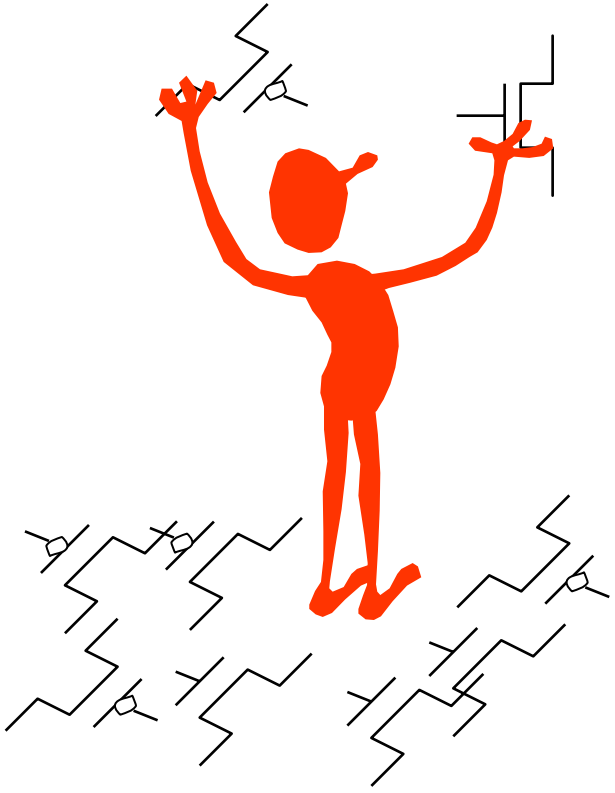
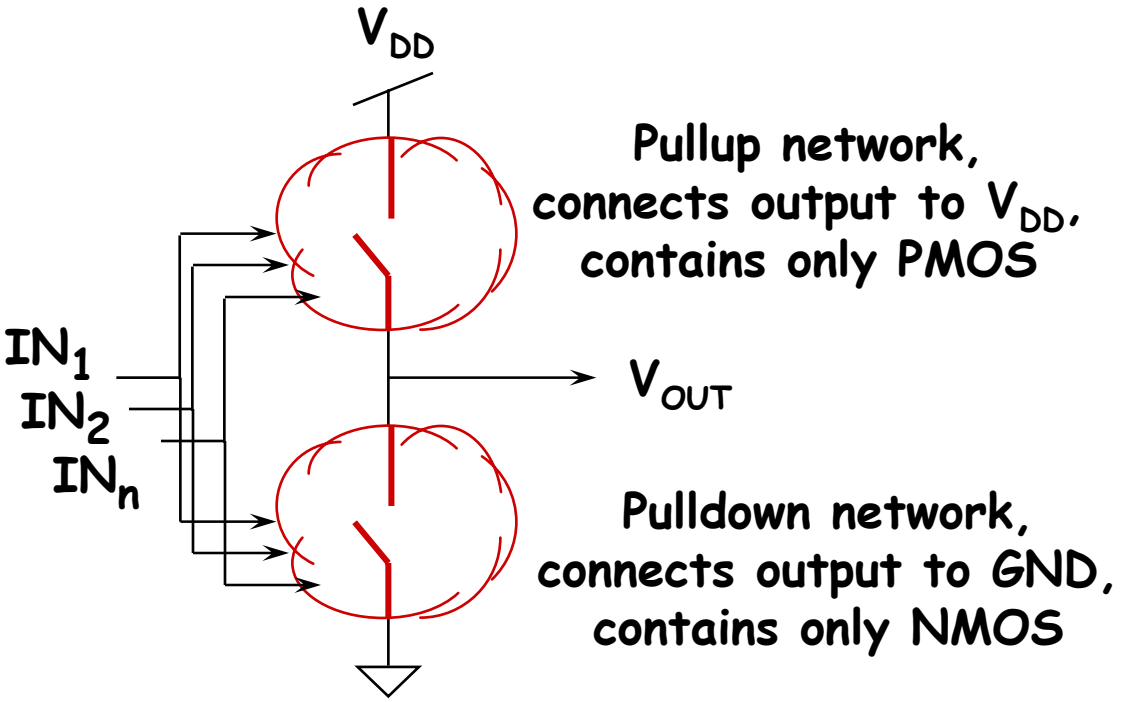
# Static CMOS Gates

