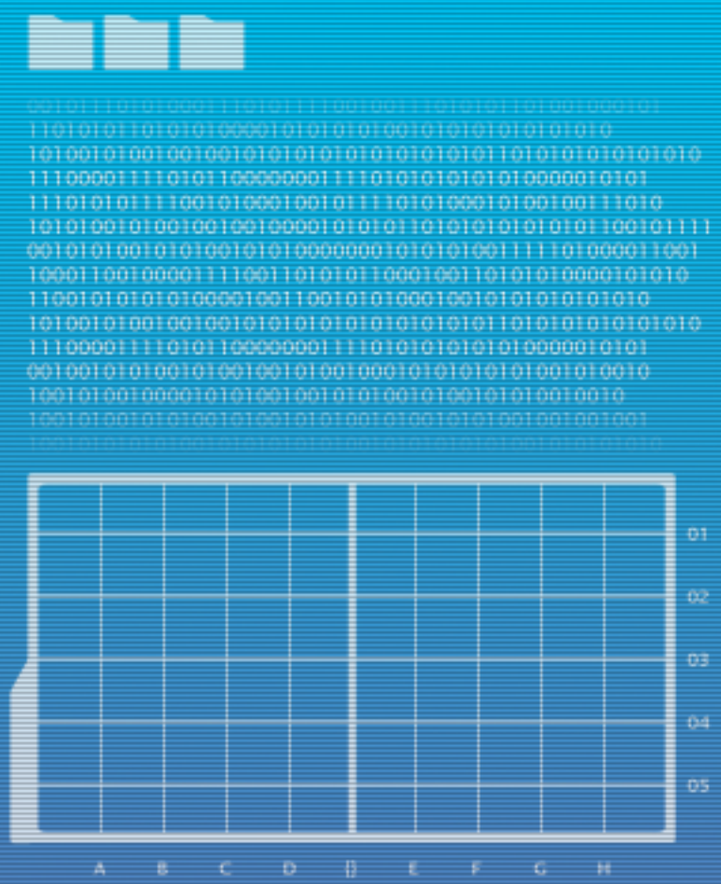
DEPARTMENT OF INFORMATION SYSTEMS AND COMPUTER SCIENCE

More CMOS Gates

Review of Digital Logic



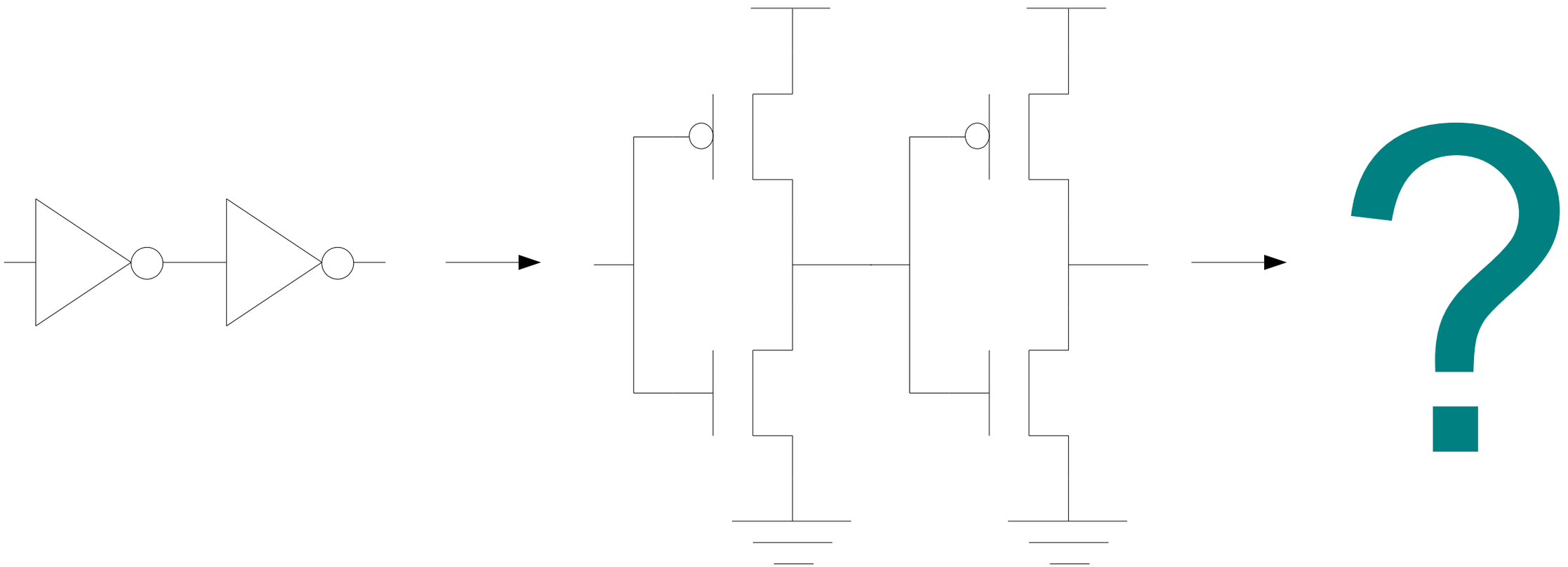
Joke Time!

