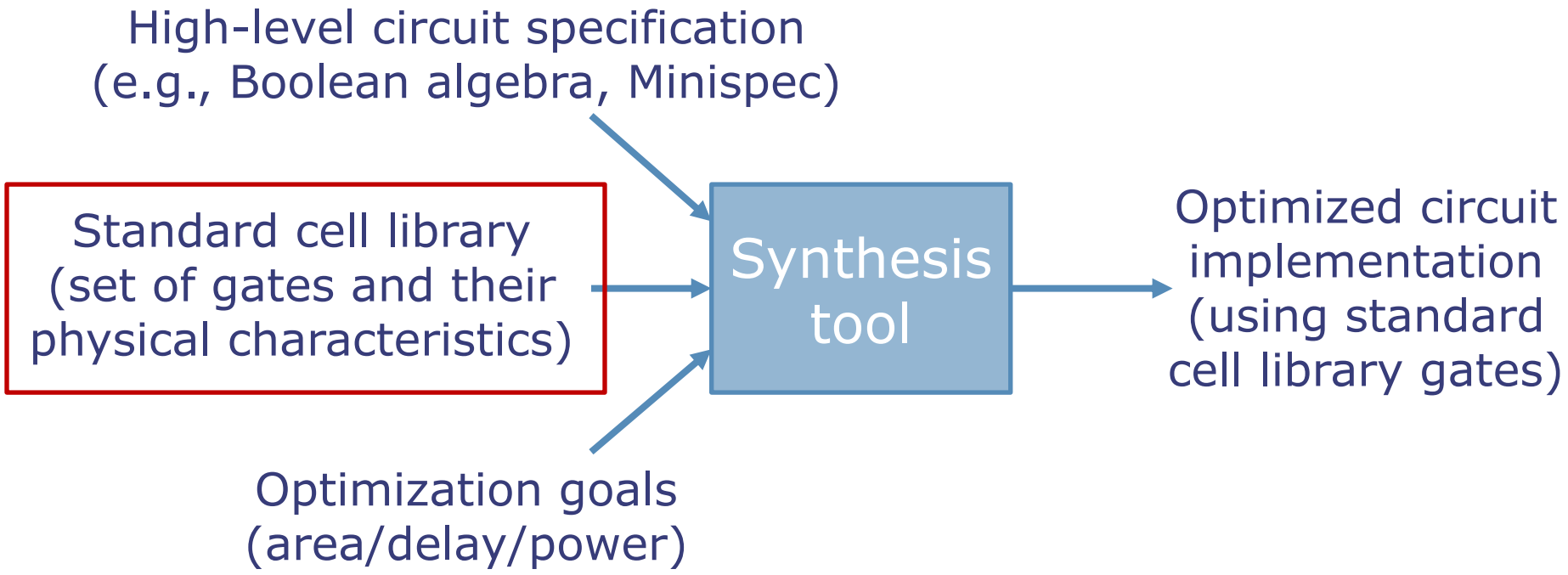
Logic Synthesis and CMOS Technology

Reminders:

- Quiz 1 on Thu March 5th 7:30-9:30pm, covers lectures L1-L7 and labs 1-2.
- See piazza / course website for exam rooms
- Quiz review: Tue March 3rd 7:30-9:30pm in 6-120.

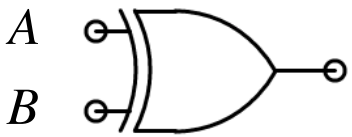
Recap: Logic Optimization

- In practice, tools use Boolean simplification and other techniques to synthesize a circuit that meets certain area, delay, and power goals:



Logic Synthesis (Continued): Other Common Gates

- XOR (Exclusive-OR)

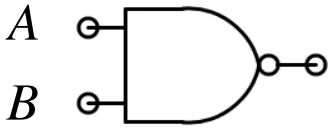


$$Z = A \oplus B = \bar{A} \cdot B + A \cdot \bar{B}$$

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

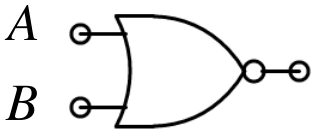
- Inverting logic

NAND



$$Z = \overline{A \cdot B}$$

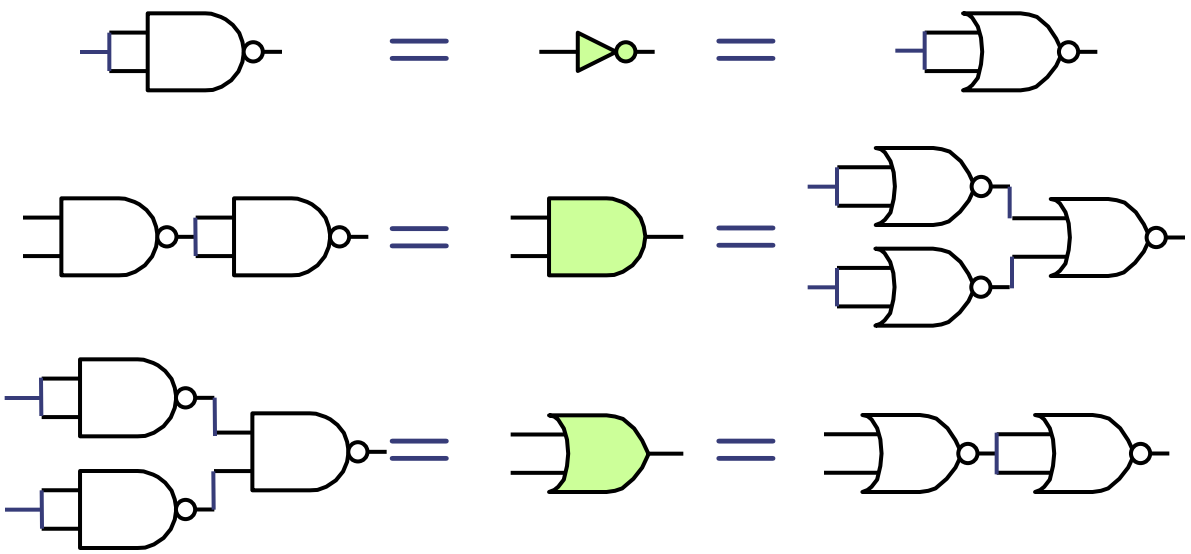
NOR



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

Universal Building Blocks

- NANDs and NORs are universal:



- Any logic function can be implemented using only NANDs (or, equivalently, NORs)

Standard Cell Library

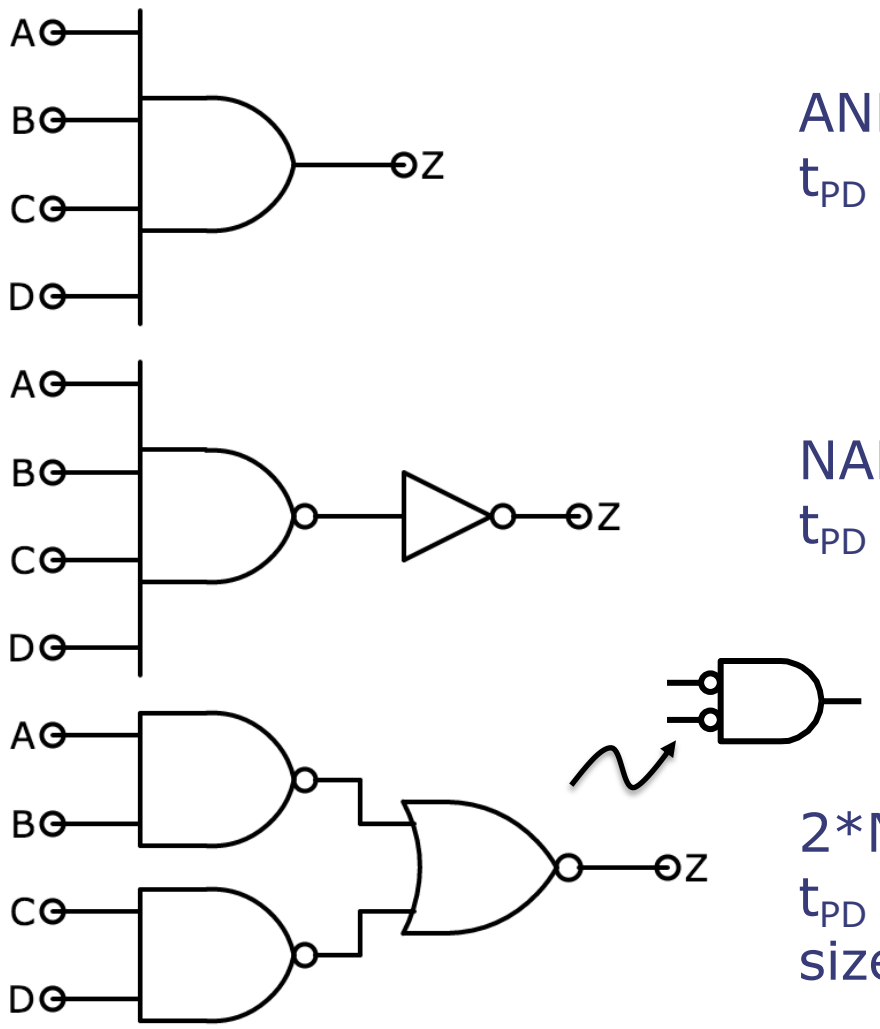
- Library of gates and their physical characteristics
- Example:

Gate	Delay (ps)	Area (μ^2)
Buffer	40	20
Inverter	20	10
AND2	50	25
NAND2	30	15
OR2	55	26
NOR2	35	16
AND4	90	40
NAND4	70	30
OR4	100	42
NOR4	80	32

Observations:

1. In current technology (CMOS), inverting gates are faster and smaller
2. Delay and area grow with number of inputs

Design Tradeoffs: Delay vs Size



AND4:
t_{PD} = 90 ps, size = 40μ²

NAND4 + INV:
t_{PD} = 90 ps, size = 40μ²

Demorgan's Laws:

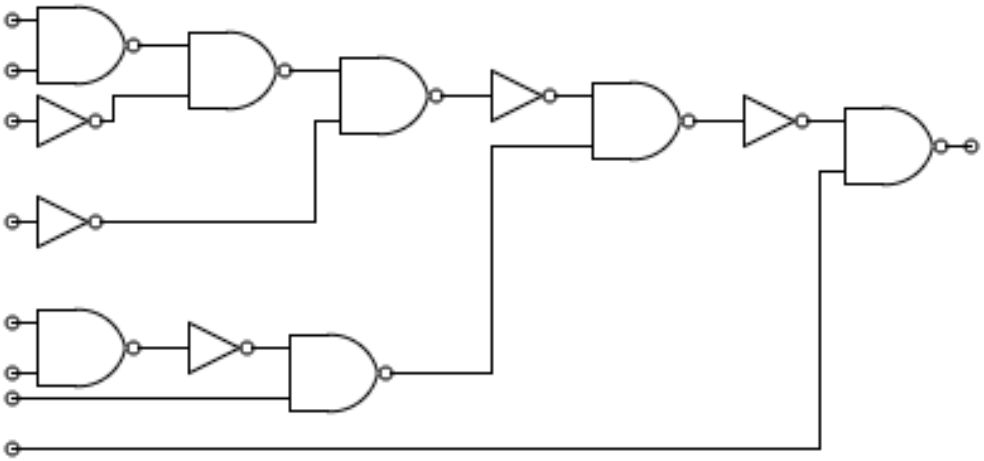
$$\overline{A} \cdot \overline{B} = \overline{A + B}$$

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

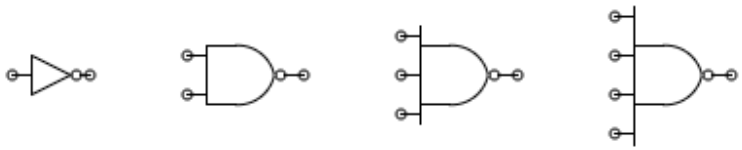
2*NAND2 + NOR2:
t_{PD} = 1 NAND2 + NOR2 = 65 ps,
size = 2 NAND2 + NOR2 = 46μ²

Example: Mapping a Circuit to a Standard Cell Library

Find an implementation of a circuit, e.g.,



Using gates from a standard cell library, e.g.,

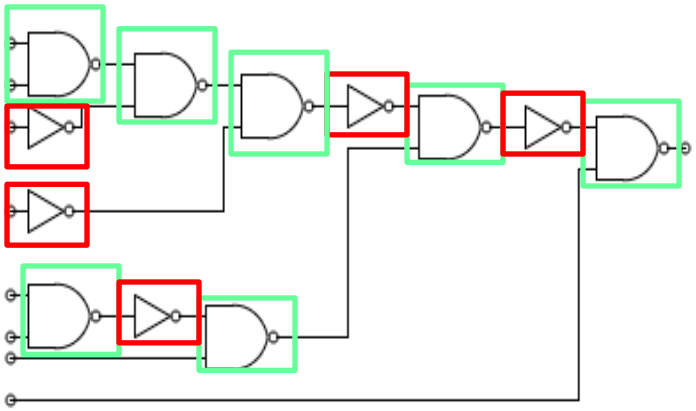


Area	2	3	4	5

That optimizes for some goal, e.g., minimum area

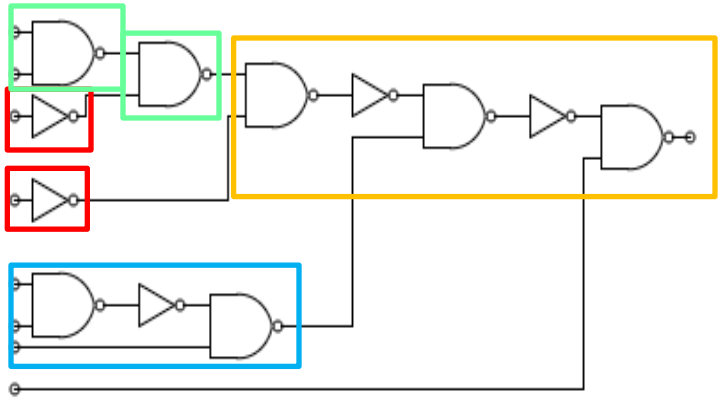
Example: Mapping a Circuit to a Standard Cell Library

Possible implementations:



7 NAND2 (3) = 21
5 INV (2) = 10

Total area cost: 31



2 INV = 4
2 NAND2 = 6
1 NAND3 = 4
1 NAND4 = 5

Total area cost: 19

Logic Optimization Takeaways

- Synthesizing an optimized circuit is a very complex problem
  - Boolean simplification
  - Mapping to cell libraries with many gates
  - Multidimensional tradeoffs (e.g., minimize area-delay-power product)

- Infeasible to do by hand for all but the smallest circuits!

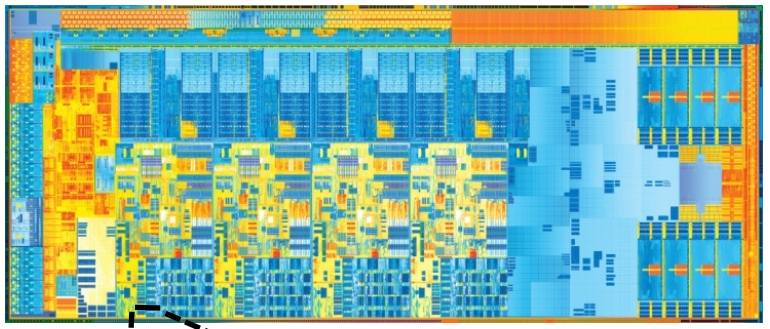
- Instead, hardware designers write circuits in a hardware description language, and use a synthesis tool to derive optimized implementations

CMOS Technology

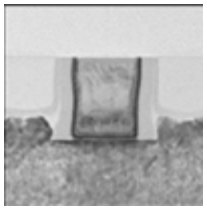
A Deep Dive Into a Chip

Packaged chip

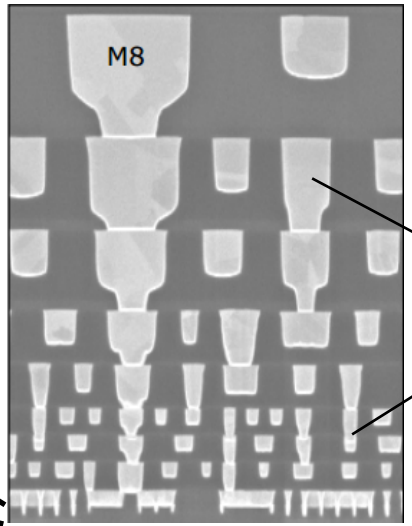
Silicon die (100-400mm^2)



Transistor (FET)



