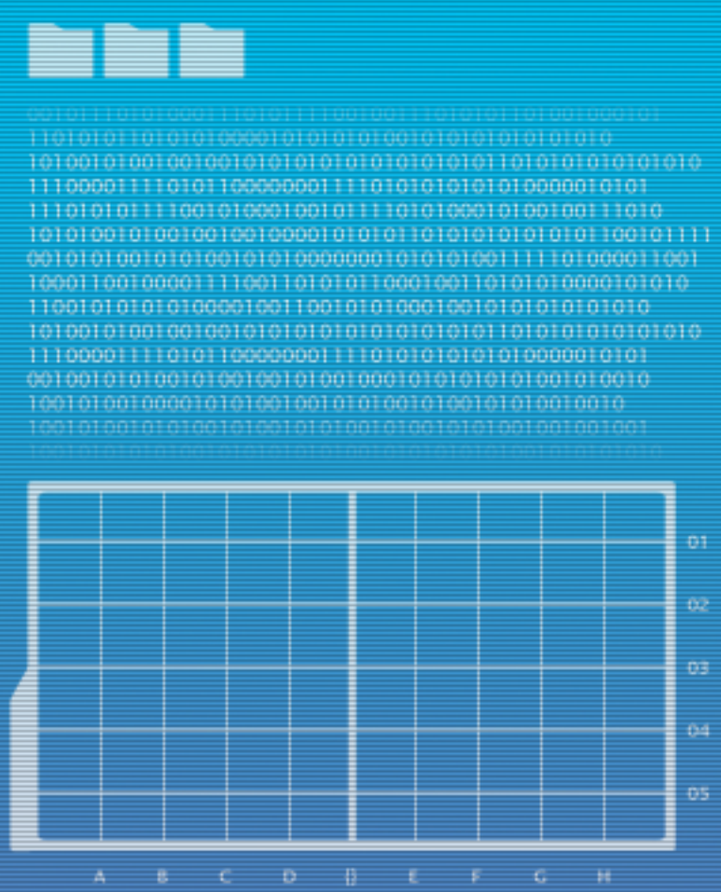
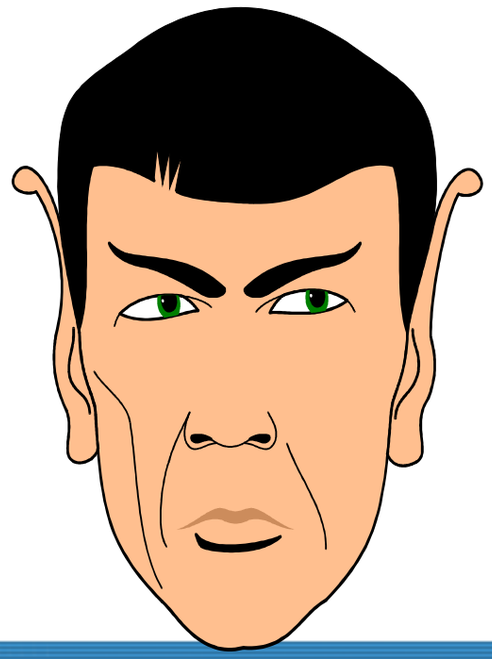
DEPARTMENT OF
INFORMATION SYSTEMS
AND COMPUTER SCIENCE

More CMOS Gates

And a Review of Digital Logic



+

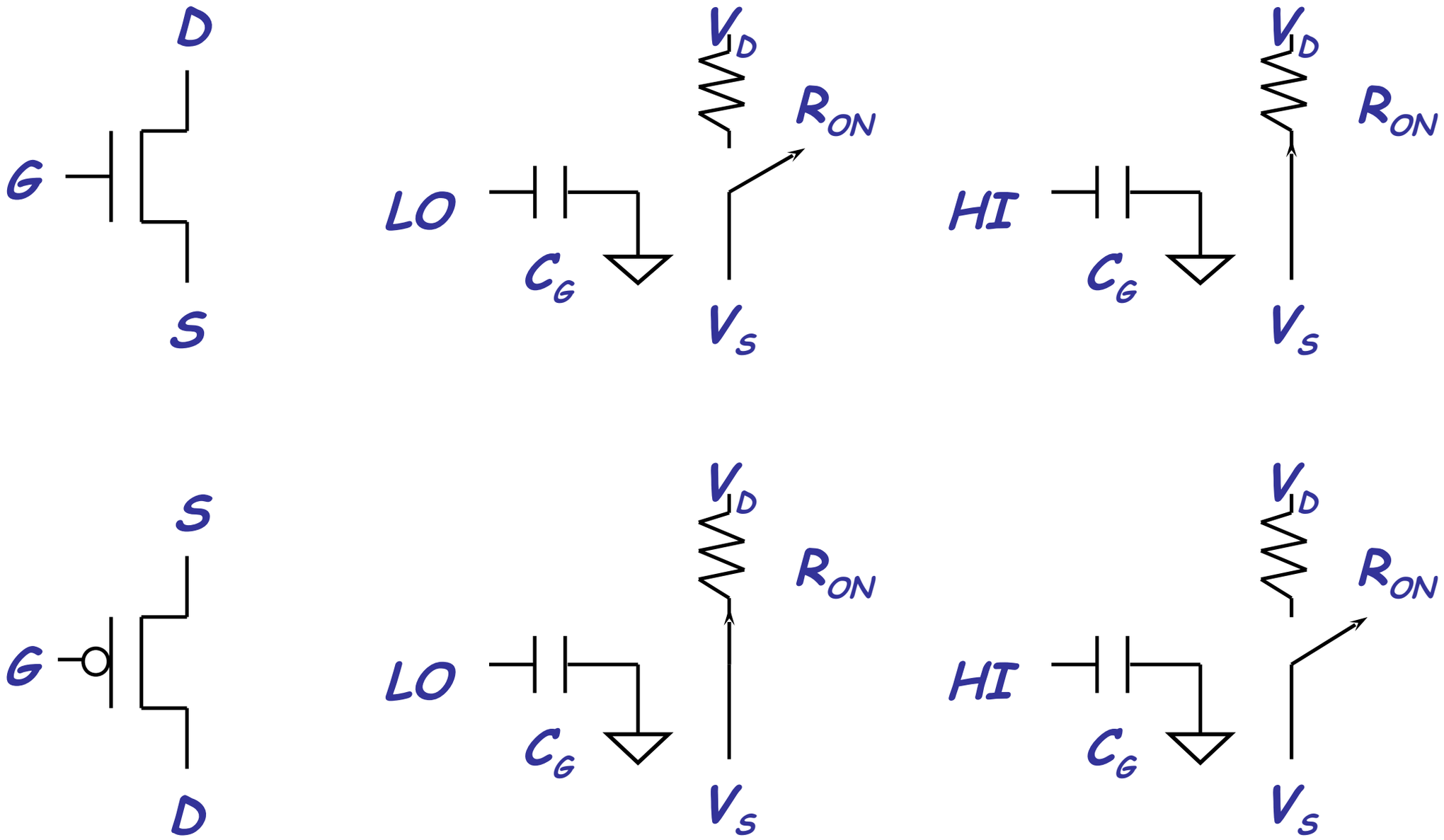


=

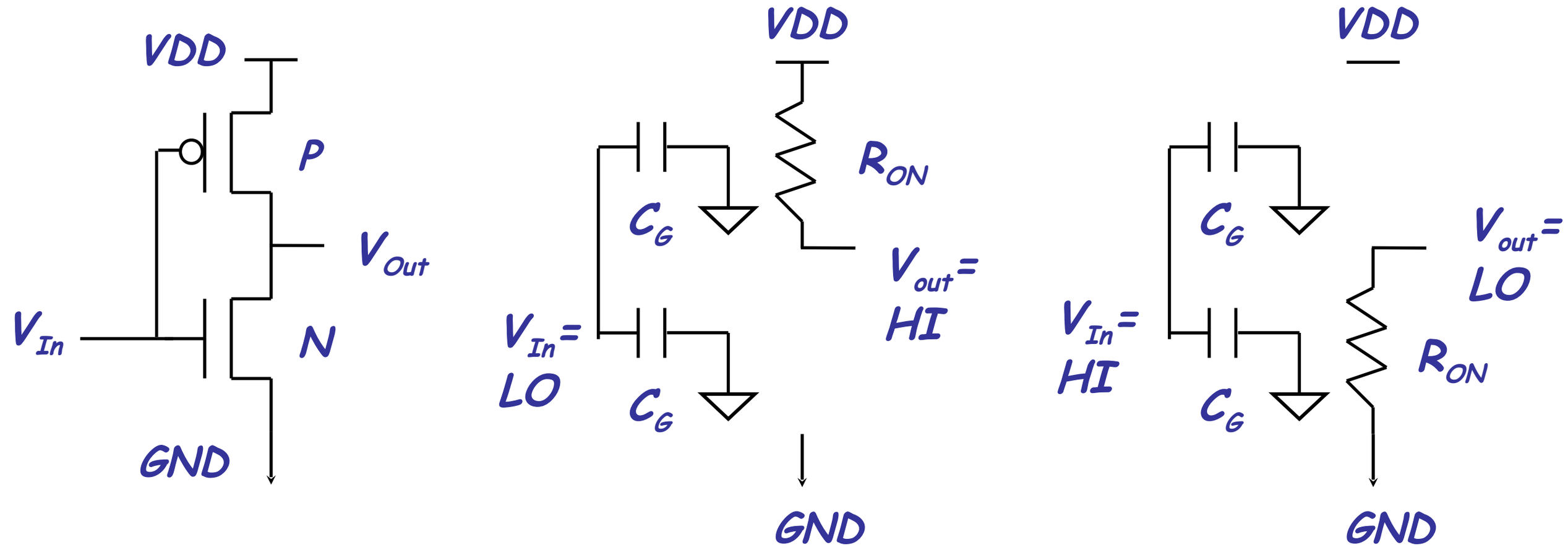
?

