Welcome to 6.111!

- Introductions, course mechanics
- Course overview
- Digital signaling
- Combinational logic
- 4 Handouts: calendar, slides, LP #1, info form

Introductions

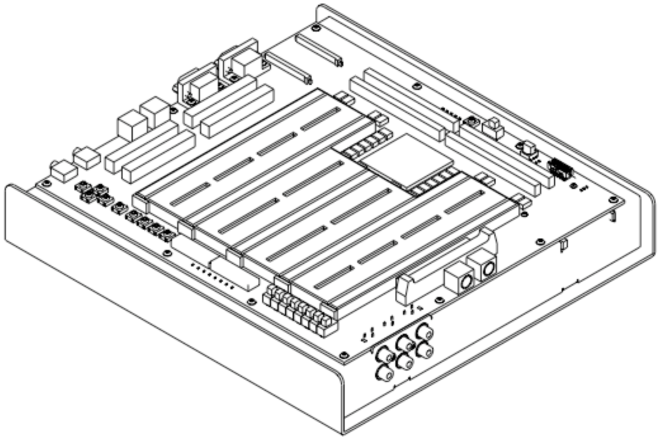
Chris Terman
Lectures

Ben Gelb
TA

Alex Valys
TA

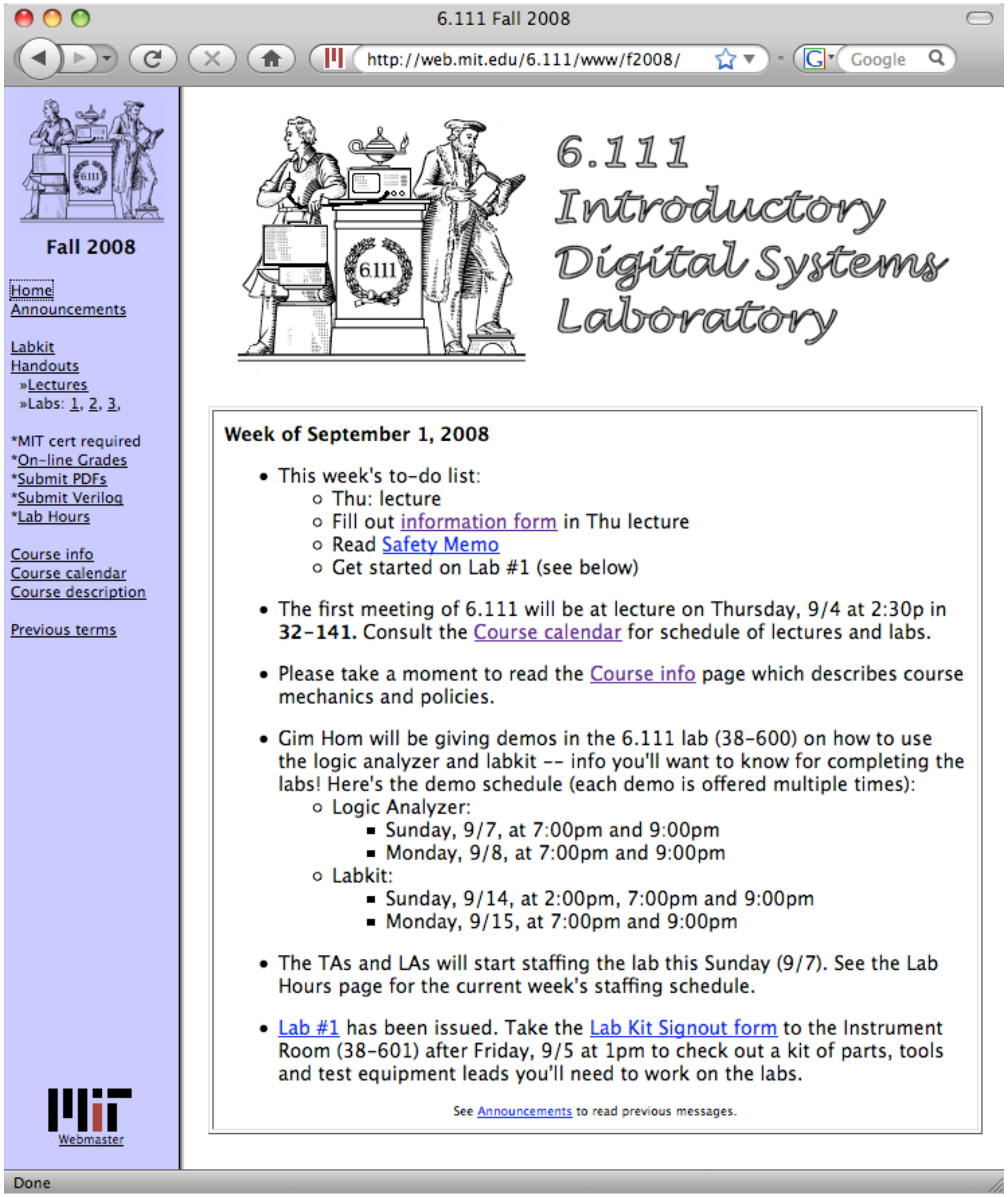
Gim Hom
Lab Guru

CI-M staff:
Don Unger
Mary Caulfield



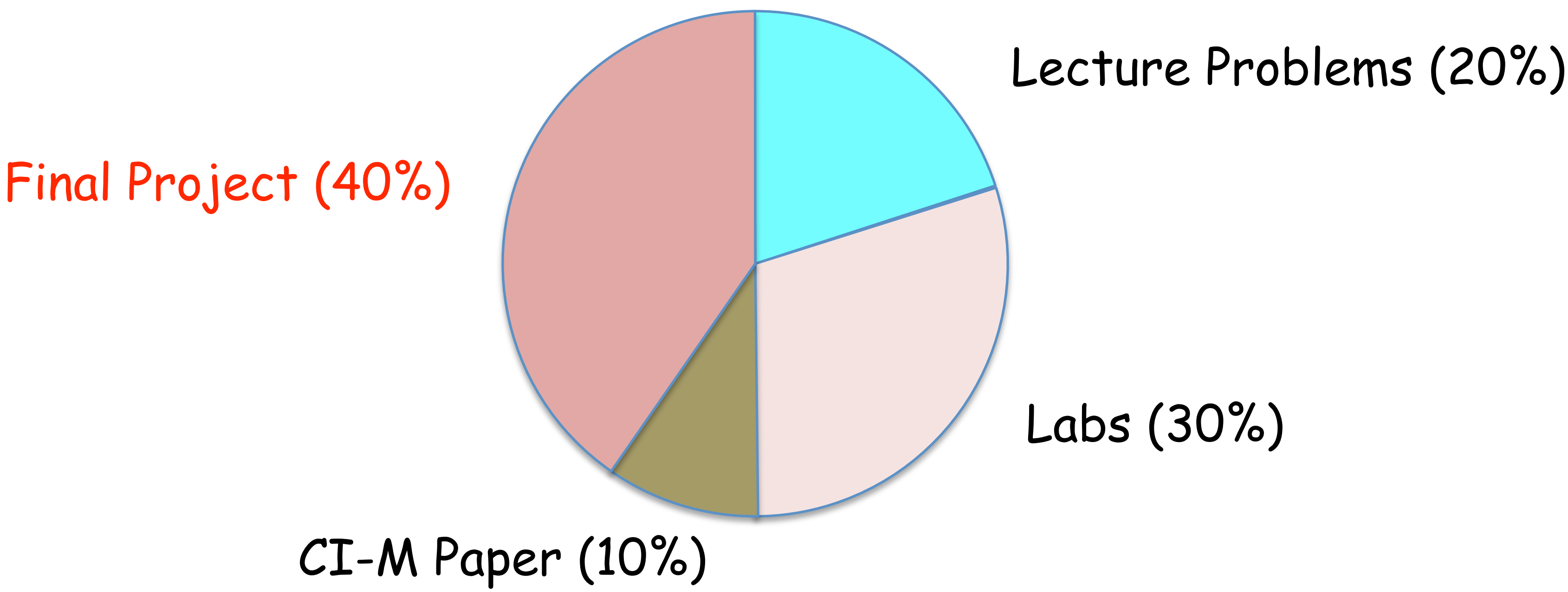
6.111 Labkit
6M-gate FPGA + audio + video + memories + …

Course Website: http://web.mit.edu/6.111/www/f2008



- One stop shopping for all 6.111 information
  - Handouts
  - Labs
  - On-line submissions
  - Policies
  - …
- Please read "Course info" to get oriented

Assignments



A large number of students do "A" level work and are, indeed, rewarded with a grade of "A". The corollary to this is that, since average performance levels are so high, punting any part of the subject can lead to a disappointing grade.