32 nm



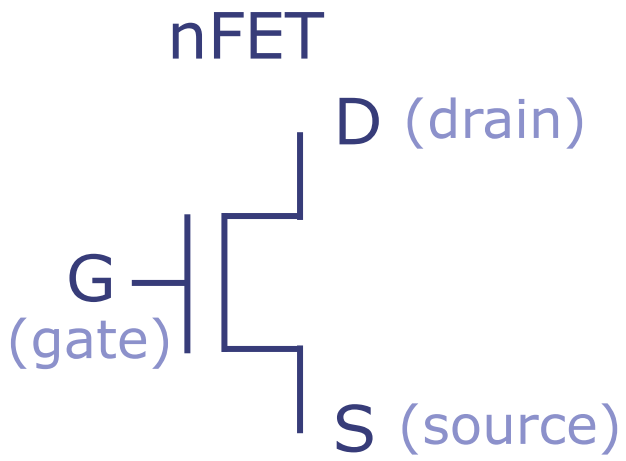
Die cross-section

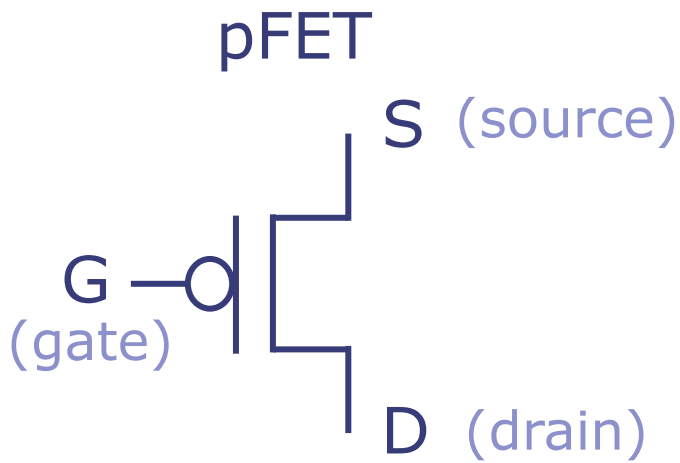
6-15 metal layers (wires)

Source: Intel

Field-Effect Transistors (FETs)

- Nearly all digital systems are built using field-effect transistors, which are voltage-controlled switches
- FETs come in two varieties: nFET and pFET



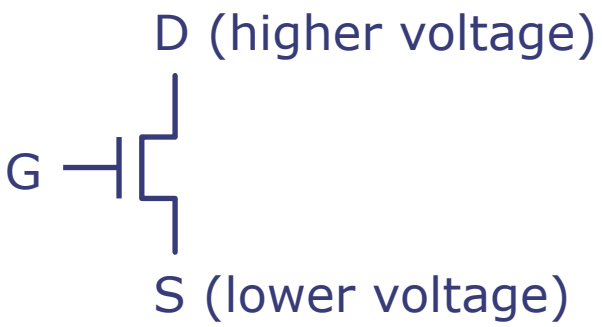


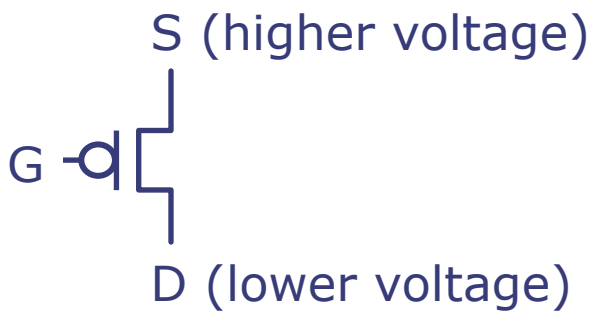
A high voltage at gate (G=1) creates conducting path between source and drain

A low voltage at gate (G=0) creates conducting path between source and drain

Labeling Source and Drain

- There is no physical difference between source and drain, called the diffusion terminals
- By convention, we label diffusion terminals as source or drain depending on their voltages:
  - On nFETs, source = diffusion terminal at lower voltage
  - On pFETs, source = diffusion terminal at higher voltage

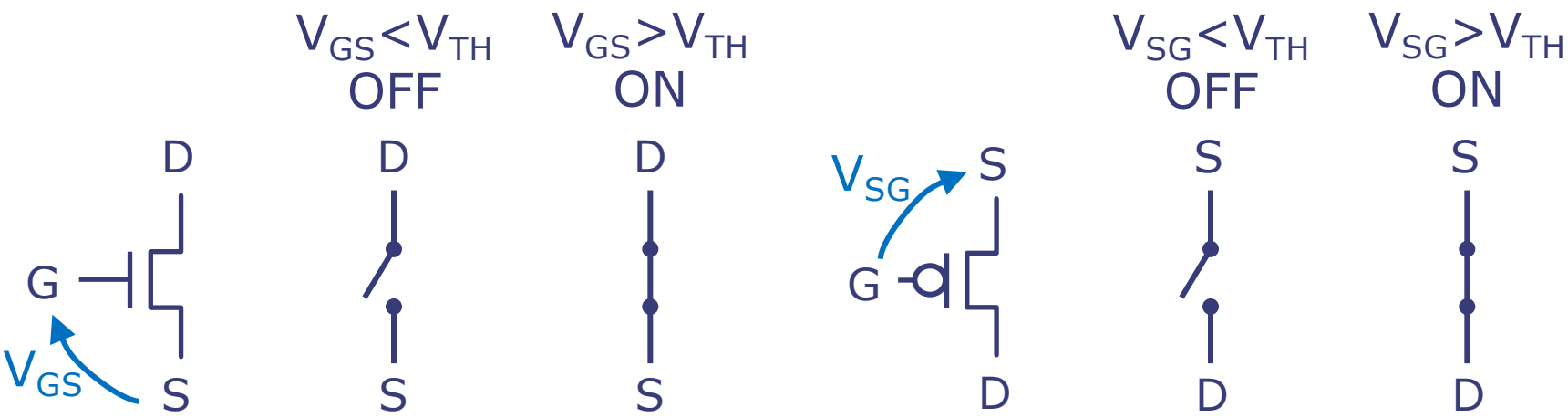




- This convention lets us define the behavior of FETs using the voltage between gate and source

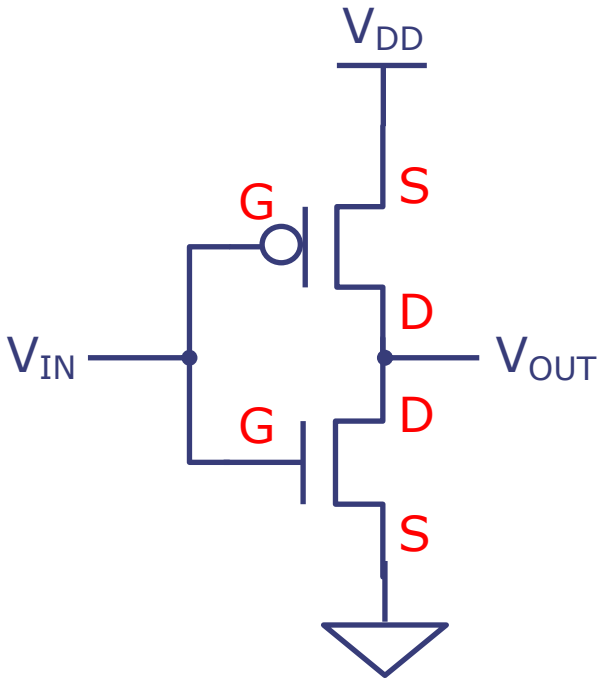
FET Switching Model

- FETs have a threshold voltage V_{TH}
- nFET is ON if the voltage between gate and source V_{GS} exceeds V_{TH}, OFF otherwise
- pFET is ON if the voltage between source and gate V_{SG} exceeds V_{TH}, OFF otherwise



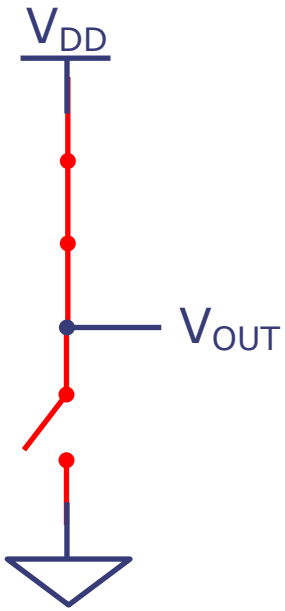
- This is a very simplified model, but it is sufficient to build logic gates

What Does This Circuit Compute?