More Transistor Limitations

- Resistance (Ron) and Capacitance (Cg)

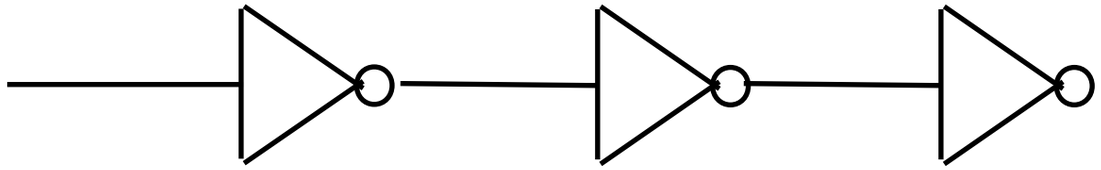


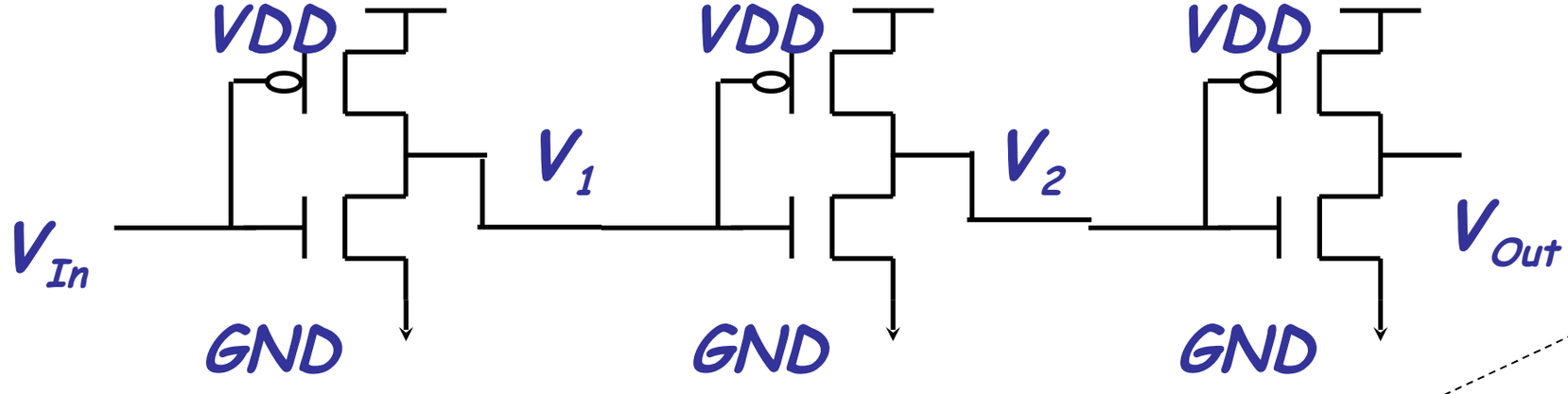
Delays: CMOS Inverter



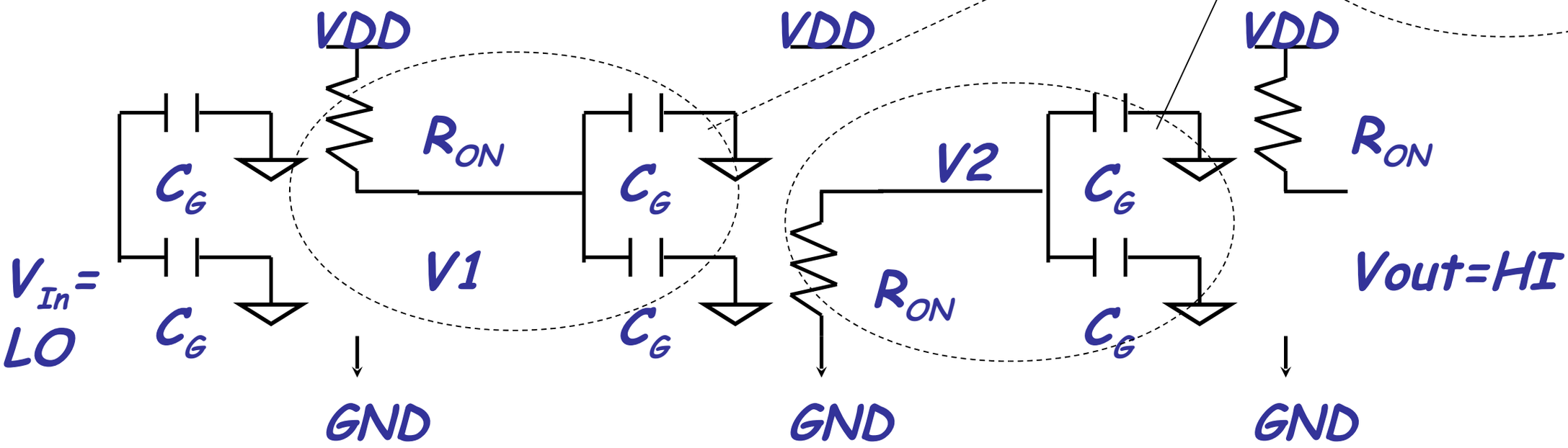
- Logic Gates have an Output Resistance and Input Capacitance.
- Form an RC circuit when chained together!

Example: Inverter Chain

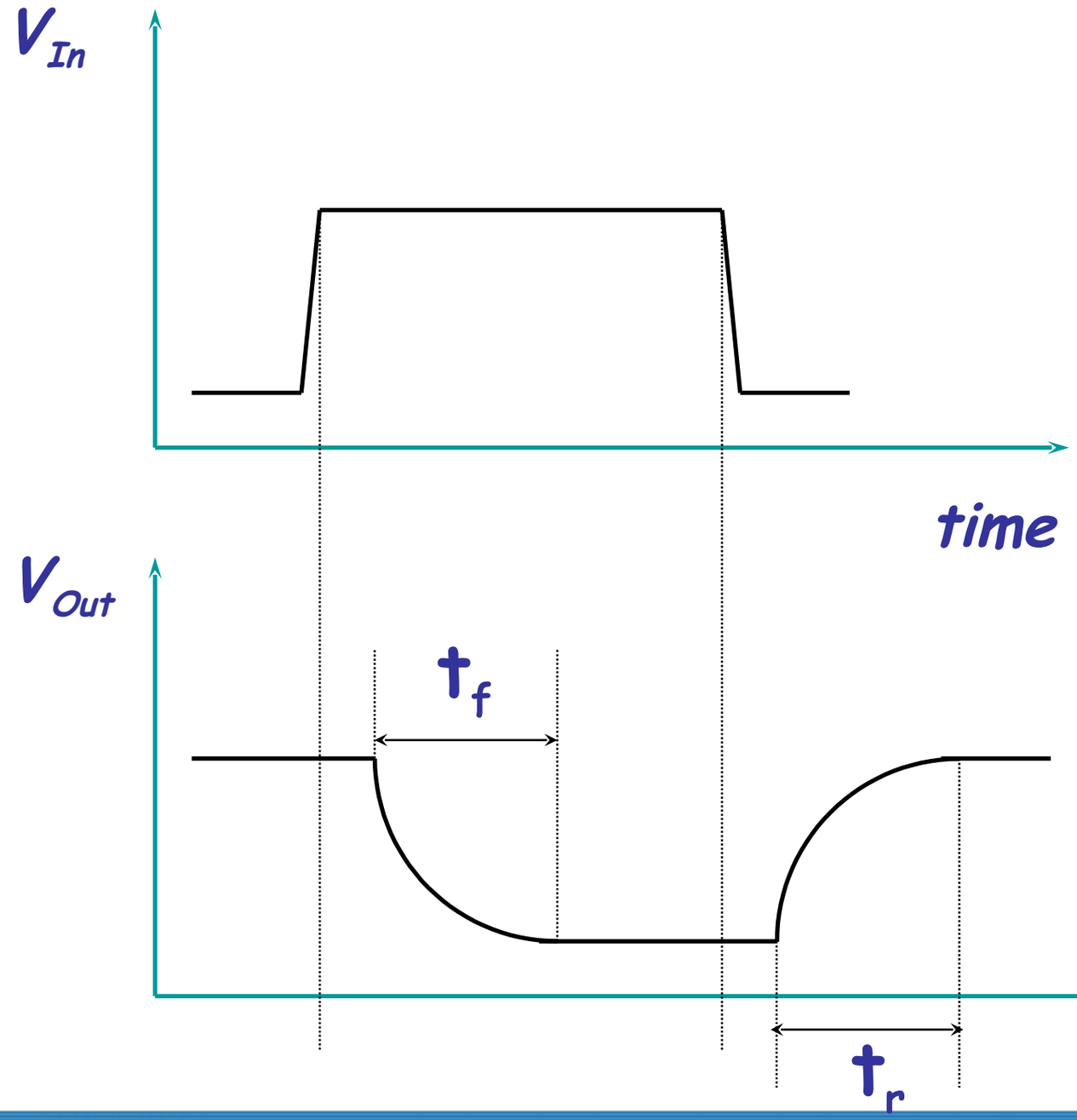


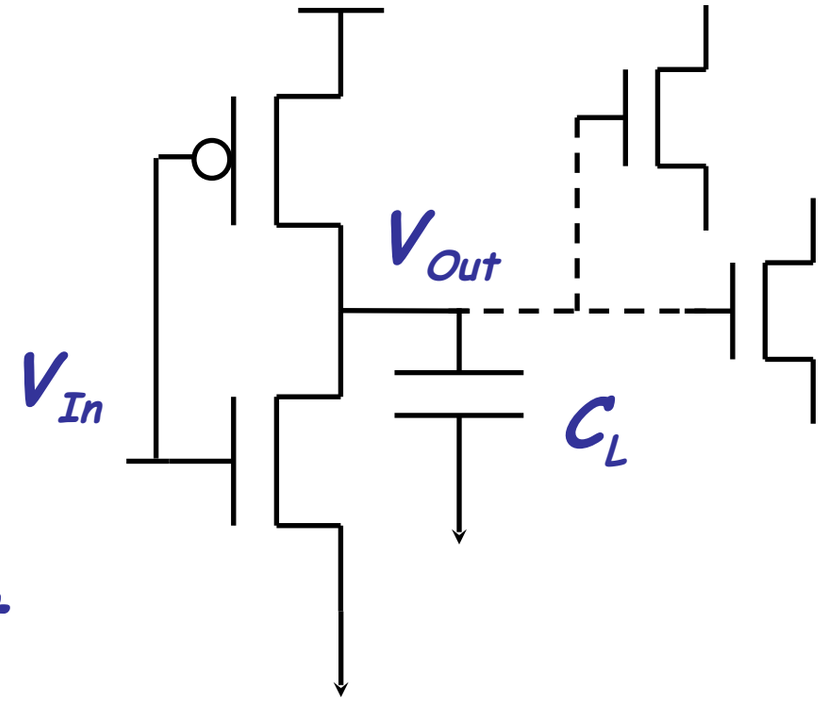


- **RC circuits:**
  - **Have *rise* and *fall* times!**
  - **Which mean delays!**