+

=

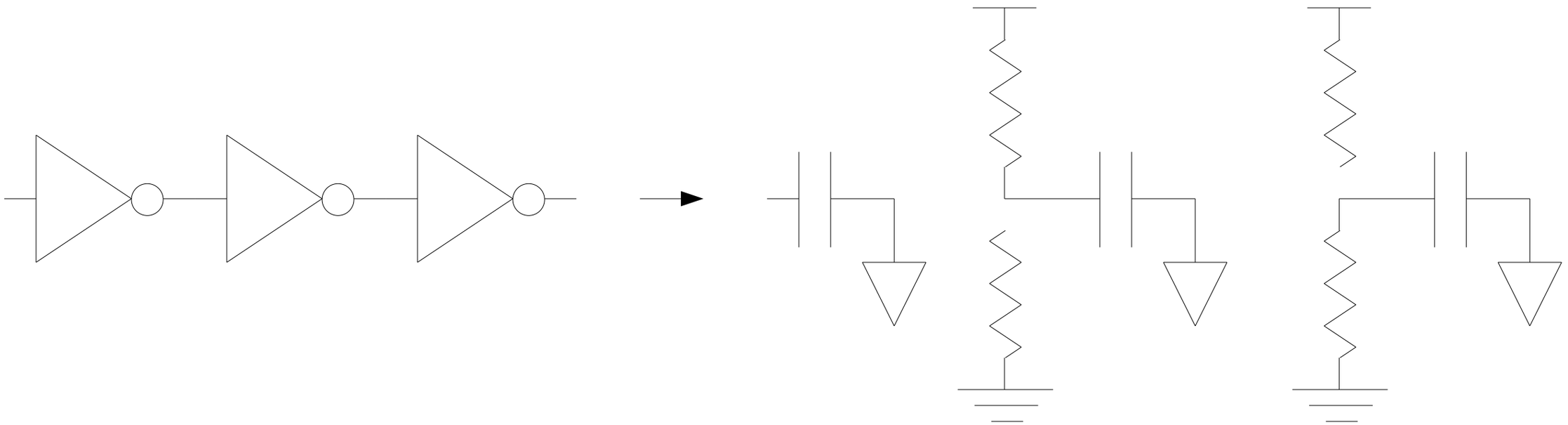
Transistor Limitations

- What happens if you chain 2 CMOS inverters?