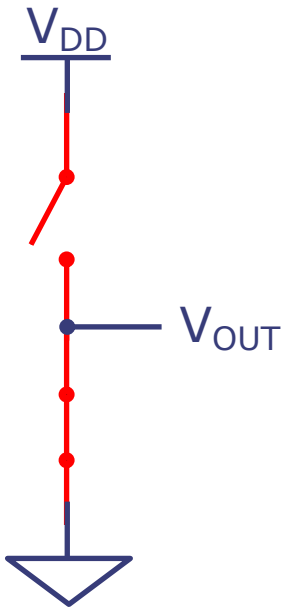


Assume $V_{TH} < V_{DD}/2$

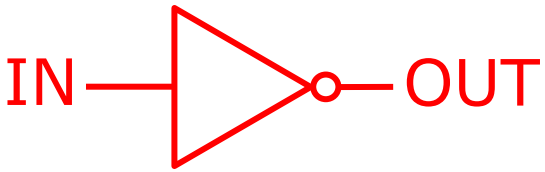
$V_{IN} < V_{TH}$



$V_{IN} > V_{DD} - V_{TH}$



CMOS inverter

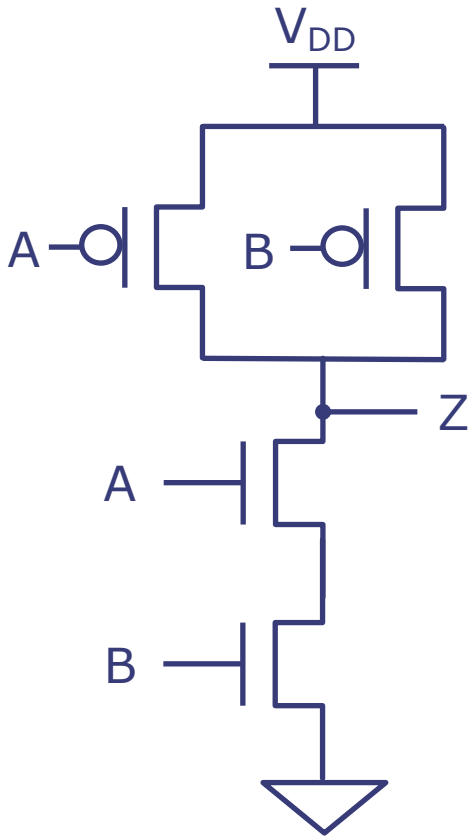


IN	OUT
0	1
1	0

Note on Terminology

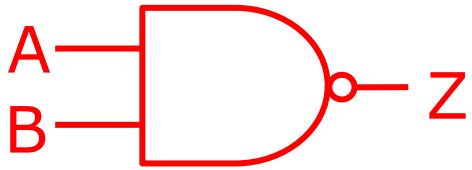
- MOSFETs (metal-oxide-semiconductor field-effect transistors) are the most common type of FET
- nFET and pFET are sometimes abbreviated as nMOS and pMOS
- CMOS stands for complementary MOS

What Does This Circuit Compute?



A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

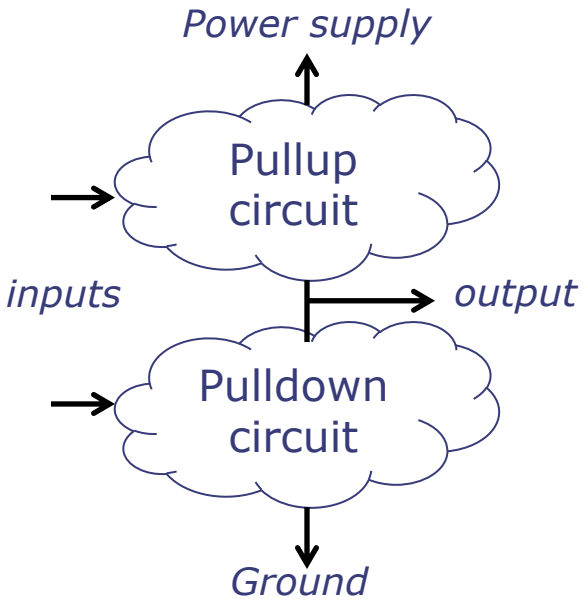
CMOS NAND gate



CMOS Logic

- CMOS gates have complementary pullup and pulldown networks, i.e., the pullup is on when the pulldown is off and vice versa

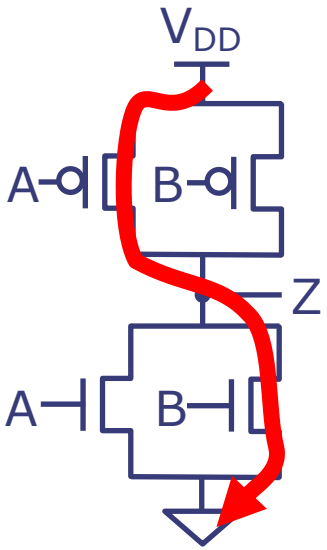
pullup	pulldown	F(inputs)
on	off	driven "1"
off	on	driven "0"
on	on	driven "X"
off	off	no connection



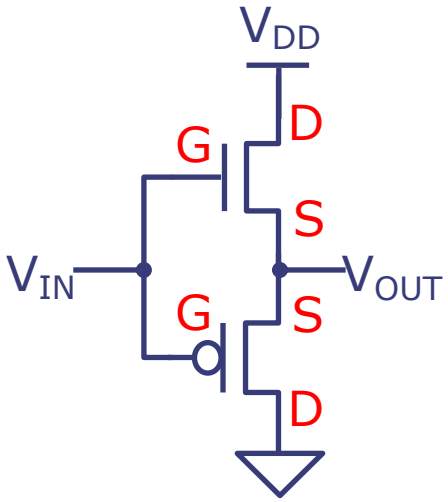
- CMOS uses pFETs to implement the pullup network and nFETs to implement the pulldown network

Some Questionable Gates

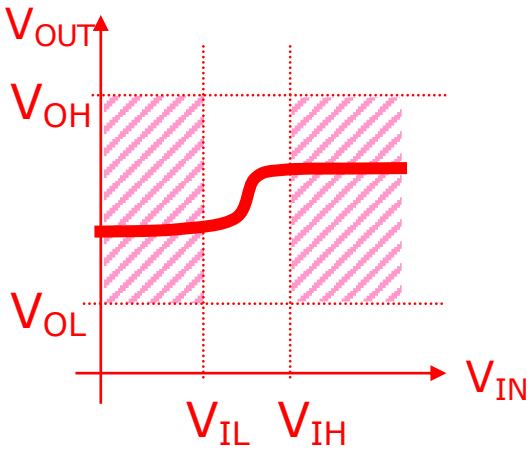
- What can go wrong with the following gates?



A=0 B=1 or A=1 B=0
connect supply and ground

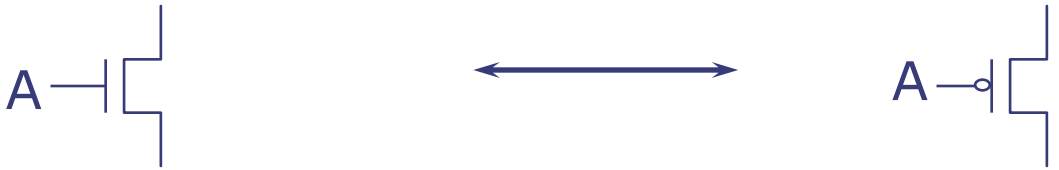


pFET doesn't pull down
V_{OUT} below V_{TH}
nFET doesn't pull up
V_{OUT} above $V_{DD} - V_{TH}$



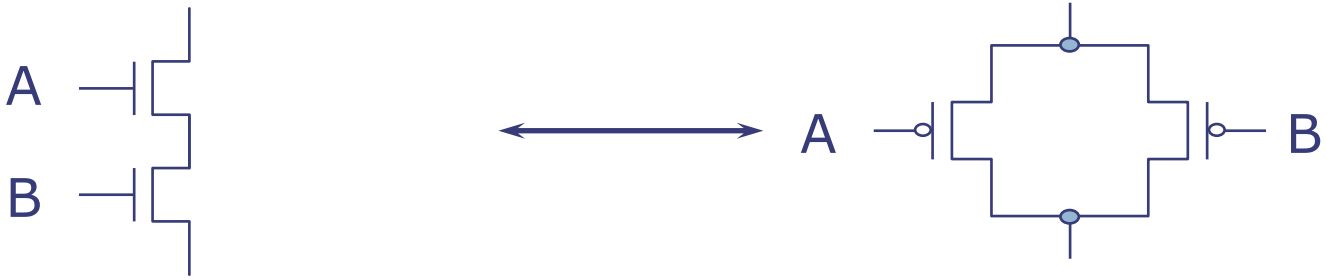
- CMOS Rule #1: Complementary pullup and pulldown networks
- CMOS Rule #2: pFETs in pullup, nFETs in pulldown

