EE-429
Fundamentals of VLSI Design

Complex Logic Gates

Andreas Burg, Alexandre Levisse

CMOS Has Many Advantages

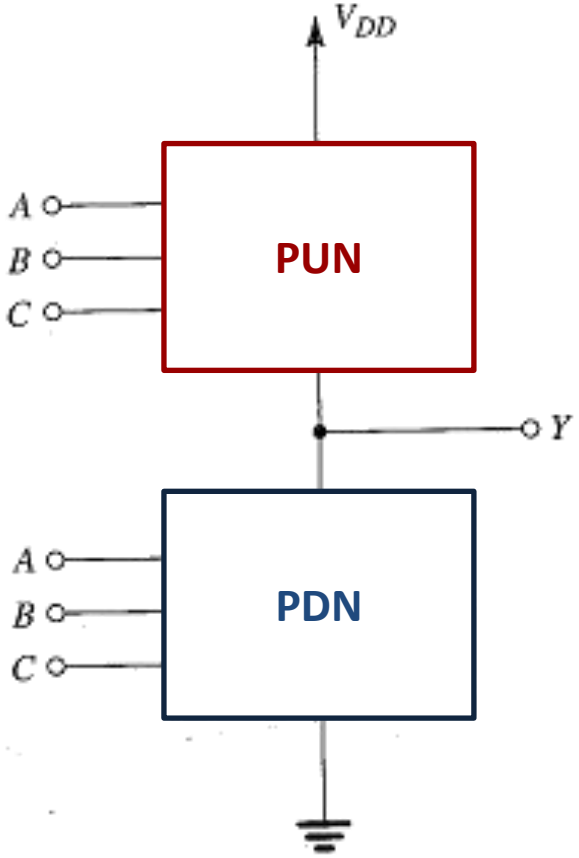
- Full rail-to-rail swing: **high noise margins**
- Logic levels not dependent upon the relative device sizes: **ratioless**
- Always a path to Vdd or Gnd in steady state: **low output impedance**
- Extremely high input resistance: **nearly zero** steady-state **input current**
- No direct path steady state between power and ground: **no static power** dissipation
- Propagation **delay is function of** load **capacitance and resistance** of transistors

Basic CMOS Logic Principal

- **With the switch model, we can construct the basic CMOS gates**

- **Basic idea:**
  - Start from the truth table (input/output relationship)
    - Pull-Up network (PUN):
      Connect OUT to VDD for input combinations that lead to a '1'
    - Pull-Down network (PDN):
      Connect OUT to GND for input combinations that lead to a '0'

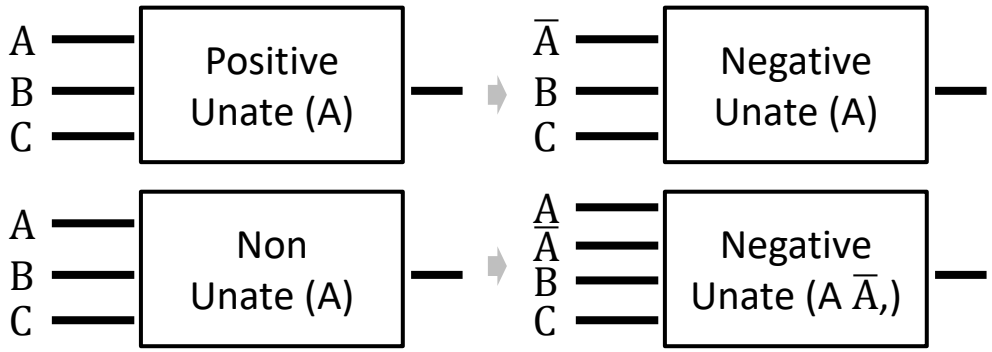
A/B/C	000	001	010	011	100	101	110	111
Y	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1



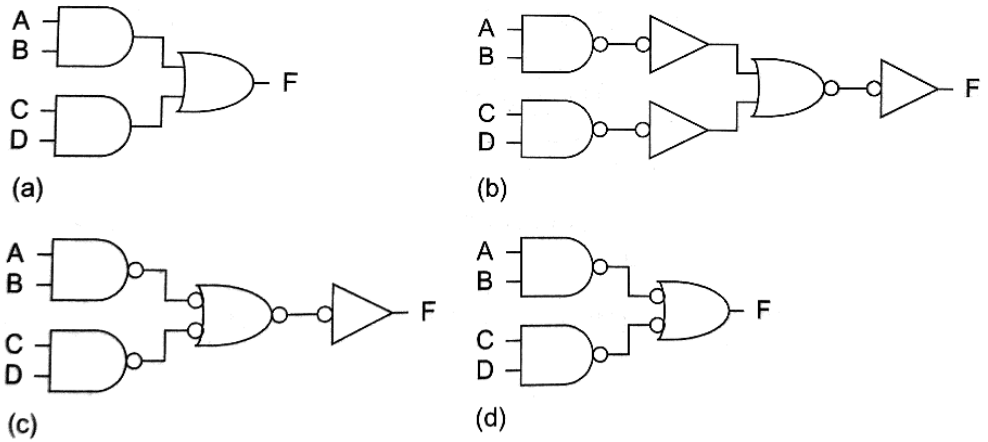
- **PUN and PDN are dual logic networks**
  - In steady state, only either VDD OR GND are connected to the output ➔ no short circuit current

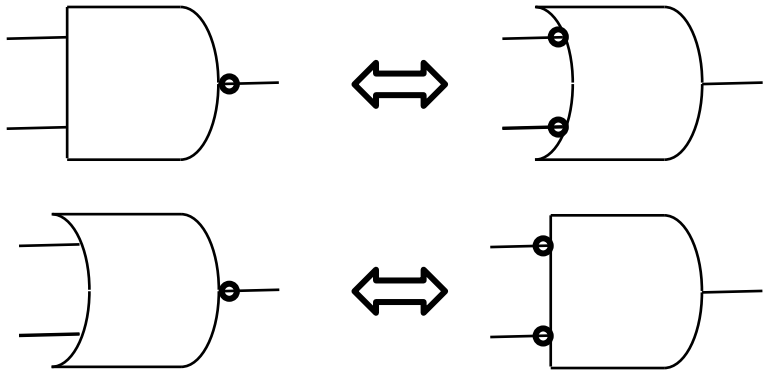
Impact of Limiting to Negative Unate Gates

- **All logic stages must be inherently inverting**
- **Two options/tools to achieve this:**
  - Using inverted input signals



  - Bubble pushing
    - Start by replacing all AND / OR gates with NAND / NOR followed by an inverter
    - Propagate inversions through the netlist and substitute gates based on DeMorgan's Laws





