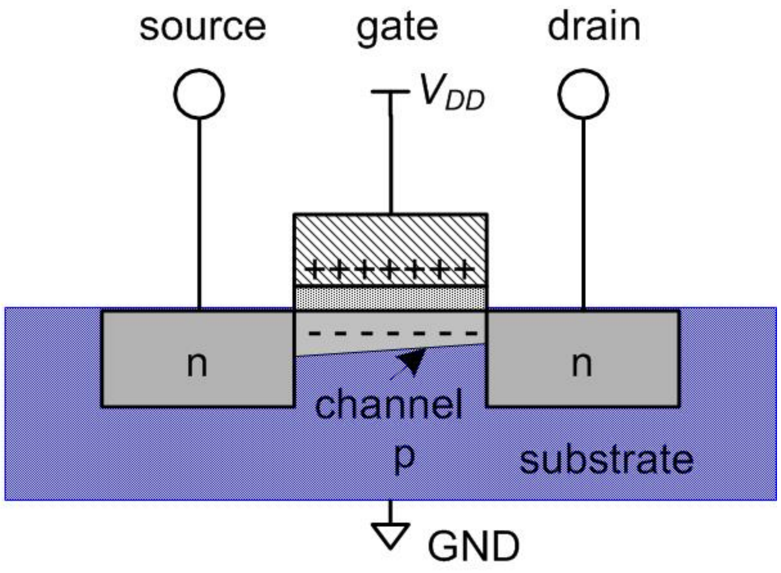
Gate = 0

OFF (no connection between source and drain)

Gate = 1

ON (channel between source and drain)

