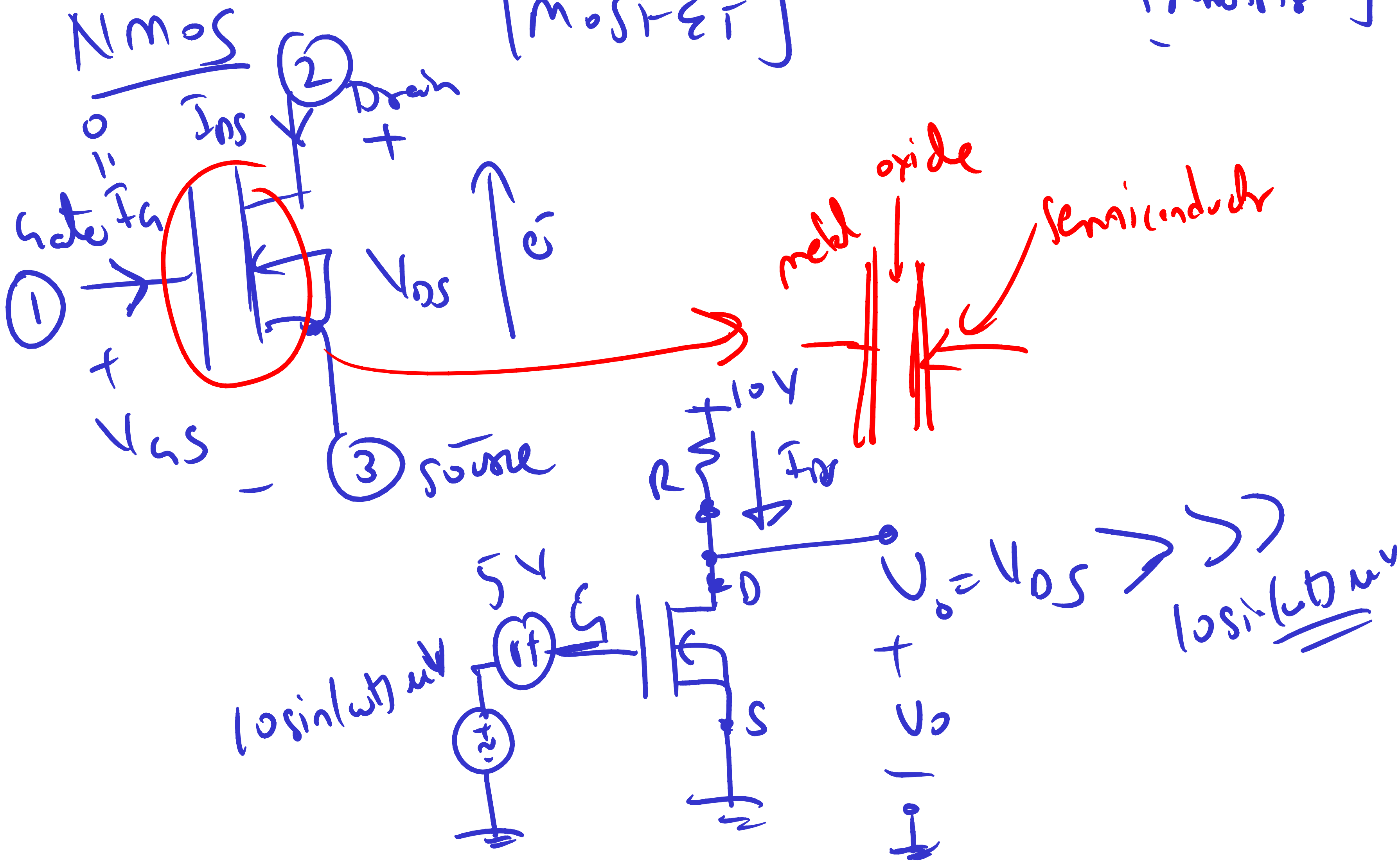
EE100Su08 Lecture #16 (August 1st 2008)

- OUTLINE
  - Project next week: Pick up kits in your first lab section, work on the project in your first lab section, at home etc. and wrap up in the second lab section. USE MULTISIM TO SIMULATE PROJECT (REFER TO MULTISIM FILE ONLINE!)
  - HW #3s-#6s: Pick up from lab, regrades: talk to Bart
  - Introduction to Boolean Algebra and Digital Circuits
  - Diode Logic
  - Transistor introduction (MOSFETs)
  - Transistor logic circuits
- Reading
  - Reader: Chapter 2, Chapter 4 and 5 (for transistors, just concentrate on logic applications).

Transistor: [Metal Oxide Semiconductor Field Effect Transistor]

[MOSFET]



Analog *vs.* Digital Signals

- **Most** (but not all) **observables are analog**

  think of analog vs. digital watches

but the most convenient way to represent & transmit information electronically is to use digital signals

think of a computer!

Digital Signal Representations

Binary numbers can be used to represent any quantity.

Counting:

Base-10

0
1
2
3
4
5
6
7
8
9
10
11
12

$10 = 1 \times 10^1 + 0 \times 10^0$

$12 = 1 \times 10^1 + 2 \times 10^0$

Base-5

0
1
2
3
4
10
11
12
13
14
20
21
22
23
24
30
31
32
33
34
40
41
42
43
44
100

$(10)_5 = 1 \times 5^1 + 0 \times 5^0$

Digital Signal Representations

Binary numbers can be used to represent any quantity.

Counting: Base - 2

0
1
10
11
100
101
110 → $(110)_2 = 1\times2^2 + 1\times2^1 + 0\times2^0$
$= (6)_{10}$
111
1000
1001
1010

Decimal Numbers: Base 10

Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Example:

$3271 = (3x10^3) + (2x10^2) + (7x10^1) + (1x10^0)$

This is a four-digit number. The left hand most number (3 in this example) is often referred as the most significant number and the right most the least significant number (1 in this example).

Numbers: positional notation

- Number Base B $\Rightarrow$ B symbols per digit:
  - –Base 10 (Decimal): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - –Base 2 (Binary): 0, 1
- Number representation:
  - – $d_{31}d_{30} \ldots d_1d_0$ is a 32 digit number
  - – value = $d_{31} \times B^{31} + d_{30} \times B^{30} + \ldots + d_1 \times B^1 + d_0 \times B^0$
- Binary: 0,1 (In binary digits called “bits”)

  11010 $= 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

  $= 16 + 8 + 2$

  $= 26$

  - –Here 5 digit binary # turns into a 2 digit decimal #

Hexadecimal Numbers: Base 16

Base-10: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

- Hexadecimal:

  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19

  – Normal digits + 6 more from the alphabet

- Conversion: Binary⇔Hex

  – 1 hex digit represents 16 decimal values

  – 4 binary digits represent 16 decimal values

  ⇒1 hex digit replaces 4 binary digits

$1 \times 16^1 + \overset{10}{A} \times 16^0 \leftarrow (1A)_{16}$

Decimal-Binary Conversion

- Decimal to Binary
  - Repeated Division By 2
    - Consider the number 2671.
  - Subtraction – if you know your 2^N values by heart.
- Binary to Decimal conversion

$$110001_2 = 1x2^5 + 1x2^4 + 0x2^3 + 0x2^2 + 0x2^1 + 1x2^0$$
$$= 32_{10} + 16_{10} + 1_{10}$$
$$= 49_{10}$$
$$= 4x10^1 + 9x10^0$$

Example: Analog to Digital conversion

Possible digital representation for the sine wave signal:

Analog representation: Amplitude in μV	Digital representation: Binary number
1	000001
2	000010
3	000011
4	000100
5	000101
8	001000
16	010000
32	100000
50	**110010**
63	111111

bit, binary digit

Binary Representation

- N bit can represent 2^N values: typically from 0 to 2^N-1
  - 3-bit word can represent 8 values: e.g. 0, 1, 2, 3, 4, 5, 6, 7

000
001
010
011
100
101
110, 111

- Conversion
  - Integer to binary
  - Fraction to binary ($13.5_{10}=1101.1_2$ and $0.392_{10}=0.011001_2$)
- Octal and hexadecimal

Negative numbers: natural way:
Sign-bit, but more efficient ways
(CS61C)

$\_\ \_\ \_\ \_\ .\ \_ \ldots$

$2^3 \quad 2^2 \quad 2^1 \quad 2^0 \quad 2^{-1}$

Logic Gates

- Logic gates
  - Combine several logic variable inputs to produce a logic variable output
- Memory
  - Memoryless: output at a given instant depends the input values of that instant.
  - Memory: output depends on previous and present input values.

