FROM ZERO TO ONE

Chapter 1

***Digital Design and Computer Architecture*, 2nd Edition**

David Money Harris and Sarah L. Harris

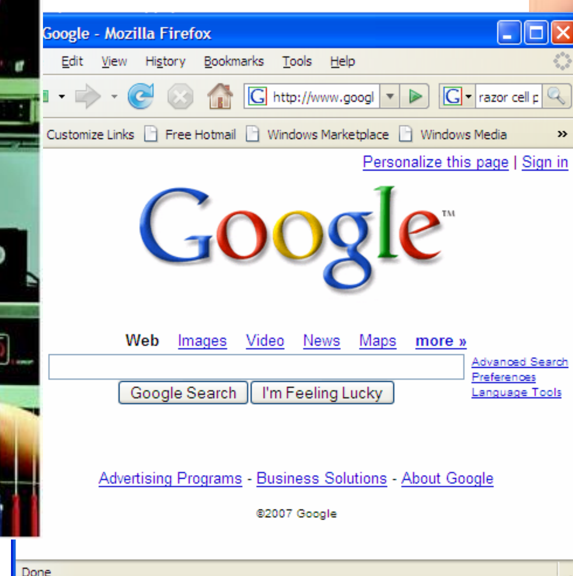
Chapter 1 :: Topics

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

FROM ZERO TO ONE

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



ELSEVIER

The Game Plan

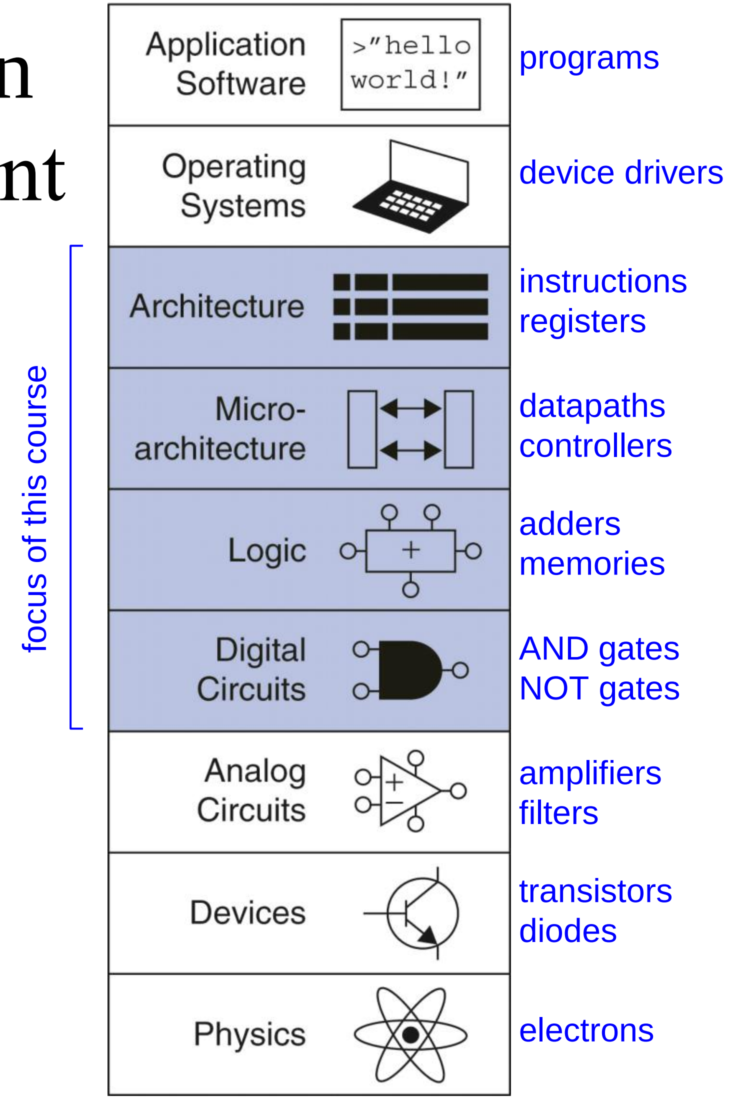
- Purpose of course:
  - Understand what's under the hood of a computer
  - Learn the principles of digital design
  - Learn to systematically debug increasingly complex designs
  - Design and build a microprocessor

The Art of Managing Complexity

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

Abstraction

- Hiding details when they aren't important



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

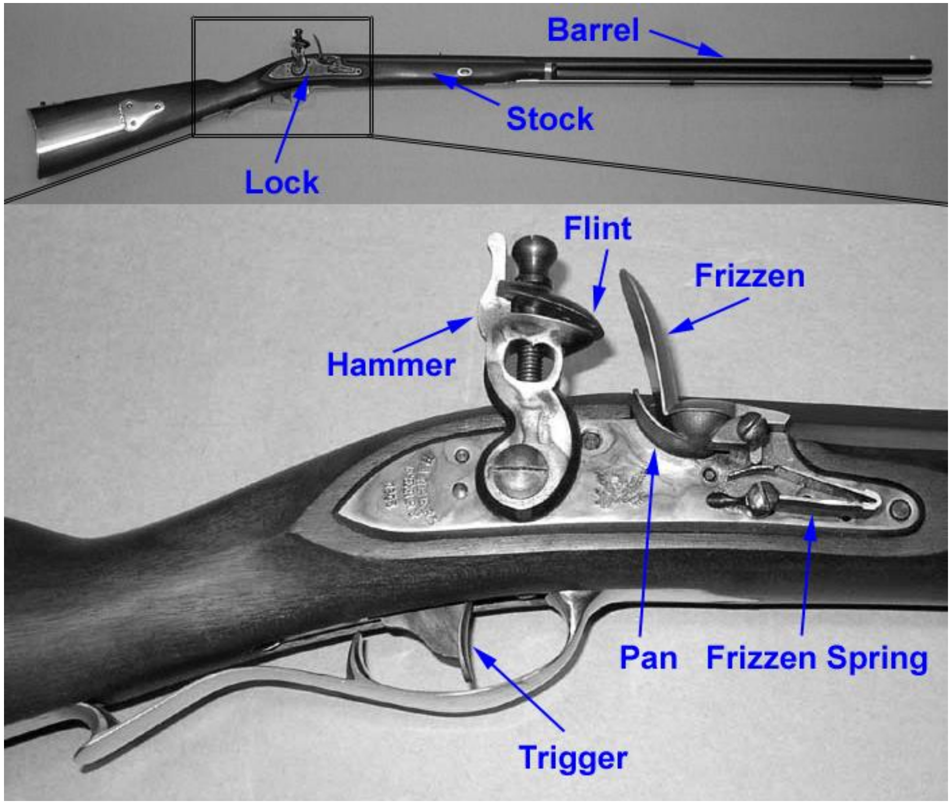
- **Hierarchy**
- **Modularity**
- **Regularity**

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

Example: The Flintlock Rifle

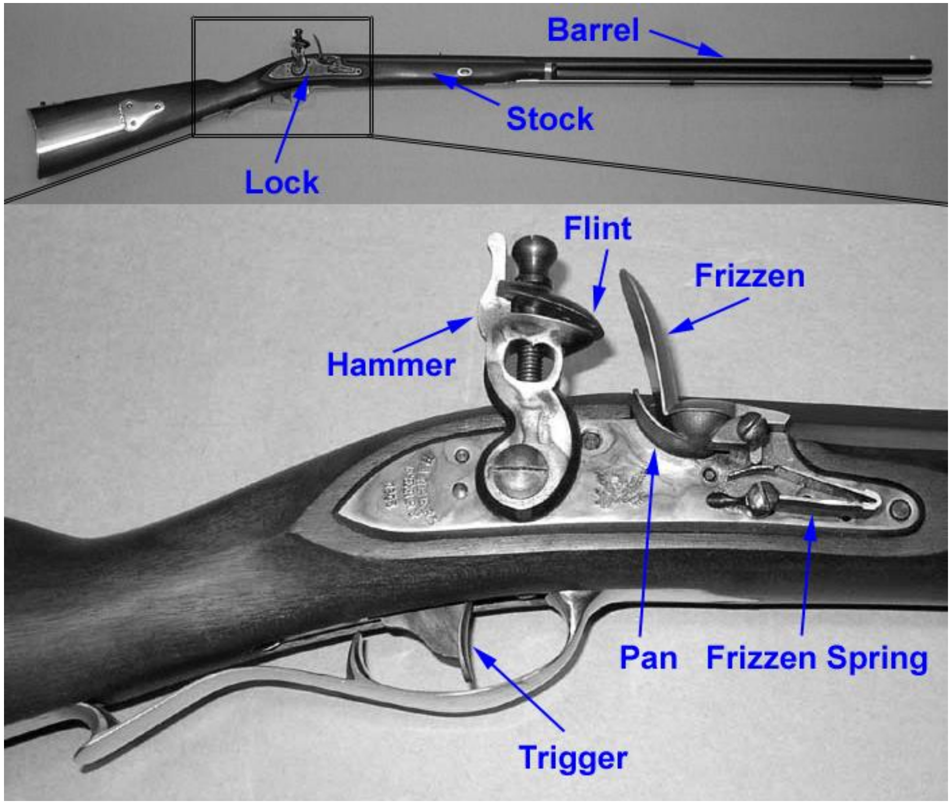
- **Hierarchy**
  - **Three main modules:** lock, stock, and barrel
  - **Submodules of lock:** hammer, flint, frizzen, etc.