CMOS Complements



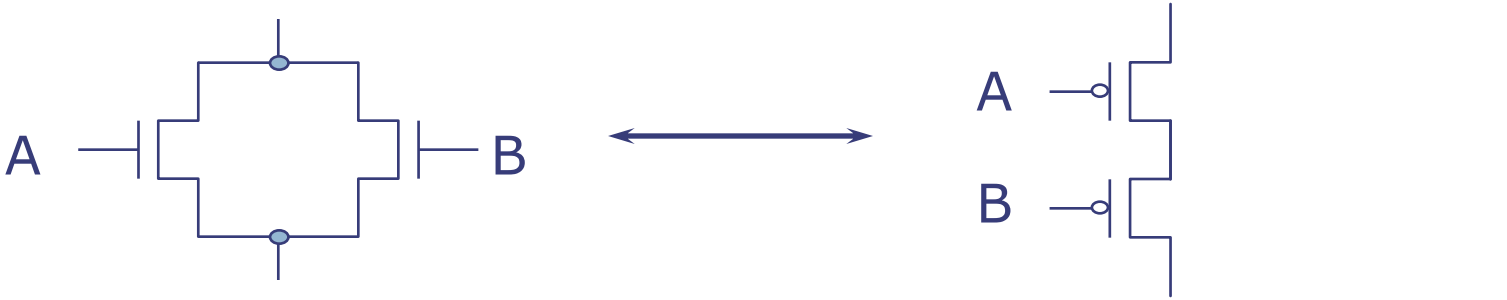
conducts when A is high

conducts when A is low: $\overline{A}$



conducts when A is high *and* B is high: $A \cdot B$

conducts when A is low *or* B is low: $\overline{A} + \overline{B} = \overline{A \cdot B}$



conducts when A is high *or* B is high: $A + B$

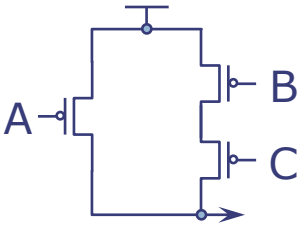
conducts when A is low *and* B is low: $\overline{A} \cdot \overline{B} = \overline{A + B}$

General CMOS Gate Recipe

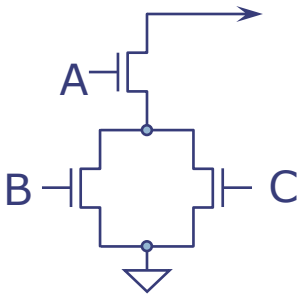
Step 1. Derive the pullup network that does what you want, e.g.,

$$F = \overline{A} + \overline{B} \cdot \overline{C}$$

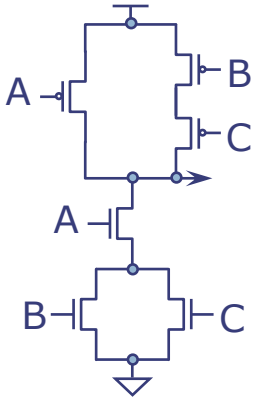
(Determine what combination of inputs generates a high output)



Step 2. Derive complementary pulldown network: replace pFETs with nFETs, series subnets with parallel subnets, and parallel subnets with series subnets



Step 3. Combine pFET pullup network from Step 1 with nFET pulldown network from Step 2 to form the CMOS gate.



Can CMOS gates implement arbitrary functions? *No*

CMOS Gates are Inverting

- In a CMOS gate, rising inputs (0→1) lead to falling outputs (1→0) and vice versa
- On a rising input,
  - nFETs go OFF→ON, so pulldown may connect output to ground
  - pFETs go ON→OFF, so pullup may disconnect output from V_{DD}
  - Output either stays the same or falls
- Corollary: Cannot build non-inverting logic using a single CMOS gate
  - Example: AND

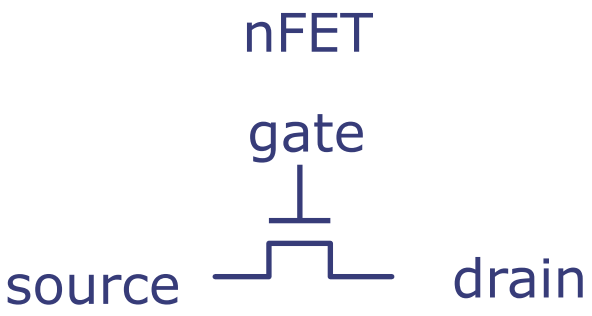
A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1

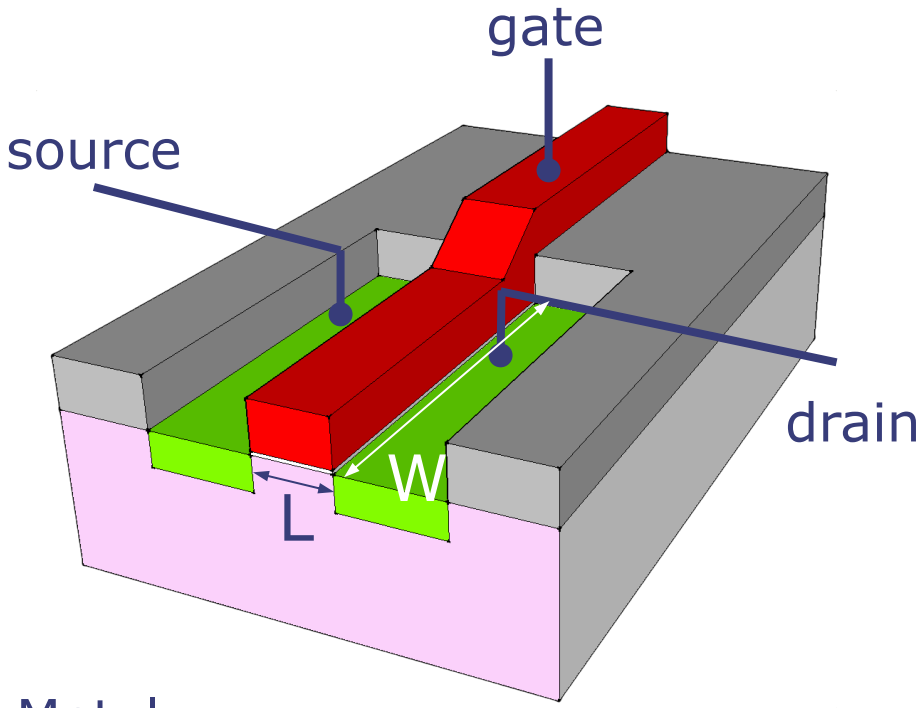
rising input rising output

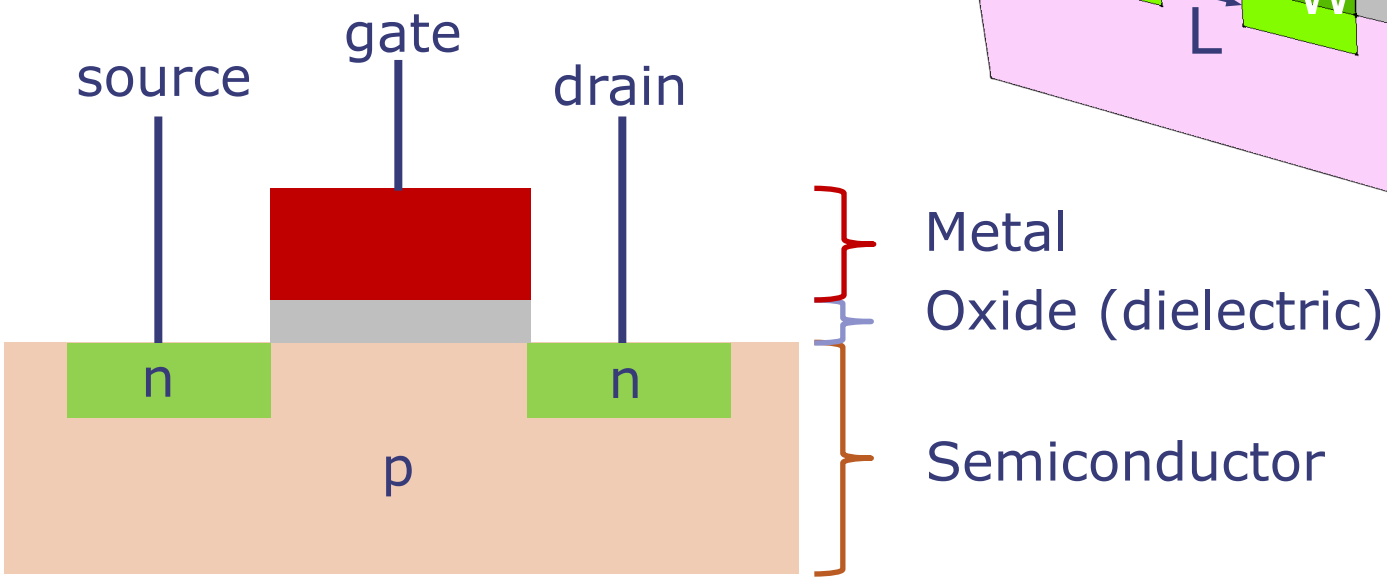
Analyzing the Delay, Area, and Power of CMOS Gates

