MSE | MASTER OF SCIENCE IN ENGINEERING

Hes·so // FRIBOURG FREIBURG
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Chapitre 2
Delay Models Used For Digital Timing Analysis
MA-AdvElDes

Pietro Buccella

A. A. 2021/22

Objectives

Objectives of the lesson:

- Knowledge of models for propagation delay calculation of CMOS gates
- Application of delay model to simple CMOS logic gates
- Use logical effort method to predict the best number of stages in a multistage networks

Contents

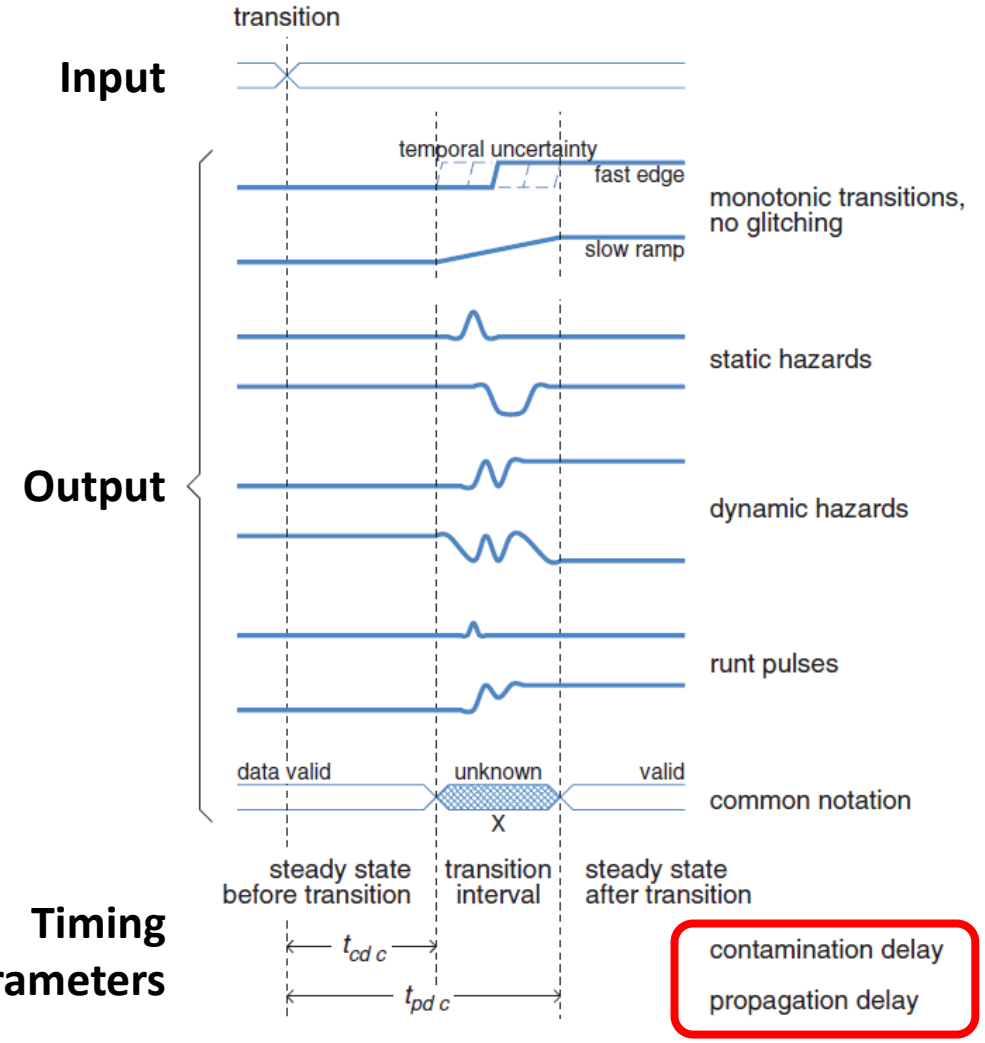
- Logic circuits: equivalent RC model
- Elmore delay
- Linear delay model of logic gates
- Delay in multistage networks
- Best number of stages

Transient Behaviour of Logic Circuits

How long transient phenomena persist at the output of a digital circuit in response to a change at one of the circuit's inputs?

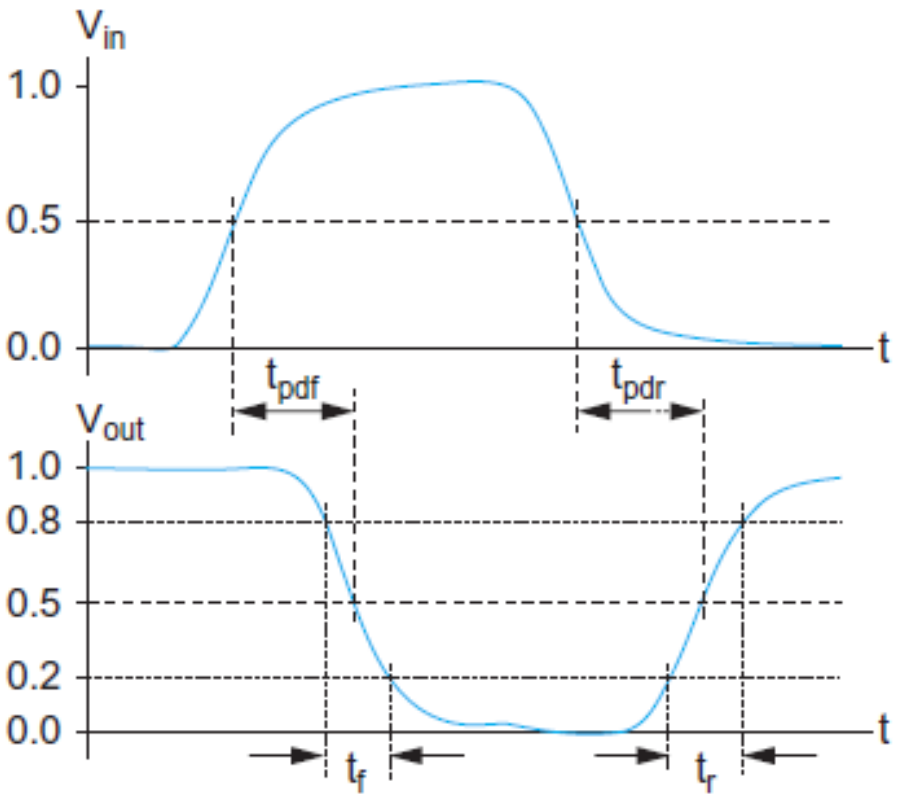
Two **Timing Parameters** :

- the output will retain its old value for at least the **contamination delay** t_{cd} and take on its new value in at most the **propagation delay** t_{pd}



Transient Behaviour of Logic Circuits

- *Contamination delay,* t_{cd} : **minimum time** from the input crossing 50% to the output crossing 50%
- *Propagation delay,* t_{pd} : **maximum time** from the input crossing 50% to the output crossing 50%
- *Rise time, tr* :time for a waveform to rise from 20% to 80% of its steady-state value
- *Fall time, tf* : time for a waveform to fall from 80% to 20% of its steady-state value
- *Edge rate, trf* = (*tr* + *tf*)/2

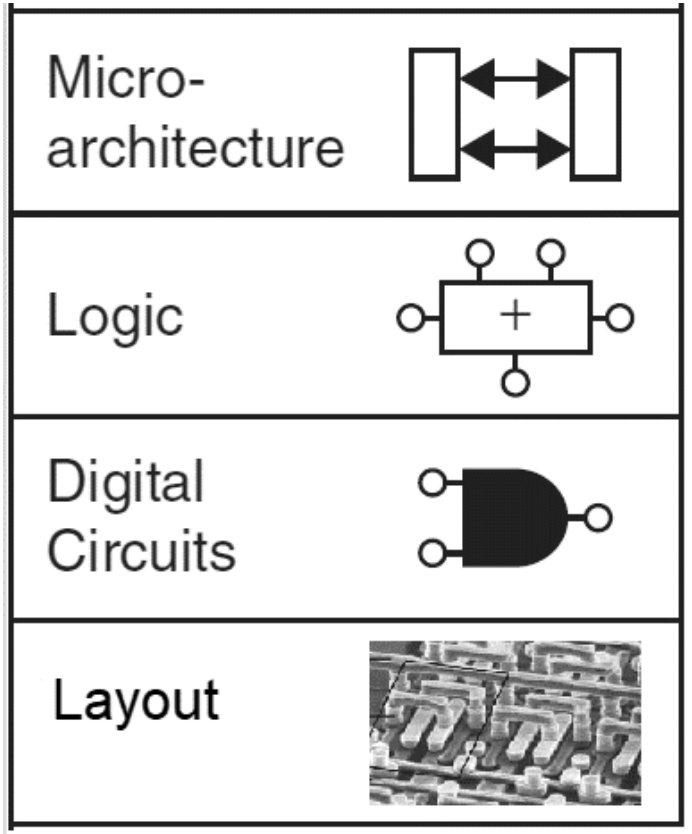


Timing Optimization

There are a number of critical paths that limit the operating speed of a system

Critical paths can be affected at four main levels:

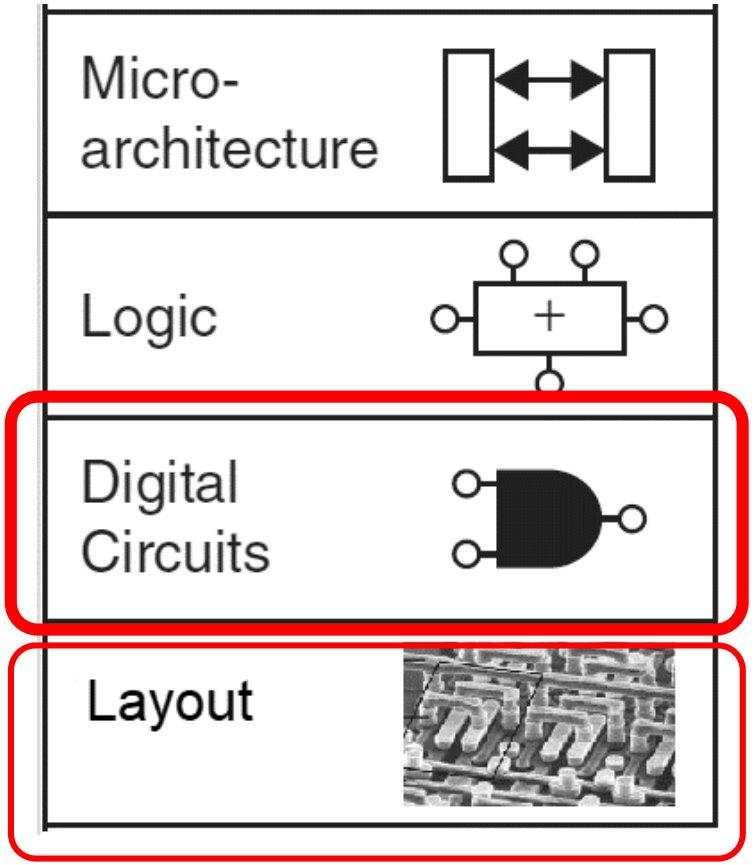
- The microarchitectural level
- The logic level
- The circuit level
- The layout level



Timing Optimization

Delay is dependent on:

