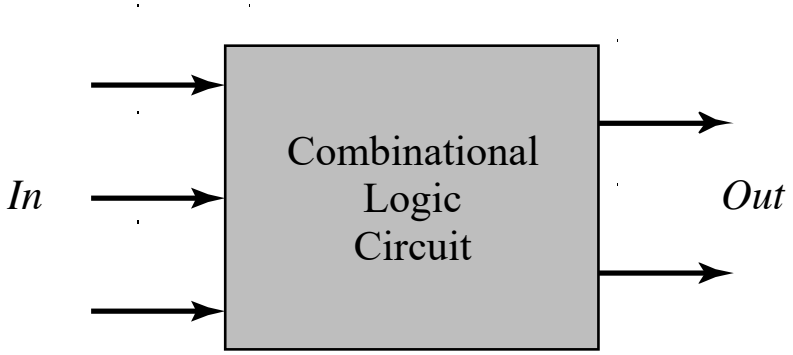
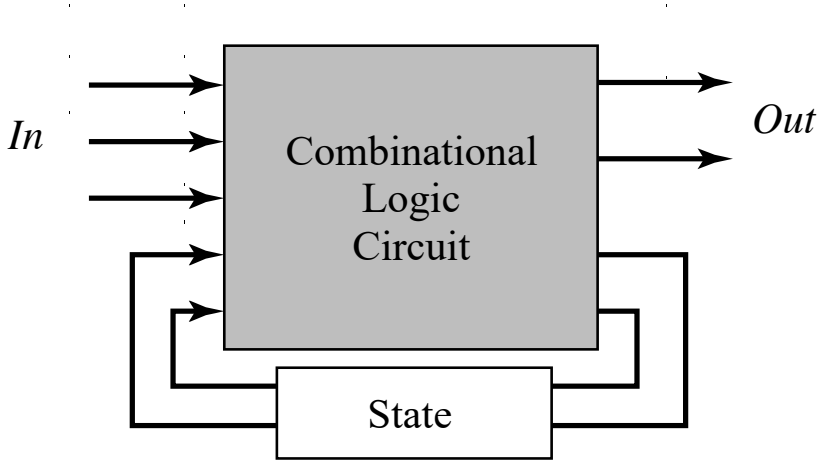
Digital Integrated Circuits

Designing Combinational Logic Circuits

Fuyuzhuo

Combinational vs. Sequential Logic





Combinational

Sequential

Output = *f*(*In*)

Output = *f*(*In, Previous In*)

agenda

- ***Static CMOS design***
- ***Ratioed logic design-Pseudo NMOS***
- ***Pass transistor design***
- ***Dynamic logic***

agenda

- ***Static CMOS design***
- *Ratioed logic design___Pseudo NMOS*
- *Pass transistor design*
- *Dynamic logic*

What is the difference between inverter and logic?

Static CMOS logic

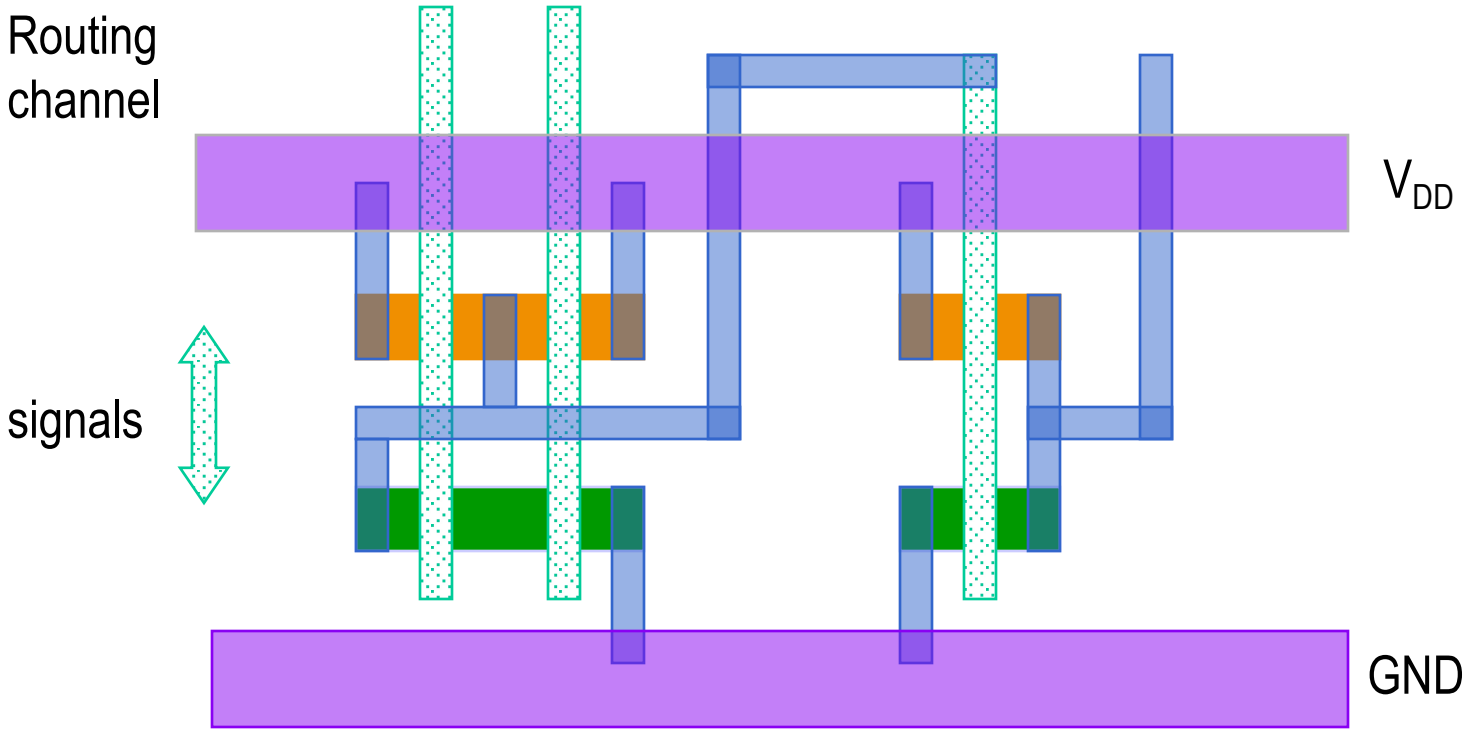
- ***CMOS static characteristic***
- *CMOS propagate delay*
- *Large fan-in technology*
- *Logic effort*
- *CMOS power analysis*

Static CMOS Circuit

- Gate output is connected to either V_{DD} or V_{SS} via a low-resistive path*
- Contrast to the dynamic circuit class, which relies on temporary storage of signal values on the capacitance of high impedance circuit nodes

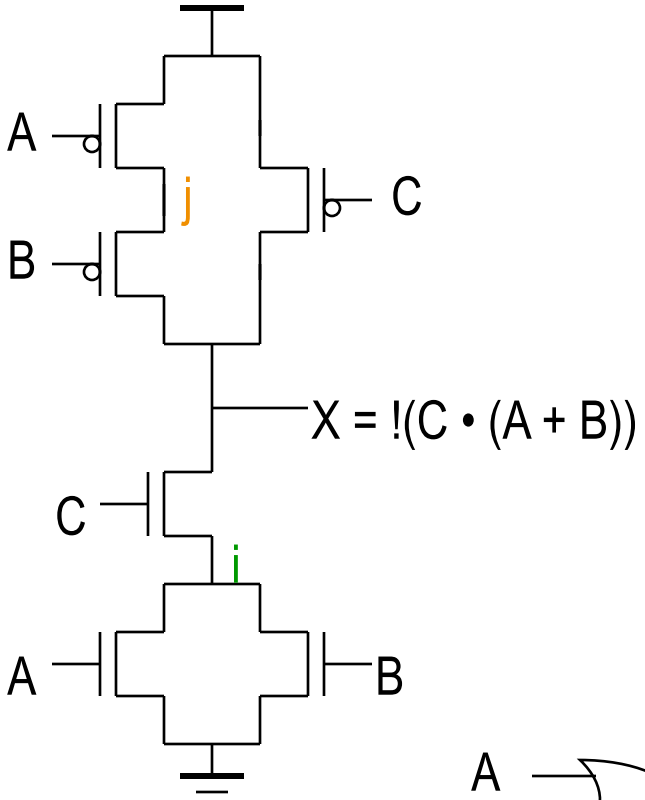
* except during the switching transients

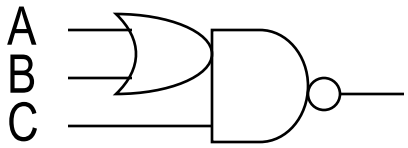
Standard Cell Layout Methodology

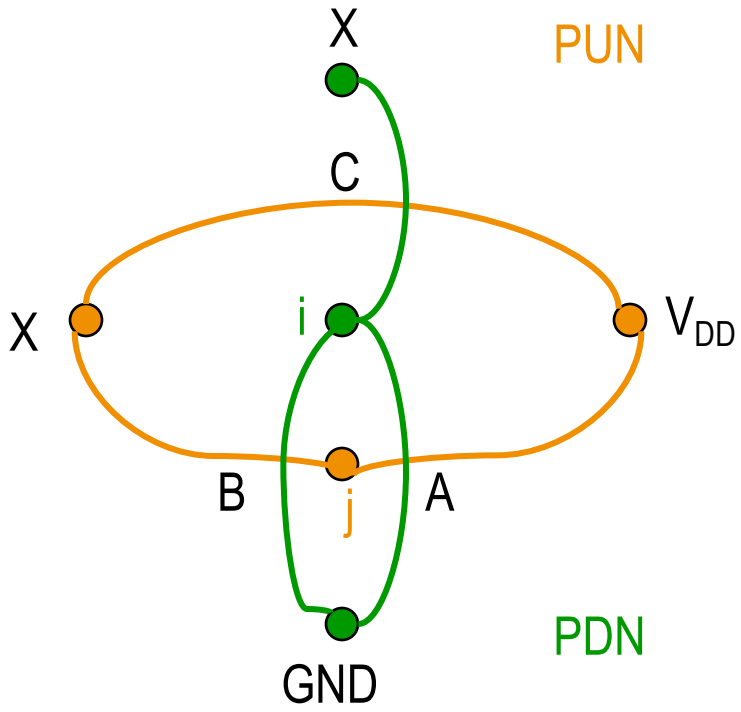


What logic function is this?

