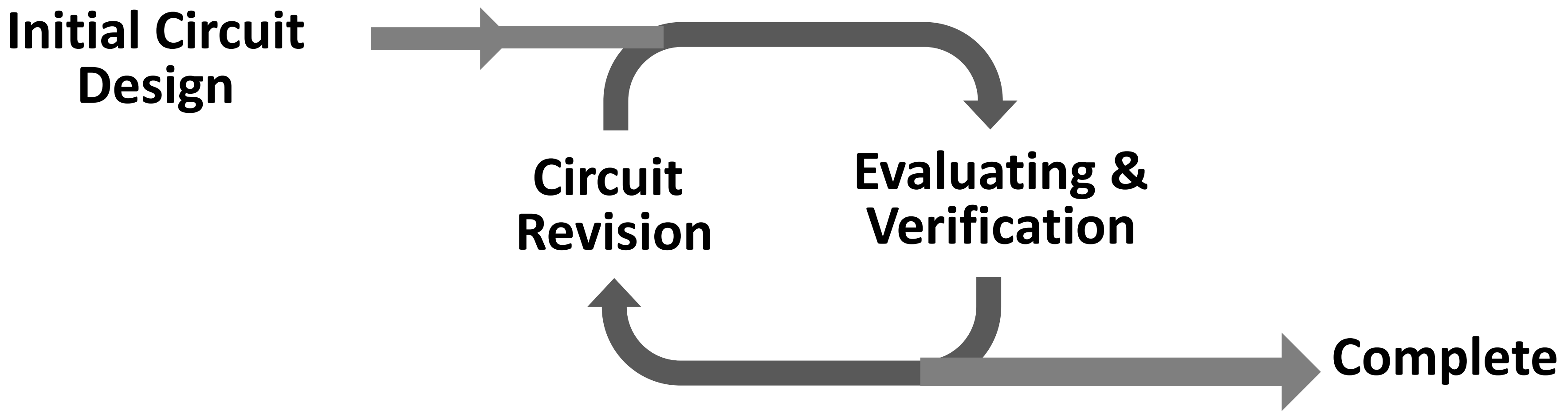
How Do You Think IC Design Flow is Like?

- There must be a target circuit spec that should be satisfied.
- How can we design a circuit that meet such spec?



- How can you evaluate and verify circuit design?

Make An IC Chip!

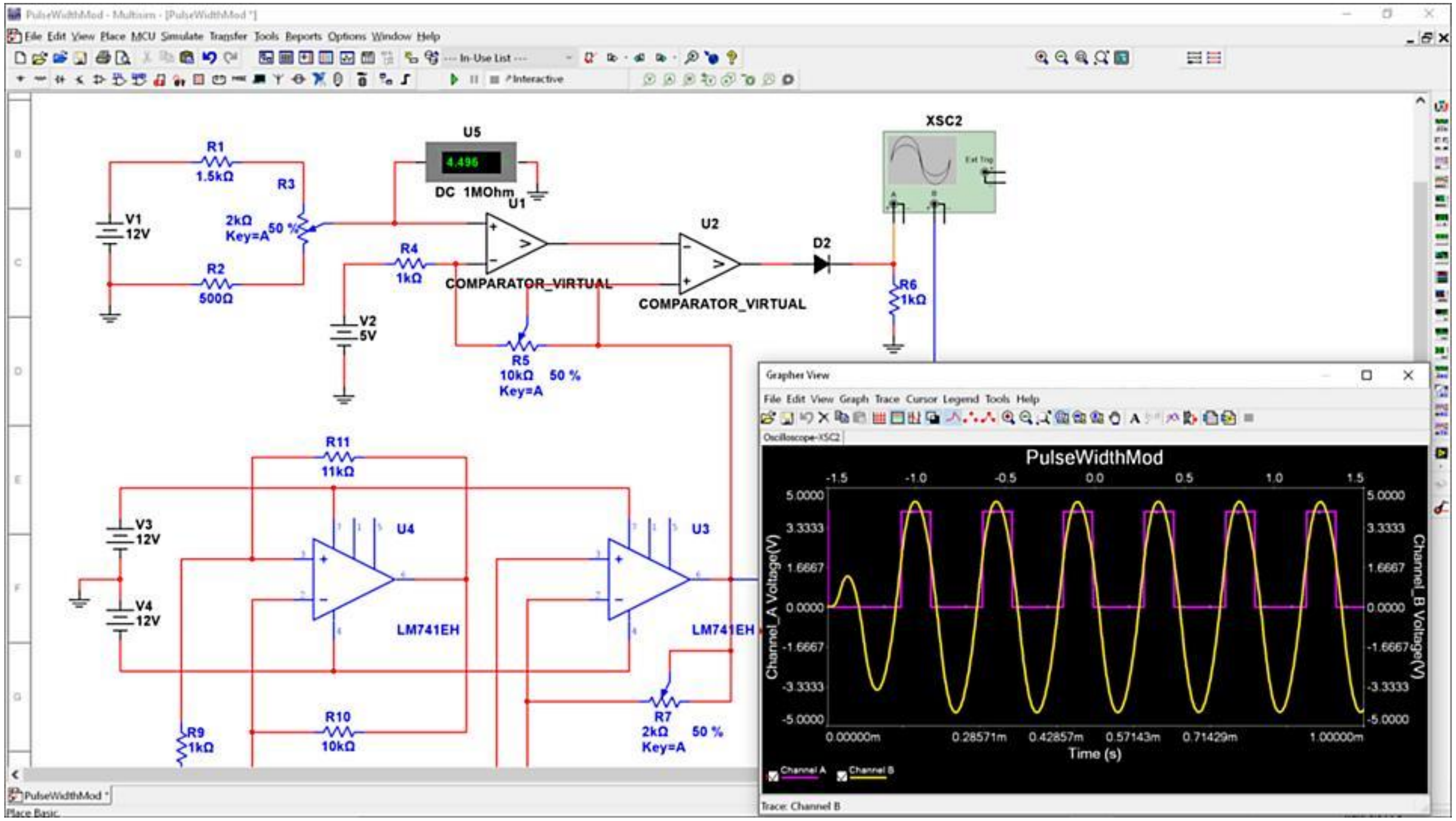
- How long does it take to get an IC chip?

➔Typically 6-8 weeks at least

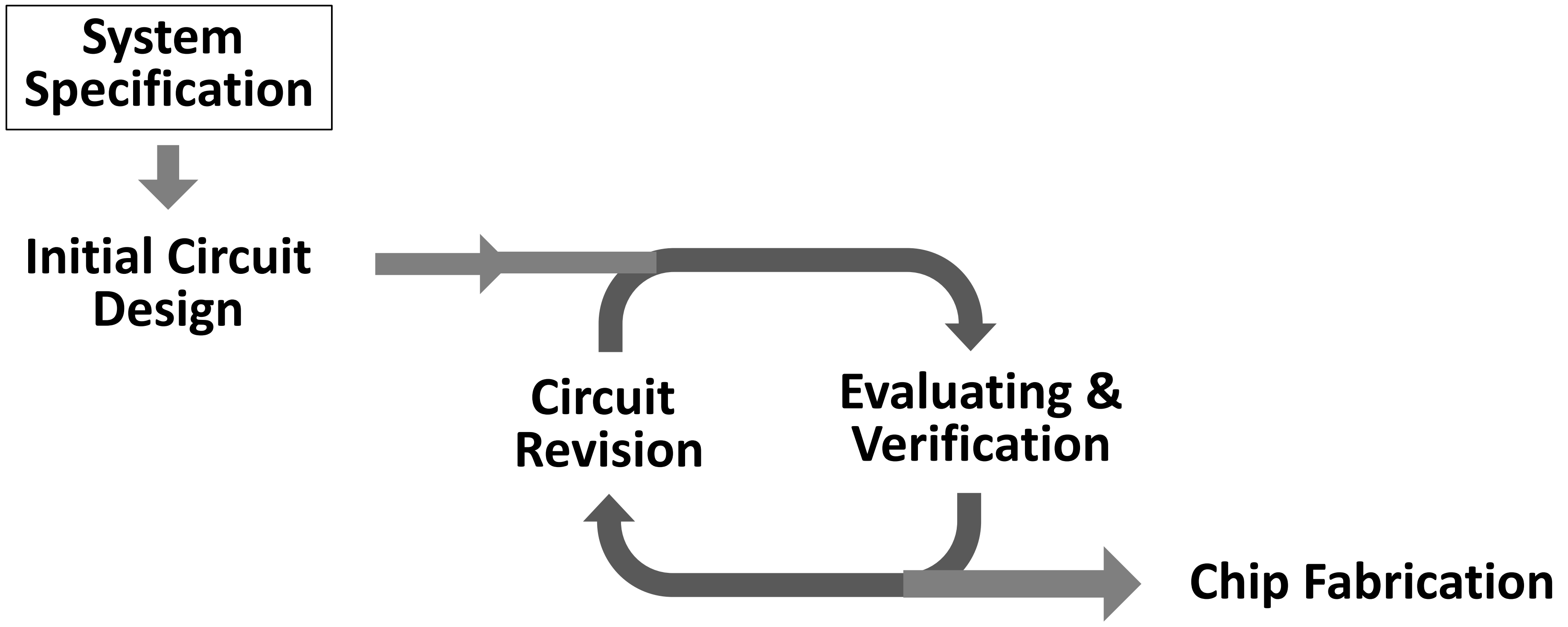
- How much does it cost?

How Can We Predict Circuit Performance?

- Circuit simulation do work but



Typical IC Design Flow



Role of Circuit Designer

- A circuit designer should be able to estimate the circuit performance analytically without simulation.
- It does not mean you should estimate the circuit performance 100% accurate in quantitative respect. (Simulation or chip measure can do this)
- Instead, you can say which is better design and why it is better to make a decision without simulation.
- Then, you can solve the problem in circuit design area.

➔ Is there any tool that can give circuit designers a good insight on circuit speed?

Delay

Hanwool Jeong

hwjeong@kw.ac.kr

Contents

Introduction

- ✓ How can we quantify the circuit speed?

Revisit Source of Delay

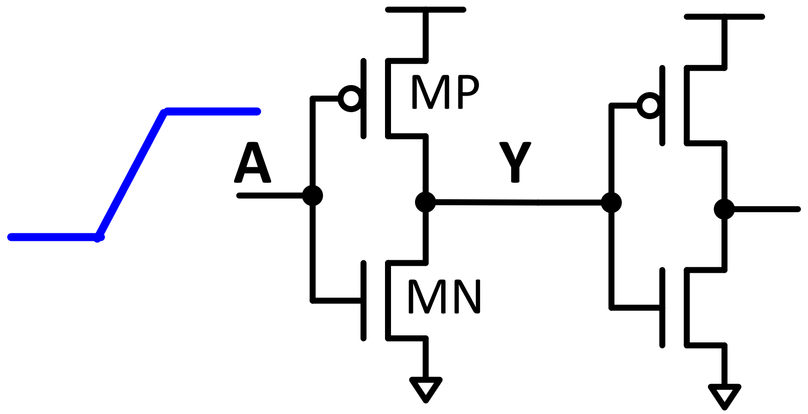
- A circuit operation takes time because
  - Capacitance exists everywhere in a real circuit and
  - The capacitance cannot change its voltage instantaneously
- If a capacitance C is changed/discharged with a current I,

$$I = C\, dV/dt$$

- How can we determine the delay in a circuit?

Example; Transient Response in An Inverter Drives an Inverter

- When the rising input is applied to A, Y would fall



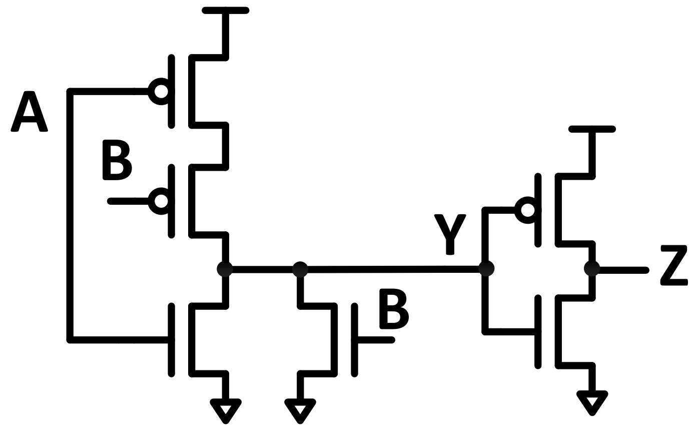
- To derive the A-Y delay, we can find **V_Y vs. time**
  ➔ **Transient response**
  - Derive the capacitance of Y node, C_Y
  - Then, $C_Y(dY/dt) = I_{MP} – I_{MN}$
  - According to V_A range and V_Y ranges ➔ Only numerical solution
  - The time when $V_Y = V_{DD}/2$ can be determined as the delay

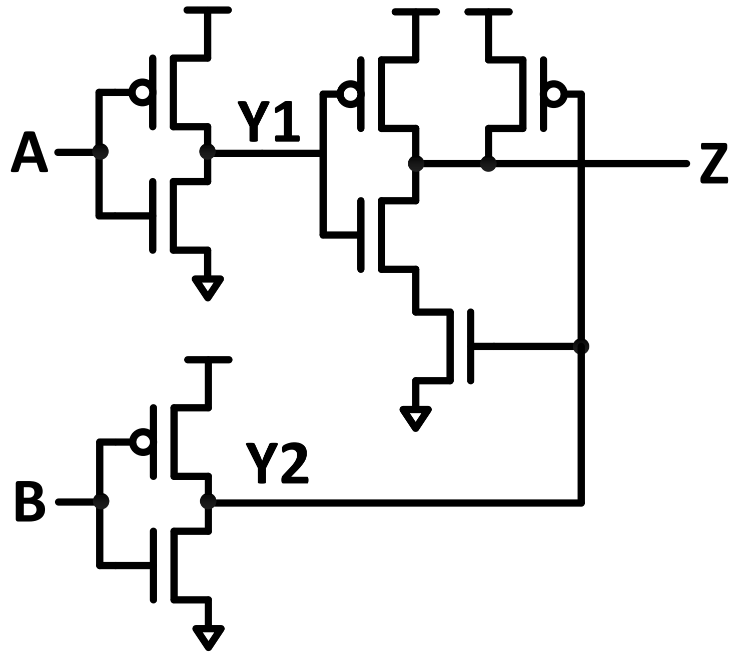
What Really Matters …

- Working out the full model is tedious and offers little insight.
- Leave it to simulation tools to derive an accurate transient response and delay.
- Instead, we need a much simpler model (high efficiency) with endurable accuracy.

Remember?

- They are both OR2. Which is faster?