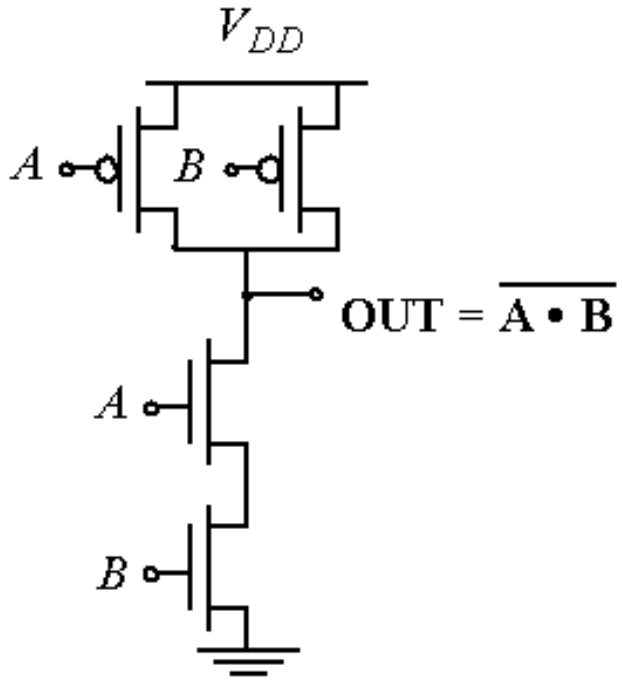
$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

$$\overline{A + B} = \bar{A} \cdot \bar{B}$$

Example: NAND Gate

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

Truth Table of a 2 input NAND gate



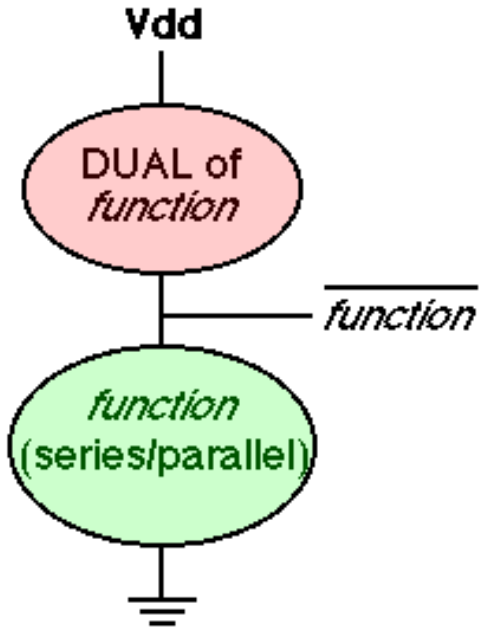
PDN: G = A B $\Rightarrow$ Conduction to GND

PUN: F= $\overline{A} + \overline{B} = \overline{AB} \Rightarrow$ Conduction to V_{DD}

$$\overline{G(In_1, In_2, In_3, \ldots)} \equiv F(\overline{In_1}, \overline{In_2}, \overline{In_3}, \ldots)$$

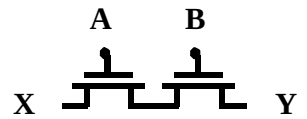
General CMOS Gate Design

1. Get into inverted form.
2. Design SERIES/PARALLEL <=> AND/OR circuit for NMOS pull-down
3. Design the Dual circuit for the PMOS pull-up

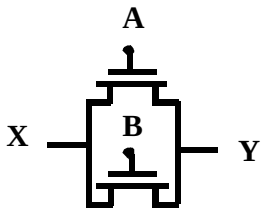


P-channel side: AND => Parallel
OR => Series

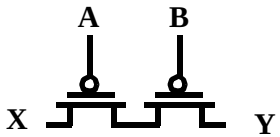
N-channel side: AND => Series
OR => Parallel



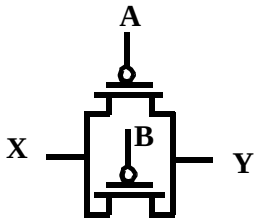
Y = X if A and B



Y = X if A OR B



Y = X if $\overline{A}$ AND $\overline{B}$ = $\overline{A + B}$



Y = X if $\overline{A}$ OR $\overline{B}$ = $\overline{AB}$

General CMOS Gate Design

Example: $f = \overline{a \cdot b + c}$

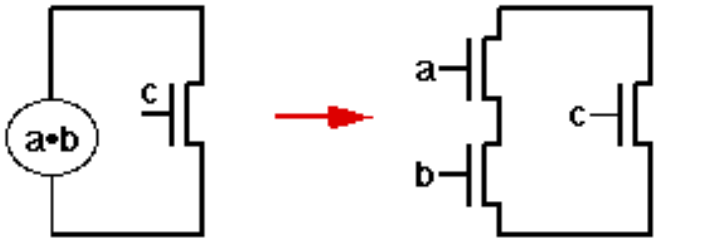
N-channel (pull-down)

Format: $\overline{(Ands,Ors)}$

$f = \overline{a \cdot b + c}$

(a AND b) OR c

(a SERIES b) PARALLEL c

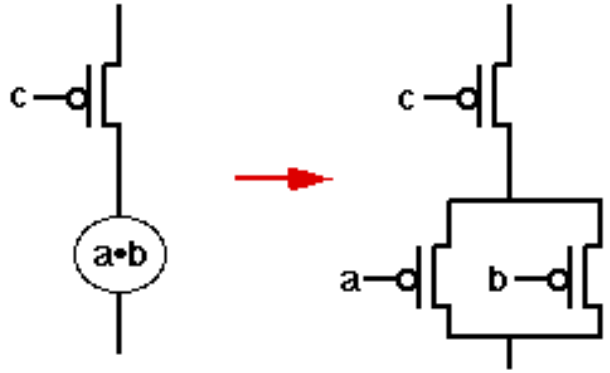


PDN is inverting function with uncomplemented inputs

P-channel (pull-up)

Format: DUAL((a AND b) OR c)

(a PARALLEL b) SERIES c



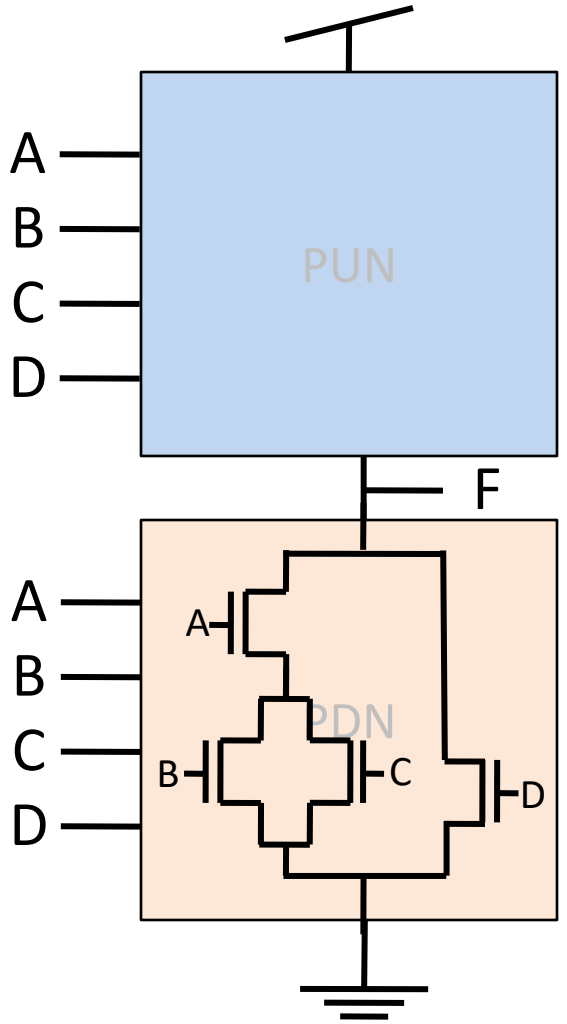
PUN is non-inverting function with complemented inputs

Example Synthesis of a Complex Gate

- Consider the following Boolean function

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

  - Function is *inverting* with *uncomplemented inputs*, so we can immediately derive the ***PDN*** from the given function.
  - ***B*** and ***C*** are *parallel*
  - ***A*** is in *series* with the ***B+C*** network
  - ***D*** is *parallel* to the ***A(B+C)*** network.