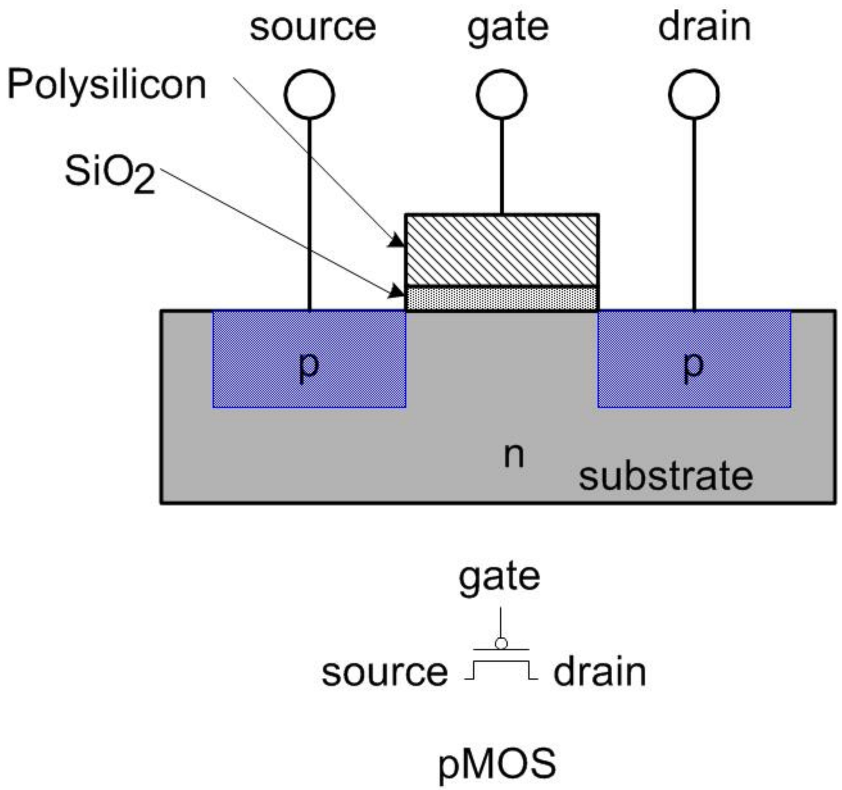




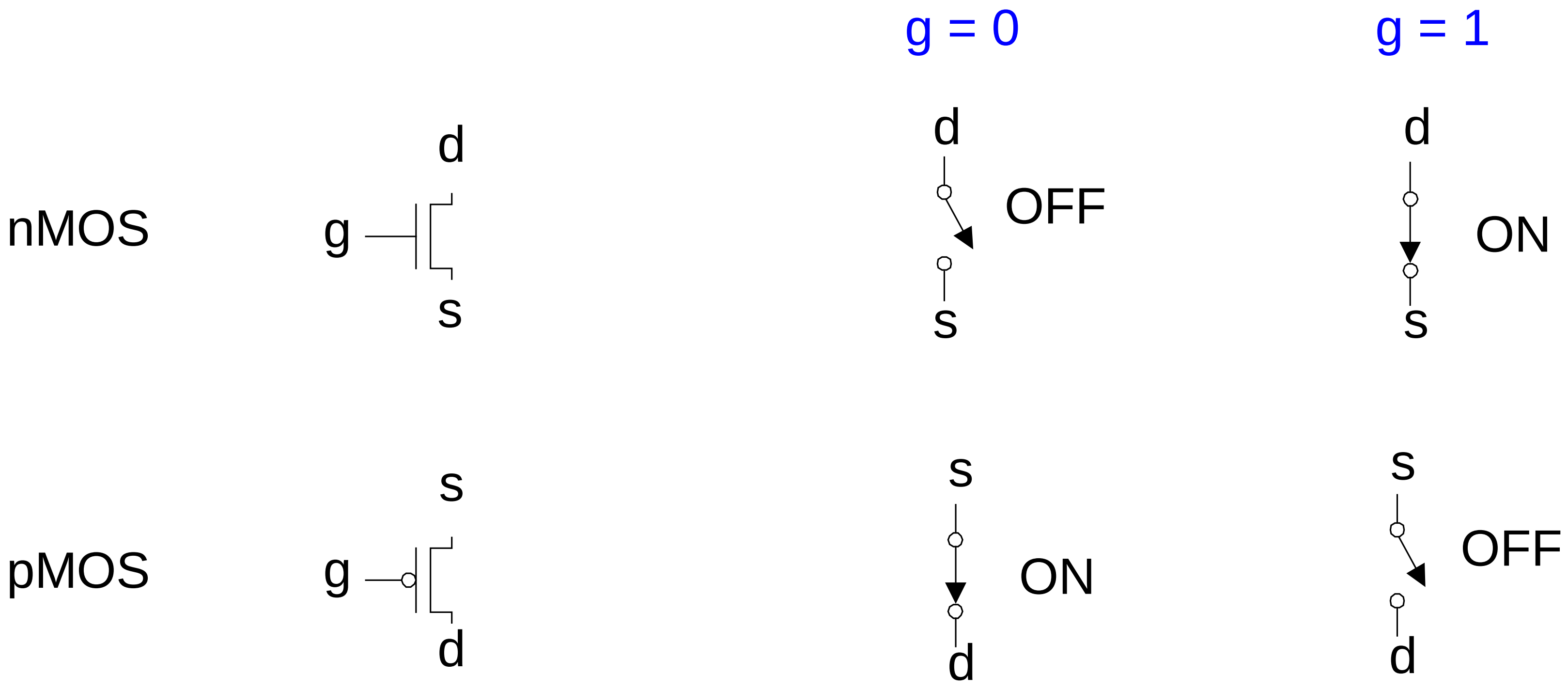
Transistors: pMOS

pMOS transistor is opposite

- **ON** when **Gate = 0**
- **OFF** when **Gate = 1**



Transistor Function



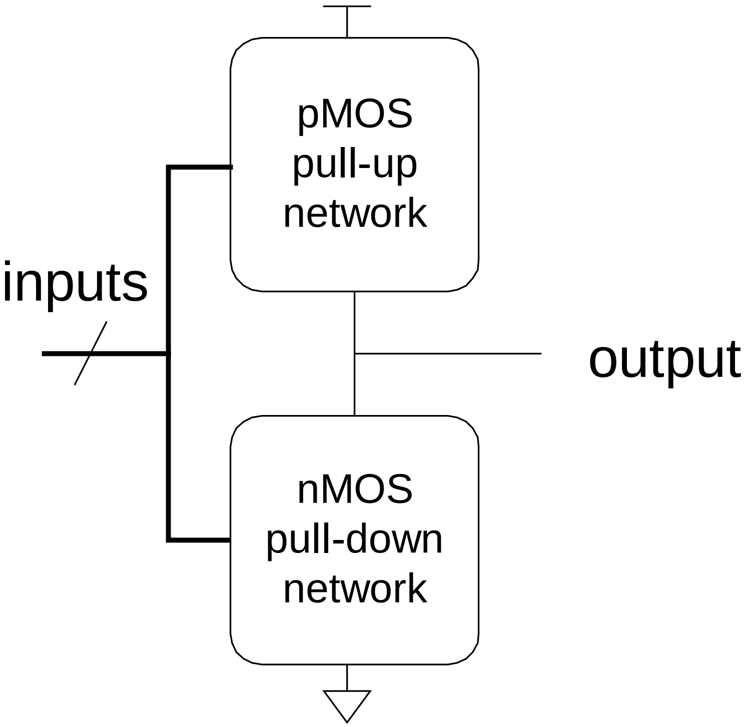
	g = 0	g = 1
nMOS	OFF	ON
pMOS	ON	OFF

Chapter 1: From Zero to One

Gates from Transistors