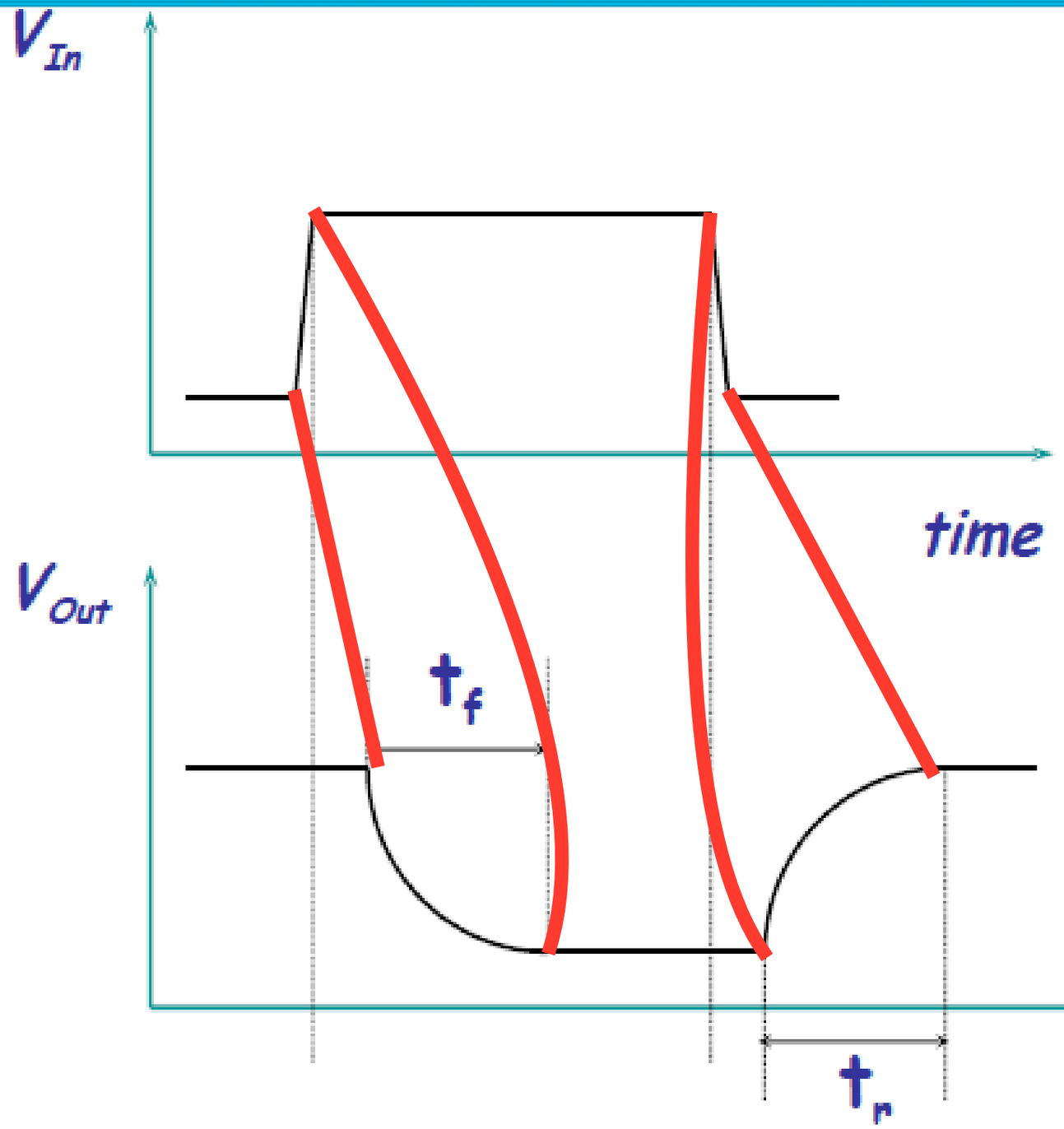


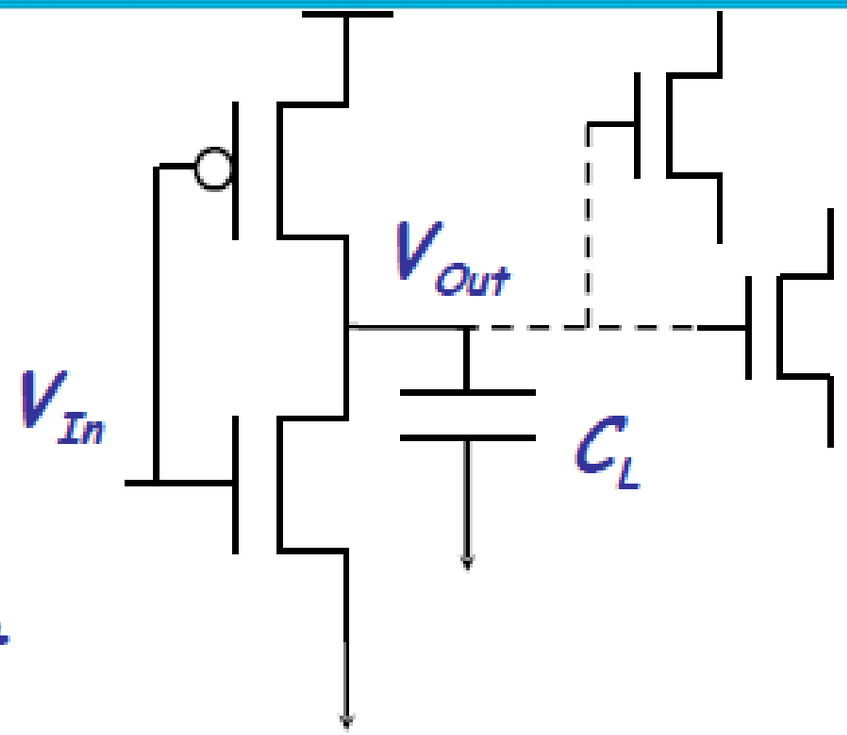
Delays

- Logic gates have:
  - Output resistance
  - Input capacitance
- They form an RC circuit when chained together!



Rise Time and Fall Time

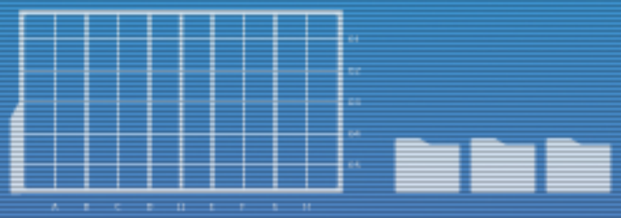




Time constant of t_f, t_r equals $R_{ON}\ C_L$

Propagation Delay

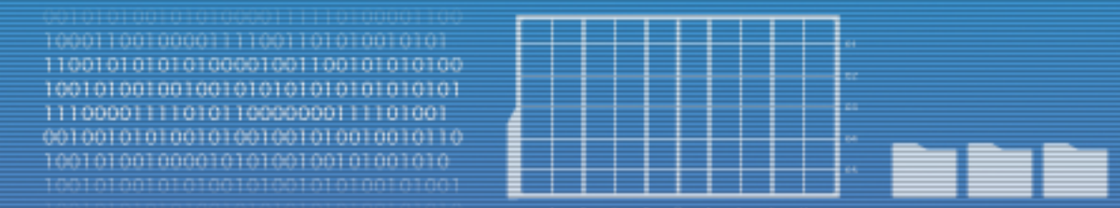
- Propagation delay (t_{pd}) is the upper bound between new valid inputs and new valid outputs, or how long it takes for the entire system to “settle” after inputs are changed.
  - In plain English, when an input is changed and becomes its new valid value, what's the maximum wait time before a corresponding valid output emerges?
  - Let's try to determine t_{pd} from the graph in the previous slide.
    - It should include the time it takes for the transistor to turn on/off, as well as the rise/fall time.