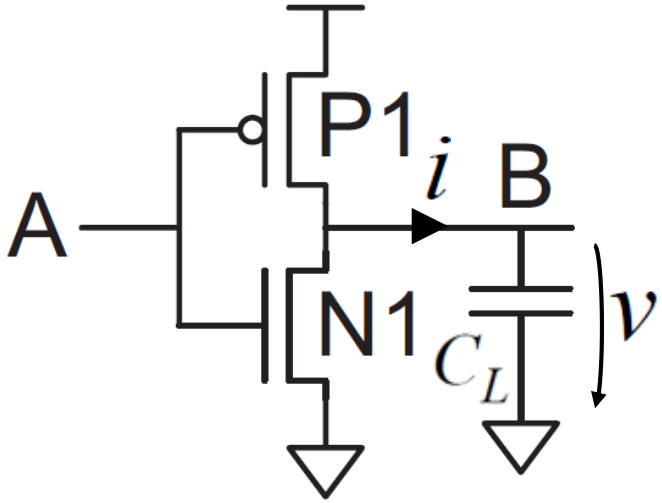
- Algorithms, number of pipeline stages, the number of execution units, size of memories
- Functional blocks, number of stages of gates in the clock cycle, the fan-in and fan-out of the gates
- Transistor sizes or styles of CMOS logic
- Layout, wire lengths and parasitic capacitance can dominate delay



Transient Response Calculation

Delay Computation in Digital Gates:

1. Build physical model of the circuit
2. Write differential equation of output voltage as a function of input voltage and time
3. The solution is the *transient response*
4. Delay is the time when the output reaches *VDD*2



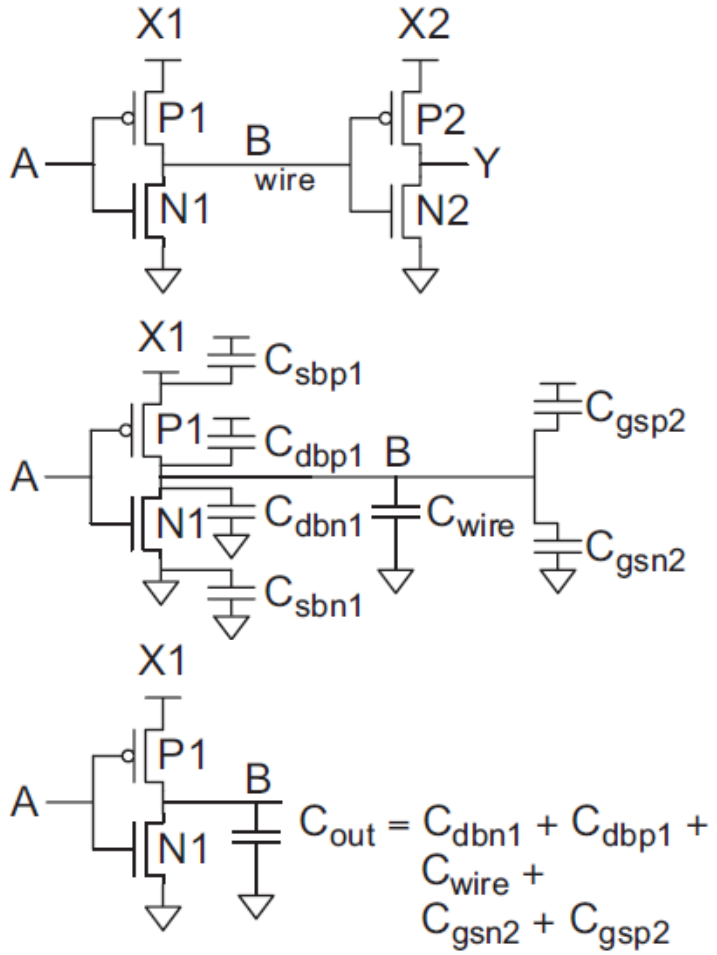
$$t_p = \int_{v_1}^{v_2} \frac{C_L(v)}{i(v)} dv$$

Transient Response of an Inverter

Logic Circuit: inverter *X*1 driving another inverter *X*2

An ***ideal voltage step*** from 0 to *VDD* with zero rise/fall time is applied to node *A*

- How compute the propagation delay t_{pd}, from the input step until node *B* crosses $V_{DD}/2$?



Transient Response of an Inverter

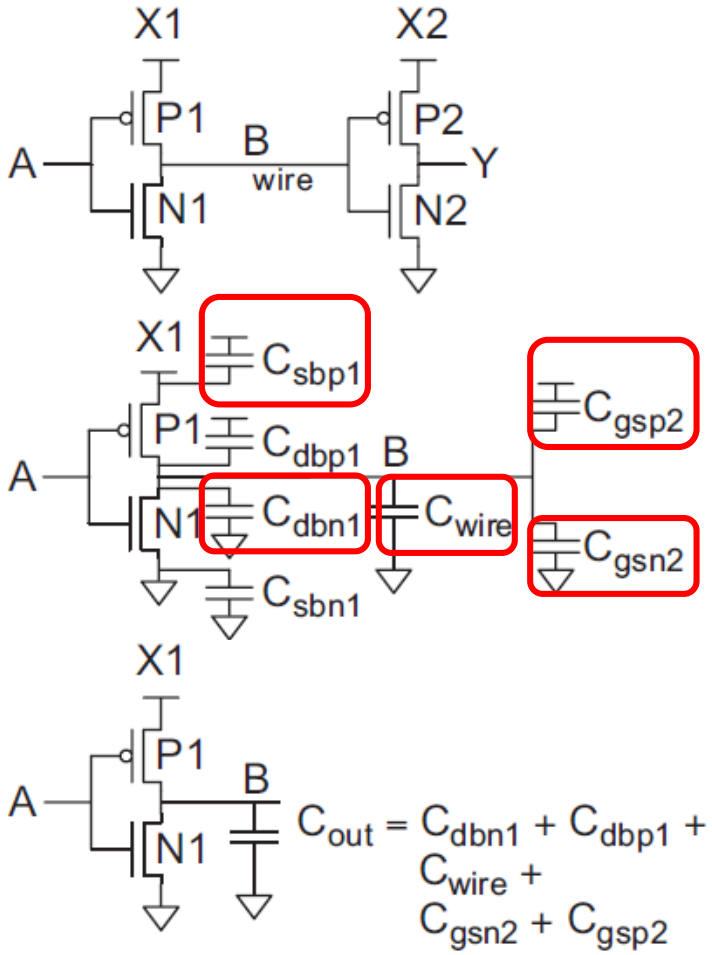
Delay Computation:

1. Build physical model of the circuit

Cdb and *Csb* : diffusion capacitances between the drain and body and source and body

Cgs: gate capacitance

Cwire: wire capacitance

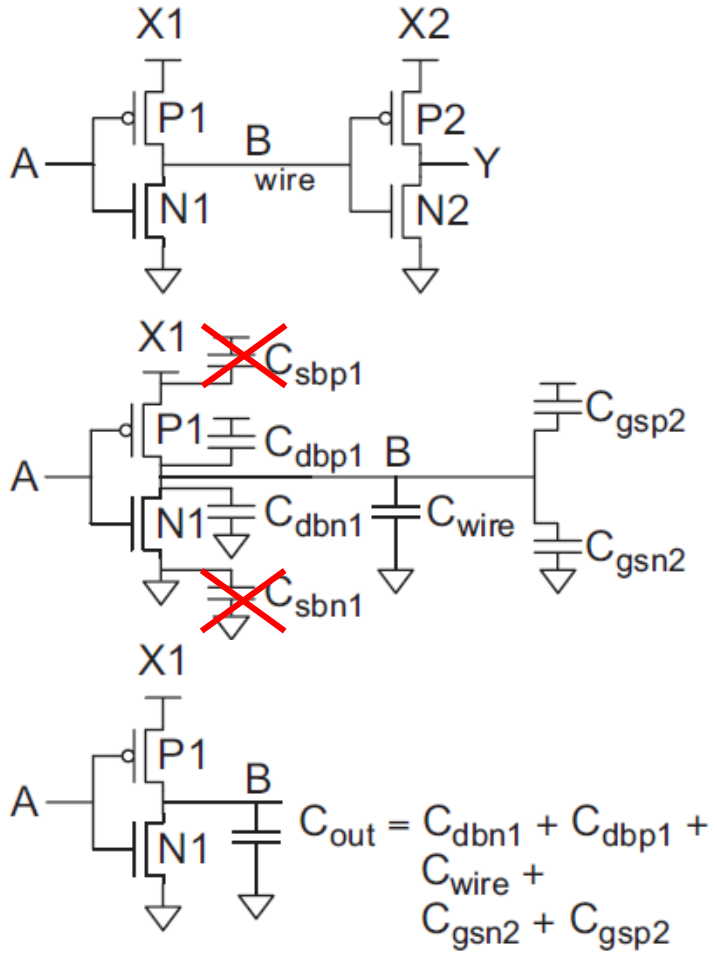


Transient Response of an Inverter

Delay Computation:

1. Build physical model of the circuit

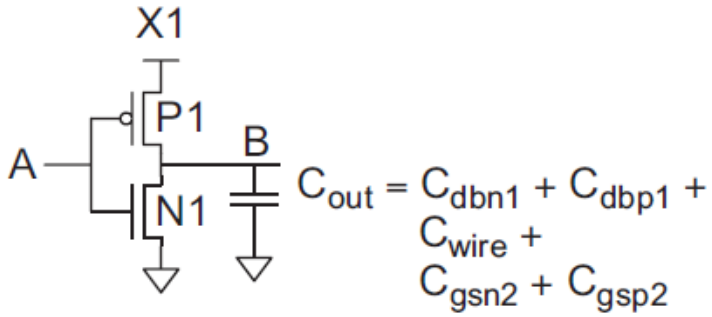
- *Cgs1* and *Cd* of transistors in *X*2 do not contribute to the switching capacitance: not connected to node *B*
- *Csbn*1 and *Csbp*1 do not contribute to the switching capacitance: both terminals tied to constant voltages
- All the capacitances are lumped into a single load capacitance C_{out}



Transient Response of an Inverter

Delay Computation:

2. Write differential equation of output voltage as a function of input voltage and time



- **Before** the ideal voltage step:

A = 0; *N*1 is OFF; *P*1 is ON→*B* = *VDD*

- After the ideal voltage step:

A = 1; *N*1 turns ON and *P*1 turns OFF and *B* drops toward 0

Transient Response of an Inverter

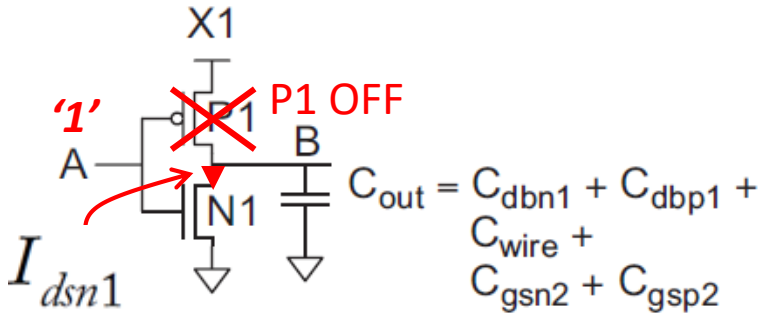
Delay Computation:

2. Write differential equation of output voltage as a function of input voltage and time

- **After** the ideal voltage step:

A = 1; *N*1 turns ON and *P*1 turns OFF and *B* drops toward 0

The current I_{DSN1} discharges C_{out} and depends on whether *N*1 is in linear or saturation regime



The rate of change *VB* depends on the output capacitance and on the current through *N*1

$$C_{out}\frac{dV_B}{dt} = -I_{dsn1}$$

Transient Response of an Inverter

Delay Computation:

2. Write differential equation of output voltage as a function of input voltage and time

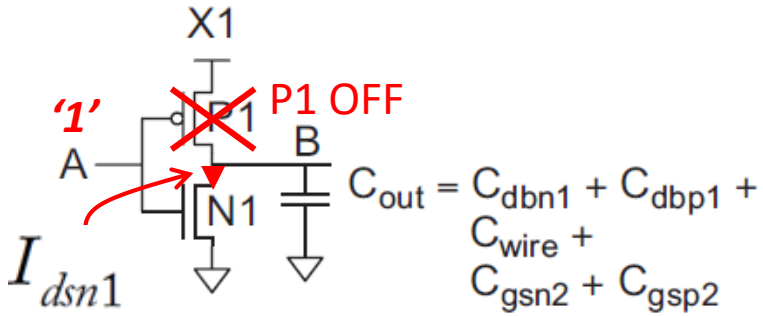
After the ideal voltage step :