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

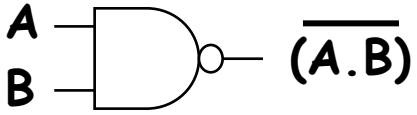


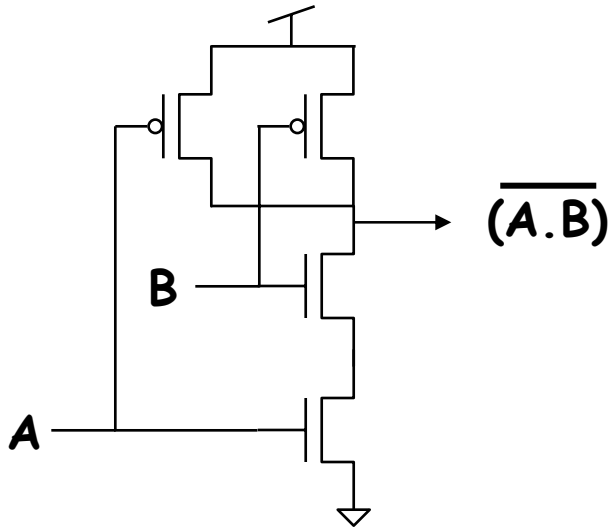
# Generic Static CMOS Gate




- For every set of input logic values, either pullup or pulldown network makes connection to VDD or GND
  - If both connected, power rails would be shorted together
  - If neither connected, output would float (tristate logic)

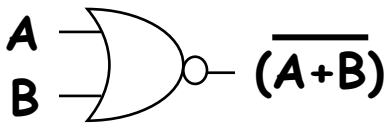
# NAND Gate

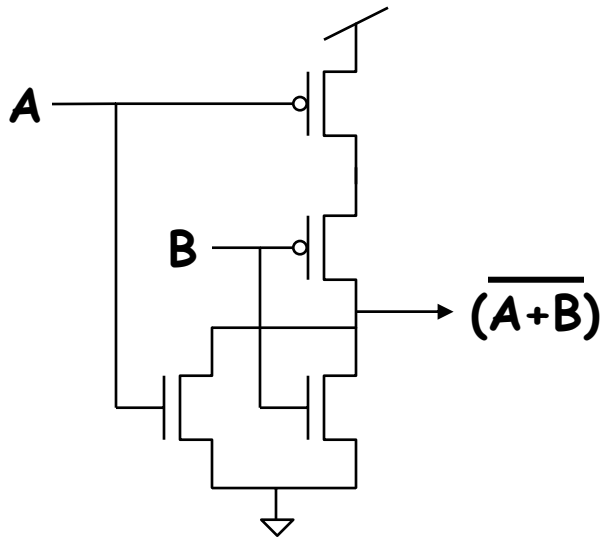



- When both A and B are high, output is low
- When either A or B is low, output is high

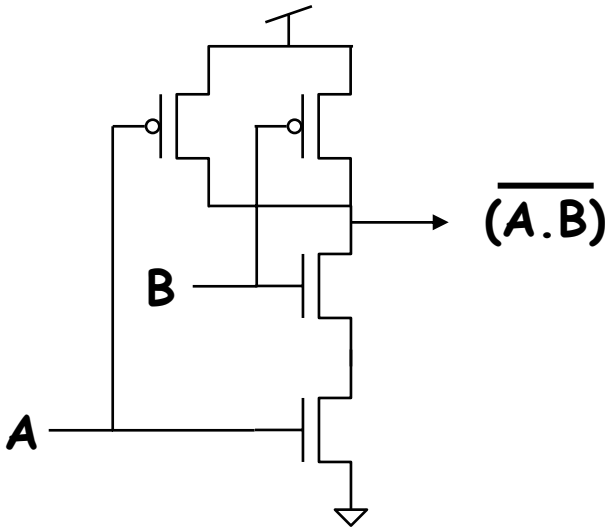
# NOR Gate




- When both A and B are low, output is high
- When either A or B is high, output is low

# NAND Gate Layout



# Methodical Gate Building

- Goal is to create a logic function $f(x_1, x_2, \ldots)$
  - must be inverting for single level of CMOS logic
- Pull up network should connect output to $V_{DD}$ when $f(x_1, x_2, \ldots) = 1$
- Pull down network should connect output to GND when $\overline{f}(x_1, x_2, \ldots) = 1$
- Because PMOS is conducting with low inputs, useful to write pullup as function of inverted inputs