NOTE: Demystification, will not be on the quiz

MOSFET Physical Structure

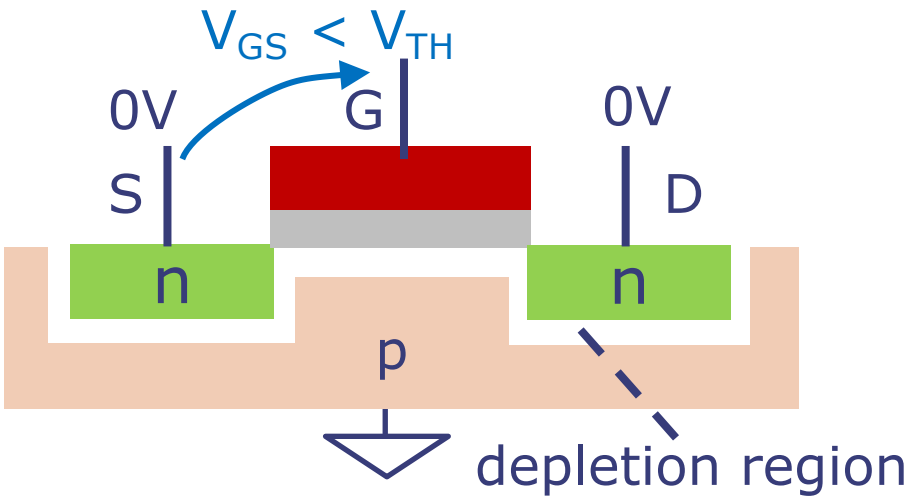




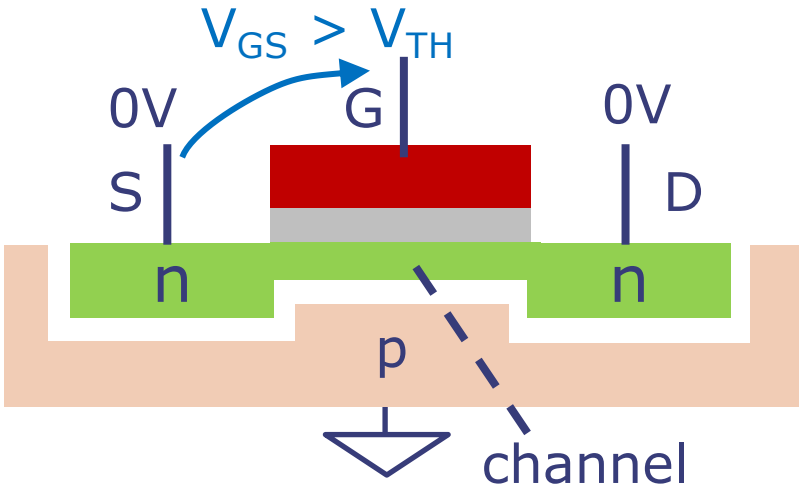


MOSFET Electrical View

With $V_{GS} < V_{TH}$, almost no current flows between source and drain

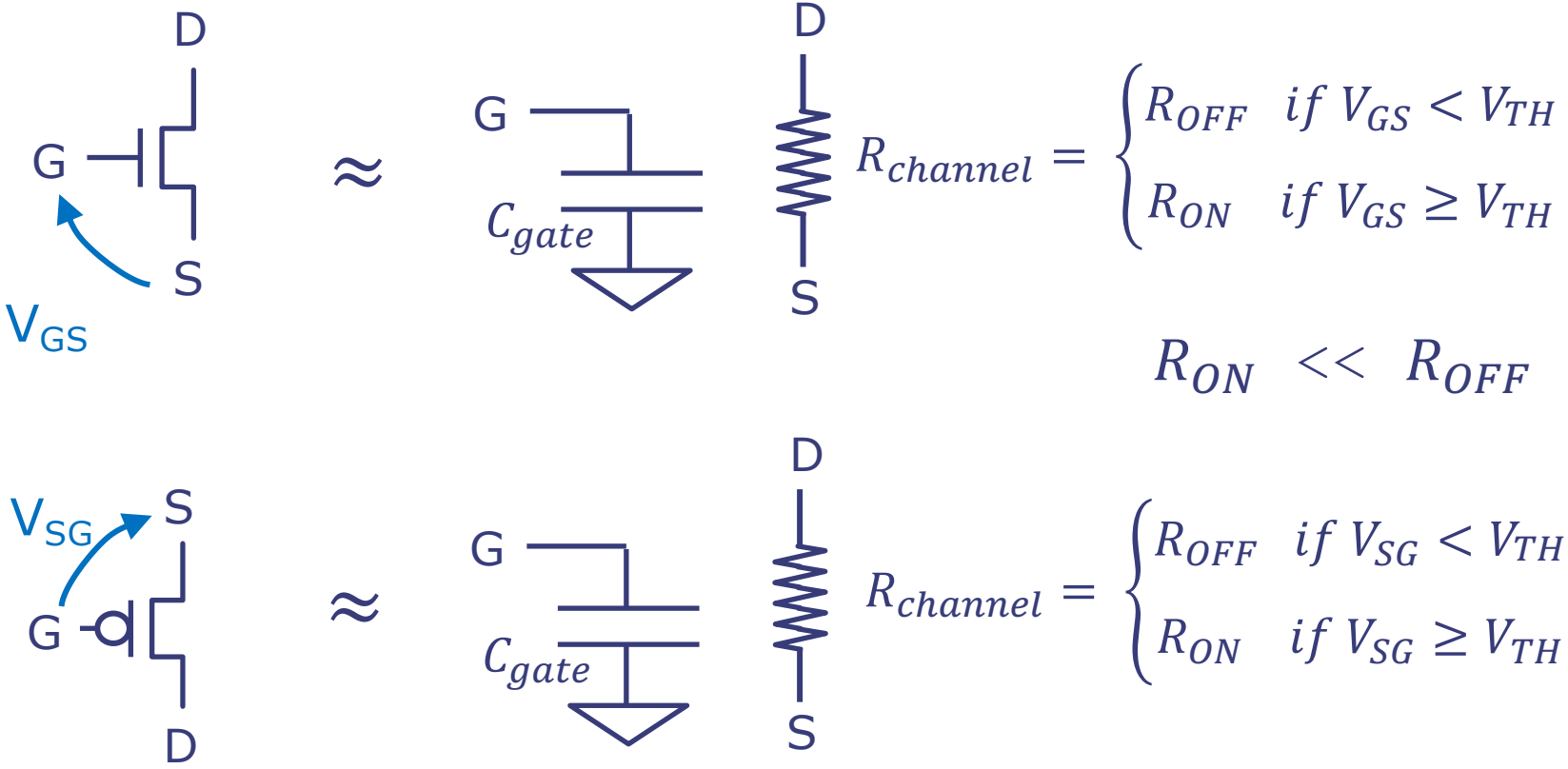


As V_{GS} reaches V_{TH}, a channel forms between source and drain



The shape of the channel (and its resistance) also depends on the voltage at the drain. But a low-resistance channel will exist while $V_{GS} > V_{TH}$

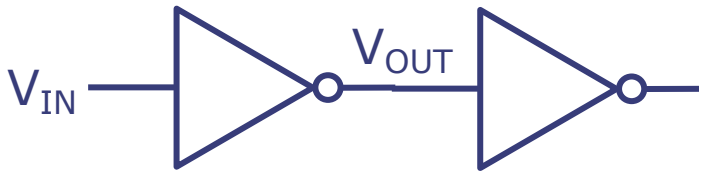
FET *First-Order* Electrical Model

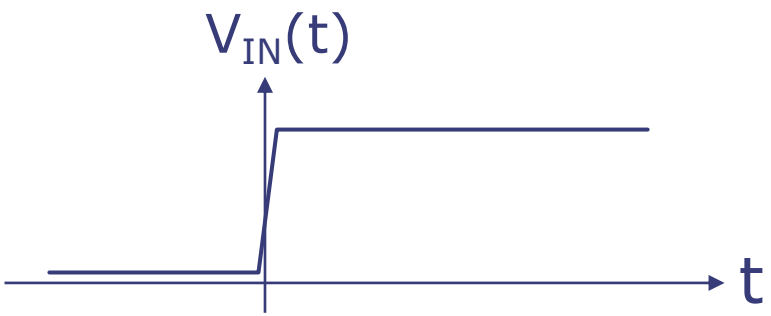


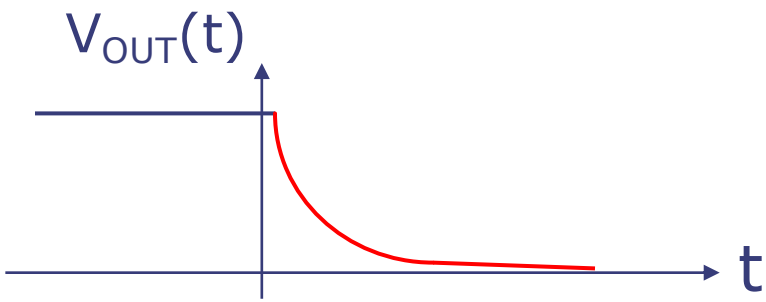
- Simplest possible model that lets us reason about delay, area, and power. Not very accurate!