A = 1; *Vgsn1* = V_{DD} → *N*1 turns ON

Initially, *Vdsn1* = V_B = V_{DD}

→ $V_B > V_{DD} - V_t$

→ *N*1 is in saturation



$$C_{\text{out}} \frac{dV_B}{dt} = -I_{dsn1}$$

$$\frac{dV_B}{dt} = -\frac{\beta}{C_{\text{out}}} \begin{cases} \dfrac{(V_{DD} - V_t)^2}{2} & \text{saturation} \quad V_B > V_{DD} - V_t \\ \left(V_{DD} - V_t - \dfrac{V_B}{2}\right) V_B & V_B < V_{DD} - V_t \end{cases}$$

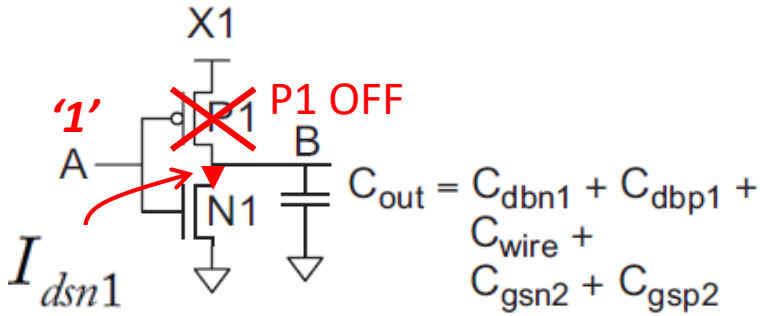
Transient Response of an Inverter

Delay Computation:

2. Write differential equation of output voltage as a function of input voltage and time

As *VB* falls below $V_{DD} - Vt$

→ *N*1 enters the linear regime



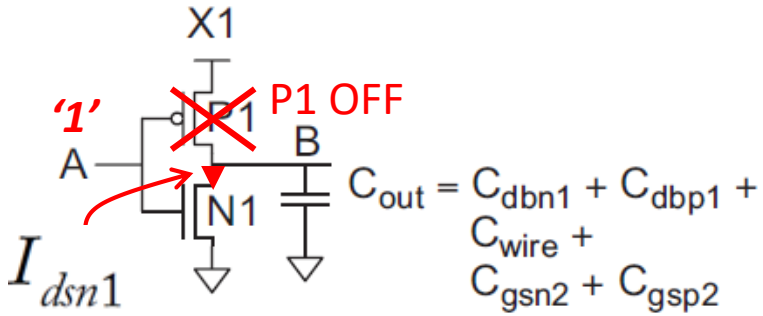
$$C_{\text{out}} \frac{dV_B}{dt} = -I_{dsn1}$$

$$\frac{dV_B}{dt} = -\frac{\beta}{C_{\text{out}}} \begin{cases} \dfrac{(V_{DD} - V_t)^2}{2} & \text{saturation} \quad V_B > V_{DD} - V_t \\ \left(V_{DD} - V_t - \dfrac{V_B}{2}\right) V_B & \text{linear} \quad V_B < V_{DD} - V_t \end{cases}$$

Transient Response of an Inverter

Delay Computation:

2. Write differential equation of output voltage as a function of input voltage and time



$$C_{\text{out}} \frac{dV_B}{dt} = -I_{dsn1}$$

- **Saturation**

I_{DSN1} is constant and *VB* drops linearly until it reaches *VDD-Vt*

- **Linear regime**

differential equation nonlinear, response computed numerically

$$\frac{dV_B}{dt} = -\frac{\beta}{C_{\text{out}}} \begin{cases} \dfrac{(V_{DD} - V_t)^2}{2} & \text{saturation } V_B > V_{DD} - V_t \\ \left(V_{DD} - V_t - \dfrac{V_B}{2}\right) V_B & \text{linear } V_B < V_{DD} - V_t \end{cases}$$

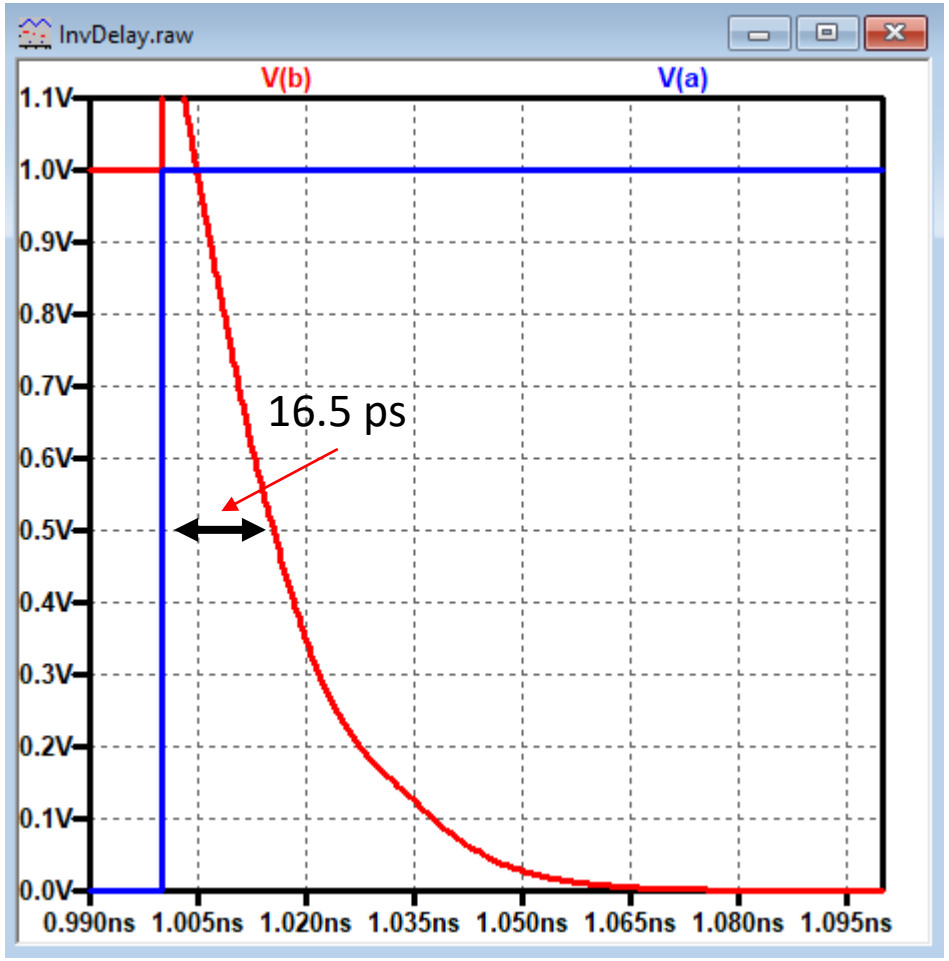
Transient Response of an Inverter

Example: Plot of the response of an inverter to a **rising step** input

The blue line indicates the input voltage V(A), rising at 0.1ps

The red blue line indicates the output voltage, V(B):

- initially it follows a straight line, as the saturated transistor N1 behaves as a constant current source
- Then curves as it approaches 0 and the transistor N1 enters the linear regime



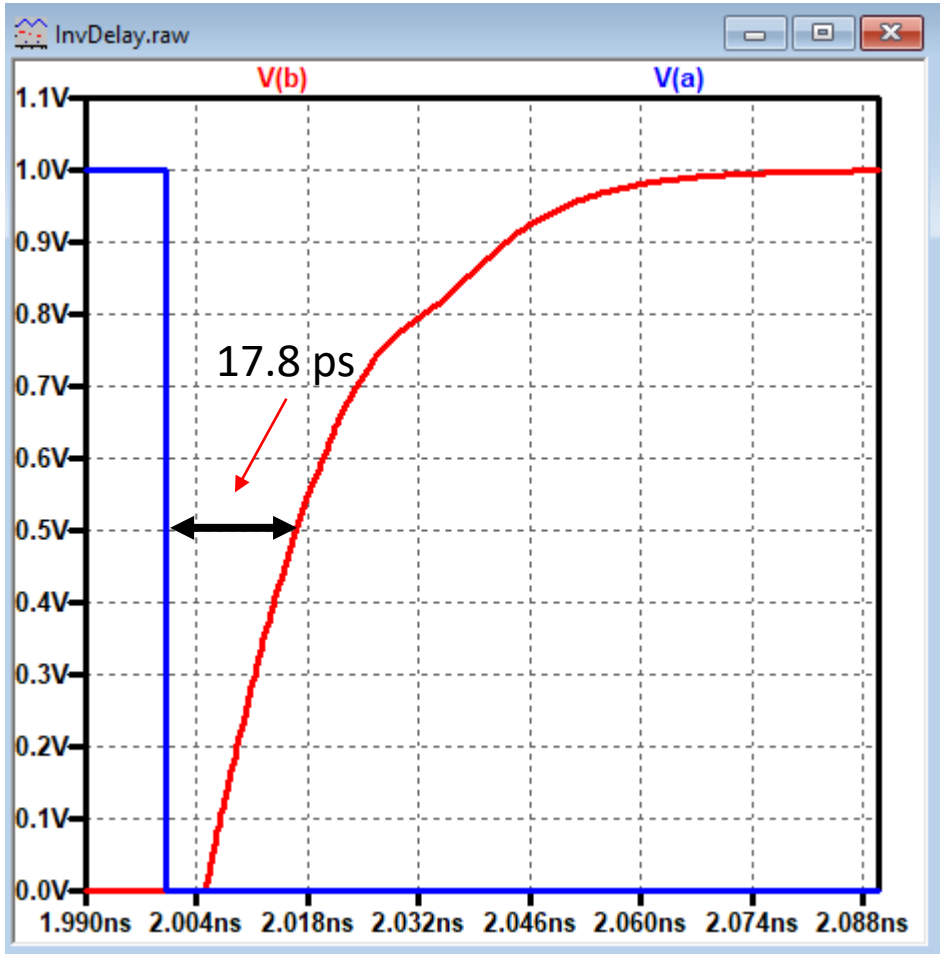
Transient Response of an Inverter

Example: Plot of the response of an inverter to a **falling step** input

The blue line indicates the input voltage V(A), falling at 0.1ps

The red blue line indicates the output voltage, V(B):

- initially it follows a straight line, as the saturated transistor P1 behaves as a constant current source
- Then curves as it approaches 0 and the transistor P1 enters the linear regime