$$p(\overline{x1}, \overline{x2}, \ldots) = f(x_1, x_2, \ldots)$$

# Pullup is Dual of Pulldown Network

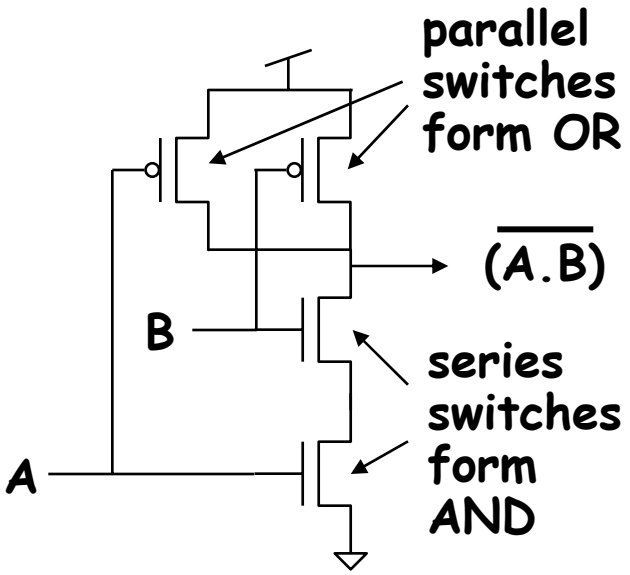
For NAND gate, $f=\overline{(A.B)}$

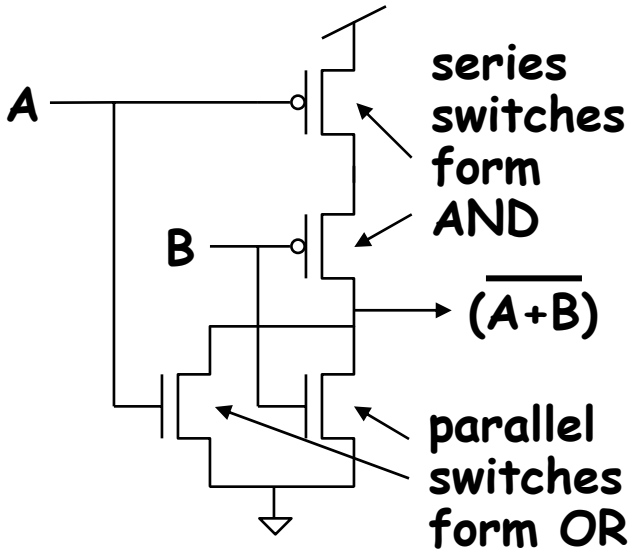
- Pulldown $\overline{f} = A.B$
- Pullup $p = f = \overline{A.B}$
  $= \overline{A}+\overline{B}$

*(De Morgan's Laws)*

For NOR gate, $f=\overline{(A+B)}$

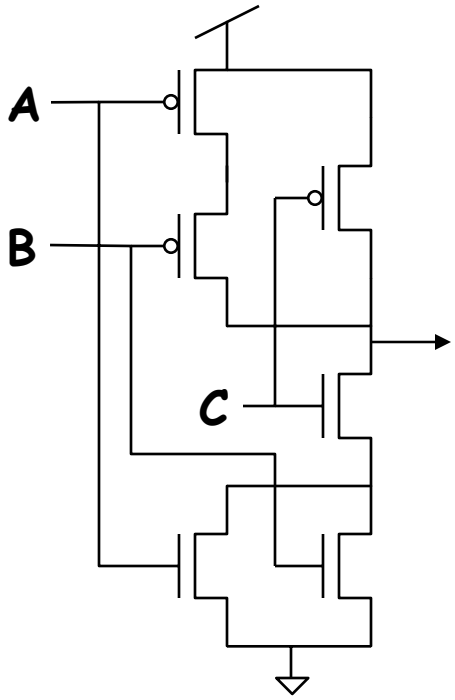
- Pulldown $\overline{f} = A+B$
- Pullup $p = f = \overline{A+B}$
  $= \overline{A}.\overline{B}$



# More Complex Example

$$f = \overline{(A+B).C}$$




pullup $p = \overline{(A+B).C}$

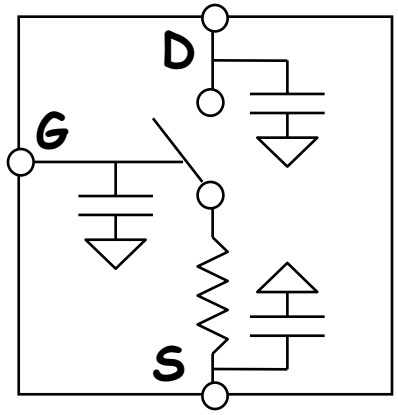
$= \overline{(\overline{A+B})+\overline{C}}$