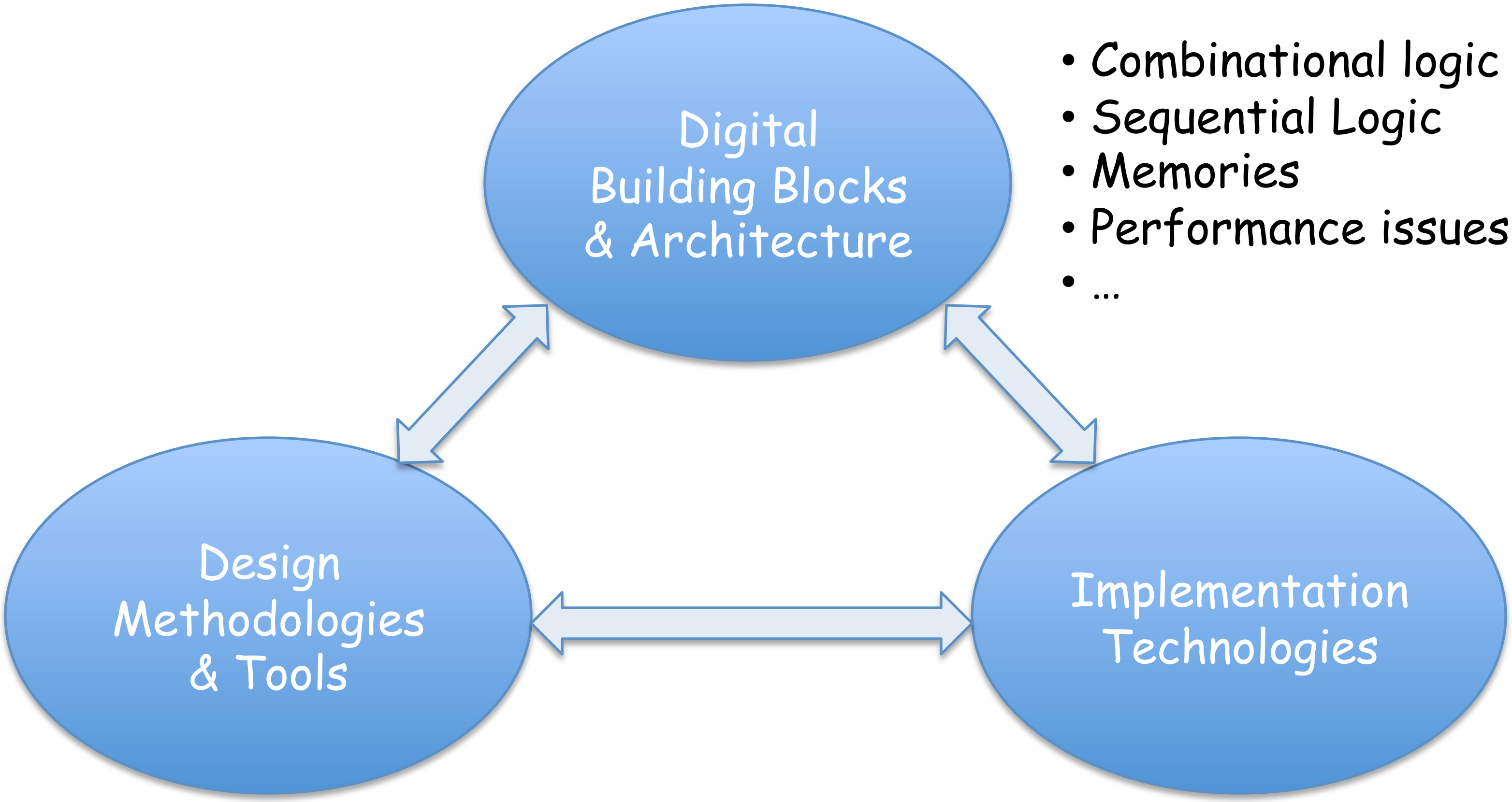
Labs: learning the ropes

- Lab 1
  - Experiment with gates, design & implement some logic
  - Learn about lab equipment in the Digital Lab (38-600): oscilloscopes and logic analyzers
- Lab 2
  - Introduction to Verilog & the labkit
- Lab 3
  - Design and implement a Finite State Machine (FSM)
  - Use Verilog to program an FPGA
  - Report and its revision will be evaluated for CI-M
- Lab 4
  - Design a complicated system with multiple FSMs (Major/Minor FSM)
  - Voice recorder using AC97 codec and SRAMs
- Lab 5
  - Video circuits: a simple Pong game

Final Project

- Done in groups of two (or sometimes three)
- Open-ended
- You and the staff negotiate a project proposal
  - Must emphasize digital concepts, but inclusion of analog interfaces (e.g., data converters, sensors or motors) common and often desirable
  - Proposal Conference, several Design Reviews
- Design presentation to staff
- Staff will provide help with project definition and scope, design, debugging, and testing
- It is extremely difficult for a student to receive an A without completing the final project. Sorry, but we don't give incompletes.

6.111 Topics

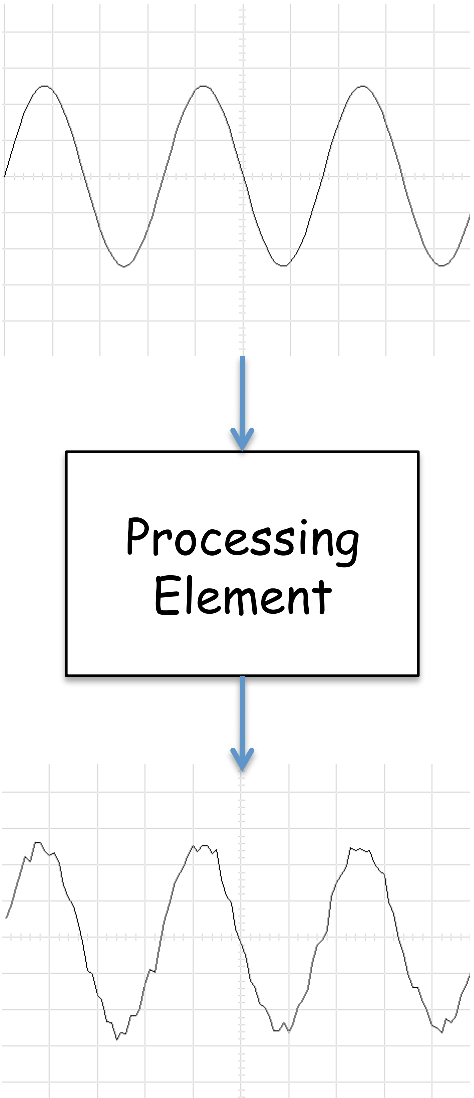


- Combinational logic
- Sequential Logic
- Memories
- Performance issues
- …

- Design metrics
- HDL: Verilog
- Simulation tools
- Synthesis, Place & Route
- …

- FPGAs
- Flash, ZBT ram
- AC97, TripleDAC
- …

The trouble with analog signaling



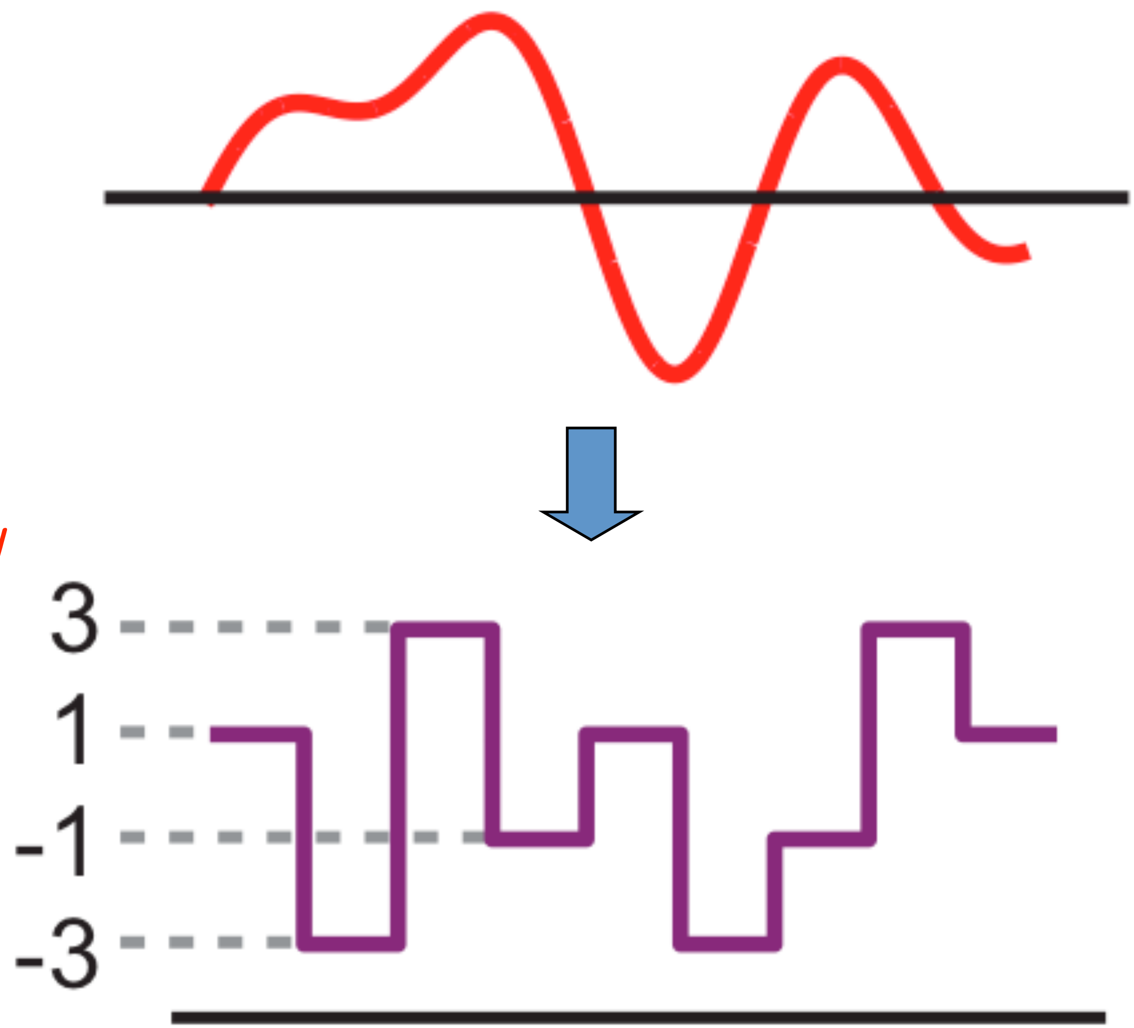
The real world is full of continuous-time continuous-value (aka "analog") signals created by physical processes: sound vibrations, light fields, voltages and currents, phase and amplitudes, …