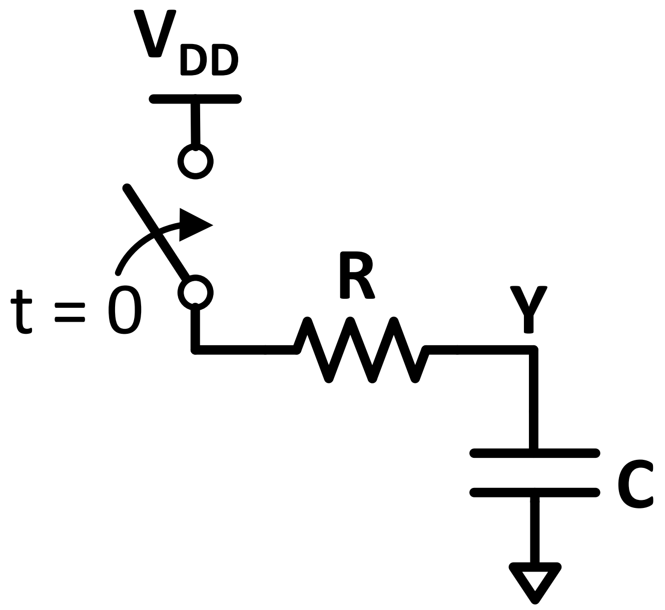


RC Delay Model

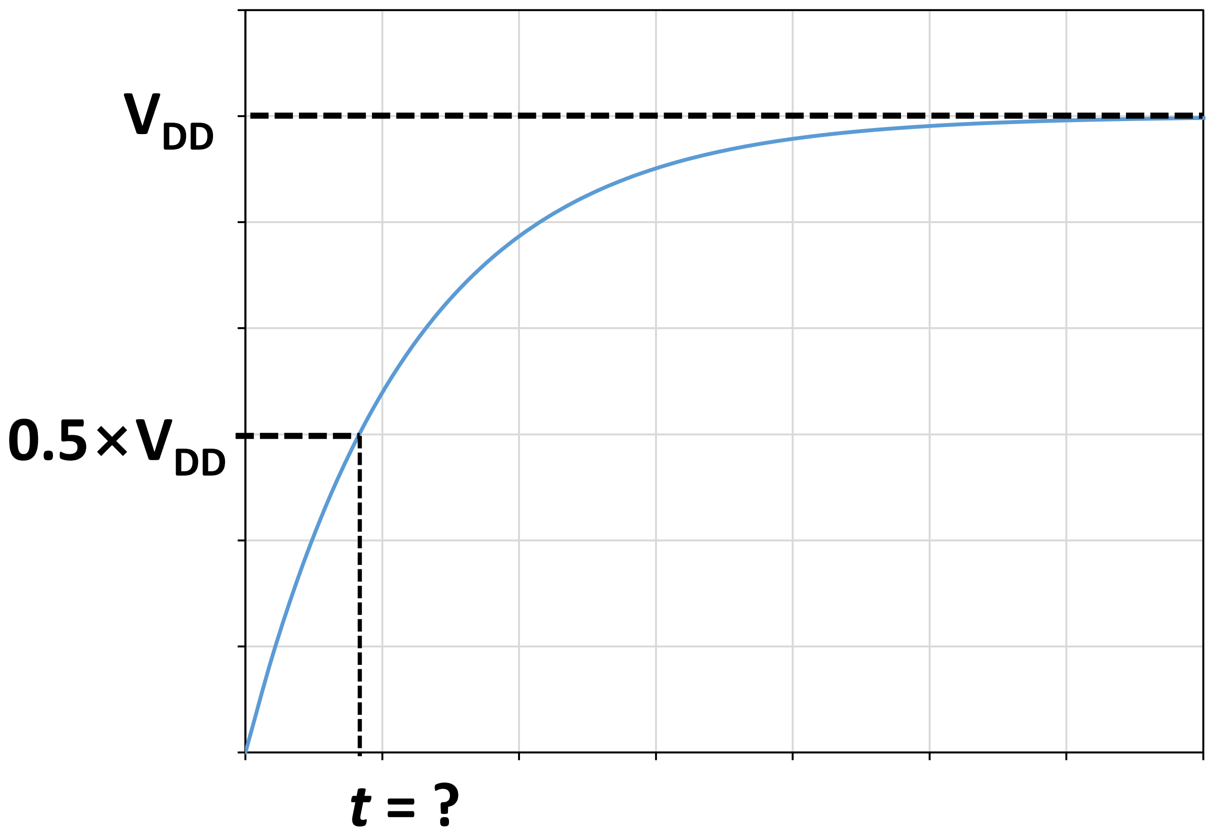
✓ Approximating nonlinear MOSFET characteristics

RC Circuit?

- We can derive the transient response of V_Y in the following 1st order RC circuit, where V_Y is initially 0V.



Delay in From 1st Order RC Circuit

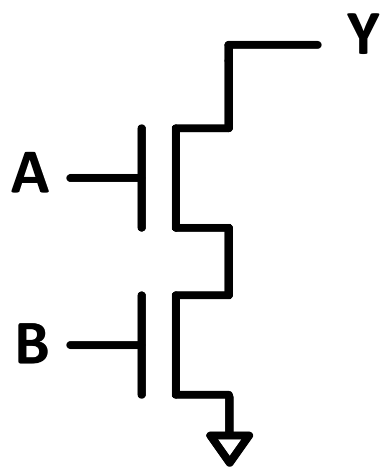


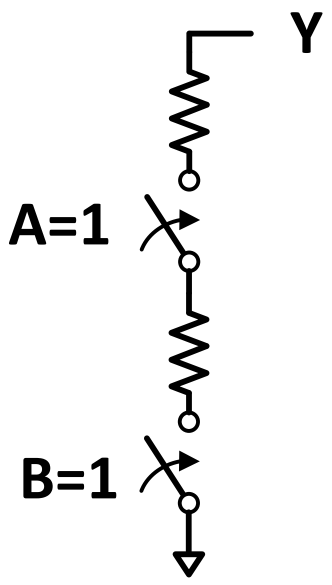
Approximate 1; Effective Resistance

- Let's treat a MOSFET as a conditionally connected resistor.
- And the value of resistance would be **$R \propto L/W$.**
- In many cases, L is fixed with minimum length, thus **$R \propto 1/W$.**

➔ Defining unit-width MOSFET as **R** is useful. Then, when $W = kW_{unit}$ the effective resistance is **R/k.**

Accurate value may not be important. Why?





Approximate 2; Gate and Diffusion Capacitance

- Let's assume that in unit-width MOSFET

$$C_{gate} = C_{drain} = C_{source} = C$$

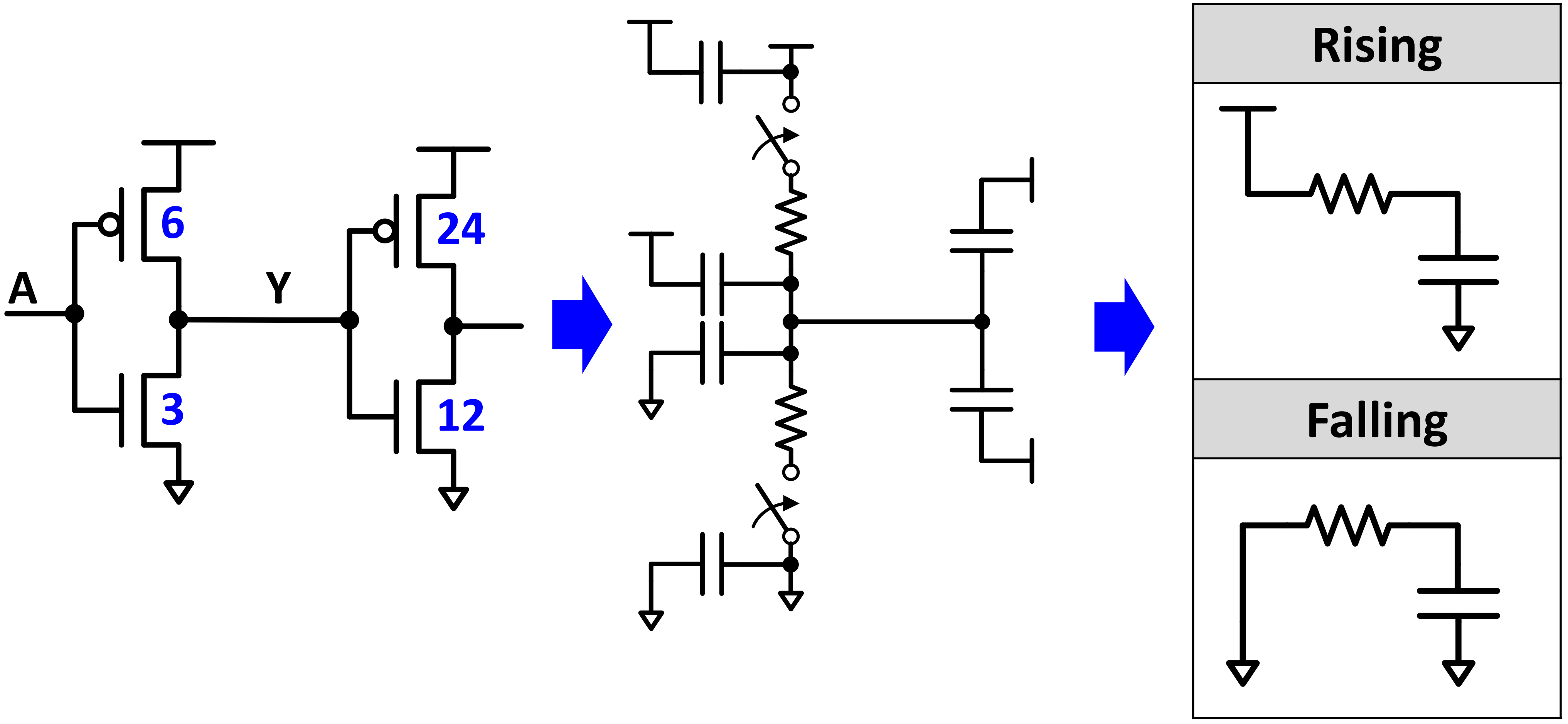
- Then a MOSFET with k time of unit width has

$$C_{gate} = C_{drain} = C_{source} = \mathbf{k}C$$

- What if L is increased?

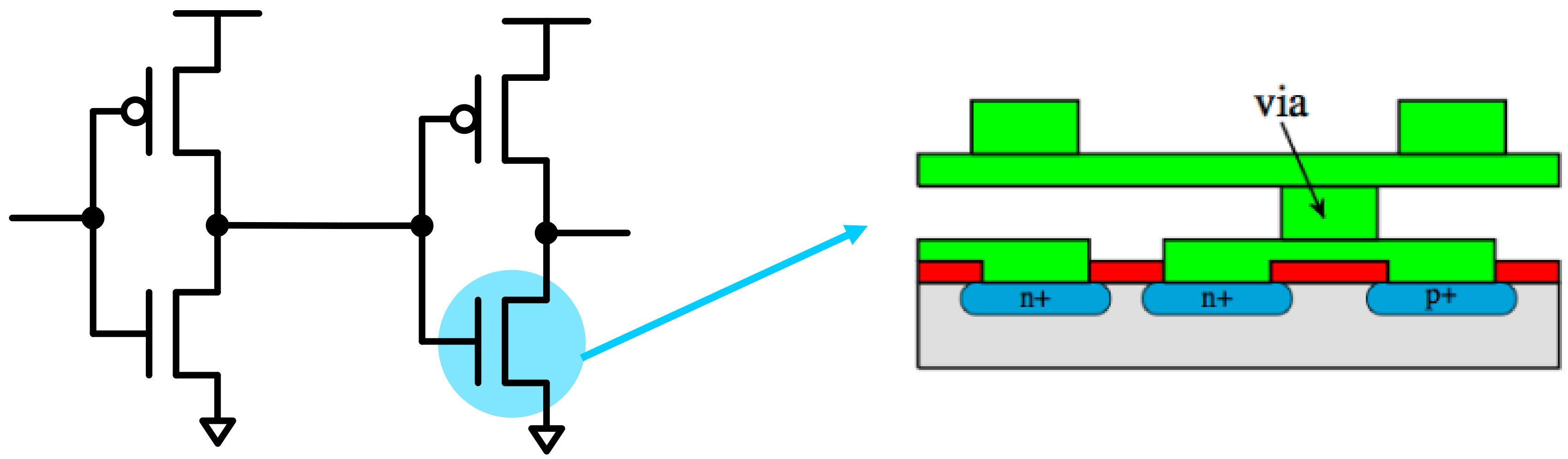
Equivalent RC Circuits; Inverter

- Suppose that we are interested in the time taken to change Y



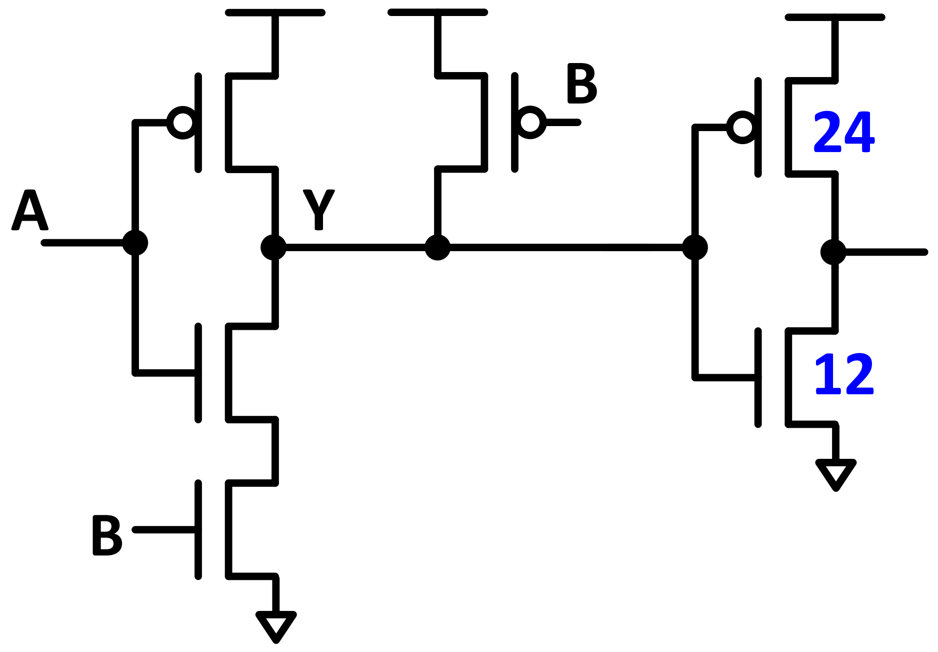
Effect of Wire?

- Wire can have capacitance and resistance as well.