Boolean algebras

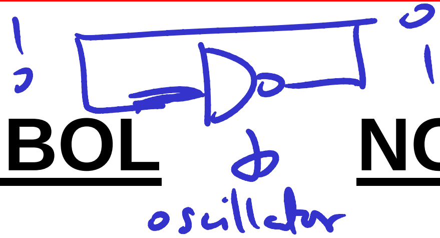
- Algebraic structures
  - "capture the essence" of the logical operations AND, OR and NOT
  - corresponding set for theoretic operations intersection, union and complement
  - named after George Boole, an English mathematician at University College Cork, who first defined them as part of a system of logic in the mid 19th century.
  - Boolean algebra was an attempt to use algebraic techniques to deal with expressions in the propositional calculus.
  - Today, Boolean algebras find many applications in electronic design. They were first applied to switching by Claude Shannon in the 20th century.

Boolean algebras

- The operators of Boolean algebra may be represented in various ways. Often they are simply written as AND, OR and NOT.
- In describing circuits, NAND (NOT AND), NOR (NOT OR) and XOR (eXclusive OR) may also be used.
- Mathematicians often use + for OR and · for AND (since in some ways those operations are analogous to addition and multiplication in other algebraic structures) and represent NOT by a line drawn above the expression being negated.

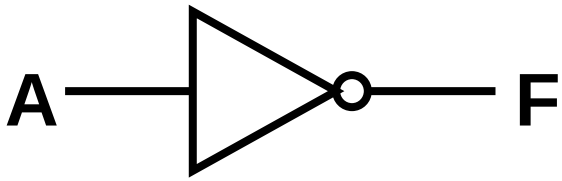
Logic Functions, Symbols, & Notation

A	F
0	0
1	0



NAME | **SYMBOL** | **NOTATION** | **TRUTH TABLE**

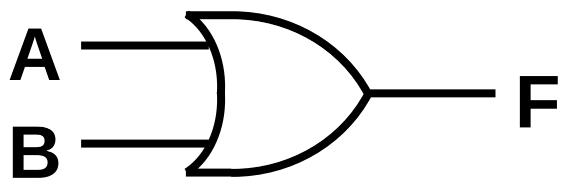
"NOT"



$F = \overline{A}$

A	F
0	1
1	0

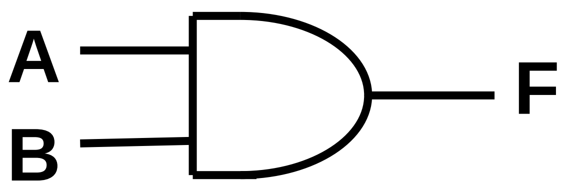
"OR"



$F = A+B$

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

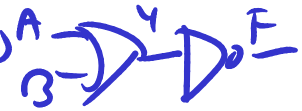
"AND"



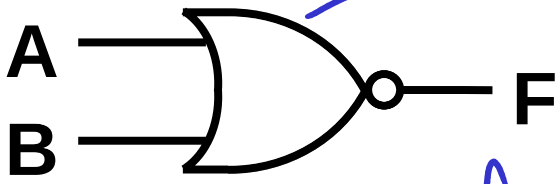
$F = A \bullet B$

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

Logic Functions, Symbols, & Notation 2

"Not of OR" → 

"NOR"

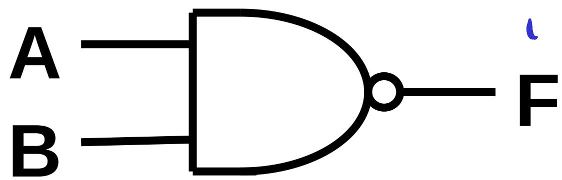


$F = \overline{A+B}$

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

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

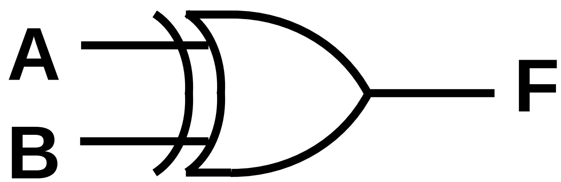
"NAND"



$F = \overline{A \bullet B}$

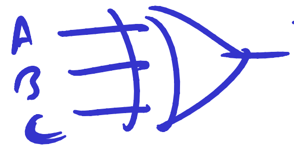
A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

"XOR"
(exclusive OR)

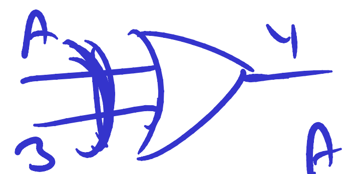


$F = A \oplus B$

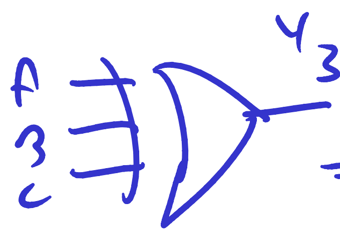
A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

XOR (3-inputs)  $Y_3 = A \oplus B \oplus C$

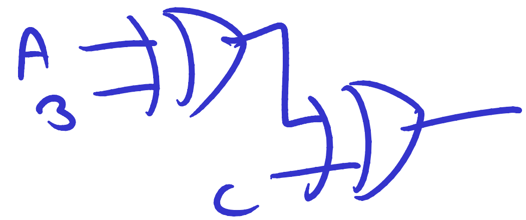
A	B	C	Y_3
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



A	B	Y_2
0	0	0
0	1	1
1	0	1
1	1	0



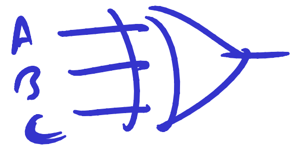
=



Associativity: $(A \oplus B) \oplus C$

$= A \oplus (B \oplus C)$

XOR (3-inputs)



$Y_3 = A \oplus B \oplus C$

A	B	C	Y_3
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

1-Bit!
Binary addition:

$0 + 0 + 0 = 0$

$0 + 0 + 1 = 1$

$0 + 1 + 0 = 1$

$0 + 1 + 1 = 10$

$1 + 0 + 0 = 1$

$1 + 0 + 1 = 10$

$1 + 1 + 0 = 10$

$1 + 1 + 1 = 11$

Boolean Algebra

- NOT operation (inverter)

$$A \bullet \overline{A} = 0$$

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

- AND operation

$$A \bullet A = A$$

$$A \bullet 1 = A$$

$$A \bullet 0 = 0$$

$$A \bullet B = B \bullet A$$

$$(A \bullet B) \bullet C = A \bullet (B \bullet C)$$

- OR operation

$$A + A = A$$

$$A + 1 = 1$$

$$A + 0 = A$$

$$A + B = B + A$$

$$(A + B) + C = A + (B + C)$$

Boolean Algebra

- Distributive Property

$$A \bullet (B + C) = A \bullet B + A \bullet C$$

$$(A + B) \bullet C = (A + B) \bullet (A + C)$$

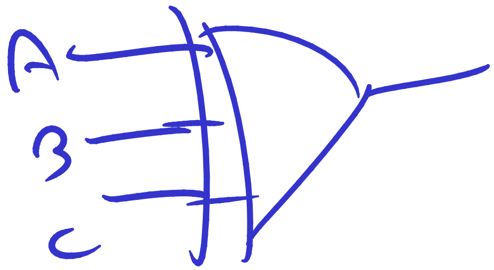
- De Morgan's laws

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

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

- An excellent web site to visit
  - http://en.wikipedia.org/wiki/Boolean_algebra

Circuit Realization: Three input adder ~~with carry~~



Y_3 = 3-input adder

Diode Logic: AND Gate

- Diodes can be used to perform logic functions:

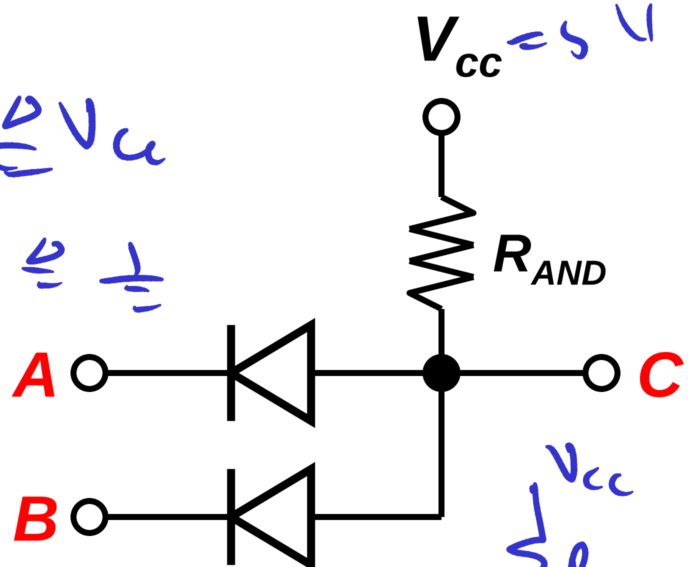
(Assuming ideal diode model)

AND gate

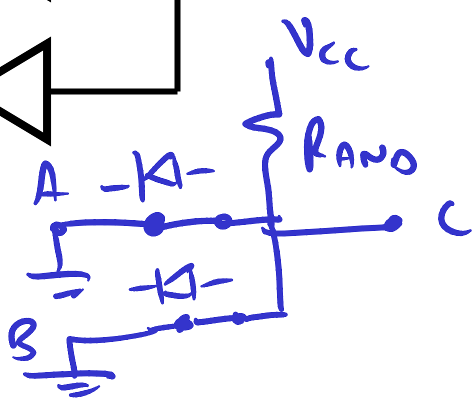
output voltage is high only if both A and B are high

"1" ≙ V_{cc}

"0" ≙



A = 0, B = 0

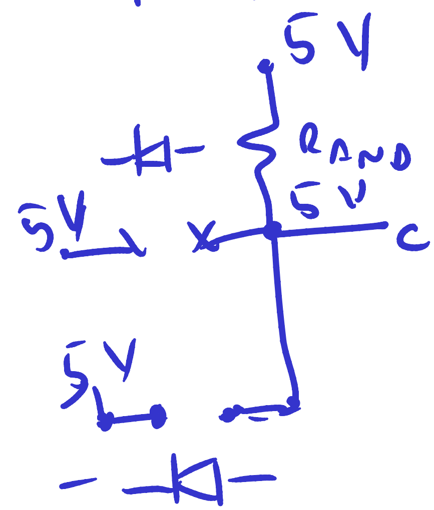


Inputs A and B vary between 0 Volts ("low") and V_{cc} ("high")

Between what voltage levels does C vary?

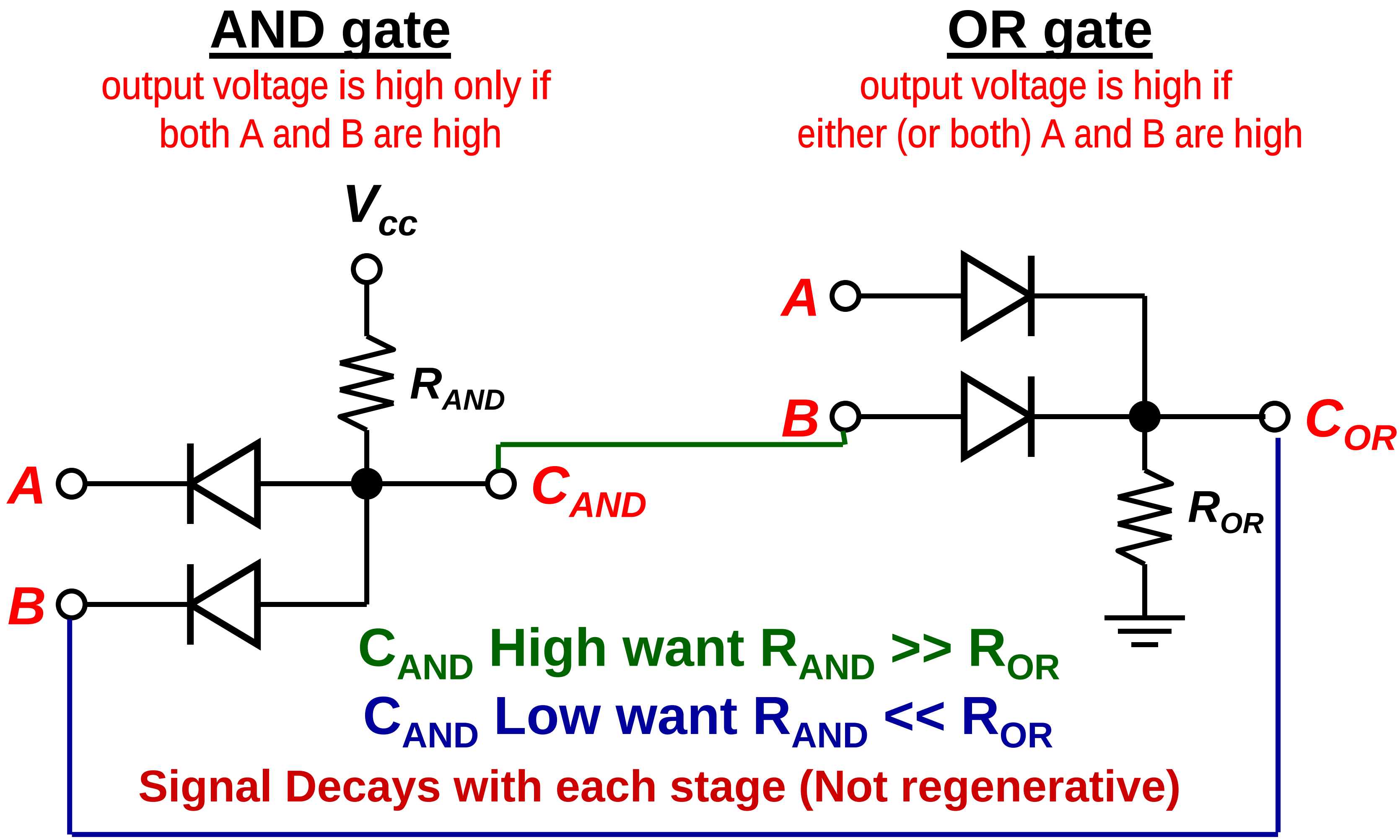
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