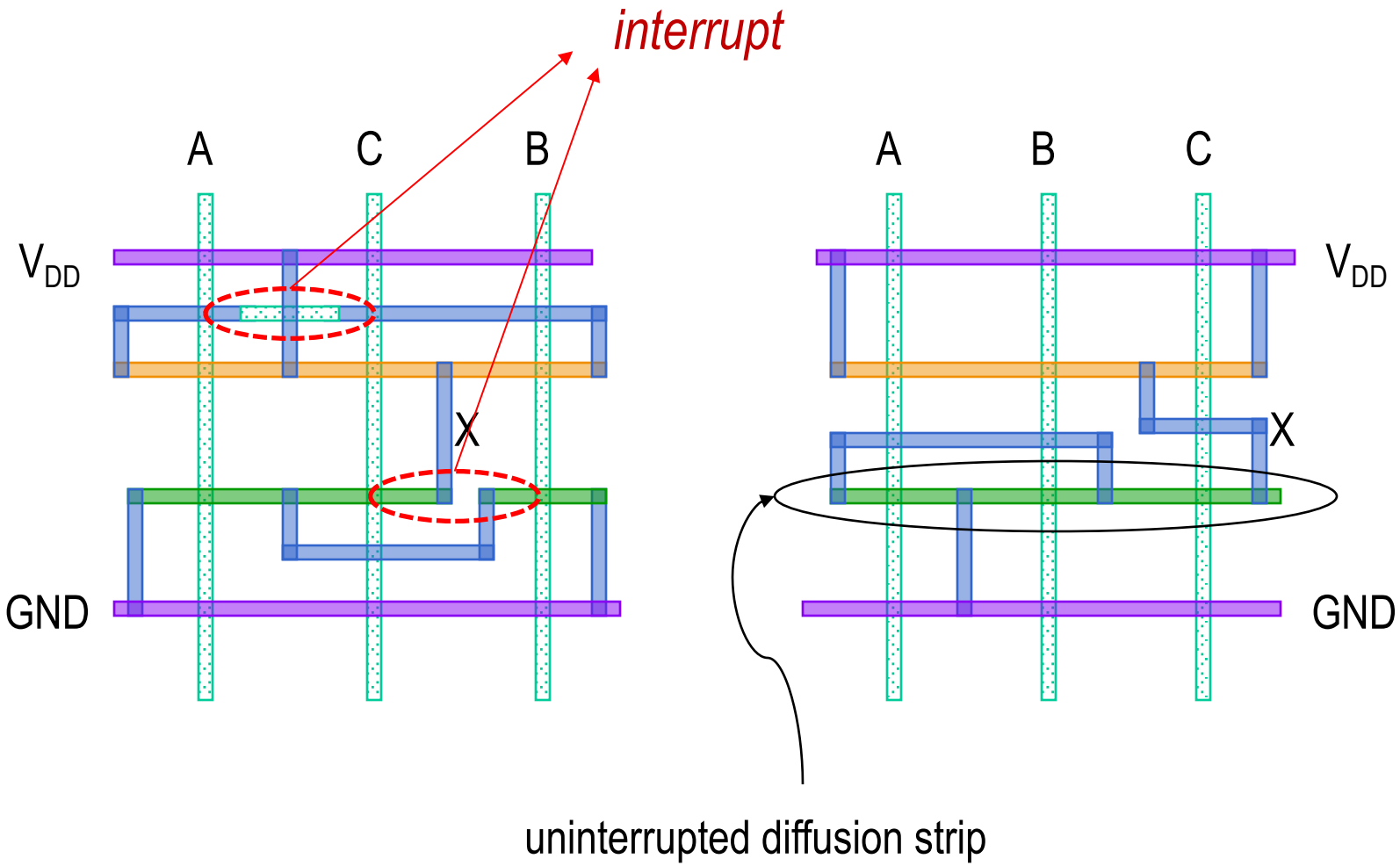
OAI21 Logic Graph





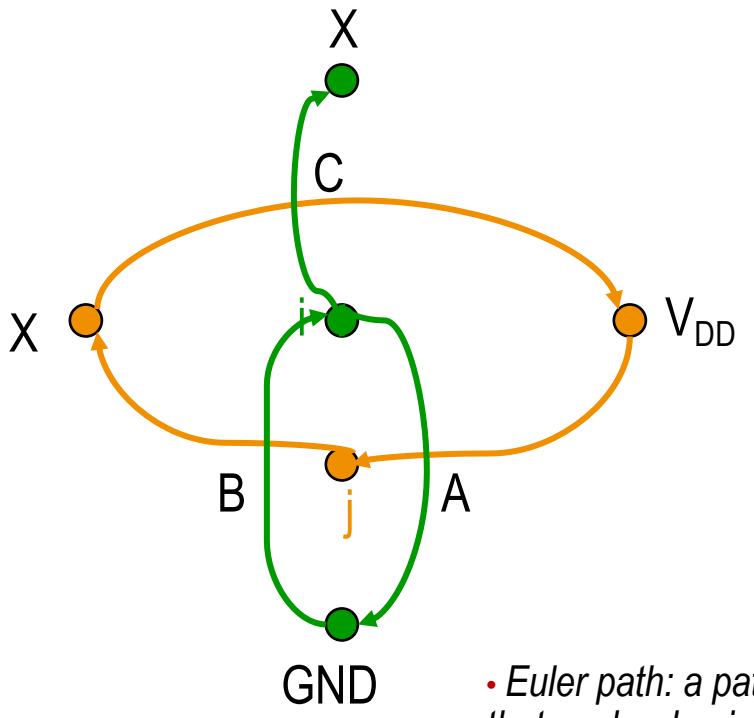


Two Stick Layouts of !(C • (A + B))



Consistent Euler Path

- An uninterrupted diffusion strip is possible only if there exists a Euler path in the logic graph

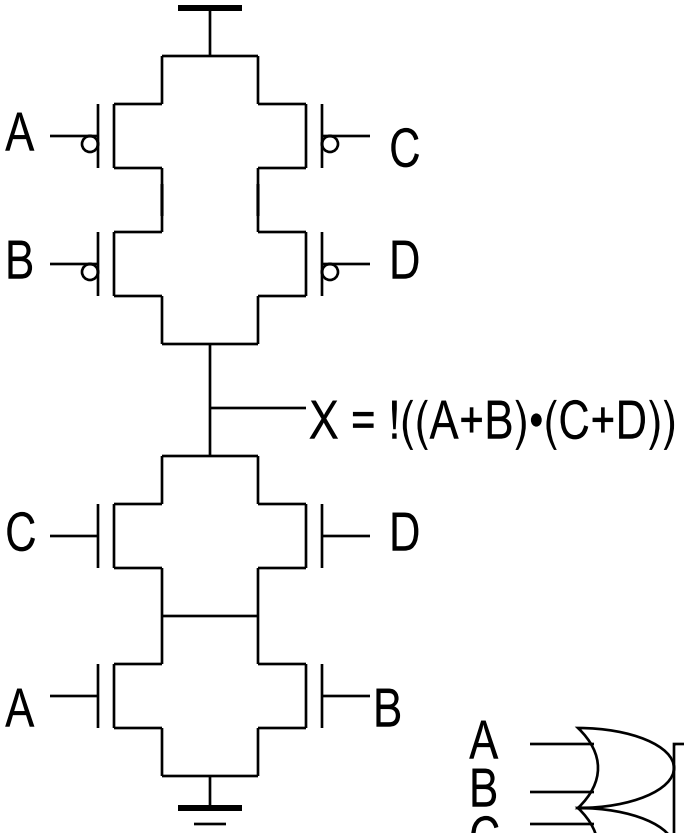


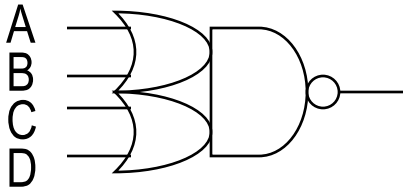
- *Euler path: a path through all nodes in the graph such that each edge is visited once and only once*

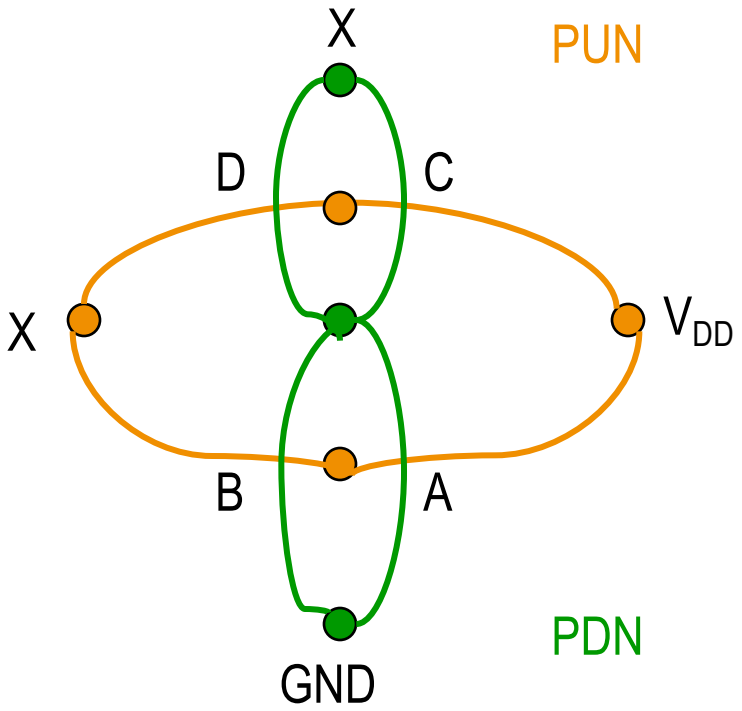
A B C

- For a single poly strip for every input signal, the Euler paths in the PUN and PDN must be consistent (the same)