But if we build processing elements to manipulate these signals we must use non-ideal components in real-world environments, so some amount of error (aka "noise") is introduced. The error comes from component tolerances, electrical phenomenon (e.g., IR and LdI/dt effects), transmission losses, thermal noise, etc. Facts of life that can't be avoided…

And the more analog processing we do, the worse it gets: signaling errors accumulate in analog systems since we can't tell from looking at signal which wiggles were there to begin with and which got added during processing.

Solution: go digital!

Continuous values
Continuous time



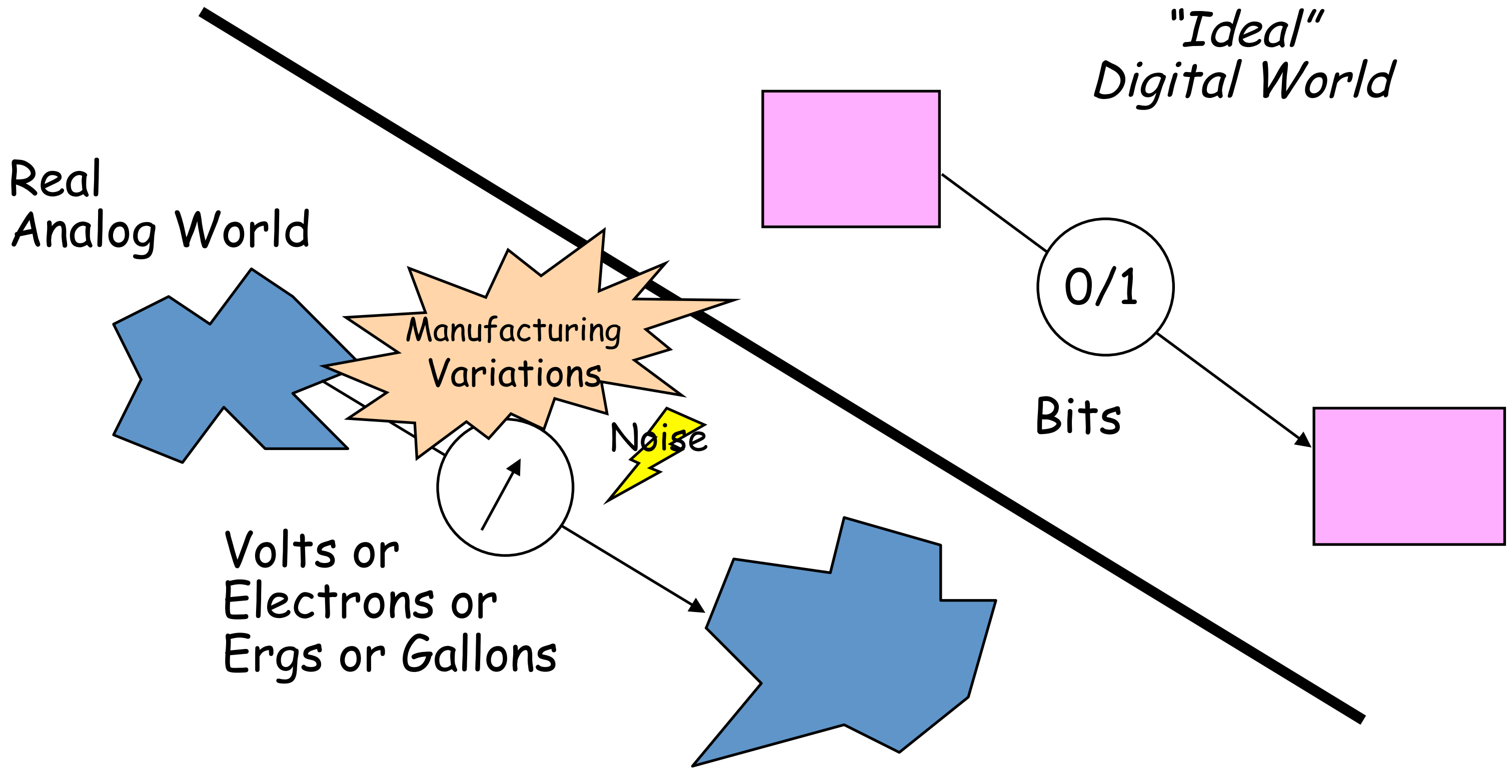
So we can detect small changes and restore original values

Discrete values
Discrete time

So we don't look while it's changing

The Digital Abstraction

Noise and inaccuracy are inevitable; we can't reliably engineer perfect components – we must design our system to tolerate some amount of error if it is to process information reliably.

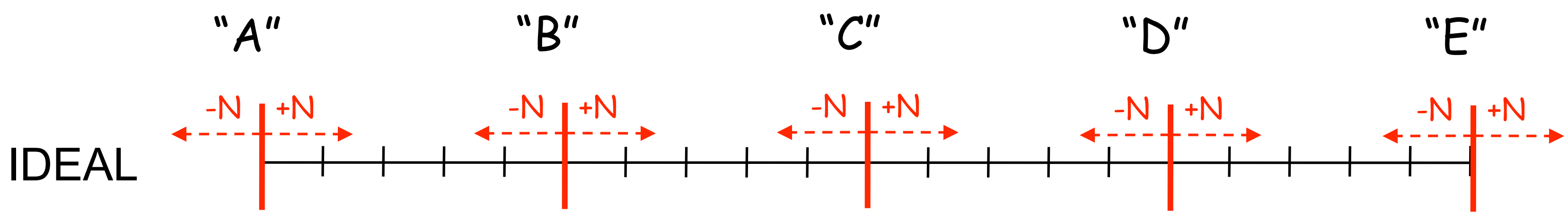


Keep in mind that the world is not digital, we would simply like to engineer it to behave that way. Furthermore, we must use real physical phenomena to implement digital designs!

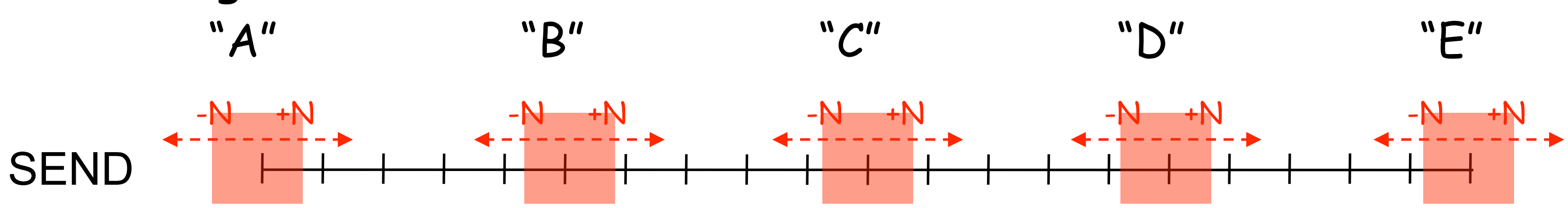
Digital Signaling: sending

To ensure we can distinguish signal from noise, we'll encode information using a fixed set of discrete values called symbols.

Given a bound N on the size of possible errors, if the analog representations for the symbols are chosen to be at least 2N apart, we should be able to detect and eliminate errors of up to ±N.