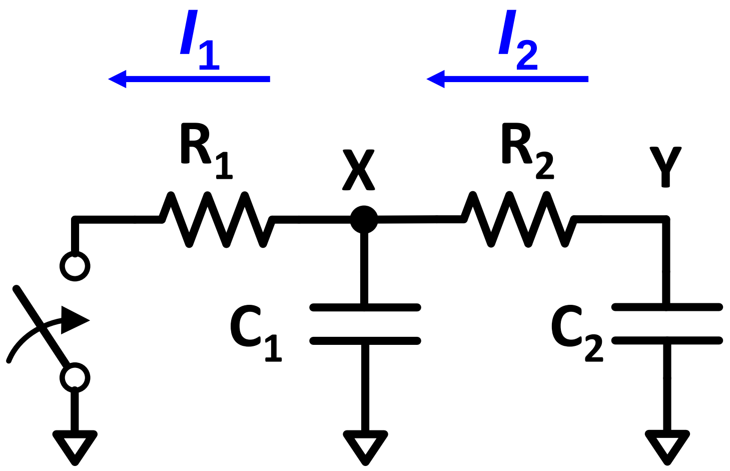


Equivalent RC Circuits; NAND2

- How should pMOSFET and nMOSFET size ratio be?
- What is the equivalent RC model for NAND2 fall delay?



Transient Response of 2nd Order RC System



$$V_{\text{out}}(t) = V_{DD}\frac{\tau_1 e^{-t/\tau_1} - \tau_2 e^{-t/\tau_2}}{\tau_1 - \tau_2}$$

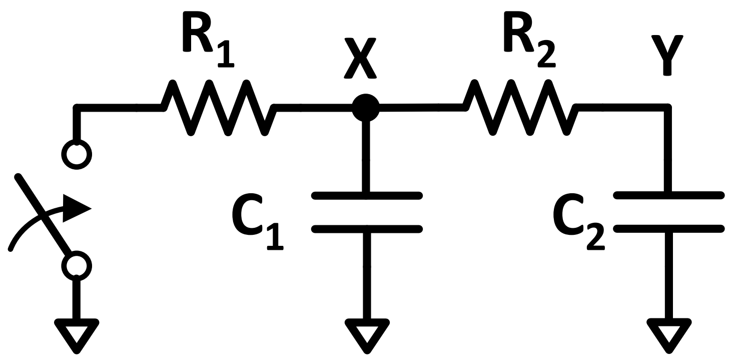
$$\tau_{1,2} = \frac{R_1C_1 + (R_1+R_2)C_2}{2}\left(1 \pm \sqrt{1 - \frac{4R^*C^*}{\left[1+(1+R^*)C^*\right]^2}}\right)$$

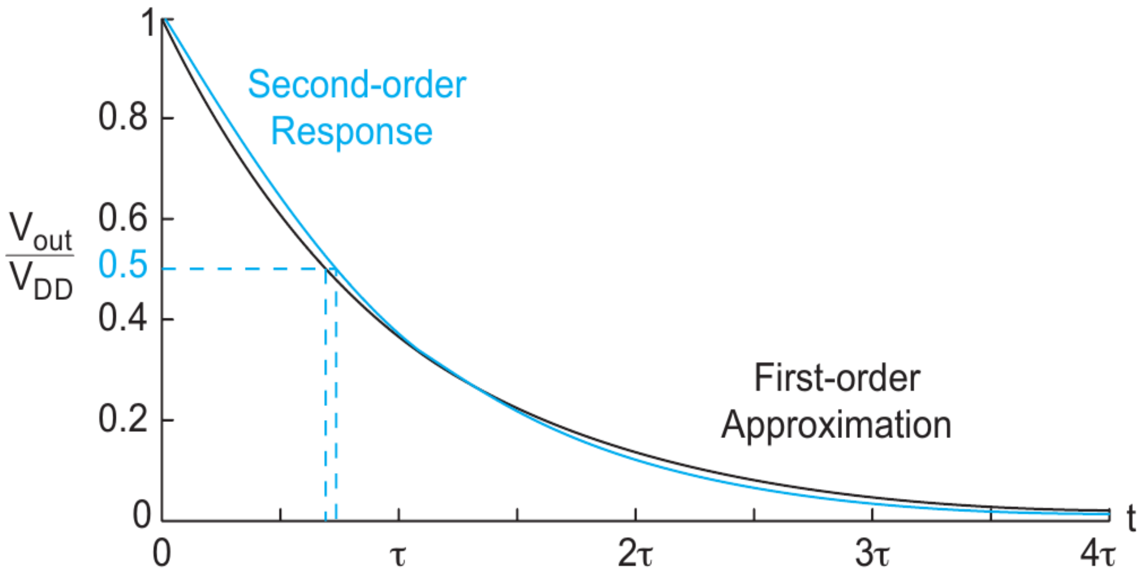
$$R^* = \frac{R_2}{R_1};\; C^* = \frac{C_2}{C_1}$$

First order Approximation

- When $R_1 = R_2$ and $C_1 = C_2$, $\tau_1 = 2.6RC$ and $\tau_2 = 0.4RC$ we can say $\tau_1 >> \tau_2$ we can approximate it with first order with

$$\tau = \tau_1 \fallingdotseq \tau_1 + \tau_2 = R_1C_1 + (R_1+R_2)C_2$$





Elmore Delay

- Generalization of RC tree delay,

$$t_{pd} = \sum_i R_{is} C_i$$

t_{pd} = propagation delay

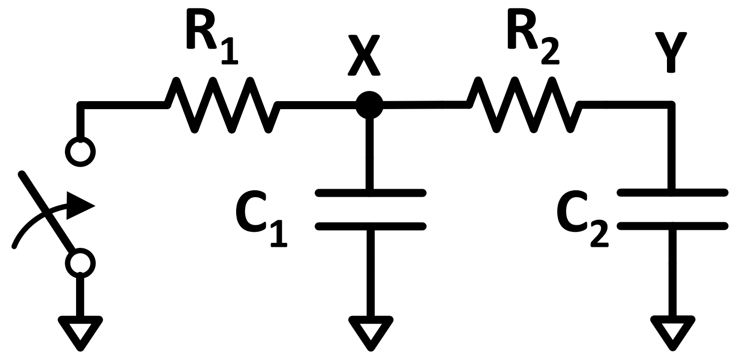
C_i = the capacitance of the node

R_{is} = resistance sum in the path of source from the node

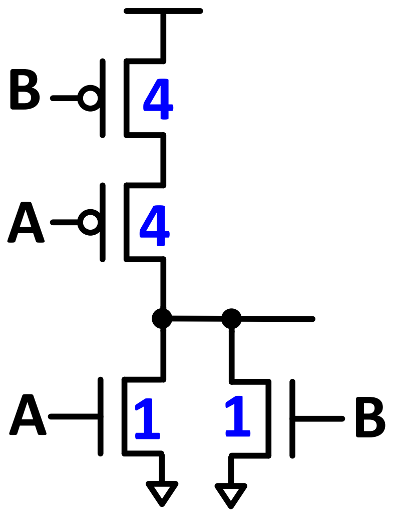
Revisit 2nd order RC System

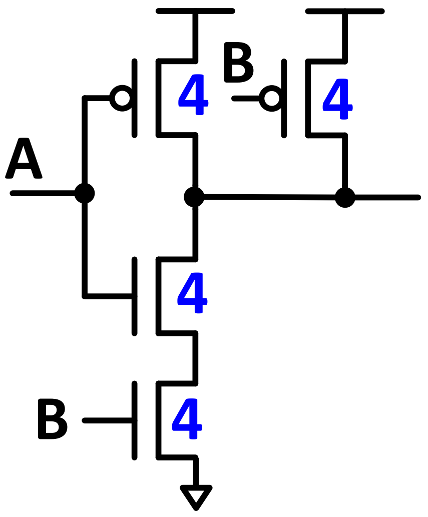
- Find Elmore delay of Y

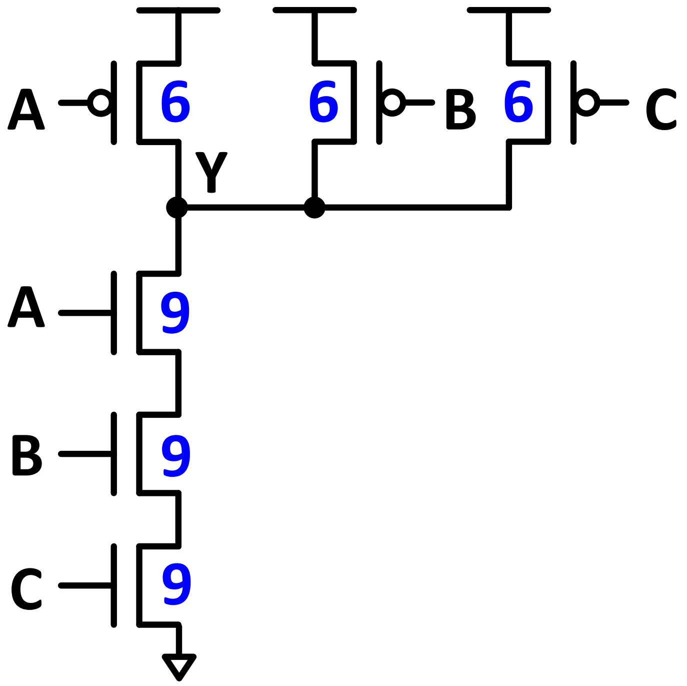
$$t_{pd} = \sum_i R_{is} C_i$$



Can You Imagine RC Tree?

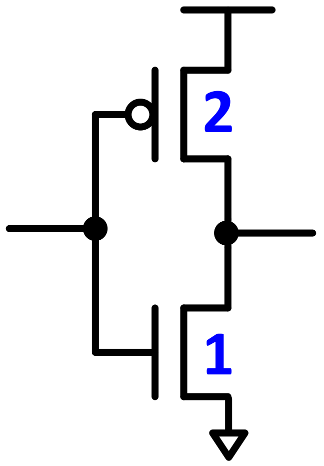


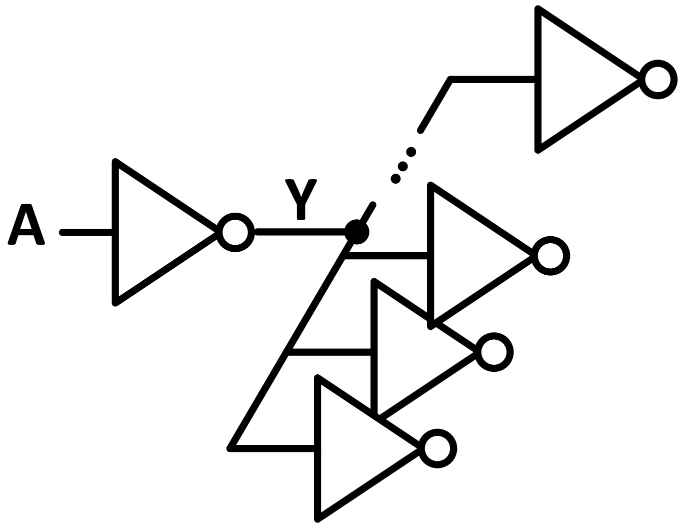




Estimating Delay of Inverter

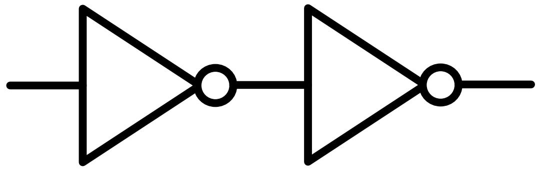
- Find Elmore delay of A to Y when a unit inverter shown in the left drives *m* identical unit inverters
- What if width is increased by *k*-times?
- What if R and C are given as 10kΩ and C = 0.1fF, respectively?



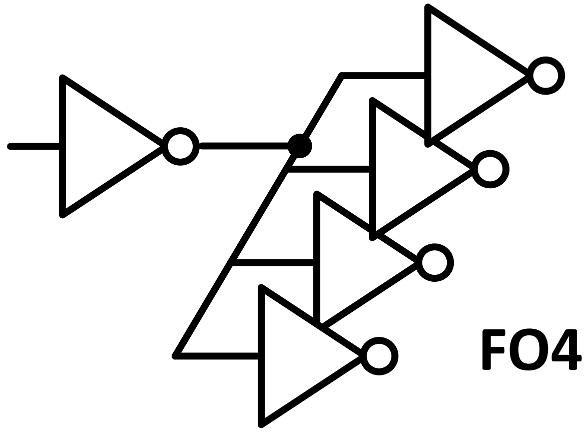


Fan-out and Normalized Delay

- Fanout (FO) means how many identical circuit a circuit drives



FO1



FO4

- When τ is defined as the FO1 inverter delay without parasitic cap, $\tau = 3RC$.
- Defining normalized delay is useful to express it with process-independent term so that circuits can be compared only based on their topologies.

$$d = t_{pd}/\tau$$

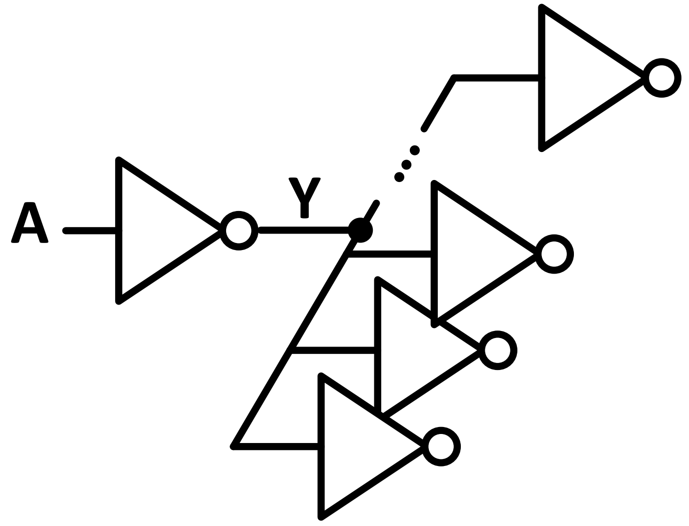
Can We Define Fanout When Different Logics Are Connected?

- We can generally define fan-out as

$$h = \frac{C_{out}}{C_{in}} = \text{fan out}$$

Normalized Delay

- Can you express the following circuit with normalized delay?