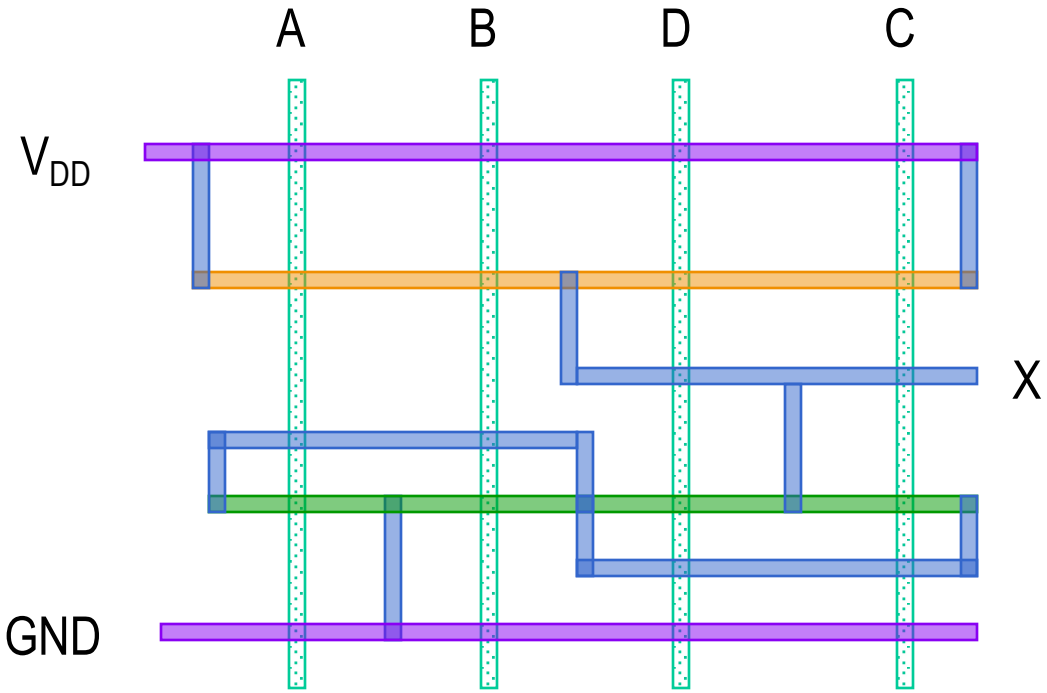
OAI22 Logic Graph







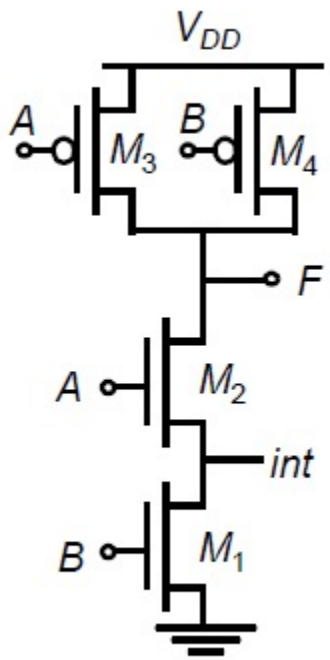
OAI22 Layout

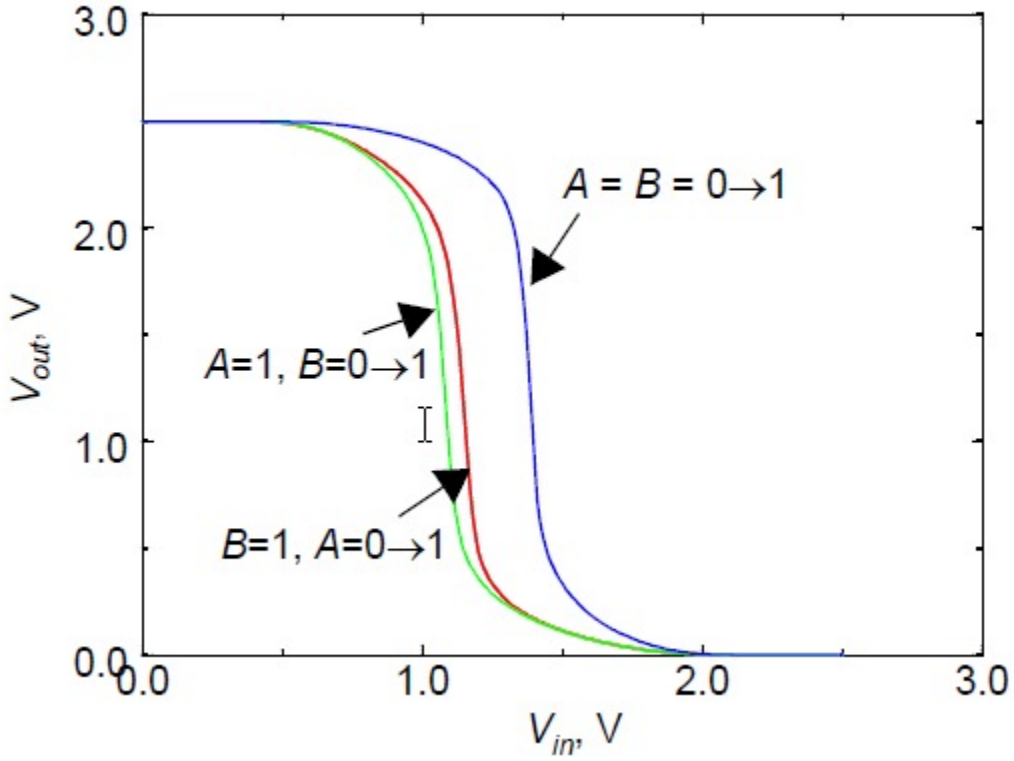


Some functions have no consistent Euler path like

x = !(a + bc + de) (but x = !(bc + a + de) does!)

VTC is Data-Dependent





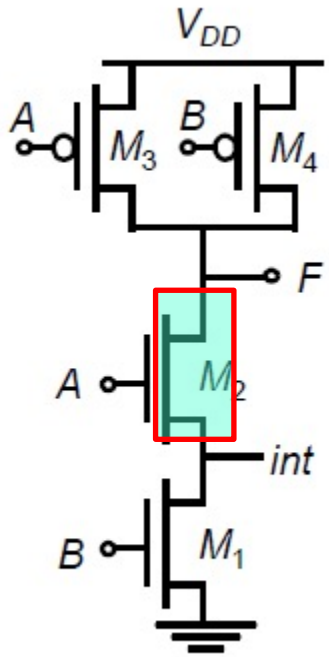
- ***The threshold voltage of M_2 is higher than M_1 due to the body effect (γ)***

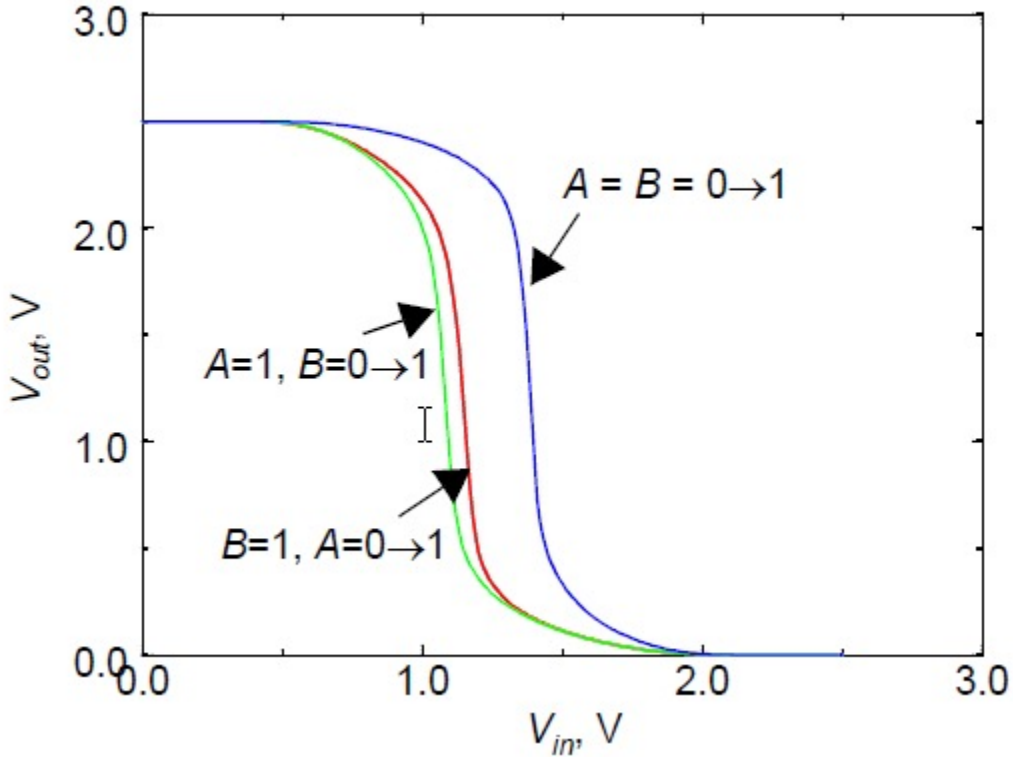
  $V_{Tn1} = V_{Tn0}$

  $V_{Tn2} = V_{Tn0} + \gamma(\sqrt{(|2\phi_F| + V_{int}) - \sqrt{|2\phi_F|})}$

  since V_{SB} of M_2 is not zero (when $V_B = 0$) due to the presence of C*int*

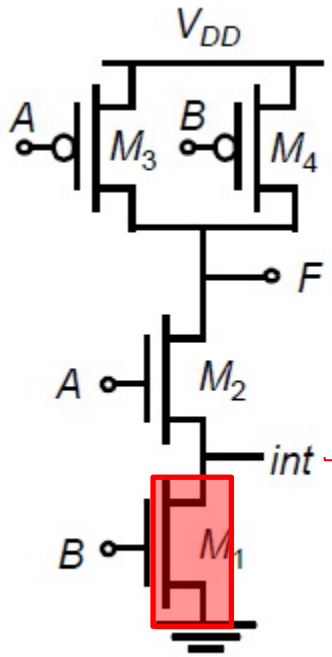
VTC is Data-Dependent

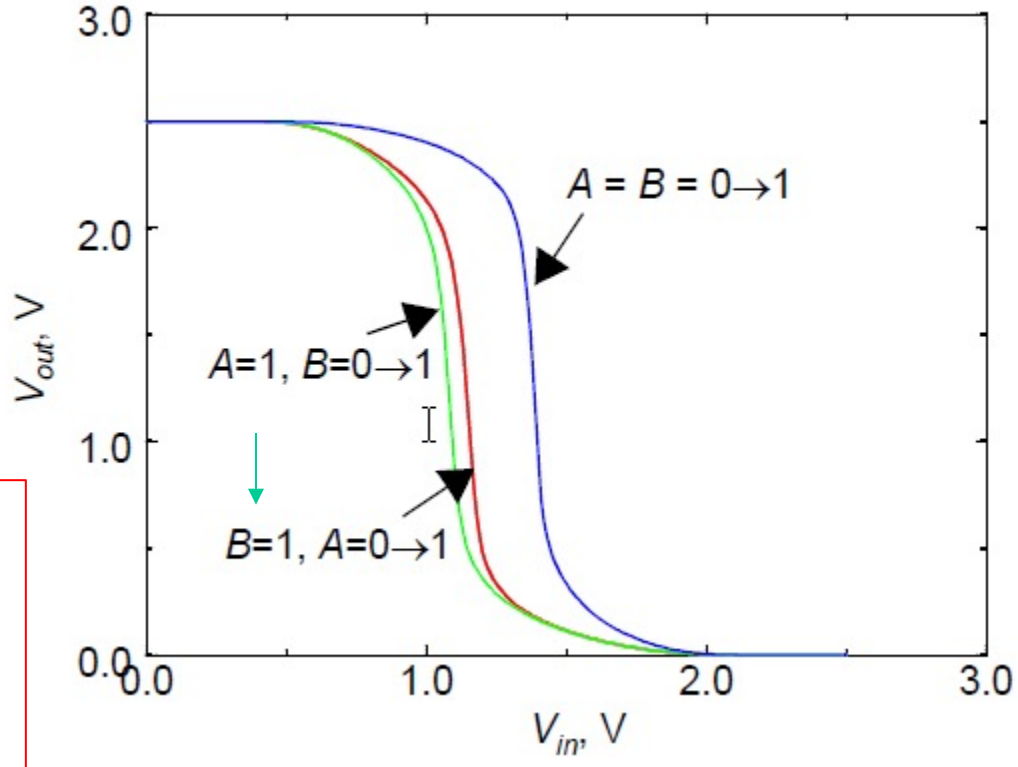




$$I_{DS-A} = k'_n \frac{W}{L}\left[(V_{GS} - V_T)V_{DS} - \frac{1}{2}V_{DS}^2\right]$$

VTC is Data-Dependent





$$I_{DS-A} = k'_n \frac{W}{L}\left[(V_{GS} - V_T)V_{Dsat} - \frac{1}{2}V_{Dsat}^2\right]$$

CMOS Properties

- Full rail-to-rail swing ***high*** noise margins
- not dependent upon device sizes ***ratioless***
- Always a path to Vdd or Gnd ***low*** output impedance
- zero steady-state input current ***high*** input resistance
- No direct path steady state ***no*** static power
- Propagation delay function of load capacitance and resistance of transistors

Static CMOS logic

- *CMOS static characteristic*
- ***CMOS propagate delay***
- *Large fan-in technology*
- *Logic effort*
- *CMOS power analysis*