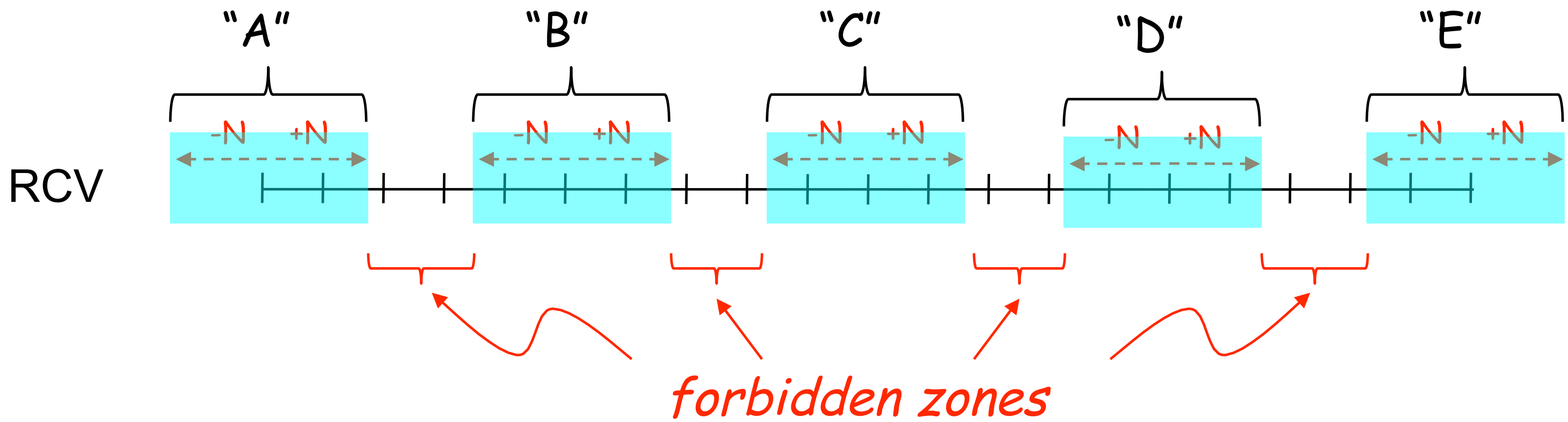


Since we will use non-ideal components in the sender, we allow each transmitted symbol to be represented by a (small) range of analog values.



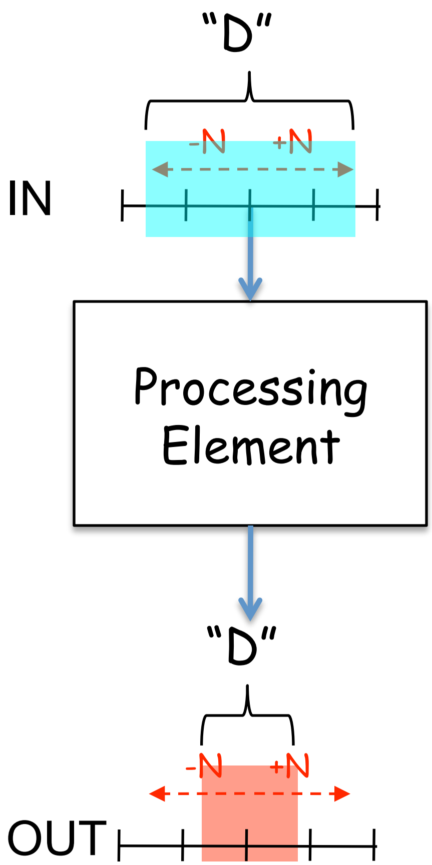
Digital Signaling: receiving

Since the channel/wire is imperfect and we will use non-ideal components in the receiver, we require the receiver to accept a (larger) range of analog values for each symbol.



To avoid hard-to-make decisions at the boundaries between symbol representations, insert a "forbidden zone" between symbols so that some ranges of received values are not required to be mapped to a specific symbol.

Digital processing elements

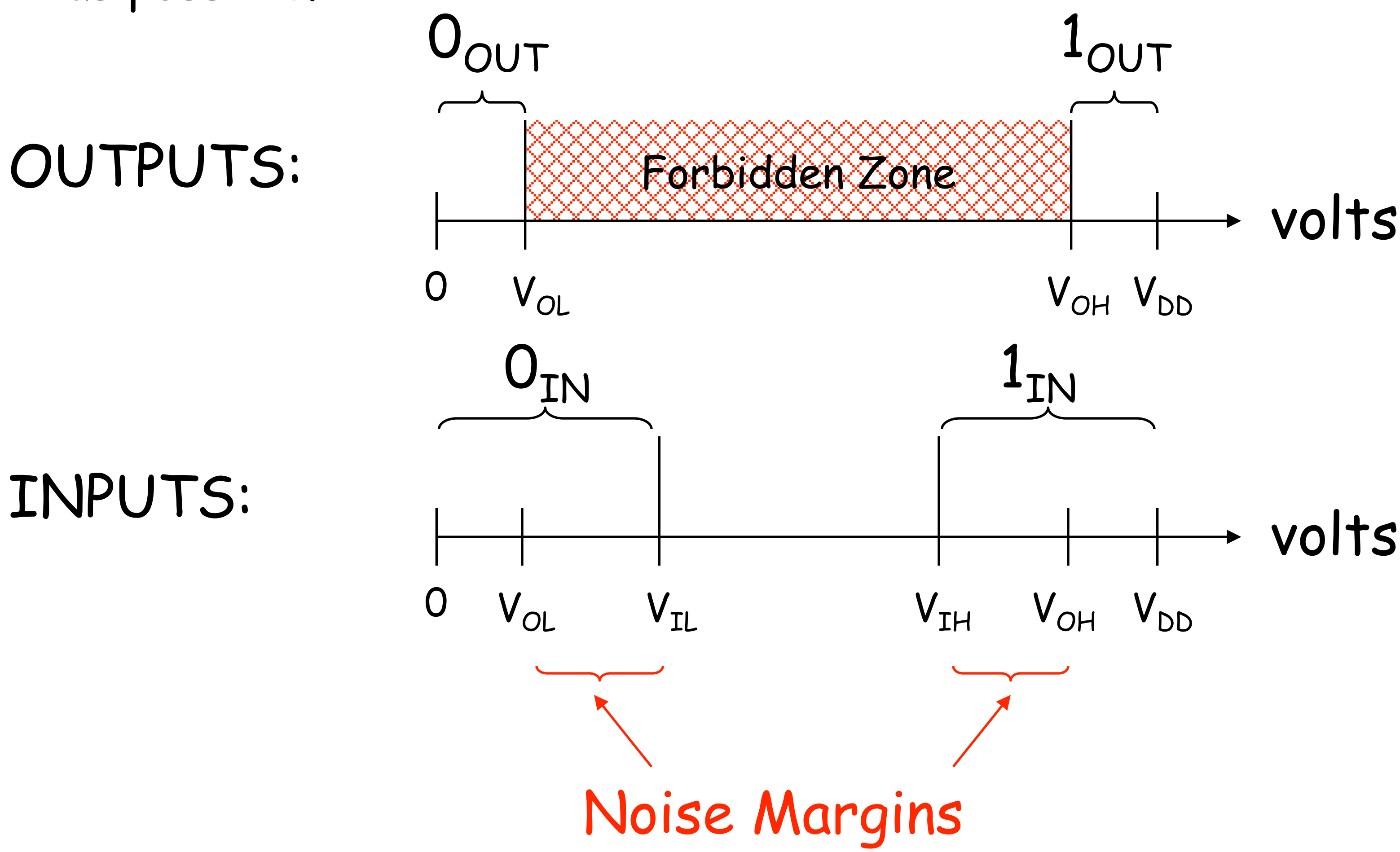


Digital processing elements *restore* noisy input values to legal output values – signaling errors don't accumulate in digital systems. So the number of processing elements isn't limited by noise problems!

The "trick" is that we've defined our signaling convention so that we can tell from looking at a signal which wiggles were there to begin with and which got added during processing.

Using voltages to encode binary values

We'll keep things simple by designing our processing elements to use voltages to encode binary values (0 or 1). To ensure robust operation we'd like to make the noise margins as large as possible.



Digital Signaling Specification

Digital input: $V_{IN} < V_{IL}$ or $V_{IN} > V_{IH}$

Digital output: $V_{OUT} < V_{OL}$ or $V_{OUT} > V_{OH}$

Noise margins: $V_{IL} - V_{OL}$ and $V_{OH} - V_{IH}$

Where V_{OL}, V_{IL}, V_{IH} and V_{OH} are part of the specification for a particular family of digital components.

Now that we have a way of encoding information as a signal, we can define what it means to be *digital device.*

Sample DC (signaling) Specification

I/O Standard	V_{IL}		V_{IH}		V_{OL}	V_{OH}	I_{OL}	I_{OH}
	V, Min	V, Max	V, Min	V, Max	V, Max	V, Min	mA	mA
LVTTL	–0.3	0.8	2.0	3.45	0.4	2.4	Note(3)	Note(3)
LVCMOS33, LVDCI33	–0.3	0.8	2.0	3.45	0.4	$V_{CCO} - 0.4$	Note(3)	Note(3)
LVCMOS25, LVDCI25	–0.3	0.7	1.7	$V_{CCO} + 0.3$	0.4	$V_{CCO} - 0.4$	Note(3)	Note(3)
LVCMOS18, LVDCI18	–0.3	35% V_{CCO}	65% V_{CCO}	$V_{CCO} + 0.3$	0.45	$V_{CCO} - 0.45$	Note(4)	Note(4)
LVCMOS15, LVDCI15	–0.3	35% V_{CCO}	65% V_{CCO}	$V_{CCO} + 0.3$	25% V_{CCO}	75% V_{CCO}	Note(4)	Note(4)
LVCMOS12	–0.3	35% V_{CCO}	65% V_{CCO}	$V_{CCO} + 0.3$	25% V_{CCO}	75% V_{CCO}	Note(6)	Note(6)
PCI33_3(5)	–0.2	30% V_{CCO}	50% V_{CCO}	V_{CCO}	10% V_{CCO}	90% V_{CCO}	Note(5)	Note(5)
PCI66_3(5)	–0.2	30% V_{CCO}	50% V_{CCO}	V_{CCO}	10% V_{CCO}	90% V_{CCO}	Note(5)	Note(5)
PCI-X(5)	–0.2	35% V_{CCO}	50% V_{CCO}	V_{CCO}	10% V_{CCO}	90% V_{CCO}	Note(5)	Note(5)