ELSEVIER

Example: The Flintlock Rifle

- **Modularity**
  - **Function of stock:** mount barrel and lock
  - **Interface of stock:** length and location of mounting pins
- **Regularity**
  - Interchangeable parts

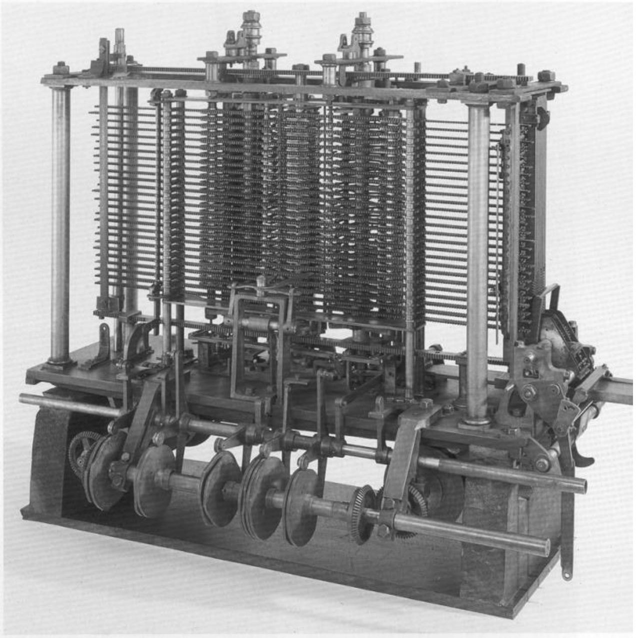


The Digital Abstraction

- 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)
- Babbage died before it was finished

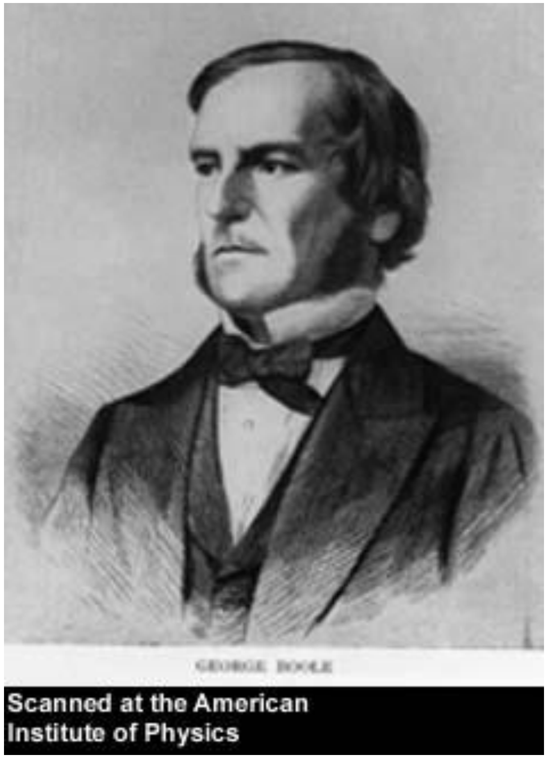


Digital Discipline: Binary Values

- **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***: *B*inary dig*it*

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



Number Systems

- Decimal numbers

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

$5374_{10} =$

- Binary numbers

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

$1101_2 =$

Number Systems

- Decimal numbers

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

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

five thousands | three hundreds | seven tens | four ones

- Binary numbers

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

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

one eight | one four | no two | one one

Powers of Two

- $2^0 =$
- $2^1 =$
- $2^2 =$
- $2^3 =$
- $2^4 =$
- $2^5 =$
- $2^6 =$
- $2^7 =$
- $2^8 =$
- $2^9 =$
- $2^{10} =$
- $2^{11} =$
- $2^{12} =$
- $2^{13} =$
- $2^{14} =$
- $2^{15} =$

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

- Decimal to binary conversion:
  - Convert 47_{10} to binary

Number Conversion

- Decimal to binary 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
  - $32\times1 + 16\times0 + 8\times1 + 4\times1 + 2\times1 + 1\times1 = 101111_2$

FROM ZERO TO ONE

ELSEVIER

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:

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] = $[000_2$ to $111_2]$**

Hexadecimal Numbers

Hex Digit	Decimal Equivalent	Binary Equivalent
0	0	
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
A	10	
B	11	
C	12	
D	13	
E	14	
F	15	

Hexadecimal Numbers

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

- Hexadecimal to decimal conversion:
  - Convert 0x4AF to decimal

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 significant bit

least significant bit

- Bytes & Nibbles

byte

10010110

nibble

- Bytes

CEBF9AD7

most significant byte

least significant byte

Large Powers of Two

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

Estimating Powers of Two

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

- How many values can a 32-bit variable represent?

Estimating Powers of Two

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

  $2^4 \times 2^{20} \approx 16$ **million**

- How many values can a 32-bit variable represent?

  $2^2 \times 2^{30} \approx 4$ **billion**

Addition

- Decimal

```
   3734
+  5168
-------
```

- Binary

```
   1011
+  0011
-------
```

Addition

- Decimal

```
     11  ← carries
   3734
 + 5168
 ------
   8902
```

- Binary

```
     11  ← carries
   1011
 + 0011
 ------
   1110
```

Binary Addition Examples

- Add the following 4-bit binary numbers

```
   1001
+  0101
-------
```

- Add the following 4-bit binary numbers

```
   1011
+  0110
-------
```


ELSEVIER

Binary Addition Examples

- Add the following 4-bit binary numbers

```
     1
  1001
+ 0101
------
  1110
```

- Add the following 4-bit binary numbers

```
  111
  1011
+ 0110
------
 10001
```

Overflow!

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

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
  - Negative number: sign bit = 1

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

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

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

  +6 =

  - 6 =

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

Sign/Magnitude Numbers

- 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

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

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

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

  +6 = **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:

```
   1110
 + 0110
 ------
  10100 (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

Two's Complement Numbers

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

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

- 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 comp number:

Two's Complement Numbers

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

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

- 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 $3_{10} = 0011_2$

"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**

  $1101 = -3_{10}$

Two’s Complement Examples

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

- What is the decimal value of 1001_2?

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

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

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

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

Two's Complement Addition

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

```
   111
   0110
+  1010
-------
  10000
```

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