Delay Definitions

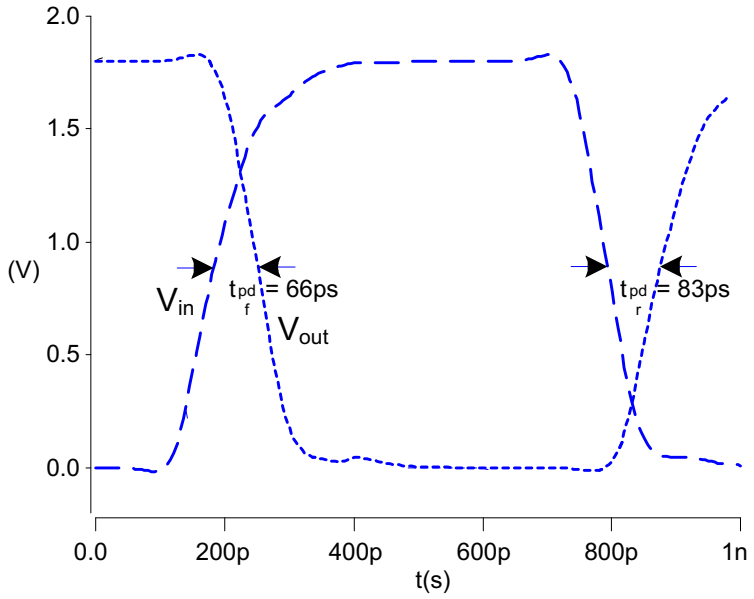
- **t_{pdr}**: *rising propagation delay*
  - From input to rising **output** crossing $V_{DD}/2$
- **t_{pdf}**: *falling propagation delay*
  - From input to falling **output** crossing $V_{DD}/2$
- **t_{pd}**: *average propagation delay(max-time)*
  - $t_{pd} = (t_{pdr} + t_{pdf})/2$
- **t_r**: *rise time*
  - From output crossing 0.1 V_{DD} to 0.9 V_{DD}
- **t_f**: *fall time*
  - From output crossing 0.9 V_{DD} to 0.1 V_{DD}

Delay Definitions

- $\mathbf{t_{cdr}}$: *rising contamination delay*
  - Minimum time from input to rising output crossing $V_{DD}/2$
- $\mathbf{t_{cdf}}$: *falling contamination delay*
  - Minimum time from input to falling output crossing $V_{DD}/2$
- $\mathbf{t_{cd}}$: *average contamination delay(min-time)*
  - Minimum time from input crossing 50% to the output crossing 50%
  - $t_{pd} = (t_{cdr} + t_{cdf})/2$

Simulated Inverter Delay

- Solving differential equations by hand is too hard
- SPICE simulator solves the equations numerically
  - Uses more accurate I-V models too!
- But simulations take time to write



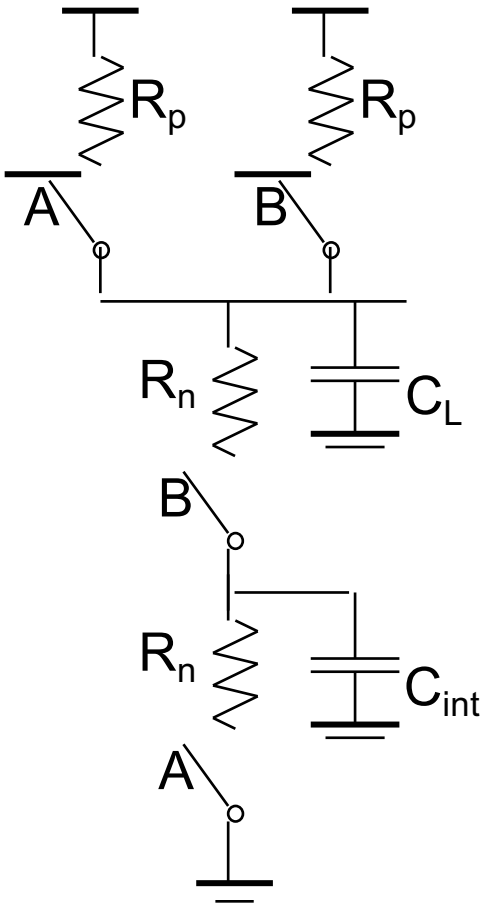
Why we need **estimation**?

- We have timing analyzer at different levels
  - The architectural/micro-architectual level
  - Logic level
  - Circuit level
  - Layout level
- GIGO(Garbage In Garbage Out)!
- Simulation could only tell how fast…, it could not tell how to modify the circuit

Delay Estimation

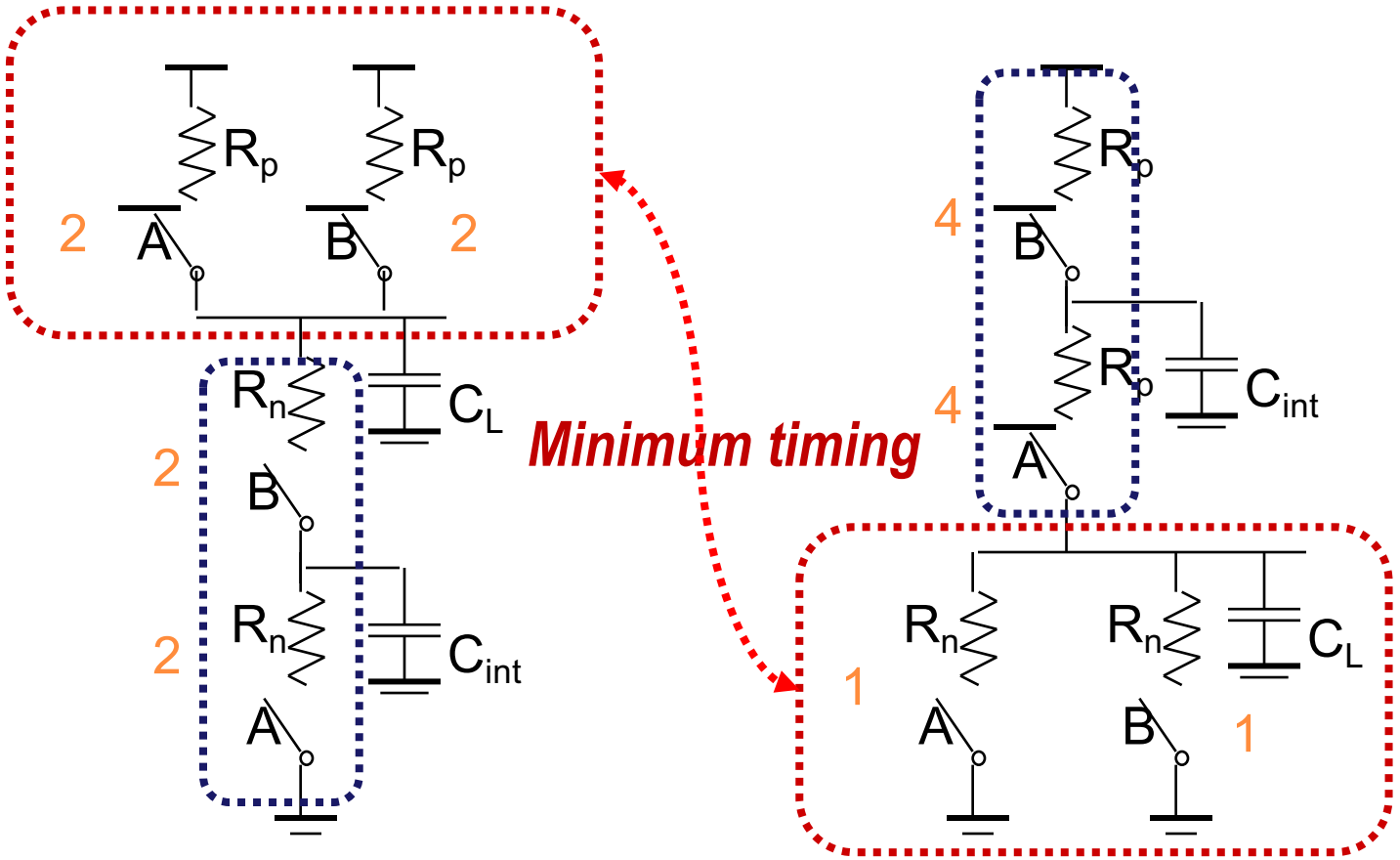
- We would like to be able to easily estimate delay
  - Not as accurate as simulation
  - But easier to ask “What if?”
- The step response usually looks like a 1st order RC response with a decaying exponential.
- Use RC delay models to estimate delay
  - C = total capacitance on output node
  - Use *effective resistance* R
  - So that t_{pd} = RC
- Characterize transistors by finding their effective R
  - Depends on average current as gate switches

Input Pattern Effects on Delay



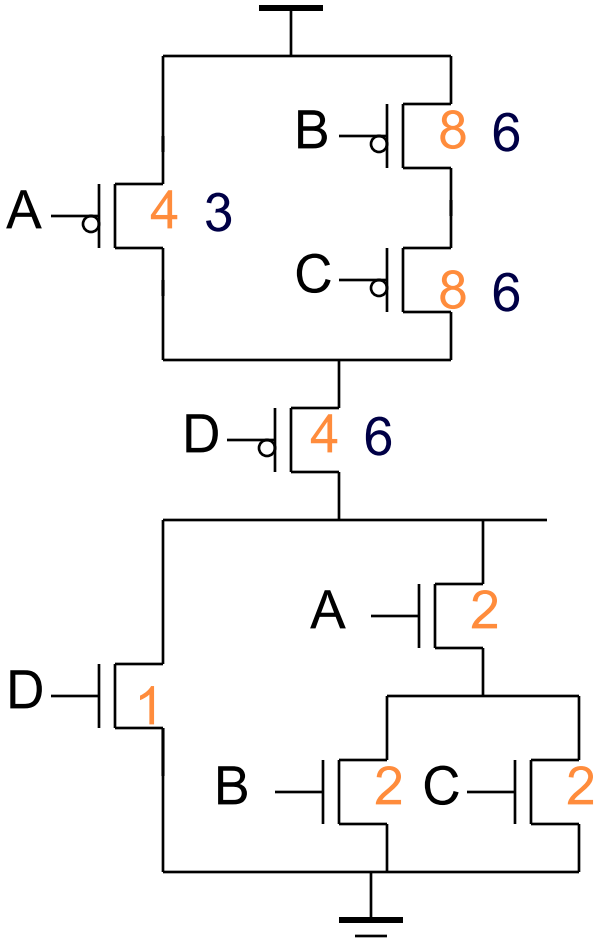
- Delay is dependent on the **pattern** of inputs
- Low to high transition
  - **both inputs** go low
    - delay is 0.69 $R_p/2$ C_L
  - **one input** goes low
    - delay is 0.69 R_p C_L
- High to low transition
  - both inputs go high
    - delay is 0.69 $2R_n$ C_L

