
CS349G: Selected Reading of Ph.D. Dissertations

<https://web.stanford.edu/class/cs349g/>

John Hennessy and Philip Levis
Stanford University
pal@cs.stanford.edu, hennessy@stanford.edu

Lots of material in this lecture borrowed from
Subhasish Mitra and Bill Dally (EE108)

Lecture Outline

Course overview

- Educational goals
- Course logistics (reading, assignments)
- Course Schedule

Introduction to digital circuits

- The Digital Abstraction
- Physical Constraints and Speed of Digital Circuits

Next lecture: basics of computer architecture (John)

Course Overview

A Ph.D. Dissertation

Rosy view: the breath-taking summary of your greatest research achievements while earning a Ph.D.

Cynical view: the document you have to convince some professors to sign in order to graduate.

Key stipulation: Describes the results of *research*, so contains new knowledge and ideas. Has to convince readers that it does.

Educational Goals (more than one may apply to you)

For Ph.D. students

- See what dissertations of leading scholars look like
- Learn how they are different than papers and why

For students interested in research

- Gain a better sense of what the actual results of research look like (when not condensed to 8-12 pages)
- See how many things we learn as fundamentals today are based on yesterday's research

For students interested in material

- See a glimpse into cutting edge computer architecture in the golden age of the field

Course Logistics

We will read 5 Ph.D. dissertations

- You may choose to not read one of them, and read a dissertation of your choice (in architecture) instead.
- If you have not taken EE282, you replace the second half of first dissertation (Mark Dean's) with background reading. Files are on Canvas.

This class is a *reading* class! You should expect to spend 6-8 hours/week reading and trying to understand the work.

- Spending 30 minutes to dig into and learn something you didn't know (e.g., what a clock buffer is) means success

Five Dissertations



Mark Dean (1992)

STRiP: A Self-Timed RISC Processor

Idea: Dynamically adjust clock speed of processor



Marc Hannah (1985)

Parallel Architectures for Computer Graphics Displays

Idea: Use parallel hardware for graphics



Kunle Olukotun (1996)

The Case for a Single-Chip Multiprocessor*

Idea: Multicore processors > multiple processors



Valerie Taylor (1991)

Application-specific architectures for large finite element applications

Idea: Hardware accelerator for sparse matrices



Timothy Pinkston (1991)

The GLORI architecture for multiprocessors

Idea: Connect processors with optics (light, not electricity)

Course Schedule

	Lecture	Reading	
Background, reading dissertations	2 weeks	2 weeks	Mark Dean
Core contributions of reading	2 weeks	2 weeks	Marc Hannah
Guest lectures, architecture today	6 weeks	2 weeks	Valerie Taylor
		2 weeks	Kunle Olukotun
		2 weeks	Timothy Pinkston

Digital Signals

Digital Circuits

This entire class is about computer architecture.

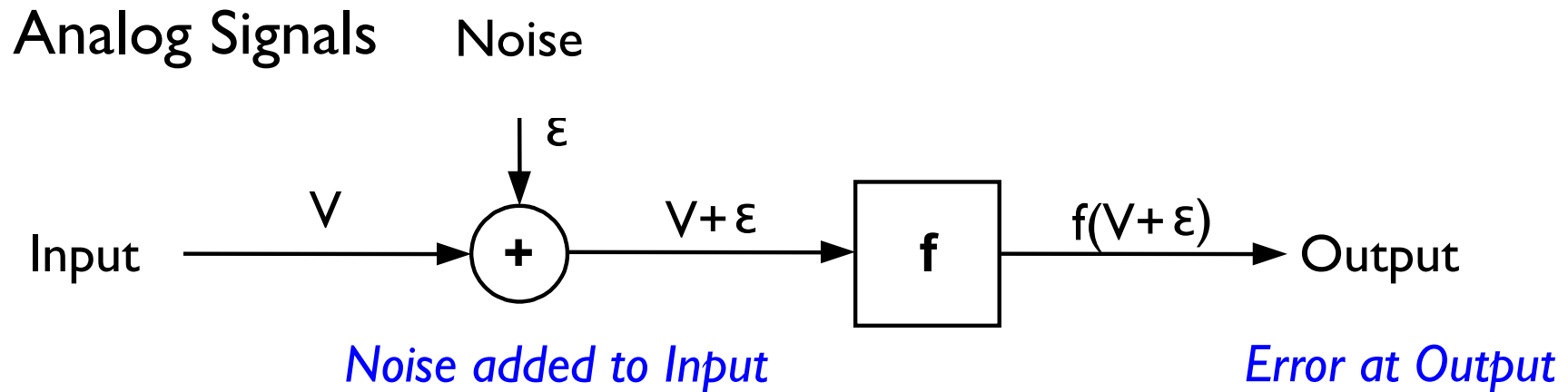
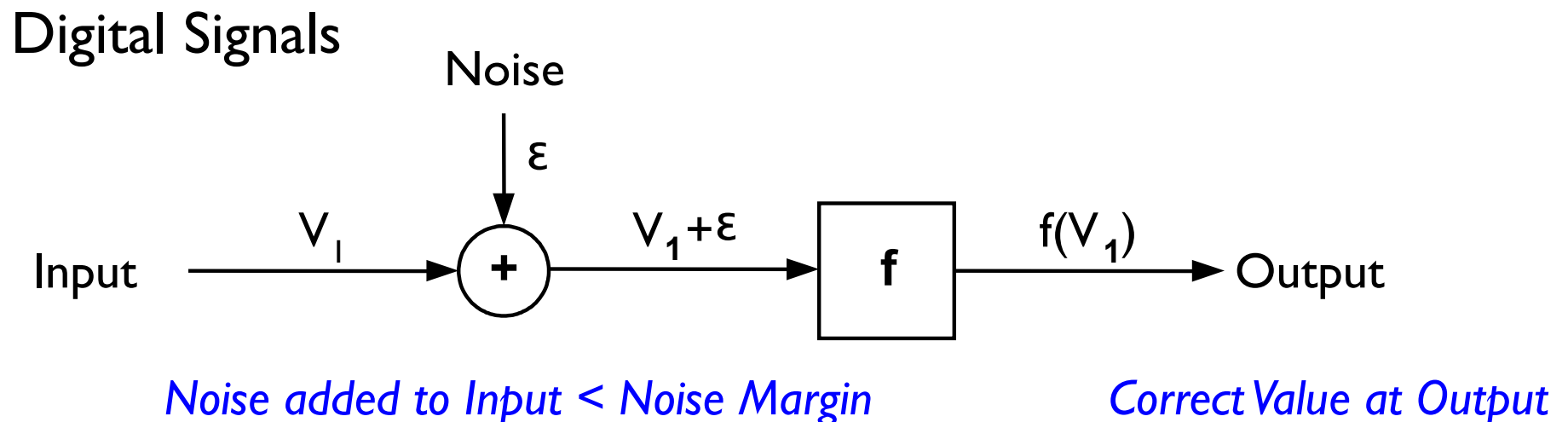
These first two lectures are a crash course on performance, why, and basic techniques to improve it. This is to help understand the reading.

We're not expecting you to be a digital designer after these 2 lectures. But we want you to understand *why* digital designers make certain decisions. These foundations are critical.

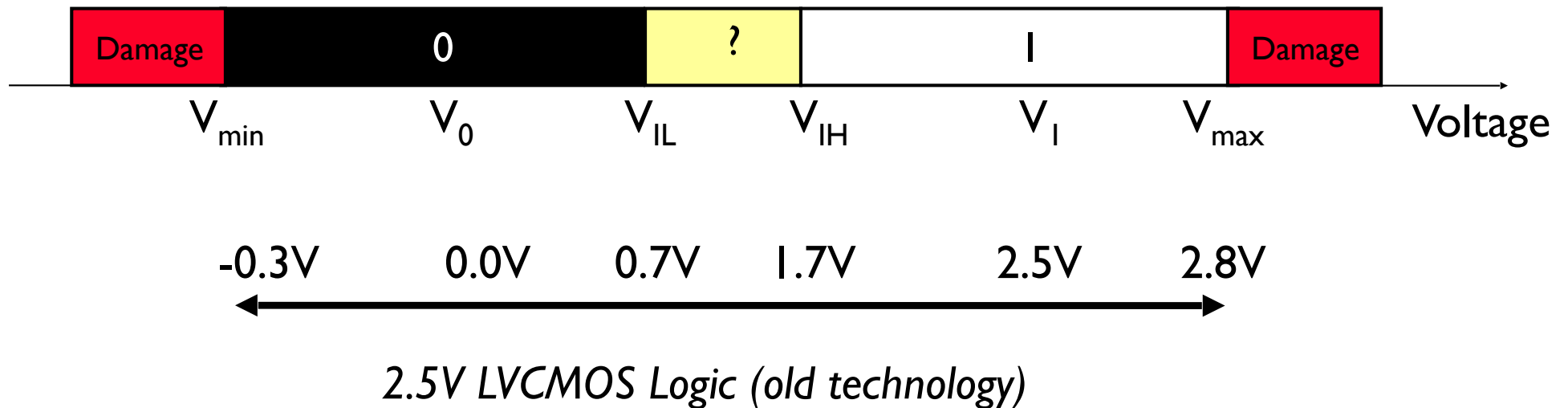
Today: digital circuits. What governs their speed, how do you make them faster? (EE108)