DISCS

Contamination Delay

- Contamination delay (t_{cd}) is lower bound between invalid inputs and invalid output, or how long it takes for one invalid signal to mess up the result.
  - In plain English, when an input is changed and enters the forbidden zone during transition, what's the minimum wait time before an output passes through the forbidden zone too?
  - t_{cd} is also known as minimum t_{pd} or $t_{pd\text{-}min}$
    - A device's t_{pd} cannot go below its t_{cd} … why not?
  - If not known, assume $t_{cd} = 0$, the worst-case scenario.
    - As soon as an invalid input enters, output becomes invalid.



Definitions

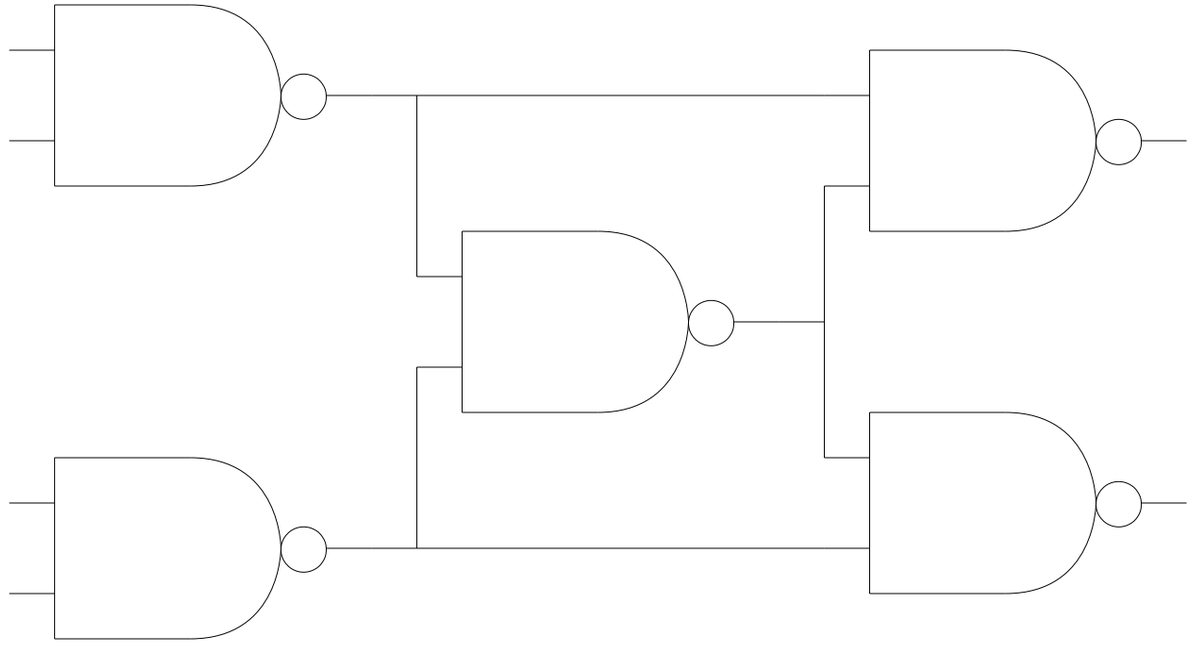
- Propagation delay t_{pd}: Worst-case delay from new valid inputs to new valid outputs.
- Contamination delay t_{cd}: Best-case delay from invalid inputs to invalid outputs.
  - This is over all input combinations, over all input-output paths!
    - That's a lot of testing / measurements!
  - Delays are additive: Device t_{pd} and t_{cd} is sum of its parts' t_{pd}s and t_{cd}s.

Definitions

- Rise time t_r: Output signal transition time from invalid low to valid high.
- Fall time t_f: Output signal transition time from invalid high to valid low.
  - While not quite true, we can assume t_r and t_f to be properties of only the last stage (in an acyclic circuit).
    - After all, final output comes from the last stage of any given acyclic circuit, so rise/fall time measurement comes from there BUT remember that these times ARE STILL affected by the rise/fall times of the preceding stages!

Delays in Combinational Circuits

- Static discipline dictates that upon receiving a logically valid A and B, the circuit below should output a logically valid C (after some delay).
- Assuming each gate has a t_{pd} = 5ns and t_{cd} = 1ns, what's the t_{pd} and t_{cd} of the circuit below?