Transistor Sizing



Balance between Pullup and Pulldown Network

Sizing has different options

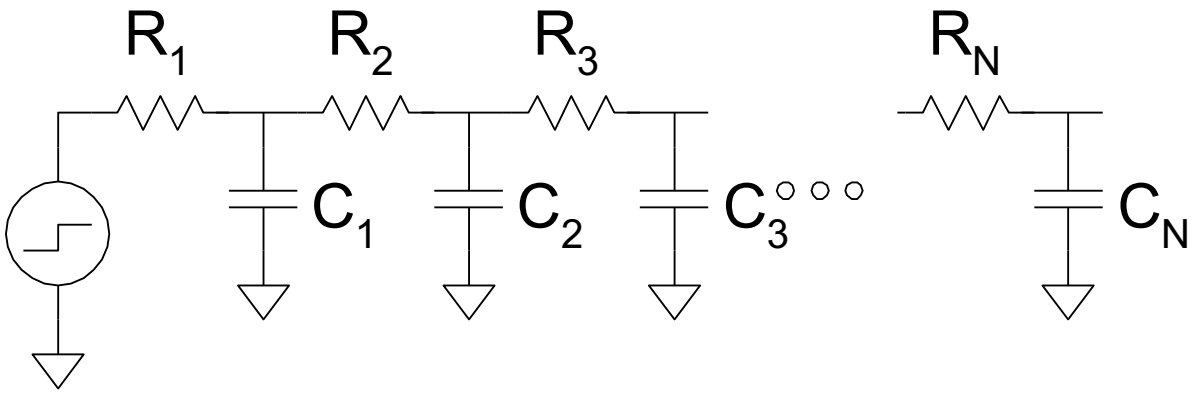


$$OUT = \overline{D + A \bullet (B + C)}$$

Elmore Delay

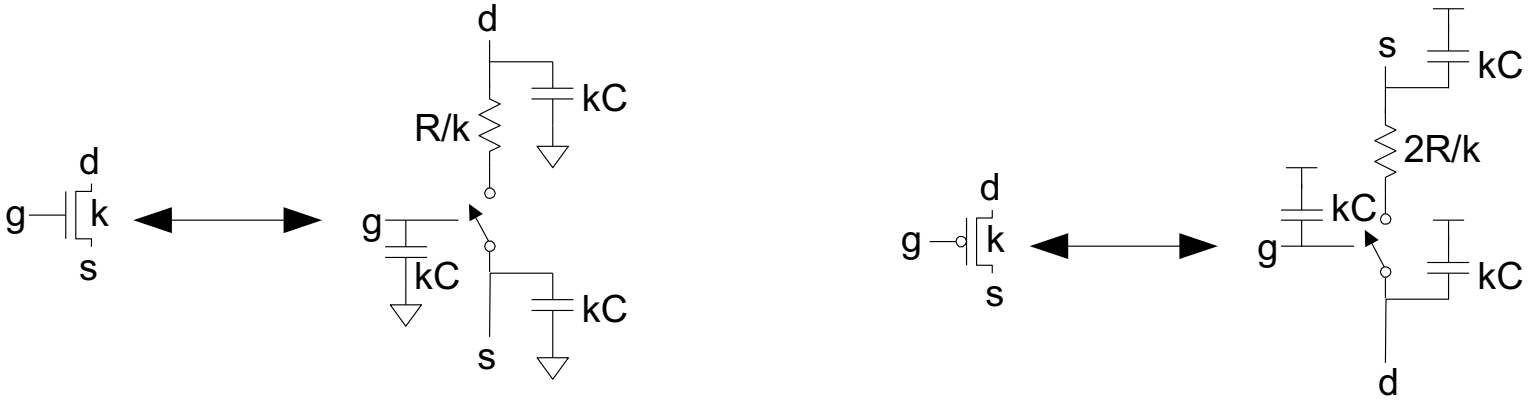
- ON transistors look like resistors
- Pullup or pulldown network modeled as *RC ladder*
- Elmore delay of RC ladder

$$t_{pd} \approx \sum_{\text{nodes } i} R_{i-to-source} C_i$$
$$= R_1 C_1 + (R_1 + R_2) C_2 + \ldots + (R_1 + R_2 + \ldots + R_N) C_N$$



RC Delay Models

- Use equivalent circuits for MOS transistors
  - Ideal switch + capacitance and ON resistance
  - Unit nMOS has resistance R, capacitance C
  - Unit pMOS has resistance 2R, capacitance C
- Capacitance proportional to width
- Resistance inversely proportional to width

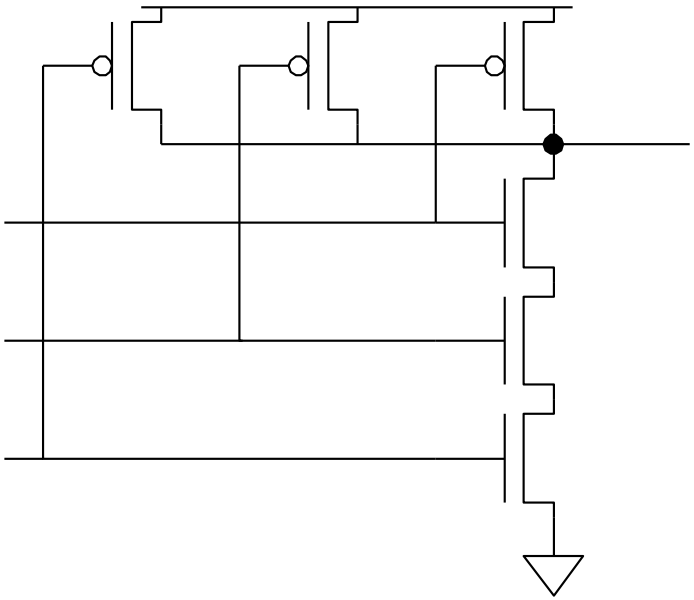


Example: 3-input NAND

- Sketch a 3-input NAND with transistor widths chosen to achieve effective rise and fall resistances equal to a unit inverter (R)

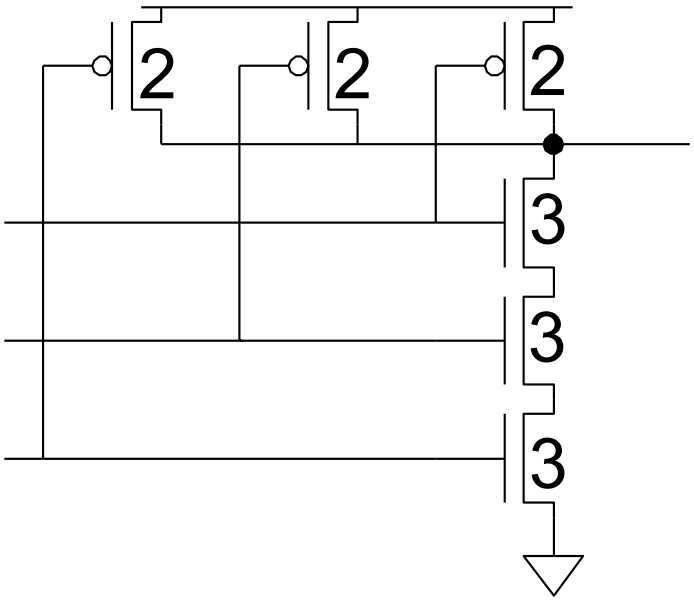
Example: 3-input NAND

- Sketch a 3-input NAND with transistor widths chosen to achieve effective rise and fall resistances equal to a unit inverter (R)



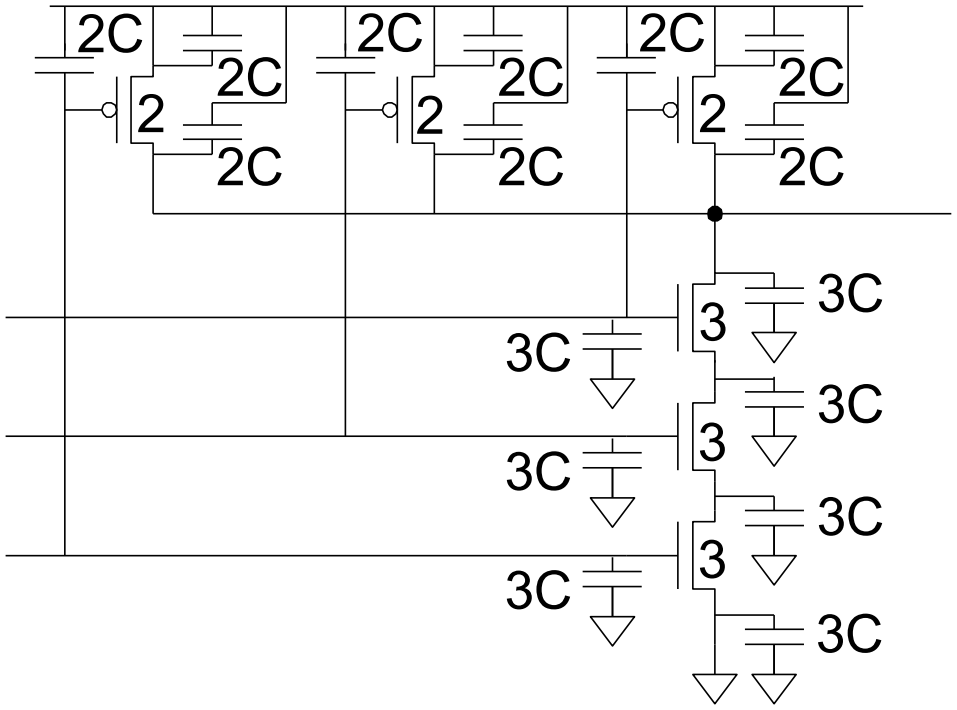
Example: 3-input NAND

- Sketch a 3-input NAND with transistor widths chosen to achieve effective rise and fall resistances equal to a unit inverter (R)



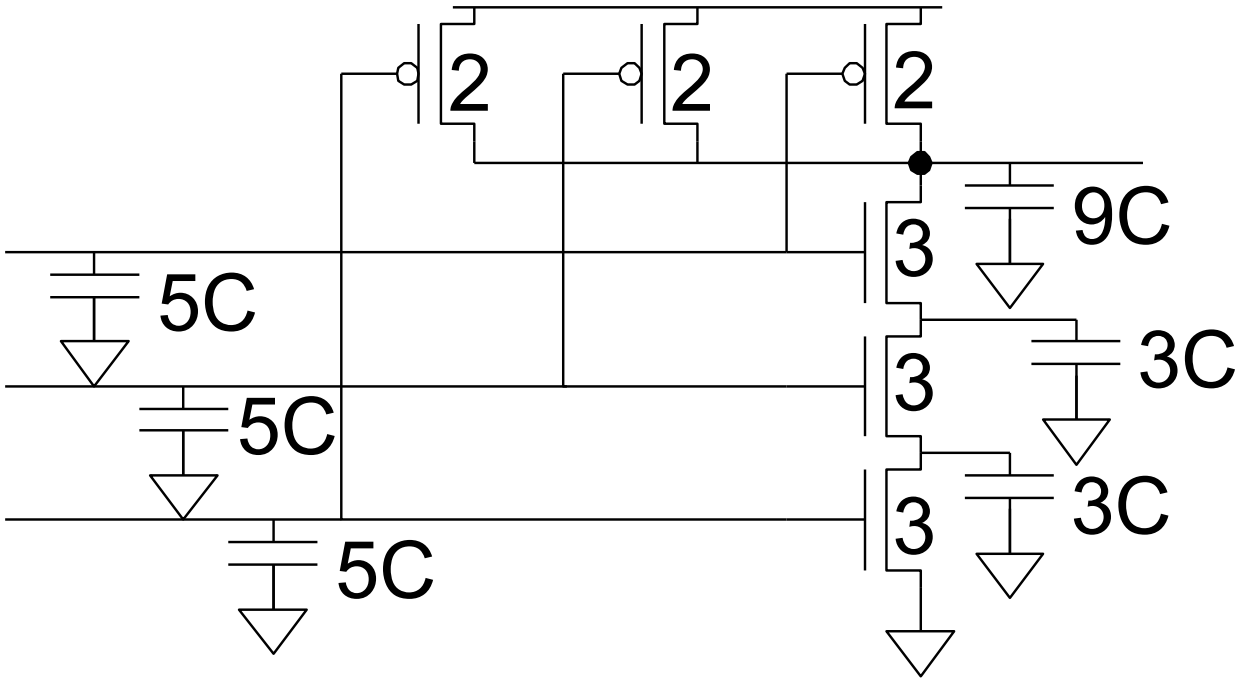
3-input NAND Caps

- Annotate the 3-input NAND gate with gate and diffusion capacitance.