A = 0, B = 1



Diode Logic: Incompatibility and Decay

- Diode Only Gates are Basically Incompatible:



C_{AND} High want R_{AND} >> R_{OR}

C_{AND} Low want R_{AND} << R_{OR}

Signal Decays with each stage (Not regenerative)

MOSFETs: Detailed outline

- OUTLINE
  - The MOSFET as a controlled resistor
  - MOSFET ID vs. VGS characteristic
  - NMOS and PMOS I-V characteristics
  - Simple MOSFET circuits
- Reading
  - Reader: Chapters 4 and 5

MOSFET

- NMOS: Three regions of operation
  - V_{DS} and V_{GS} normally positive valus
  - $V_{GS}<V_t$:cut off mode, $I_{DS}=0$ for any V_{DS}
  - $V_{GS}>V_t$:transistor is turned on
    - $V_{DS}< V_{GS}-V_t$: Triode Region
    - $V_{DS}> V_{GS}-V_t$: Saturation Region
    - Boundary $v_{GS}-V_t=v_{DS}$

$$i_D = K\left[2(v_{GS}-V_t)v_{DS}-v_{DS}{}^2\right]$$

$$i_D = K\left[2(v_{GS}-V_t)^2\right]$$

$$K=\frac{W}{L}\frac{KP}{2}$$

MOSFET

- PMOS: Three regions of operation (interchange > and < from NMOS)
  - V_{DS} and V_{GS} Normally negative values
  - $V_{GS} > V_t$:cut off mode, $I_{DS}=0$ for any V_{DS}
  - $V_{GS} < V_t$:transistor is turned on
    - $V_{DS} > V_{GS}-V_t$: Triode Region
    - $V_{DS} < V_{GS}-V_t$: Saturation Region
    - Boundary $v_{GS} - V_t = v_{DS}$

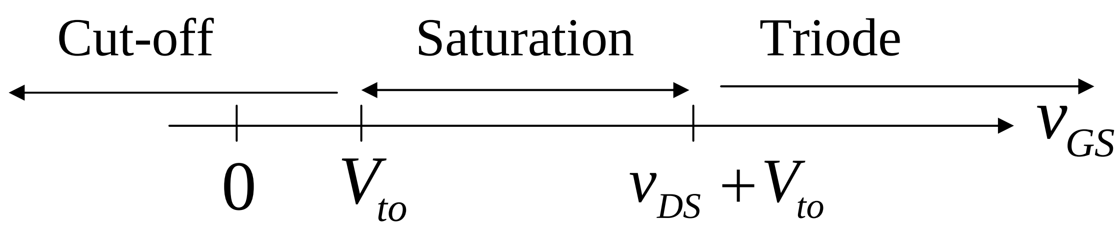
$$i_D = K\left[2(v_{GS} - V_t)v_{DS} - v_{DS}{}^2\right]$$