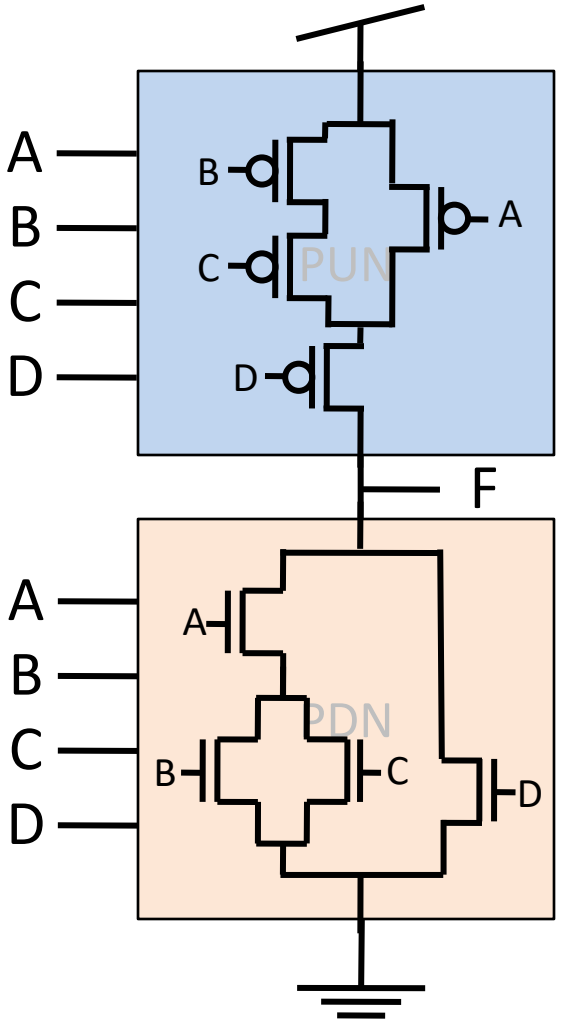
Synthesis of a complex gate

- **Next: use *DeMorgan* to derive the *PUN*:**

$$F = \overline{D + A \cdot (B + C)} = \overline{D} \cdot \overline{A \cdot (B + C)} =$$
$$= \overline{D} \cdot \left(\overline{A} + \overline{B + C}\right) = \overline{D} \cdot \left(\overline{A} + \overline{B} \cdot \overline{C}\right)$$

  - Our ***PUN*** function is *non-inverting* with complemented inputs.
  - ***B*** and ***C*** are in *series*
  - ***A*** is *parallel* with the ***BC*** network
  - ***D*** is in *serial* with the ***A+(BC)*** network.

- **The PUN can also be derived from the PDN**

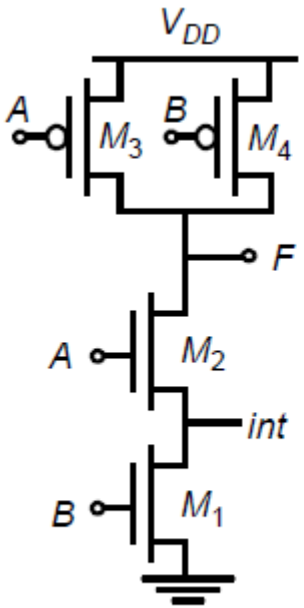


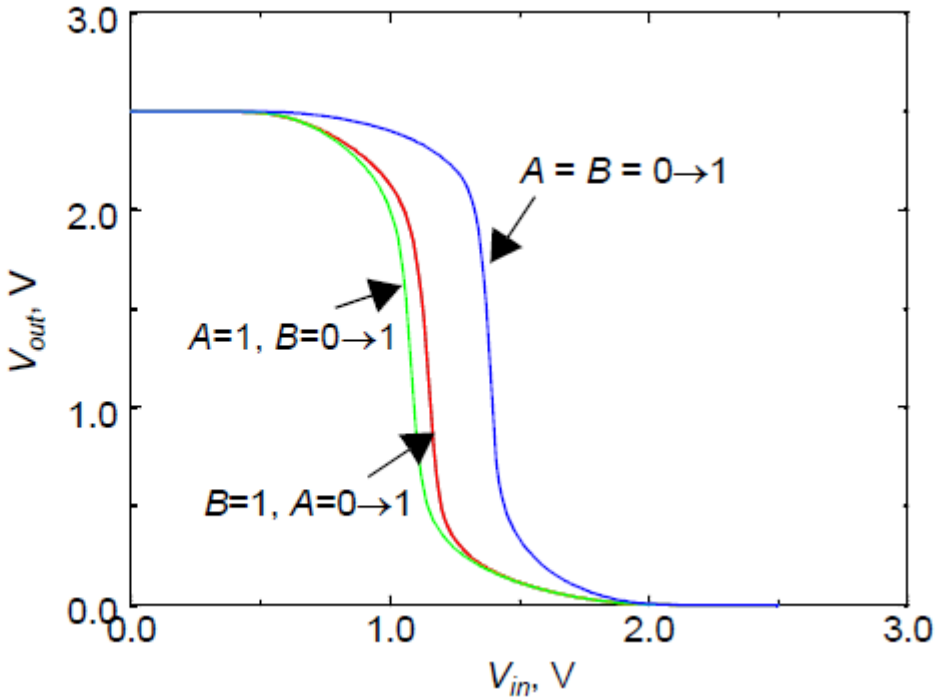
Transistor Sizing for Complex Gates (Static)

- Similar to the inverter, we **first aim for a balanced gate**
  - **Upsizing** comes as a second step: **increase of all transistor sizes by a common factor S**

- Unfortunately, even the **VTC** of a complex gate **depends on all inputs**

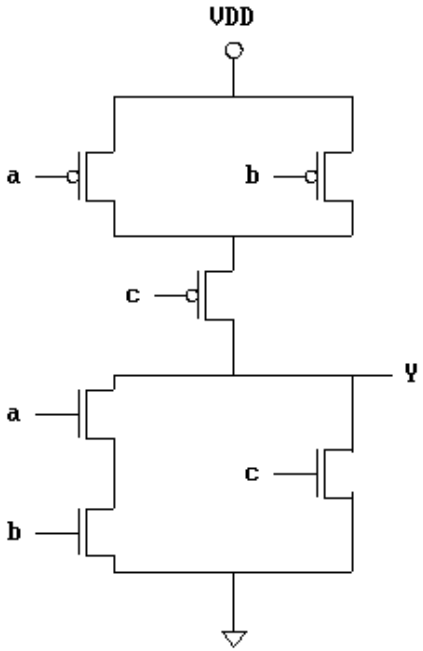
- **Example VTC**: 2-input NAND (with compensated μ_p/μ_n)
  - VTC (**transition of a single input**) **depends on state of other inputs**
    - However, **differences are small**
  - **Concurrent input changes have the strongest impact**