Propagation & Contamination Delay

- Propagation delay time t_{pd}

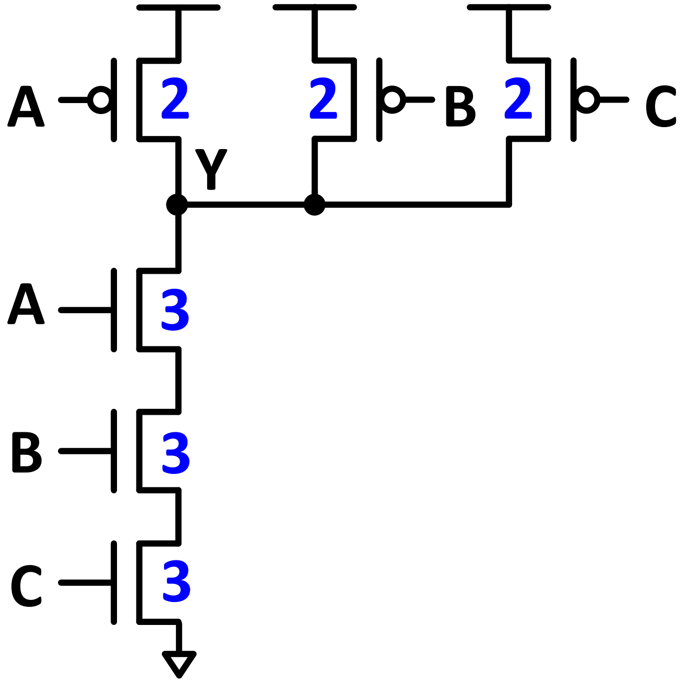
= **maximum** time from the input crossing 50% to the output crossing 50%

- Contamination delay time t_{cd}

= **minimum** time from the input crossing 50% to the output crossing 50%

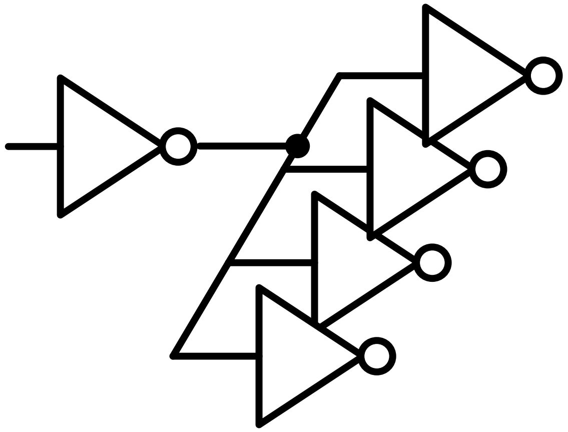
Example; NAND3

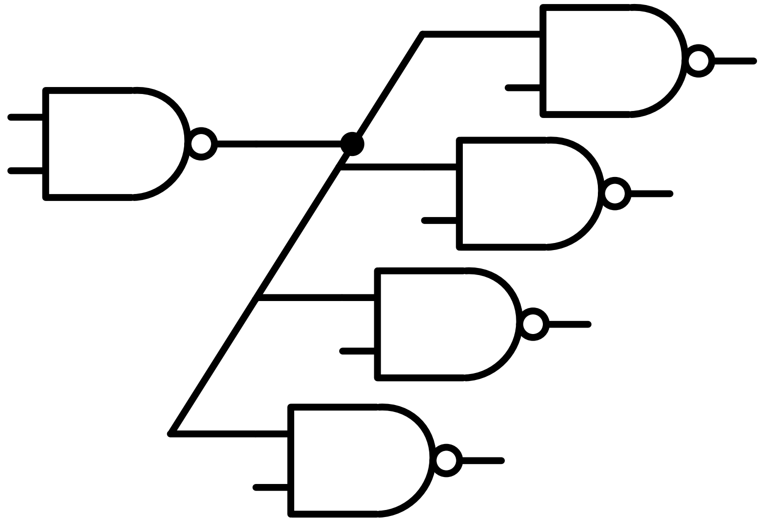
- Find t_{pdf}, t_{pdr}, t_{cdr}, t_{cdf} of the following NAND3 when
  1) The effective resistance and gate cap of unit width MOSFET are R and C, respectively.
  2) The following NAND3 drives *m* identical NAND3 gates.



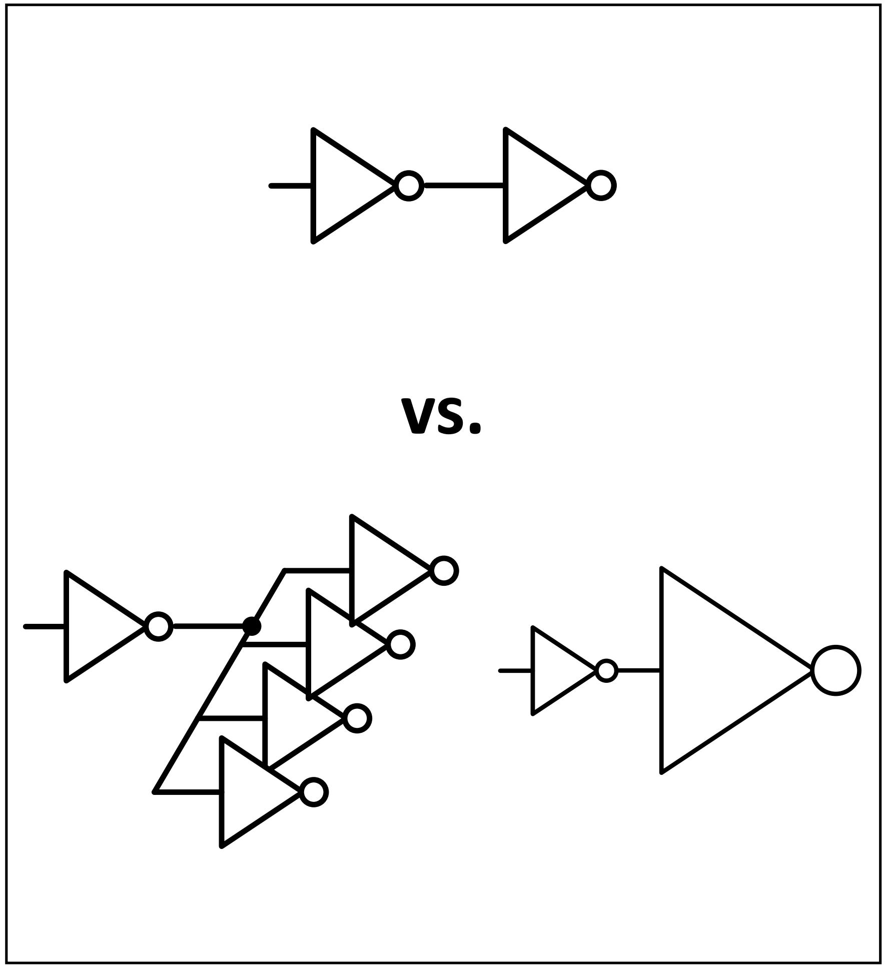
Can You See The Difference?

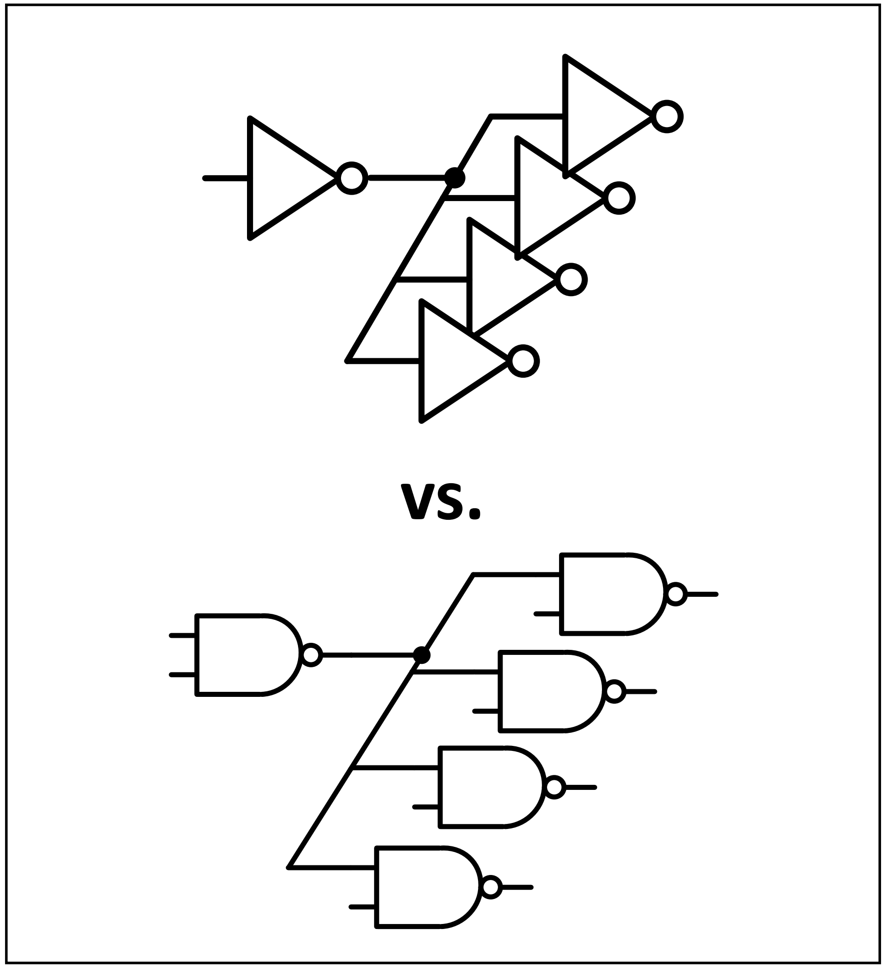
- They are both FO4 but differ in the normalized t_{pd}.





We Roughly See the Effects of…



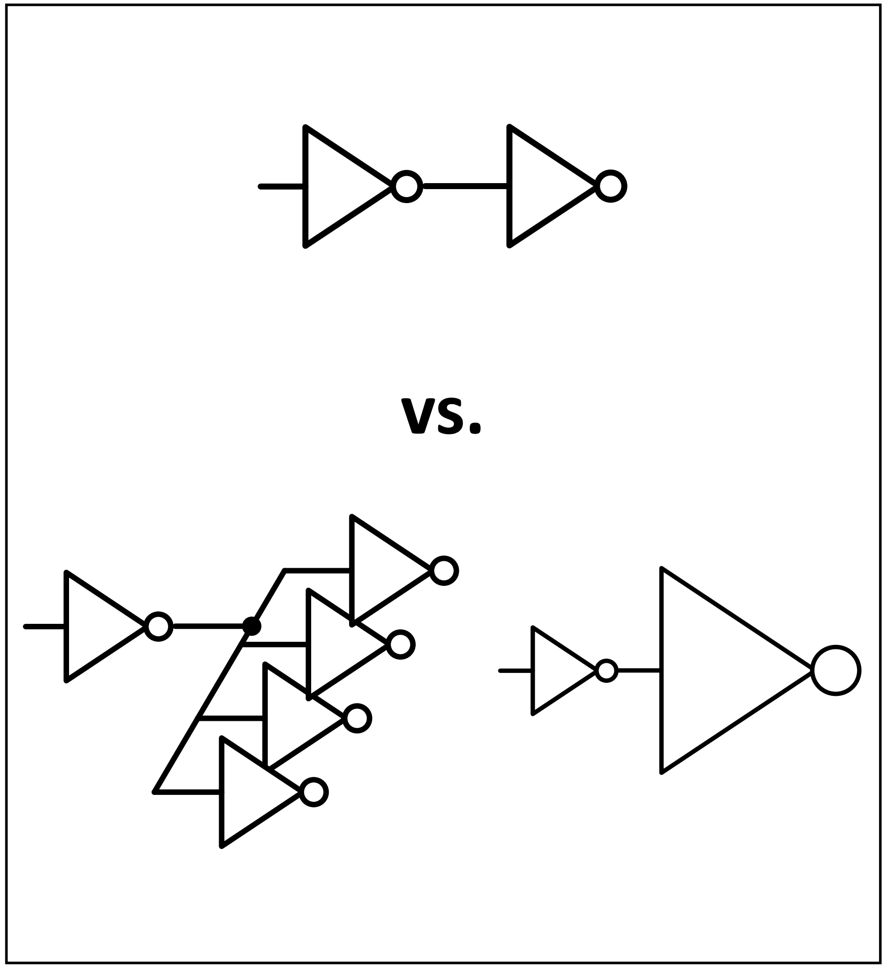


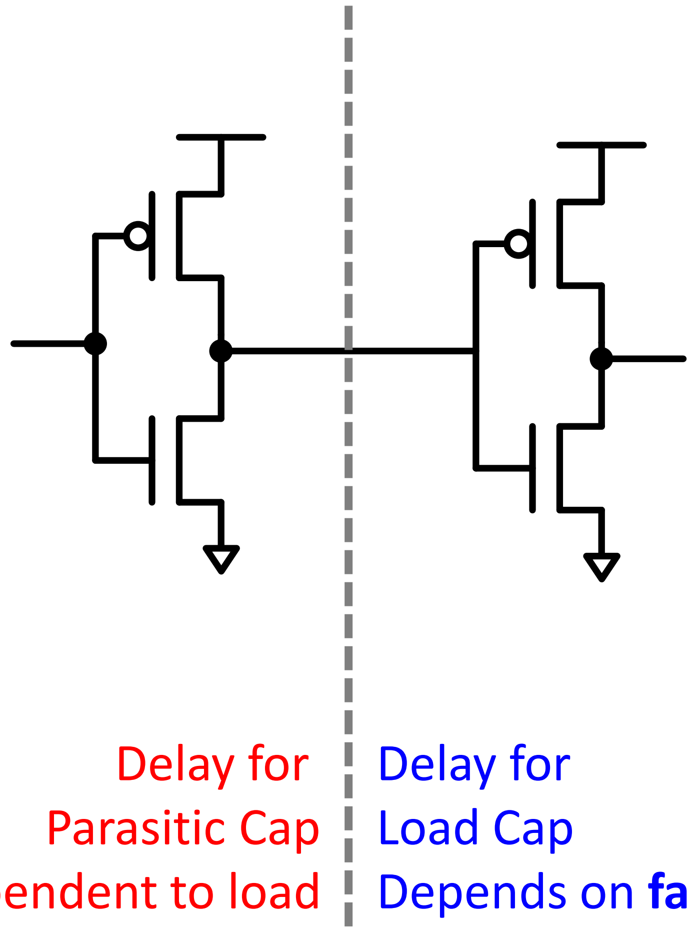
➔ **Can you generalize this respect and develop a model ?**

Linear Delay Model

- ✓ Extension RC delay model for taking into account fanout and logic complexity

How Can We Consider Fan-out?