Source: Xilinx Virtex 5 Datasheet

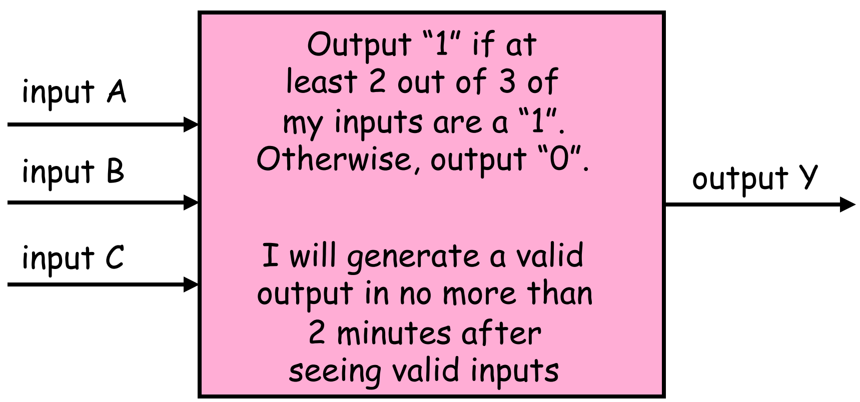
A Digital Processing Element

A *combinational device* is a processing element that has

Static discipline

- one or more digital *inputs*
- one or more digital *outputs*
- a *functional specification* that details the value of each output for every possible combination of valid input values
- a *timing specification* consisting (at minimum) of an upper bound t_{pd} on the required time for the device to compute the specified output values from an arbitrary set of stable, valid input values

One of two discrete values

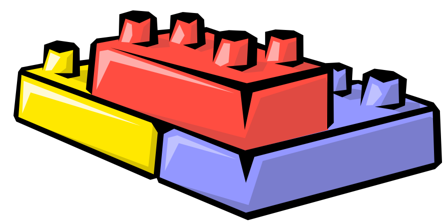


Why have processing blocks?

- The goal of modular design:

ABSTRACTION

- What does that mean anyway:
  - Rules simple enough for a 6-3 to follow…
  - Understanding BEHAVIOR without knowing IMPLEMENTATION
  - Predictable composition of functions
  - Tinker-toy assembly
  - Guaranteed behavior under REAL WORLD circumstances

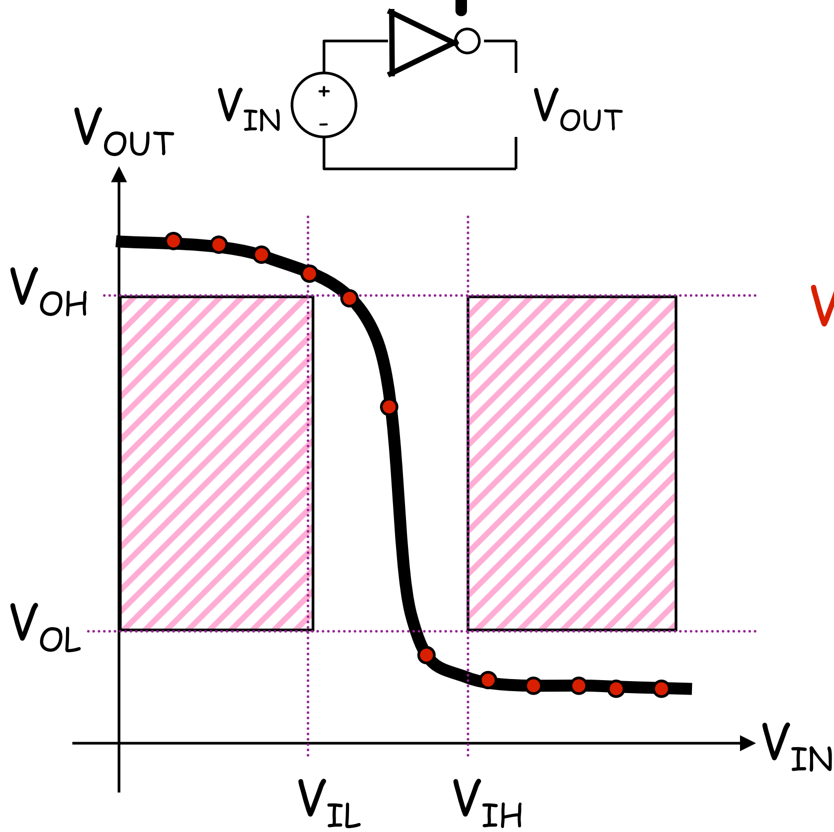


A Combinational Digital System

- A set of interconnected elements is a *combinational device* if
  – each circuit element is a combinational device
  – every input is connected to exactly one output or a constant (e.g., some vast supply of 0's and 1's)
  – the circuit contains no directed cycles

- Why is this true?
  – Given an acyclic circuit meeting the above constraints, we can derive functional and timing specs for the input/output behavior from the specs of its components!
  – We'll see lots of examples soon. But first, we need to build some combinational devices to work with…

Example Device: An Inverter



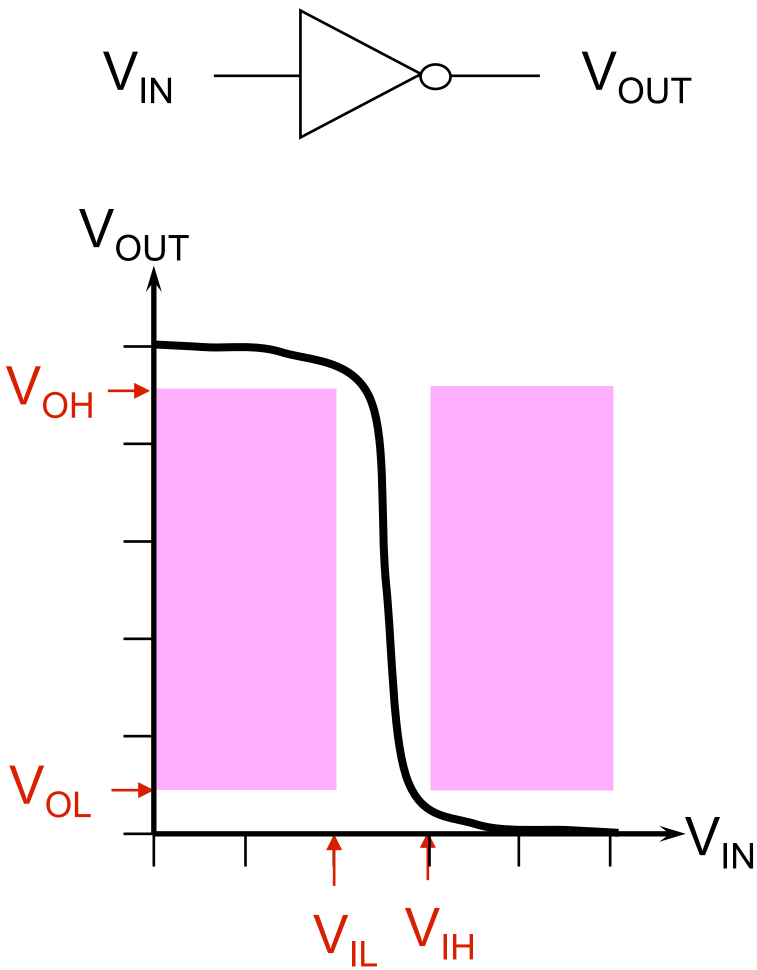
Voltage Transfer Characteristic:
Plot of V_{OUT} vs. V_{IN} where each measurement is taken after any transients have died out.

Note: VTC does not tell you anything about how fast a device is—it measures static behavior not dynamic behavior

Static Discipline requires that we avoid the shaded regions (aka "forbidden zones"), which correspond to *valid* inputs but *invalid* outputs. Net result: combinational devices must have GAIN > 1 and be NONLINEAR.

