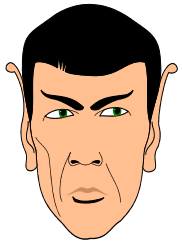
Logic Gates

1. Timing Issues
2. Specifying Logic Functions
3. Logic Design



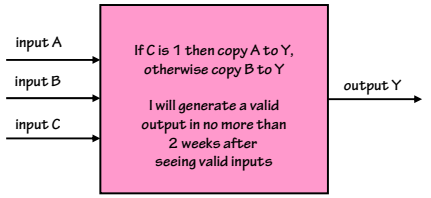

Handouts: Lecture Slides

Combinational Devices

Static discipline

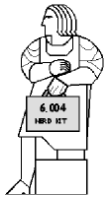
A ***combinational device*** **is a circuit element that has**

- 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



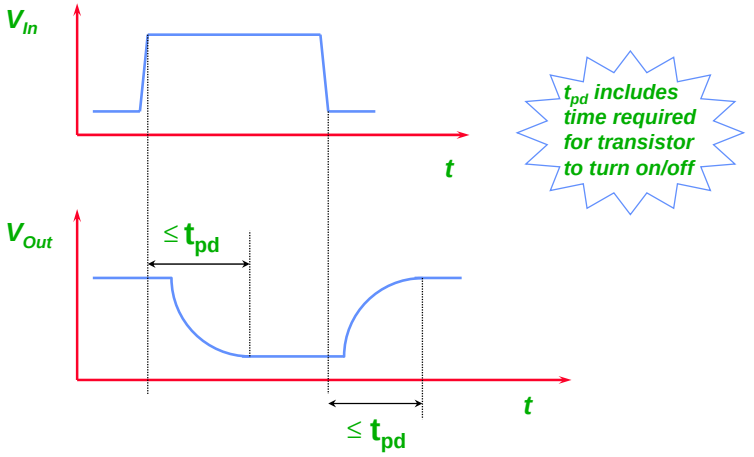
A Combinational Digital System

- **A set of interconnected elements is a combinational device if**
  - each circuit element is combinational
  - every element input is connected to exactly one element output or 1 (power supply) or 0 (ground)
  - 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!



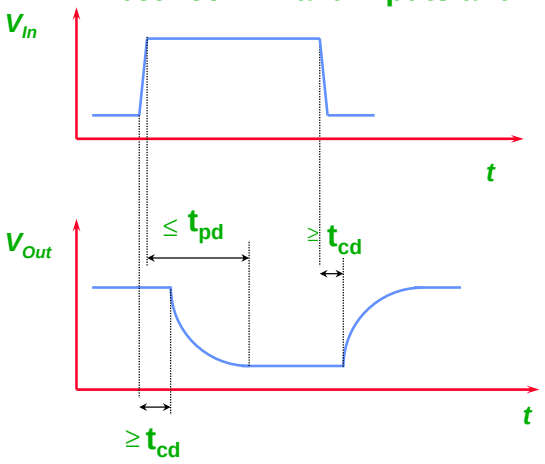
Signal Timing: Propagation Delay

Propagation delay t_{pd} is <u>upper bound</u> between new valid inputs and new valid output



Signal Timing: Contamination Delay

Contamination delay t_{cd} is <u>lower bound</u> between invalid inputs and invalid output

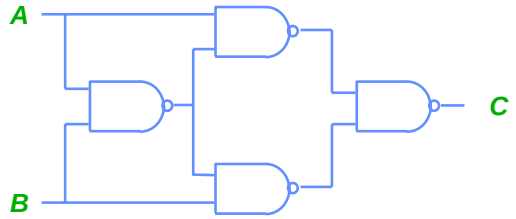


Do we really need t_{CD}?

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.