Wednesday: architecture and techniques. How do we use faster circuits to make programs faster? (EE180)

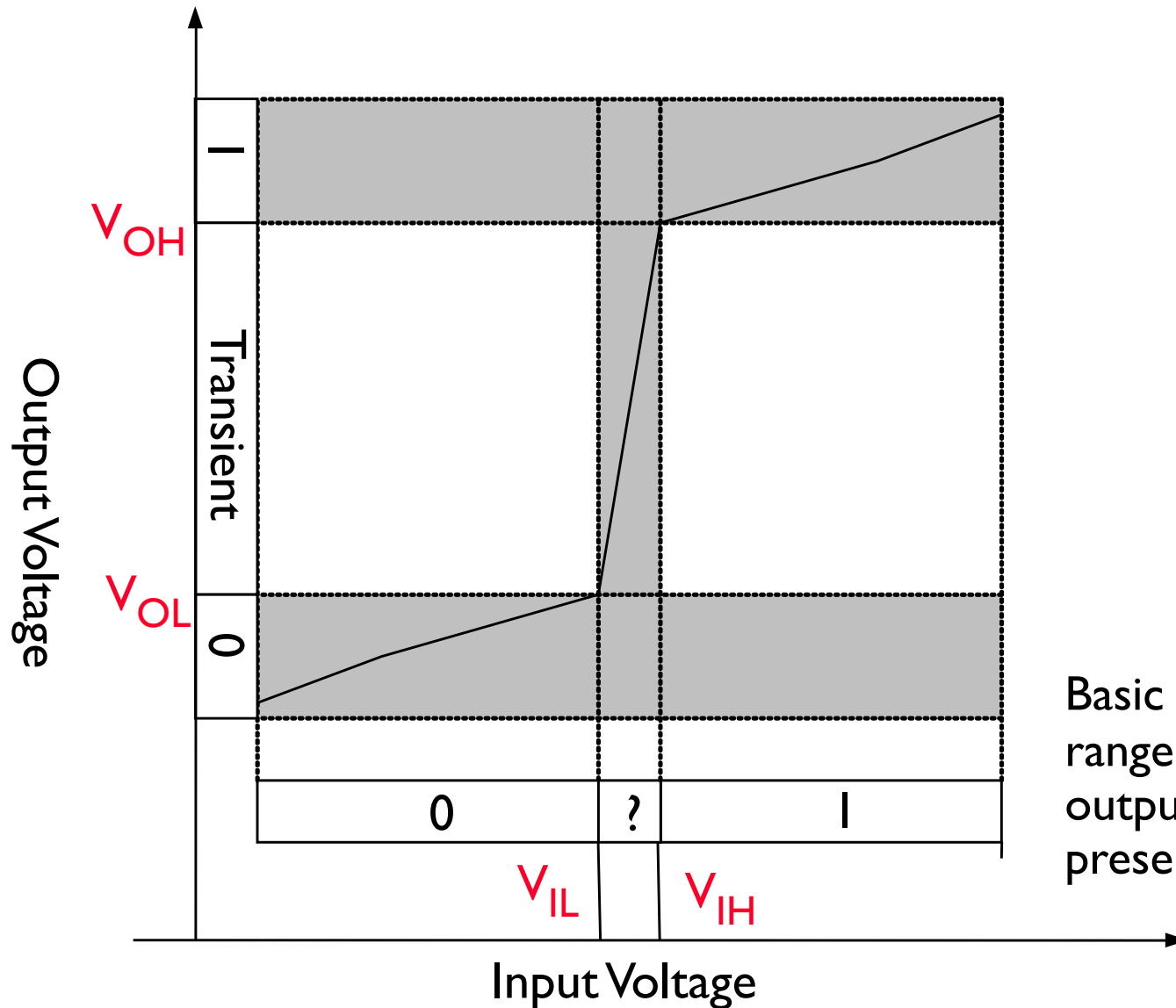
Effect of Noise on Analog & Digital Signals



Voltage Ranges of Binary Signals



Digital Voltage Transfer Function



V_{IL} : Maximum input voltage recognized as logic 0

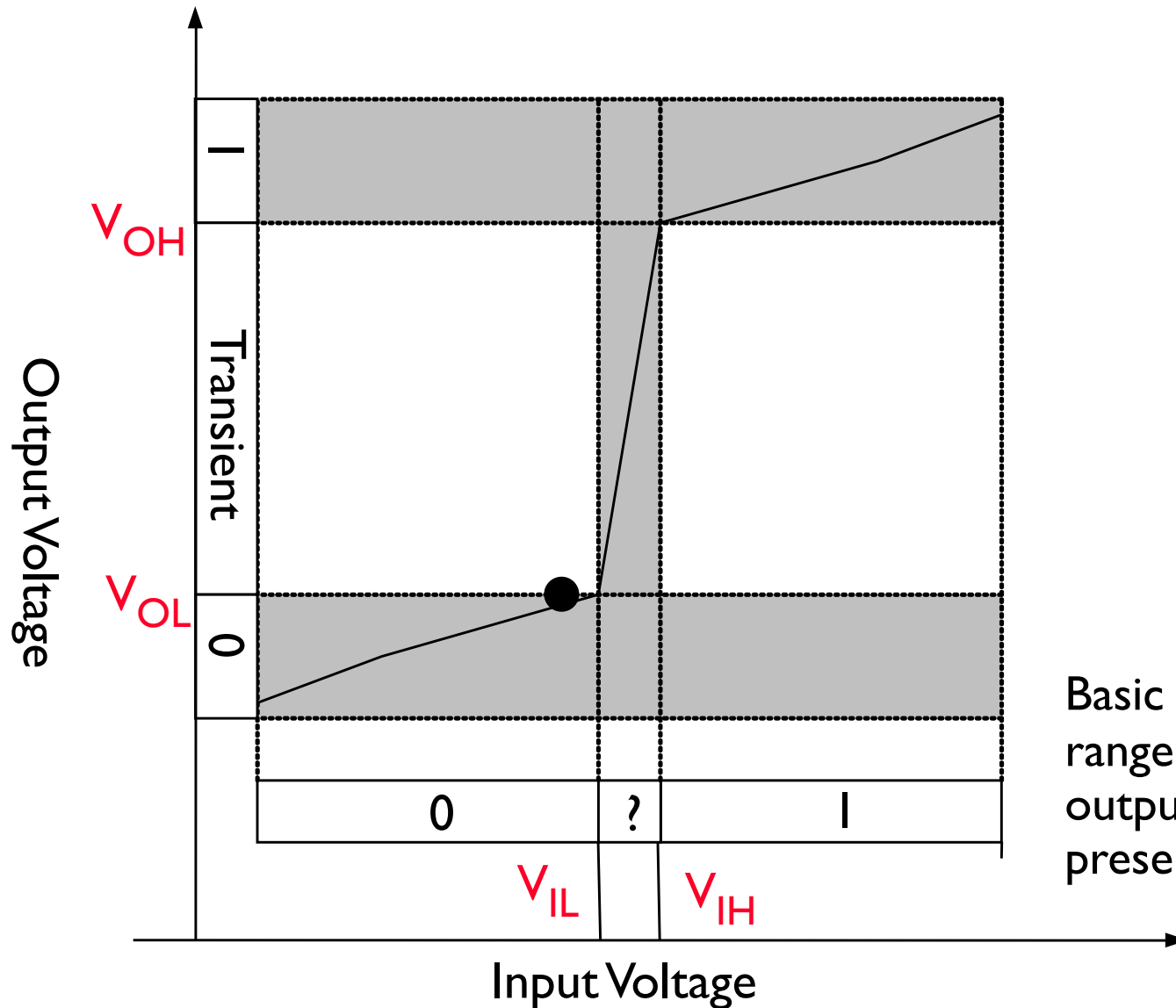
V_{IH} : Minimum input voltage recognized as logic 1

V_{OL} : Maximum output voltage corresponding to logic 0

V_{OH} : Minimum output voltage corresponding to logic 1

Basic idea: accept a much wider range of input than resulting output. Restore signal in presence of noise.