Delay for
Parasitic Cap
Independent to load

Delay for
Load Cap
Depends on **fanout**

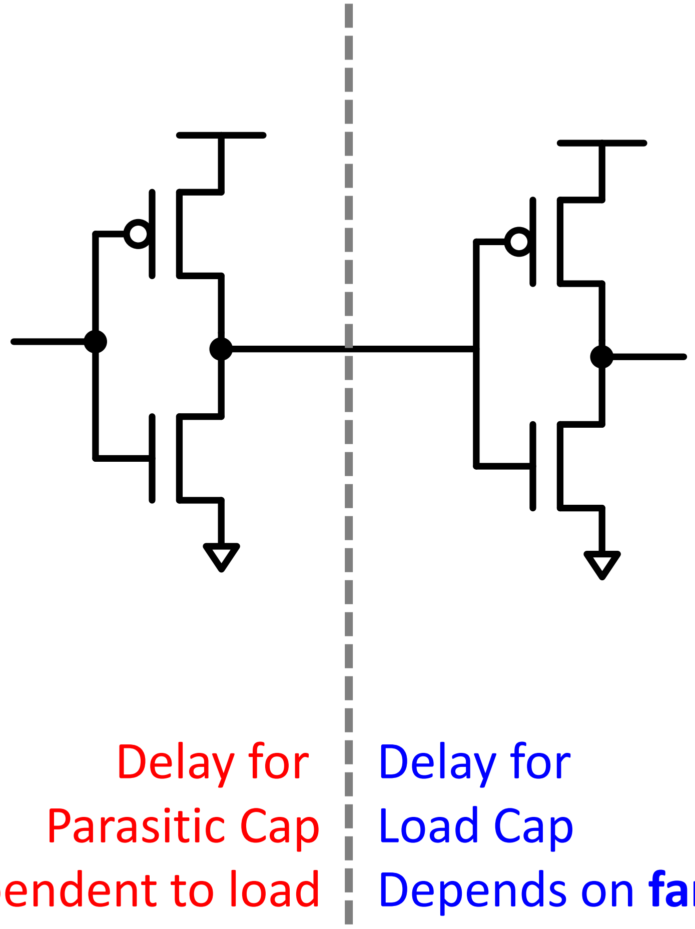
Normalized Propagation Delay Model

- We can develop the following model in terms of normalized delay:

$$d = f + p$$

where
f = effort delay or stage effort
p = parasitic delay

- How can we take into account the effect of $\mathbf{W_{load}/W_{driver}}$ into f?

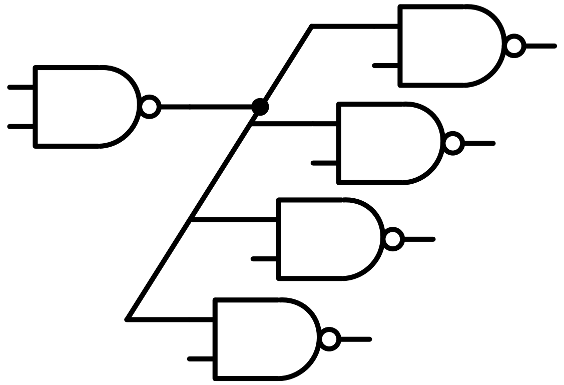


Let’s Focus on Stage Effort

- When we define the normalized delay, d=1 means FO1 unit inverter delay without parasitic delay.
- Then how about FO2, FO3, FO4? If it is not unit inverter?
- Then can’t we just define *f* as follows?

$$f = \frac{C_{out}}{C_{in}} = \text{fanout} = \text{electrical effort}$$

- No. what if



Now It is Time to Think How We Consider

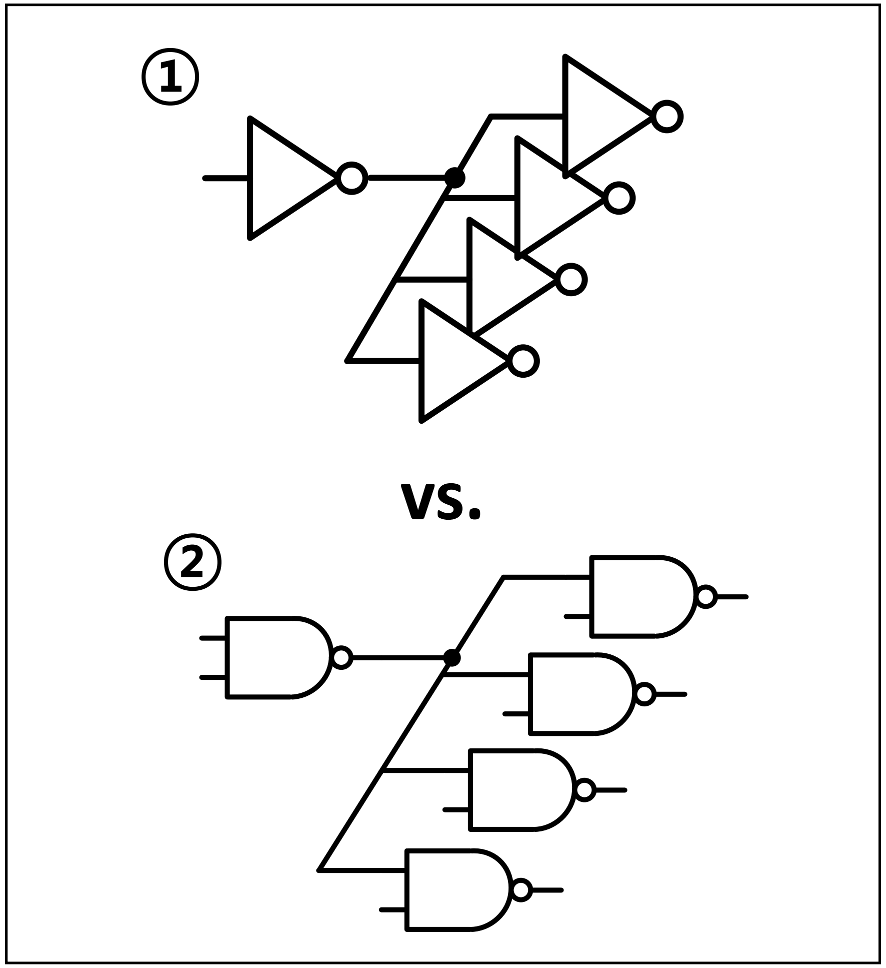
- In the previous model of

  $$d = f + p$$

  in *f* and *p*, there should be not only W_{load}/W_{driver} effect but also that of the logic complexity.

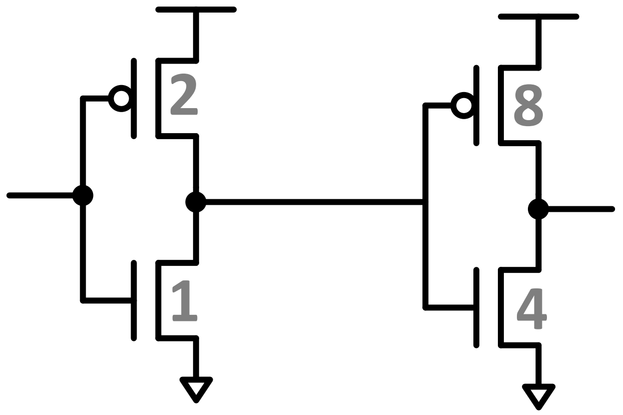
- We can just start from directly derive the delay ① vs. ②, and **feel** the difference

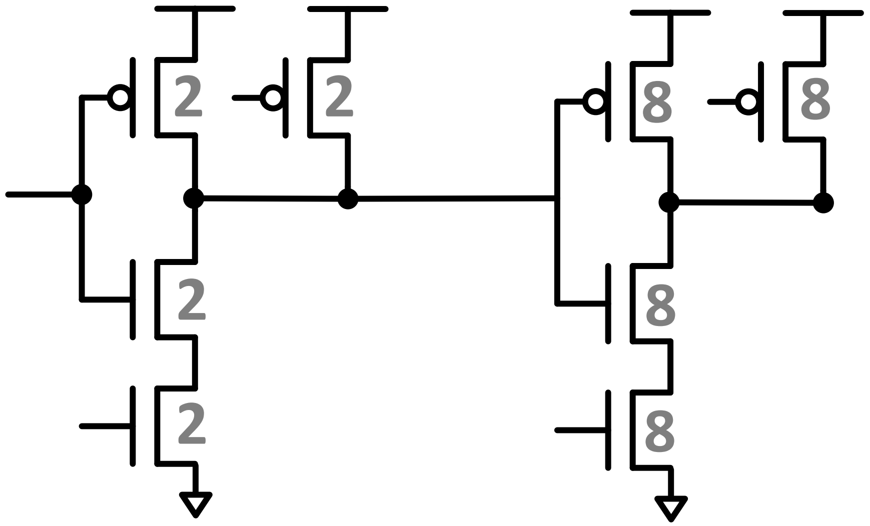
  ➔ Then, we can may find how the model should look like.



Compare Delay

- And feel the difference





- Note that falling & rising strength are matched
- In $d = f + p$,

f is increased by   times

p is increased by   times

Refining Model

- In the previous model of

$$\boldsymbol{d = f + p}$$

- If we define fan-out (=electrical effort) as *h*

$$h = C_{out}/C_{in}$$

- Then, we can make

$$\boldsymbol{d = g_{logic}h + p_{logic}}$$

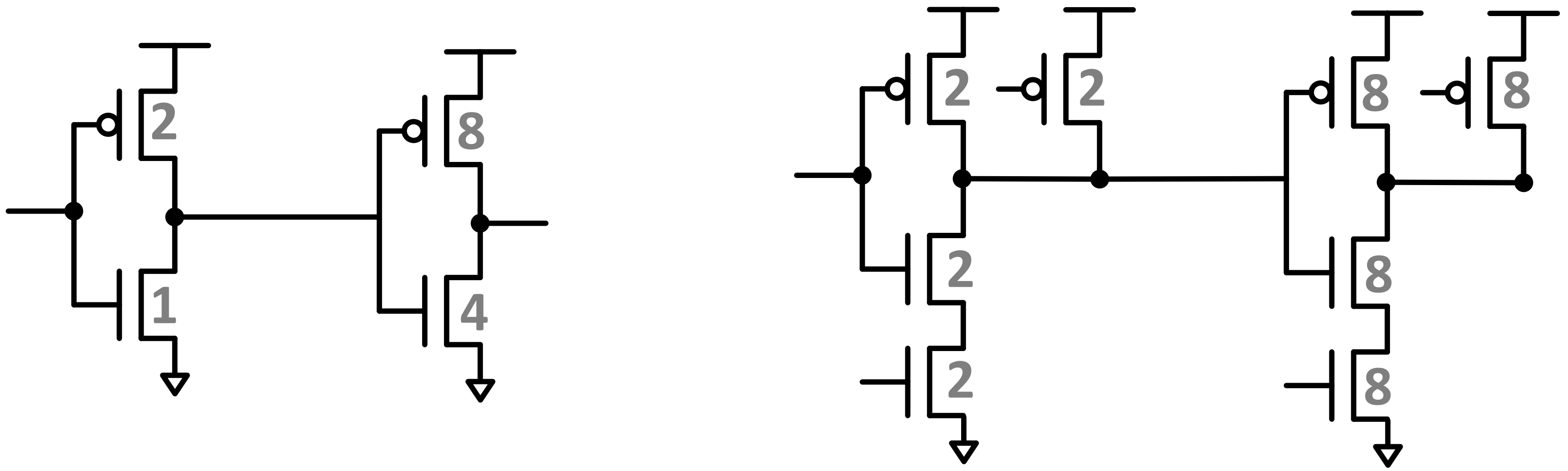
- Or simply,

$$\boldsymbol{d = gh + p}$$

where g is **logical effort** and p is **parasitic delay**

Logical Effort *g*?

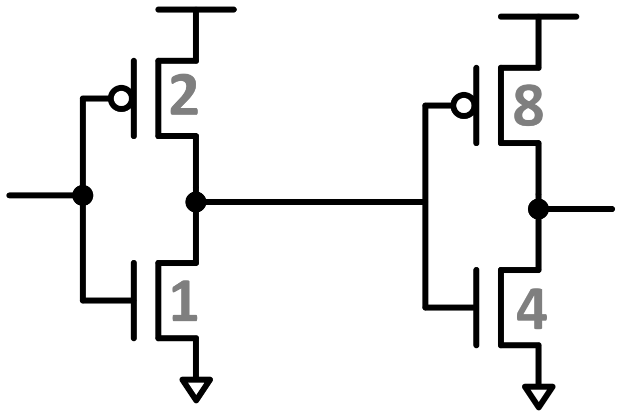
- Where does it come from?

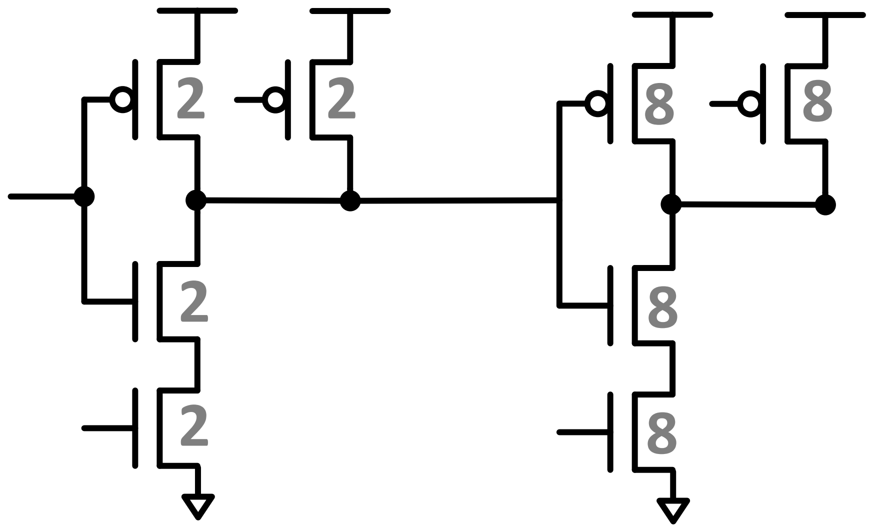


- It is attributed to the gate cap increase for matching the driving current.
- Definition: the ratio of the input capacitance of the gate to the input capacitance of an inverter that can deliver the same output current.
- Can you predict ***g*** of NAND3, NAND4, or NOR2, NOR3, NOR4, etc.?

Parasitic Delay for Various Logics?

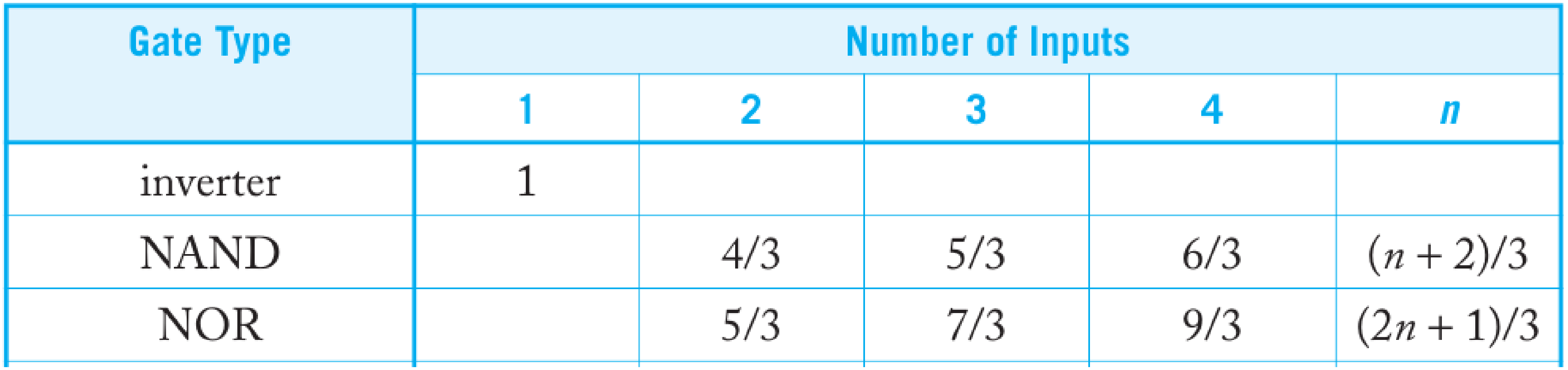
- Where does it come from?