Delays: Rise and Fall Time

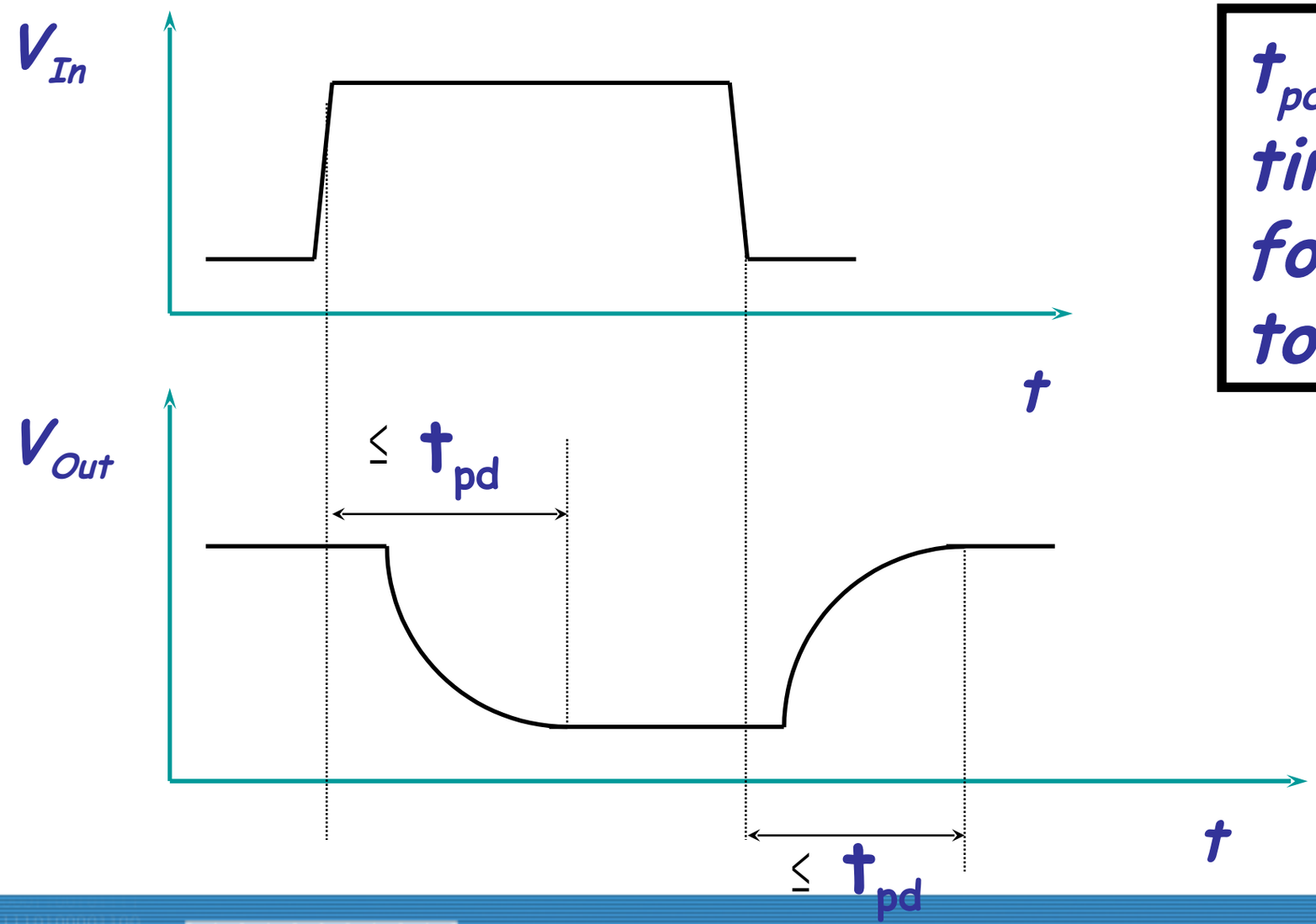




Time constant of t_f, t_r equals $R_{ON}\ C_L$

Delays: Propagation Delay

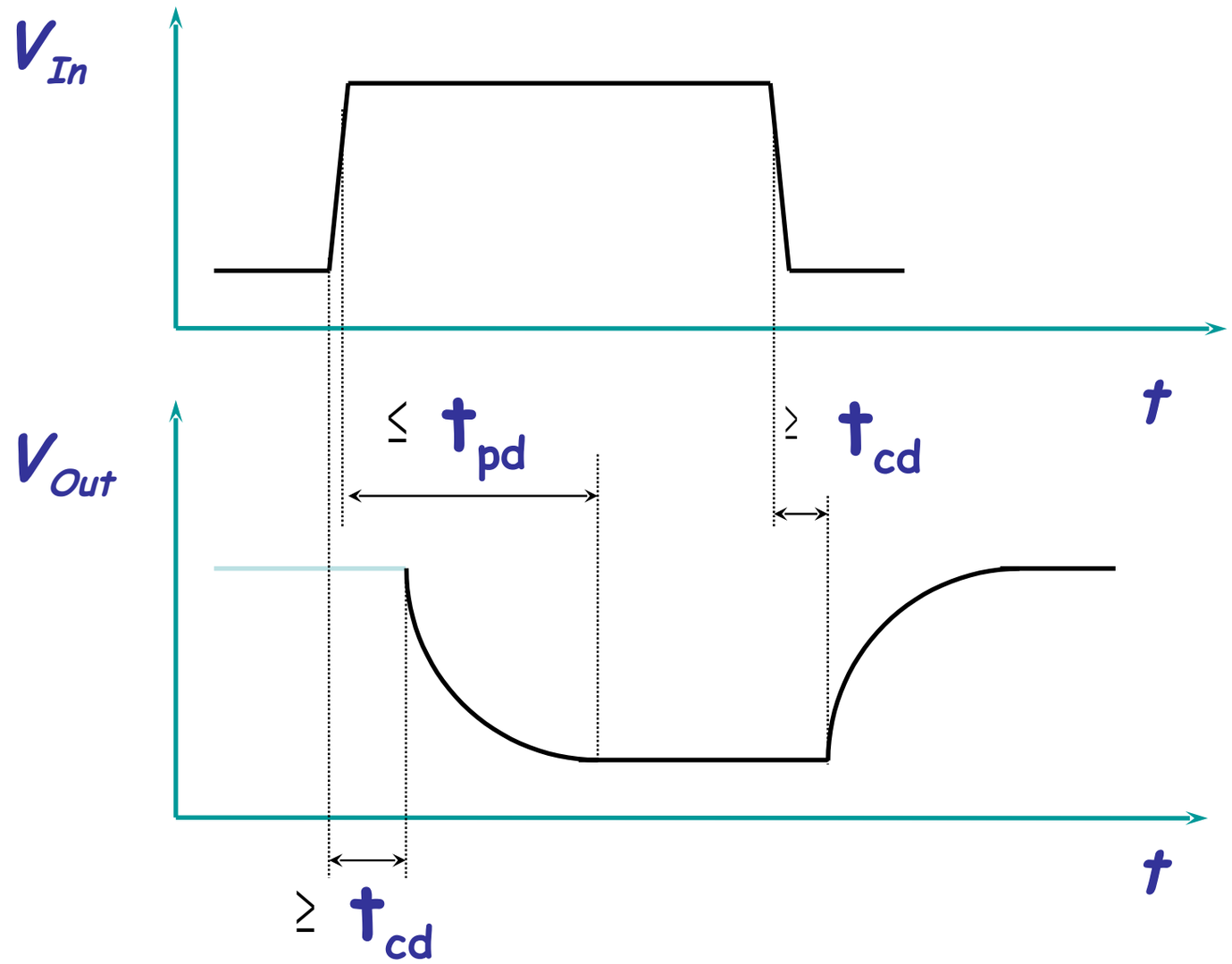
Propagation delay (t_{pd}) is upper bound between new valid inputs and new valid output, or how long it takes for the entire system to "settle" after inputs are changed.



t_{pd} includes time required for transistor to turn on/off.

Signal Timing: Contamination Delay

Contamination delay t_{cd} is lower bound between <u>invalid</u> inputs and <u>invalid</u> output, or how long it takes for one invalid signal to mess up the result.



- tcd is also known as *minimum tpd* or tpd_{min} but this obviously causes confusion.
  - <u>A device's tpd *cannot* go below its tcd!</u>
- If not known, assume tcd = 0, which is the worst-case scenario: as soon as an invalid input enters, output becomes invalid.

Formal Definitions

additive: Device t_{pd} and t_{cd} is sum of its parts.

- **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.

Over all input combinations, over all input-output paths!

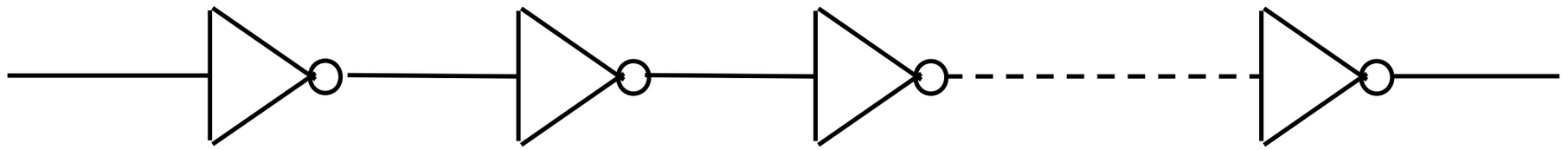
Property of last stage only (in an acyclic circuit).

- **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.

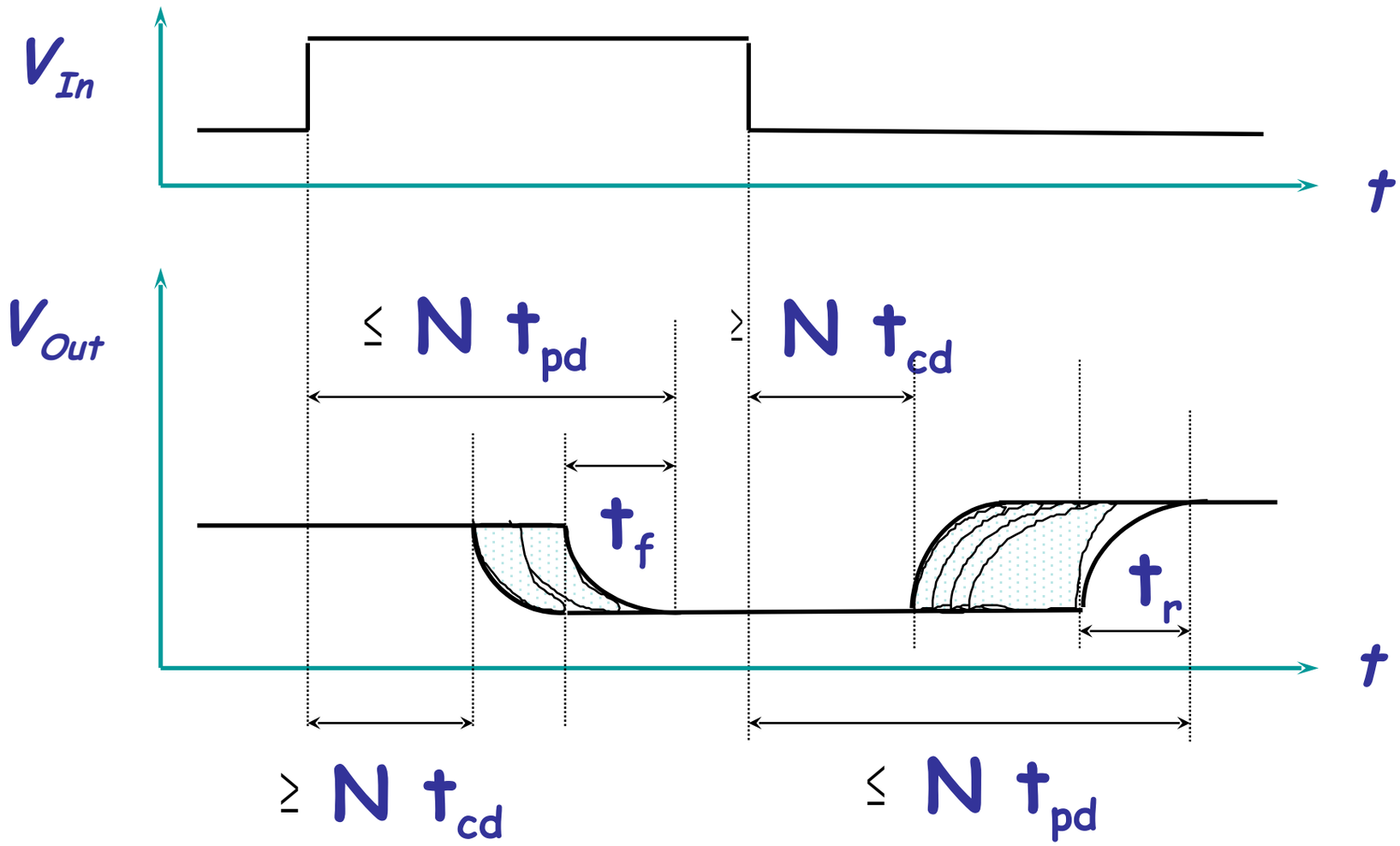
Not quite true, but a convenient assumption!

Note: Final output comes from the last stage of any given acyclic circuit, so rise/fall time measurement comes from there BUT these times ARE STILL affected by the rise/fall times of the preceding stages, don't forget!