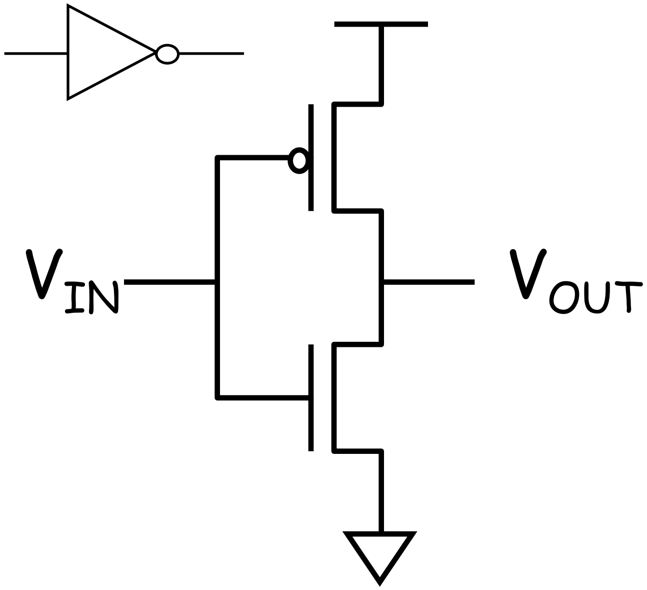
$$\frac{\partial V_{OUT}}{\partial V_{IN}}$$

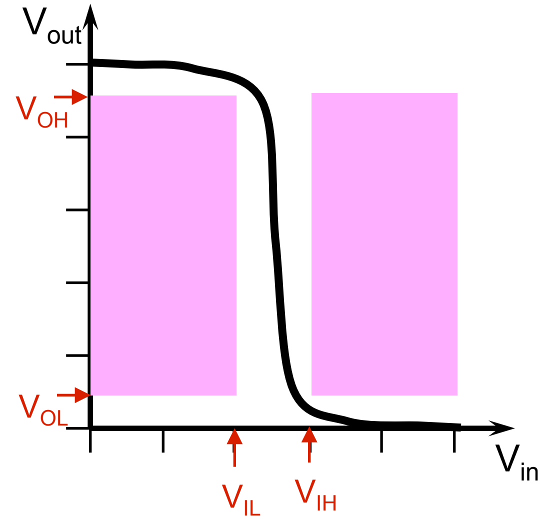
Combinational Device Wish List

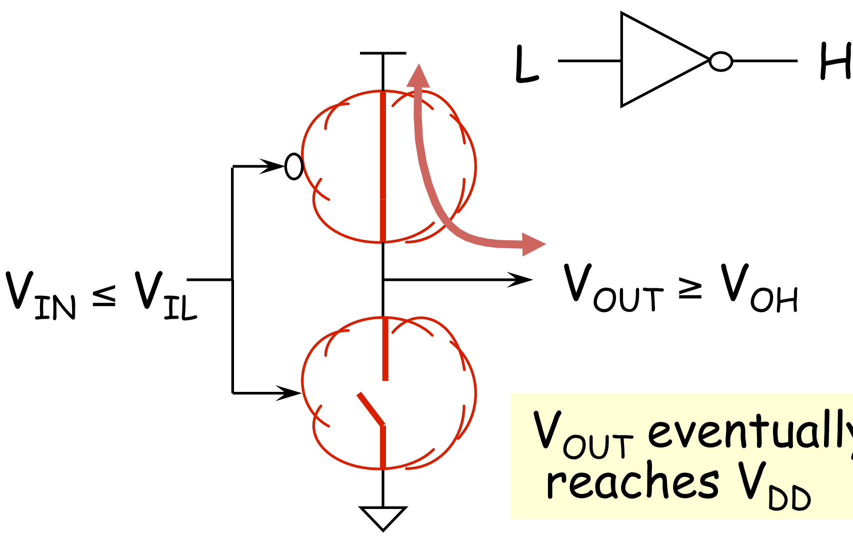


- ✓ Design our system to tolerate some amount of error
  - ⇒ Add positive noise margins
  - ⇒ VTC: gain>1 & nonlinearity
- ✓ Lots of gain ⇒ big noise margin
- ✓ Cheap, small
- ✓ Changing voltages will require us to dissipate power, but if no voltages are changing, we'd like zero power dissipation
- ✓ Want to build devices with useful functionality (what sort of operations do we want to perform?)