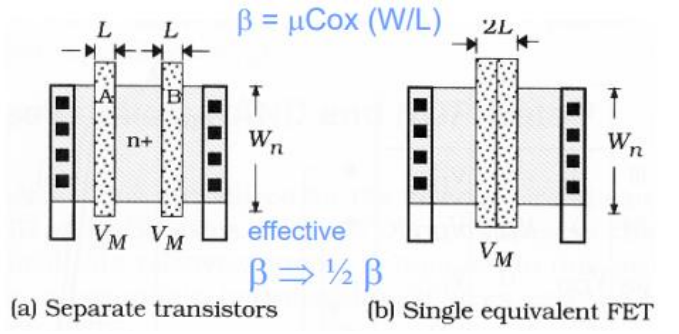


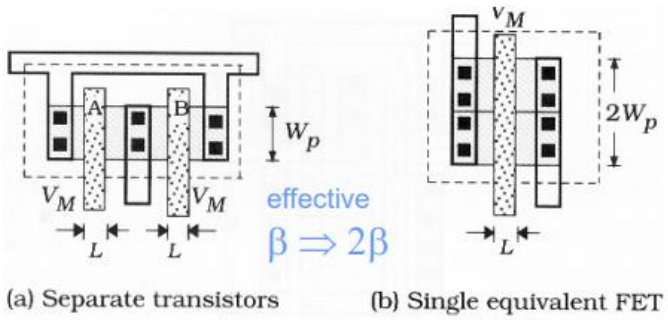


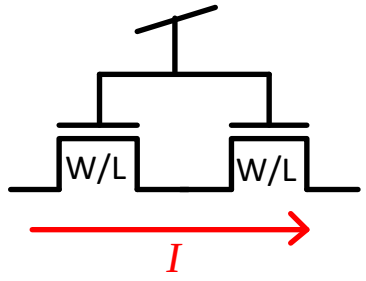
Transistor Sizing for Complex Gates (Dynamic)

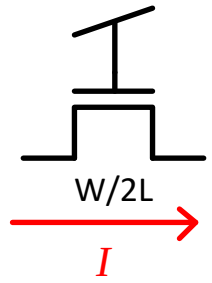
- In the dynamic case, the **delay depends on the on-resistance** of the path **that charges or discharges the output**
- In complex gates, **different paths may contain different number of transistors**
  - **Transistors in series, have a higher on-resistance**
  - **Transistors in parallel have a lower on-resistance (depending on the input)**

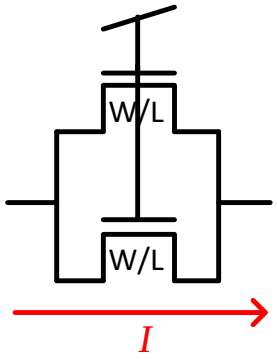


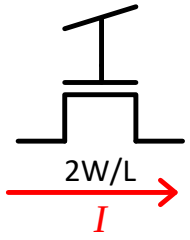






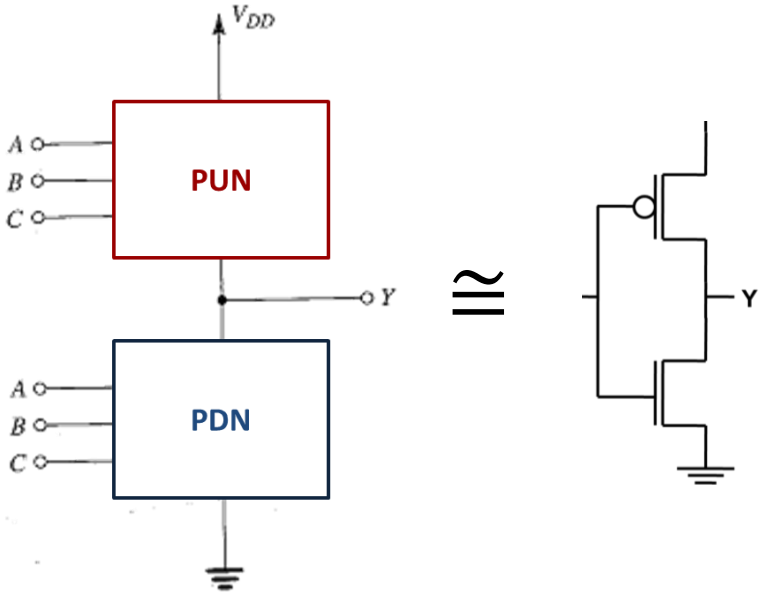






Transistor Sizing for Complex Gates

- **How to balance a complex gate** with multiple different VTCs AND different paths that determine the on-resistance (i.e., speed)**?**
- **Consider the following:**
  - Multiple inputs are unlikely to change exactly at the same time: **Consider only single-input transitions**
  - VTC state dependency with a single input change is small: Ignore (steady-) state dependency for the optimization
  - **Optimize speed for the worst-case**: Upsize slow paths to match the fast (single transition) paths
  - Target same drive strength for all “minimum size” gates without the need to artificially slow down the fastest minimum size gate (=INVERTER)

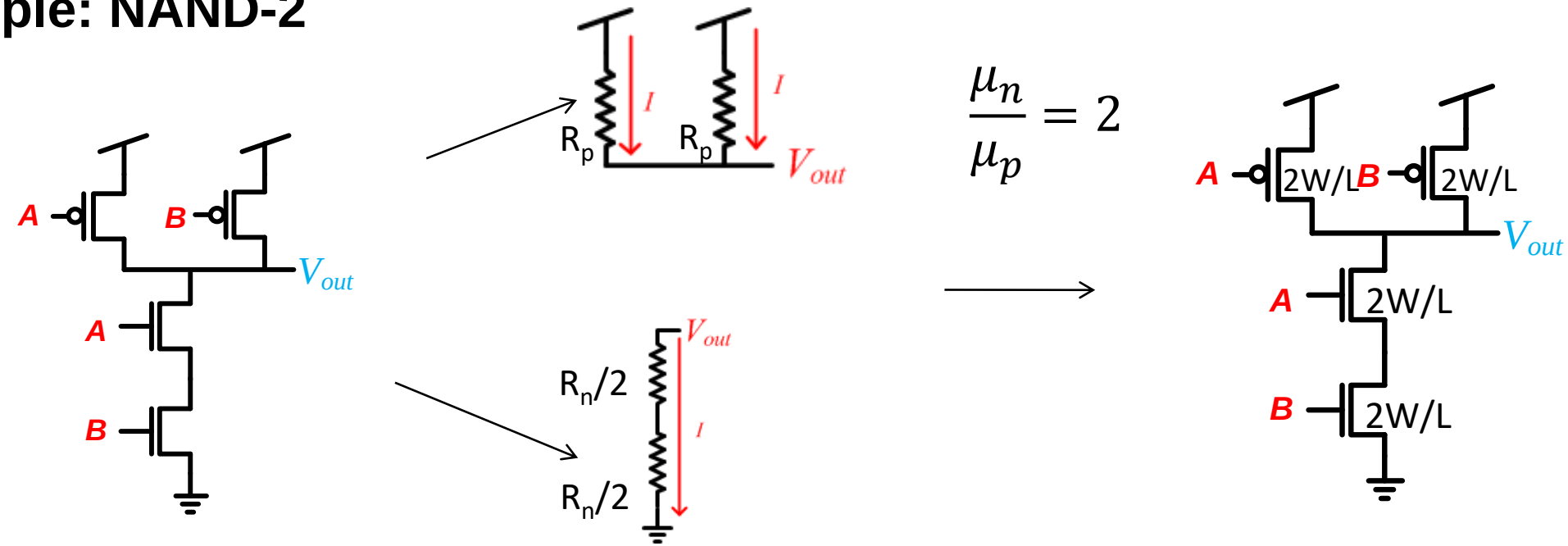


- **Size transistors of a minimum-size complex gate to match the minimum drive strength to that of a minimum size inverter**