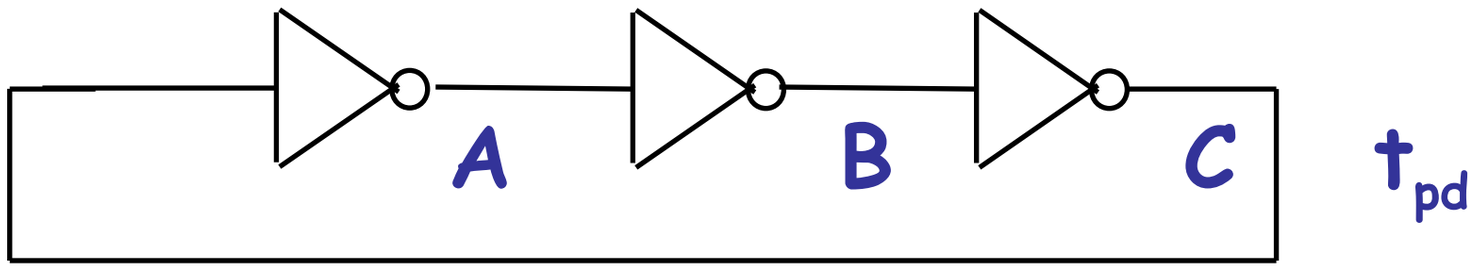
Cascaded Inverters



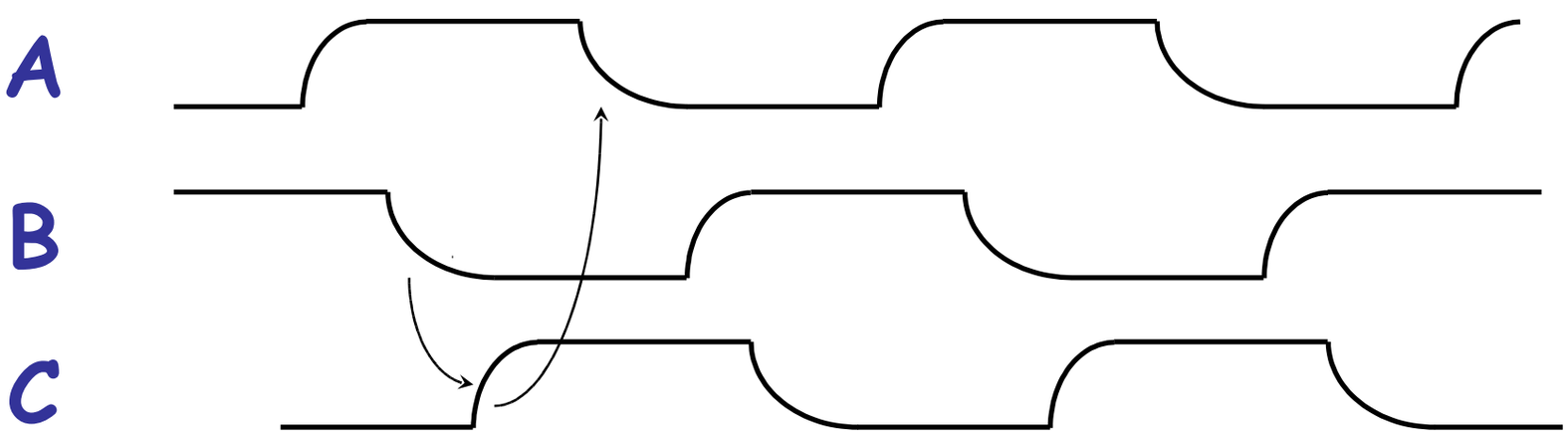
N inverters each with t_{pd} , t_{cd} , t_r , and t_f



Caveat: A Cyclic Circuit

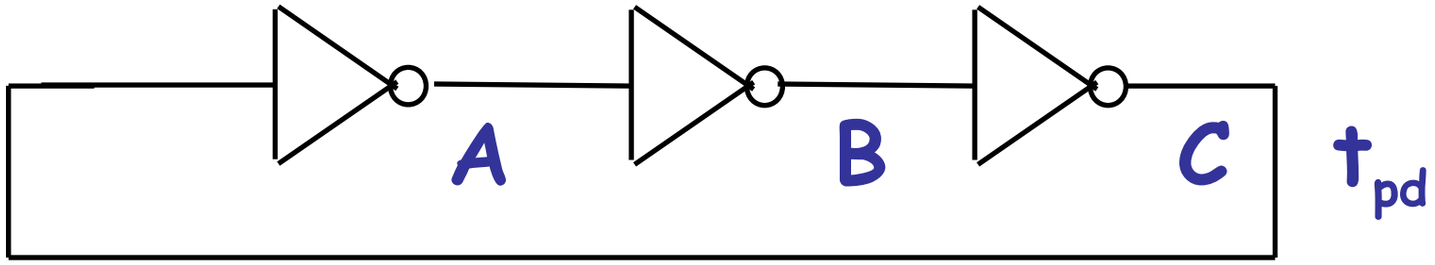


- Not combinational!
- Plausible behavior: Oscillates with period $6t_{pd}$

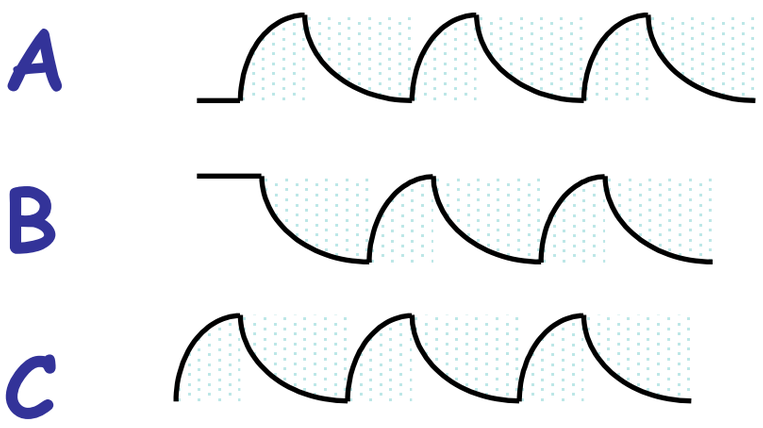


- Is the behavior guaranteed by static discipline?
  Answer: NO!

Other Behaviors

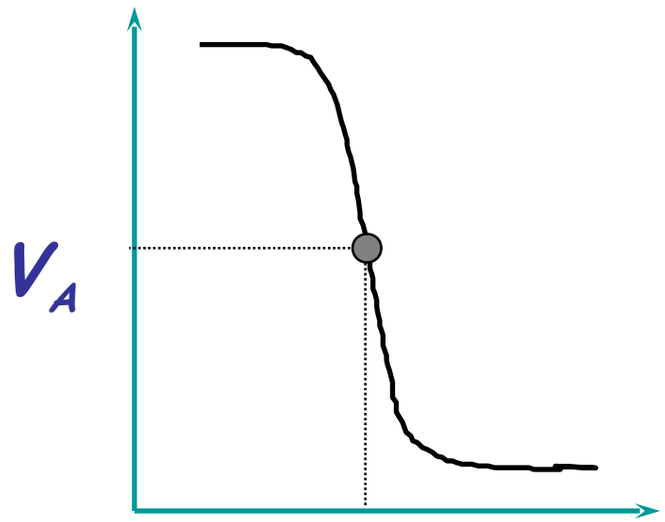


Continuous invalidity?



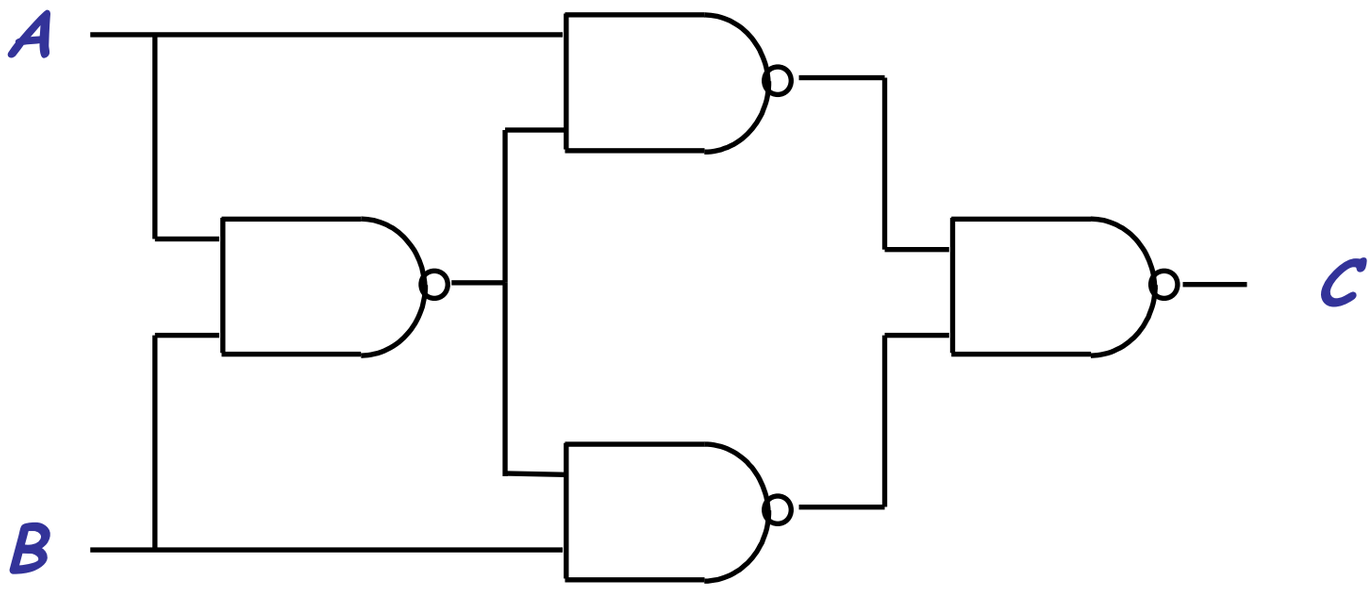
If no signal is ever valid for t_{pd}

Steady DC signal?



$V_A = V_B = V_C$, and are in the forbidden zone!

Delays in Combinational Circuits



Static Discipline: Logically valid A, B ⇨ Logically valid C (after delay)

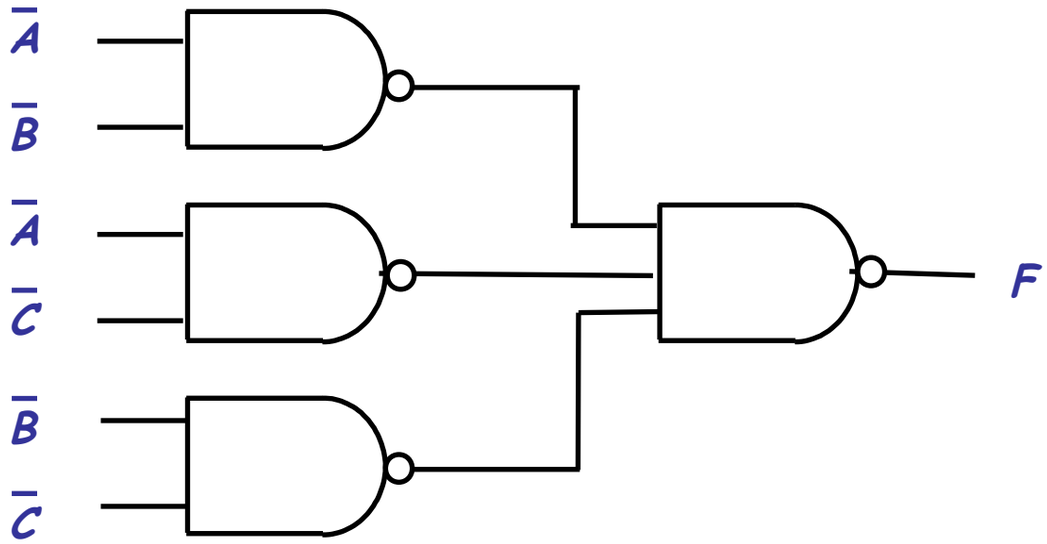
t_{pd} = 5ns and t_{cd} = 1ns for each NAND gate