$= (\overline{A}.\overline{B})+\overline{C}$

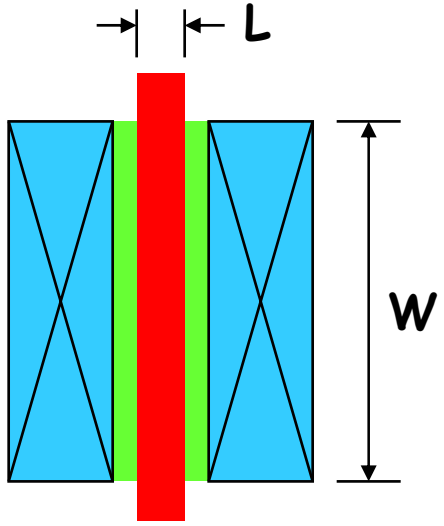
pulldown $\overline{f} = (A+B).C$

# Transistor R and C

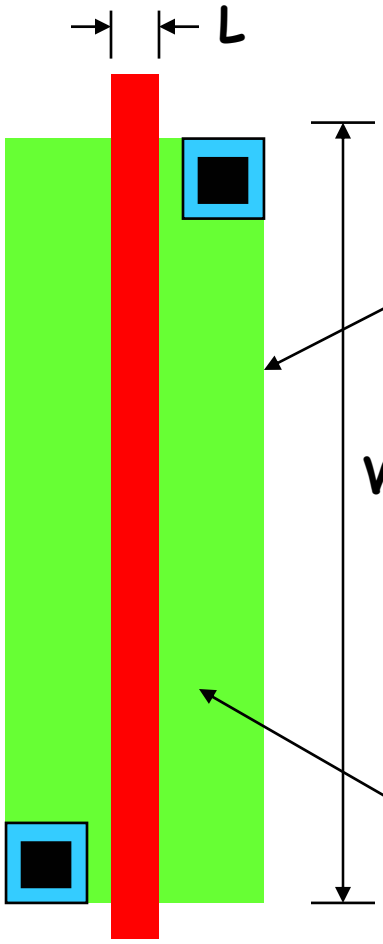
Simple Equivalent RC Model of Transistor




- Nearly all transistors in digital CMOS circuits have minimum L
  - but might use slightly longer L to cut leakage in parts of modern circuits
- Can scale transistor R and C parameters by width W
- Effective R scales linearly with 1/W
  - ~4kΩμm NMOS, ~9kΩμm PMOS, in 0.25μm technology
- Gate capacitance scales linearly with W
  - ~2fF/μm
- Diffusion capacitance scales linearly with W
  - sum contributions from perimeter and area, ~2fF/μm



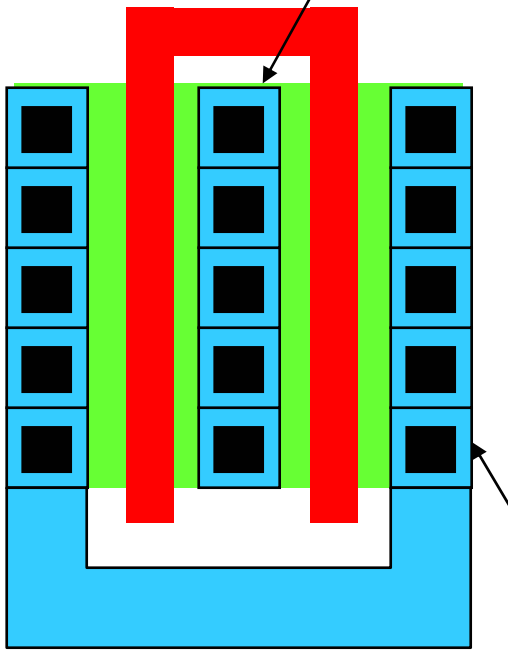
# Layout Tips




Long transistors don't lay out well, and have a lot of parasitic diffusion capacitance