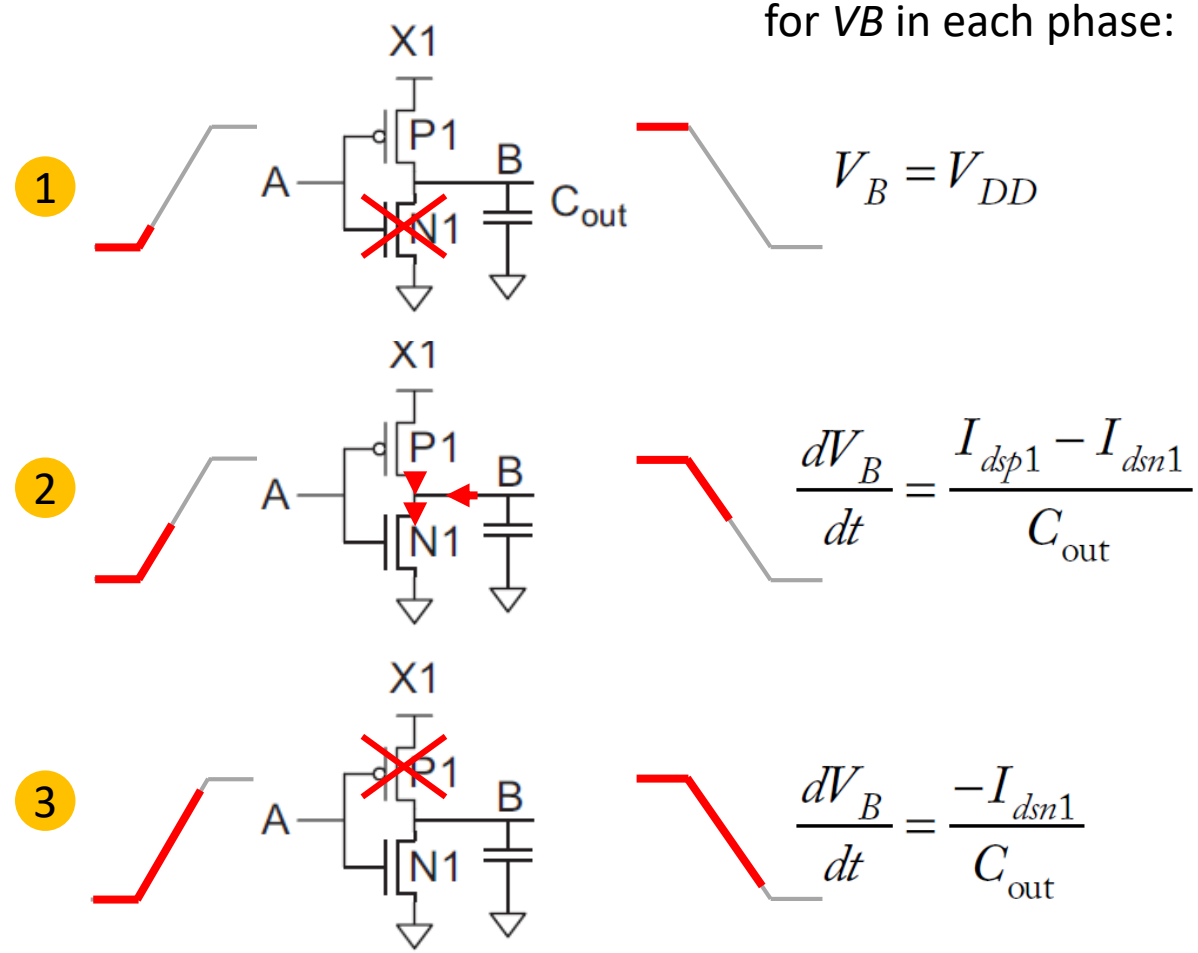


Transient Response of an Inverter

- In a real circuit, the input has a **non-zero rise/fall time**
- Three phases:

1. When *A* starts to rise, *P*1 is ON and *N*1 remains OFF and *B* remains at V_{DD}
2. When *A* reaches *Vtn*, *P*1 is still ON, *N*1 turns ON and starts to gradually pull *B* down
3. When *A* gets close enough to *VDD*, *P*1 turns OFF and *B* falls to 0

Differential equations for *VB* in each phase:



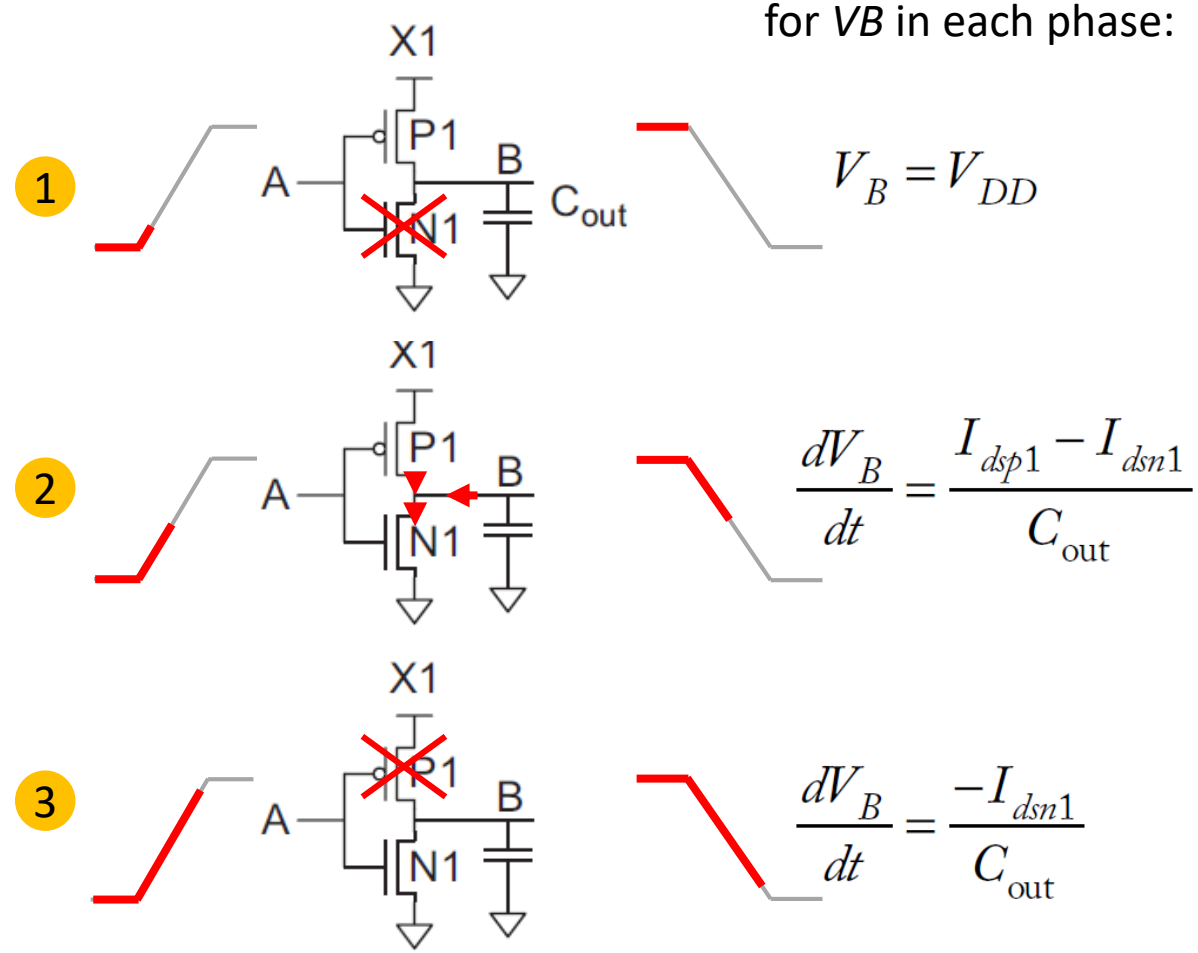
Transient Response of an Inverter

- In a real circuit, the input has a **non-zero rise/fall time**

The currents could be estimated using transistor current model

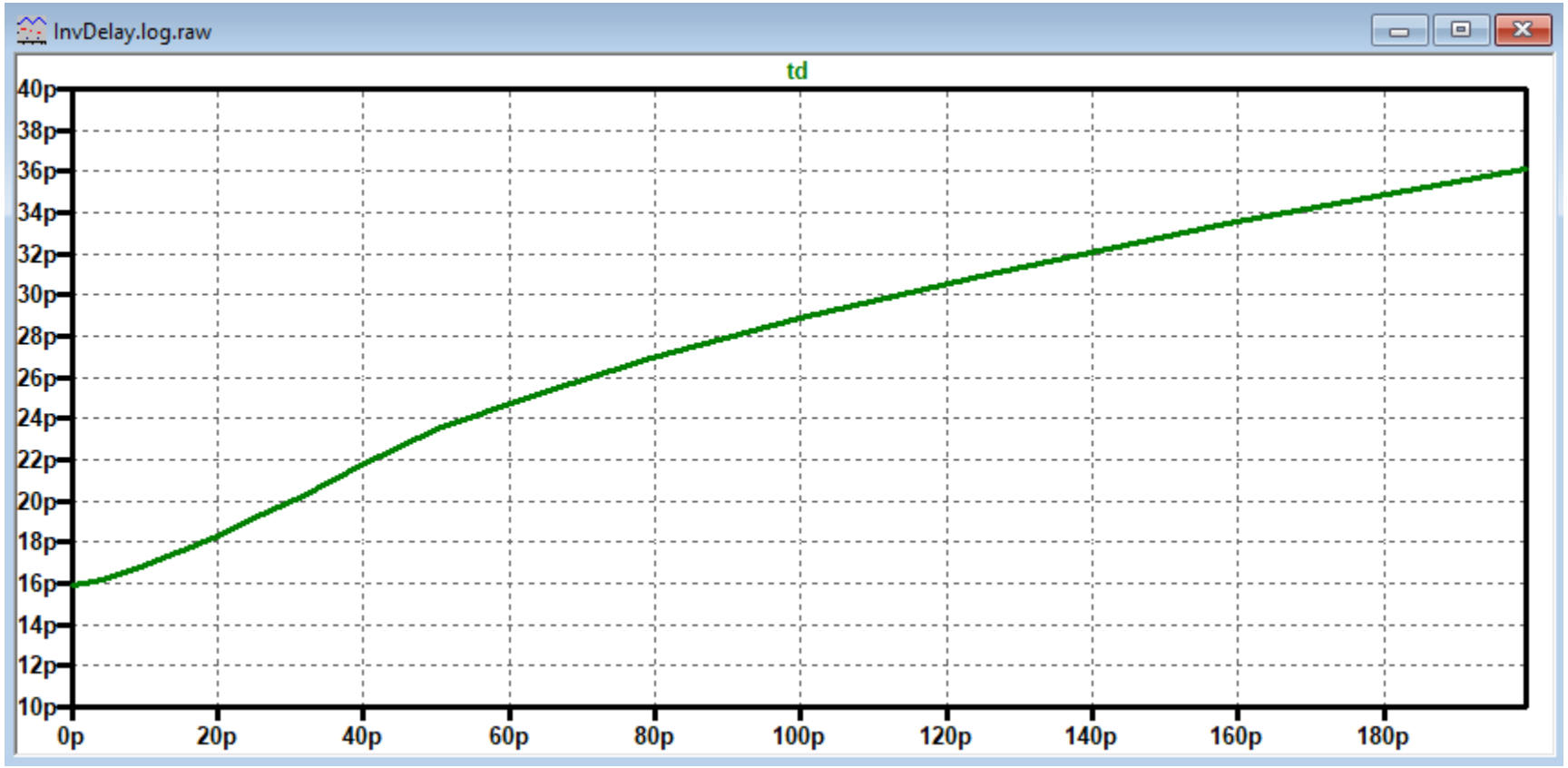
Key observation: the propagation delay increases with **non-zero rise/fall time** because *N*1/P1 is not fully ON right away and because it must struggle with *P*1/N1 in Phase 2

Differential equations for *VB* in each phase:



Transient Response of an Inverter

- Plot of average inverter propagation delay vs. input rise time.