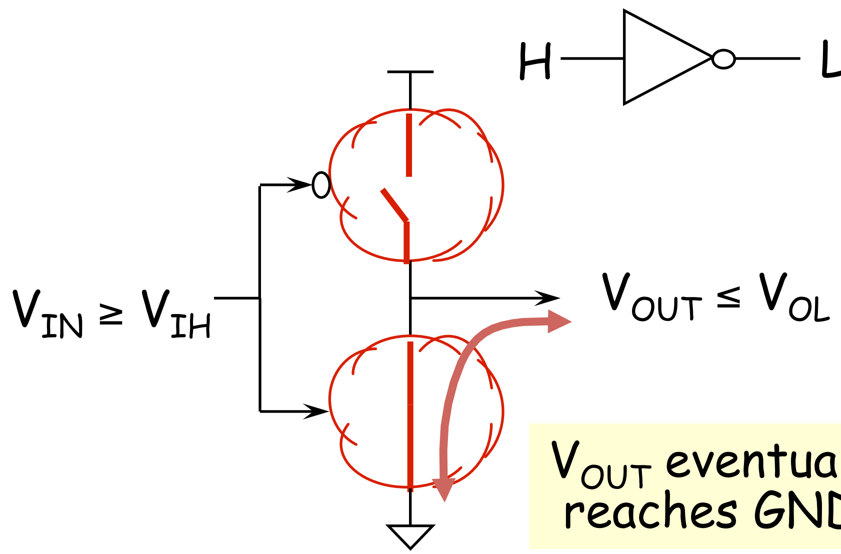
Wishes Granted: CMOS





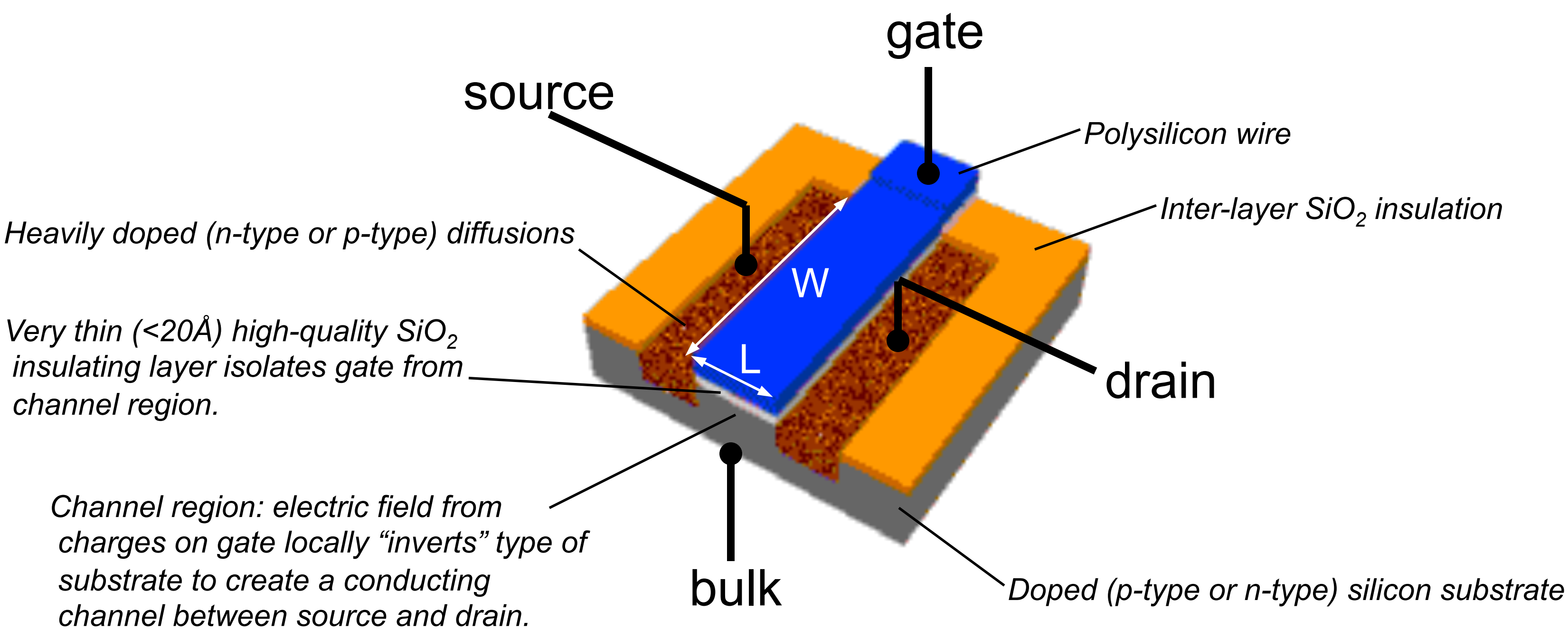


V_{OUT} eventually reaches V_{DD}



V_{OUT} eventually reaches GND

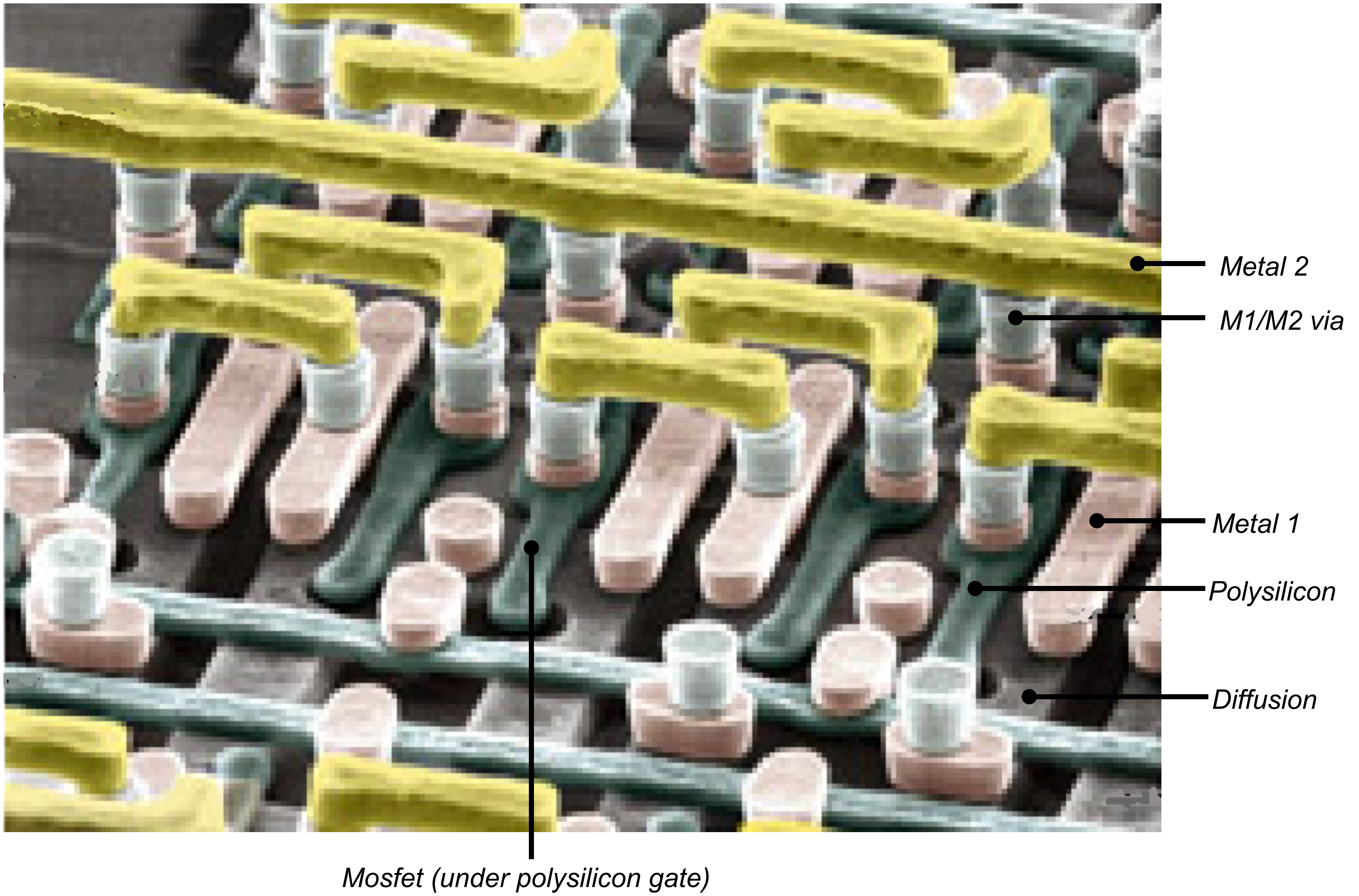
MOSFETS: Gain & Non-linearity



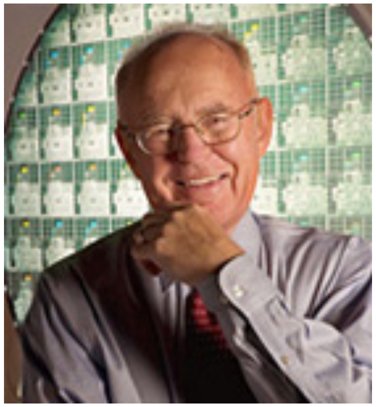
MOSFETs (metal-oxide-semiconductor field-effect transistors) are four-terminal voltage-controlled switches. Current flows between the diffusion terminals if the voltage on the gate terminal is large enough to create a conducting "channel", otherwise the mosfet is off and the diffusion terminals are not connected.

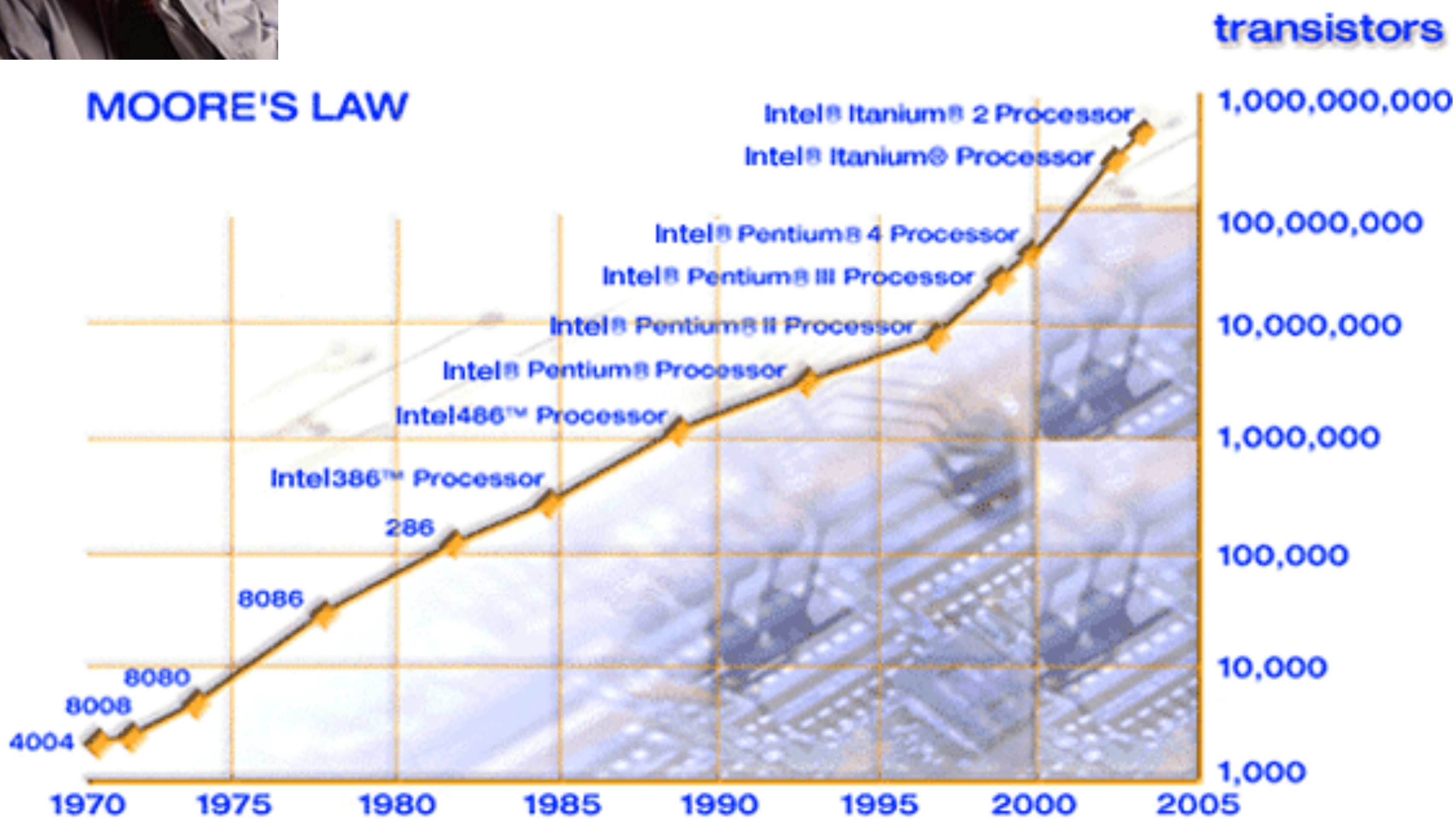
Digital Integrated Circuits

IBM photomicrograph (SiO_2 has been removed!)

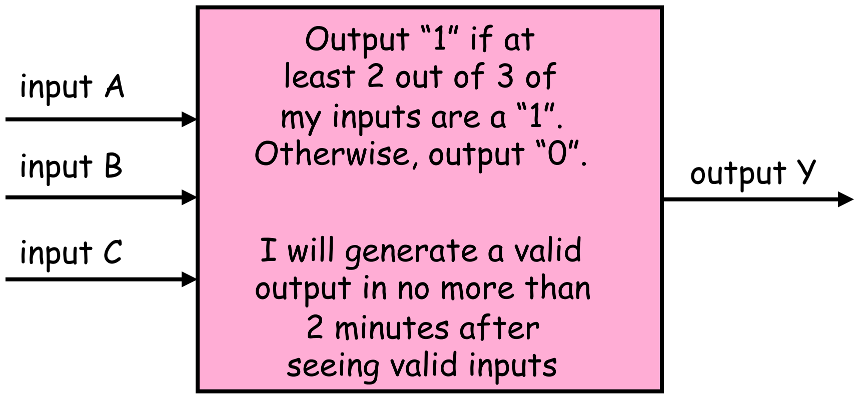


CMOS Forever!?