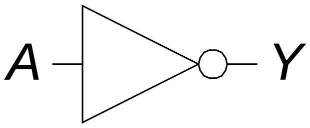
Transistor Function

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



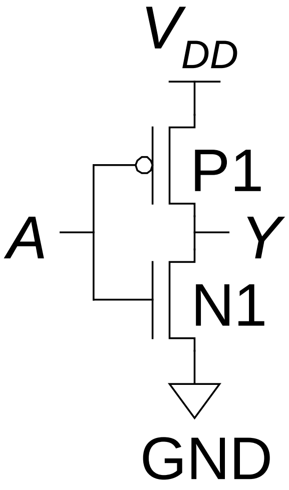
CMOS Gates: NOT Gate

NOT



$Y = \overline{A}$

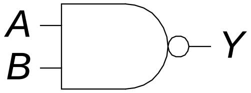
A	*Y*
0	1
1	0



A	P1	N1	*Y*
0			
1			

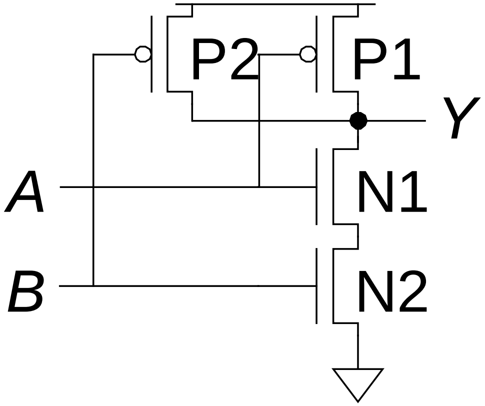
CMOS Gates: NAND Gate

NAND