```
   111
   1110
+  0011
-------
  10001
```

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:

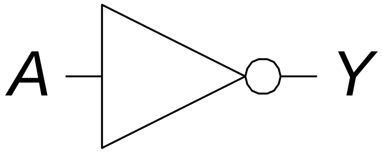
	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Unsigned									0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Two's Complement	1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111								
Sign/Magnitude		1111	1110	1101	1100	1011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

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

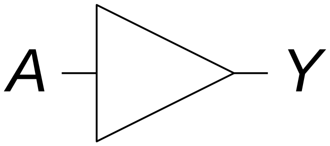
NOT



$$Y = \overline{A}$$

A	Y
0	
1	

BUF

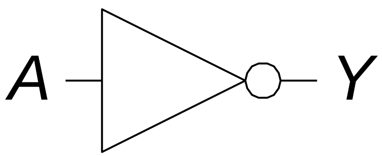


$$Y = A$$

A	Y
0	
1	

Single-Input Logic Gates

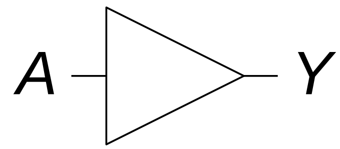
NOT



$$Y = \overline{A}$$

A	Y
0	1
1	0

BUF

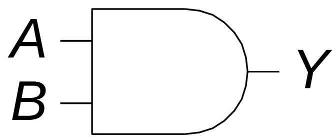


$$Y = A$$

A	Y
0	0
1	1

Two-Input Logic Gates

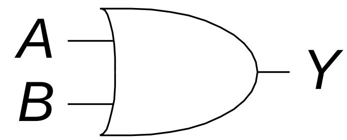
AND



$Y = AB$

A	B	Y
0	0	
0	1	
1	0	
1	1	

OR

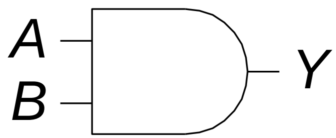


$Y = A + B$

A	B	Y
0	0	
0	1	
1	0	
1	1	

Two-Input Logic Gates

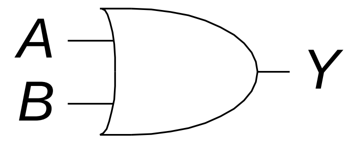
AND



$Y = AB$

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

OR



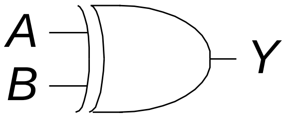
$Y = A + B$

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

FROM ZERO TO ONE

More Two-Input Logic Gates

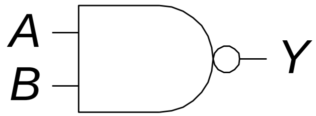
XOR