Functional Specifications



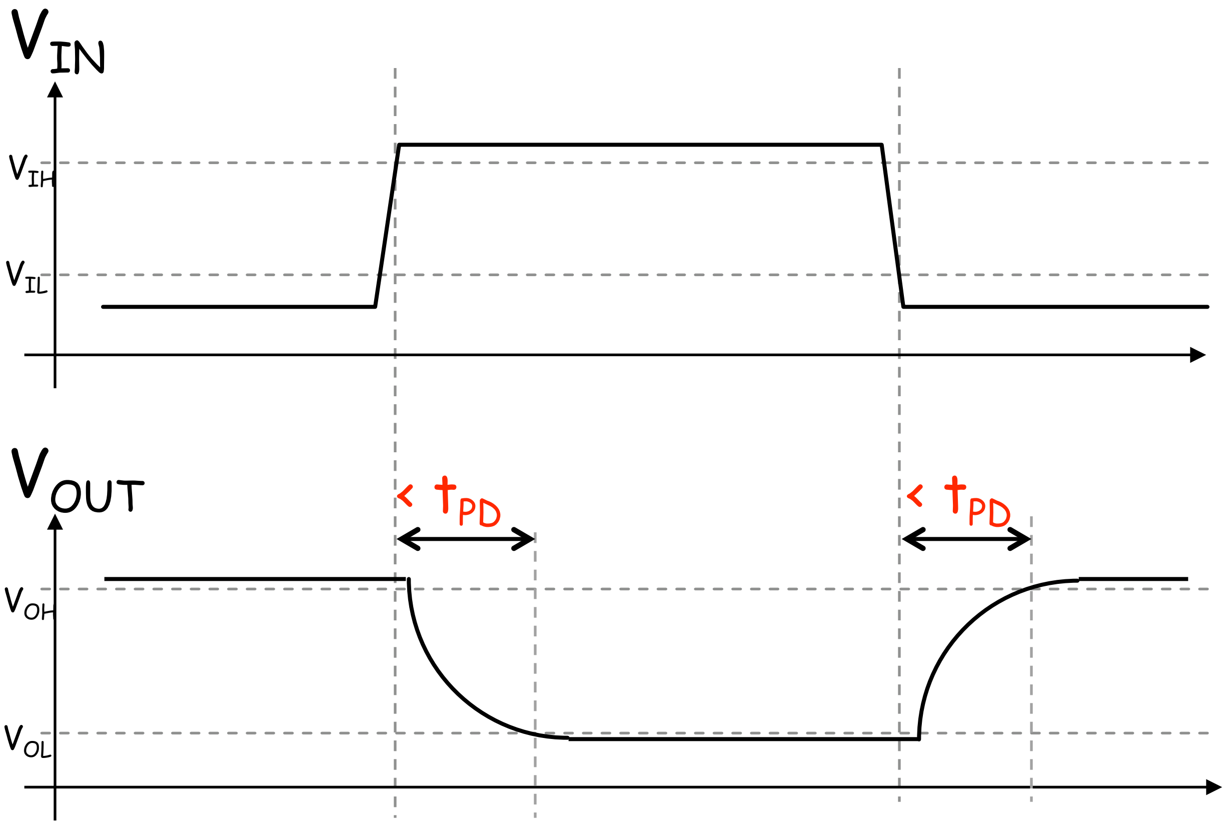
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

3 binary inputs
so $2^3 = 8$ rows in our truth table

An concise, unambiguous technique for giving the functional specification of a combinational device is to use a *truth table* to specify the output value for each possible combination of input values (N binary inputs -> 2^N possible combinations of input values).

Timing Specifications

Propagation delay (t_{PD}): An *upper bound* on the delay from valid inputs to valid outputs (aka "$t_{PD,MAX}$")

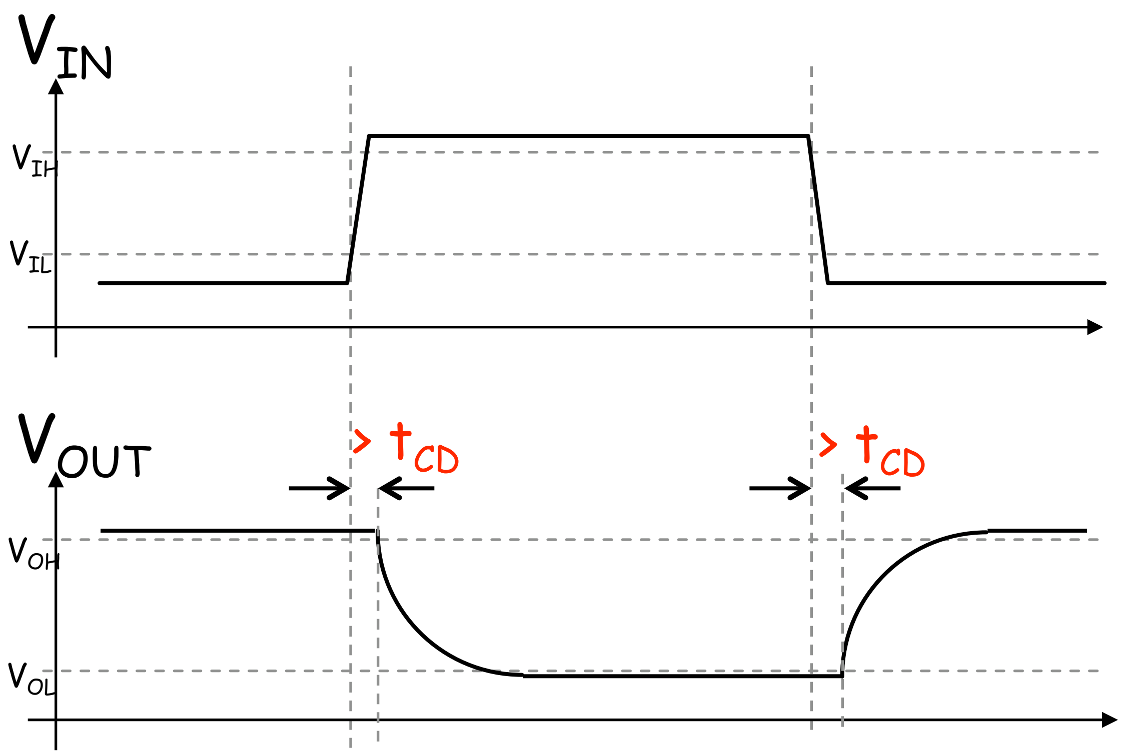


Design goal:
minimize propagation delay

Contamination Delay

an optional, additional timing spec

Contamination delay(t_{CD}): A *lower bound* on the delay from invalid inputs to invalid outputs (aka "$t_{PD,MIN}$")

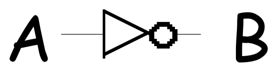


Do we really need t_{CD}?

Usually not… it'll be important when we design circuits with registers (coming soon!)

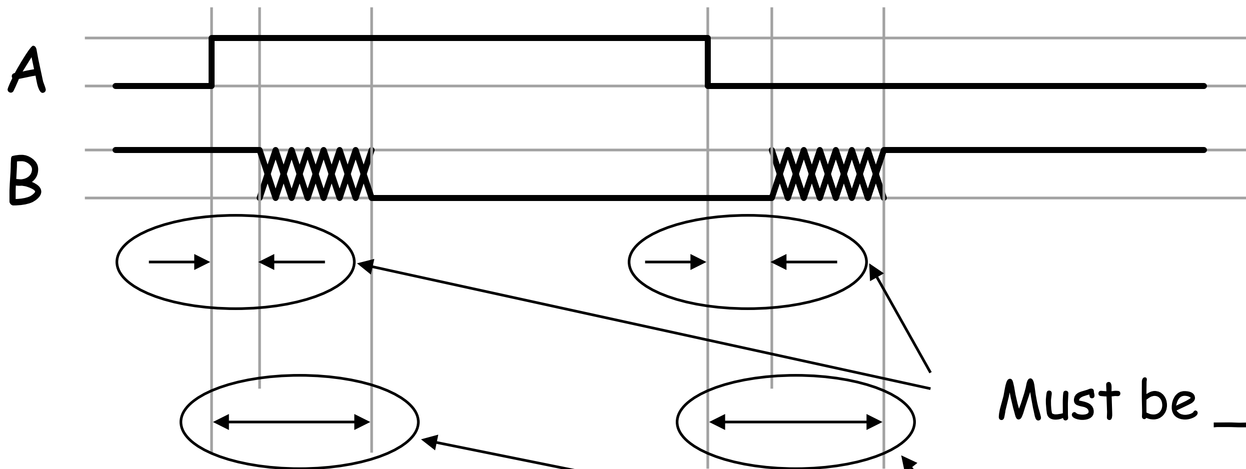
If t_{CD} is not specified, safe to assume it's 0.

The Combinational Contract



A	B
0	1
1	0

t_{PD} propagation delay
t_{CD} contamination delay



Must be $> t_{CD}$

Must be $< t_{PD}$

Note:

1. *No Promises* during 
2. Default (conservative) spec: $t_{CD} = 0$

Summary

- Use voltages to encode information
- "Digital" encoding
  - valid voltage levels for representing "0" and "1"
  - forbidden zone avoids mistaking "0" for "1" and vice versa
- Noise
  - Want to tolerate real-world conditions: NOISE.
  - Key: tougher standards for output than for input
  - devices must have gain and have a non-linear VTC
- Combinational devices
  - Each logic family has Tinkertoy-set simplicity, modularity
  - predictable composition: "parts work → whole thing works"
  - static discipline
    - digital inputs, outputs; restore marginal input voltages
    - complete functional spec, *e.g.*, a truth table
    - valid inputs lead to valid outputs in bounded time ($<t_{PD}$)