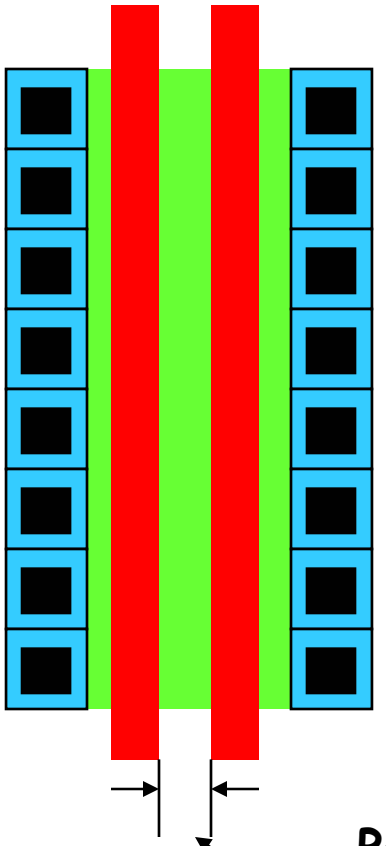
Diffusion has high resistance

"Fold" transistor to reduce perimeter diffusion cap

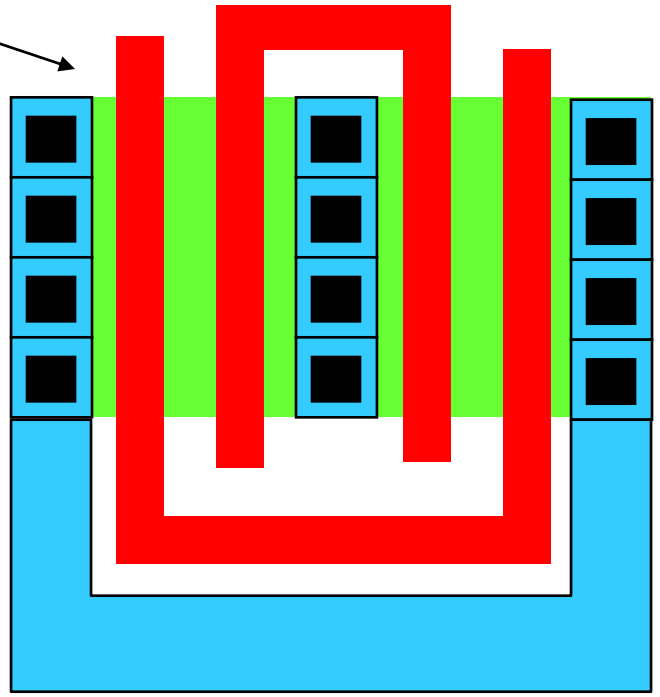



Use multiple contacts to diffusion to reduce resistance

# Layout Tips - Stacks



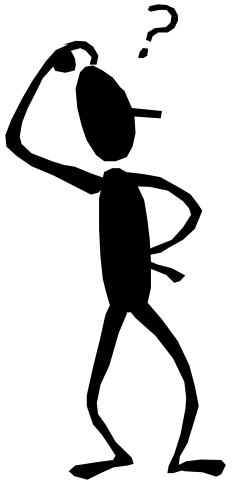
Fold whole stack, not individual transistors



Pack series transistors as close as possible to minimize diffusion R&C parasitics

# Even More Complex Gates?

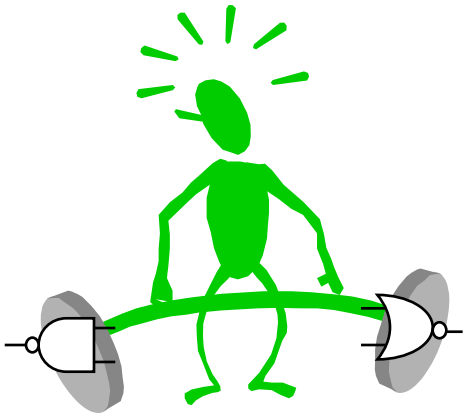
- Can build arbitrarily complex logic function into one gate, e.g.



$$F = \overline{(A+B).(C+D).E.G+H.(J+K)}$$

  - But don't want to:
    - Usually less total delay using a few smaller logic gates rather than one large complex gate
    - Only want to design and characterize a small library of gates
  - What's the best way to implement a given logic function?

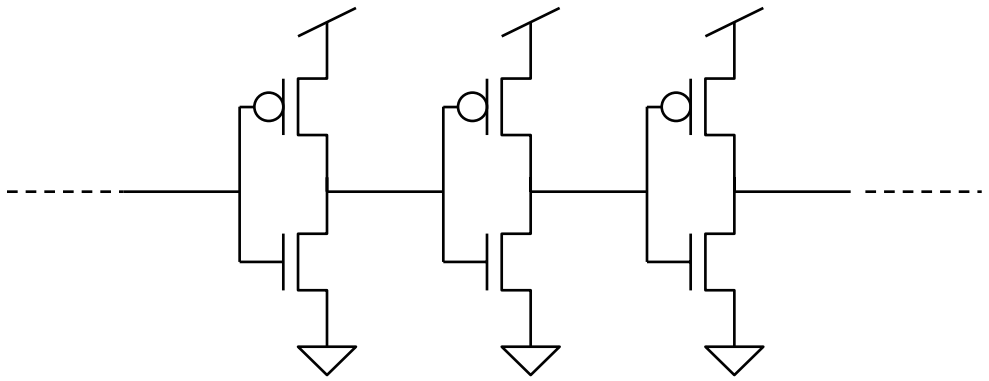
# Method of Logical Effort
## (Sutherland and Sproul)



- Easy way to estimate delays in CMOS process
- Indicates correct number of logic stages and transistor sizes
- Based on simple RC approximations
- Useful for back-of-the-envelope circuit design