$Y = A \oplus B$

A	B	Y
0	0	
0	1	
1	0	
1	1	

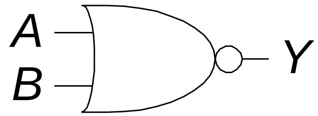
NAND



$Y = \overline{AB}$

A	B	Y
0	0	
0	1	
1	0	
1	1	

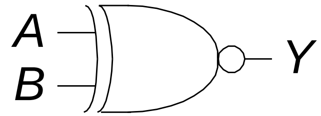
NOR



$Y = \overline{A + B}$

A	B	Y
0	0	
0	1	
1	0	
1	1	

XNOR



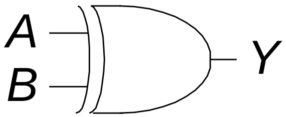
$Y = \overline{A \oplus B}$

A	B	Y
0	0	
0	1	
1	0	
1	1	

ELSEVIER

More Two-Input Logic Gates

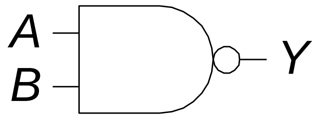
XOR



$$Y = A \oplus B$$

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

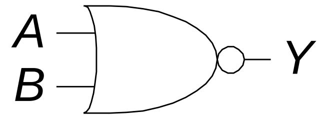
NAND



$$Y = \overline{AB}$$

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

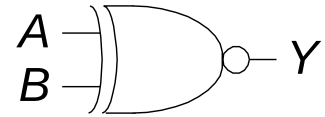
NOR



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

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

XNOR

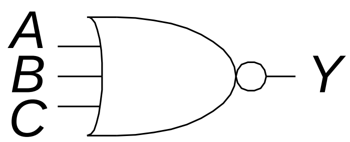


$$Y = \overline{A \oplus B}$$

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

Multiple-Input Logic Gates

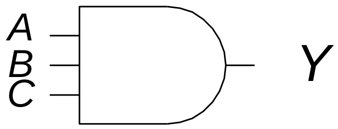
NOR3



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

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

AND3

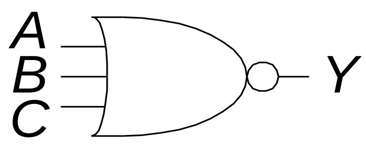


$Y = ABC$

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	

Multiple-Input Logic Gates

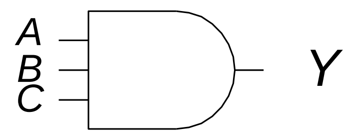
NOR3



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