Digital Voltage Transfer Function



V_{IL} : Maximum input voltage recognized as logic 0

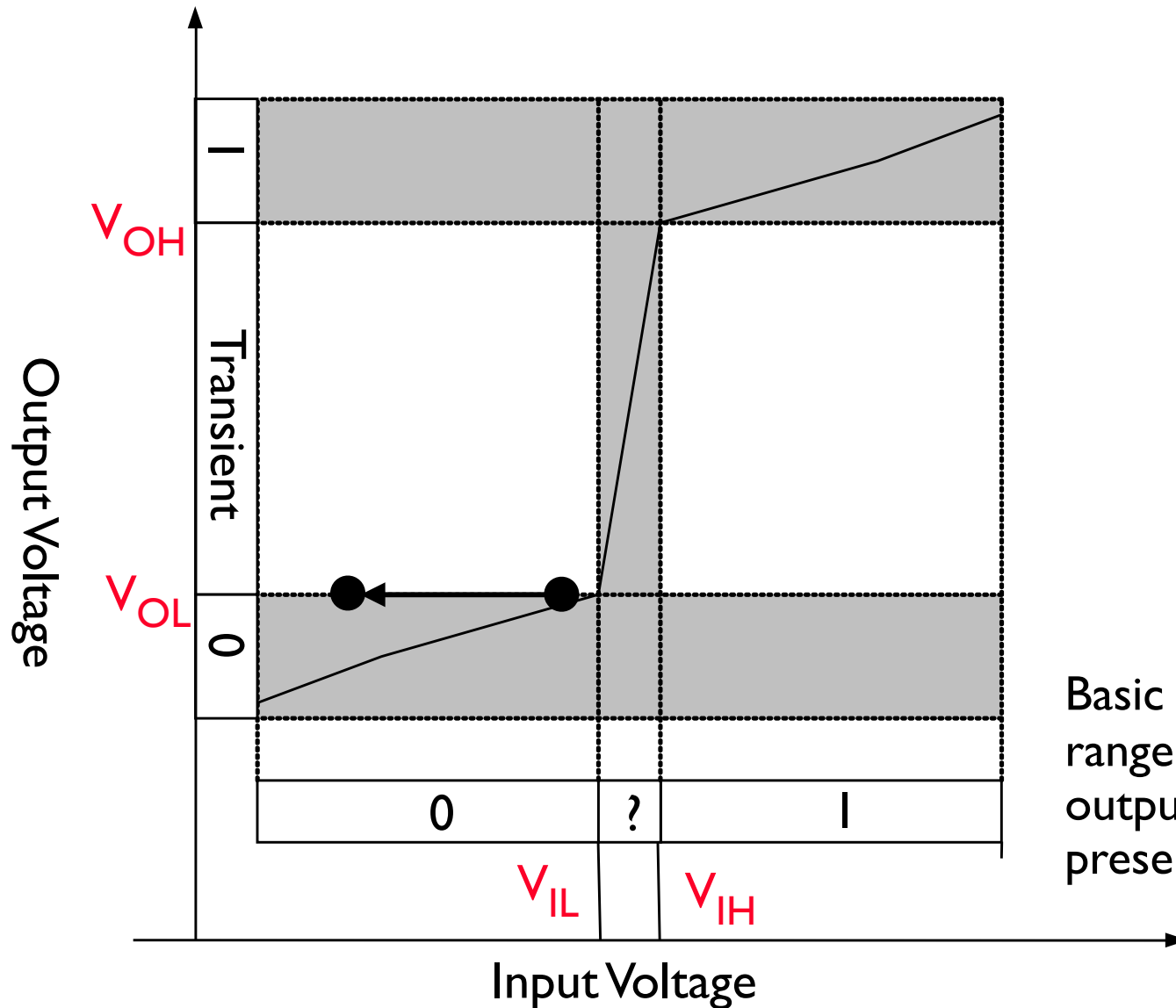
V_{IH} : Minimum input voltage recognized as logic 1

V_{OL} : Maximum output voltage corresponding to logic 0

V_{OH} : Minimum output voltage corresponding to logic 1

Basic idea: accept a much wider range of input than resulting output. Restore signal in presence of noise.

Digital Voltage Transfer Function



V_{IL} : Maximum input voltage recognized as logic 0

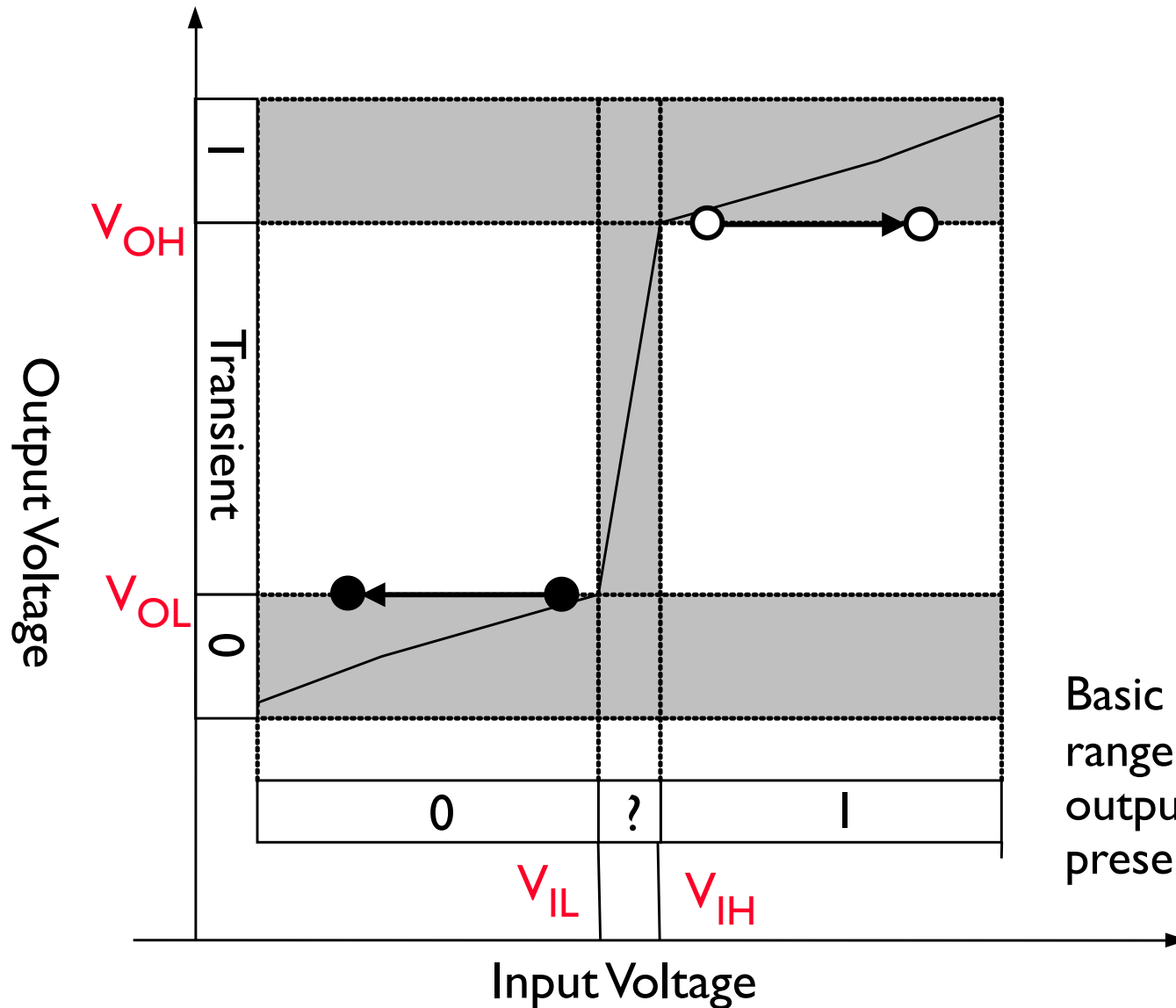
V_{IH} : Minimum input voltage recognized as logic 1

V_{OL} : Maximum output voltage corresponding to logic 0

V_{OH} : Minimum output voltage corresponding to logic 1

Basic idea: accept a much wider range of input than resulting output. Restore signal in presence of noise.

Digital Voltage Transfer Function



V_{IL} : Maximum input voltage recognized as logic 0

V_{IH} : Minimum input voltage recognized as logic 1

V_{OL} : Maximum output voltage corresponding to logic 0

V_{OH} : Minimum output voltage corresponding to logic 1

Basic idea: accept a much wider range of input than resulting output. Restore signal in presence of noise.

The Power of the Transfer Function

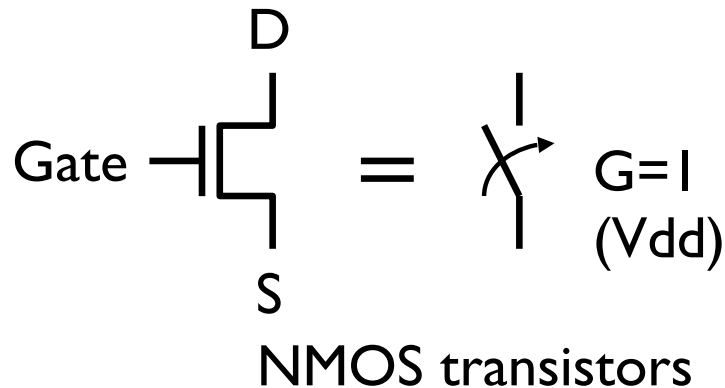
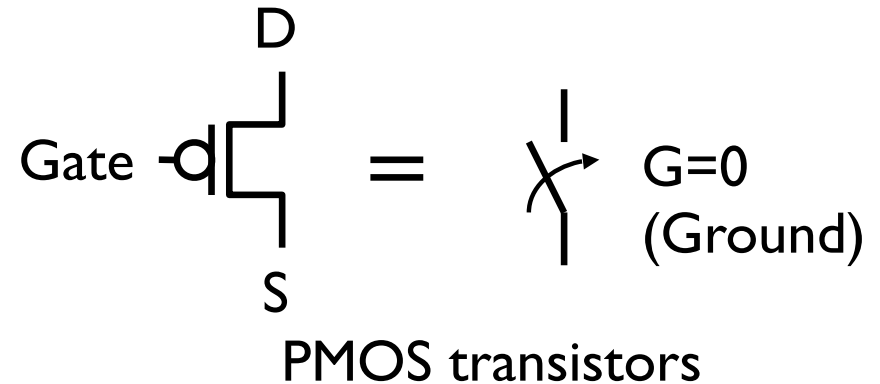
Each transistor restores the signal.