Transistor Sizing for Complex Gates (Example)

- **Compensate for higher resistance in series connected transistors** by upsizing
  - Assume: series resistance is linear in the number of transistors
- **For parallel paths, assume worst case (only one turned on)**

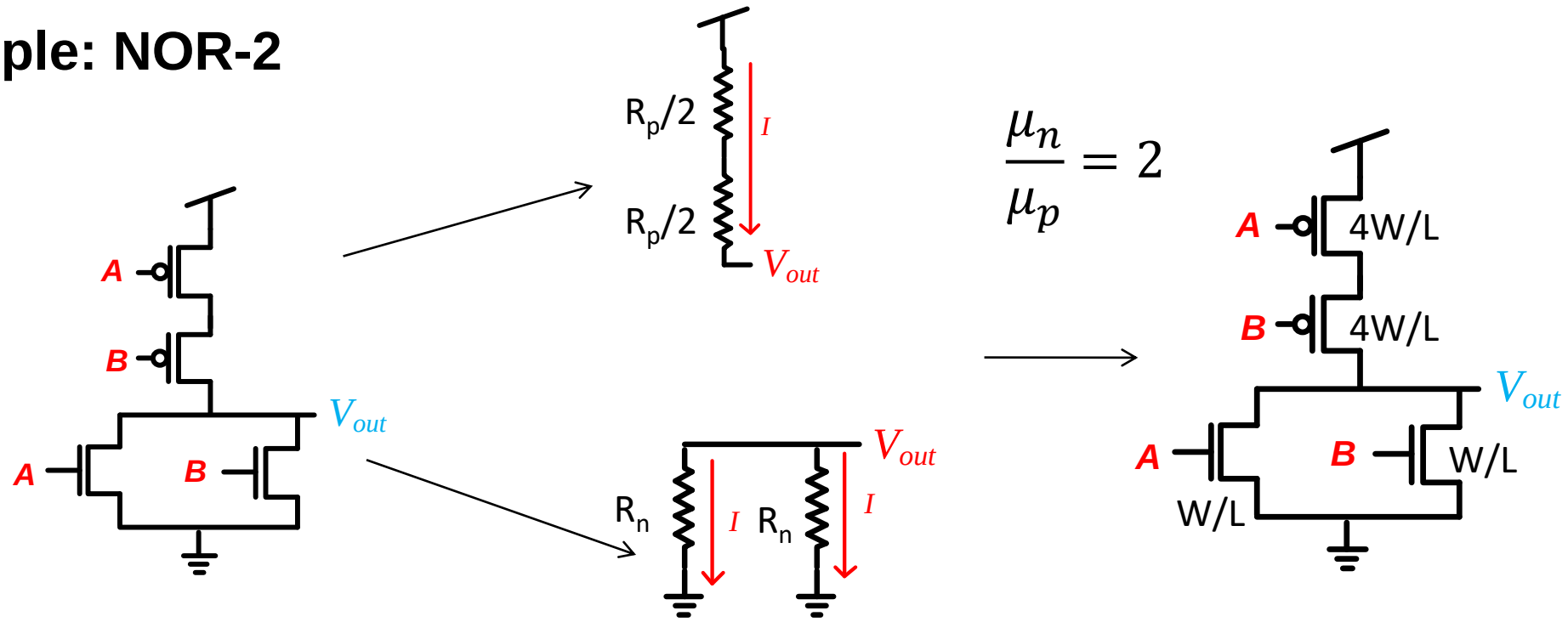
- **Example: NAND-2**



Transistor Sizing for Complex Gates (Example)

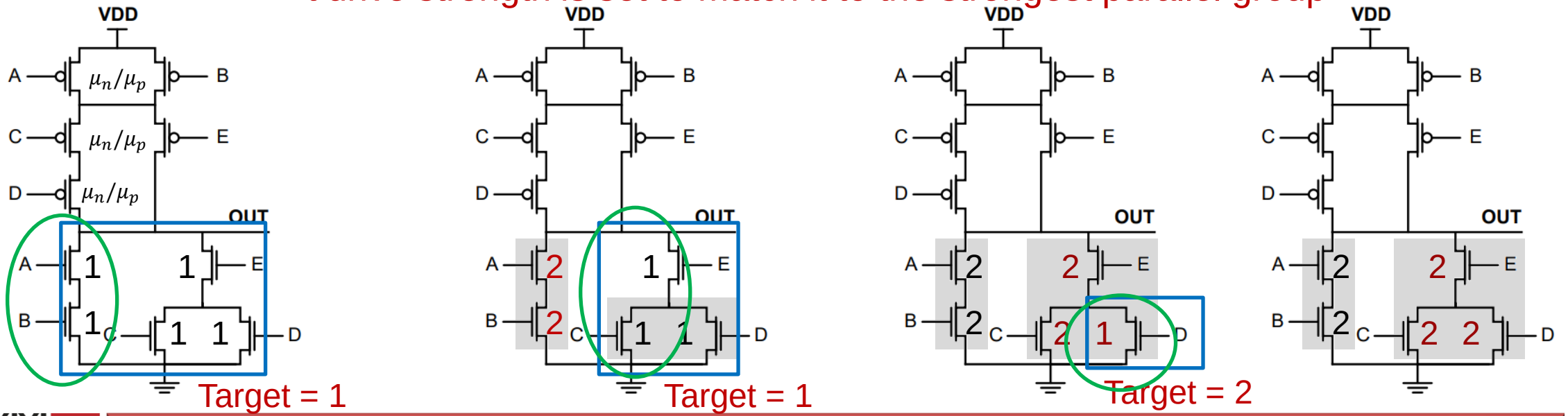
- **Compensate for higher resistance in series connected transistors** by upsizing
  - Assume: series resistance is linear in the number of transistors
- **For parallel paths, assume worst case (only one turned on)**

- **Example: NOR-2**



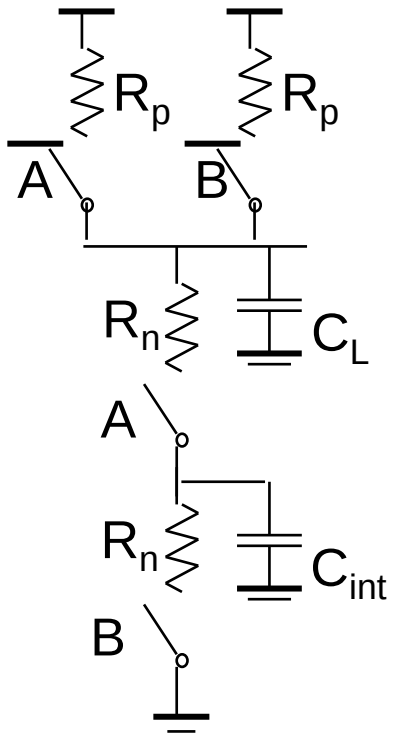
Strategy for Transistor Sizing for Complex Gates