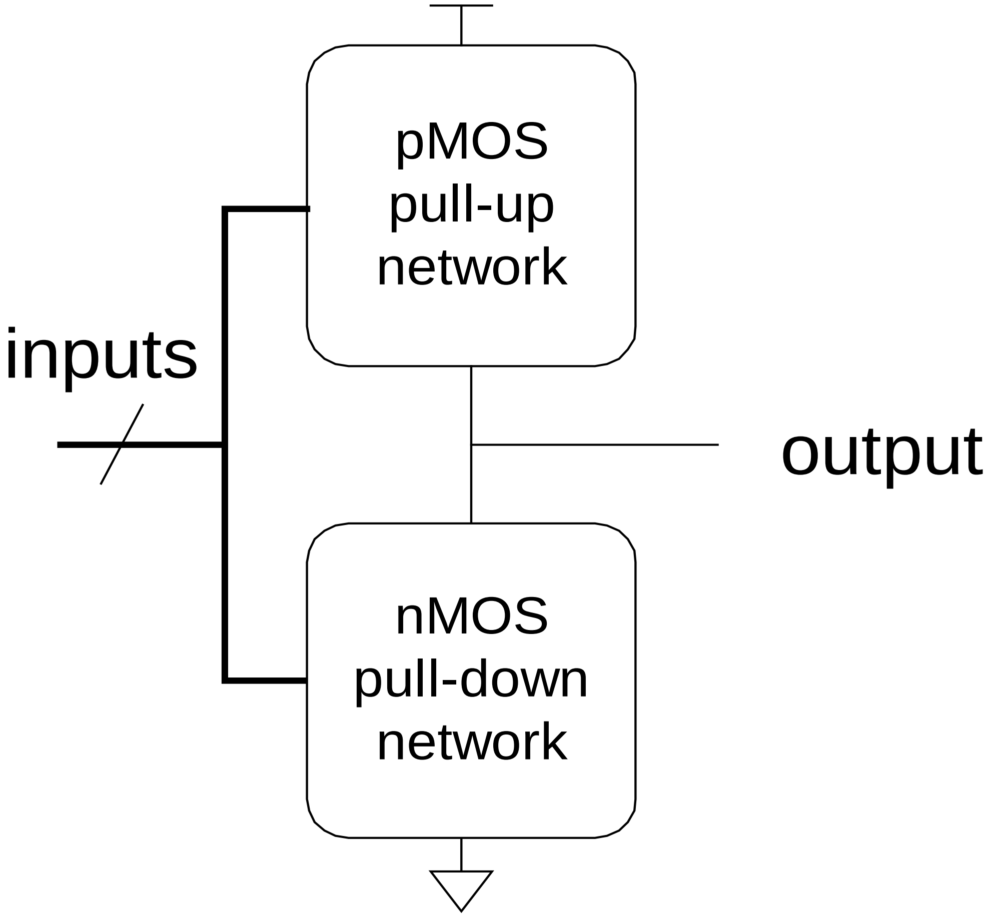
$Y = \overline{AB}$

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



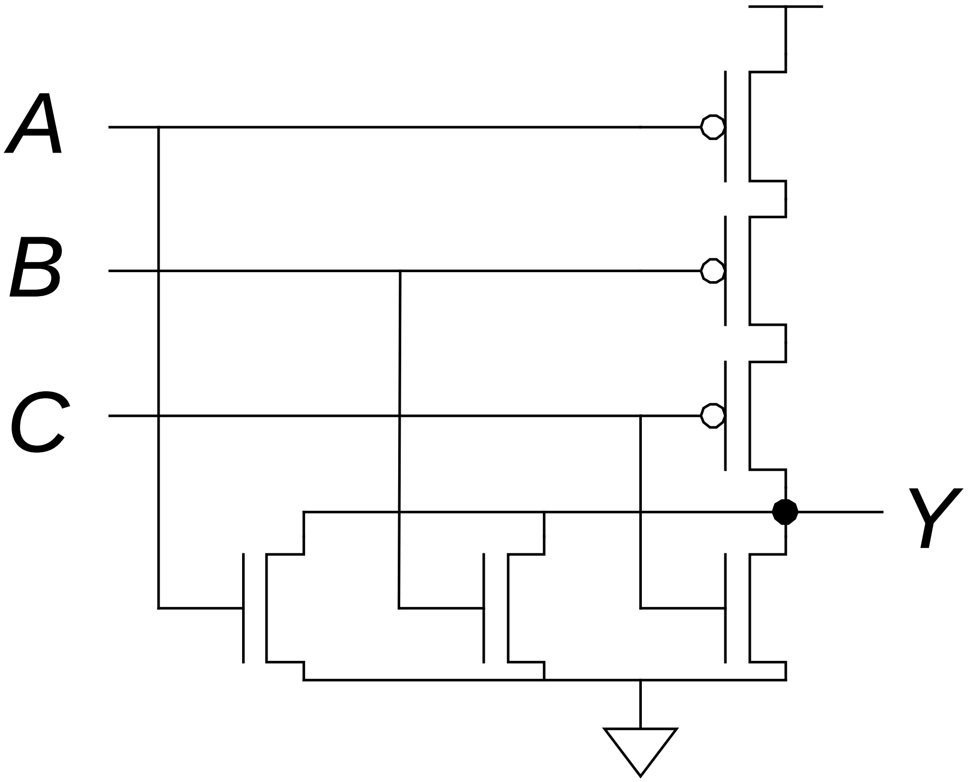
A	*B*	**P1**	**P2**	**N1**	**N2**	***Y***
0	0					
0	1					
1	0					
1	1					

CMOS Gate Structure



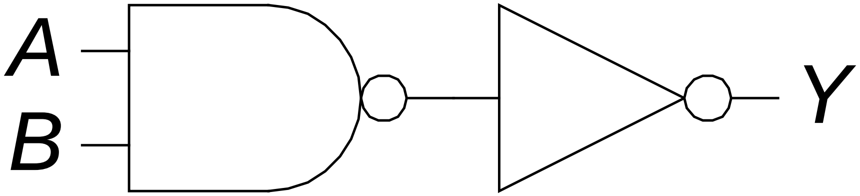
NOR3 Gate

How do you build a three-input NOR gate?