# Technology Speed Parameter: $\tau$

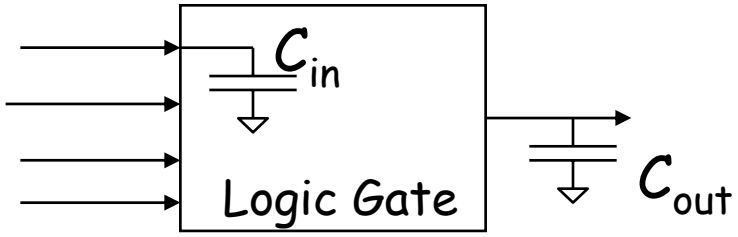
- Characterize process speed with single delay parameter: $\tau$
- $\tau$ is delay of inverter driving same-sized inverter, with no parasitics other than gate



$\tau$ ~16-20ps for 0.25$\mu$m process

# Gate Delay Components

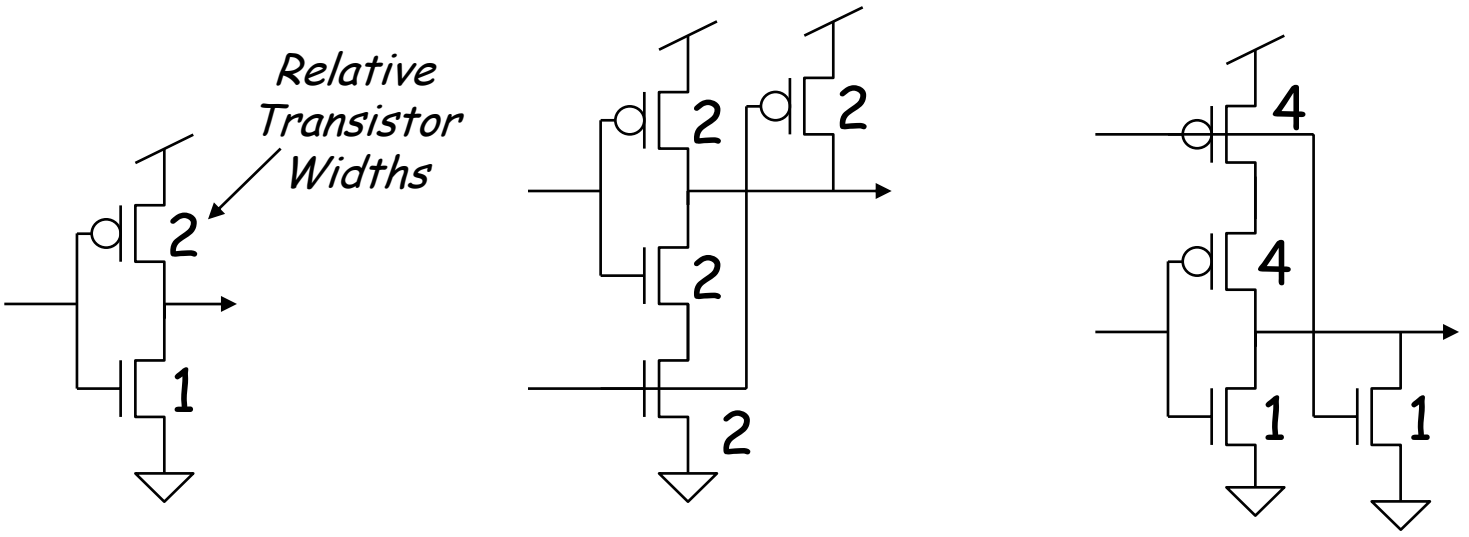
**Delay = Logical Effort x Electrical Effort + Parasitic Delay**




- **Logical Effort**
  - Complexity of logic function (Invert, NAND, NOR, etc)
  - Define inverter has logical effort = 1
  - Depends only on topology not transistor sizing
- **Electrical Effort**
  - Ratio of output capacitance to input capacitance $C_{out}/C_{in}$
- **Parasitic Delay**
  - Intrinsic self-loading of gate
  - Independent of transistor sizes and output load

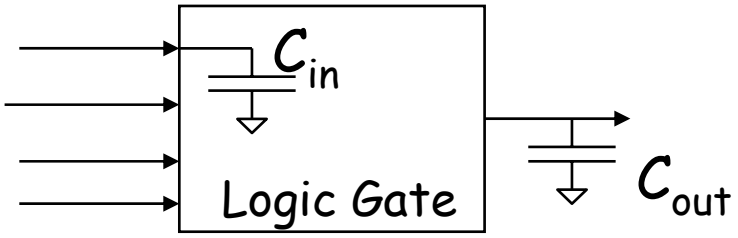
# Logical Effort for Simple Gates

- Define Logical Effort of Inverter = 1
- For other gates, Logical Effort is ratio of logic gate's input cap. to inverter's input cap., when gate sized to give same current drive as inverter




| Inverter | NAND | NOR |
|---|---|---|
| Input Cap = 3 units | Input Cap = 4 units | Input Cap = 5 units |
| L.E.=1 (definition) | L.E.=4/3 | L.E.=5/3 |