$$i_D = K\left[2(v_{GS} - V_t)^2\right]$$

$$K = \frac{W}{L}\frac{KP}{2}$$

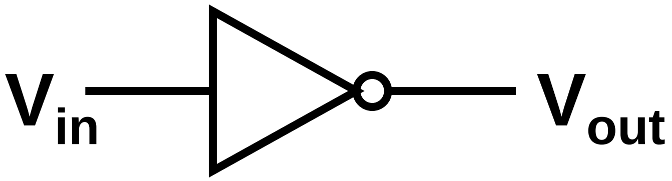
MOSFET Operating Regions

NMOS

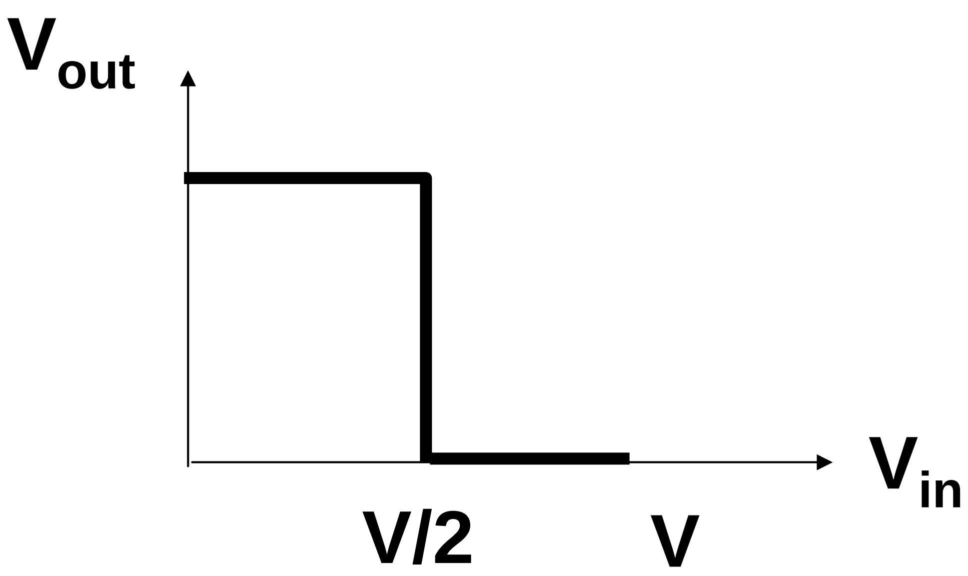


PMOS

Inverter = NOT Gate

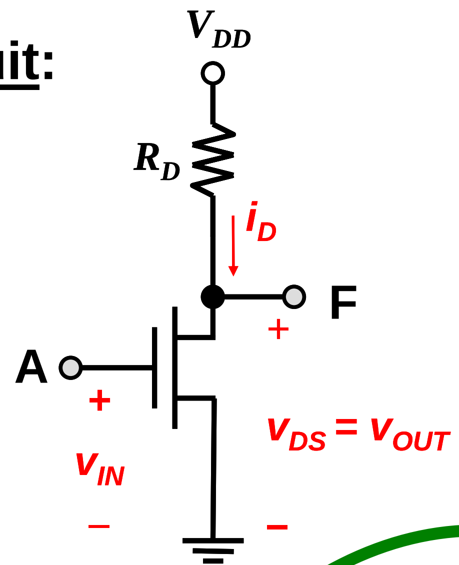


Ideal Transfer Characteristics

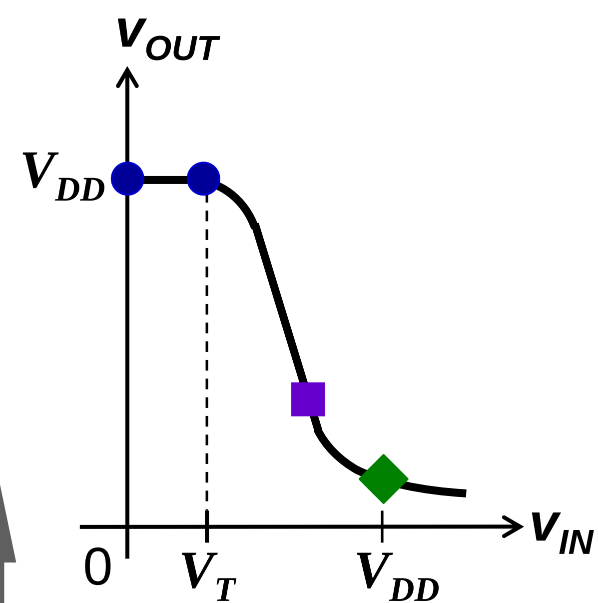


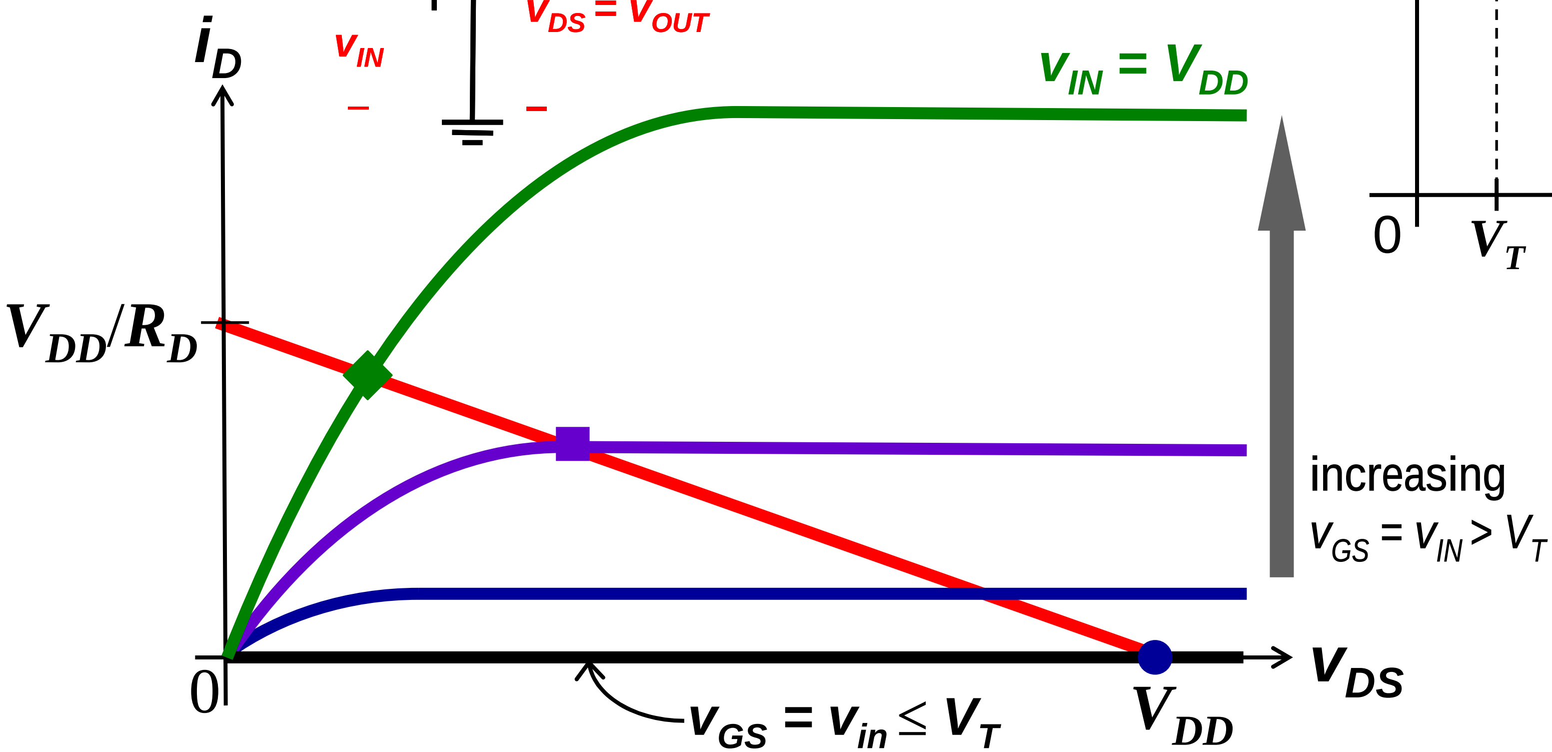
NMOS Resistor Pull-Up

Circuit:



Voltage-Transfer Characteristic





A	F
0	1
1	0

Disadvantages of NMOS Logic Gates

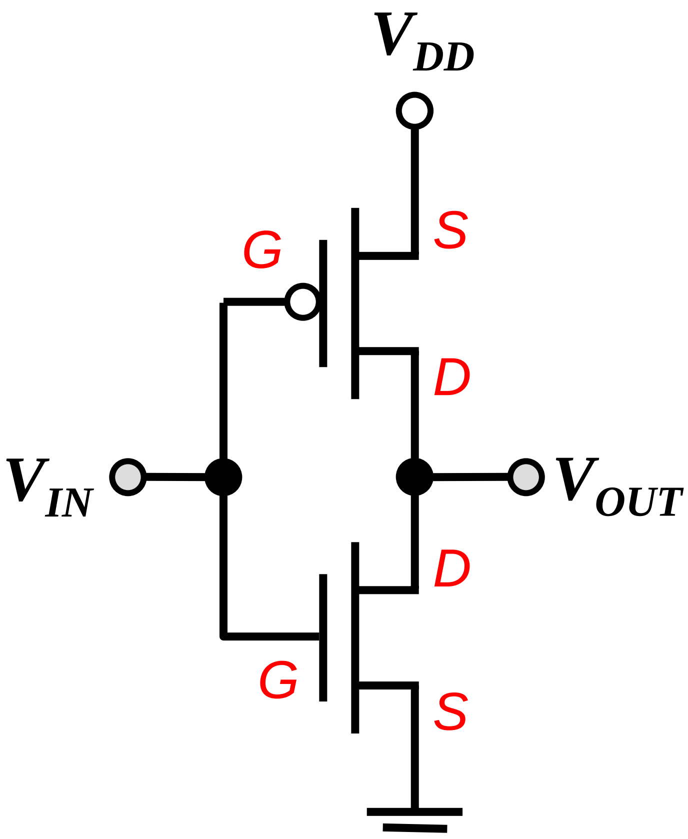
- Large values of $\boldsymbol{R_D}$ are required in order to
  - achieve a low value of V_{OL}
  - keep power consumption low

→ Large resistors are needed, but these take up a lot of space.

- One solution is to replace the resistor with an NMOSFET that is always on.