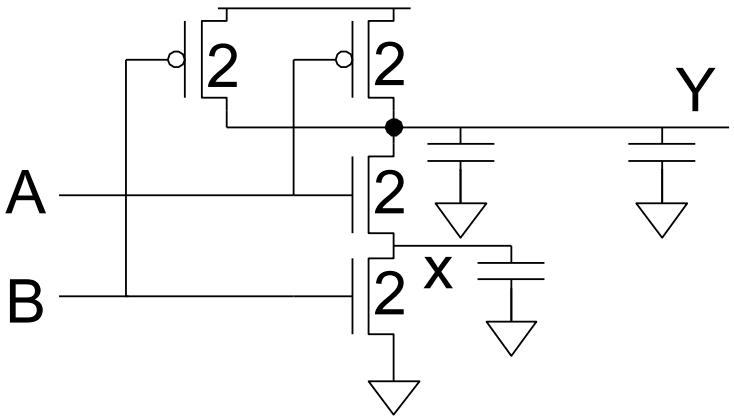
3-input NAND Caps

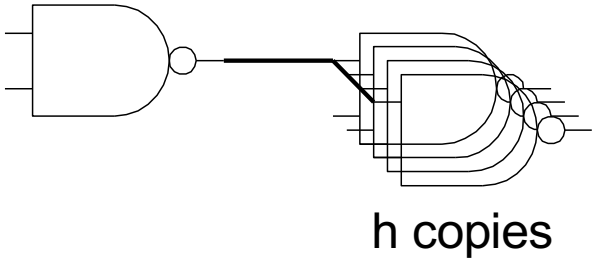
- Annotate the 3-input NAND gate with gate and diffusion capacitance.



Example: 2-input NAND

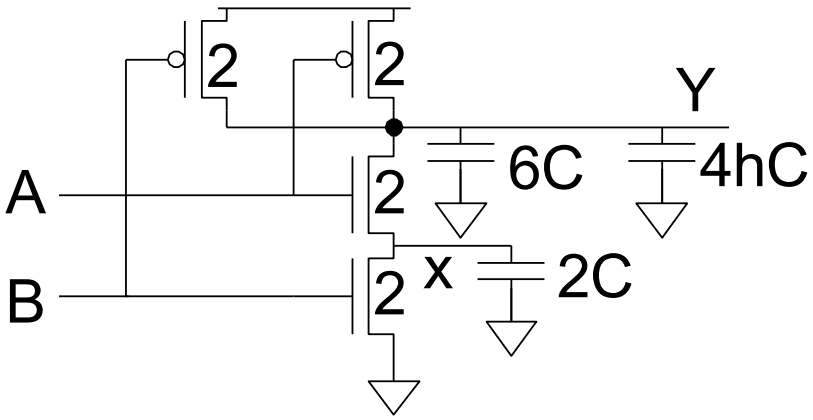
- Estimate worst-case rising and falling delay of 2-input NAND driving *h* identical gates.

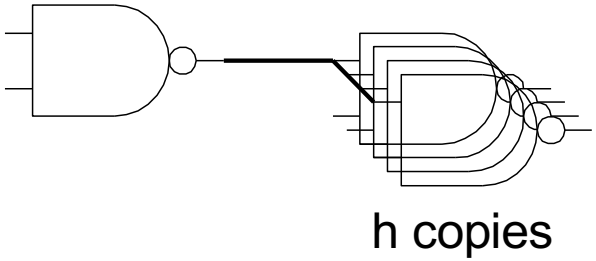




Example: 2-input NAND

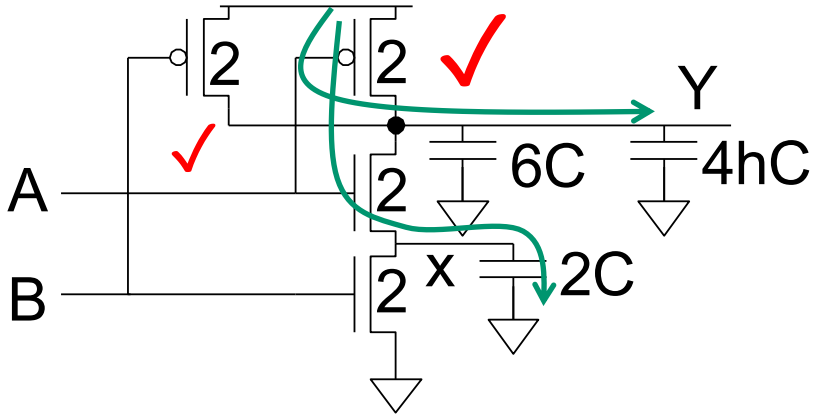
- Estimate rising and falling propagation delays of a 2-input NAND driving *h* identical gates.

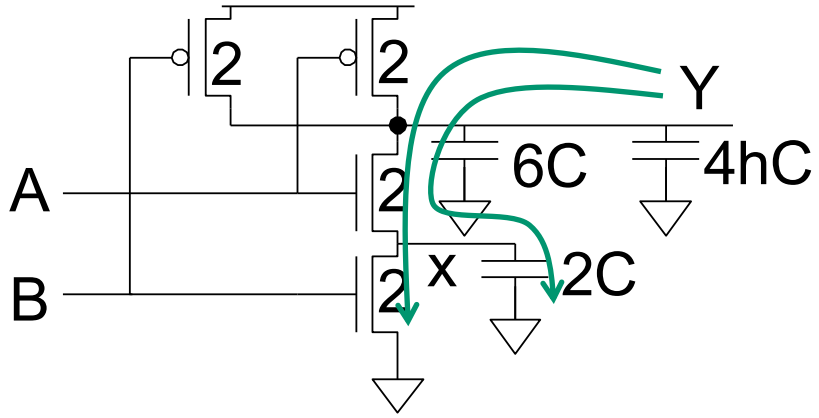




Example: 2-input NAND

Estimate rising and falling propagation delays of a 2-input NAND driving *h* identical gates.

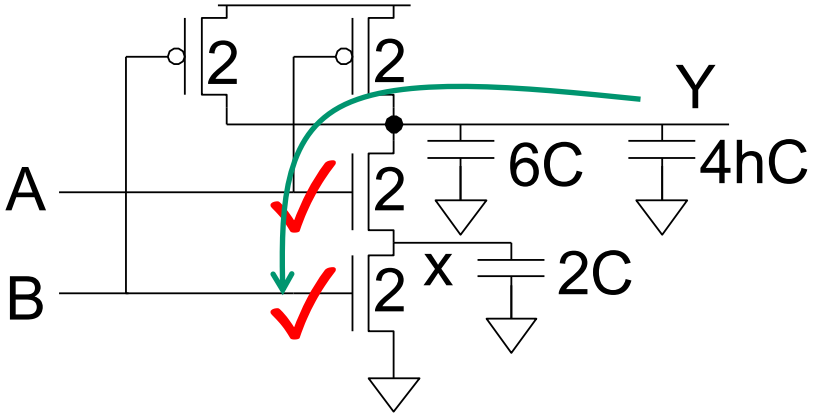


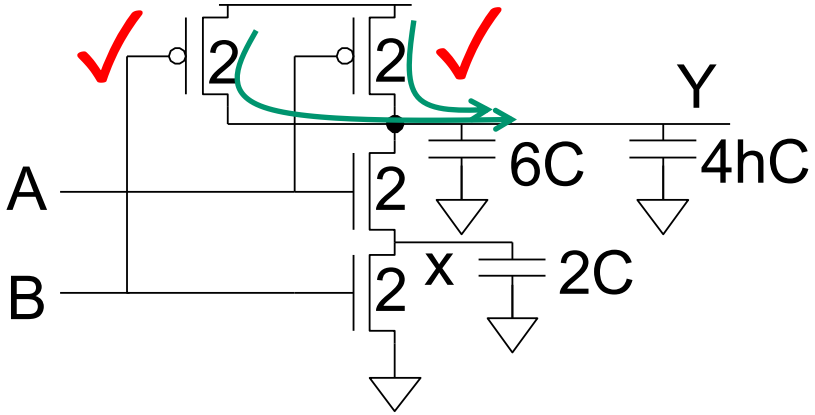


$$t_{pLH} = ln2((6+4h)CR + 2CR) \qquad t_{pHL} = ln2((6+4h)CR + 2C\frac{R}{2})$$

Contamination Delay

- Best-case (contamination) delay can be substantially less than propagation delay.
- Ex: If both inputs fall simultaneously





$$t_{cHL} = ln2((6 + 4h)CR)$$

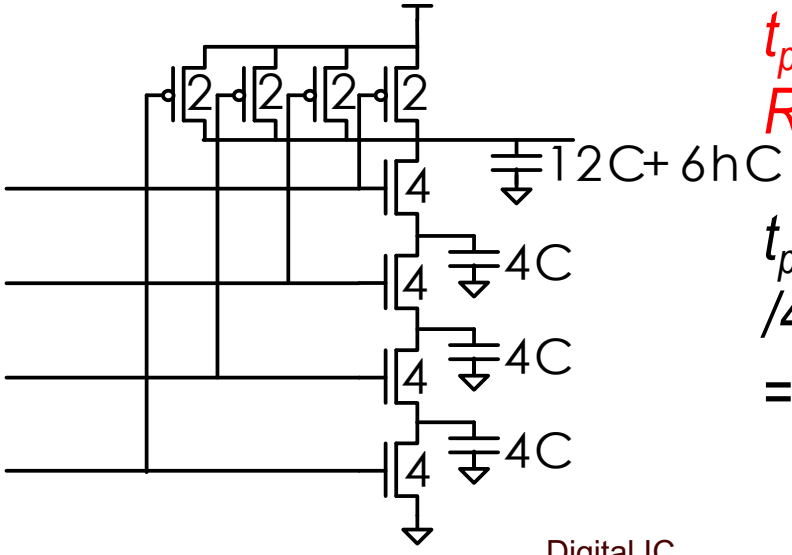
$$t_{cLH} = ln2((6 + 4h)C\frac{R}{2})$$

Practice:

1. Sketch a 4-input NAND gate with transistor widths chosen to achieve effective rise and fall resistance equal to a unit inverter.