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

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?

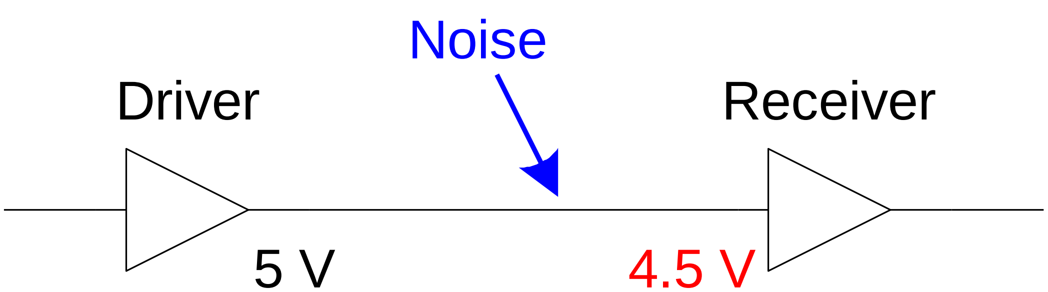
Logic Levels

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

What is Noise?

What is Noise?

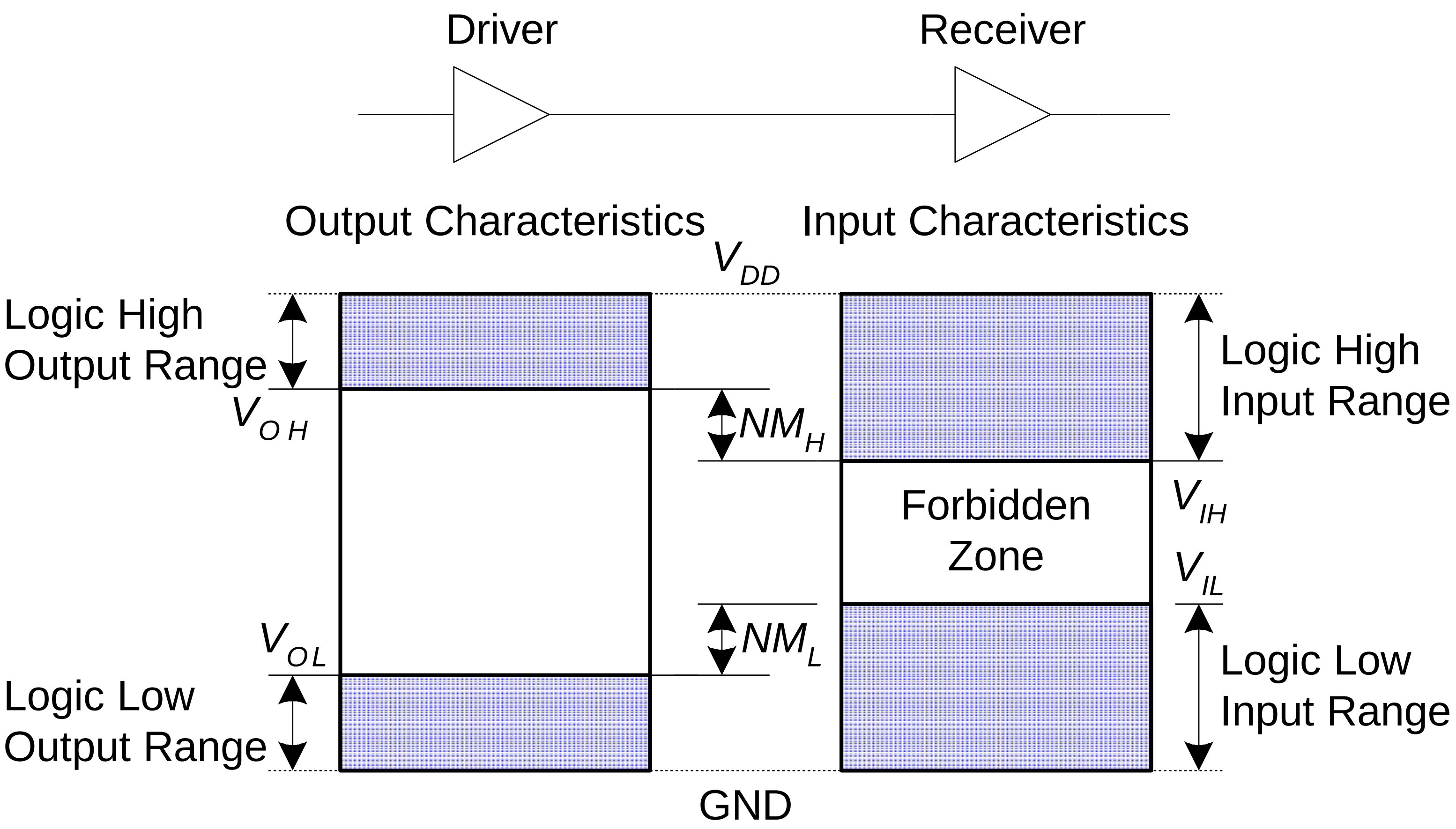
- **Anything that degrades the signal**
  - E.g., resistance, power supply noise, coupling to neighboring wires, etc.
- **Example:** a gate (driver) outputs 5 V but, because of resistance in a long wire, receiver gets 4.5 V



The Static Discipline

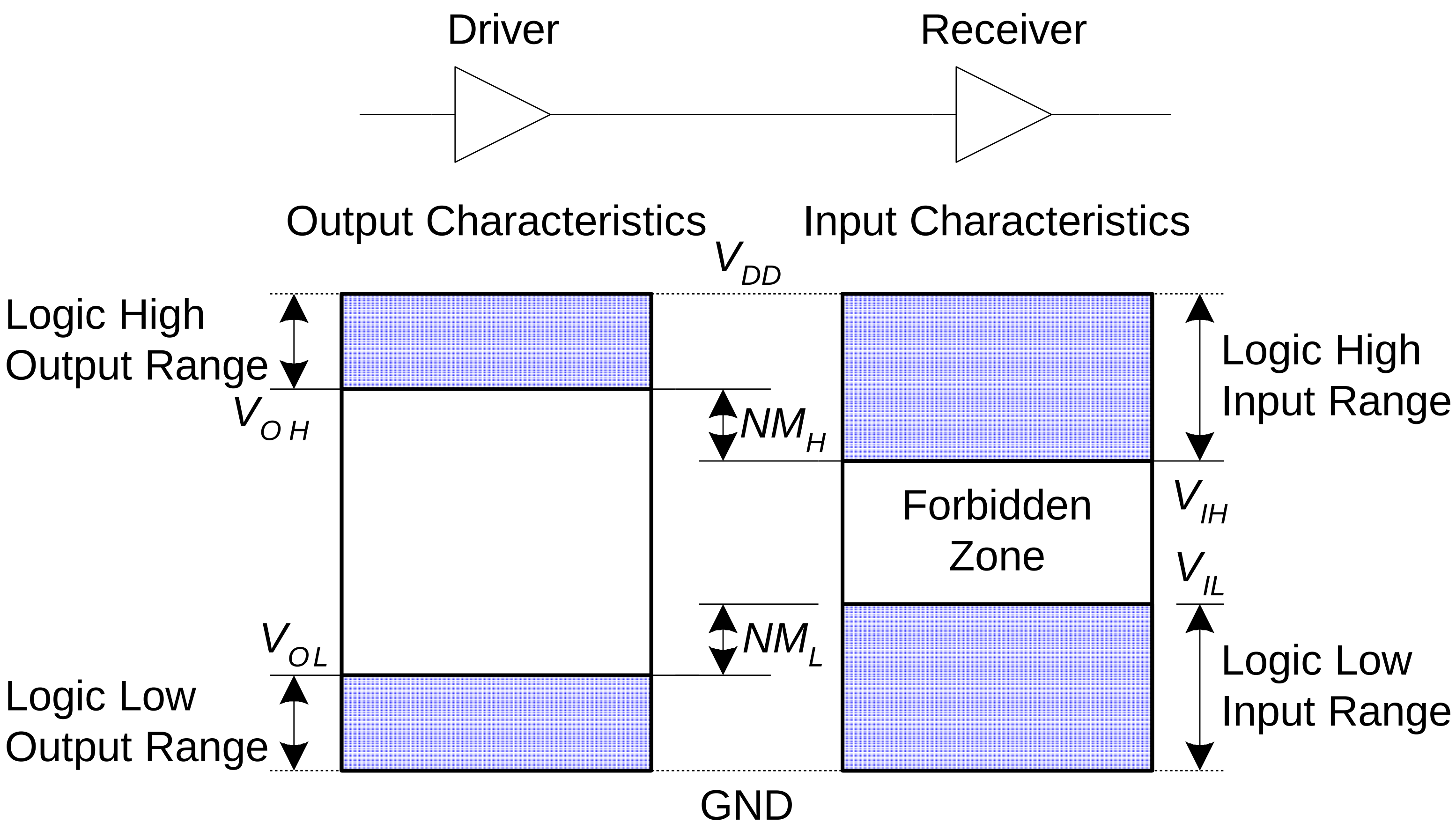
- With logically valid inputs, every circuit element must produce logically valid outputs
- Use limited ranges of voltages to represent discrete values

Logic Levels



FROM ZERO TO ONE

Noise Margins

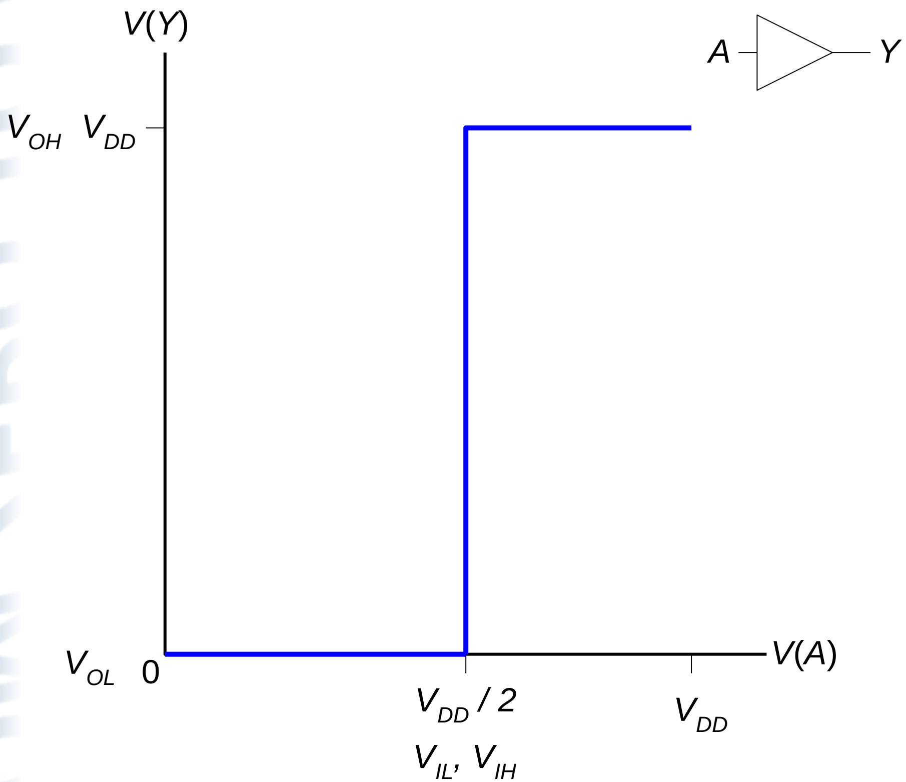


$$NM_H = V_{OH} - V_{IH}$$

$$NM_L = V_{IL} - V_{OL}$$

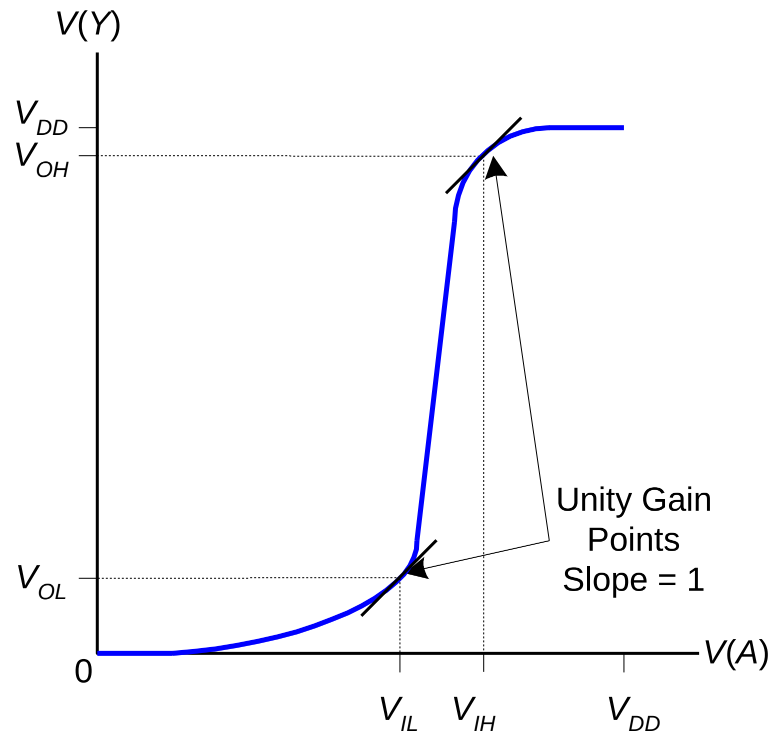
DC Transfer Characteristics

Ideal Buffer:



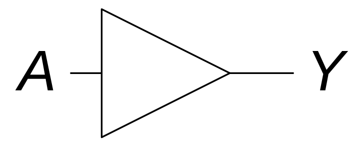
$NM_H = NM_L = V_{DD}/2$

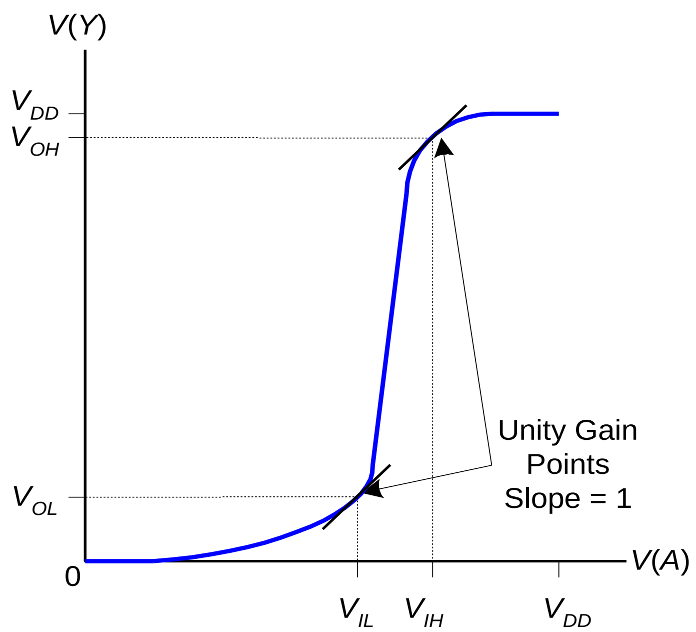
Real Buffer:

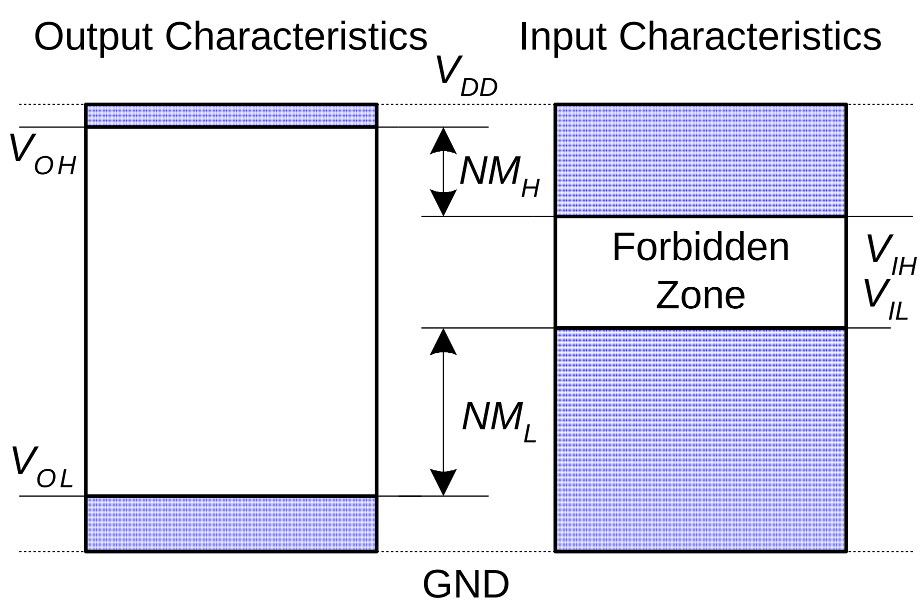


$NM_H , NM_L < V_{DD}/2$

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

Chips operate because they contain magic smoke

Proof:

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

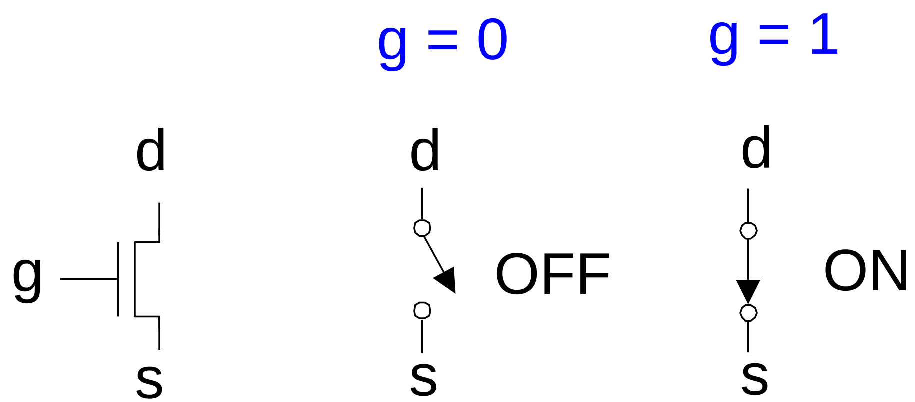
FROM ZERO TO ONE