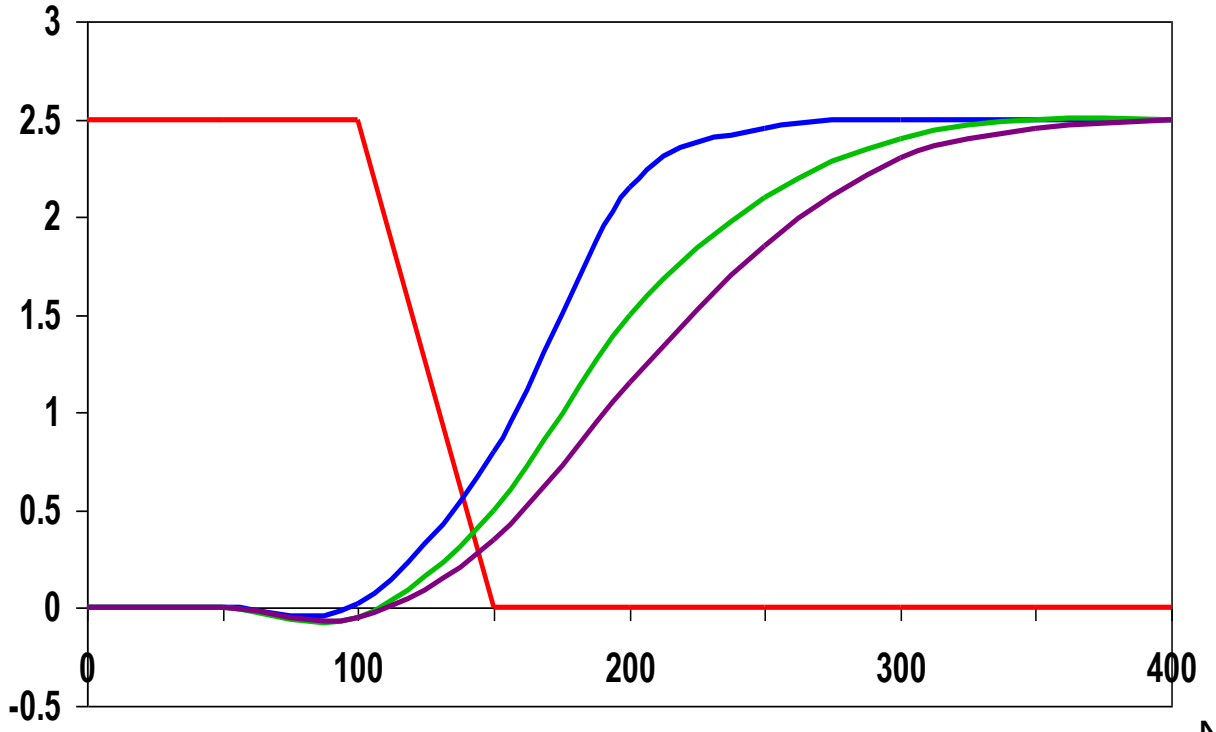
- **The algorithm below applies to both nMOS and pMOS transistors**
  1. Start from all minimum size transistors (pMOS start from the mobility ratio μ_n/μ_p)
  2. The first target drive strength is 1 and the first group includes all devices
  3. Identify the largest stack of devices ***N*** in the group
  4. Scale the width of devices to match the target drive strength
  5. Identify those devices that are in parallel to the resized stack or any part thereof as new group
  6. The new target drive strength is set to match it to the strongest parallel group



Residual Input Pattern Effects

- Equalizing on-resistance should equalize the delays
- However, we still observe differences in delays, even through the same path





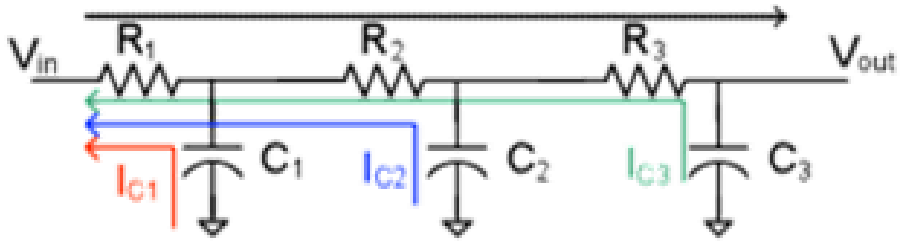
Input Data Pattern	Delay (psec)
A=B=0→1	69
A=1, B=0→1	62
A= 0→1, B=1	50
A=B=1→0	35
A=1, B=1→0	76
A= 1→0, B=1	70

NMOS = 0.5μm/0.25 μm
PMOS = 0.75μm/0.25 μm
C_L = 100 fF

- Reason: distributed intrinsic capacitance

Delay Estimation using the Elmore Delay Model

- **Charging and discharging of** the internal (**distributed**) **capacitances can be computed with** a special case of **the Elmore Delay Model**



$$\tau_{elmore} = R_1C_1 + (R_1 + R_2)C_2 + (R_1 + R_2 + R_3)C_3$$

- **Upsizing** to keep a constant delay (for the compact model) **yields**
  - Assume: *N* series transistors