Propagation delay for circuit is _____
(*Worst* case delay from new valid inputs to new valid outputs)

Contamination delay for circuit is _____
(*Best* case delay from invalid inputs to invalid outputs)

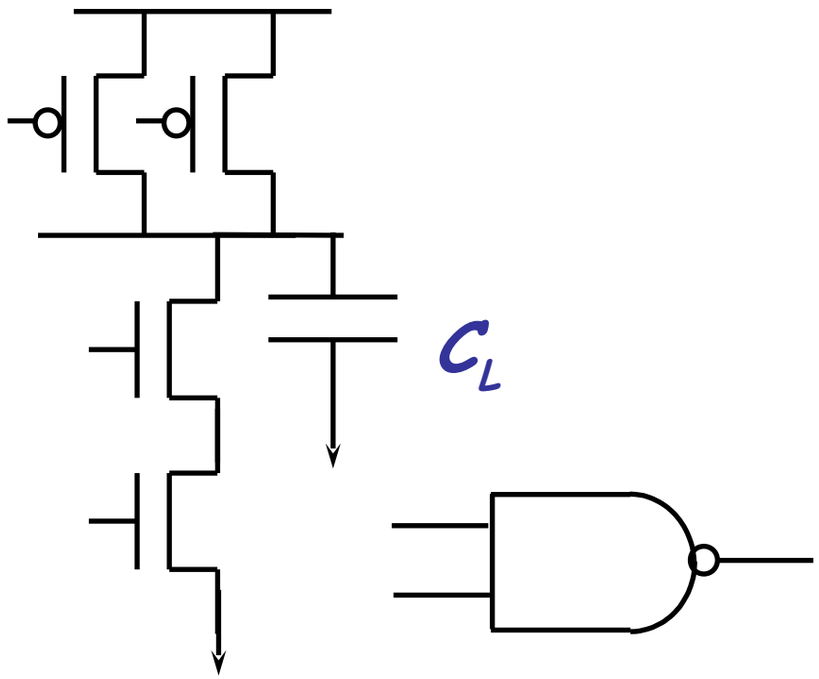
Simplistic Viewpoint

- All we need are two-level circuits!

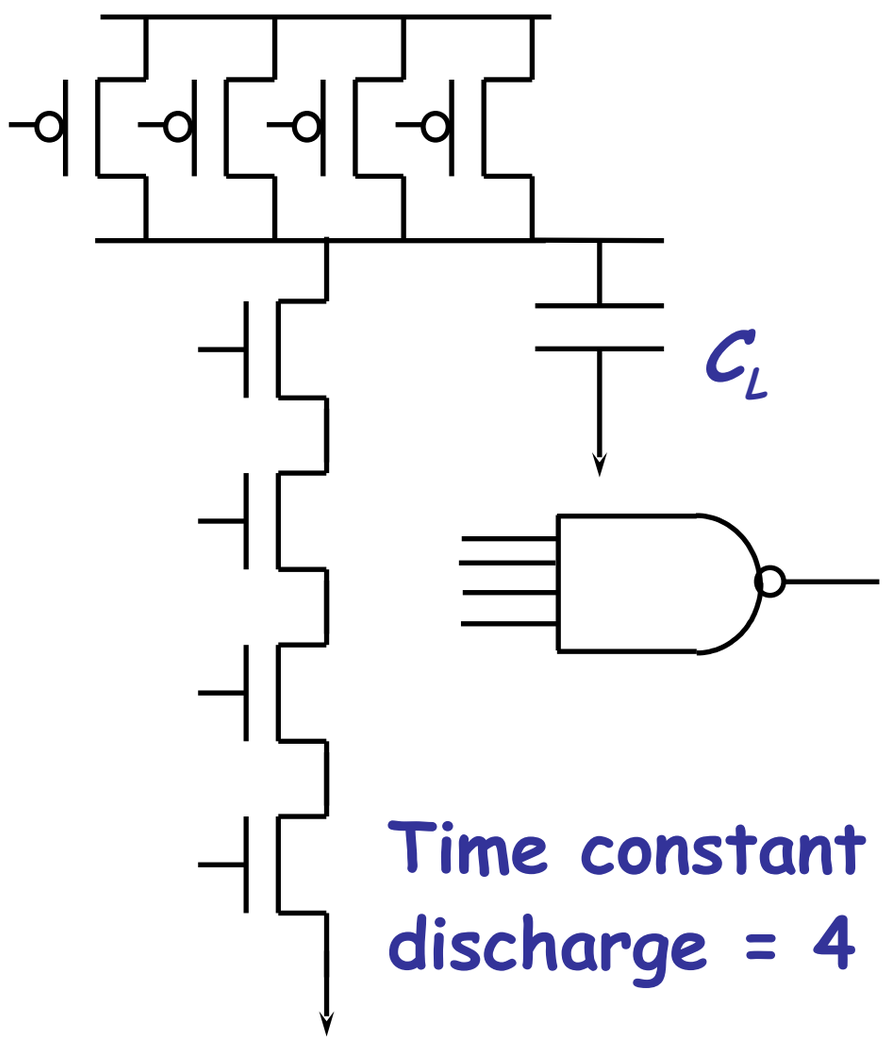


- Can implement all logic functions.
- Two-level circuits are fast because transitions only propagate through two levels of logic (ignoring input inverters).

Circuit Delay: Is Fanin Free?



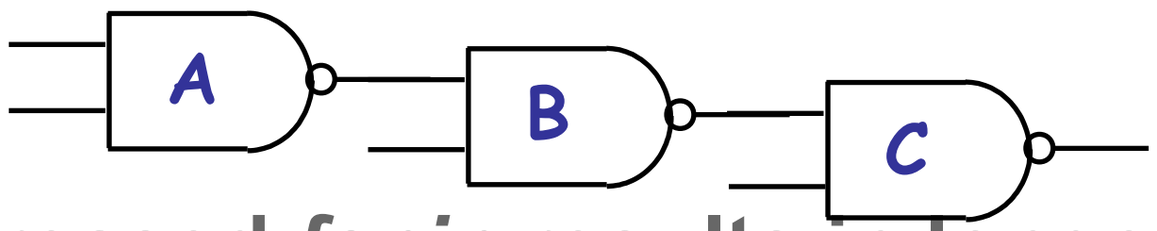
Time constant for C_L
discharge = 2 R_{ON} C_L



Time constant for C_L
discharge = 4 R_{ON} C_L

Circuit Delay: Synopsis

- Propagation delay accumulates with each level.

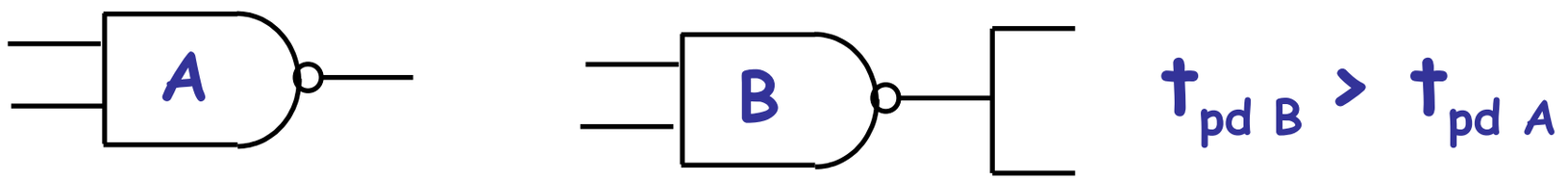


$t_{pd\ A} + t_{pd\ B} + t_{pd\ C}$

- Increased *fanin* results in longer series paths and therefore increased delay.

$t_{pd\ B} > t_{pd\ A}$

- Increased *fanout* results in increased load capacitance and therefore increased delay.



$t_{pd\ B} > t_{pd\ A}$

- Chip makers usually specify max fanout (to guarantee tpd, as well as voltage and current).

How do we specify what we want?

Truth Tables (Can you handle the truth?)

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

____ **rows**

input combinations

Shortcut

Assume unspecified input combinations produce 0.

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

- # of distinct Truth Tables with **k** inputs is _____
- same as # of distinct **k**-input boolean functions

Truth Tables and Logic Equations

- Truth tables define Boolean functions

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

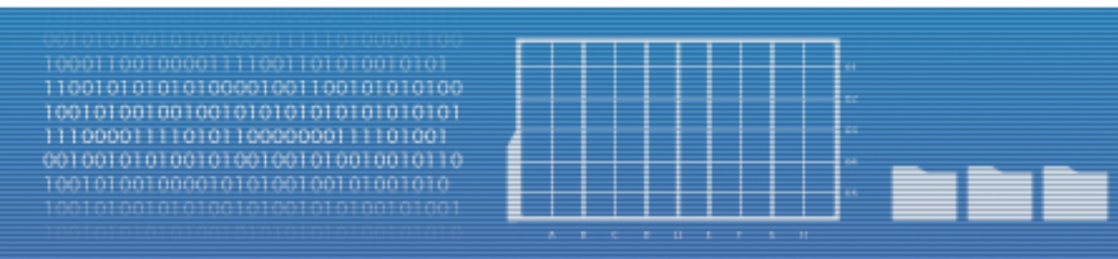
F = 0 otherwise

Write as a logic equation (Boolean function):