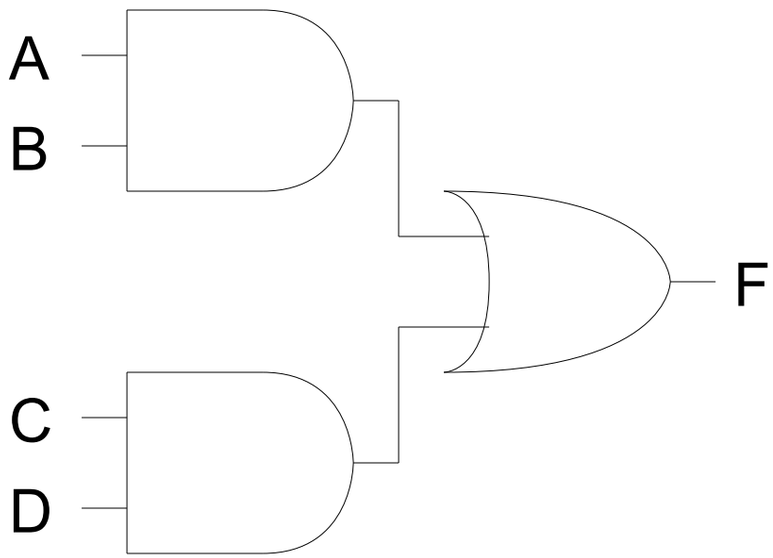


CMOS is cool, but...

- We can't make customized gates every time we want a Boolean function implemented!
- Why not use what's already available?
- Already have 1-input to 4-input gates!
- We can make anything entirely out of NANDs or NORs!
  - Lab exercises in optimization, yo.
- Boolean functions consist of NOTs (inverters), ANDs, and ORs anyway!

Simplistic Viewpoint

- All we need are two-level circuits!
  - Inverters for inputs not included.
- Example: F = AB + CD

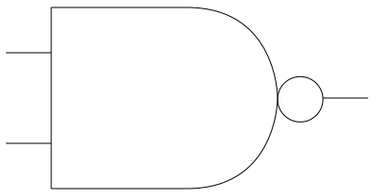


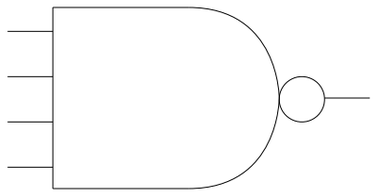
Simplistic Viewpoint

- Two-level circuits are fast because transitions only propagate through at most two levels of logic.
  - Again, not including input inverters.
- Make two-level circuits for:
  - F = (!A)BC + DE
  - F = !A + BC
  - F = AB + C(D + E)
  - F = A(B + C(D + E))
  - F = !(A (B + C(D + E)))

Circuit Delay

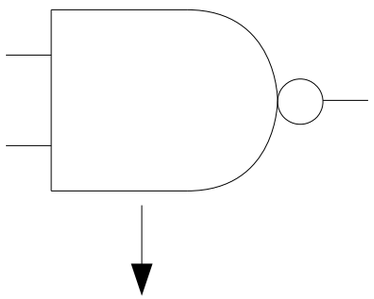
- Fan-in refers to the number of digital inputs a logic gate has.
- Since we're talking about making two-level circuits, wouldn't it be nice to have 128-input versions of our current logic gates?



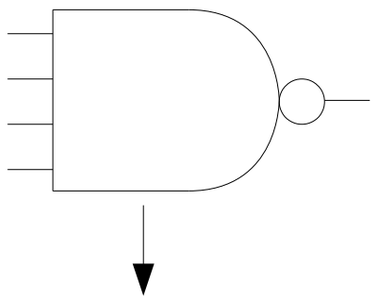


Circuit Delay

- Higher fan-in may mean more transistors in series.
- This means higher resistance encountered.
- Since gates are often connected to each other, this means greater RC delays.



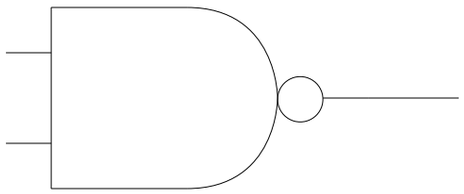
2 NFETs
in series
in CMOS

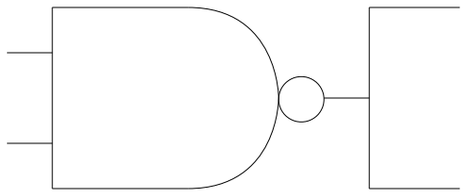


4 NFETs
in series
in CMOS

Circuit Delay

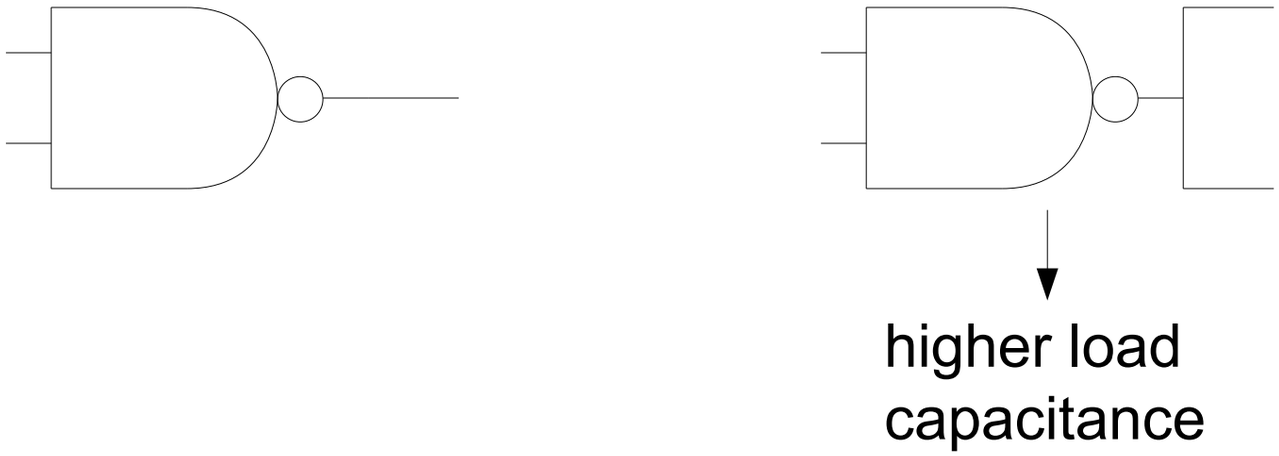
- Fan-out refers to the maximum number of input nodes that a logic gate can output to.
- Chip makers usually specify this (and voltage and current) to guarantee their advertised t_{pd}.