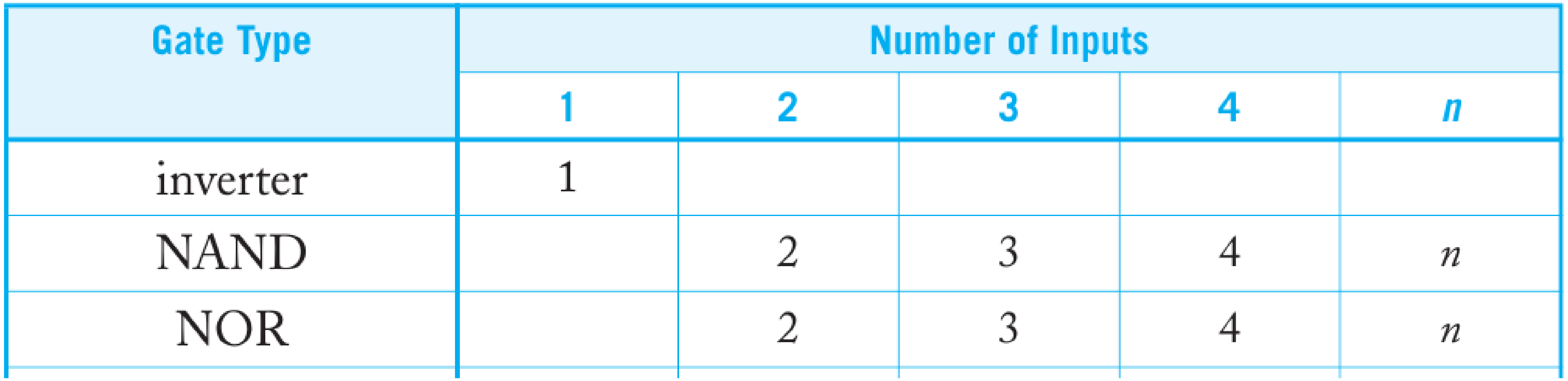


- It is attributed to the drain cap increase for matching the driving strength and the number of MOSFETs
- Definition: the delay of the gate when it drives zero load.
- Can you predict ***p*** of NAND3, NAND4, or NOR2, NOR3, NOR4, etc.?

g and p for Various Logics



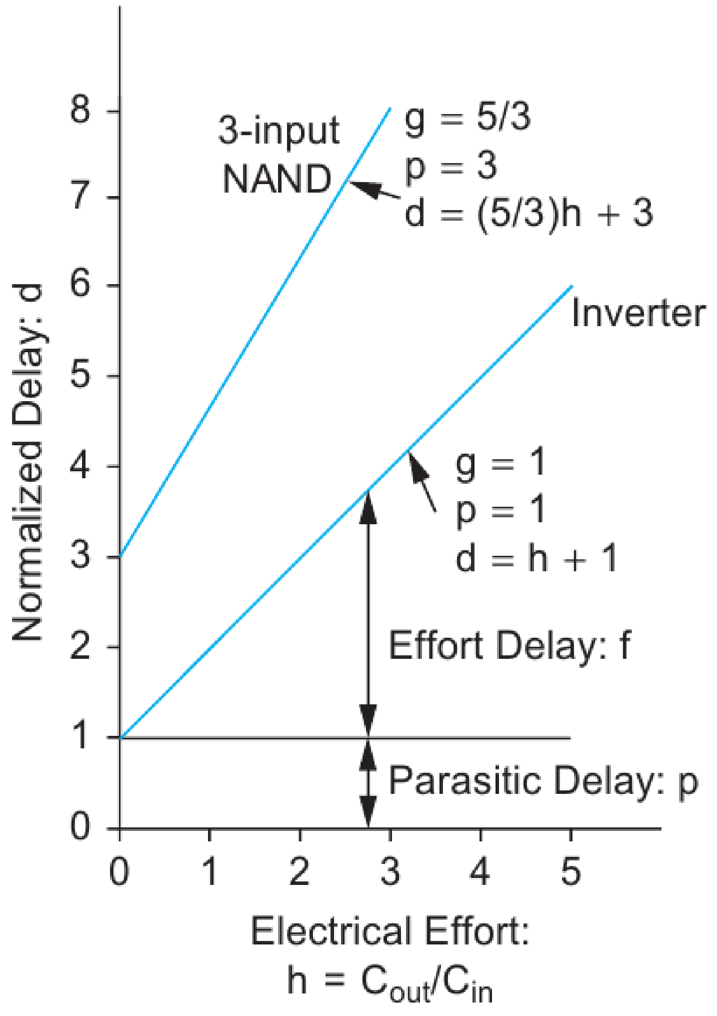
Gate Type	Number of Inputs				
	1	2	3	4	n
inverter	1				
NAND		4/3	5/3	6/3	$(n+2)/3$
NOR		5/3	7/3	9/3	$(2n+1)/3$



Gate Type	Number of Inputs				
	1	2	3	4	n
inverter	1				
NAND		2	3	4	n
NOR		2	3	4	n

Normalized Delay vs. Fanout

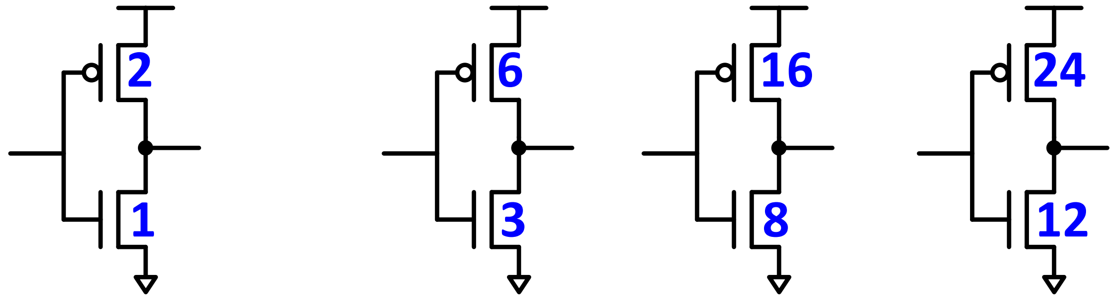
- $d = gh + p$

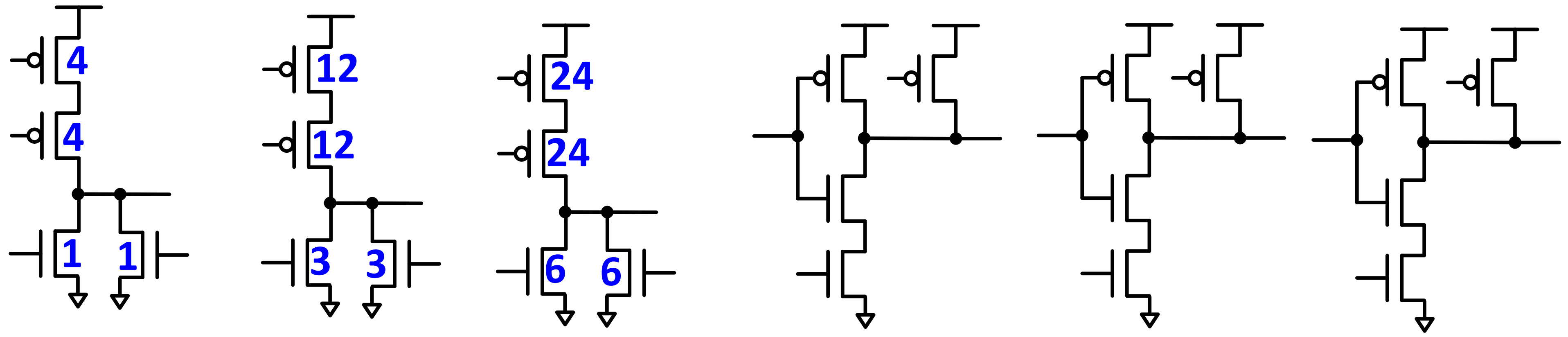


Drive

- Generally, drive x is defined as

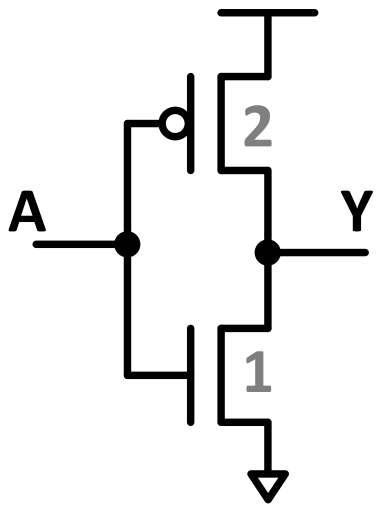
$$x = C_{in}/g$$

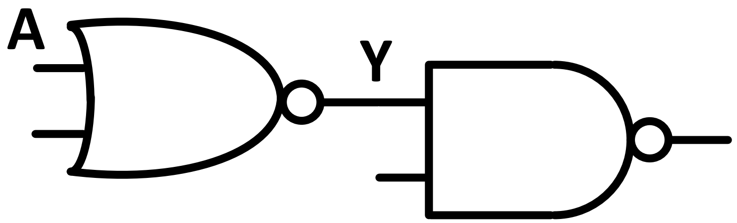




Methodology to Derive g, p and h Generally

- Refer the unit inverter and start from deriving ***g*** by its definition of "the ratio of the input cap of the gate to the input cap of an inverter that can deliver the same output current."



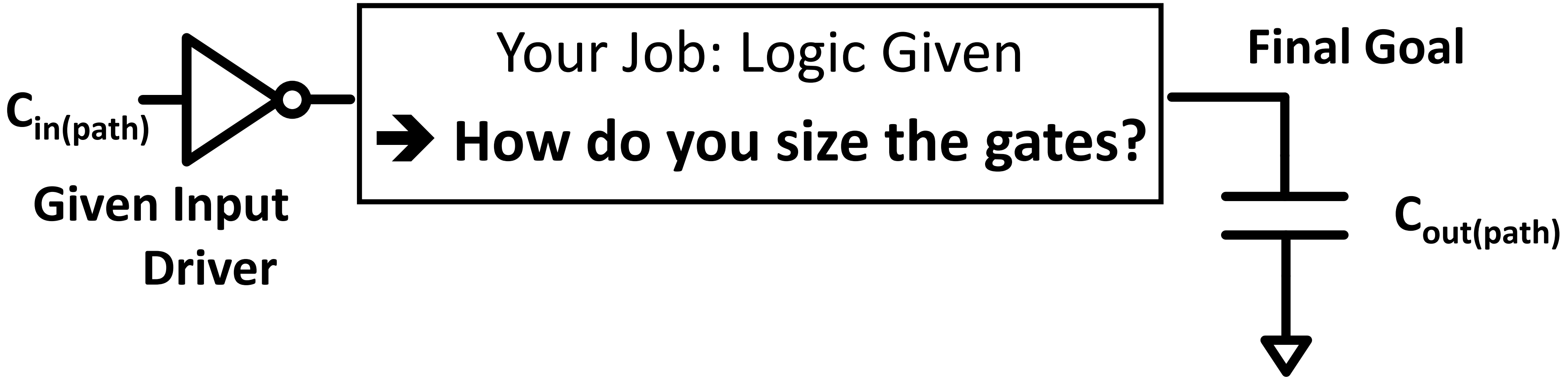


Logical Effort of Paths

- ✓ Path delay characterization and Optimization

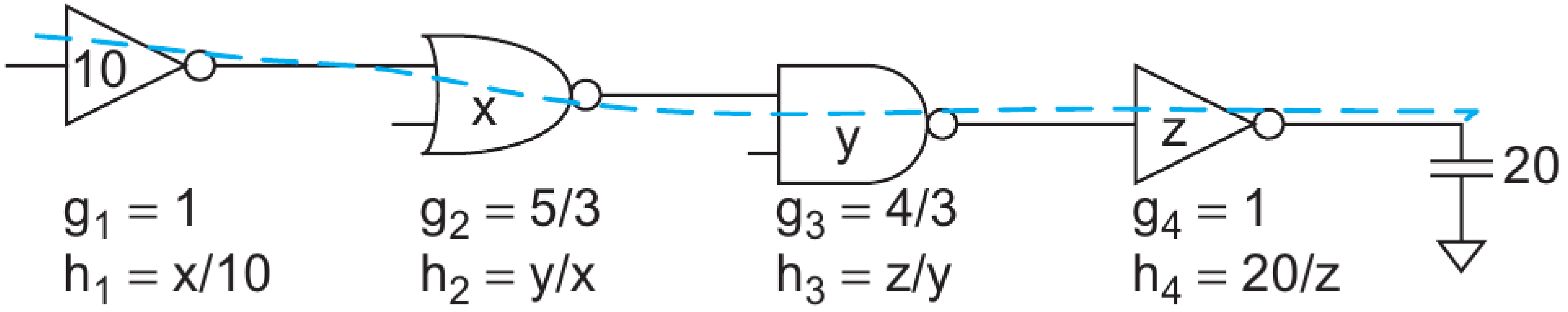
Delay Optimization

- Imagine what situation you possibly be in



Delay in Multistage Logic Networks

- The number marked in the gates denote C_{in}.



- Then how can you calculate the path delay?