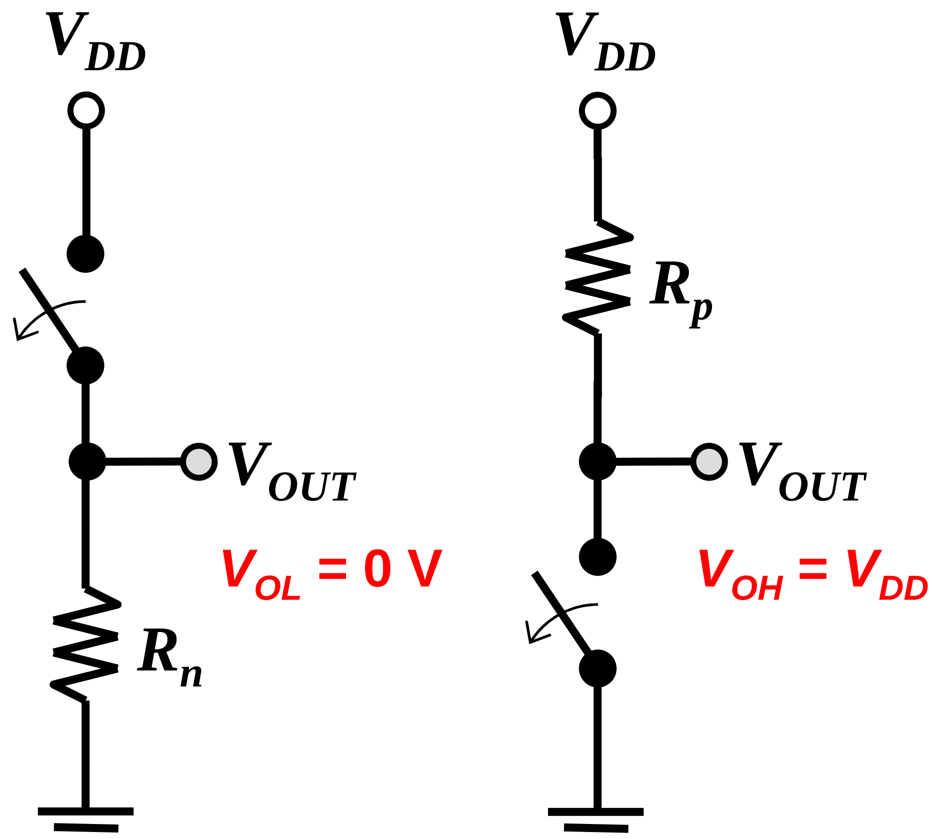
The CMOS Inverter: Intuitive Perspective

CIRCUIT



Low static power consumption, since one MOSFET is always off in steady state

SWITCH MODELS



$V_{IN} = V_{DD}$

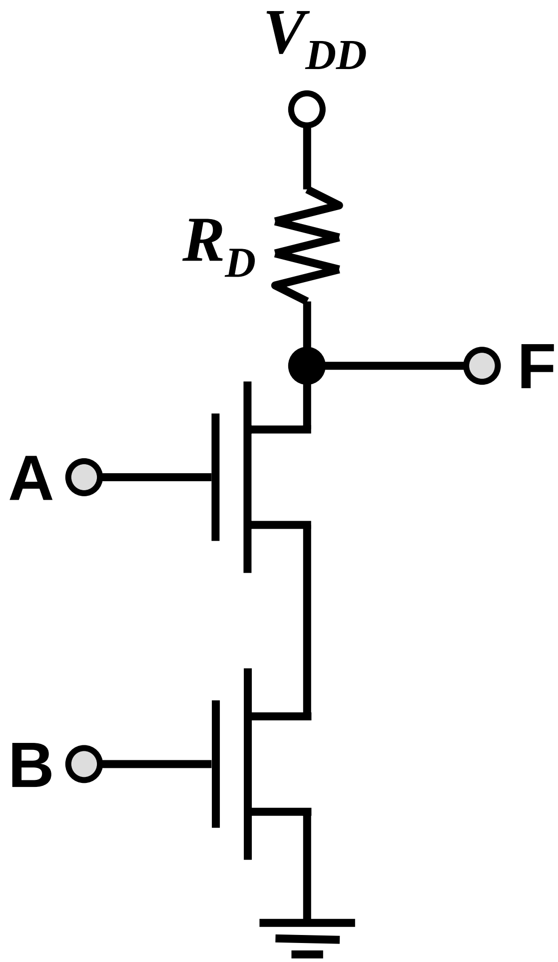
$V_{IN} = 0\text{ V}$

Features of CMOS Digital Circuits

- The output is always connected to V_{DD} or **GND** in steady state
  - $\rightarrow$ Full logic swing; large noise margins
  - $\rightarrow$ Logic levels are not dependent upon the relative sizes of the devices ("ratioless")
- There is no direct path between V_{DD} and **GND** in steady state
  - $\rightarrow$ no static power dissipation