$$R_n = R_{eq}/N$$

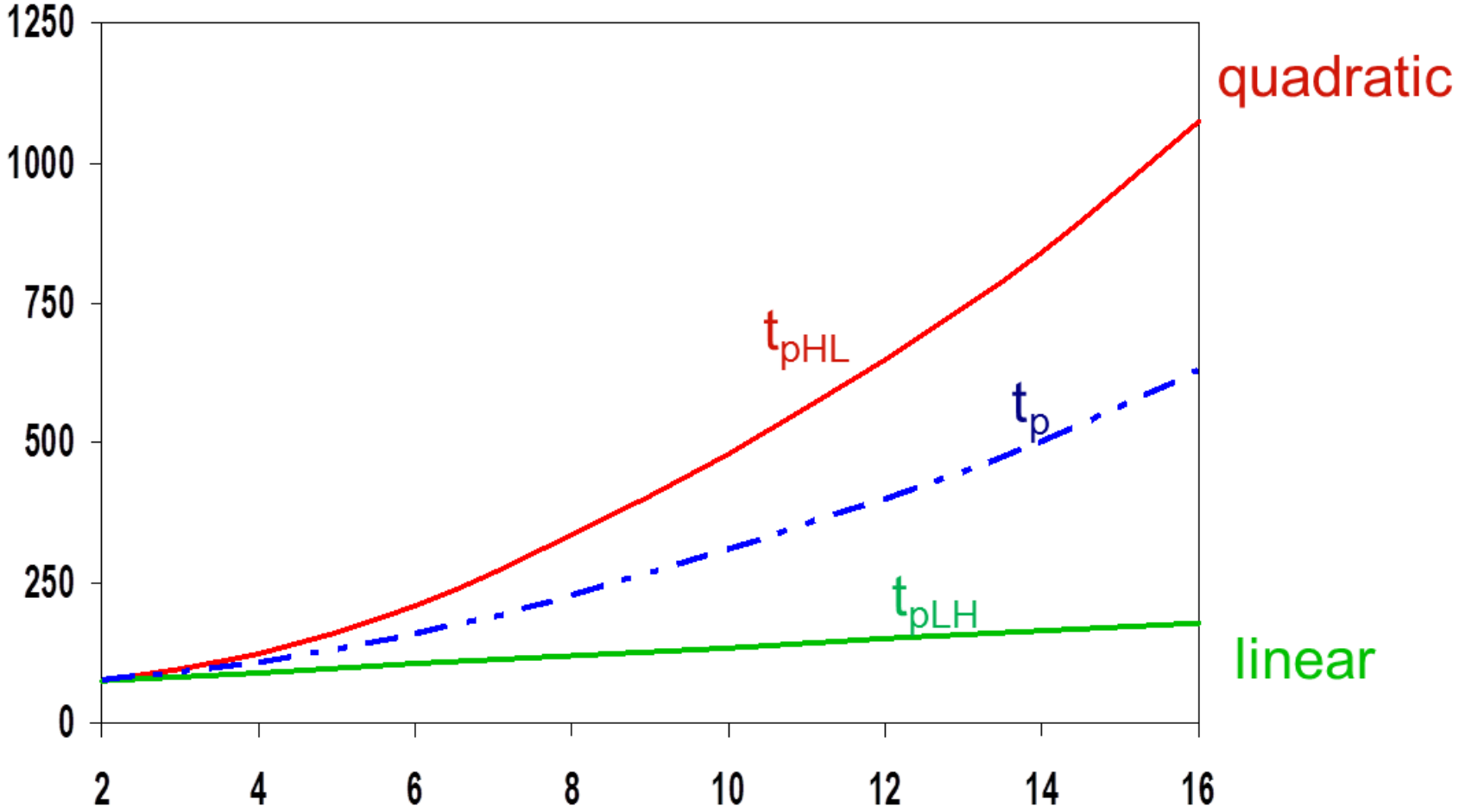
$$C_n = C_{eq} \cdot N$$

$$\tau_{elmore} = \frac{R_{eq}}{N} \cdot N \cdot C_{eq} \cdot (1 + 2 + \cdots + N)$$

$$= R_{eq} \cdot C_{eq} \cdot \frac{N(N+1)}{2} \sim N^2$$

- **Delay scales quadratically with number of inputs even after sizing**

Delay Scaling with Number of Inputs



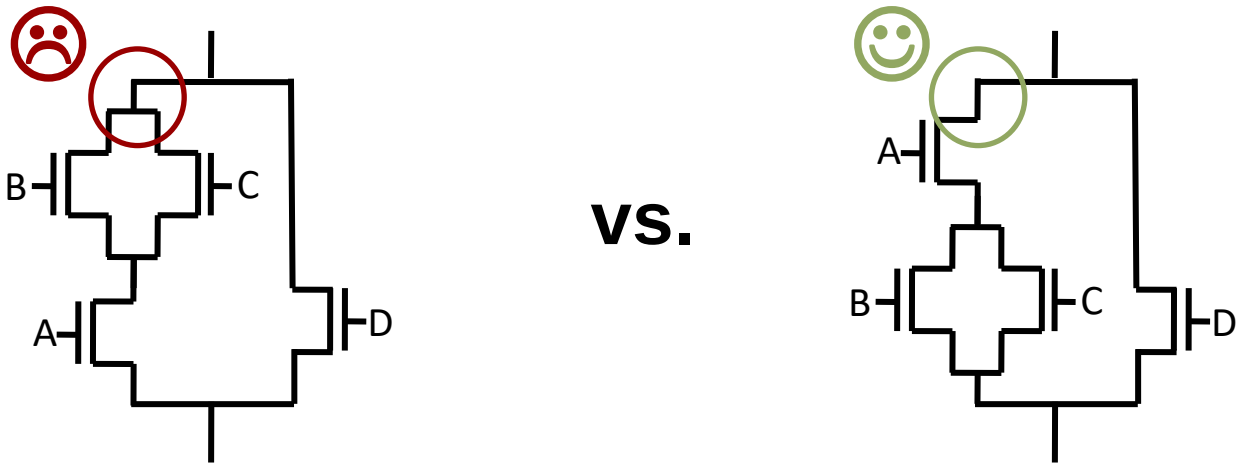
- **Avoid gates with more than four (4) inputs**

Optimizations: Stack Re-Ordering

- **When stacking** heterogeneous groups of transistors, **different orders provide**
  - the same functionality
  - In our simple model, order has no impact on series resistance
- **Stack-order of transistors (or groups of transistors) can be optimized**

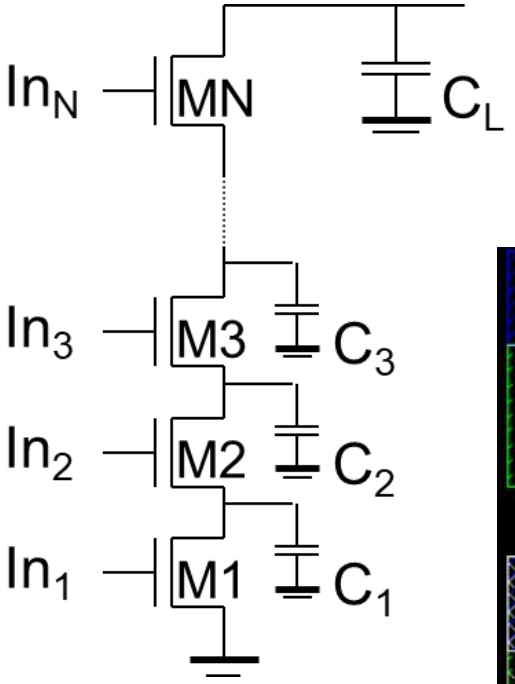
- **Objective: minimize capacitance that is far from the rails**

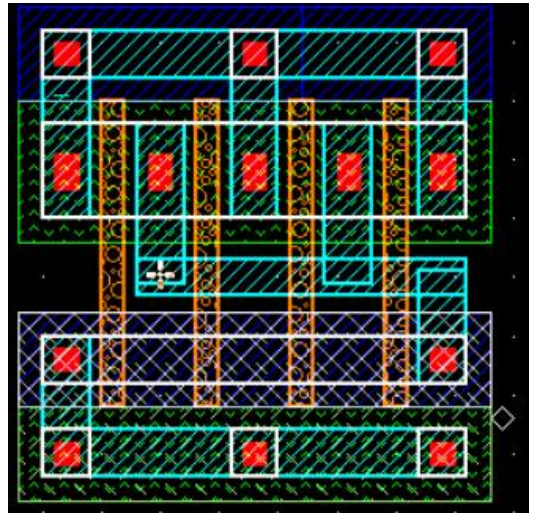
- Example:
  - Layout may play a role as well