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

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

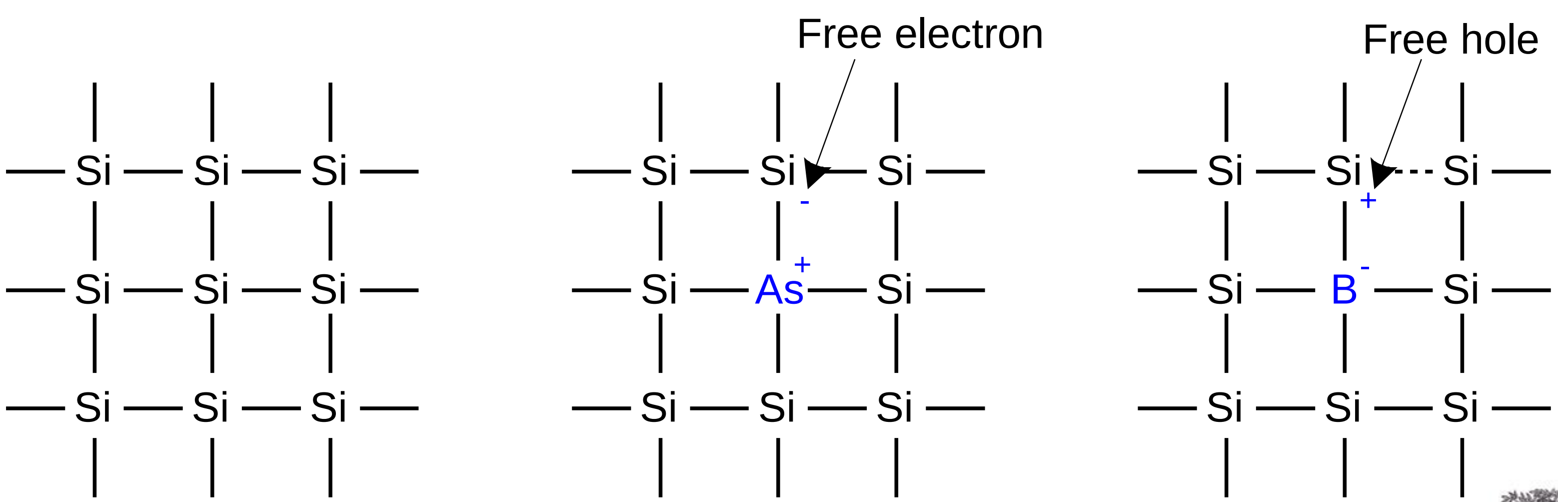


Robert Noyce, 1927-1990

- Nicknamed "Mayor of Silicon Valley"
- Cofounded Fairchild Semiconductor in 1957
- Cofounded Intel in 1968
- Co-invented the integrated circuit

Silicon

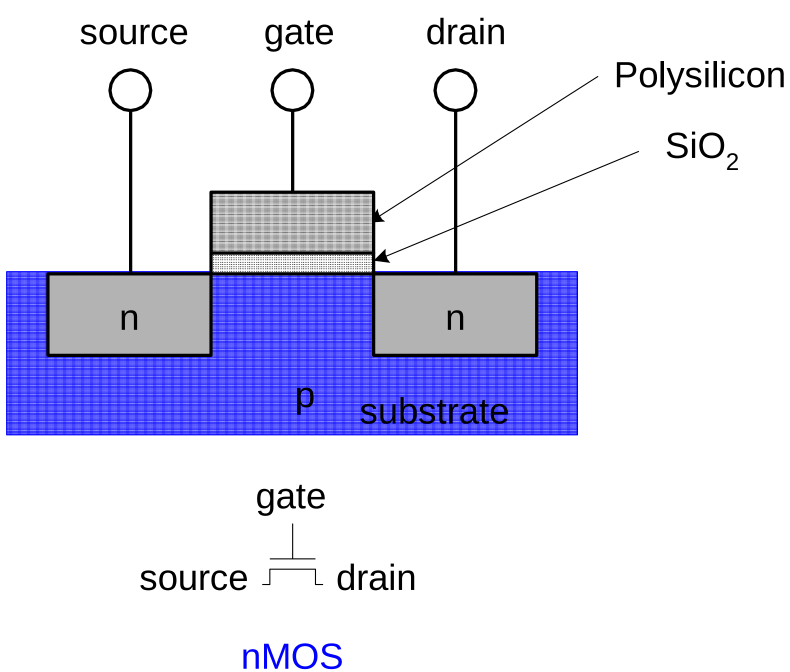
- Transistors built from silicon, a semiconductor
- Pure silicon is a poor conductor (no free charges)
- Doped silicon is a good conductor (free charges)
  - n-type (free *n*egative charges, electrons)
  - p-type (free *p*ositive charges, holes)



Silicon Lattice **n-Type** **p-Type**

MOS Transistors

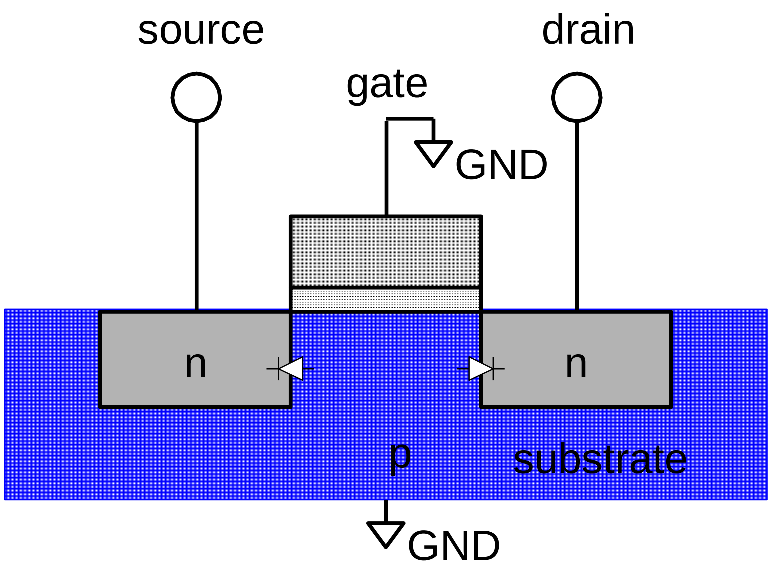
- **Metal oxide silicon (MOS) transistors:**
  - Polysilicon (used to be **metal**) gate
  - **Oxide** (silicon dioxide) insulator
  - Doped **silicon**



Transistors: nMOS

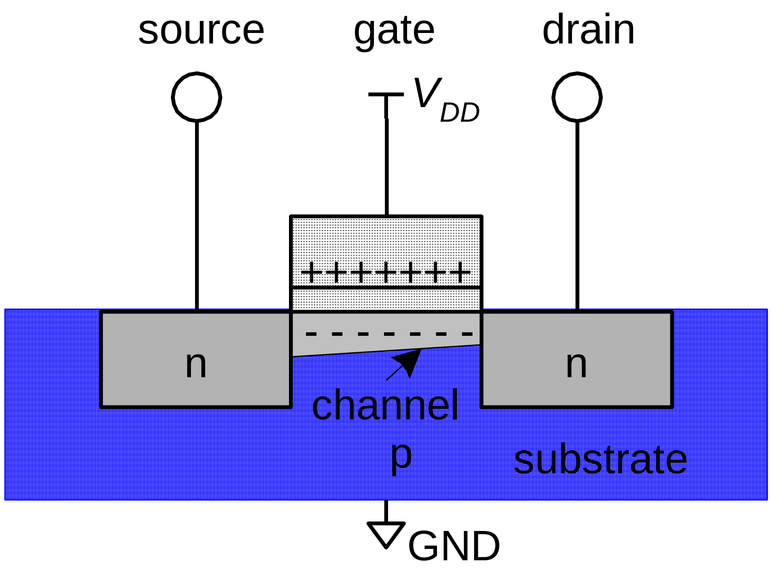
Gate = 0

OFF (no connection between source and drain)



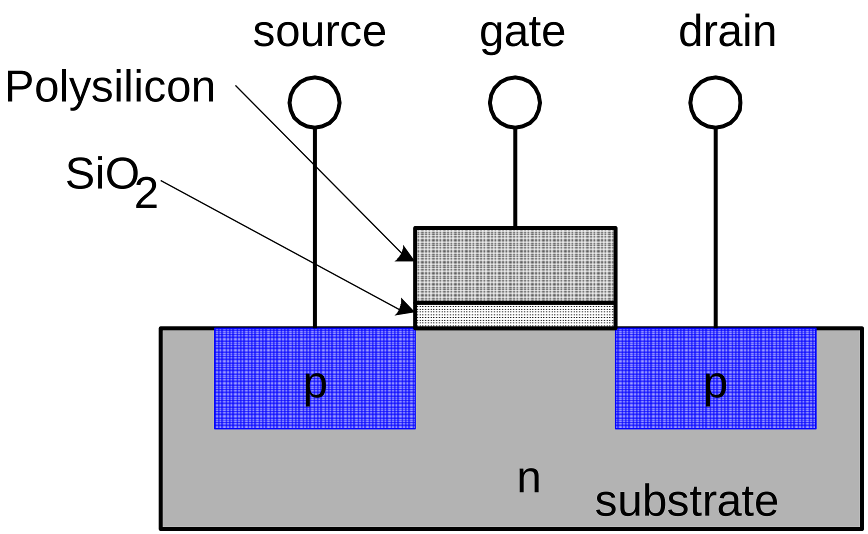
Gate = 1

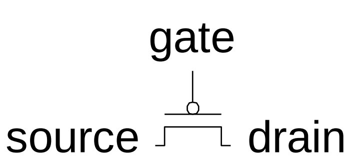
ON (channel between source and drain)



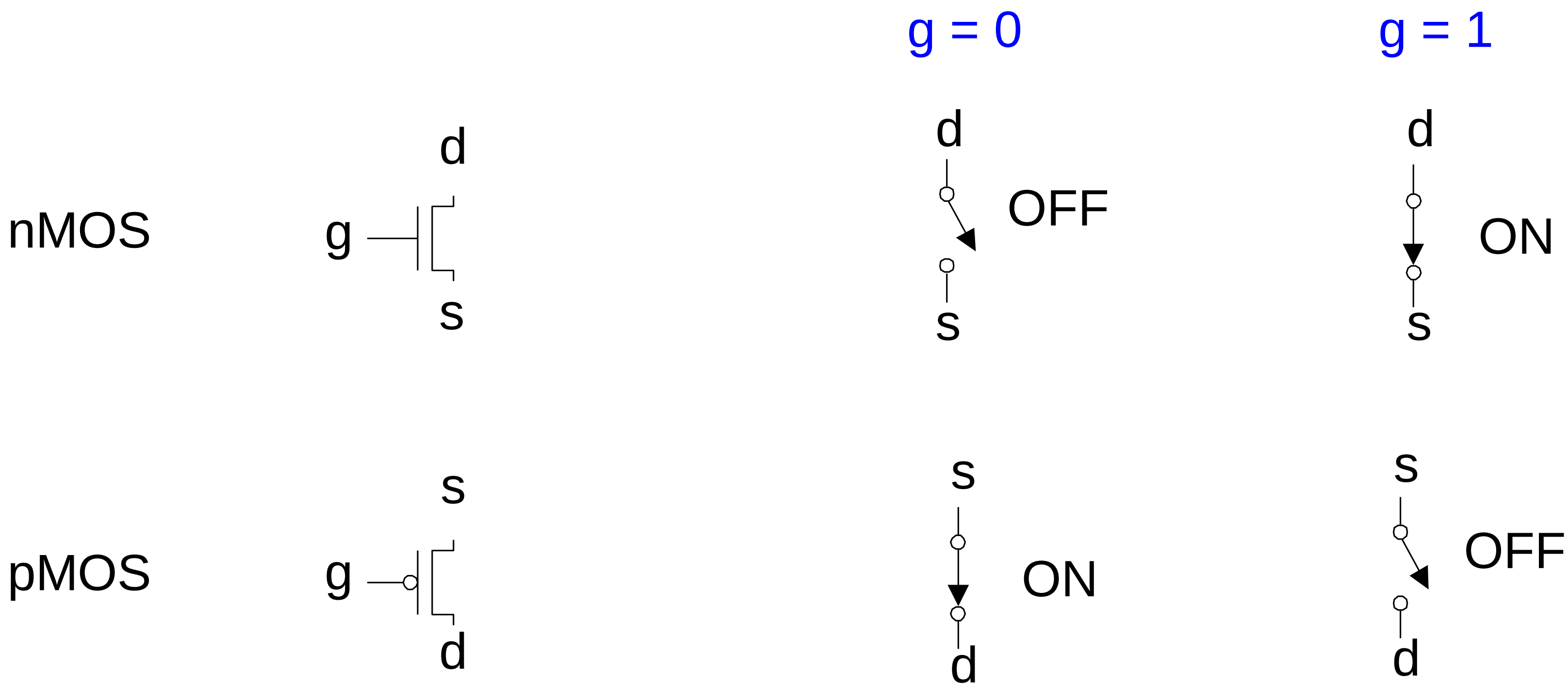
Transistors: pMOS

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



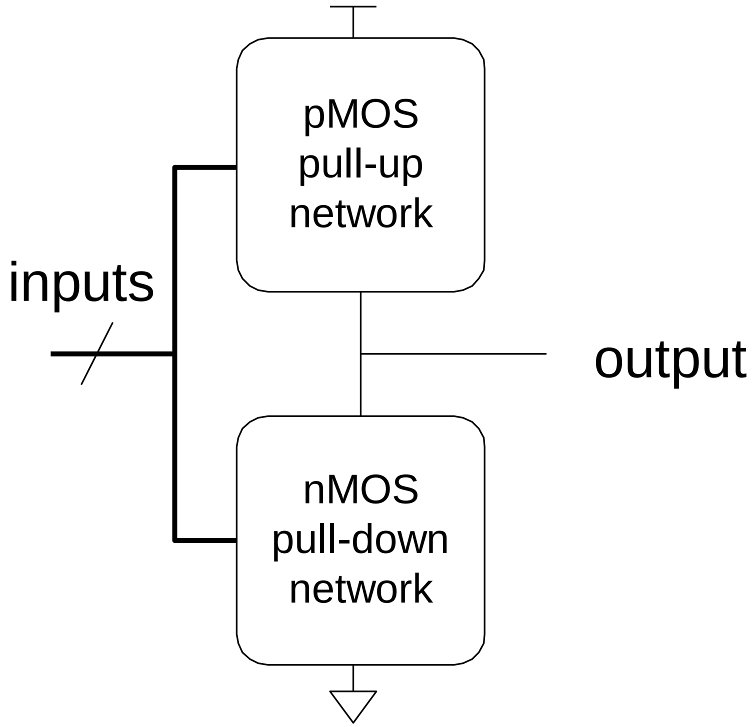


Transistor Function



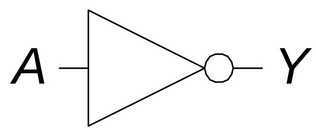
Transistor Function