# Electrical Effort

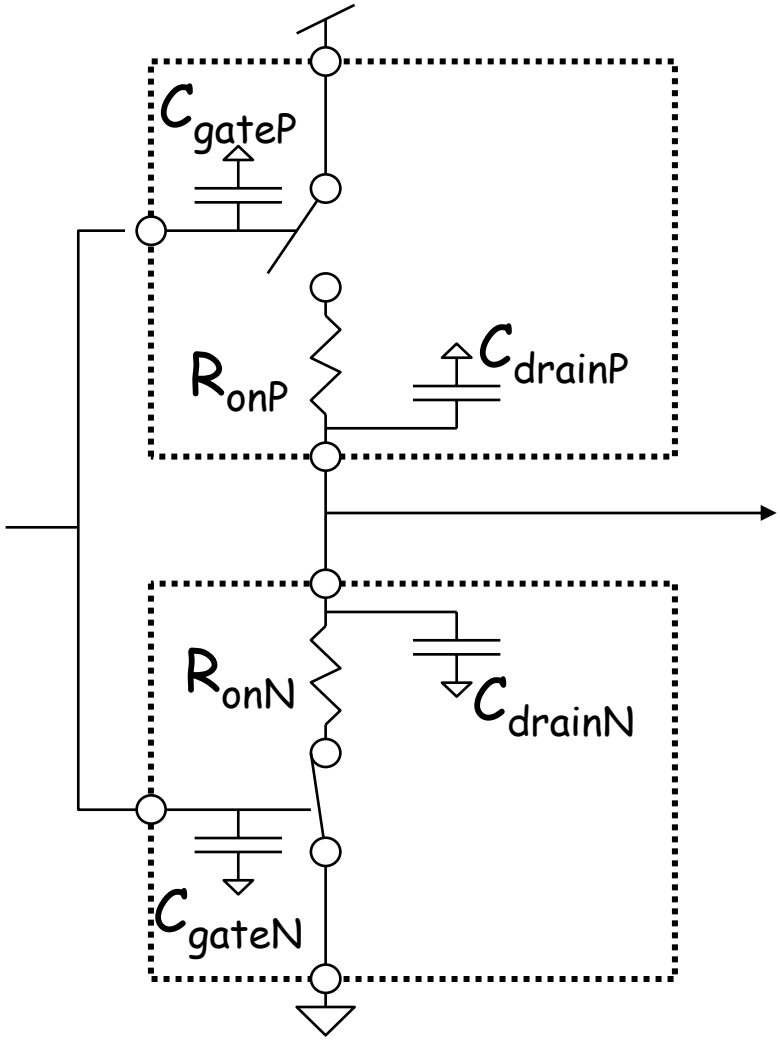



- Ratio of output load capacitance over input capacitance:

$$\text{Electrical Effort} = C_{out}/C_{in}$$

- Usually, transistors have minimum length
- Input and output capacitances can be measured in units of transistor gate widths

# Parasitic Delay



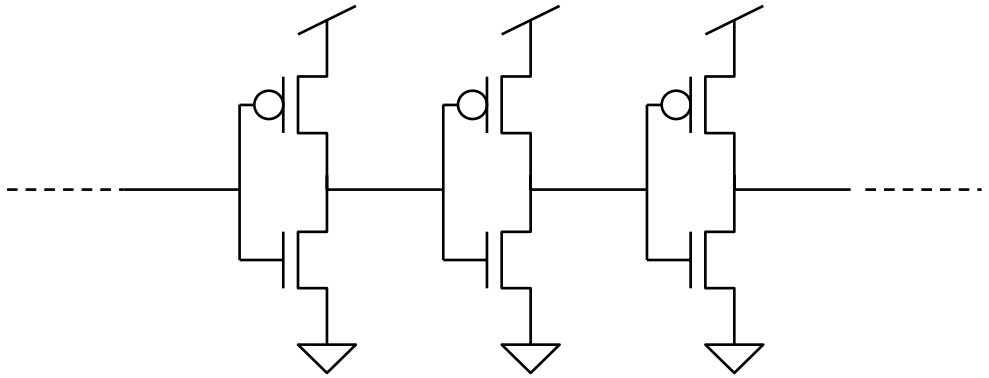

- Main cause is drain capacitances
- These scale with transistor width so P.D. independent of transistor sizes
- Useful approximation:

$$C_{gate} \sim= C_{drain}$$

- For inverter:

Parasitic Delay ~= 1.0 $\tau$

# Inverter Chain Delay



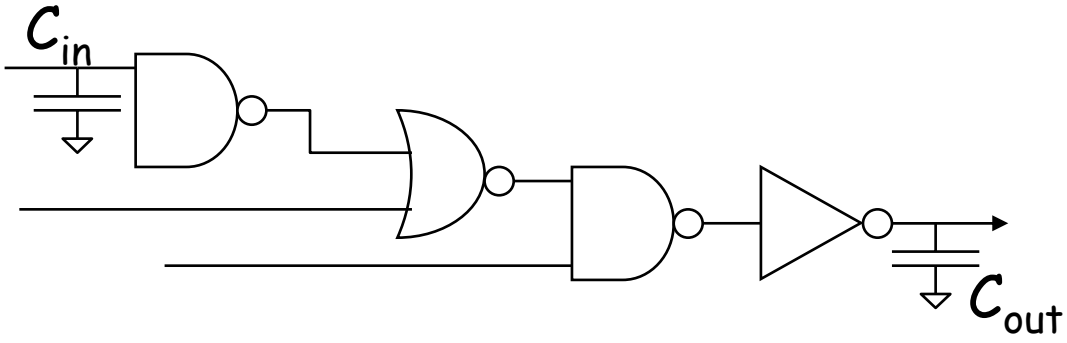
For each stage:

Delay = Logical Effort x Electrical Effort + Parasitic Delay

= 1.0 *(definition)* x 1.0 *(in = out)* + 1.0 *(drain C)*

= 2.0 units

# Optimizing Circuit Paths

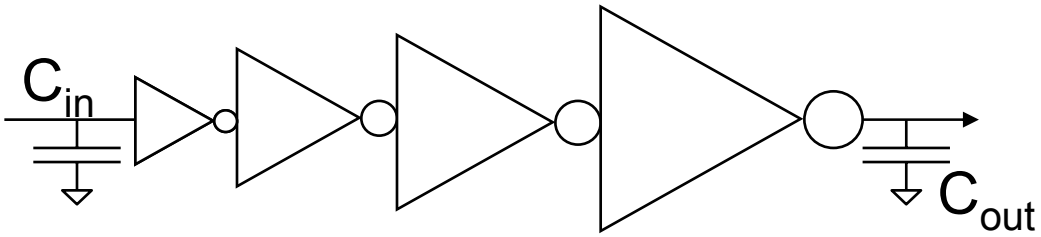



- Path logical effort, $G = \Pi\ g_i$ ($g_i$ = L.E. stage i)
- Path electrical effort, $H = C_{out}/C_{in}$ ($h_i$ = E.E. stage i)
- Parasitic delay, $P = \Sigma\ p_i$ ($p_i$ = P.D. stage i)
- Path effort, $F = GH$
- Minimum delay when each of N stages has equal effort

$$\text{Min. } D = NF^{1/N} + P$$

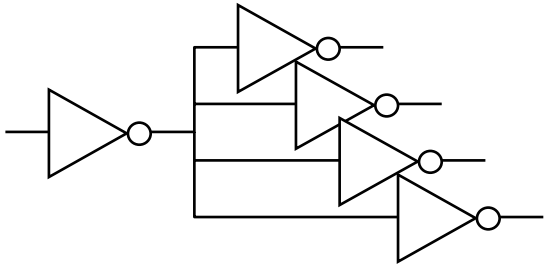
$$\text{i.e. } g_i\, h_i = F^{1/N}$$

# Optimal Number of Stages




- **Minimum delay when:**
  **stage effort = logical effort x electrical effort ~= 3.4-3.8**
  - **Some derivations have *e* = 2.718.. as best stage effort – this ignores parasitics**
  - **Broad optimum, stage efforts of 2.4-6.0 within 15-20% of minimum**
- **Fan-out-of-four (FO4) is convenient design size *(~5τ)***

FO4 delay: Delay of inverter driving four copies of itself



# Using Logical Effort in Design

- For given function, pick candidate gate topology
- Determine optimal stage effort
  - equal for all stages
- Starting at last gate
  - output load is known
  - logical effort is known (from gate topology)
  - calculate transistor size to give required stage effort
  - gives output load for preceding stage
  - lather, rinse, repeat...
- Can modify stage efforts up or down to reduce area, power, or to fit fixed set of library cells
  - optimal sizing has broad optimum