NMOS NAND Gate

- Output is low only if both inputs are high

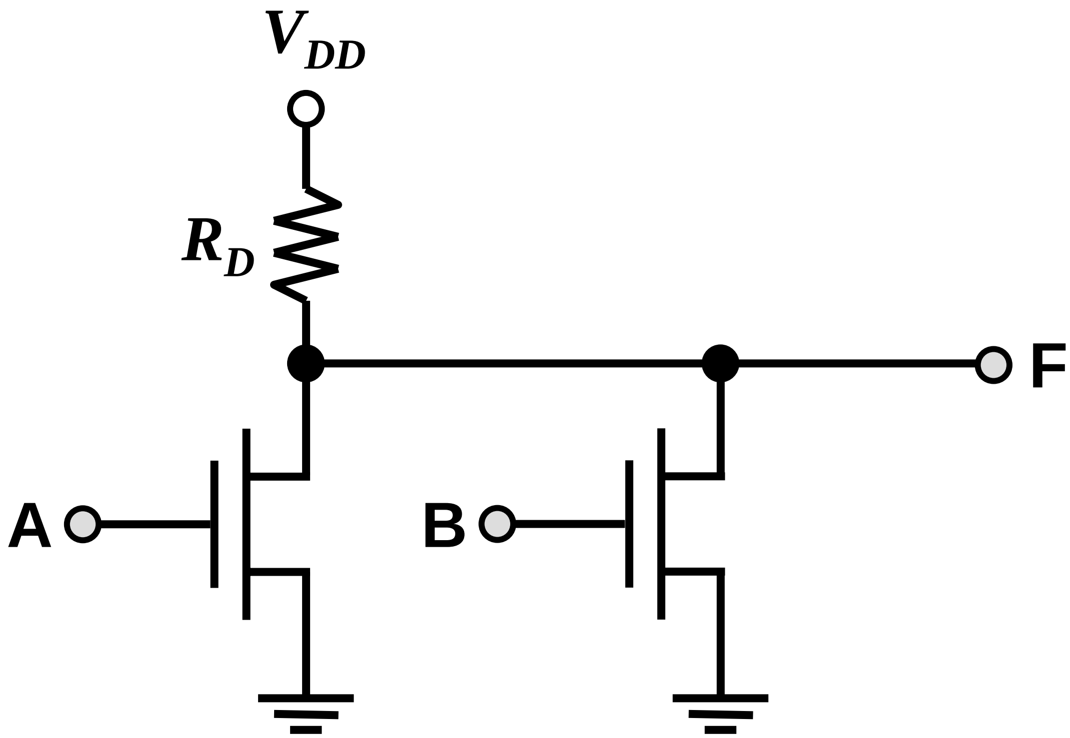


Truth Table

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

NMOS NOR Gate

- Output is low if either input is high

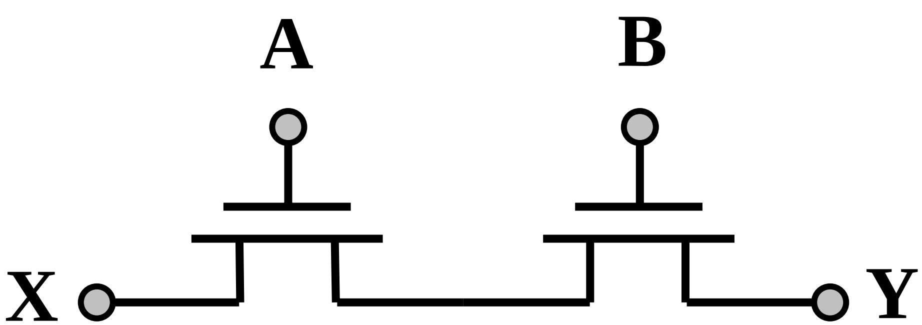


Truth Table

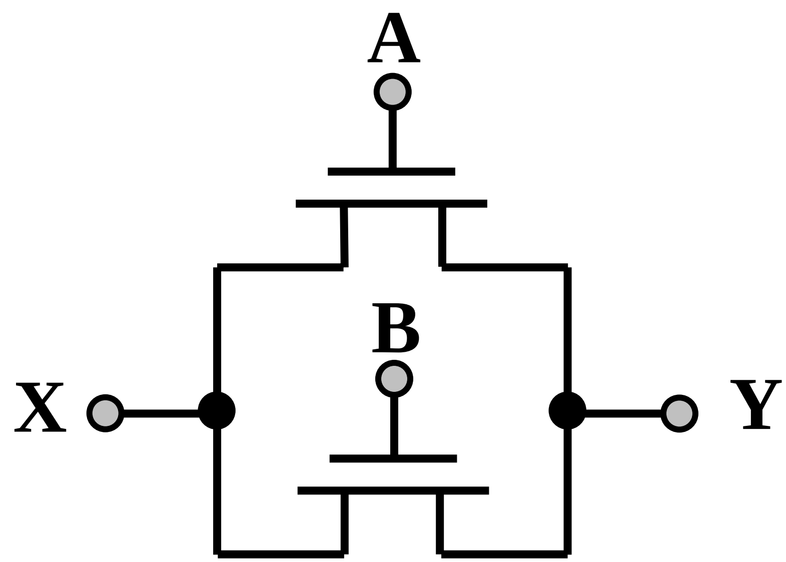
A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

N-Channel MOSFET Operation

An NMOSFET is a closed switch when the input is high



Y = **X** if **A** and **B**

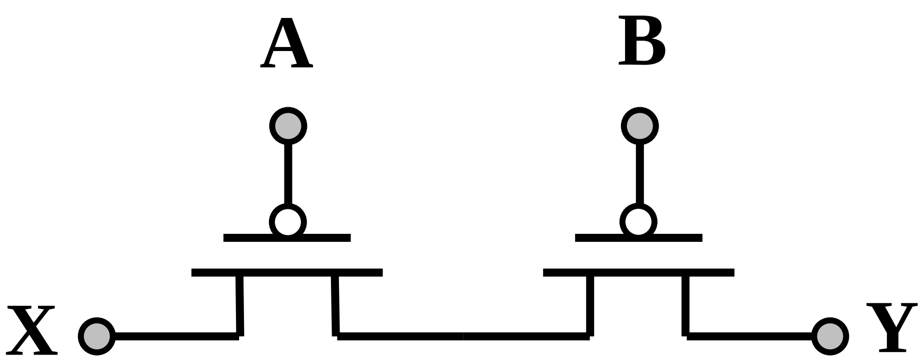


Y = **X** if **A** or **B**

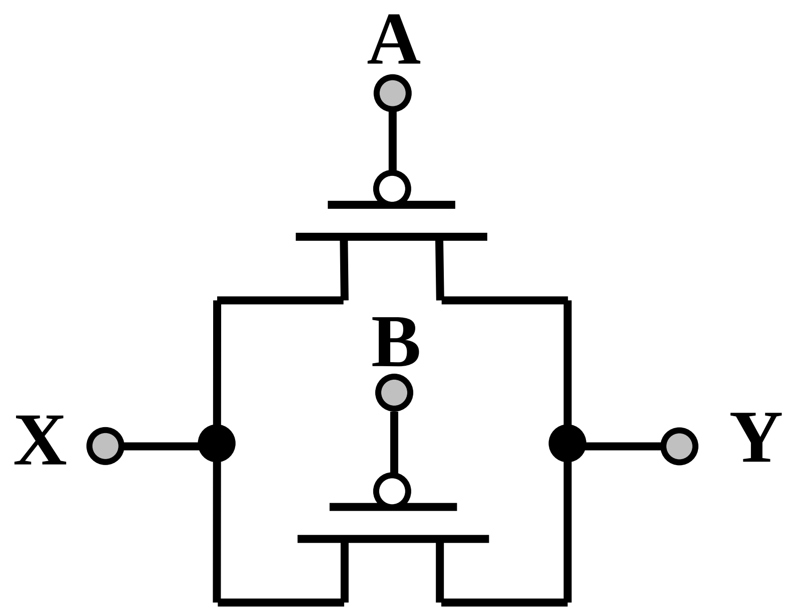
NMOSFETs pass a "strong" **0** but a "weak" **1**

P-Channel MOSFET Operation

A PMOSFET is a closed switch when the input is low



$$\mathbf{Y} = \mathbf{X} \text{ if } \overline{\mathbf{A}} \text{ and } \overline{\mathbf{B}}$$
$$= \overline{(\mathbf{A} + \mathbf{B})}$$

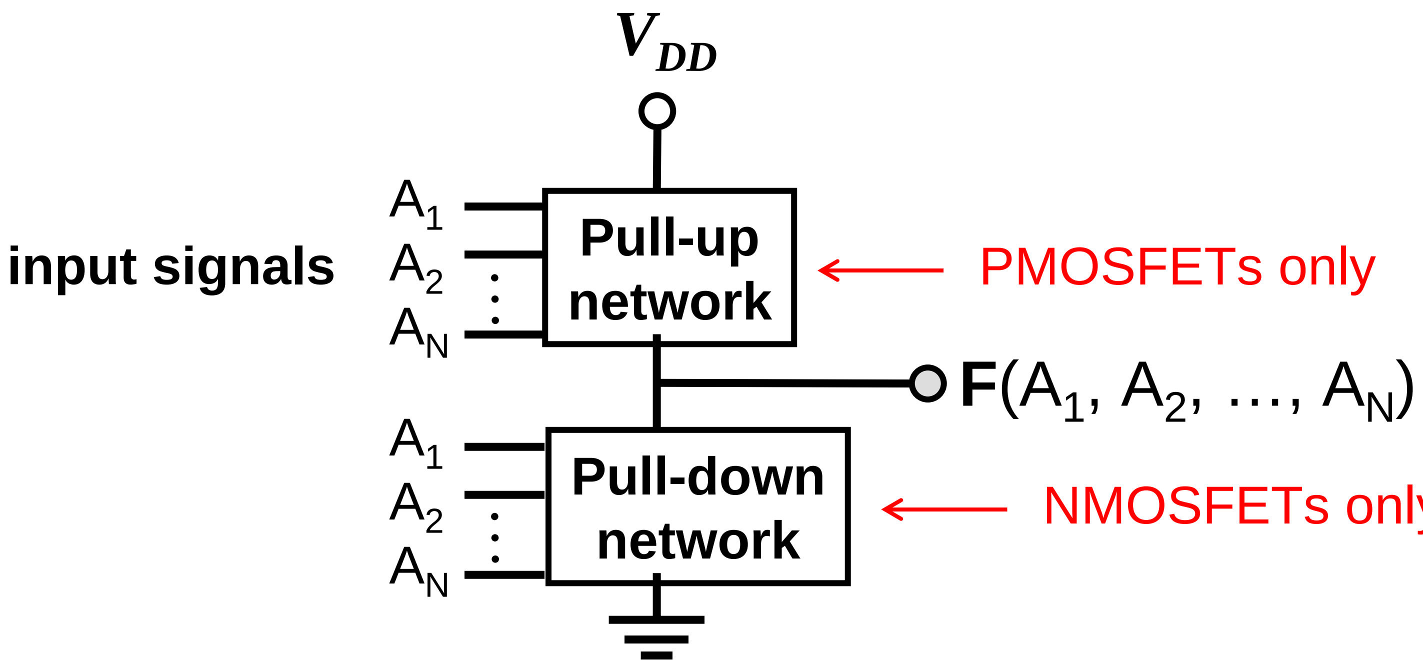


$$\mathbf{Y} = \mathbf{X} \text{ if } \overline{\mathbf{A}} \text{ or } \overline{\mathbf{B}}$$
$$= \overline{(\mathbf{AB})}$$