This allows us to build tremendously deep and complex digital circuits out of many transistors.

This is in contrast to analog circuits, which need to carefully manage how noise is introduced and propagated (the "black art" of analog design).

Transistors Make Gates

Simplest Model for MOS transistors



Let value 1 be Vdd (e.g., 1.8V, 2.5V), value 0 be ground

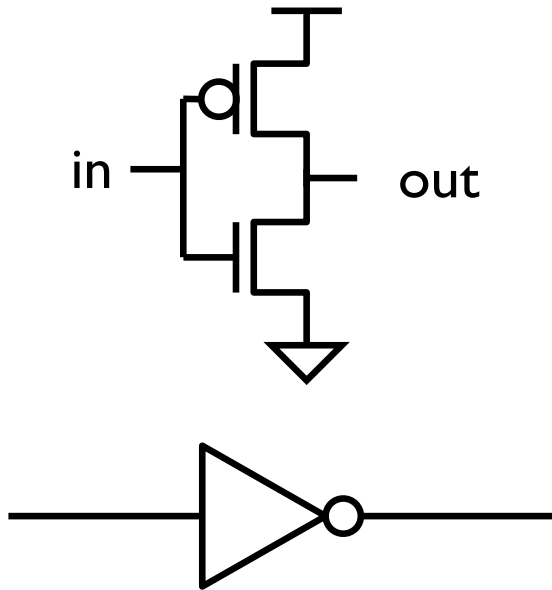
NMOS devices are switches

- Gate is 1 -> the drain D and source S **are** connected
- Gate is 0 -> the drain D and source S **are not** connected

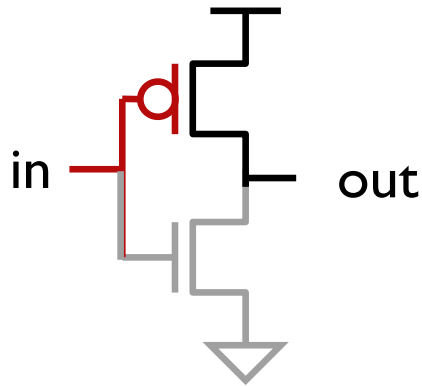
PMOS devices are switches

- Gate is 0 -> the drain D and source S **are** connected
- Gate is 1 -> the drain D and source S **are not** connected

Simple Case: Inverter

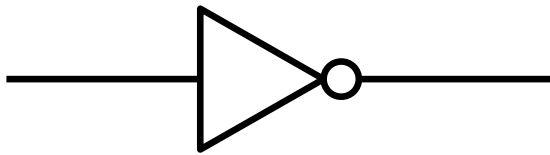


Simple Case: Inverter

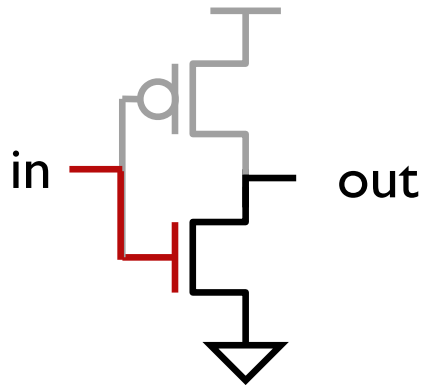


If the input signal is 0, PMOS connects output to V_{dd} (1).

If the input signal is 1, NMOS connects output to ground (0).

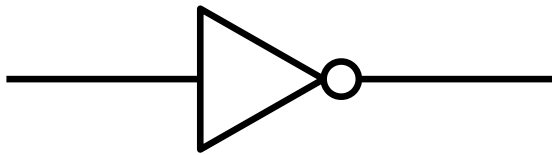


Simple Case: Inverter

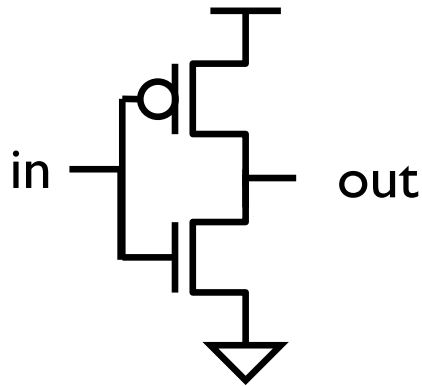


If the input signal is 0, PMOS connects output to V_{dd} (1).

If the input signal is 1, NMOS connects output to ground (0).

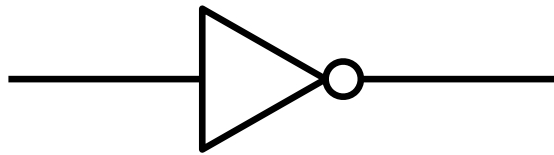


Simple Case: Inverter



If the input signal is 0, PMOS connects output to V_{DD} (1).

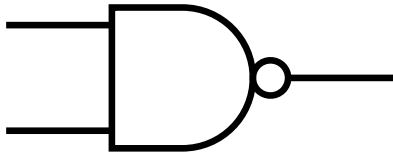
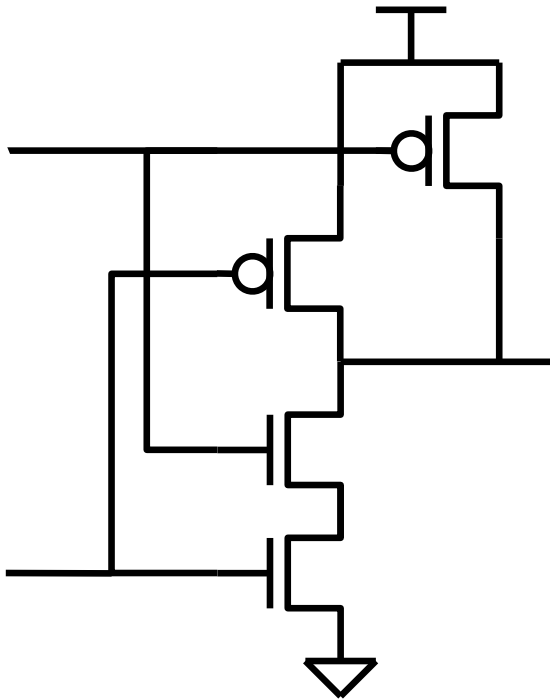
If the input signal is 1, NMOS connects output to ground (0).



Meets the rules

- Output is always driven (Either pMOS or nMOS is always active)
- V_{DD} and Gnd are never shorted (directly connected)
 - At least with valid inputs

NAND Gate (universal!)



A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

nMOS in *series* pulling to Gnd

- If *both* A and B are 1, connect out to ground

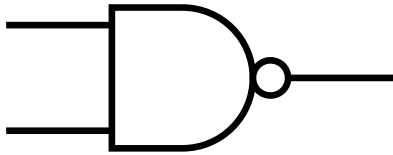
pMOS in *parallel* pulling to Vdd

- If *either* A or B is 0, connect out to Vdd

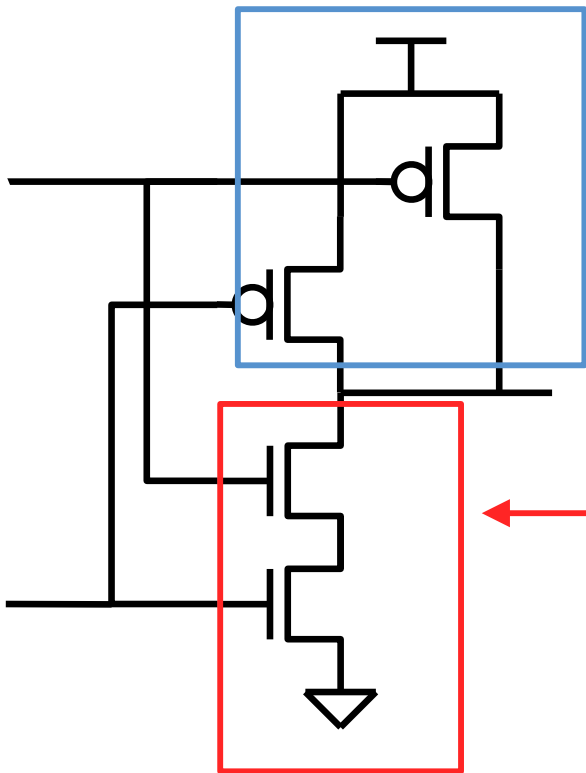
Out always driven by pMOS's or nMOS's but never both

Number of inputs (“fan-in”) can be increased

NAND Gate (universal!)