- *Compute the t_{LH} and t_{HL} propagation delays(in terms of R and C) of the NAND gate driving h identical NAND gates using the ELMORE delay model*

A:

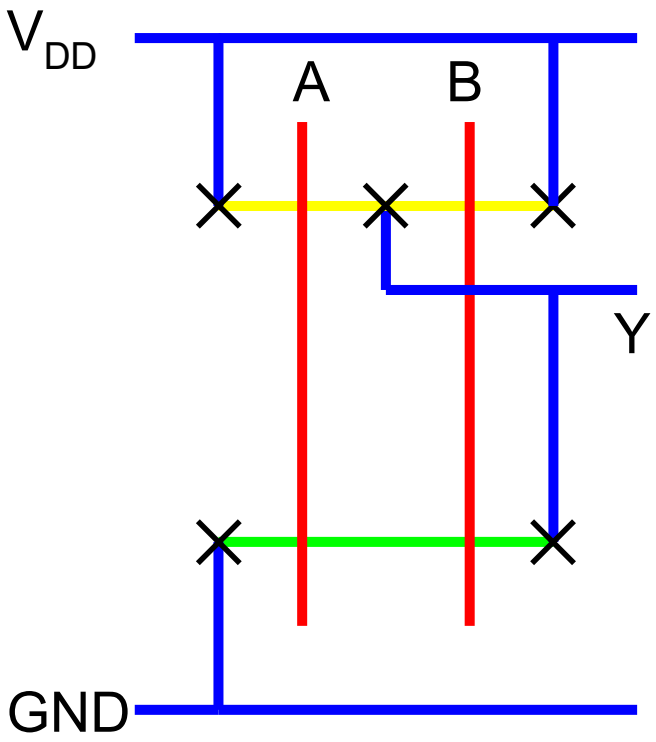


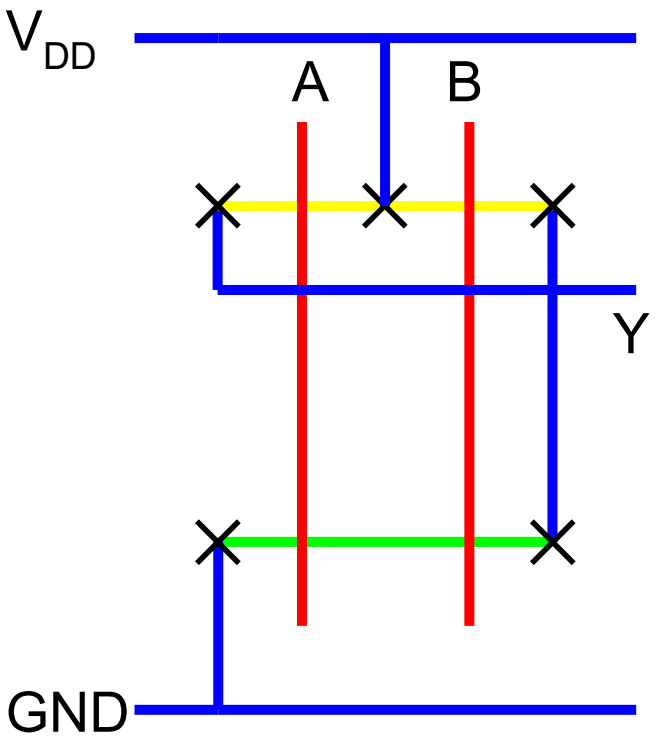
t_{pLH}=ln2[4C*R+4C*R+4C*R+ (12C+6Ch) R*]=RCln2[24+6h]

t_{pHL}=ln2[4C*R/4+2R/4*4C+3R/4*4C+4R /4*(12C+6Ch)]=RCln2[1+2+3+(12+6h)] =ln2(18+6h)

Layout Comparison

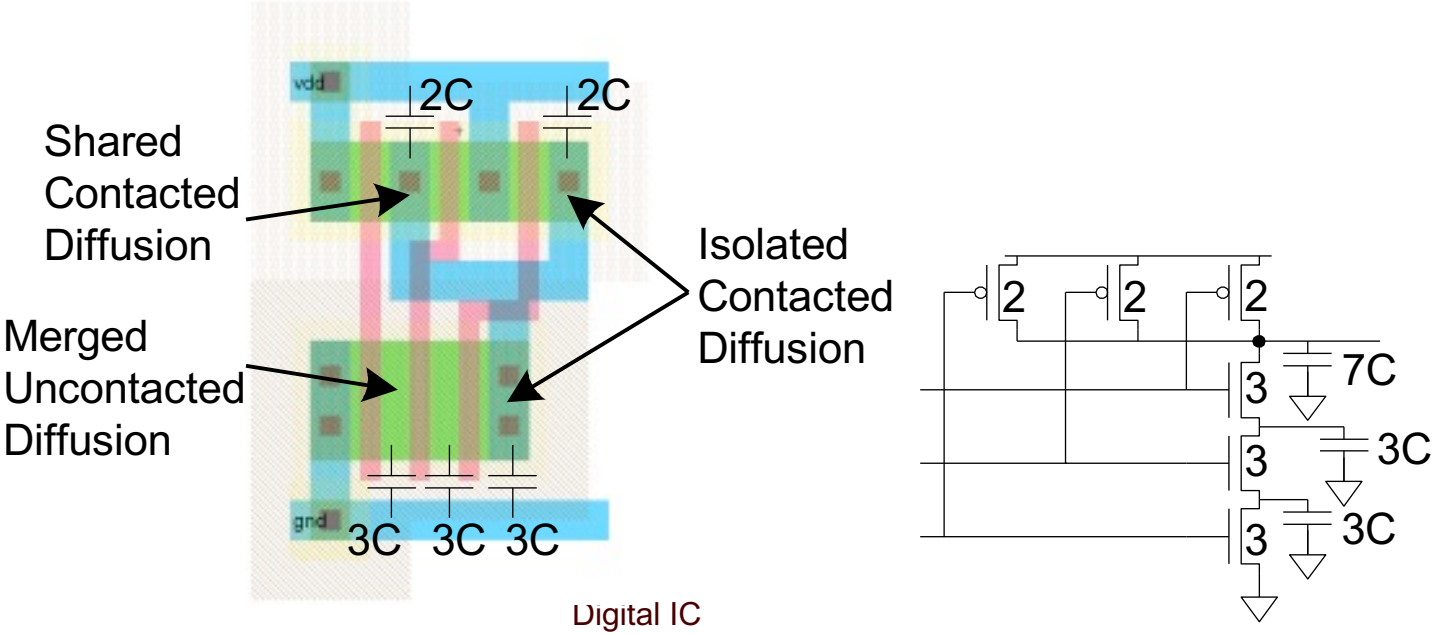
- Which layout is better?



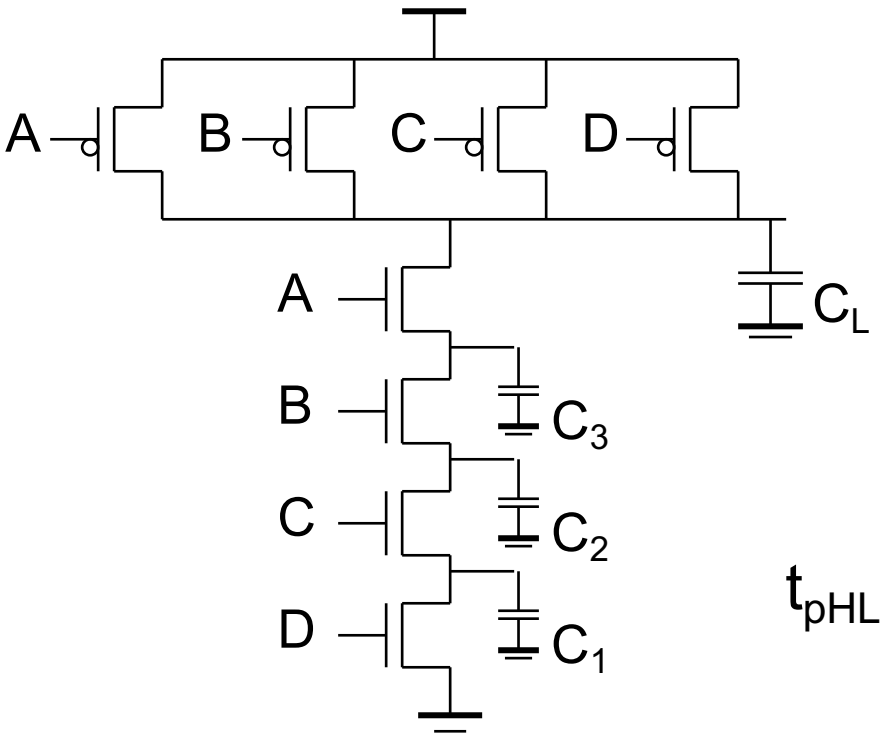


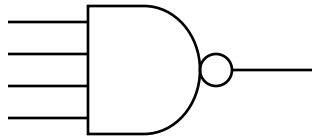
Diffusion Capacitance

- assumed contacted diffusion on every s / d.
- Good layout minimizes diffusion area
- Ex: NAND3 layout shares one diffusion contact
  - Reduces output capacitance by 2C
  - Merged uncontacted diffusion might help too



Fan-In Considerations



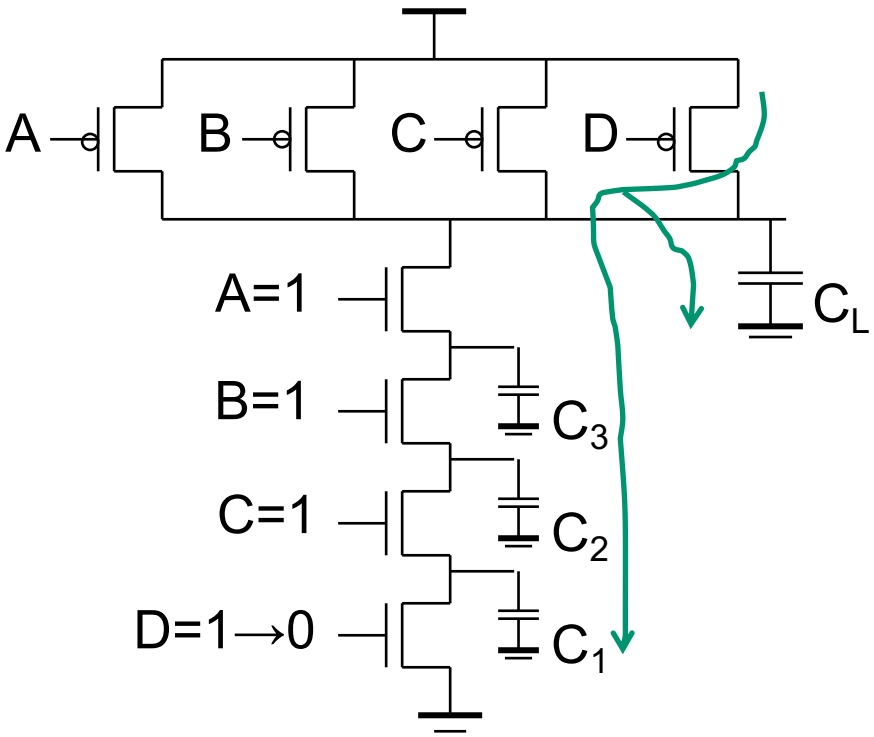


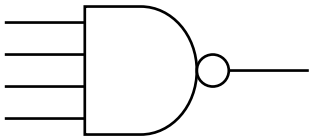
Distributed RC model
(Elmore delay)

$$t_{pHL} = 0.69\, R_{eqn}(C_1+2C_2+3C_3+4C_L)$$

Propagation delay deteriorates rapidly as a function of fan-in quadratically in the worst case