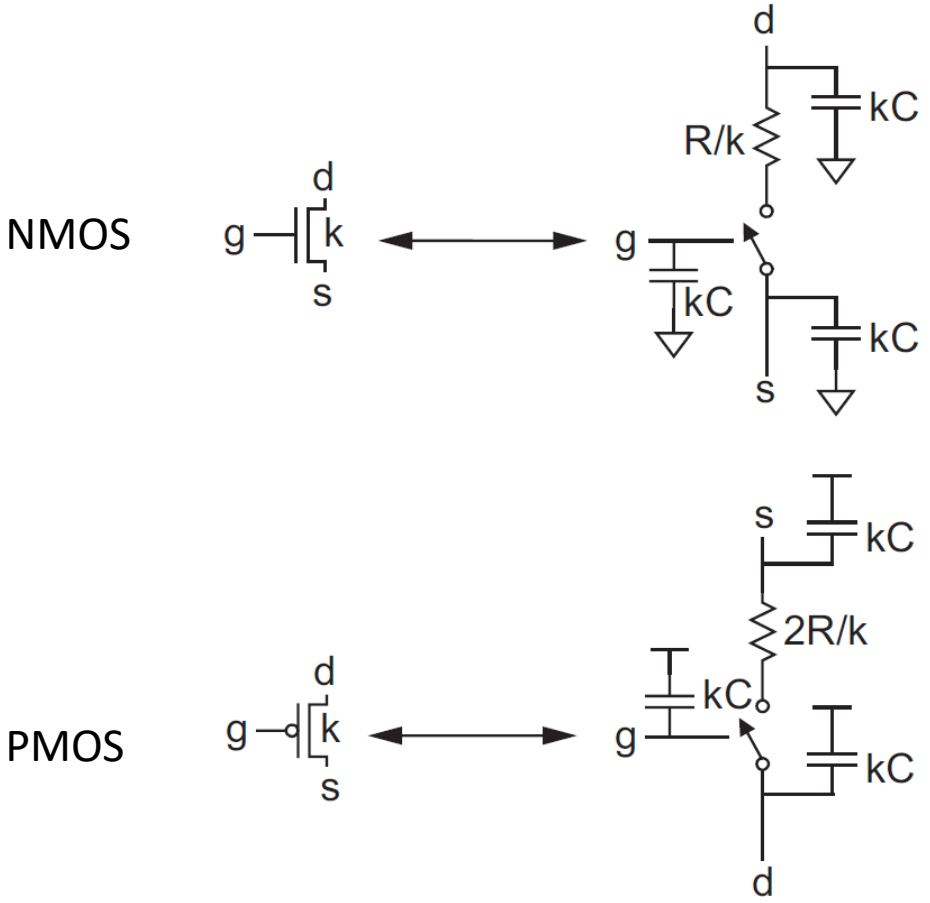
Transient Response of an Inverter

- The physical modelling give good insight on delay
  - delay increases with the output capacitance and decreases with the driver current
- However, equations are nonlinear to solve in closed form
  - simulators solve numerically delay equations and give accurate predictions of delay but offer less insight
- Need of a simpler delay model that offers more insight and tolerable accuracy

RC Delay Model

Nonlinear transistor I-V and C-V characteristics as an average resistance and capacitance over the switching interval

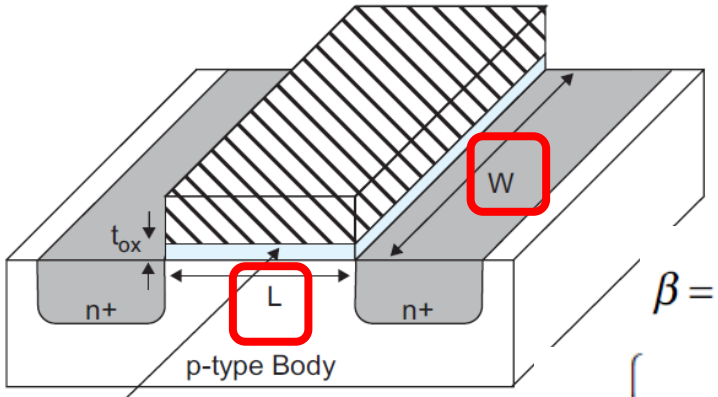
- transistor as a switch in series with a resistor with *effective resistance* + Gate and Diffusion Capacitances

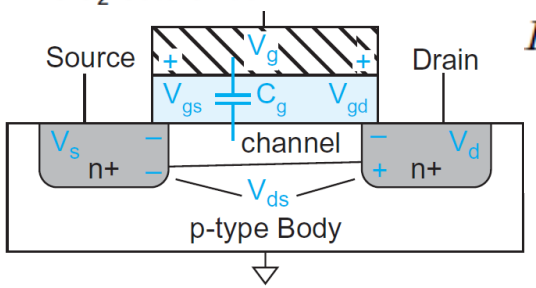


RC Delay Model: Effective Resistance

Effective resistance R is referred tc the unit nMOS transistor

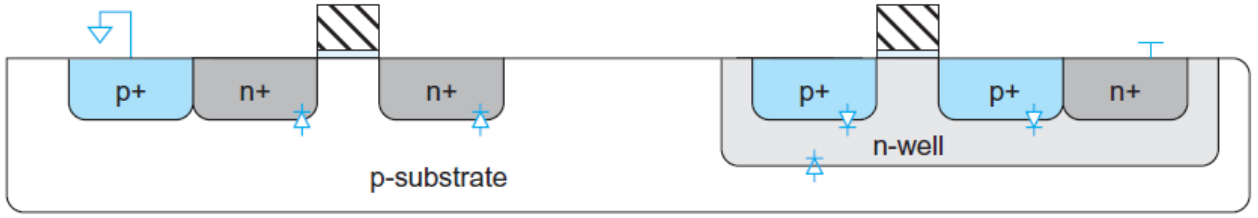
- unit transistor size: minimum length and minimum width
- an nMOS transistor of *k* times unit width delivers *k* times as much current → resistance=*R/k*





$$\beta = \mu C_{ox}\frac{W}{L};\ V_{GT} = V_{gs} - V_t$$

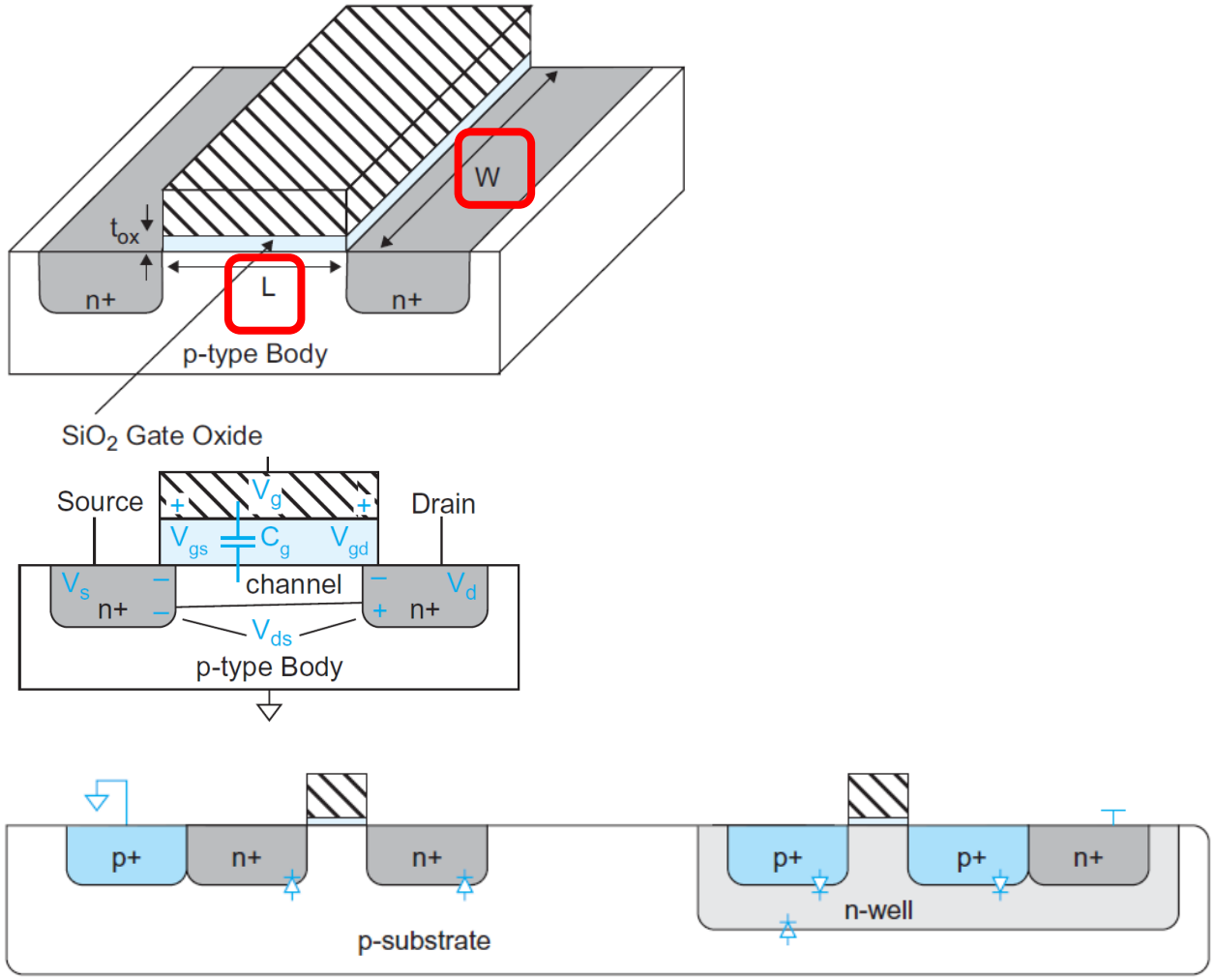
$$I_{ds} = \begin{cases} 0 & V_{gs} < V_t \quad \text{Off} \\ \beta\left(V_{GT} - V_{ds}/2\right)V_{ds} & V_{ds} < V_{dsat} \quad \text{Lin.} \\ \frac{\beta}{2}V_{GT}^2 & V_{ds} > V_{dsat} \quad \text{Sat.} \end{cases}$$



Note: Most transistors used in logic are of minimum length for greater speed and lower dynamic power consumption

RC Delay Model: Gate and Diffusion Capacitance

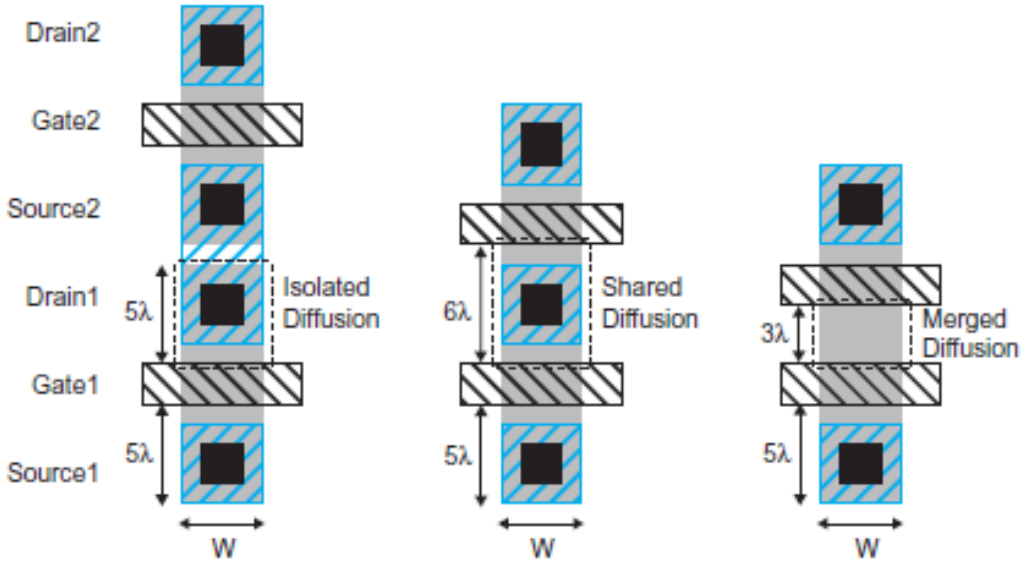
- *C* to be the gate capacitance of a unit transistor of NMOS
  - A transistor of *k* times unit width has capacitance *kC*
  - Increasing channel length increases gate capacitance proportionally
- Diffusion capacitance depends on the size of the source/drain region
  - Wider transistors have proportionally greater diffusion capacitance
  - Increasing channel length does not affect diffusion capacitance



RC Delay Model: Gate and Diffusion Capacitance

- For hand estimation, diffusion capacitances *Csb* and *Cdb* of contacted source and drain regions are comparable to the gate capacitance