Circuit Delay

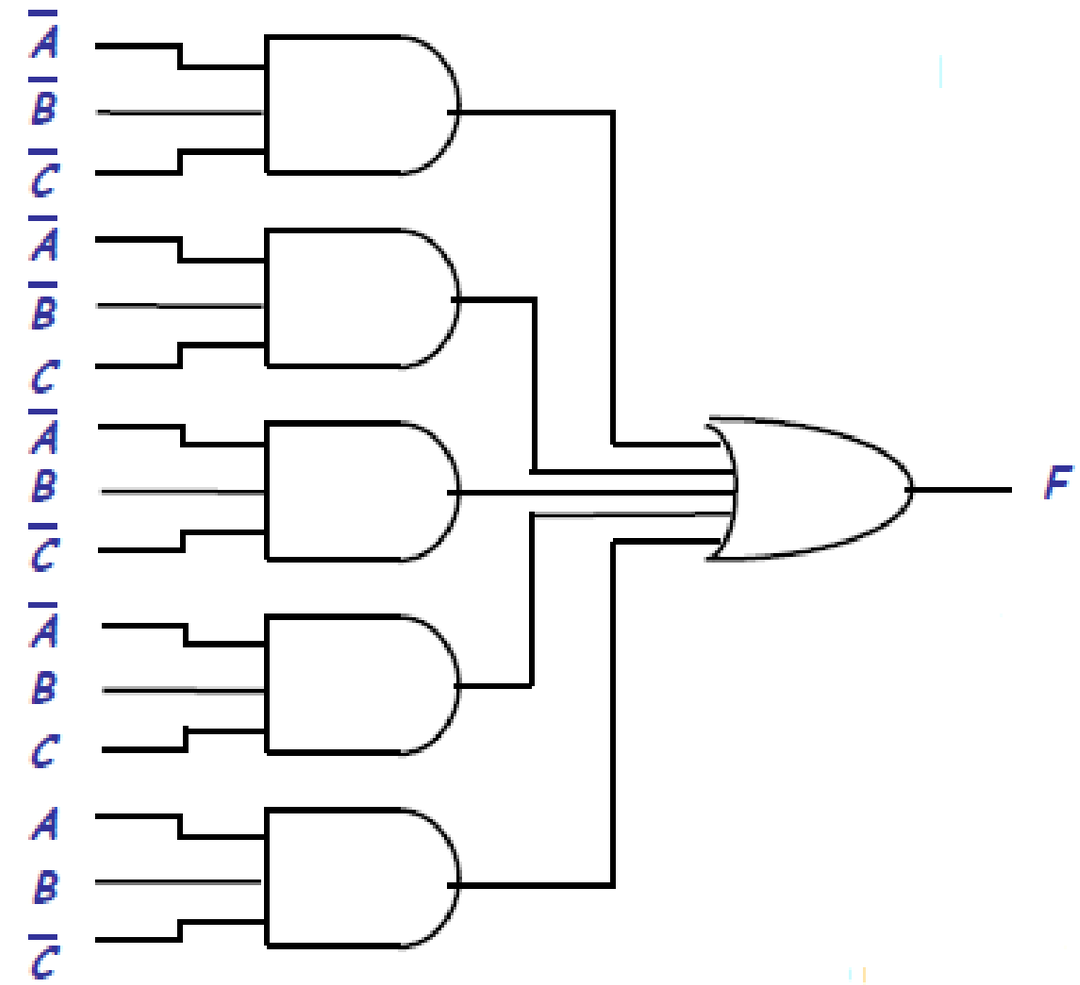
- Higher fan-out means higher capacitance encountered.
- This means greater RC delays.



Truth Tables and Logic Circuits

- Truth tables define Boolean functions...
- … which can be implemented as 2-level logic circuits.