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



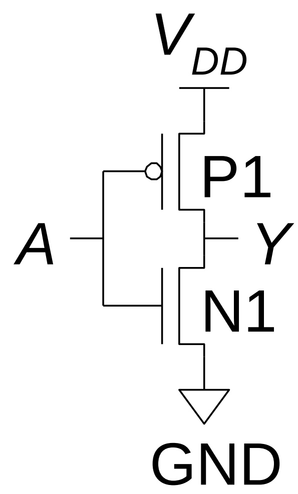
CMOS Gates: NOT Gate

NOT



$Y = \overline{A}$

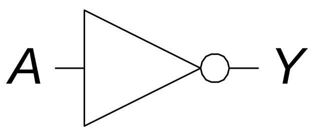
A	Y
0	1
1	0



A	P1	N1	*Y*
0			
1			

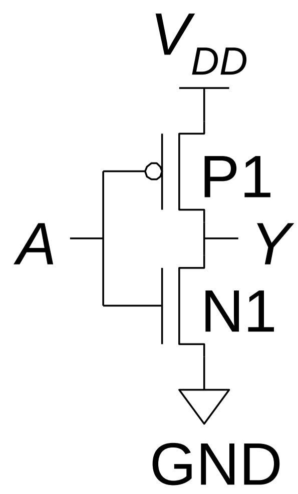
CMOS Gates: NOT Gate

NOT



$$Y = \overline{A}$$

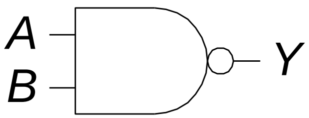
A	Y
0	1
1	0



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

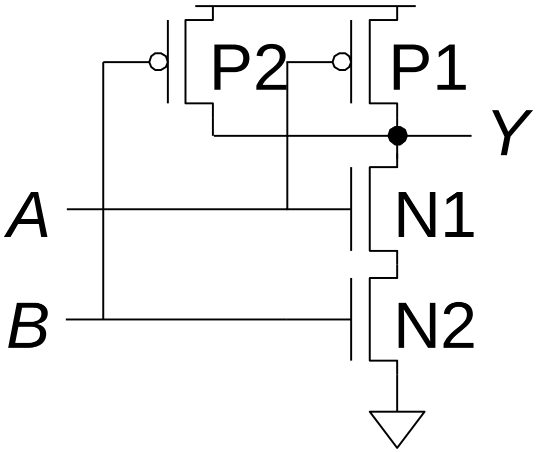
CMOS Gates: NAND Gate

NAND



$Y = \overline{AB}$

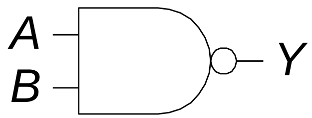
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0



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

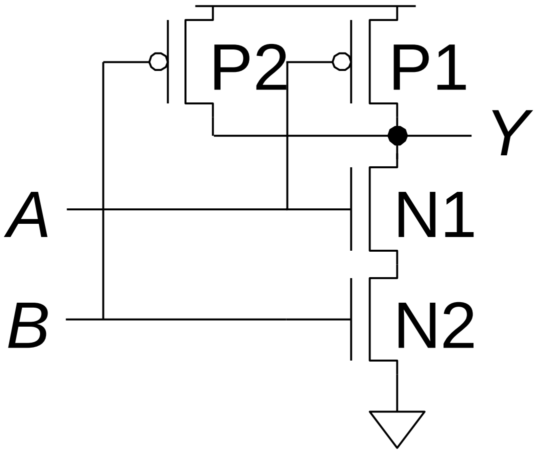
CMOS Gates: NAND Gate

NAND



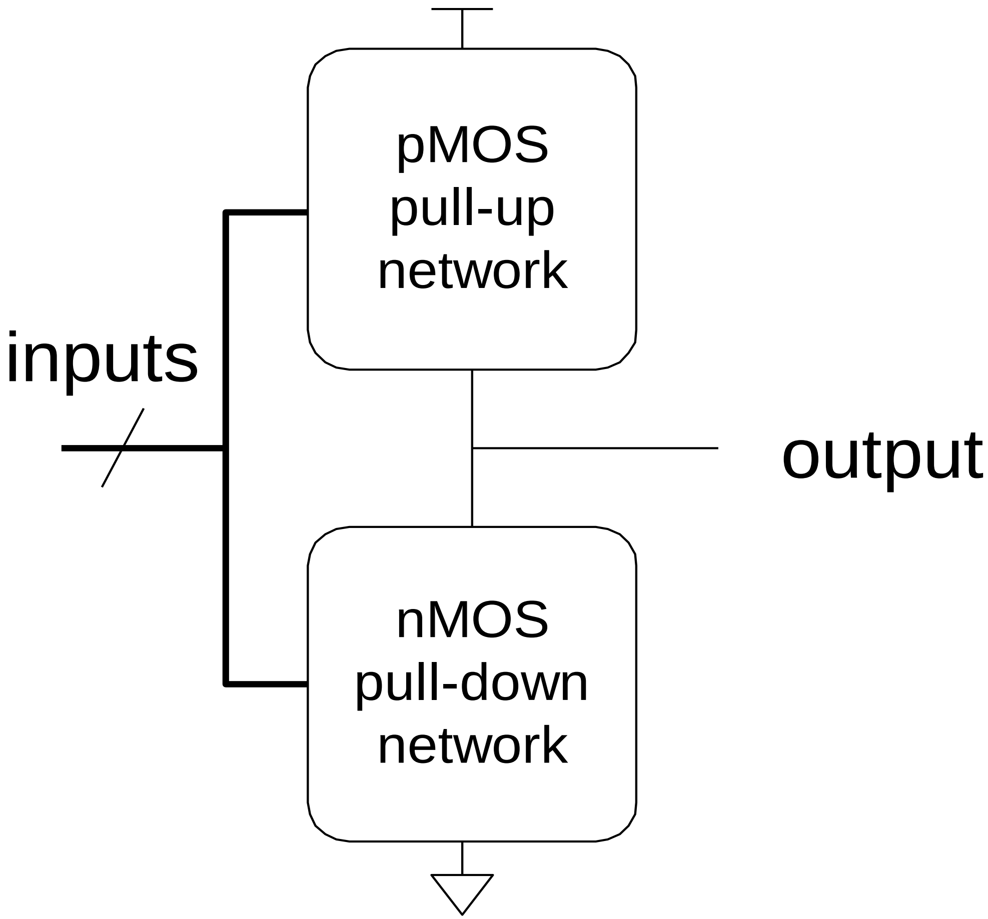
$Y = \overline{AB}$

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

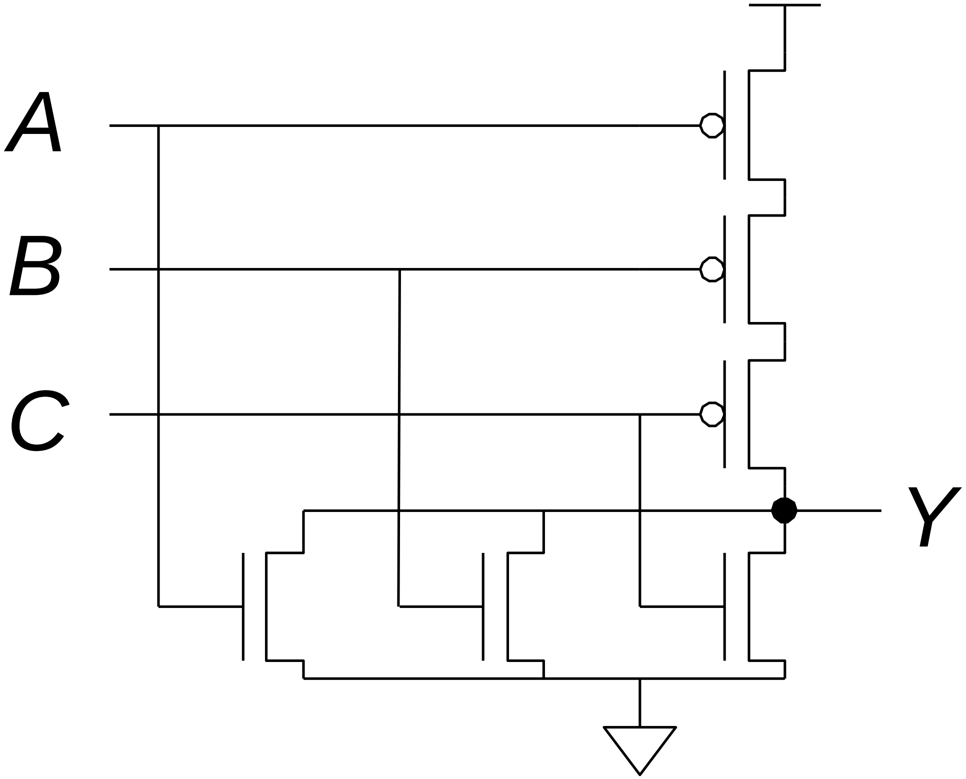
CMOS Gate Structure



NOR Gate

How do you build a three-input NOR gate?

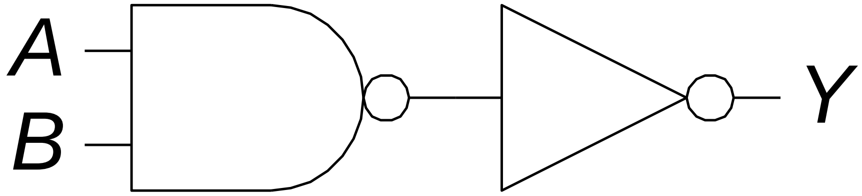
NOR3 Gate



Other CMOS Gates

How do you build a two-input AND gate?

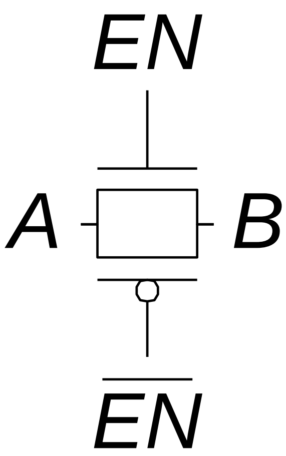
AND2 Gate



FROM ZERO TO ONE

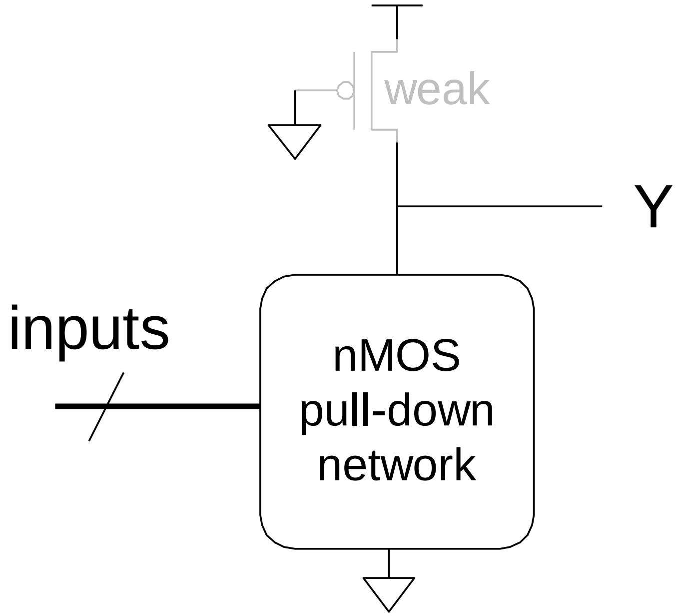
Transmission Gates

- nMOS pass 1's poorly
- pMOS pass 0's poorly
- Transmission gate is a better switch
  - – passes both 0 and 1 well
- When *EN* = 1, the switch is ON:
  - – *EN* = 0 and *A* is connected to *B*