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

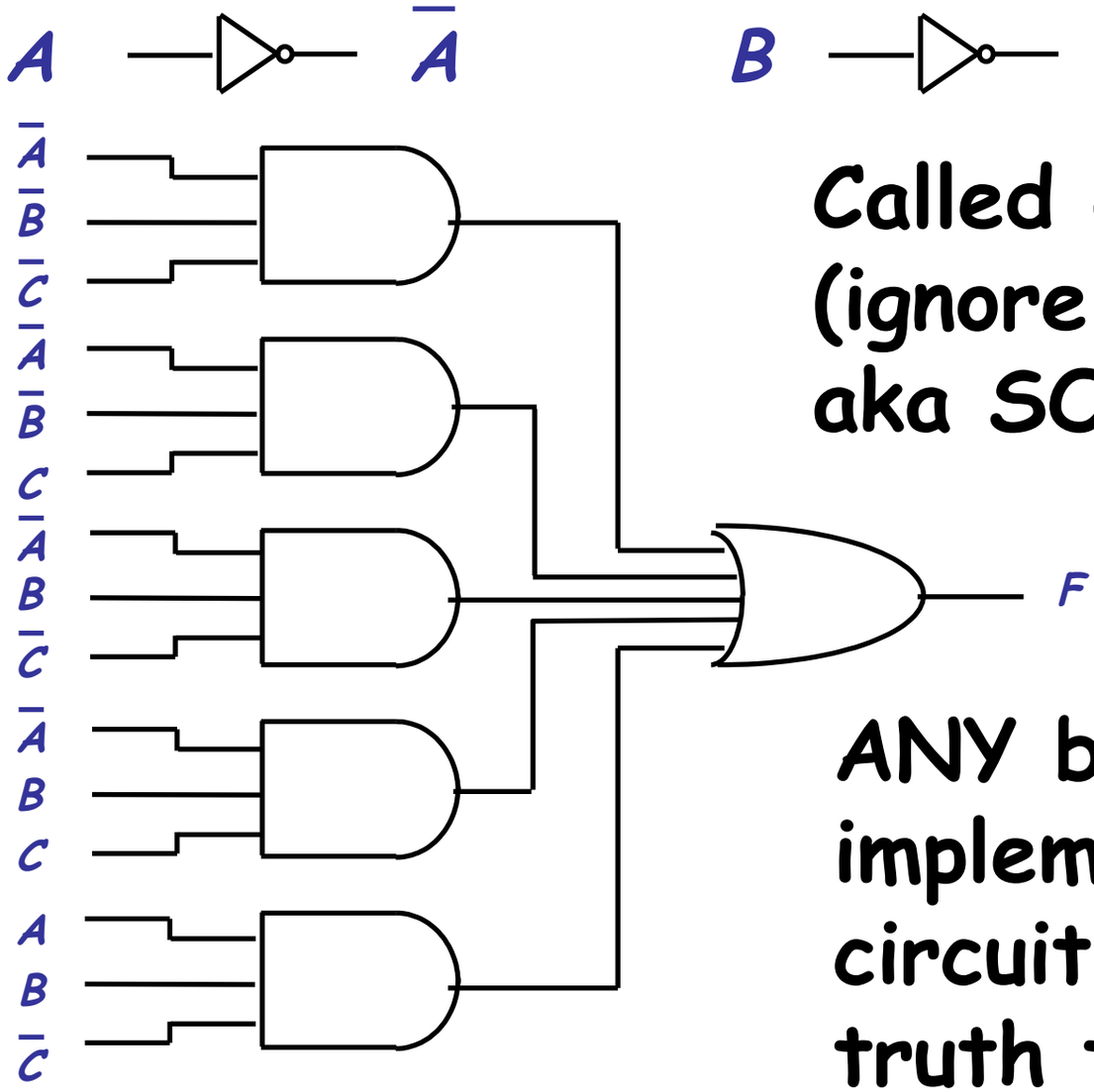
or

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



Logic Equations and AND-OR Circuits

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

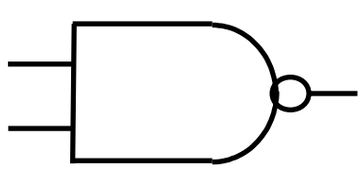


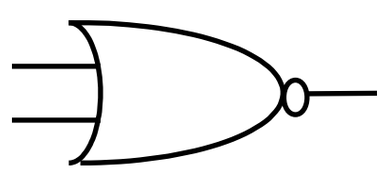
Called a *two-level circuit* (ignore input inverters).
aka SOP - sum of products

ANY boolean function can be implemented as a two-level circuit! (Just implement the truth table.)

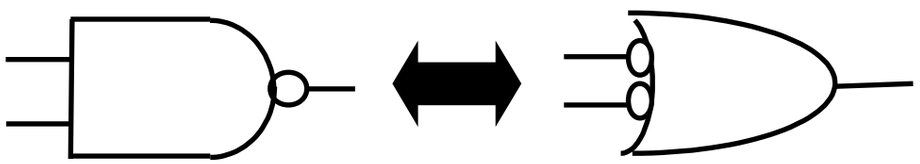
NAND-NAND and NOR-NOR Circuits

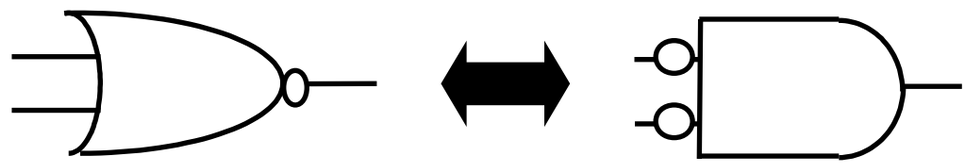
- All circuits can be implemented with NOT, AND, OR!
- In fact, all circuits can be implemented using just NANDs or just NORs!
- Proof:
  - N = NOT





  - De Morgan's Law ("pushing bubbles")
  - Change the gate when pushing bubbles through it.

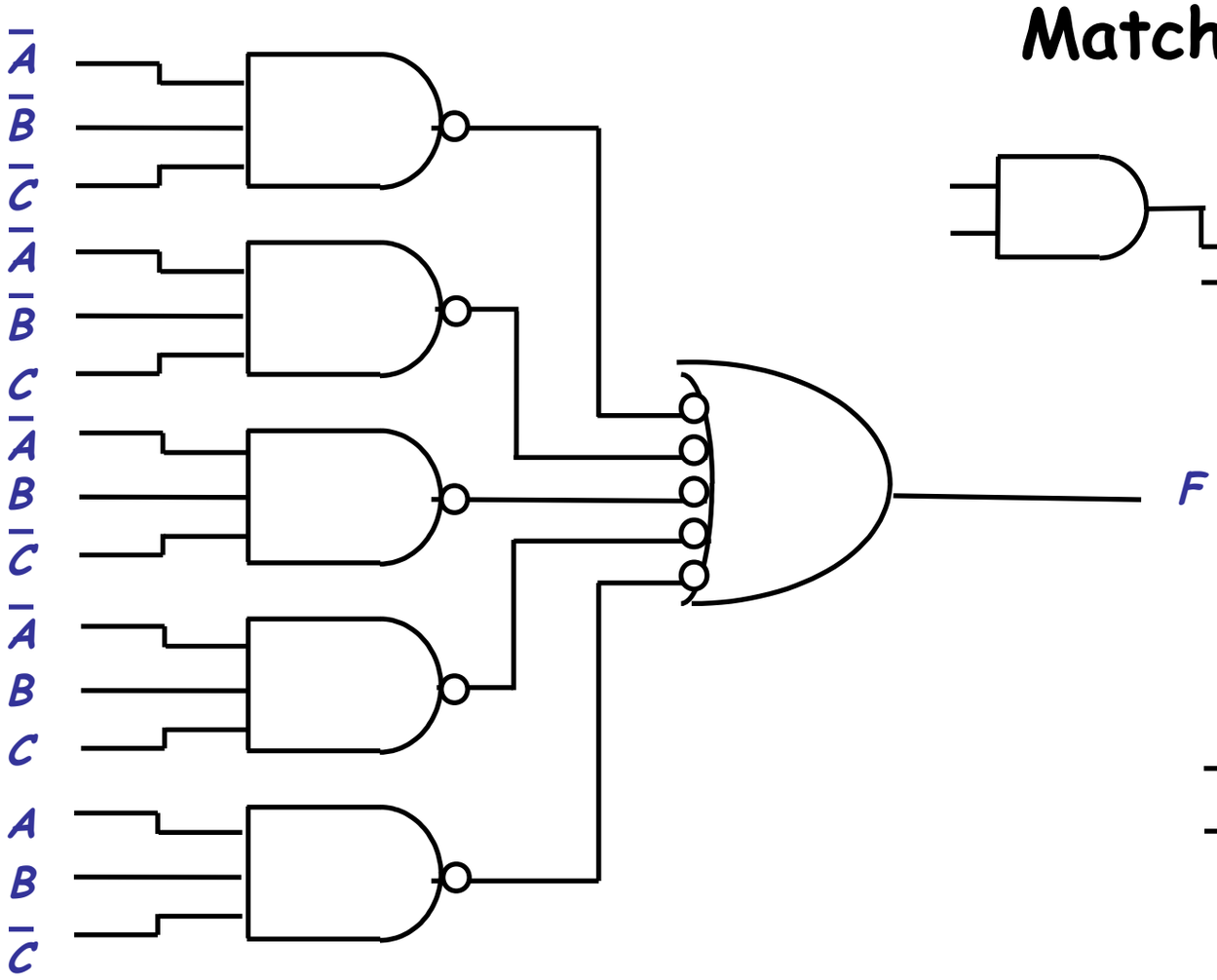




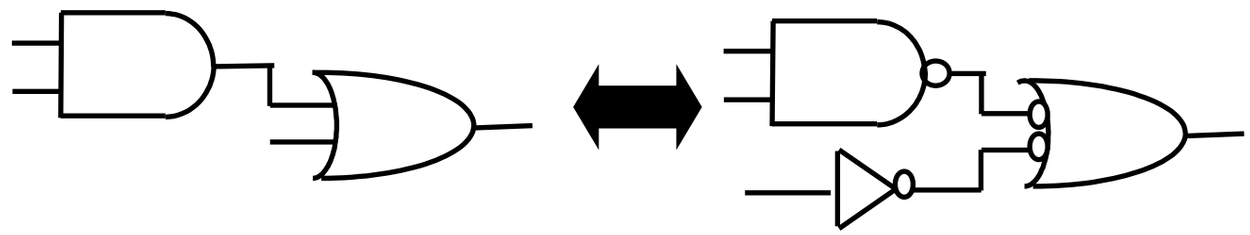
- Can implement AND and OR.
  - NAND is just OR with inverted inputs.
  - NOR is just AND with inverted inputs.

NAND-NAND Circuit

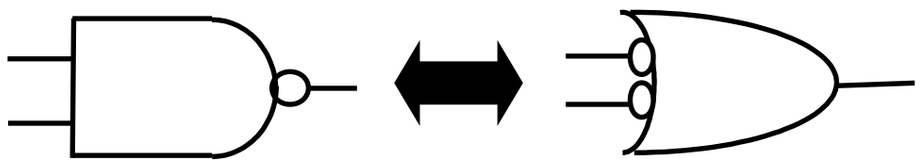
$$F = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,B\,\overline{C}$$



Matching bubbles (negative logic)



DeMorgan's law:



Number of transistors in above circuit (ignoring input inverters):
40 = 5*6 + 1*10 (3-NAND = 6 trans., 5-NAND = 10 trans.)

Simplifying Logic: Review

Given logic functions f, g, h

$$f \cdot f = f \qquad f + f = f$$
$$f \cdot 1 = f \qquad f + 1 = 1$$
$$f \cdot 0 = 0 \qquad f + 0 = f$$
$$f \cdot \overline{f} = 0 \qquad f + \overline{f} = 1$$