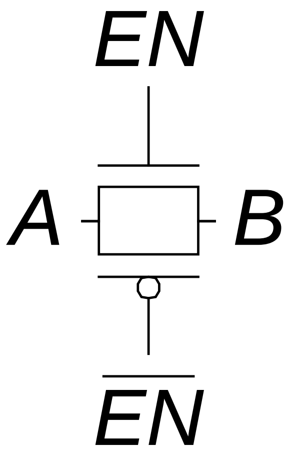
AND2 Gate

How do you build a two-input AND gate?



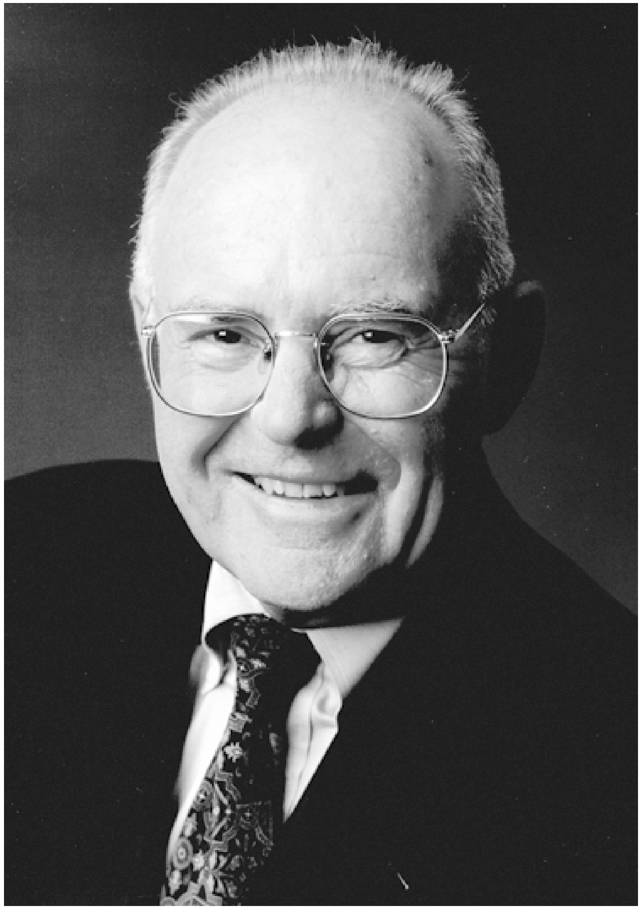
Transmission Gates

- nMOS pass 1's poorly
- pMOS pass 0's poorly
- The parallel combination of the two passes, or a *transmission gate* is a better switch
  - passes both 0 and 1 well
- When $EN = 1$, the switch is ON:
  - $\overline{EN} = 0$ and *A* is connected to *B*
- When $EN = 0$, the switch is OFF:
  - *A* is not connected to *B*