A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0



pMOS in *parallel* pulling to Vdd

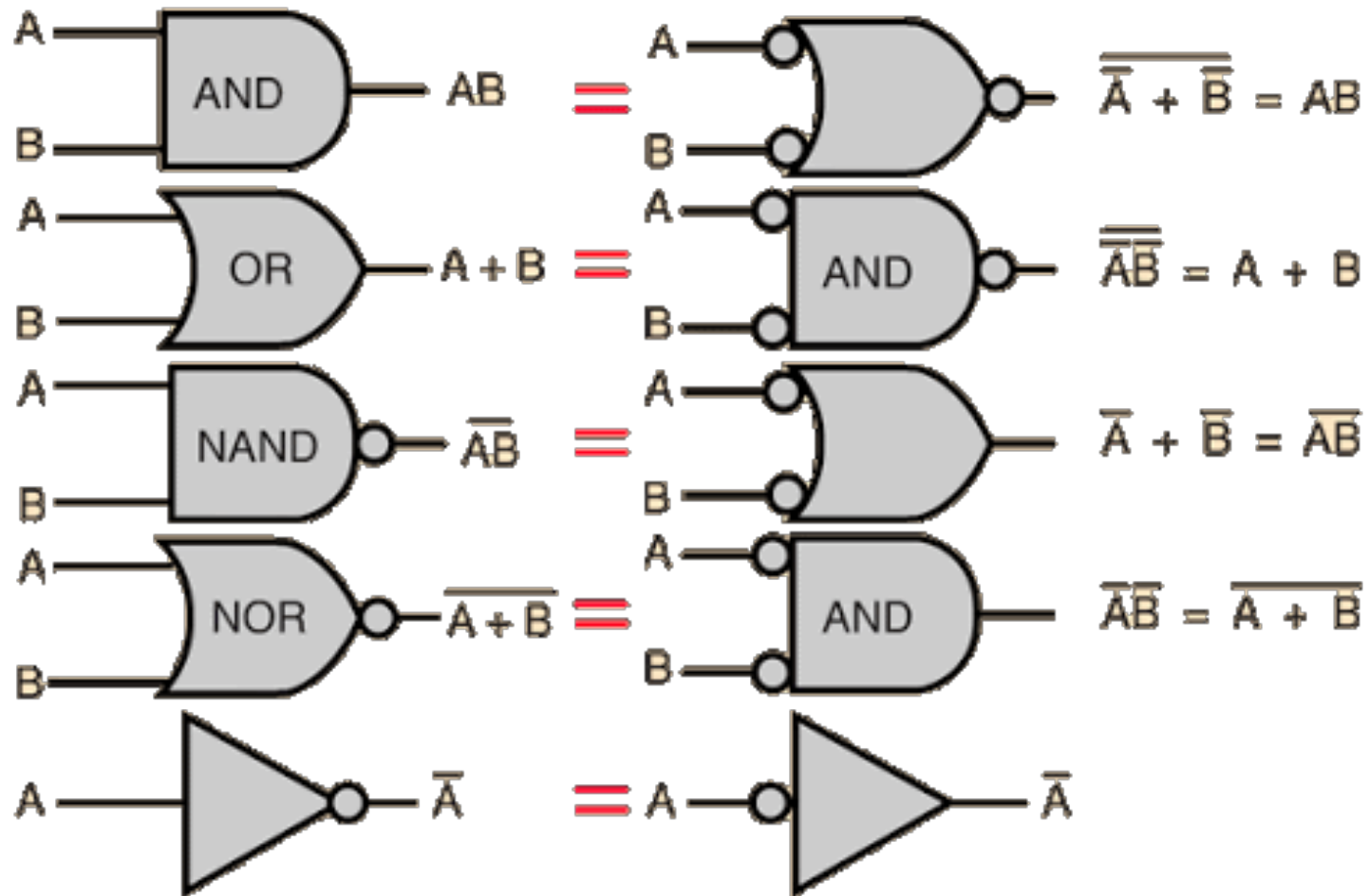
- If *either* A or B is 0, connect out to Vdd

nMOS in *series* pulling to Gnd

- If *both* A and B are 1, connect out to ground
- Out always driven by pMOS's or nMOS's but never both

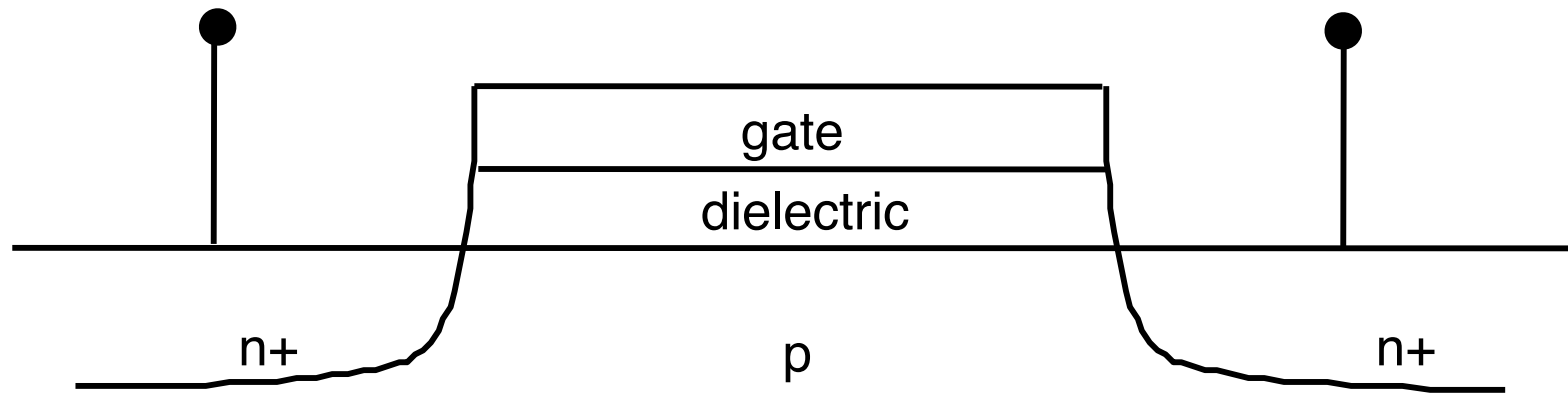
Number of inputs ("fan-in") can be increased

Gates



Transistor Performance

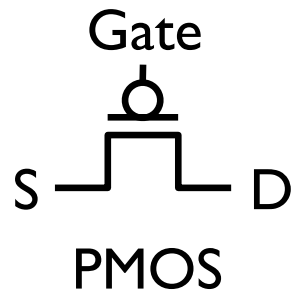
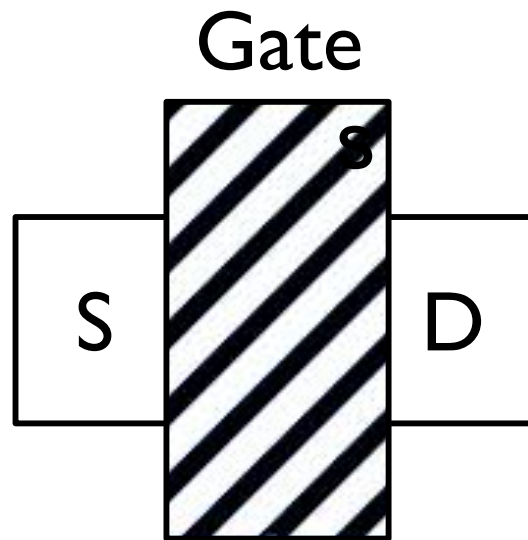
Transistors are Physical Devices