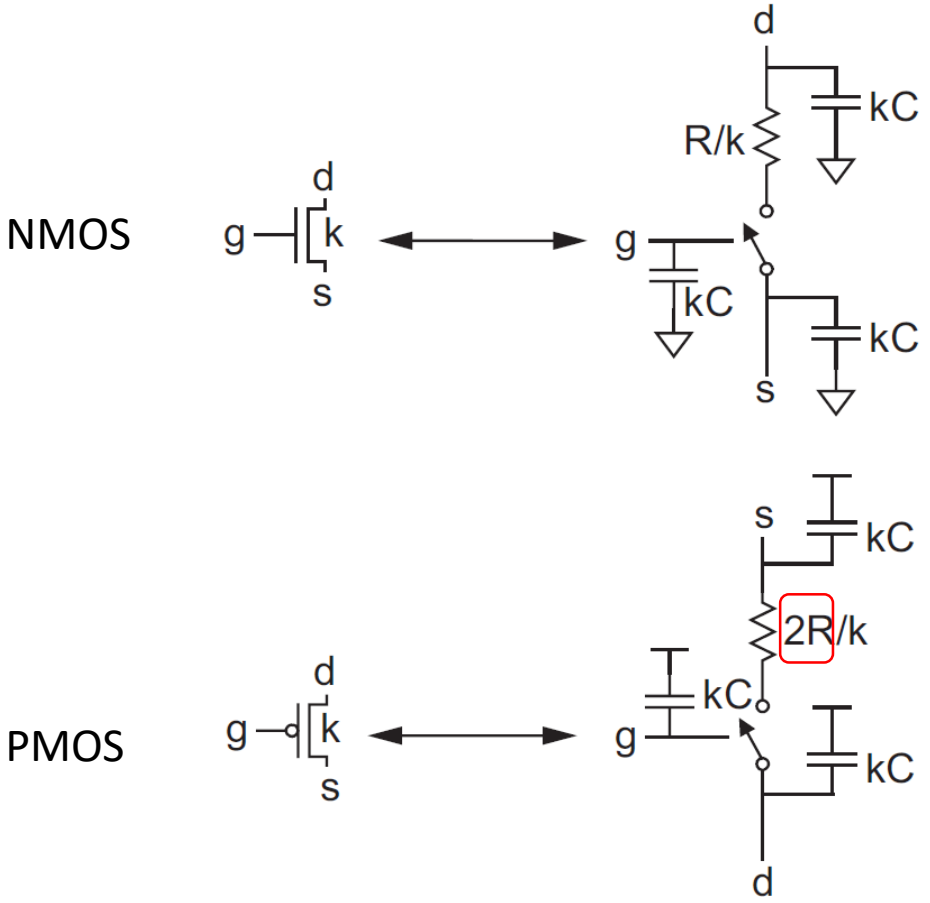
$$C_g = C_{ox} WL$$



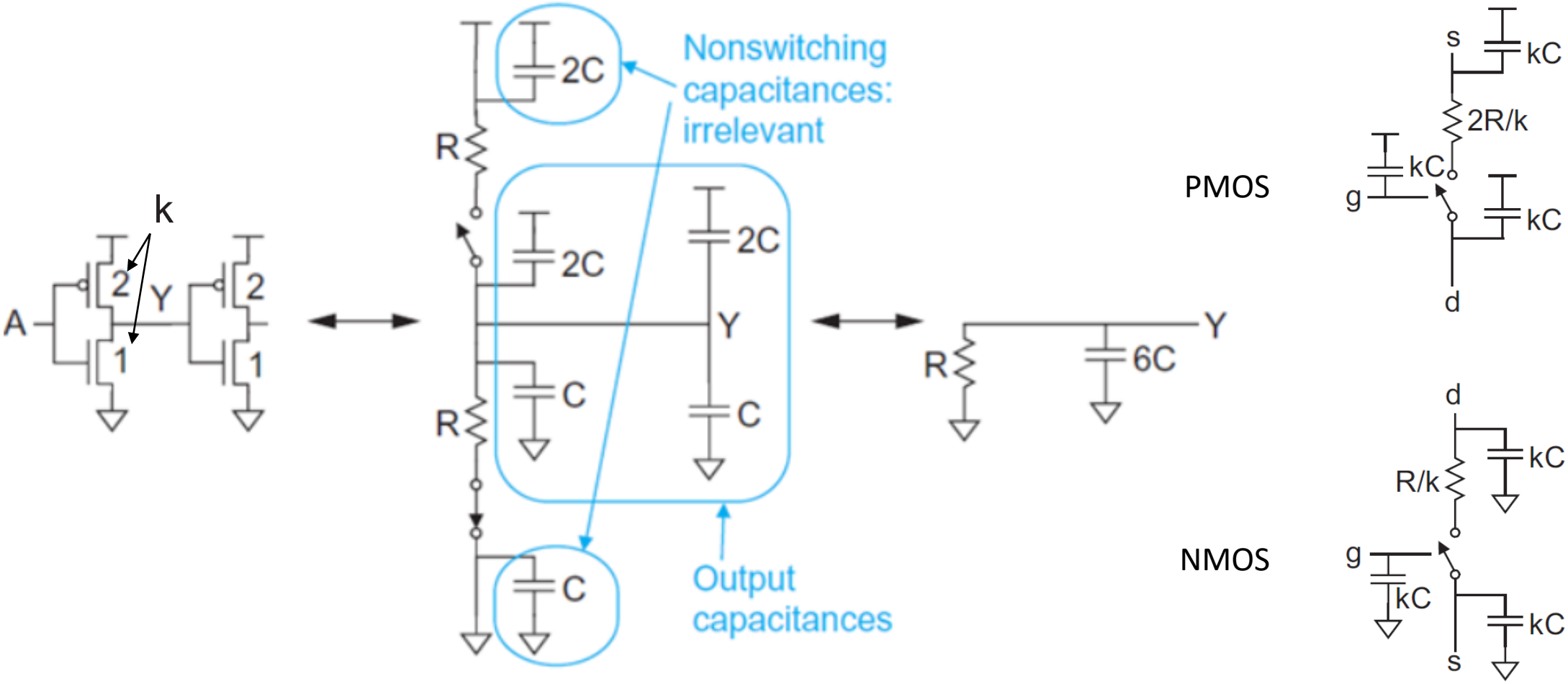
Equivalent RC Circuits

Equivalent RC circuit models for nMOS and pMOS transistors of width k

- the pMOS transistor has approximately **twice the resistance** of the nMOS transistor because holes have lower mobility than electrons

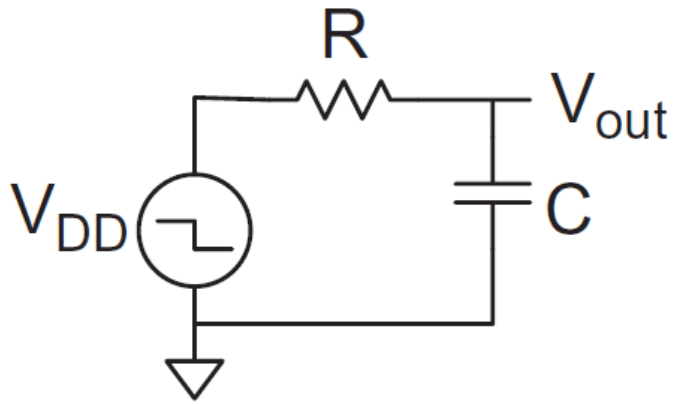


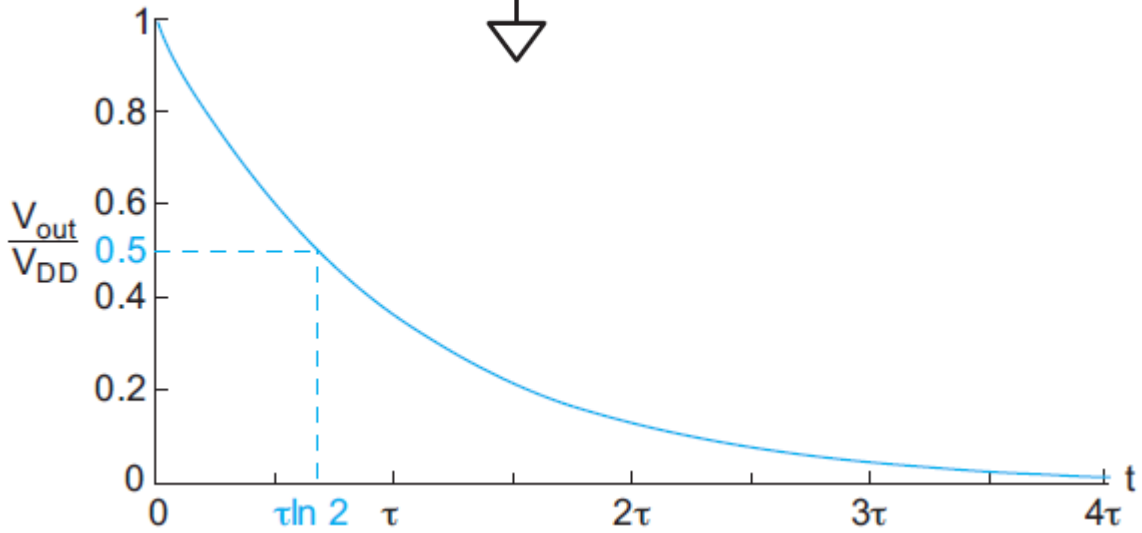
Equivalent RC circuit for an Inverter



Transient Response for an Inverter

- RC model to estimate the step response of the first-order circuit
- The propagation delay is the time at which *V*out reaches *VDD* /2





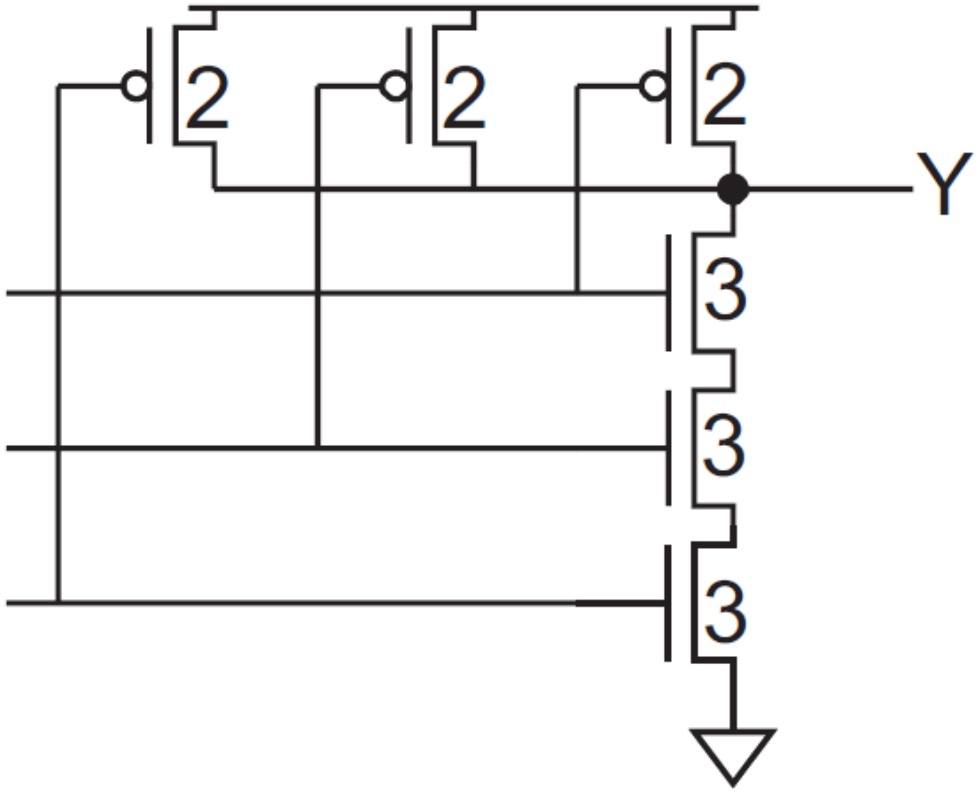
$$V_{\text{out}}(t) = V_{DD} e^{-t/\tau}$$