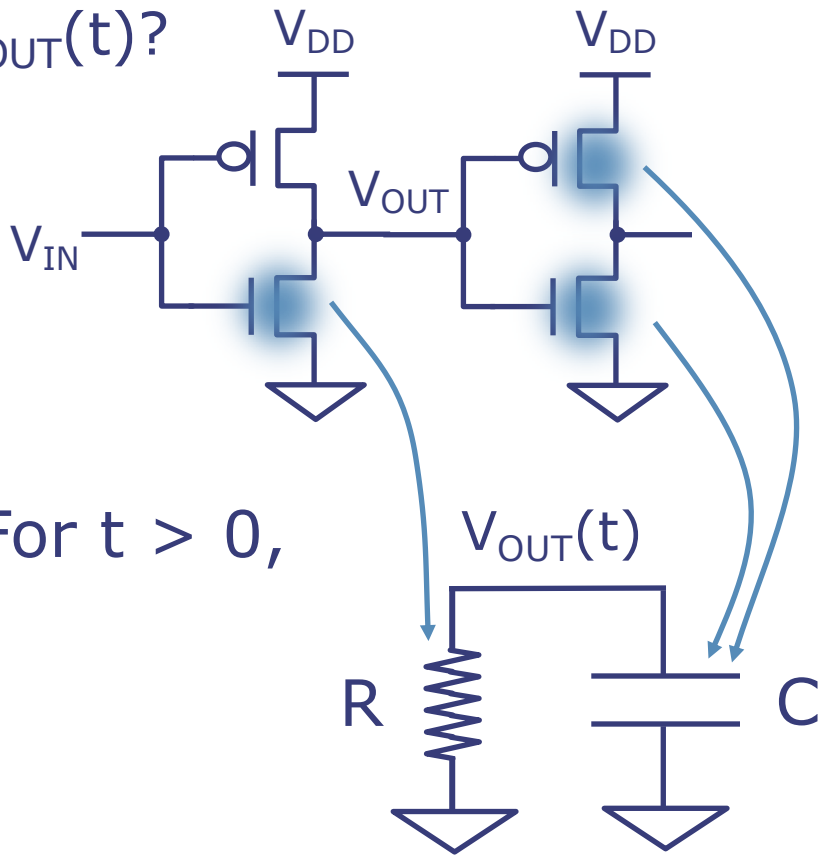
CMOS Gate Delay

Consider the following circuit.
Given $V_{IN}(t)$, can you derive $V_{OUT}(t)$?





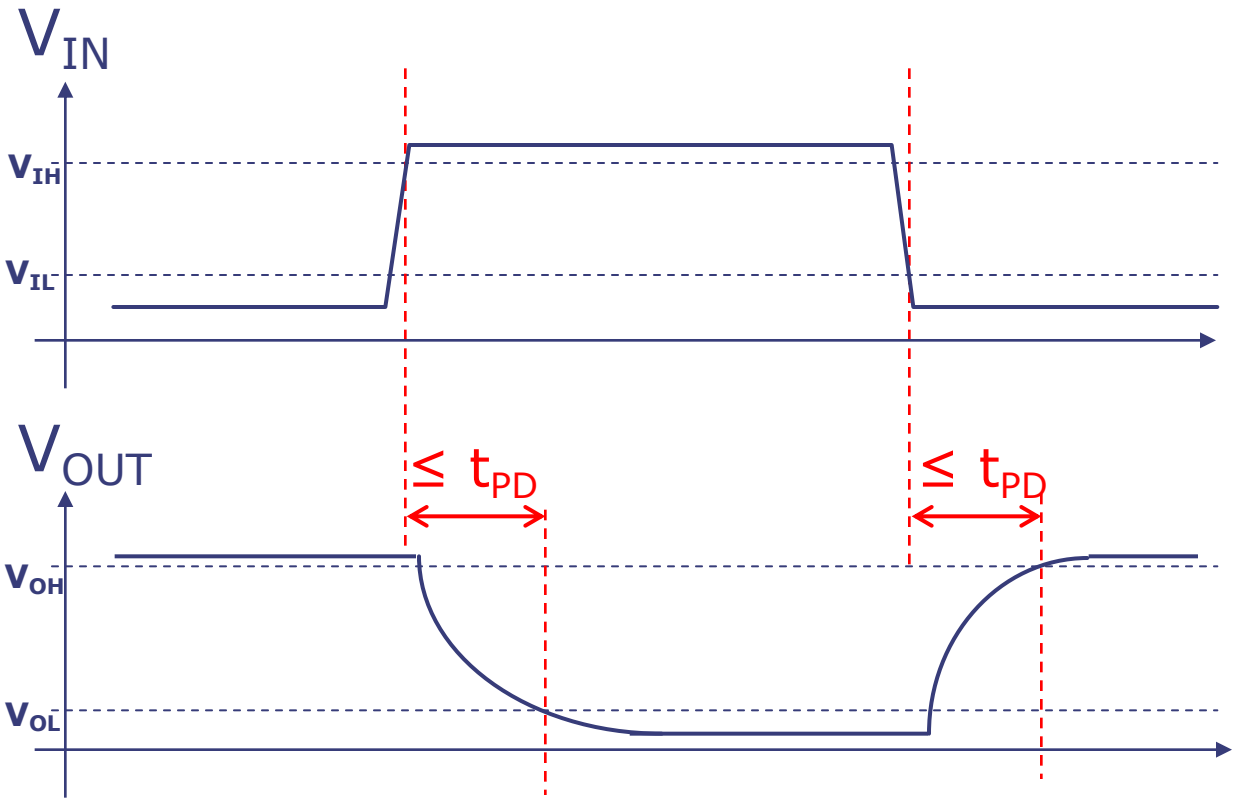




$$V_{OUT}(t) = VOUT(0)e^{-t/RC}$$

Propagation Delay

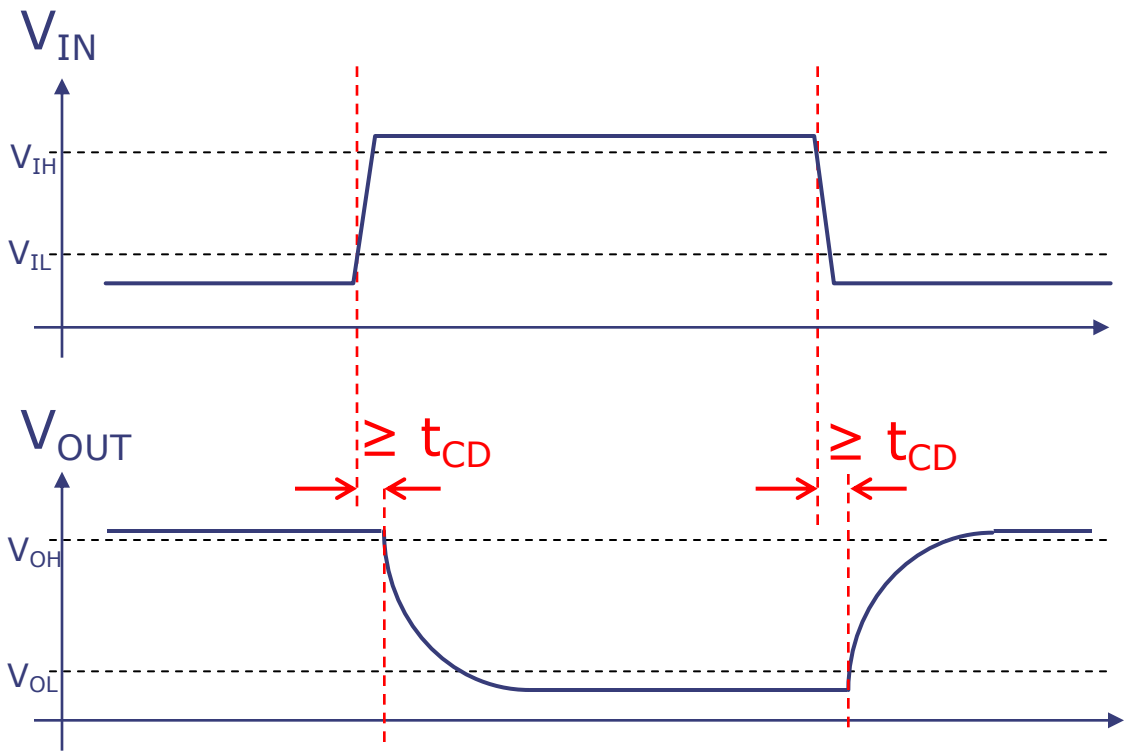
Propagation delay (t_{PD}): Upper bound on the delay from valid inputs to valid outputs.