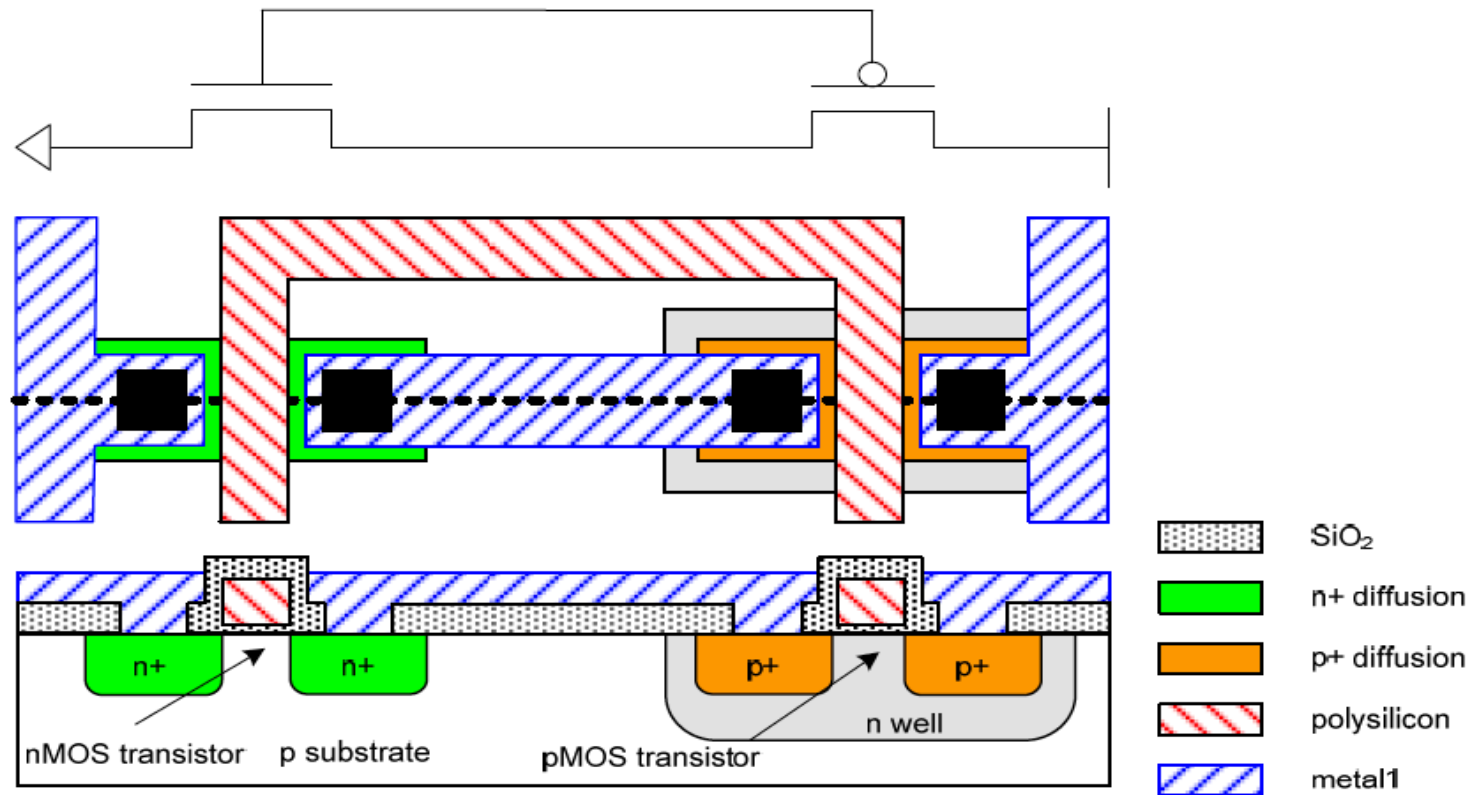
Gates used to be polysilicon (high purity poly-crystalline Si),
now metal gates with high-k dielectric (like HfO_2)

Advanced transistors are 3D (e.g., FinFETs) we'll just talk
about planar for simplicity.

Transistor Geometry



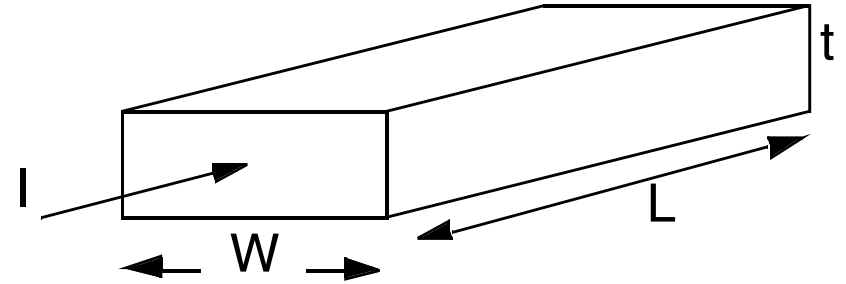
Example Layout of an Inverter



Resistance of a Conductor

Resistance of a conductor

- Resistivity ρ * Length/Area
- Designer does not control ρ , t
- Generally deal with ρ / t
- *Can* control W , L



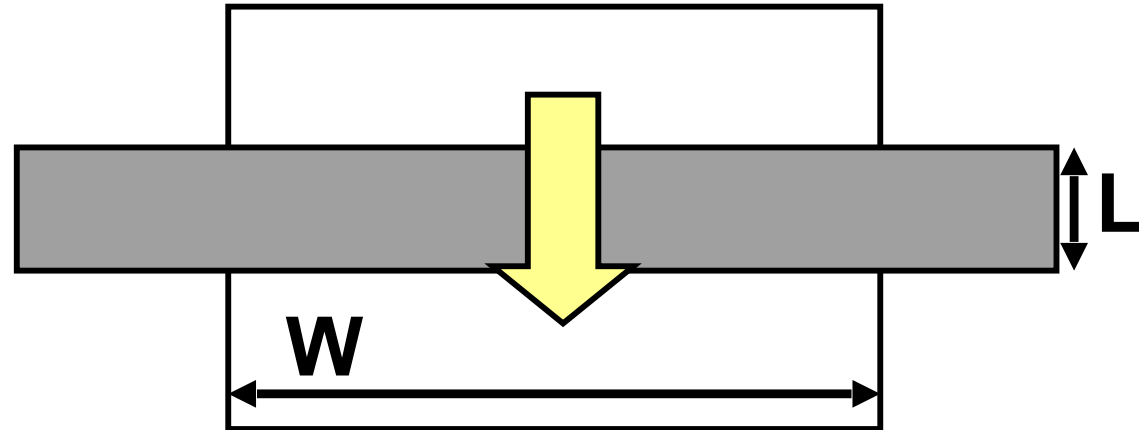
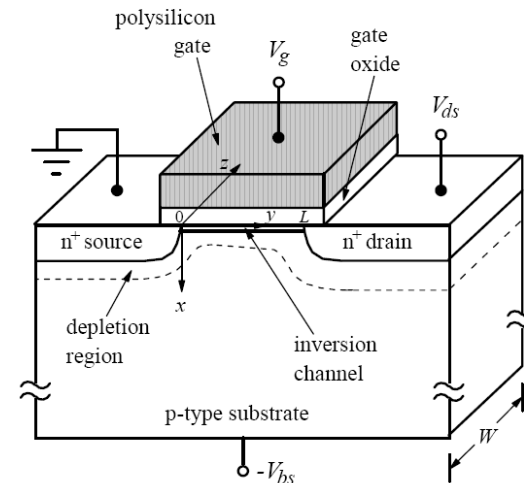
$$R = \frac{\rho L}{tW} = \frac{\rho}{t} \frac{L}{W}$$

That's why **LONG, NARROW**
(and **THIN**) wires have higher
resistances

Resistance

For transistors

- Designer chooses
 - W and L
- Wider transistor
 - Lower resistance
 - More current
 - *Longer to switch*



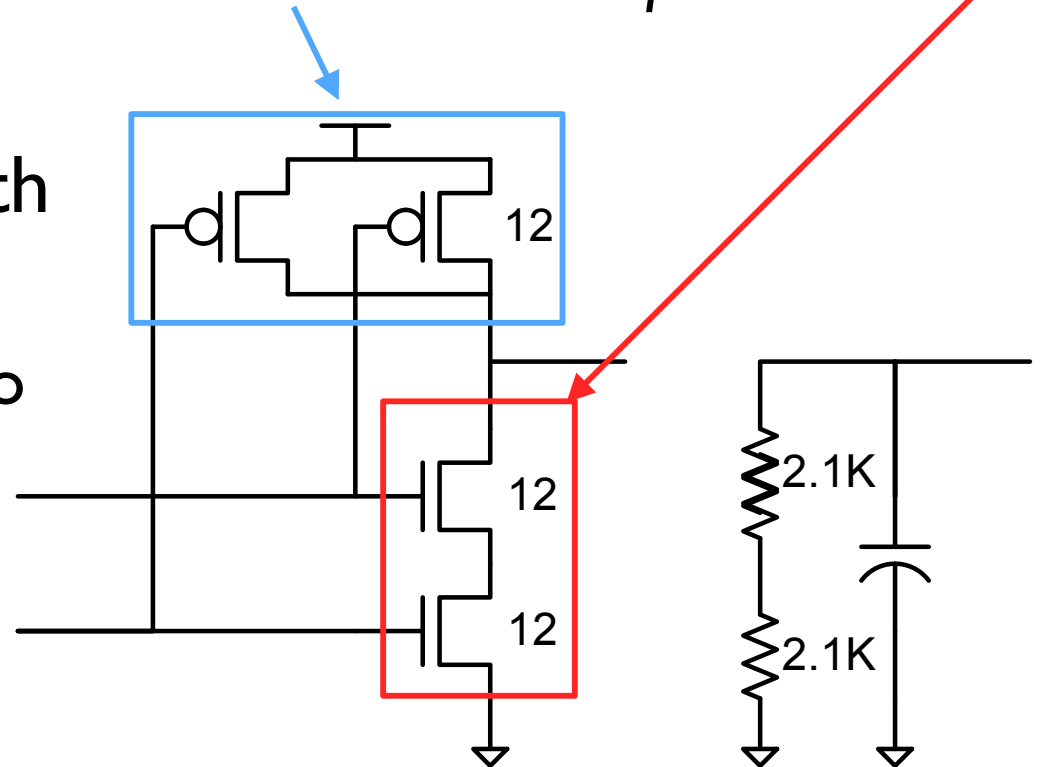
Let's Go Back to the NAND Gate

Pulldown (NMOS) transistors are in *series*

- 0 has the resistance of two NMOS transistors in *series*
- 1 has the resistance of two PMOS transistors in *parallel*

Need the resistance of both paths to be equal!