Optimizations: Progressive Sizing

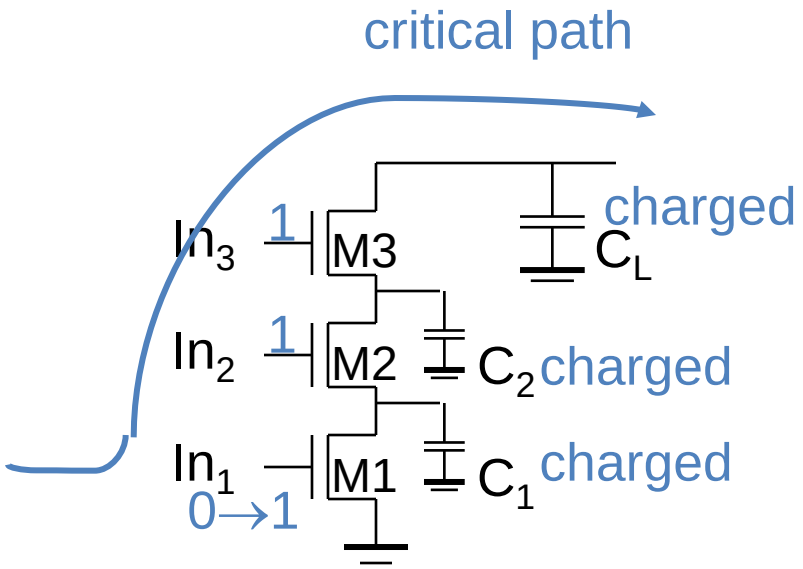
- **Progressive sizing**
  - Note that M1 is on the path of all Capacitors, while M*N* is only on the path of CL.
  - M1 is more critical than M*N* and should be wider
    - **M1 > M2 > M3 > … > MN**
    - **the FET closest to the output is the smallest**
  - Delay reduction by more than 20% possible
  - However, often **large area overhead in layout…**



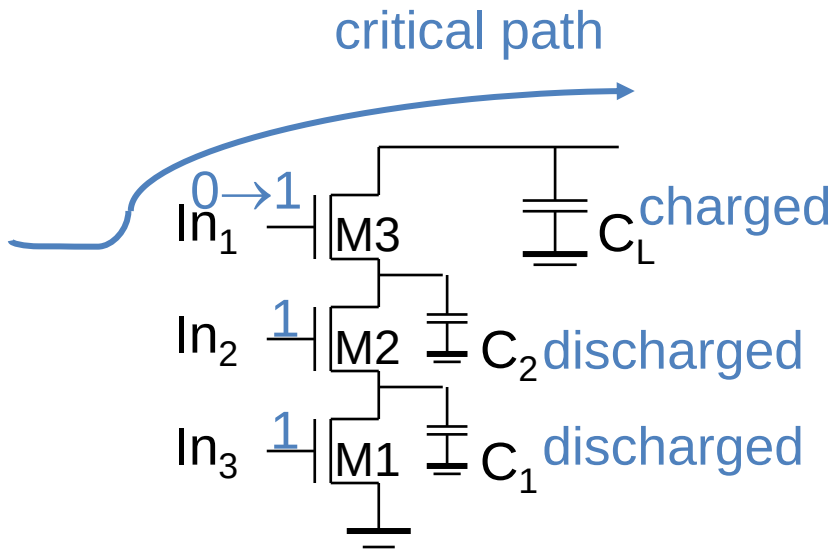


Optimizations: Input Reordering

- **Transistor ordering** based on arrival times of the inputs
- Place **latest arriving signal** (critical path) **closest to the output**, to minimize the capacitance that needs to be discharged when the signal arrives



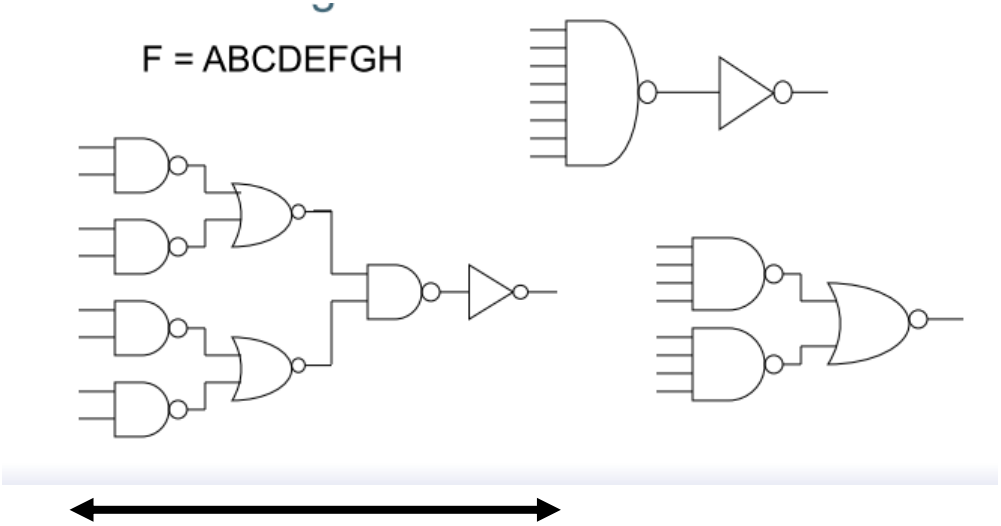
delay determined by time to discharge C_L, C_1 and C_2



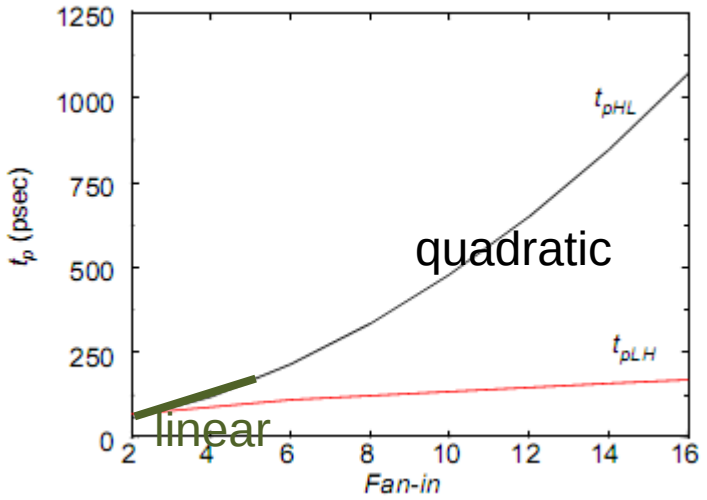
delay determined by time to discharge C_L

Optimizations: Logic Restructuring

- Logic can be **re-written in multiple stages**
  - Large fan in gates (with quadratic dependence) **reduce to** low fan in gates (linear dependence)
  - Number of stages grows only logarithmically with overall Fan-In cone



Logarithmic increase with *N*



Optimizations: Buffer Insertion

- **A complex gate is a “weak driving gate”**
- **Upsizing the entire gate is costly in terms of area and internal capacitance (intrinsic delay)**
- **Better solution:** minimum size complex high fan-in gate followed by a strong driver (buffer or inverter)
  - Isolates fan-in from fan-out
  - Avoids increasing te load on the previous stage
  - Complex gate only worries about its intrinsic delay