- When *EN* = 0, the switch is OFF:
  - – *A* is not connected to *B*



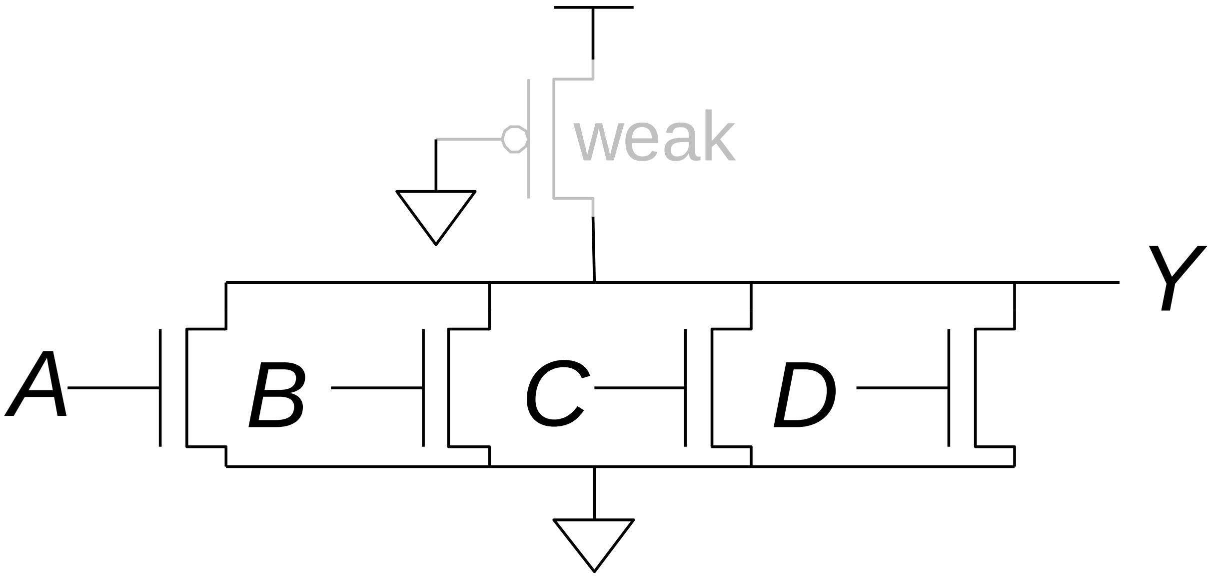
Pseudo-nMOS Gates

- Replace pull-up network with *weak* pMOS transistor that is always on
- pMOS transistor: pulls output HIGH *only* when nMOS network not pulling it LOW



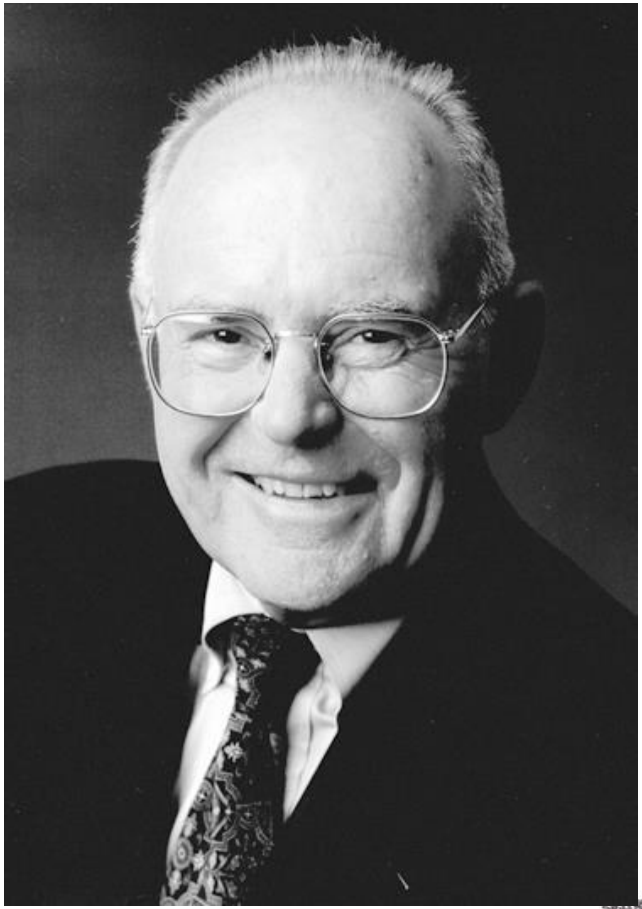
Pseudo-nMOS Example

Pseudo-nMOS **NOR4**

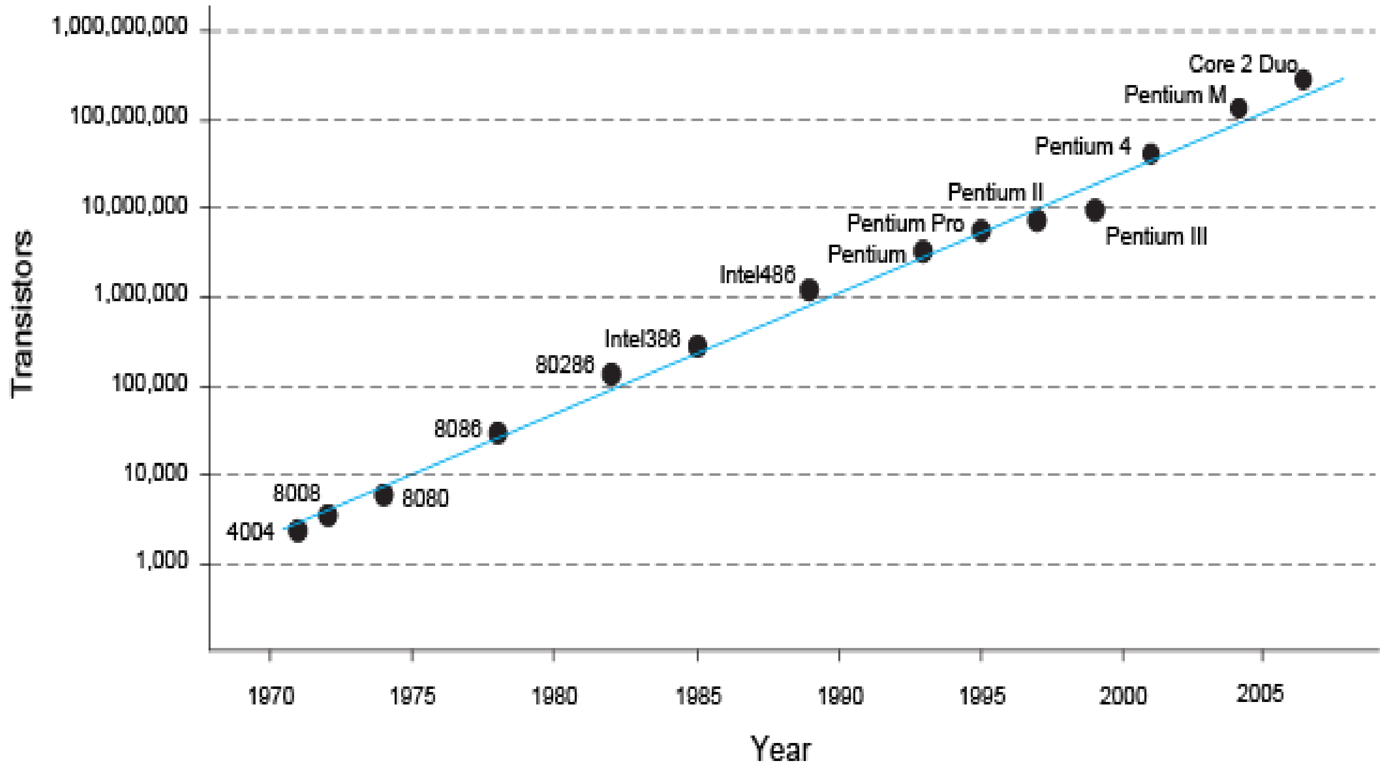


Gordon Moore, 1929-

- 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

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: transistors switch (from 1 to 0 or vice versa) at that frequency
  - Capacitor is charged $f/2$ times per second (discharging from 1 to 0 is free)
- Dynamic power consumption:

$$P_{dynamic} = \frac{1}{2}CV_{DD}^2 f$$

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:

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

Power Consumption Example

- Estimate the power consumption of a wireless handheld computer
  - V_{DD} = 1.2 V
  - C = 20 nF
  - f = 1 GHz
  - I_{DD} = 20 mA

Power Consumption Example

- Estimate the power consumption of a wireless handheld computer
  - V_{DD} = 1.2 V
  - C = 20 nF
  - f = 1 GHz
  - I_{DD} = 20 mA

$P = ½CV_{DD}^2 f + I_{DD}V_{DD}$

$= ½(20\text{ nF})(1.2\text{ V})^2(1\text{ GHz}) +$

$(20\text{ mA})(1.2\text{ V})$

$= (14.4 + 0.024)\text{ W} \approx 14.4\text{ W}$