$$\tau = RC$$

$$\boxed{t_{pd} = RC \ln 2}$$

Propagation Delay of Complementary CMOS Gates

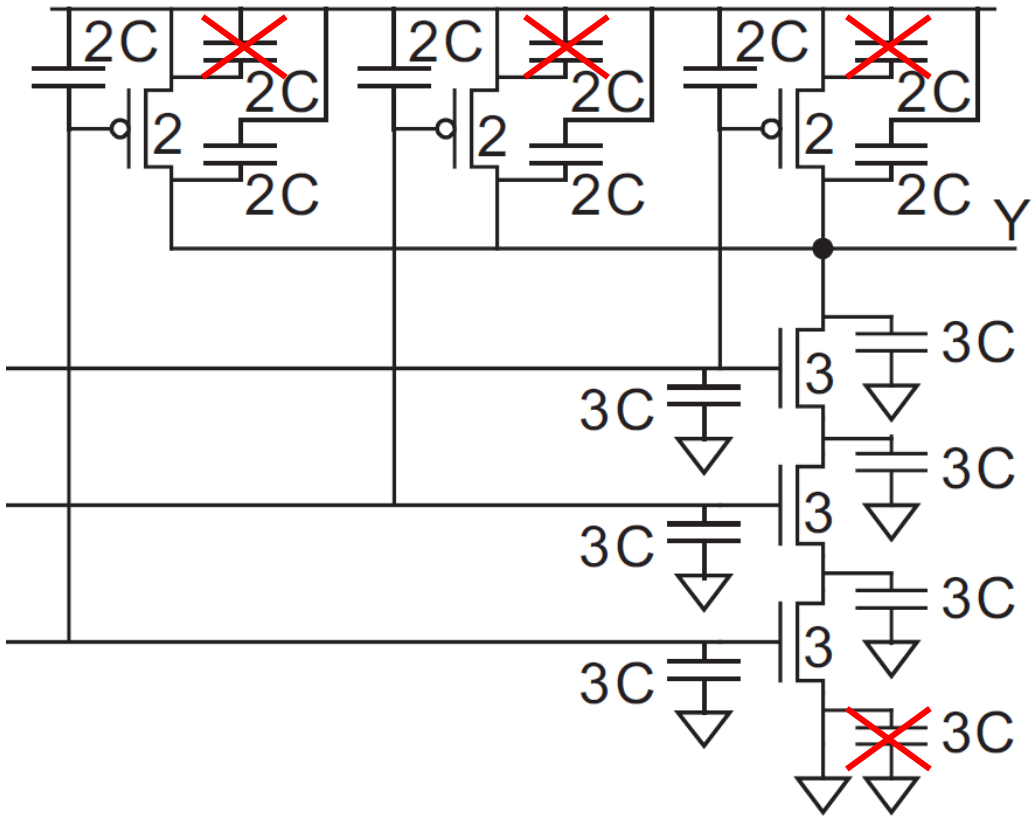
- NAND3 gate with effective rise and fall resistance equal to that of a unit inverter (*R*)
- three nMOS transistors in series
  - each with resistance *R*/3: the series combination has resistance *R*
- three pMOS transistors in parallel
  - each with twice unit width: when only one is ON it has resistance *R*



Propagation Delay of Complementary CMOS Gates

NAND3 gate with capacitances

- Each input presents five units of gate capacitance to whatever circuit drives that input
- pMOS source diffusions capacitors have both terminals shorted together so they are irrelevant to circuit operation

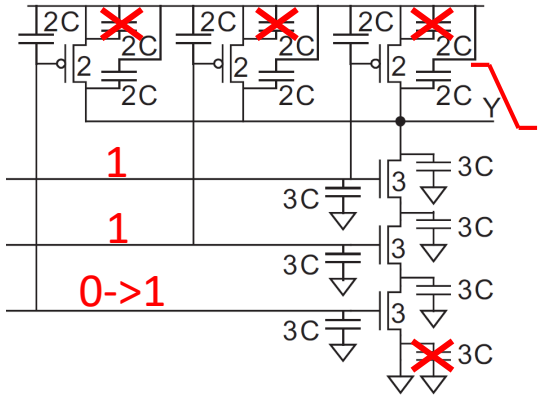


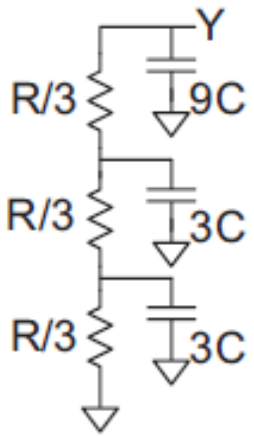
Propagation Delay of Complementary CMOS Gates

NAND3 gate: equivalent RC model

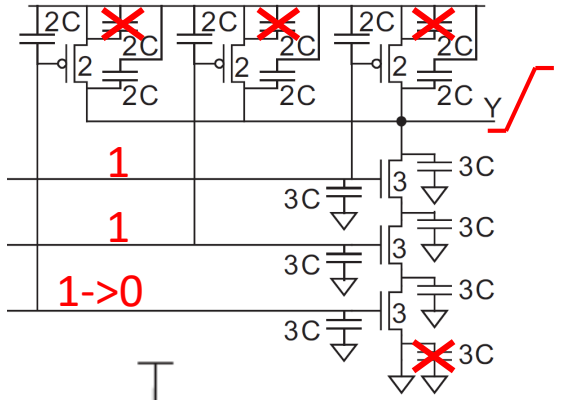
- worst case falling
  - the upper two inputs are 1 and the bottom one rises to 1
- worst case rising
  - the upper two inputs are 1 and the bottom one falls to 0
  - the output pulls up through a single pMOS transistor

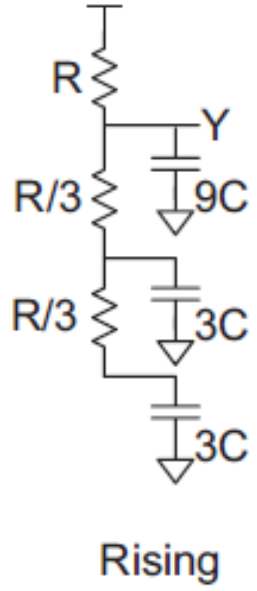
worst case falling





worst case rising



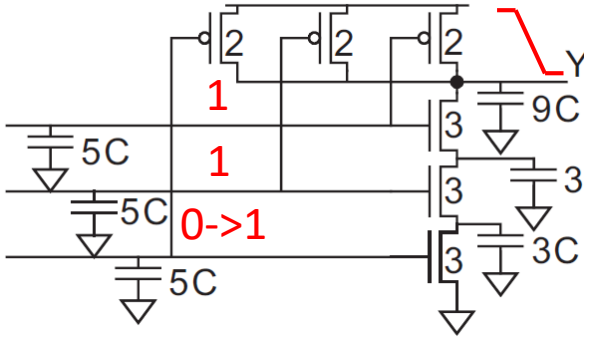


Propagation Delay of Complementary CMOS Gates

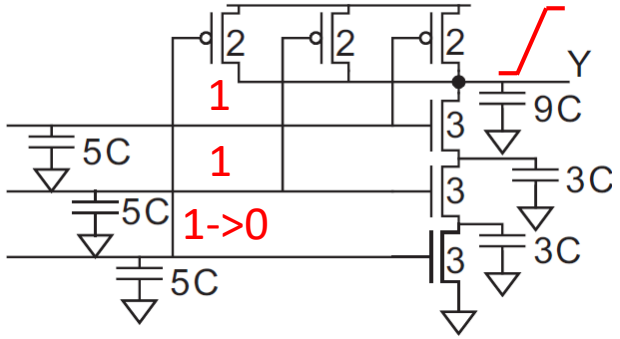
NAND3 gate: equivalent RC model

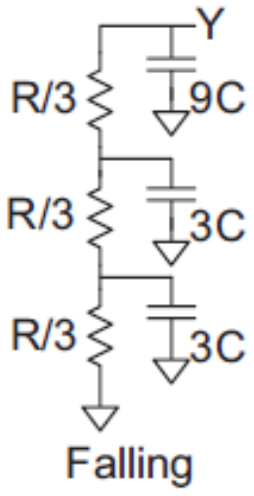
- Delay? second-order RC system
- complex problem that prevents the simplification of a CMOS circuit into an equivalent RC network!