Worst case of t_{pLH}





$$\tau = 2R/2*(2NC+C_{ext})+2R/2*NC*N=2NRC+RC_{ext}+N^2RC$$

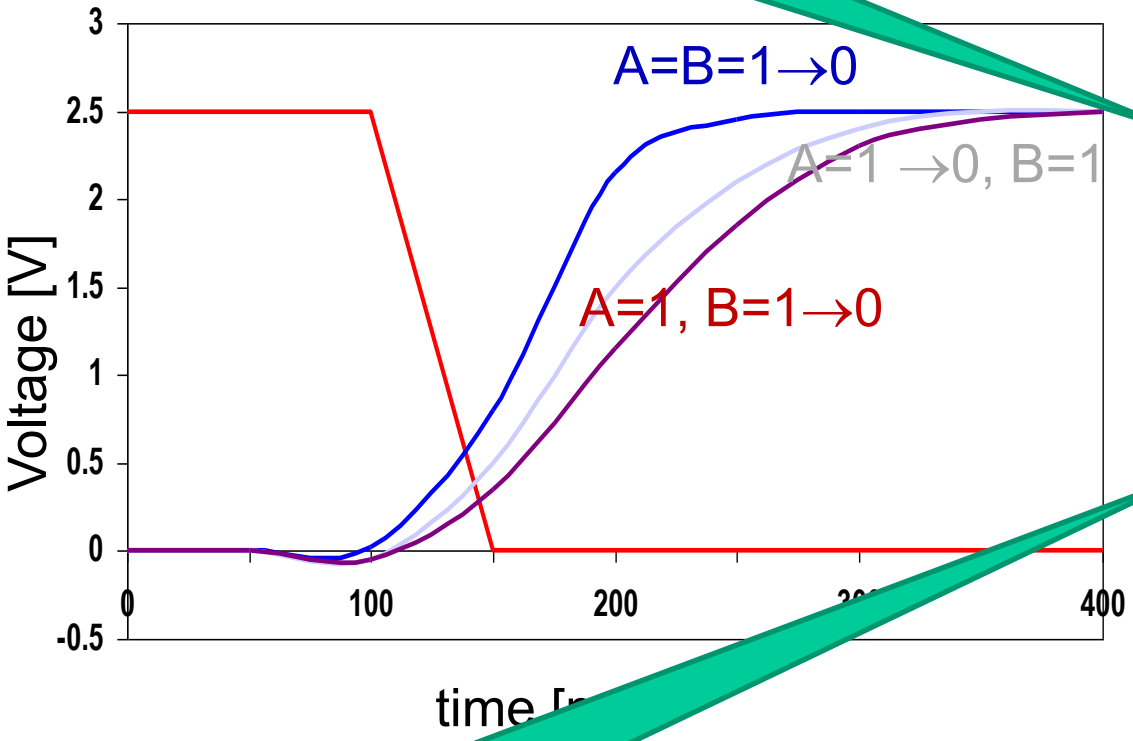
t_p as a function of fan-In and fan-out

- Fan-in: quadratic due to increasing resistance and capacitance
- Fan-out: each additional fan-out gate adds two gate capacitances to C_L

$$t_p = a_1 F_I + a_2 F_I^2 + a_3 F_O$$

Delay Dependence on Input Patterns

Shared cap. **discharging**



Shared cap. **charging**

Input Data Pattern	Delay (psec)
A=B=0→1	69(max)
A=1, B=0→1	62
A= 0→1, B=1	50
A=B=1→0	35(min)
A=1, B=1→0	76
A= 1→0, B=1	57

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

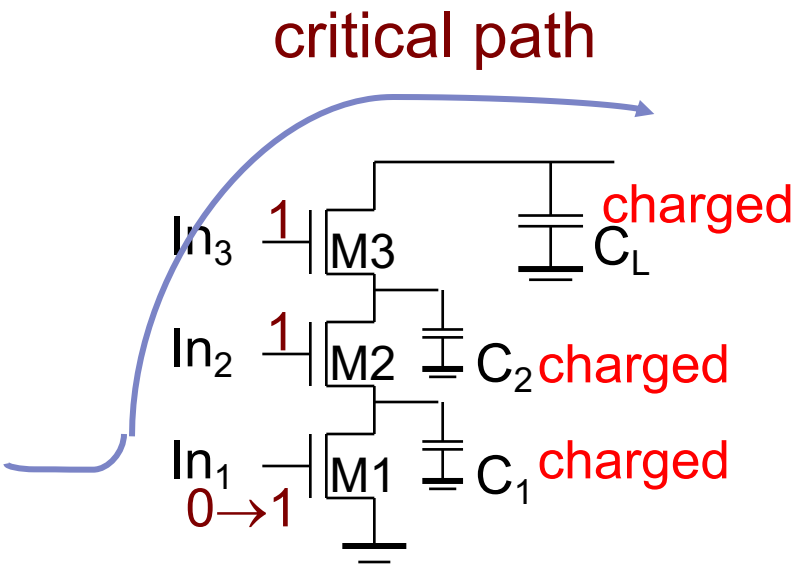
Static CMOS logic

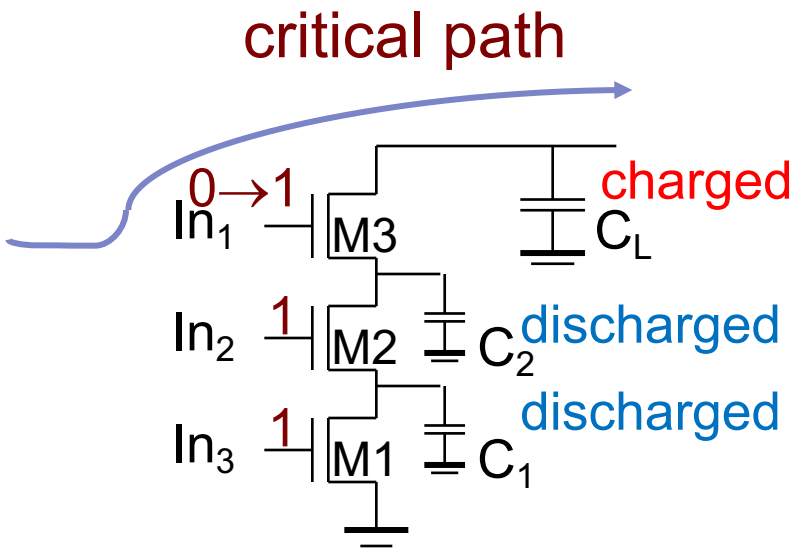
- *CMOS static characteristic*
- *CMOS propagate delay*
- ***Large fan-in technology***
- *Logic effort*
- *CMOS power analysis*

How to choose design techniques for large fan-in

- **Larger parasitic capacitor, larger load to the preceding gate**
- **Load is dominated by fan-out, the design technique makes sense**
- Solution
  - Progressive transistor sizing
  - Transistor ordering
  - Alternative logic structures
  - Isolating fan-in from fan-out using buffer insertion

Transistor ordering



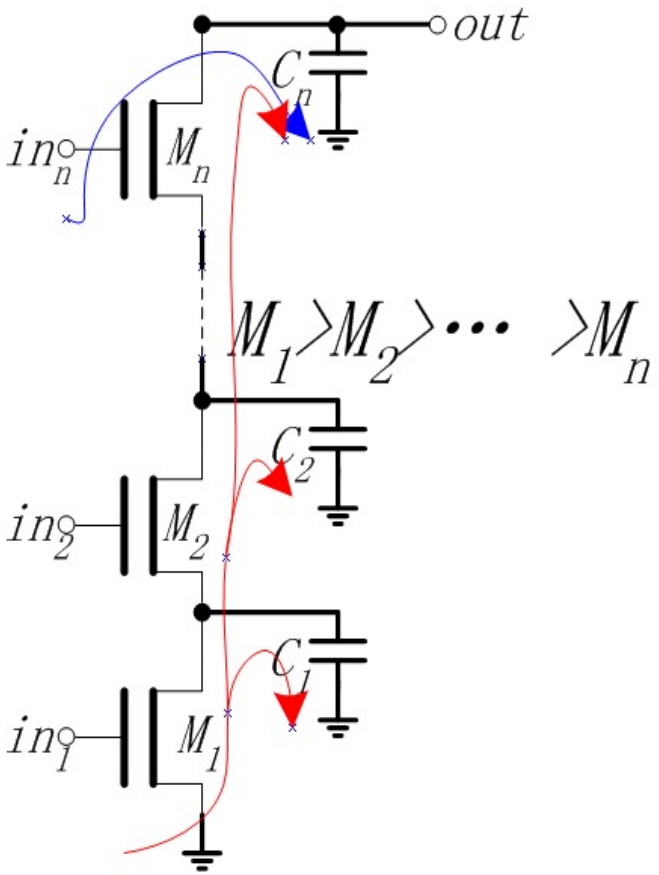


delay determined by time to discharge C_L, C_1 and C_2

delay determined by time to discharge C_L

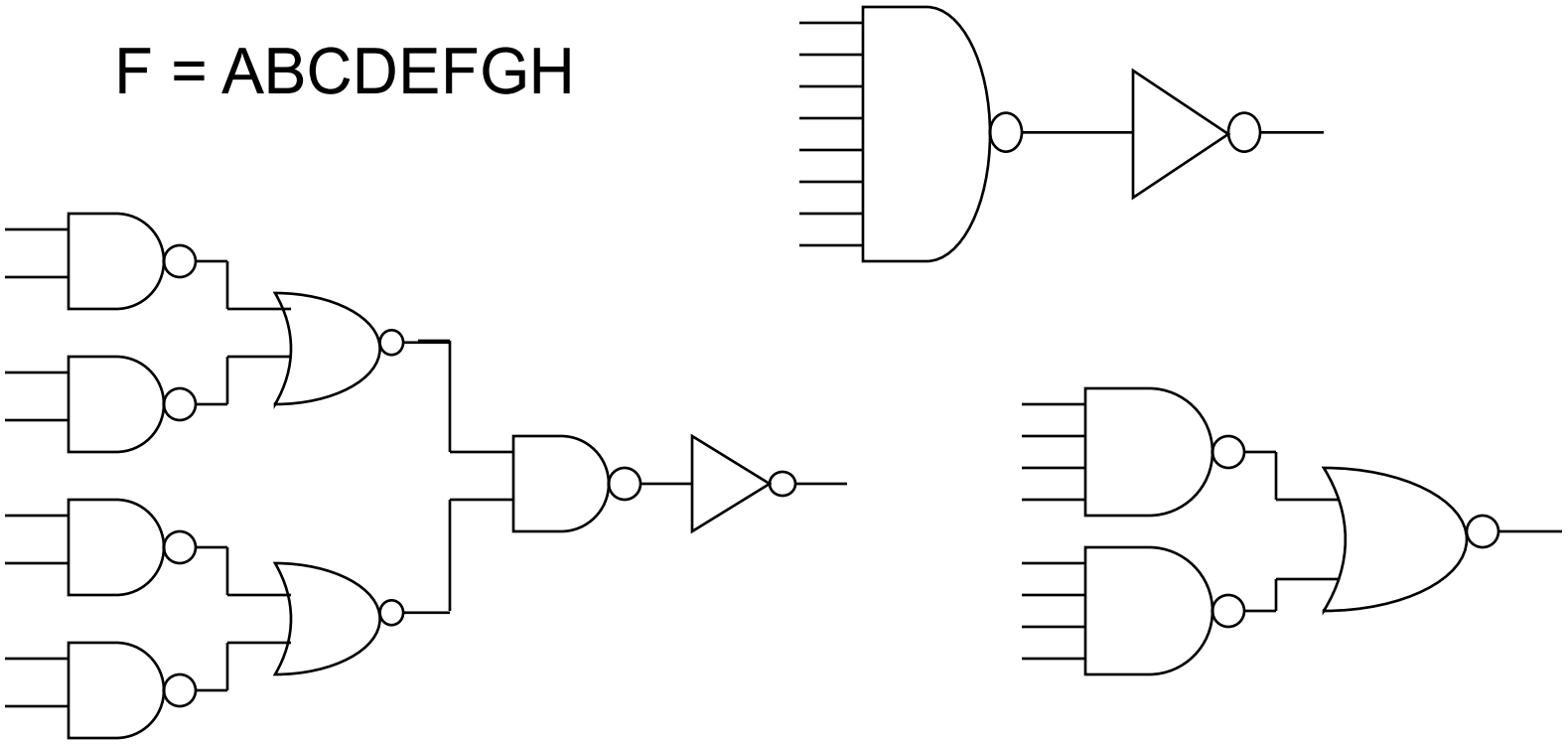
Progressive sizing

- Distributed RC line
  - $M_1 > M_2 > M_3 > \ldots > M_N$
- the closest to the output is the smallest
- Can reduce delay by more than 20%; decreasing gains as technology shrinks



Alternative logic structures

F = ABCDEFGH



Isolating fan-in from fan-out using buffer insertion