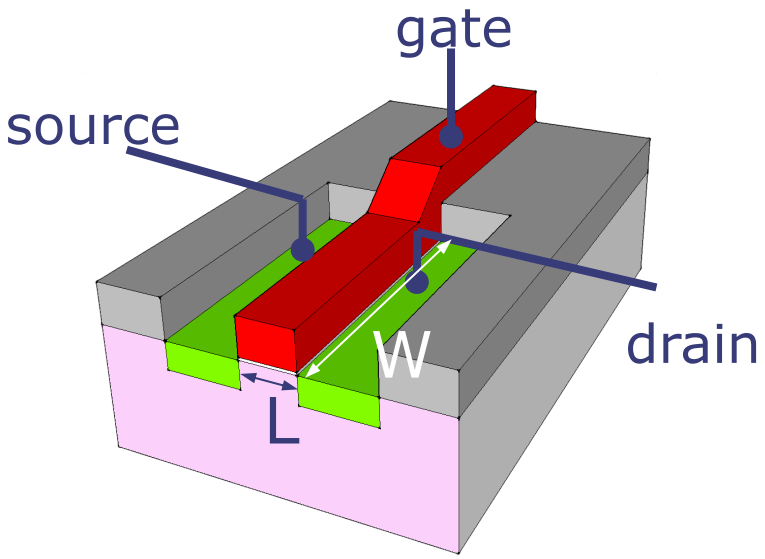
To minimize t_{PD}, must keep resistances and capacitances low

Contamination Delay

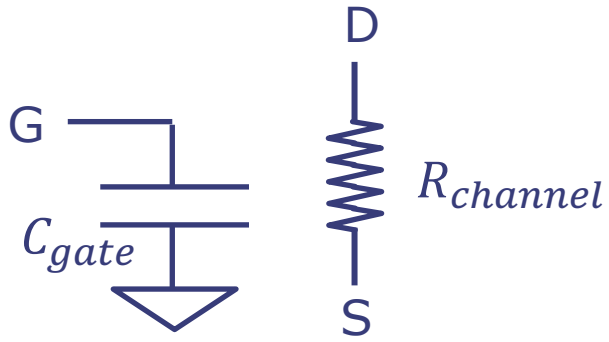
Contamination delay (t_{CD}): Lower bound on the delay from any invalid input to an invalid output



MOSFET Sizing



≈



How do C_{gate} and $R_{channel}$ change with L and W?

$$C_{gate} \propto L \cdot W$$

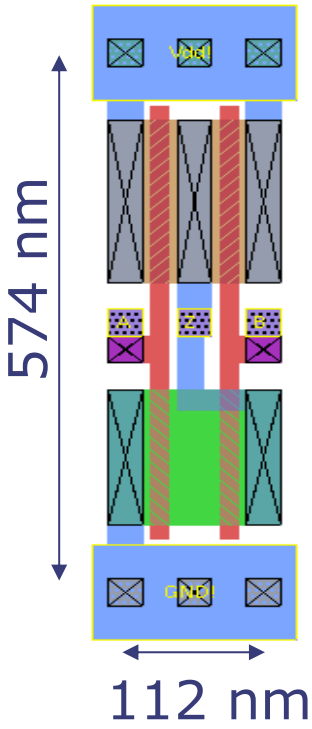
$$R_{channel} \propto L/W$$

- CMOS gates use MOSFETs with smallest possible L and choose W to set performance
  - Wider FETs drive more current (lower R), but their gates are harder to drive (higher C) and they take more area

Standard Cell Libraries

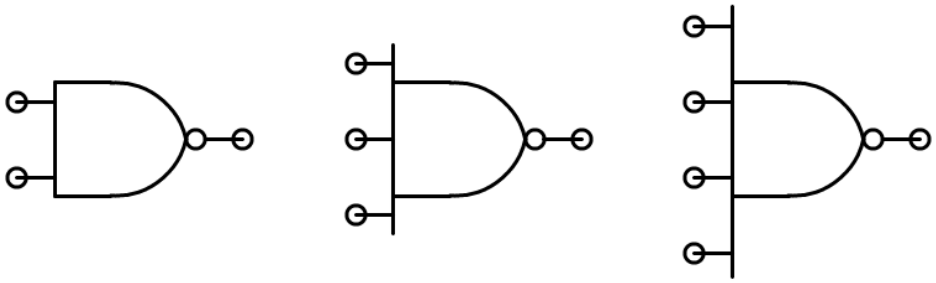
- A standard cell library provides implementations of common gates (NAND, NOR, XOR, etc.) for a specific implementation technology
- Each gate includes
  - Electrical parameters (e.g., Rs and Cs)
  - Physical layout
- Synthesis tools use gates from the standard library instead of sizing and placing individual transistors

Example:
NAND2



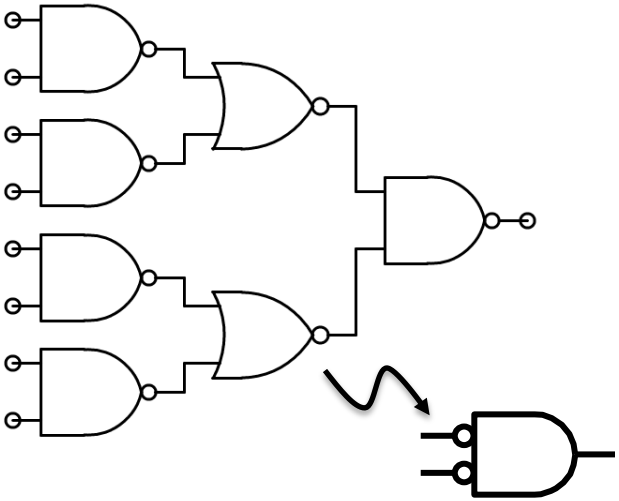
Wide (High-Fanin) Gates

Most standard cell libraries include 2-, 3- and 4-input devices:



But for a large number of inputs, the series connections of too many MOSFETs can lead to very large effective $R_{pulldown}$ or R_{pullup}. Instead, use trees of smaller devices…

Example: 8-input NAND



How does t_{PD} grow with the number of inputs N?

If we use a single CMOS gate, $t_{PD} \propto N$

If we use a tree of gates, $t_{PD} \propto \log(N)$

CMOS Power Dissipation

- Total power dissipation: $P = P_{dynamic} + P_{static}$
- Dynamic power: Caused by 0↔1 transitions of nodes in the circuit
  - Charging/discharging each capacitor consumes $\frac{1}{2}CV_{DD}^2$ energy
  - If on average C_S capacitance across the chip switches each cycle, and there are f_{CLK} cycles per second

$$P_{dynamic} = \frac{1}{2} C_S V_{DD}^2 f_{CLK}$$

- Static power: Caused by
  - Subthreshold leakage: Even when the FET is off, a very small current flows from source to drain ($R_{OFF} < \infty$)
  - Tunneling current: Gate and channel are separated by a very thin (<1nm) dielectric, so some electrons tunnel through

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

  - Static power is typically 10-30% of total power

Summary

- FETs behave as voltage-controlled switches
- CMOS gates:
  - Use complementary pullup and pulldown networks
  - Use pFETs in pullup, nFETs in pulldown network
- CMOS gates are inverting (rising inputs can only cause falling outputs, and vice versa)

Thank you!

Next lecture:

Combinational devices,
introduction to minispec