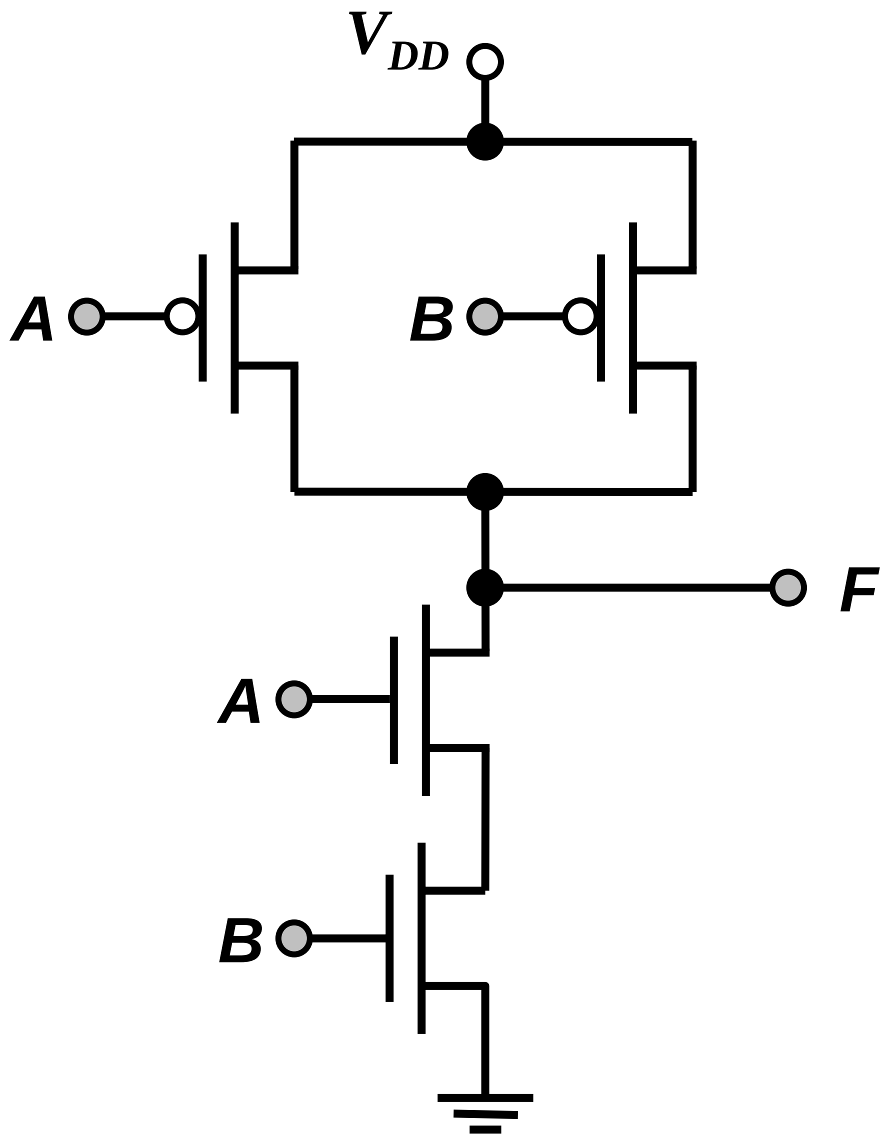
PMOSFETs pass a “strong” **1** but a “weak” **0**

Pull-Down and Pull-Up Devices

- In CMOS logic gates, NMOSFETs are used to connect the output to **GND**, whereas PMOSFETs are used to connect the output to $\boldsymbol{V_{DD}}$.
  - An NMOSFET functions as a ***pull-down device*** when it is turned on (gate voltage = $\boldsymbol{V_{DD}}$)
  - A PMOSFET functions as a ***pull-up device*** when it is turned on (gate voltage = ***GND***)

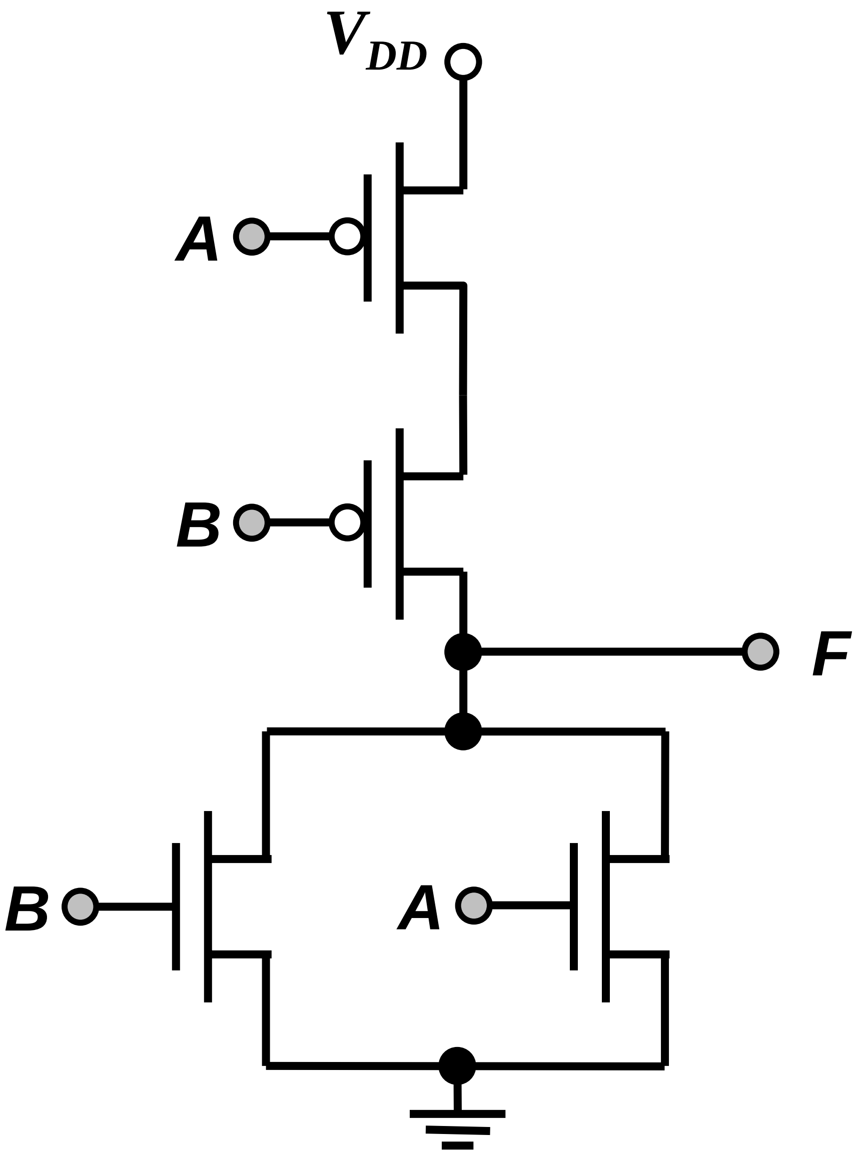


CMOS NAND Gate



A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

CMOS NOR Gate



A	B	F
0	0	1
0	1	0
1	0	0
1	1	0