Distributive laws

$$f\ (g + h) = f\ g + f\ h$$

$$(f + g)\ (f + h) = f + g\ h$$

Simplifying Logic

Apply identities to simplify equation for F

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

$$= _____ + _____ + A\,B\,\overline{C}$$

$$= _____ + A\,B\,\overline{C}$$

$$= _____$$

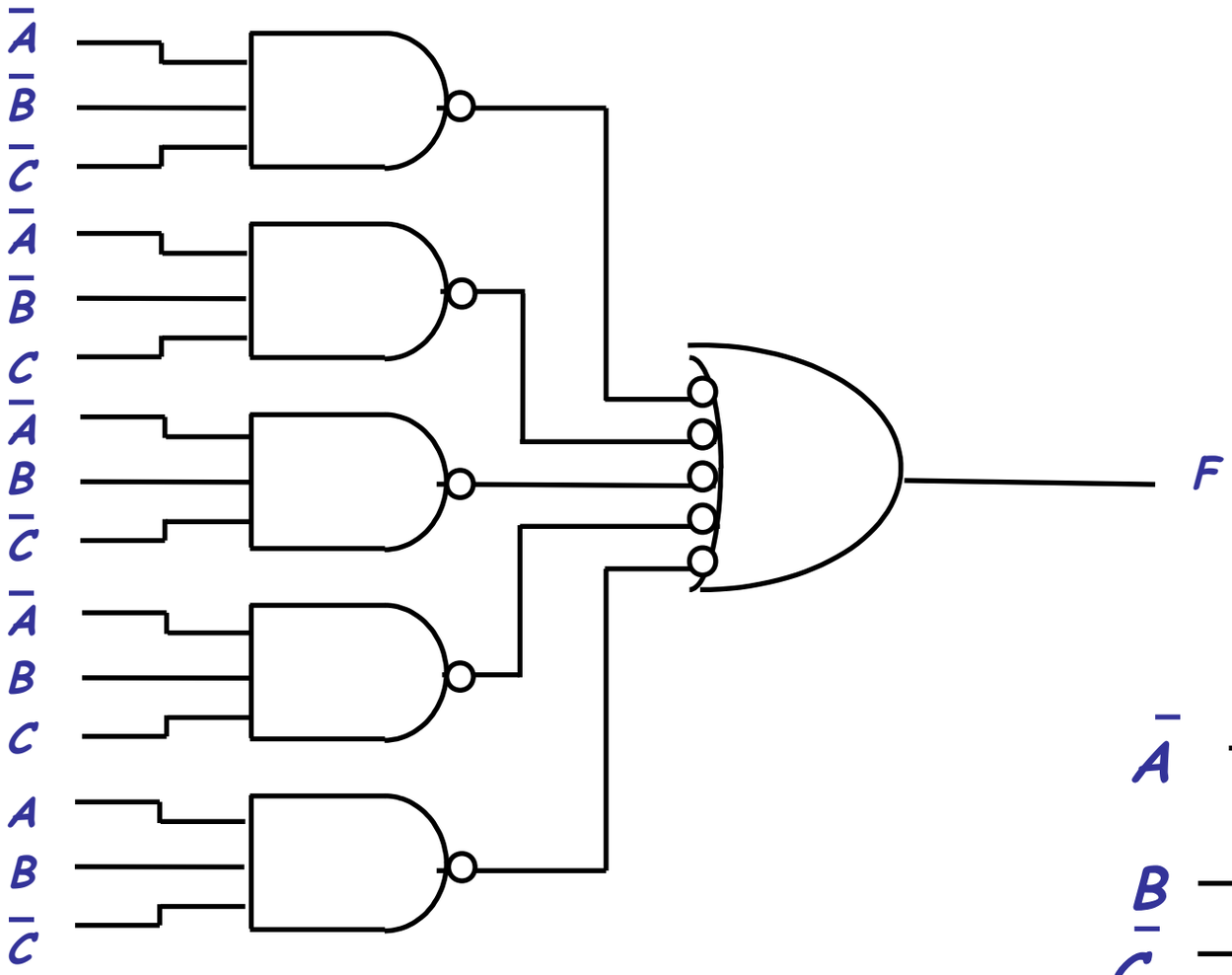
$f \cdot f = f$	$f + f = f$
$f \cdot 1 = f$	$f + 1 = 1$
$f \cdot \underline{0} = 0$	$f + \underline{0} = f$
$f \cdot f = 0$	$f + f = 1$

$$f\,(g + h) = f\,g + f\,h$$
$$(f + g)\,(f + h) = f + g\,h$$

DISCS

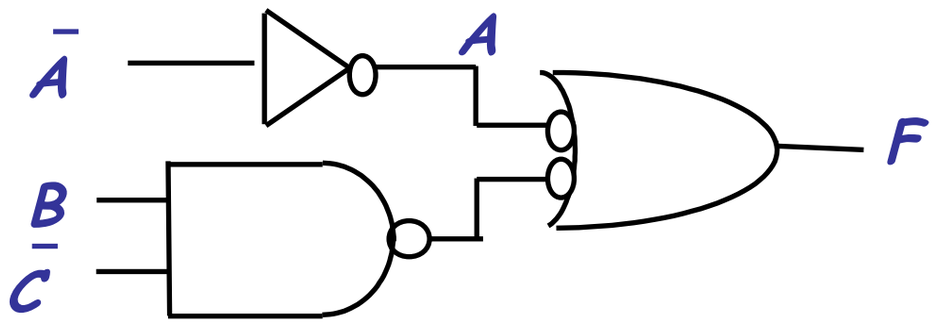
Simplifying Logic: Comparison

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



40 Transistors

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

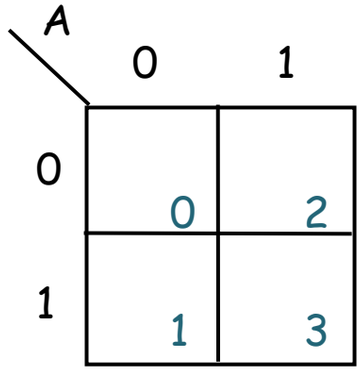


8 Transistors

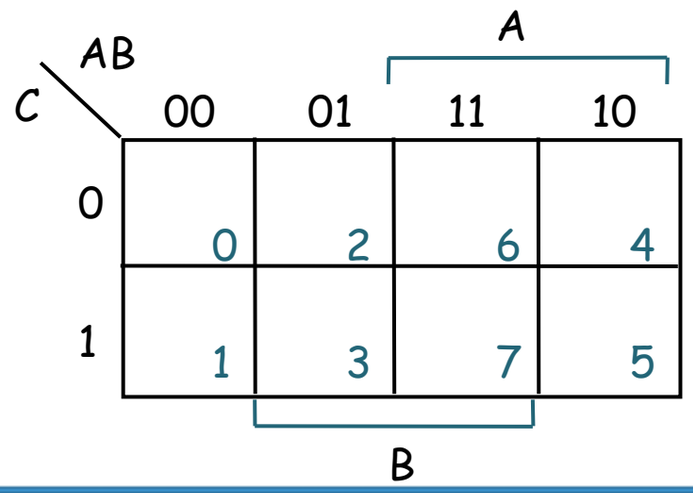
Review: Karnaugh Maps

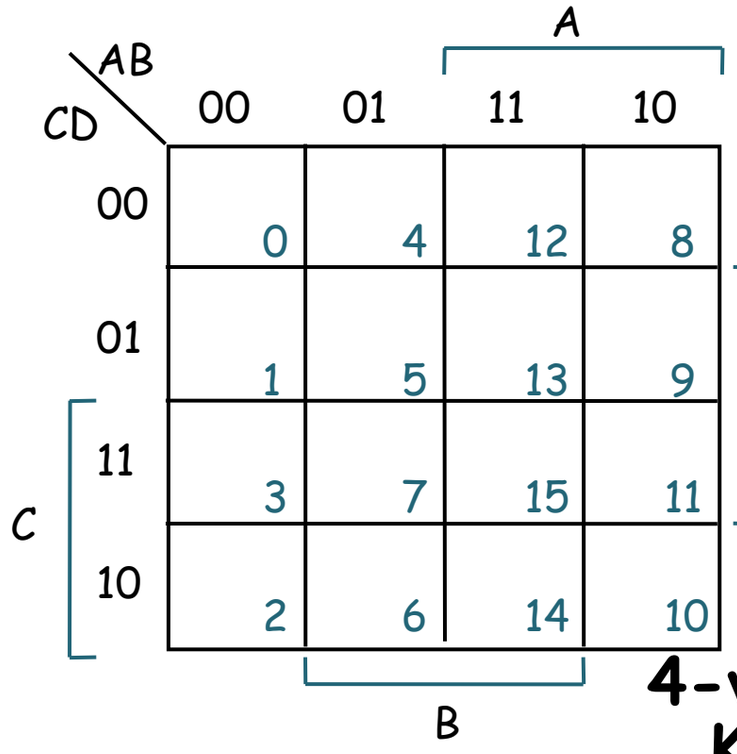
- Graphical representation of truth table.
- Gray code: 00, 01, 11, 10
  - **Only a single bit changes bet. adjacent code words.**
  - **Adjacent boxes on K-map represent AND-terms (products) w/ only 1-bit difference.**

2-variable K-map



3-variable K-map

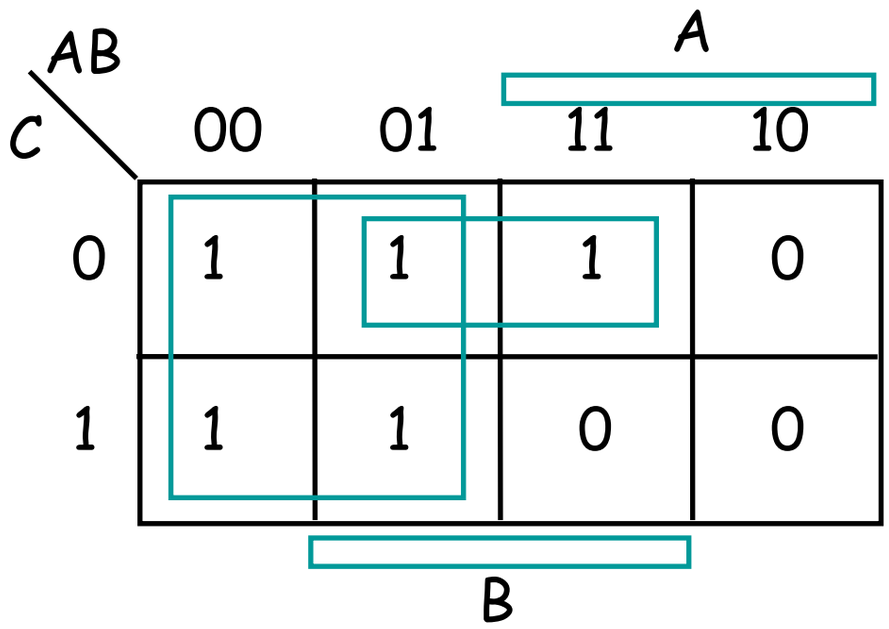




4-variable K-map

Example: 3-variable K-Map

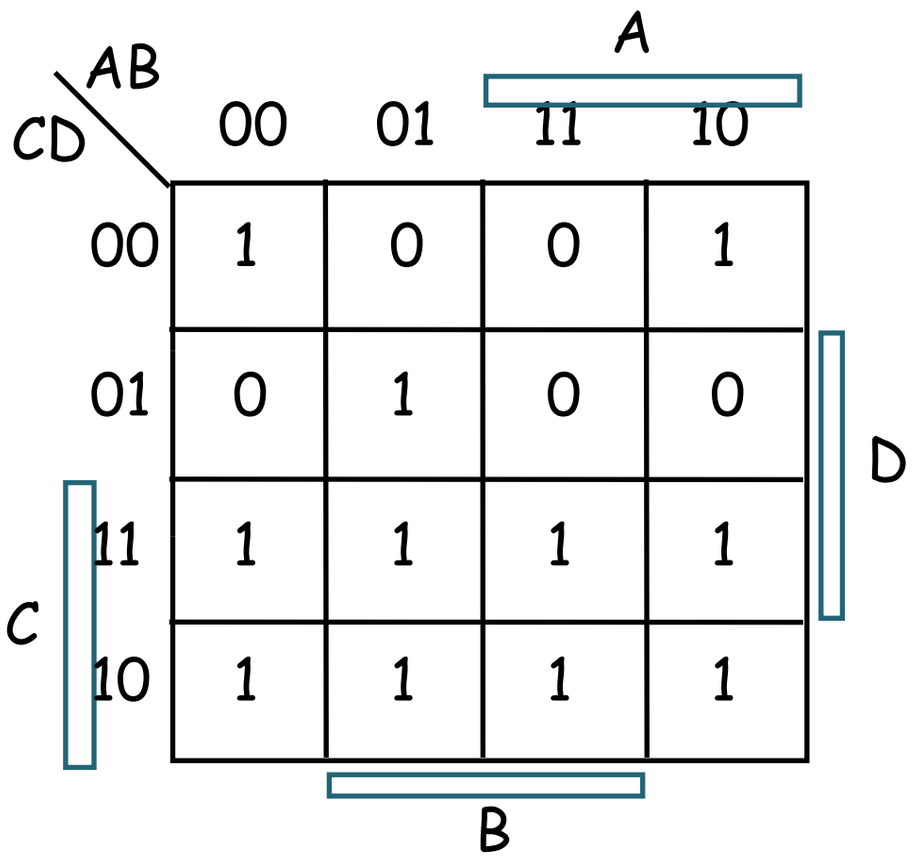
$$F = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,B\,\overline{C}$$