Timing Analysis



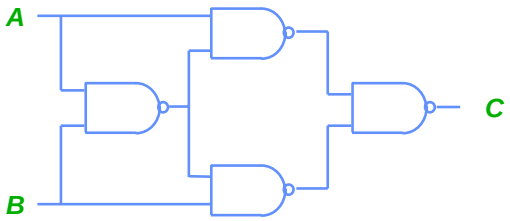
Logically valid A, B → Logically valid C (static discipline)

$t_{pd} = 5$ $t_{cd} = 1$ for each NAND gate

<u>Over all input combinations, over all input-output paths</u>

What is the propagation delay for the circuit?
What is the contamination delay for circuit?

Logical Analysis



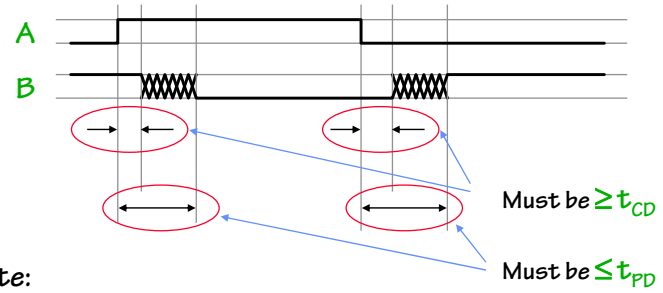
Logic Function?

A	B	C
0	0	
0	1	
1	0	
1	1	

Combinational Contract Summary

A	B
0	1
1	0

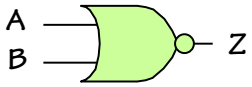
t_{PD} propagation delay
t_{CD} contamination delay



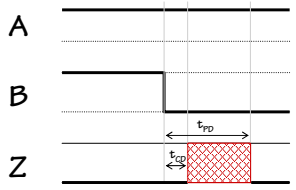
Note:
1. *No Promises* during
2. Default (conservative) spec: $t_{CD} = 0$

Oh yeah… one last issue

NOR:



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

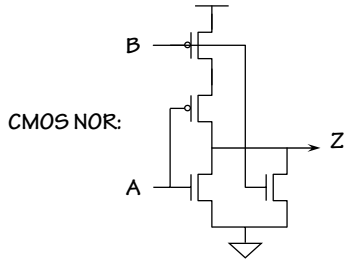


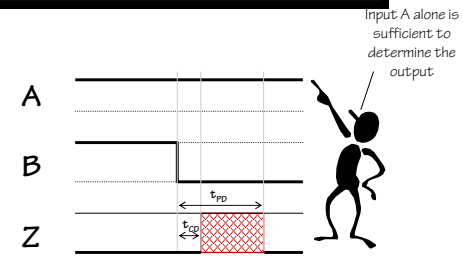
- **Recall the rules for *combinational devices*:**

  Output guaranteed to be valid when all inputs have been valid for at least t_{PD}, and, outputs may become invalid no earlier than t_{CD} after an input changes!

  Many CMOS gate implementations adhere to even tighter restrictions.

What happens in this case?





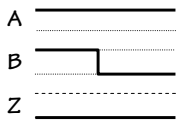
- ***LENIENT* Combinational Device:**
  Output guaranteed to be valid when any combination of inputs sufficient to determine output value has been valid for at least t_{PD}. *Tolerates transitions -- and invalid levels -- on irrelevant inputs!*

NOR:

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

Lenient NOR:

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



How do we specify what we want?

Truth Tables

k inputs — *output*

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

2^k rows

input combination

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

Assume unspecified input combinations produce 0

Number of Distinct truth tables with k inputs is 2^{2^k}

Truth Tables and Logic Equations

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

F = 1 if A=0 AND B=0 AND C=0 OR
A=0 AND B=0 AND C=1 OR
A=0 AND B=1 AND C=0 OR
A=0 AND B=1 AND C=1 OR
A=1 AND B=1 AND C=0
F = 0 otherwise

Write as a logic equation:

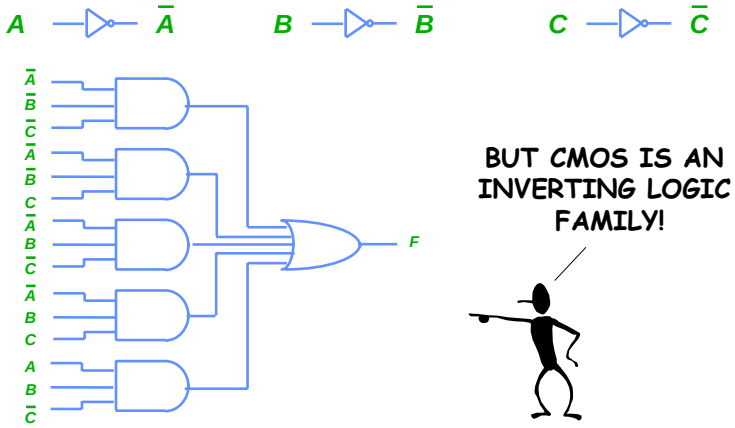
$$F = \bar{A}\cdot\bar{B}\cdot\bar{C} + \bar{A}\cdot\bar{B}\cdot C + \bar{A}\cdot B\cdot\bar{C} + \bar{A}\cdot B\cdot C + A\cdot B\cdot\bar{C}$$

or

$$F = \bar{A}\,\bar{B}\,\bar{C} + \bar{A}\,\bar{B}\,C + \bar{A}\,B\,\bar{C} + \bar{A}\,B\,C + A\,B\,\bar{C}$$

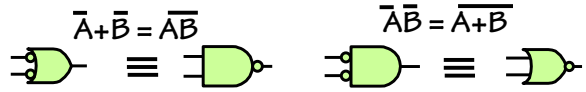
Logic Equations and AND-OR Circuits

$$F = \bar{A}\,\bar{B}\,\bar{C} + \bar{A}\,\bar{B}\,C + \bar{A}\,B\,\bar{C} + \bar{A}\,B\,C + A\,B\,\bar{C}$$

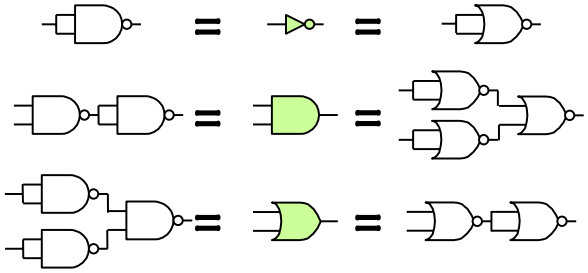


No problem…

- DeMorgan's laws

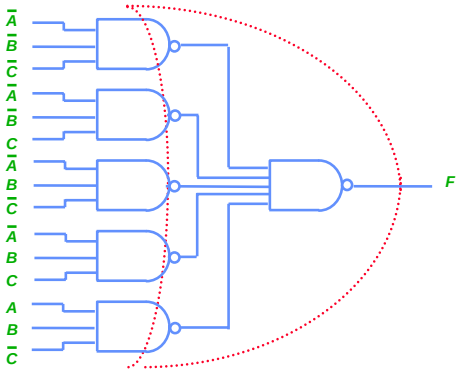


- NAND and NORs are universal



NAND-NAND Circuit

$$F = \bar{A}\,\bar{B}\,\bar{C} + \bar{A}\,\bar{B}\,C + \bar{A}\,B\,\bar{C} + \bar{A}\,B\,C + A\,B\,\bar{C}$$



DeMorgan's law:

$$f_1 + f_2 + f_3 = \overline{\bar{f}_1 \cdot \bar{f}_2 \cdot \bar{f}_3}$$

Number of transistors if implemented in CMOS: 40 (ignoring input inverters)

Identities for Logic Functions

Given logic functions f, g, h

$f \cdot f = f$	$f + f = f$
$f \cdot 1 = f$	$f + 1 = 1$
$f \cdot 0 = 0$	$f + 0 = f$
$f \cdot \bar{f} = 0$	$f + \bar{f} = 1$

Distributive laws

$$f \cdot (g + h) = f \cdot g + f \cdot h$$
$$(f + g) \cdot (f + h) = f + g \cdot h$$ WHY?