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

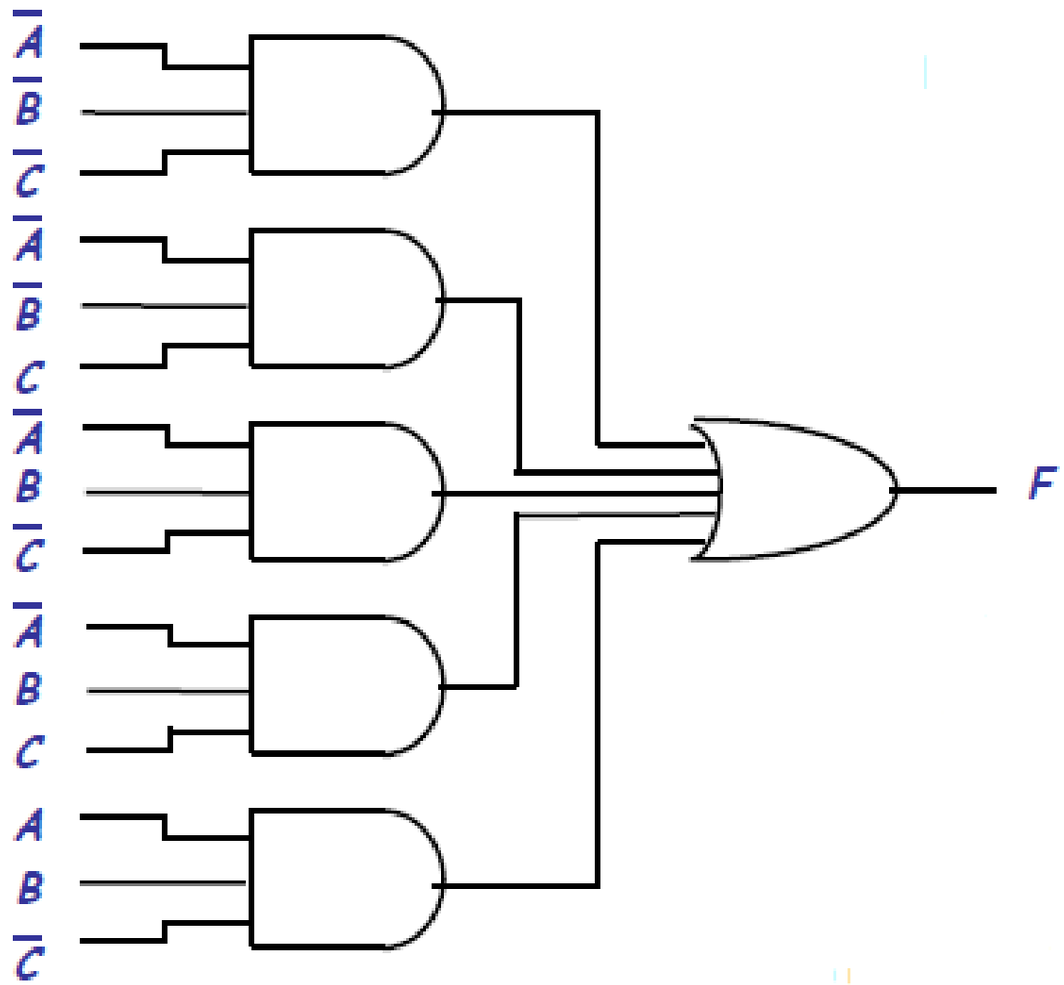


Can We Do Better?

- ANDs tend to be either NANDs with an output inverter or NORs with input inverters.
- ORs tend to be either NORs with an output inverter or NANDs with input inverters.
- This will affect transistor count!

Expensive AND Slow

- Note: Assume !A, !B, and !C are already available.
- 3-input AND gate
  = 3-input NAND + inverter
  = 6 + 2 = 8 transistors
- 5 of those, so...
  = 5 * 8 = 40 transistors
- 5 input OR gate
  = 5-input NOR + inverter
  = 10 + 2 = 12 transistors
- Total
  = 40 + 12 = 52 transistors



Does it have to be AND-OR?

- DeMorgan's Law states that an OR gate with inverted inputs is a NAND gate.
  - While we are *logically* adding inverters, we are actually converting the OR to a NAND and therefore *physically* removing inverters.
- We can't just add an inverter to each input wire connecting to the OR gate – that will change the actual Boolean formula.
  - If I add an inverter, what can I add to cancel it out?

Does it have to be AND-OR?