Introducing Path Level Parameters

- There is an incentive to introduce path-level delay parameters that are independent to the each gate size.
- That is, path electrical effort H

$$H = \frac{C_{out(path)}}{C_{in(path)}}$$

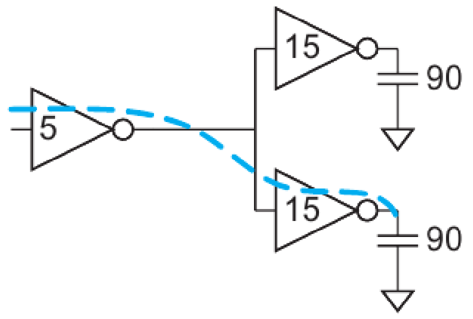
- Path logical effort G

$$G = \prod g_i$$

- Path effort

$$F = \prod f_i$$

Need For Branching Effort



- Can we say F = GH? as in the stage effort case?
- No. What if there are branch paths?
- Thus, to relate *F* with *G* and *H*, we need additional parameter that can characterize branch feature.
- In each stag, branching effort b is defined as

$$b = \frac{C_{onpath}+C_{offpath}}{C_{onpath}}$$

- Path branching effort B is defied as

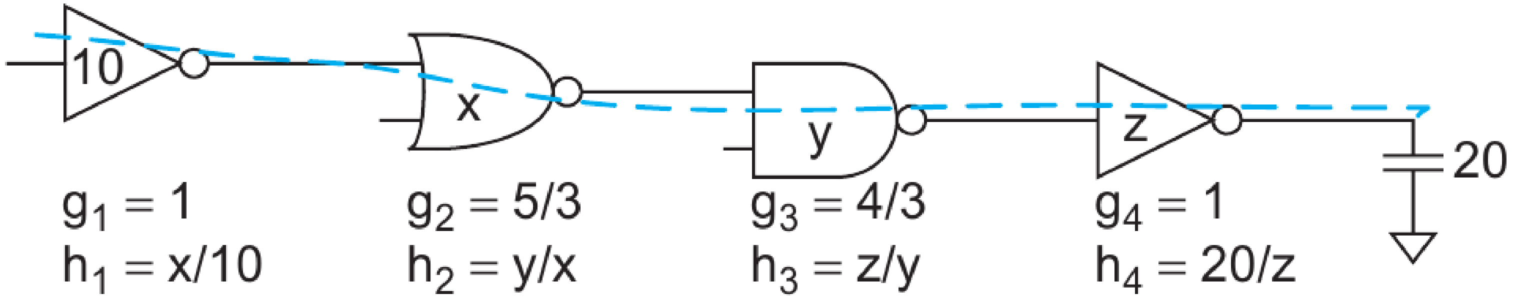
$$B = \prod b_i$$

- Thus,

$$F = GBH$$

Delay

- You remember



$$\textit{Path delay } D = \sum d_i = \sum f_i + \sum p_i$$

- If we define *path effort delay* $D_F = \sum f_i$ and path parasitic delay $P = \sum p_i$, then

$$D = \sum d_i = D_F + P$$

Minimizing D_F

- Focus on D_F

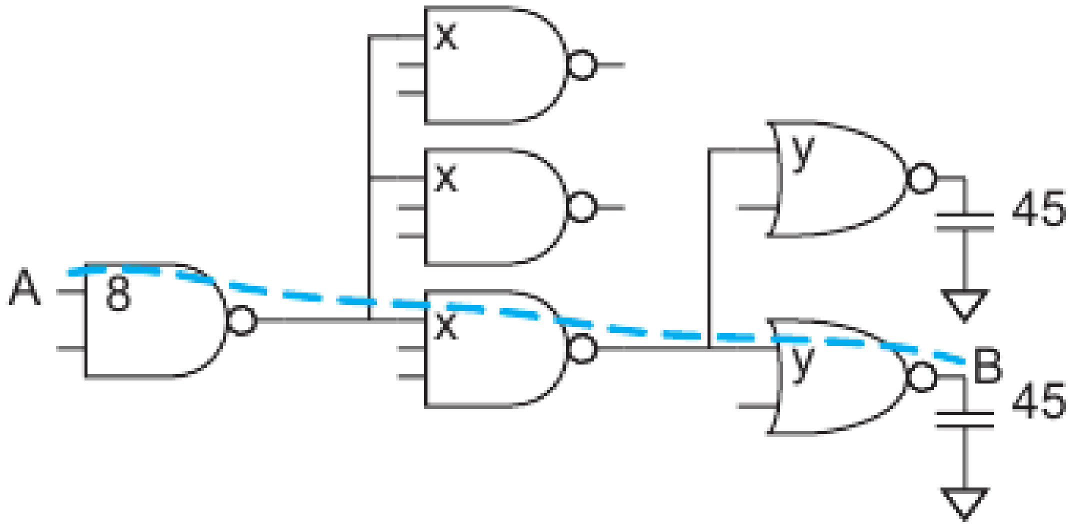
$$D_F = \sum f_i$$

- We know that $F = \prod f_i$ is constant, no matter how the sizing of each stage is made.
- Let me ask one, when $\underline{x}y = 2$, what is the minimum of $x + y$?
- When $\prod f_i$ is constant, what is the minimum of $\sum f_i$?
- The sum of numbers whose product is constant is minimized by setting all the numbers equal. Thus, D_F is minimized if all f_i is equal to $\hat{f}$

$$\hat{f} = g_i h_i = F^{1/N}$$ ➔ **Gate Sizing Strategy**

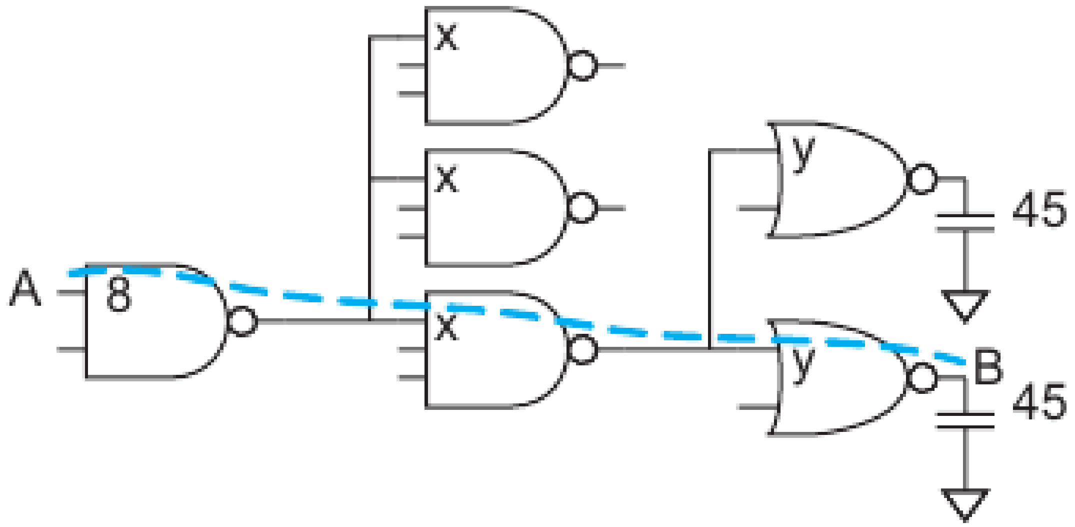
➔ The minimum delay of the path can be estimated knowing only the *N*, *F*, and *P* without the need to assign transistor sizes.

Minimizing Delay Example



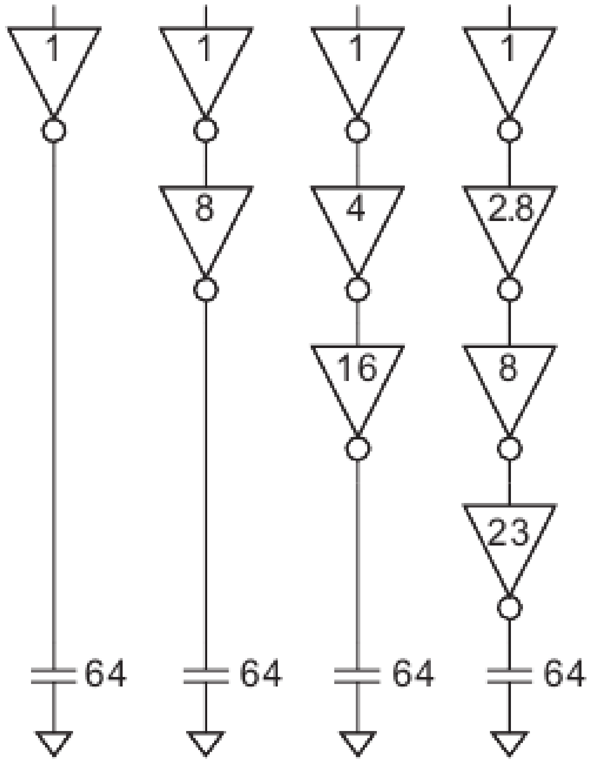
The Larger W is, The Faster Circuit is?

- No. Why?
- Not only current increases, but also its cap increases



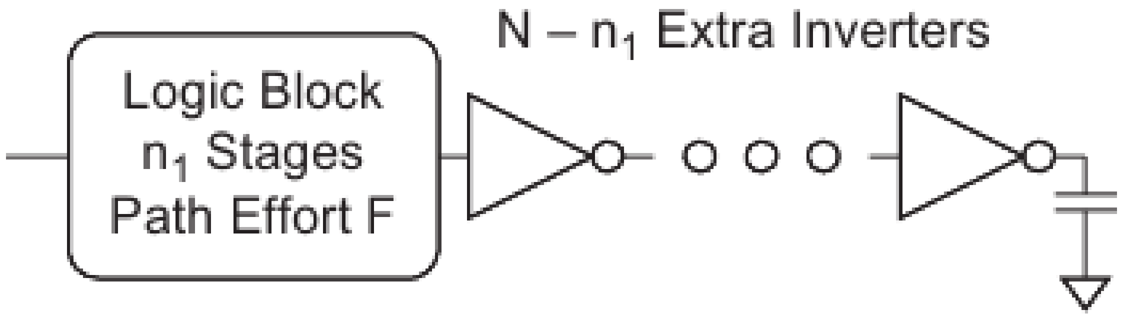
Choosing the Best Number of Stages

- Is smaller N always results in smaller delay?



N	1	2	3	4
f	**64**	**8**	**4**	**2.8**
D	**64+1**	**16+2**	**12+3**	**2.8*4+4**

Generally, We Can Change N Easily



- After N is determined, you can choose $f = F^{1/N}$ to minimize the delay. Question is how to determine N. Strategy?
- Express *D* in terms of *N* then differentiate it with respect to *N*.

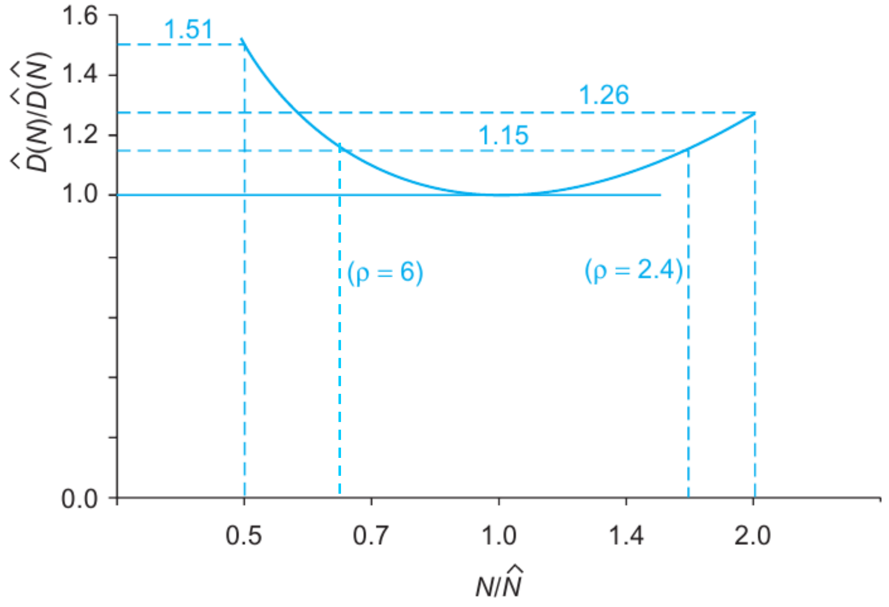
$$D = NF^{1/N} + P = NF^{1/N} + \sum_{i=1}^{n_1} p_i + (N-n_1)p_{\text{inv}}$$

- Differentiating with respect to N, and defining $\rho = F^{1/\hat{N}}$

$$\frac{\partial D}{\partial N} = -F^{1/N}\ln F^{1/N} + F^{1/N} + p_{\text{inv}} = 0$$
$$\Rightarrow p_{\text{inv}} + \rho\left(1 - \ln\rho\right) = 0$$