Simplifying Logic Equations

Apply identities to simplify equation for F

$$F = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + AB\bar{C}$$

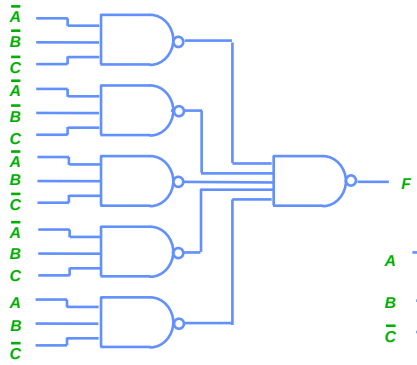
$$= \bar{A}\bar{B} + \bar{A}B + AB\bar{C}$$

$$= \bar{A} + AB\bar{C}$$

Can we simplify further?

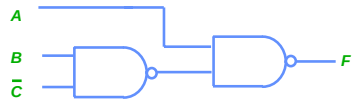
Comparison

$$F = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + AB\bar{C}$$



40 Transistors

$$F = \bar{A} + B\bar{C}$$



8 Transistors

Parity (XOR) Functions

Count the number of 1's in the inputs and produce a 1 if odd

A	B	P_2
0	1	1
1	0	1

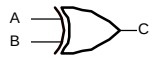
A	B	C	P_3
0	0	1	1
0	1	0	1
1	0	0	1
1	1	1	1

Smallest AND-OR circuit for P_n will contain 2^{n-1} AND gates

Similar increase for adders, subtractors, multipliers, and most arithmetic functions

Combinational Composition

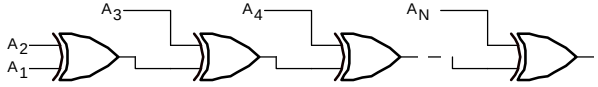
Suppose we have some 2-input XOR gates:



$t_{pd} = 1$
$t_{cd} = 0$

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

And we want an N-input XOR:

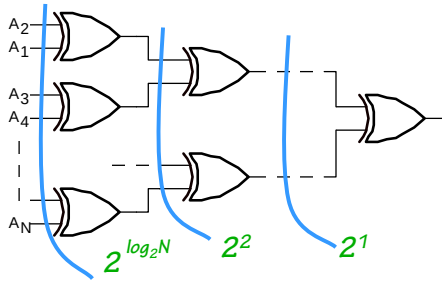


output = 1 iff number of 1s input is ODD ("ODD PARITY")

$t_{pd} = O(N)$ -- WORST CASE.

Can we compute N-input XOR faster?

I think that I shall never see
a circuit lovely as...

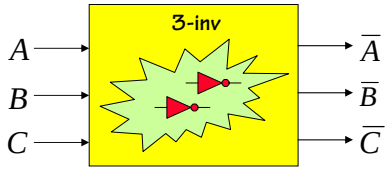


N-input TREE has $O(\log_2 N)$ levels.

Signal propagation takes $O(\log_2 N)$ gate delays.

Logic Geek Party Games

- **You have plenty of ANDs and ORs, but only 2 inverters. Can you invert more than 2 independent inputs?**



- **CHALLENGE: Come up with a combinational circuit using ANDs, ORs, and at most 2 inverters that inverts A, B, and C !**
- **Such a circuit exists. What does that mean?**
  - **If we can invert 3 signals using 2 inverters, can we use 2 of the pseudo-inverters to invert 3 more signals?**
  - **Do we need only 2 inverters to make ANY combinational circuit?**
- **Hint: Is our 3-inv device *LENIENT?***