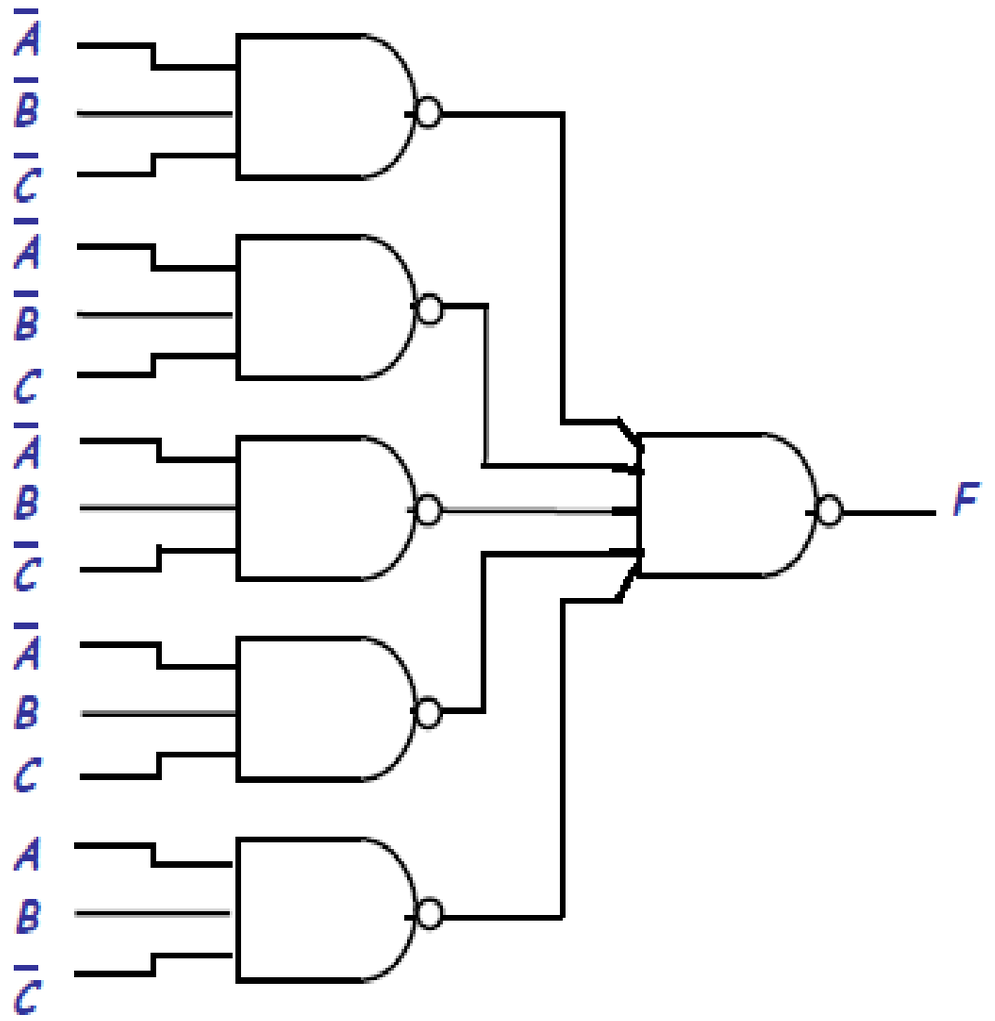
- We add two inverters to each input wire connecting to the OR gate!
  - This will allow us to replace the OR with NAND (one inverter from each input wire used)
- What about the second inverter?
  - Each input wire of the OR gate is the output wire of what gate?

Does it have to be AND-OR?

- An AND gate with an inverter at its output is simply a NAND gate.
  - We can replace the ANDs with NANDs (another inverter from each input wire used)
- We *logically* added inverters...
- … to *physically* remove inverters...
- … and *improve performance* (reduce t_{pd} / t_{cd})!

… Still Expensive

- Note: Assume !A, !B, and !C are already available.
- 3-input NAND gate
  = 6 transistors
- 5 of those, so...
  = 5 * 6 = 30 transistors
- 5 input NAND gate
  = 10 transistors
- Total
  = 30 + 10 = 40 transistors



Can We Do Better?

- Do you remember how to simplify logic equations?
  - To be reviewed.
- Do you remember how to use Karnaugh maps?
  - Not to be reviewed, but you can use them during quizzes/exercises/tests if you want to.

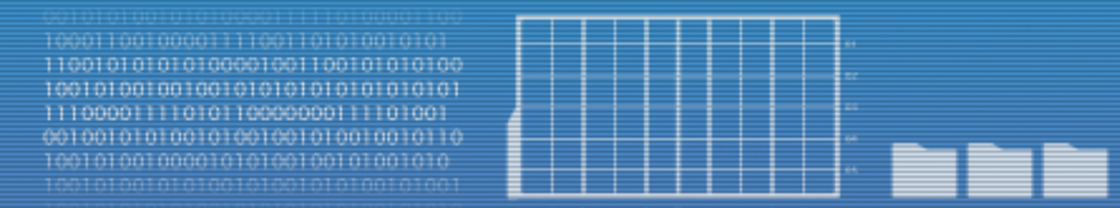
Review: Simplifying Logic

- Given logic functions `f`, `g`, and `h`:

```
(f)(f) = f        f + f = f

(f)(1) = f        f + 1 = 1

(f)(0) = 0        f + 0 = f

(f)(!f) = 0       f + !f = 1
```

- Distributive Laws

```
f (g + h) = fg + fh

(f + g)(f + h) = f + gh
```



Review: Simplifying Logic

F = (!A!B!C) + (!A!B C) + (!A B!C)
  + (!A B C) + (A B!C)

= _____ + _____ + (A B!C)

= _____ + (A B !C)

= _____

Cheap AND Fast

- Note: Assume !A, !B, and !C are already available.
- Inverter (1-input NAND?)
  = 2 transistors
- 2-input NAND
  = 4 transistors
- Another 2 input NAND
  = 4 transistors
- Total
  = 2 + 4 + 4 = 10 transistors