- NMOS transistors need to be wider than PMOS



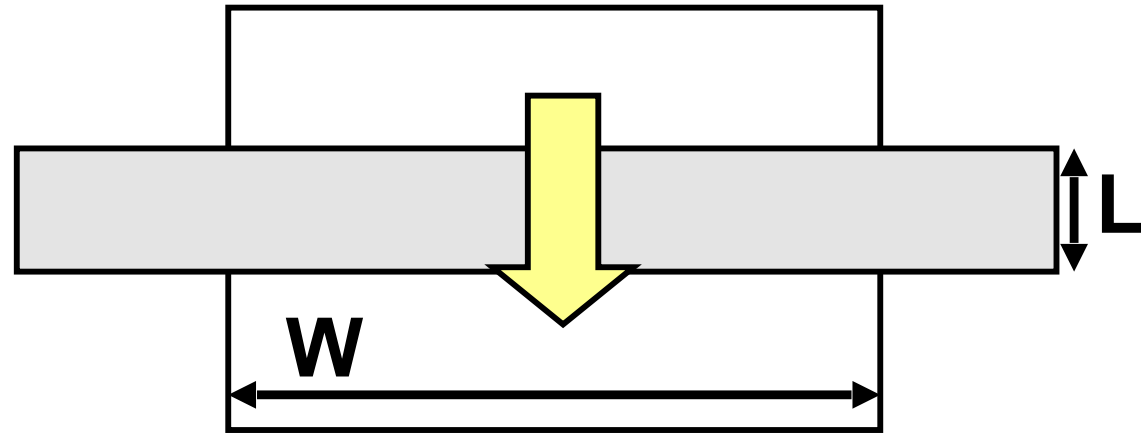
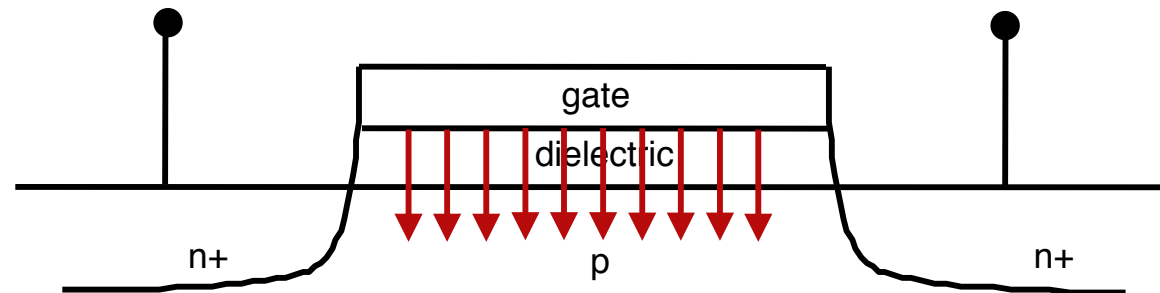
Wider Gates are Slower

A transistor is a capacitor

Switching the transistor requires building an electric field: takes time

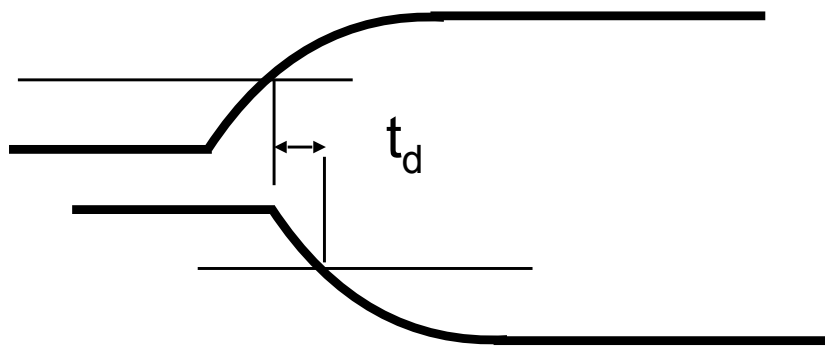
A wider gate requires more charge to build the field and takes longer to switch

High fan-in gates are slower



Circuit Timing

Delay and Rise/Fall Time



Delay, t_d , is measured from 50% point to 50% point.

Curve depends on resistance of circuit to transistor gate and capacitance of transistor gate.

- Lower resistance lowers t_d
- Smaller capacitance lowers t_d
- Tradeoff: smaller capacitance has higher resistance for next stage

Let's Go Back to the NAND Gate

2 input NAND has 2 NMOS in series

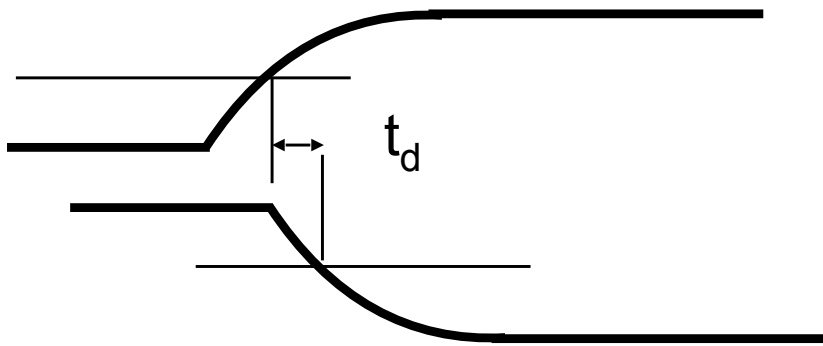
3 input NAND has 3 NMOS in series

- Need to have the same resistance
- 3 NMOS transistors of 3 input NAND must be wider
- 3 input NAND is slower

Result: build very large circuits out of smaller ones

- Build a 1024 input NAND built out of a tree of 4 input NANDs: logarithmic, not linear, growth of delay.

Delay and Rise/Fall Time



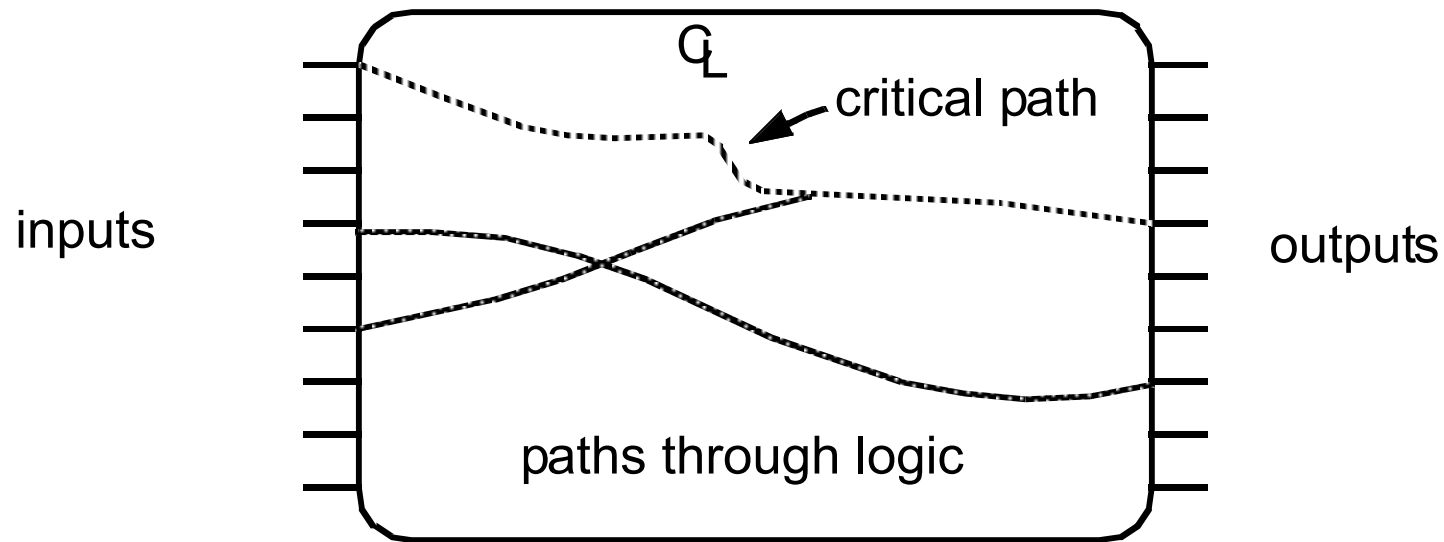
Delay, t_d , is measured from 50% point to 50% point.

Curve depends on resistance of circuit to transistor gate and capacitance of transistor gate.