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

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

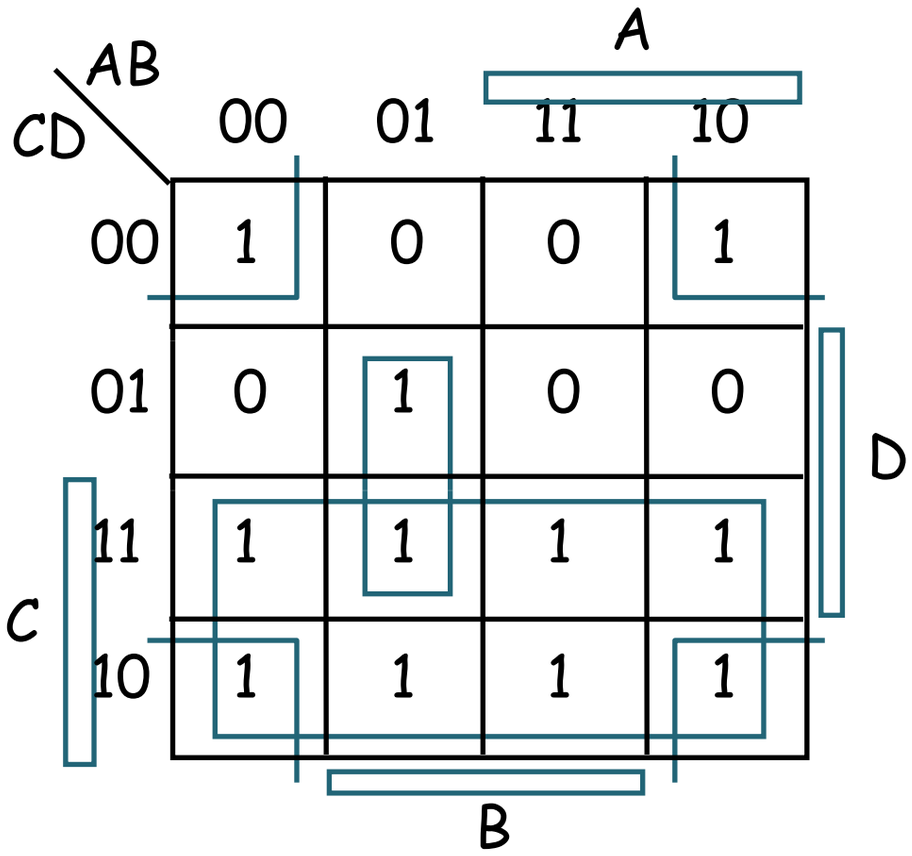
Example: 4-variables



F(A,B,C,D) =

- Adjacencies wrap-around left-right, top-bottom, and around corners.
- Find the smallest number of the largest possible squares that cover the ON-set.

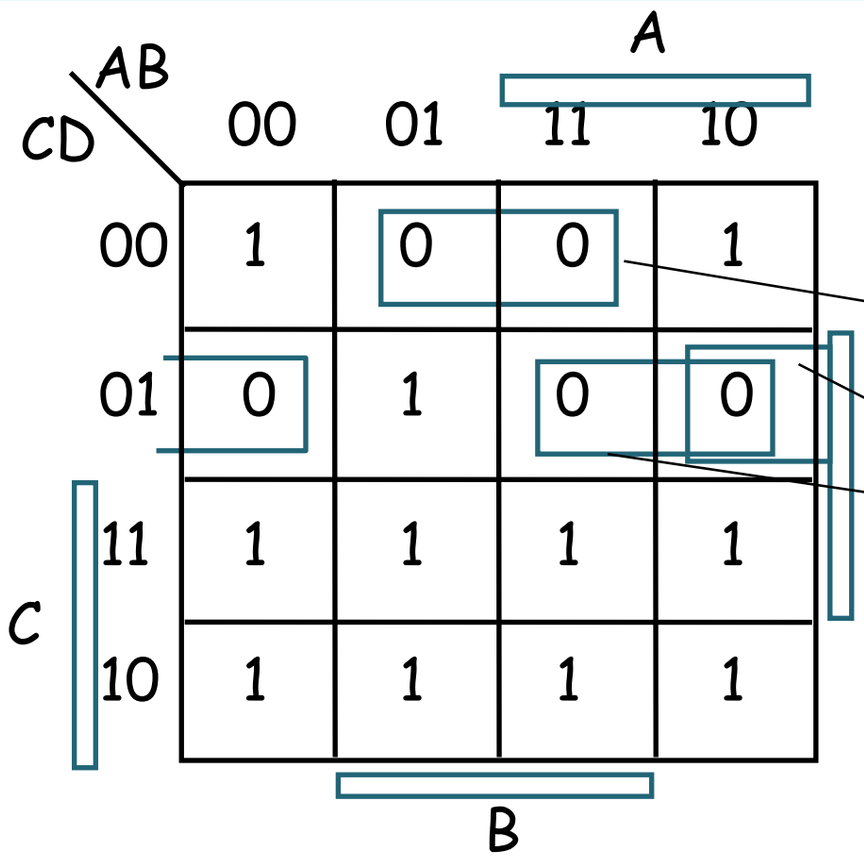
Example: 4-variables



$$F(A,B,C,D) = C + \overline{A}\,B\,D + \overline{B}\,\overline{D}$$

- Adjacencies wrap-around left-right, top-bottom, and around corners.
- Find the smallest number of the largest possible squares that cover the ON-set.

Product-of-Sums form



$$F = (\overline{B} + C + D)\,(\overline{A} + C + \overline{D})\,(B + C + \overline{D})$$

- Circle 0's instead.
- Flip +'s and •'s.
- Flip (invert) variables.

Side note: using DeMorgan's Law

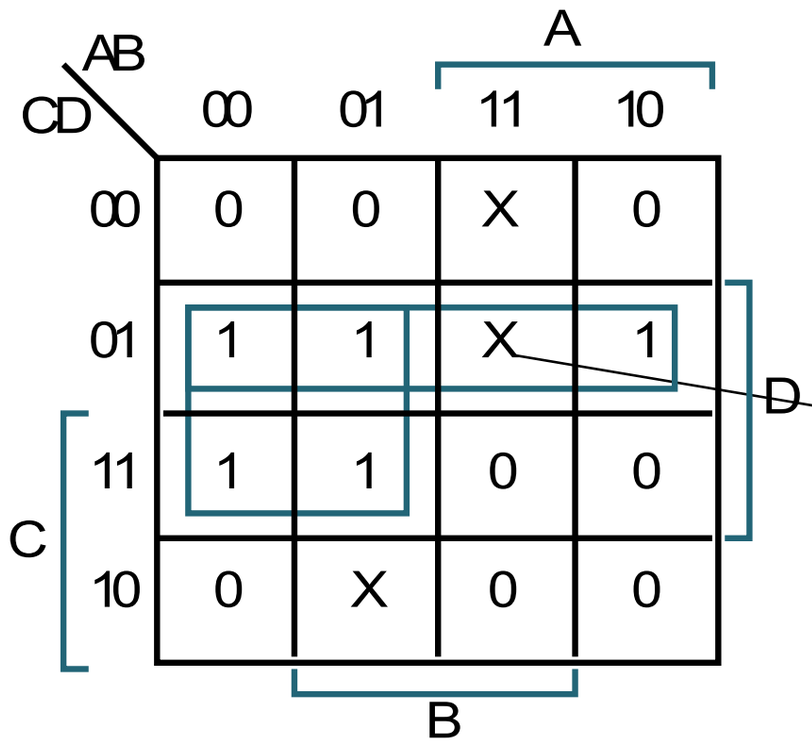
$$\overline{F} = B\,\overline{C}\,\overline{D} + A\,\overline{C}\,D + B\,C\,D$$

$$\overline{\overline{F}} = \overline{B\,\overline{C}\,\overline{D} + A\,\overline{C}\,D + B\,C\,D}$$

$$F = (\overline{B} + C + D)\,(\overline{A} + C + \overline{D})\,(B + C + \overline{D})$$

Don't Cares

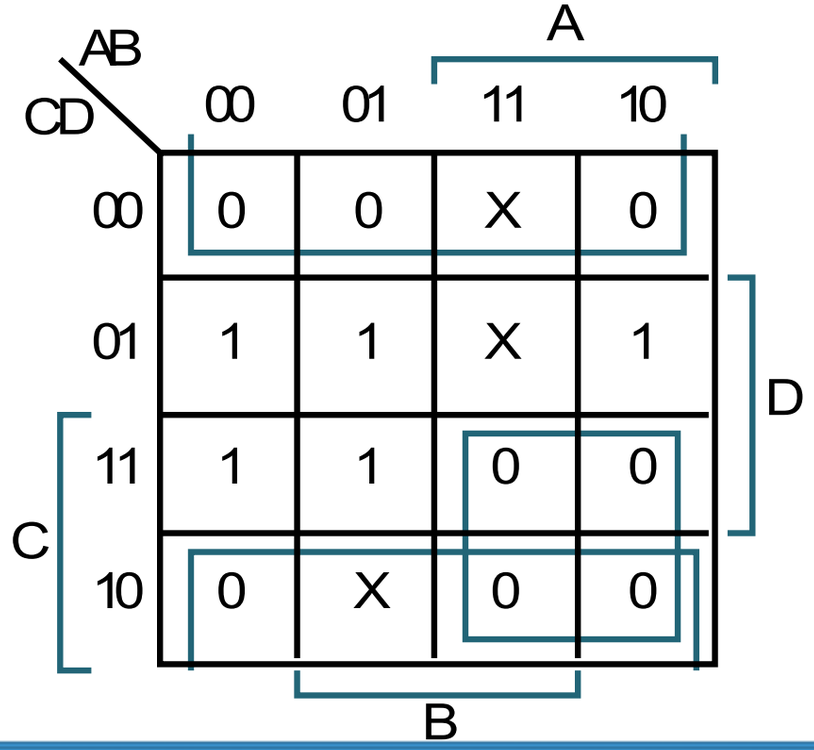
Don't Cares can be treated as 1's or 0's if it is advantageous to do so



F = A'D + B' C' D w/o don't cares

F = A' D + C' D w/ don't cares

By treating this X as a "1", we save gates and literals



In PoS form: F = D (A' + C')

Same answer as above, but less literals.

(Note: don't cares are treated differently as convenient.)

Another example: BCD +1

BCD = "Binary Coded Decimal"

A B C D	W X Y Z
0 0 0 0	0 0 0 1
0 0 0 1	0 0 1 0
0 0 1 0	0 0 1 1
0 0 1 1	0 1 0 0
0 1 0 0	0 1 0 1
0 1 0 1	0 1 1 0
0 1 1 0	0 1 1 1
0 1 1 1	1 0 0 0
1 0 0 0	1 0 0 1
1 0 0 1	0 0 0 0
1 0 1 0	x x x x
1 0 1 1	x x x x
1 1 0 0	x x x x
1 1 0 1	x x x x
1 1 1 0	x x x x
1 1 1 1	x x x x

only care about 0 to 9

W

CD \ AB	00	01	11	10
00	0	0	X	1
01	0	0	X	0
11	0	1	X	X
10	0	0	X	X

X

CD \ AB	00	01	11	10
00	0	1	X	0
01	0	1	X	0
11	1	0	X	X
10	0	1	X	X

Y

CD \ AB	00	01	11	10
00	0	0	X	0
01	1	1	X	0
11	0	0	X	X
10	1	1	X	X

Z

CD \ AB	00	01	11	10
00	1	1	X	1
01	0	0	X	0
11	0	0	X	X
10	1	1	X	X

W = ? X = ? Y = ? Z = ?