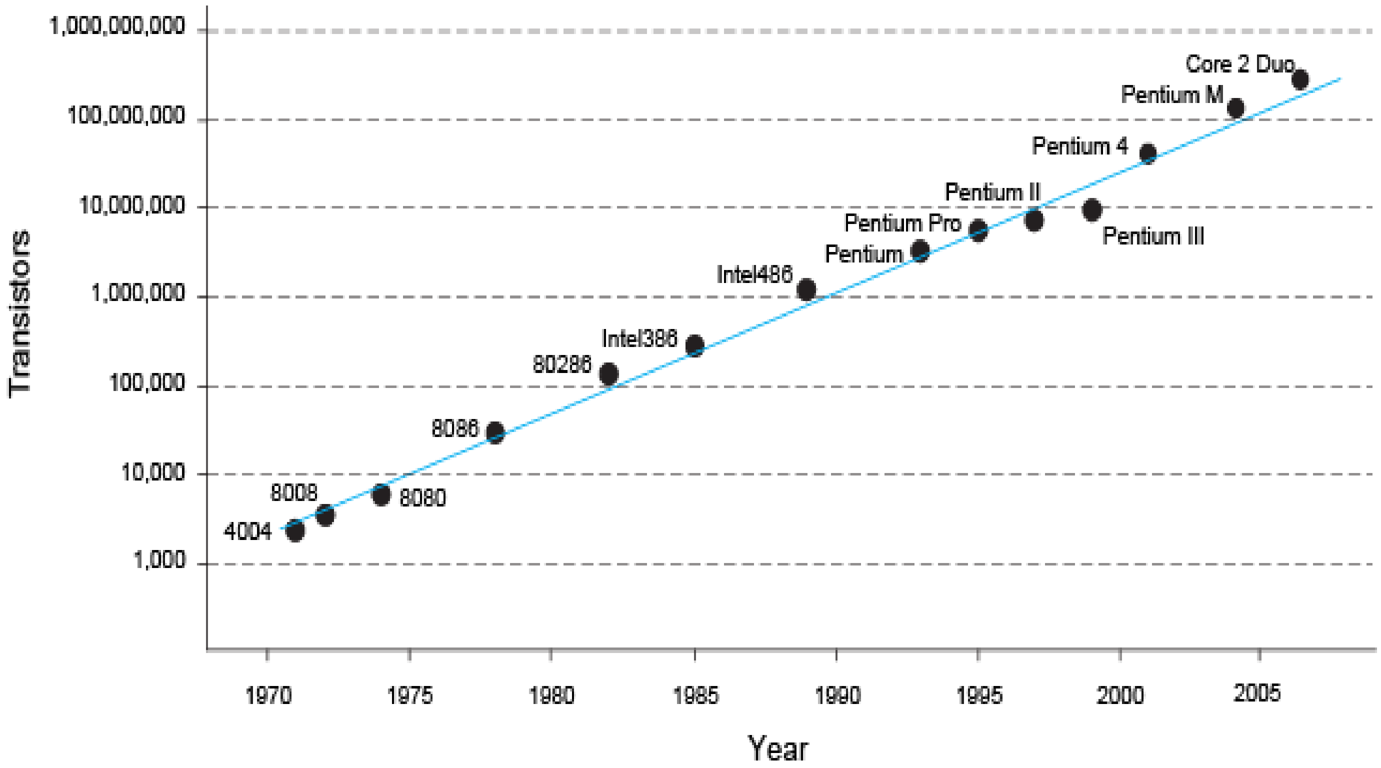


Gordon Moore, 1929-

- 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.
- Corollaries: transistors get faster and lower power



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 Cringely, Infoworld)

– Robert Cringley

Chapter 1: From Zero to One

Power Consumption

Power Consumption

Power = Energy consumed per unit time

- Dynamic power consumption
- Static power consumption

Dynamic Power Consumption

- **Power to charge transistor gate capacitances**
  - Energy required to charge a capacitance, *C*, to V_{DD} is CV_{DD}^2
  - Circuit running at frequency *f* (*f* cycles per second)
  - Capacitor is charged α times per cycle (discharging from 1 to 0 is free)
- **Dynamic power consumption:**

$$\boxed{P_{dynamic} = \alpha C V_{DD}^2 f}$$

Static Power Consumption

- Power consumed when no gates are switching
- Caused by the *quiescent supply current*, I_{DD} (also called the *leakage current*)
- Static power consumption:

$$\boxed{P_{static} = I_{DD}V_{DD}}$$

Power Consumption Example

- Estimate the power consumption of a mobile phone running Angry Birds
  - V_{DD} = 0.8 V
  - C = 5 nF (5×10^{-9} Farads)
  - f = 2 GHz (2×10^{9} Hertz)
  - α = 0.1
  - I_{DD} = 100 mA

$$\boldsymbol{P = \alpha C V_{DD}^2 f + I_{DD} V_{DD}}$$

$$= (0.1)(5\text{ nF})(0.8\text{ V})^2(2\text{ GHz}) + (100\text{ mA})(0.8\text{ V})$$

$$\mathbf{= (0.64 + 0.08)\ W \approx 0.72\ W}$$

Power Consumption Example

- If the phone has a 8 W-hr battery, estimate its battery life sitting idle in your pocket.
  - V_{DD} = 0.8 V
  - I_{DD} = 100 mA

$P_{static} = I_{DD}V_{DD}$ = 0.08 W

Battery life = Capacity / Consumption

= (8 W-hr) / (0.08 W) = 100 hr (4 days)