- Lower resistance lowers t_d
- Smaller capacitance lowers t_d
- Tradeoff: smaller capacitance has higher resistance for next stage

Critical Paths

There are many signal paths through logic

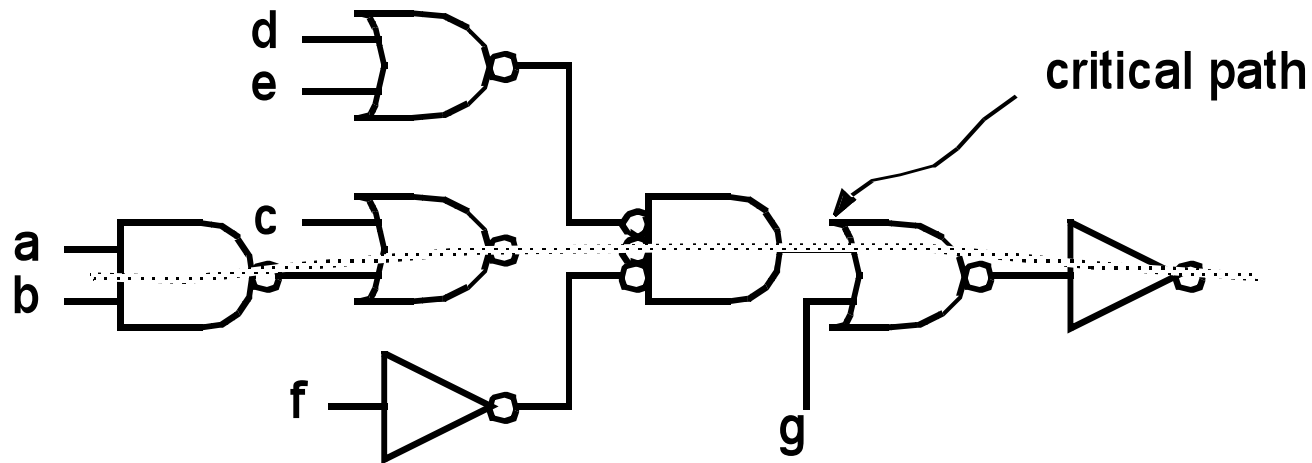


Not all the paths have the same delay

- Path from input to latest output is called the critical path
- It is this path that specifies the maximum rate the circuit can generate results (inverse of path delay)

Critical Path Example

This circuit computes the function $(\neg(a \wedge b) \vee c) \wedge (d \vee e) \wedge f \vee g$



The *critical path* traverses a 2-NAND, 2 2-NORs, a 3-NAND, and an inverter. The delay of this path determines the maximum delay of a change in the inputs, e.g, $ab\bar{c}def\bar{g}$ becomes $\bar{a}b\bar{c}def\bar{g}$.

Clock Speed

Sequential Logic

So far, we have only looked at combinational circuits:
input and outputs, no state

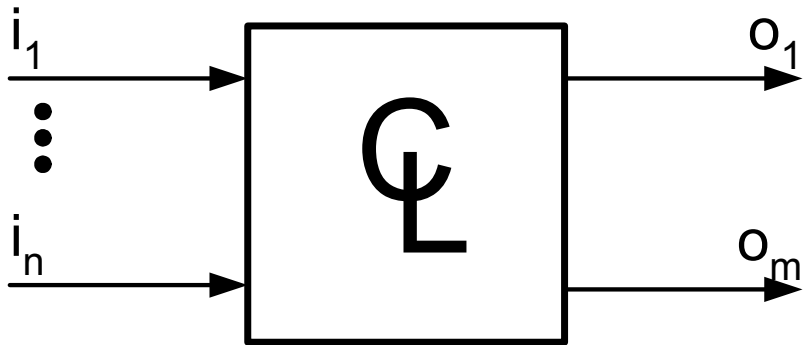
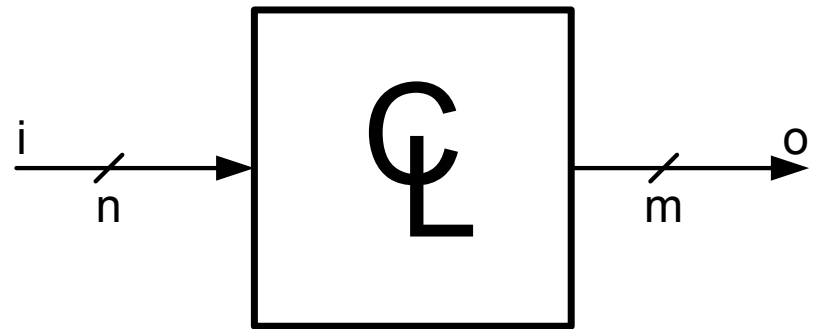
Digital circuits keep state

- Configuration
- Registers (data)

State is stored in flip-flops

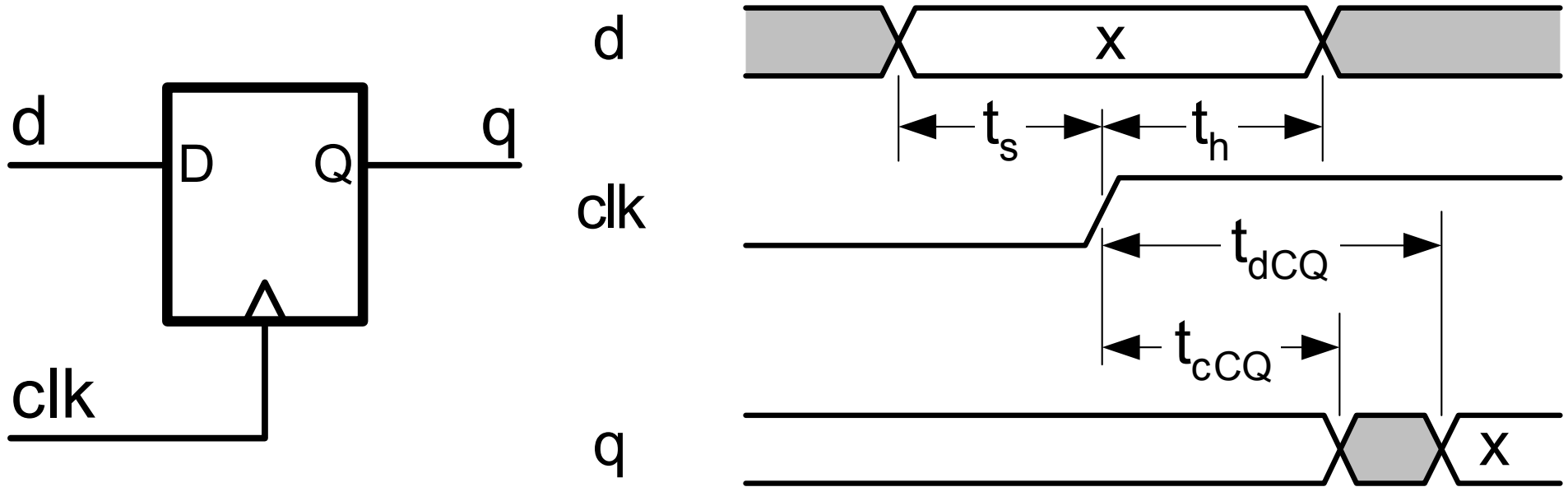
- Keep a steady output until clock ticks

Combinational logic is memoryless



$$o = f(i)$$

Our friend the D flip-flop



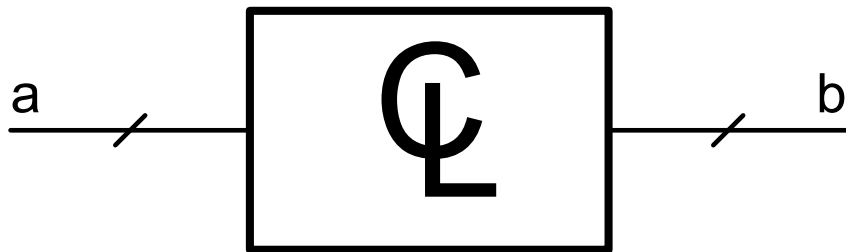
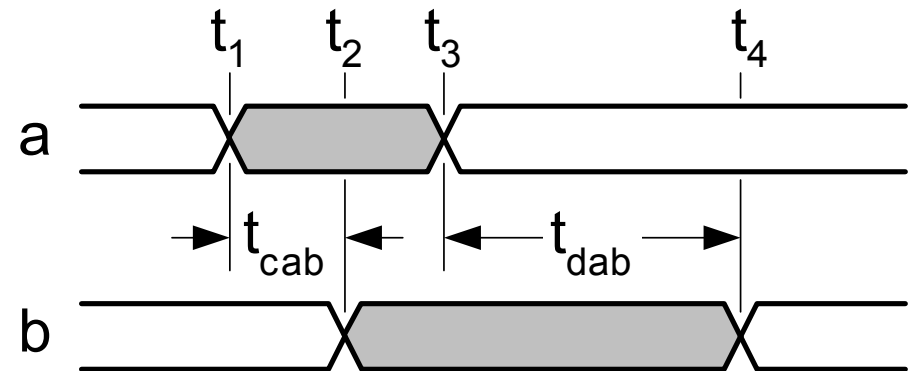
State is stored in flip-flops

- Keep a steady output until clock ticks
- Output changes to input on clock tick

Propagation Delay and Contamination Delay

Propagation Delay – Time from last input change until last output change. (Input at steady state to output at steady state.)

Contamination Delay – Time from first input change until first output change. (Input contaminated to output contaminated)



Making Circuits Correct

Our clock speed can't be faster than the longest propagation delay of our system

- Otherwise, output might not be stable when we clock the flip-flop and store the results
- Faster transistors have higher clock speeds
- Shallower circuits have higher clock speeds

Contamination delay can't be faster than the time it takes for the flip-flop to store the state

- Otherwise, the state changes on the flip-flop before it's able to store it.

The Essence of Hardware Performance

Software likes to think of hardware as just 1s and 0s

Hardware is built out of physical objects that rely on storing charge to build electrical fields to conduct.

Smaller transistors (narrower gates) are faster; shallower circuits are faster.

These physical properties govern performance and so influence *how you design a high-performance system*.