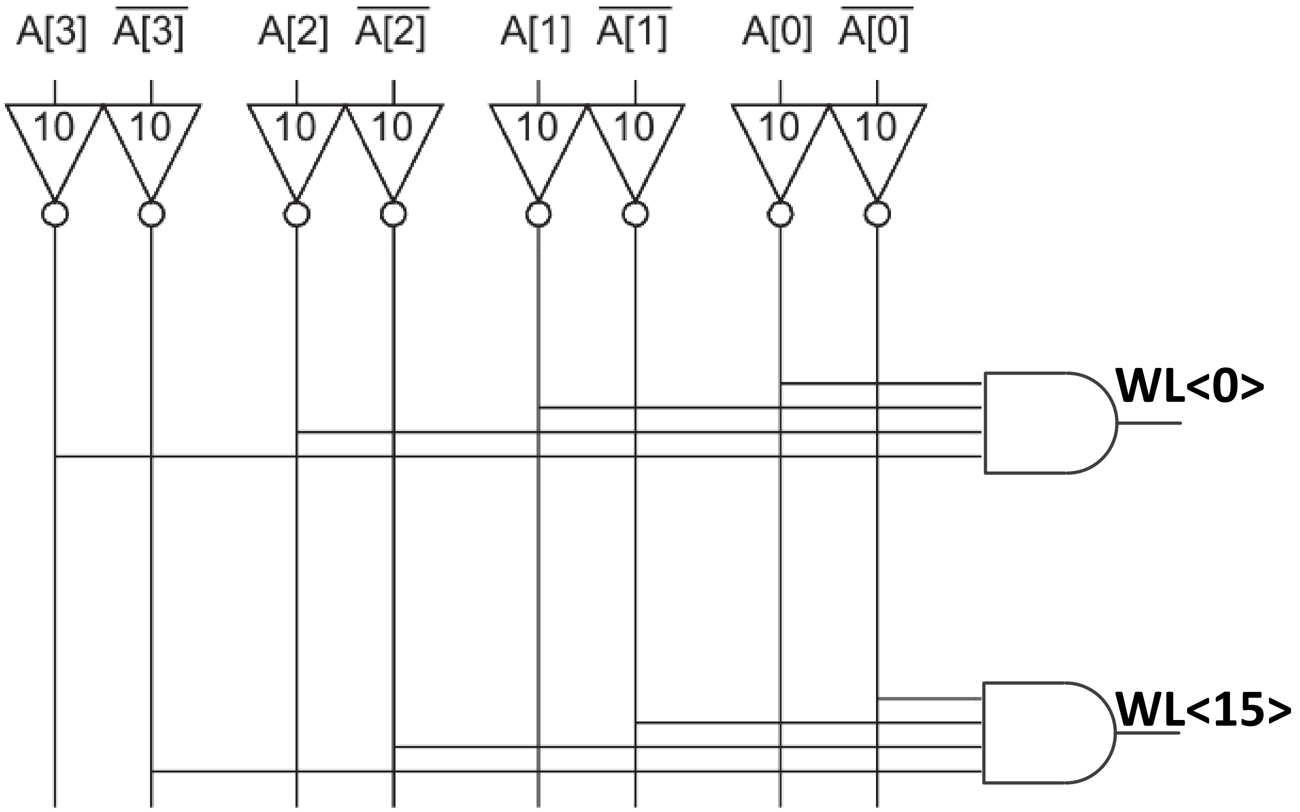
Solving Numerically,

- D becomes minimized when **$\rho = F^{1/N} \fallingdotseq 3.6$**

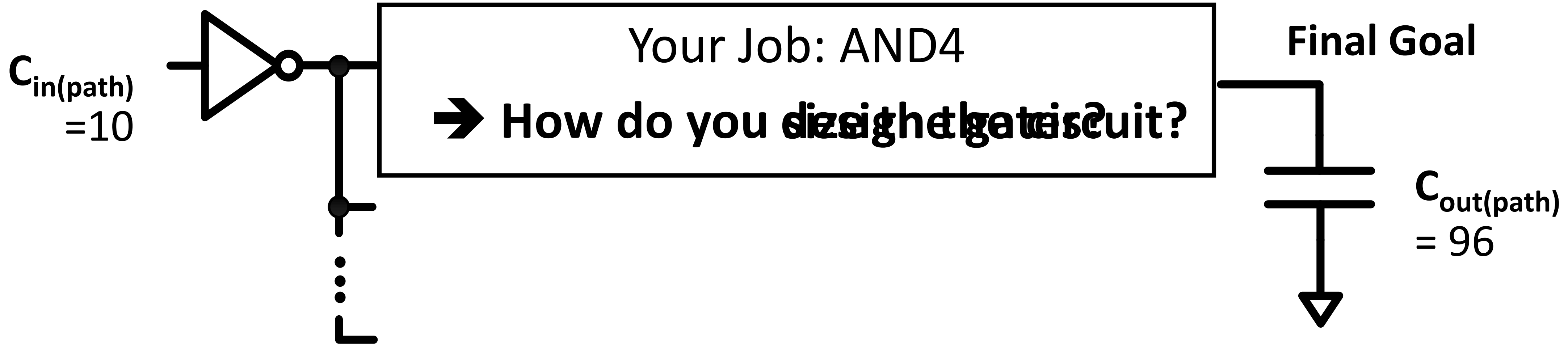


- Using a stage effort of 4 is a convenient choice and simplifies mentally choosing the best number of stages.
- This effort gives delays within 2% of minimum for p_{inv} in the range of 0.7 to 2.5. ➔ FO4 is representative sizing strategy.

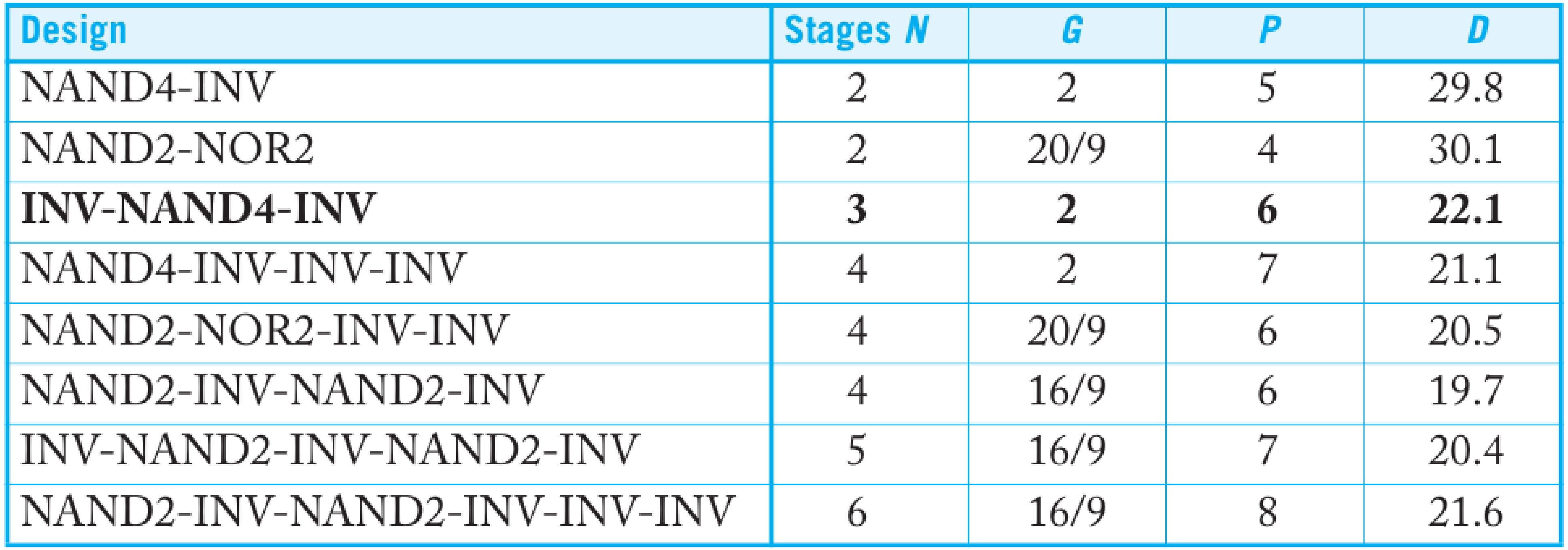
Implementing 4 to 16 Decoder



Implementing Decoder

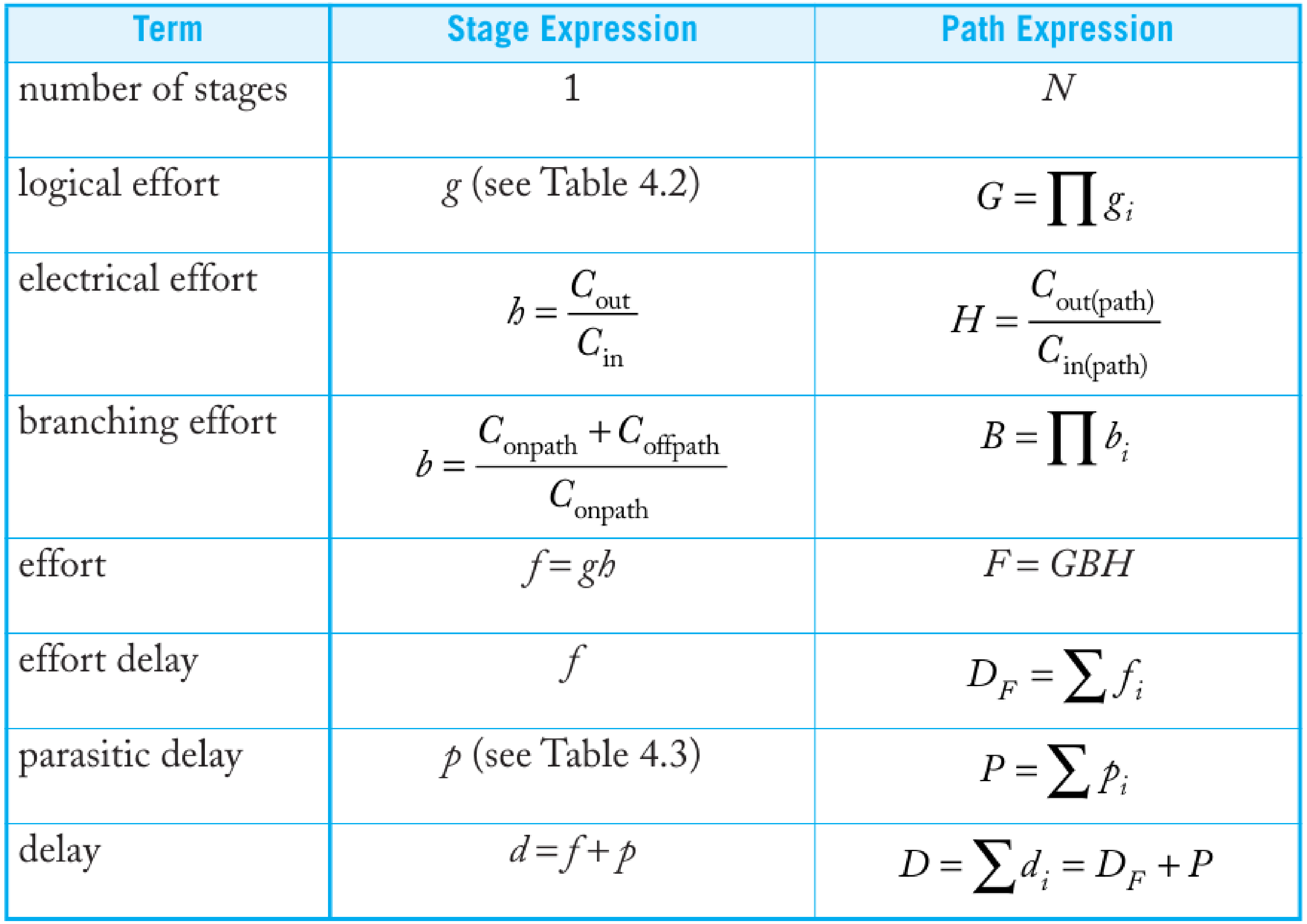


Comparing Designs



Design	Stages *N*	*G*	*P*	*D*
NAND4-INV	2	2	5	29.8
NAND2-NOR2	2	20/9	4	30.1
INV-NAND4-INV	**3**	**2**	**6**	**22.1**
NAND4-INV-INV-INV	4	2	7	21.1
NAND2-NOR2-INV-INV	4	20/9	6	20.5
NAND2-INV-NAND2-INV	4	16/9	6	19.7
INV-NAND2-INV-NAND2-INV	5	16/9	7	20.4
NAND2-INV-NAND2-INV-INV-INV	6	16/9	8	21.6

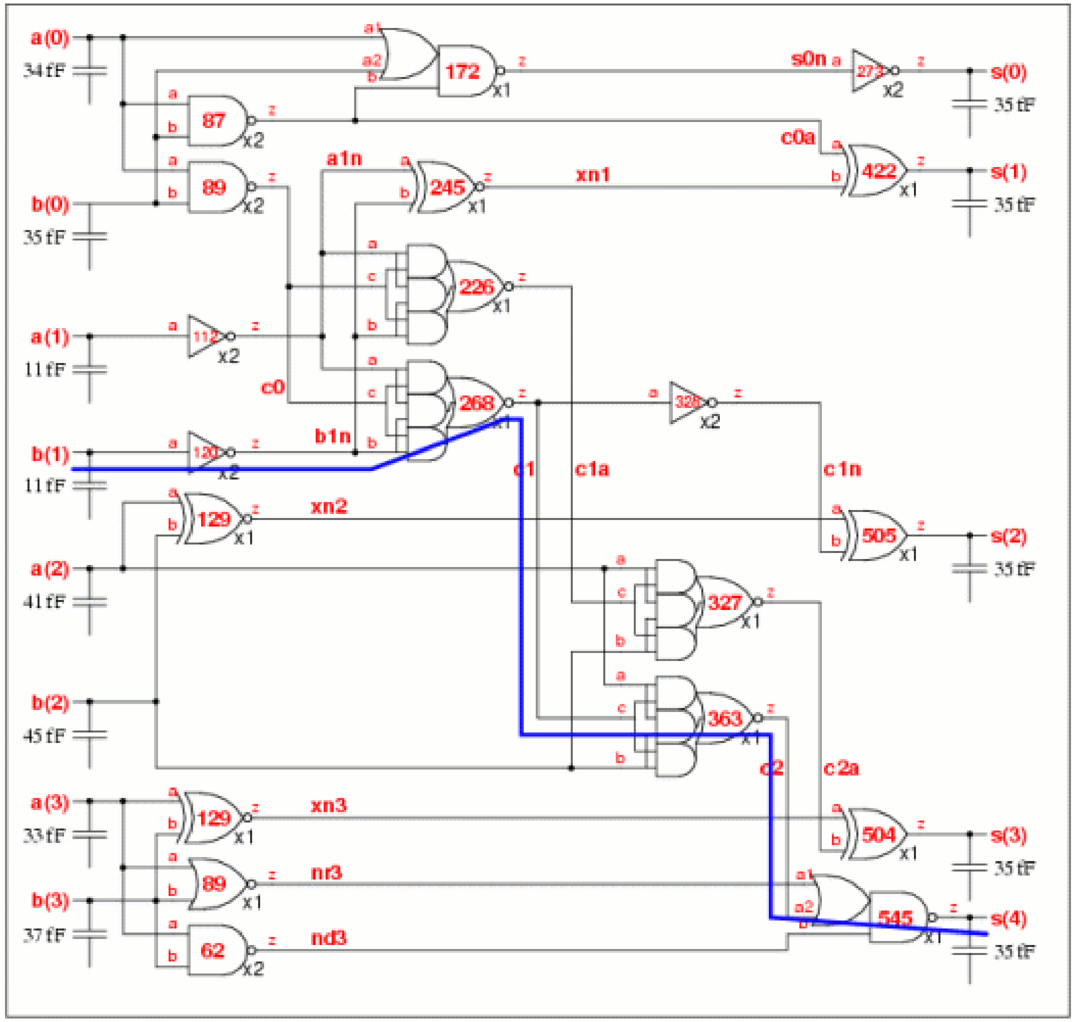
Summary



Term	Stage Expression	Path Expression
number of stages	1	N
logical effort	g (see Table 4.2)	$G = \prod g_i$
electrical effort	$h = \frac{C_{\text{out}}}{C_{\text{in}}}$	$H = \frac{C_{\text{out(path)}}}{C_{\text{in(path)}}}$
branching effort	$b = \frac{C_{\text{onpath}} + C_{\text{offpath}}}{C_{\text{onpath}}}$	$B = \prod b_i$
effort	$f = gh$	$F = GBH$
effort delay	f	$D_F = \sum f_i$
parasitic delay	p (see Table 4.3)	$P = \sum p_i$
delay	$d = f + p$	$D = \sum d_i = D_F + P$

Another Thing, Critical Path

- Almost paths are already fast enough for the timing goals of the system.
- However, there are critical paths that limit the speed of the system


a(0)
34fF
b(0)
35fF
a(1)
11fF
b(1)
11fF
a(2)
41fF
b(2)
45fF
a(3)
33fF
b(3)
37fF
172
87
89
a1n
245
xn1
226
112
c0
268
b1n
120
c1
c1a
328
c1n
129
xn2
505
327
363
c2
c2a
129
xn3
504
89
nr3
545
62
nd3
s0n
273
c0a
422
s(0)
s(1)
s(2)
s(3)
s(4)
35fF