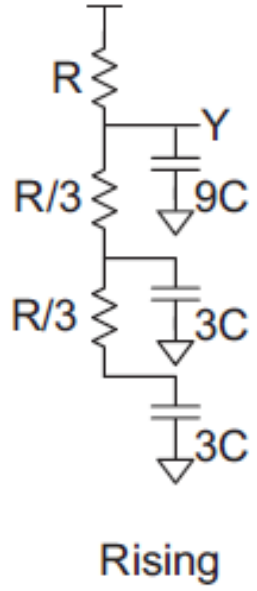
worst case falling



worst case rising



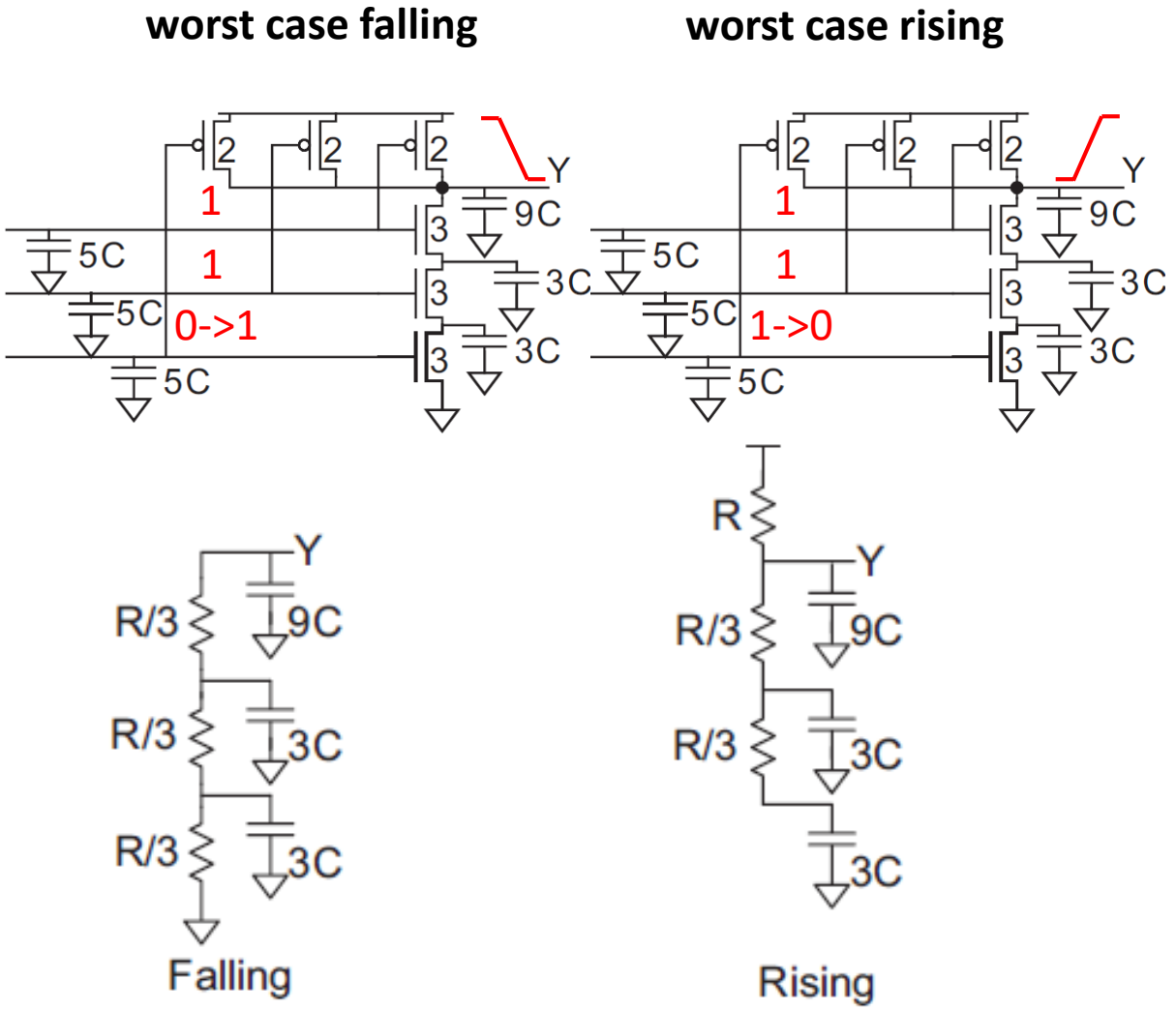




Propagation Delay of Complementary CMOS Gates

NAND3 gate: equivalent RC model

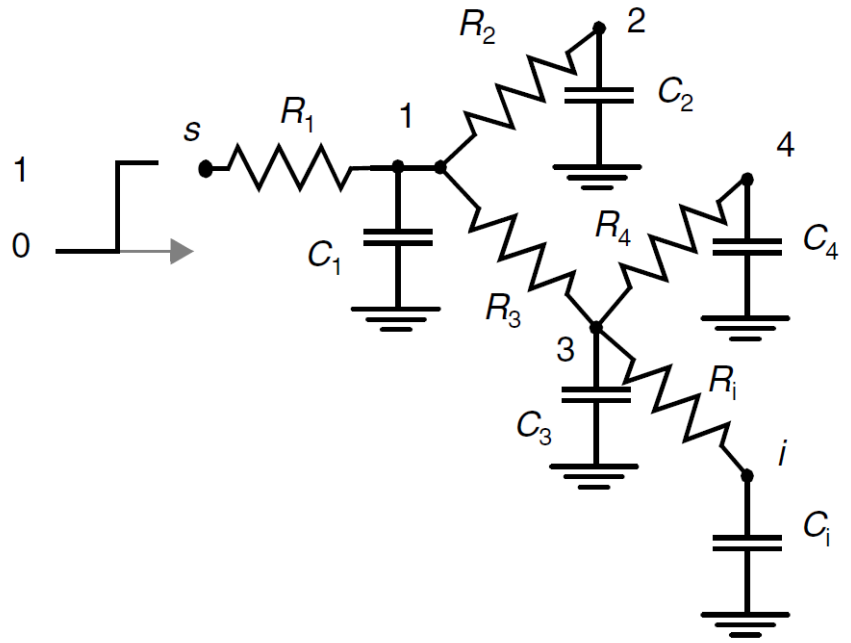
- Delay? Elmore Delay Model
- The equivalent RC circuit is an *RC tree*, i.e., an RC circuit with no loops



Elmore Delay Model

RC tree network, properties:

- the network has a single input node (*s*)
- all the capacitors are between a node and the ground
- the network does not contain any resistive loops (which makes it a tree)



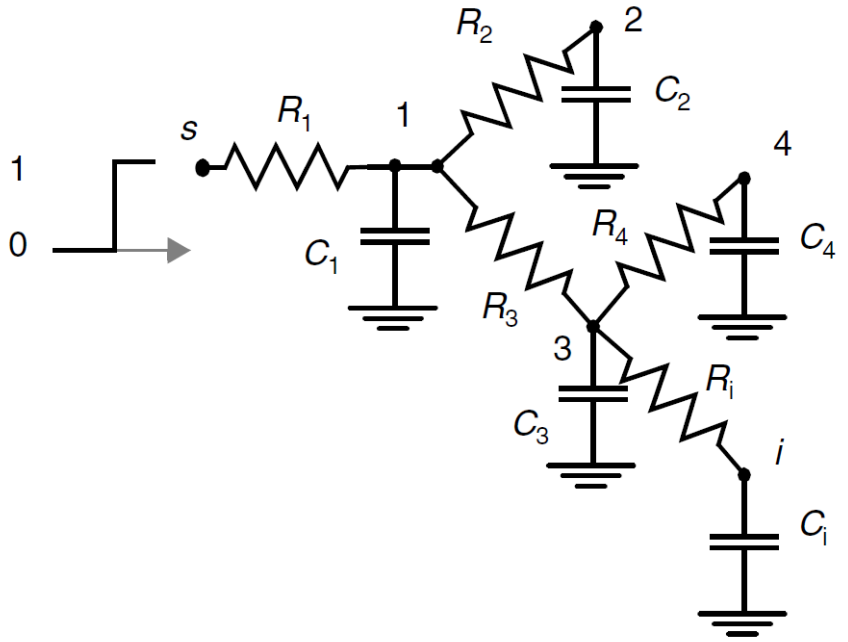
Elmore Delay Model

RC tree network, properties:

there exists a unique resistive path between the source node *s* and any node *i* of the network

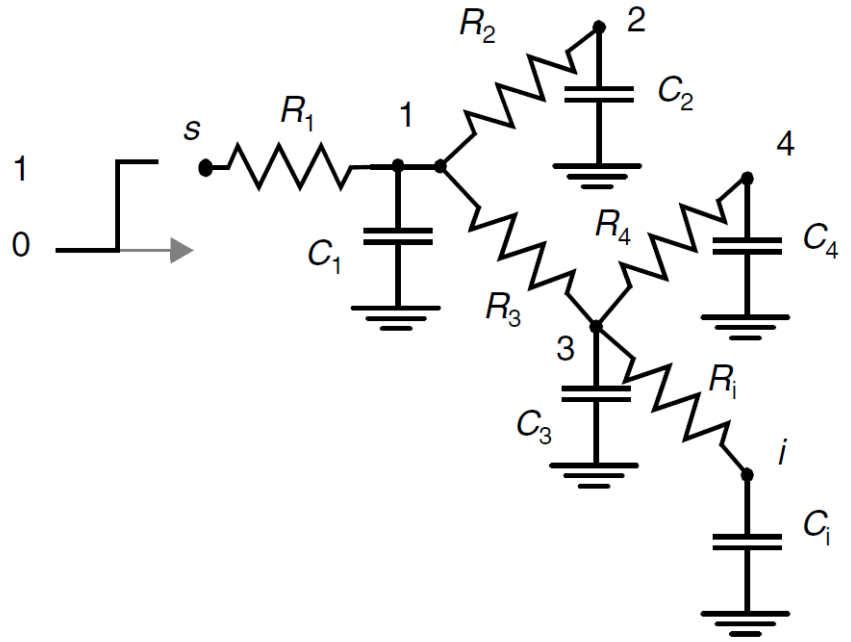
The total resistance along this path is called the *path resistance Rii*

- resistance between the node *s* and node 4 equals *R*s4 = *R*1 + *R*3 + *R*4
- *shared path resistance Rik*, which represents the resistance shared among the paths from the root node *s* to nodes *k* and *i*
- *Ri*4 = *R*1 + *R*3 while *Ri*2 = *R*1.



Elmore Delay Model

- Assumption: each of the *N* nodes of the network is initially discharged to GND
- A step input is applied at node *s* at time *t* = 0
- The Elmore delay at node *i* is given by the following expression:



$$\tau_{Di} = \sum_{k=1}^{N} C_k R_{ik}$$

$$\tau_{Di} = R_1C_1 + R_1C_2 + (R_1 + R_3)C_3 + (R_1 + R_3)C_4 + (R_1 + R_3 + R_i)C_i$$

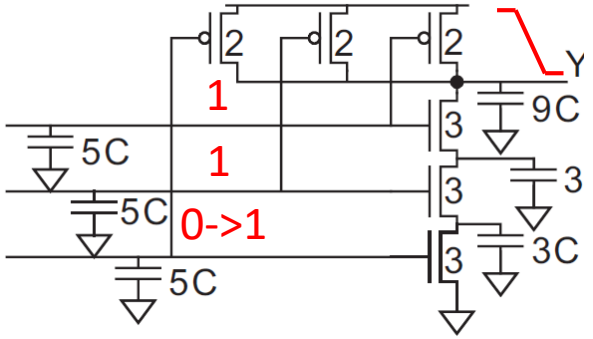
Propagation Delay of Complementary CMOS Gates

NAND3 gate: implementing Elmore Delay Model

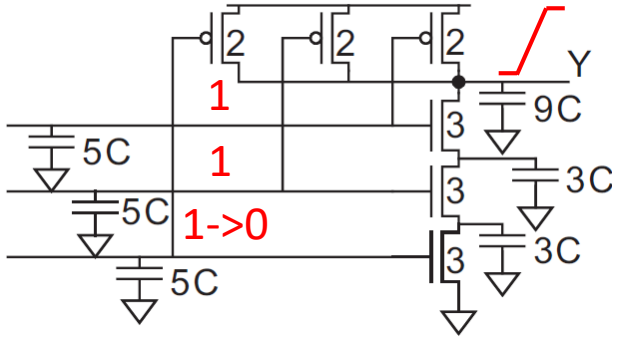
worst case:

Output not loaded

worst case falling

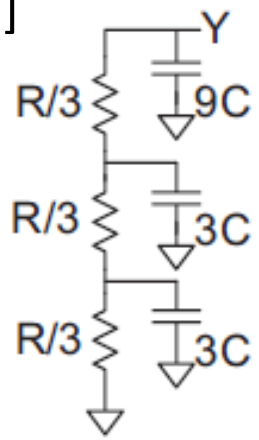


worst case rising

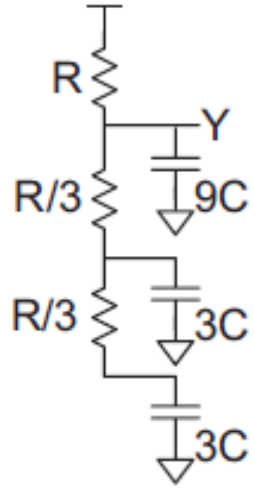


t_{pdf}=ln2[(3C)(R/3)+(3C)(R/3+R/3)+(9C)(R/3+R/3+R/3)]
=ln2(12RC)

t_{pdr}=ln2[(3C)(R)+(3C)(R)+(9C)(R)]
=ln2(15RC)



Falling



Rising

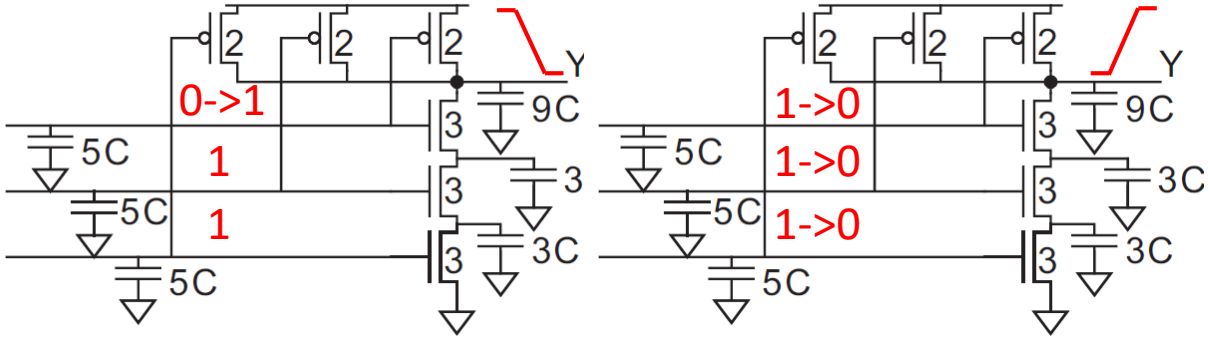
Propagation Delay of Complementary CMOS Gates

NAND3 gate: implementing Elmore Delay Model

best case: contamination delays

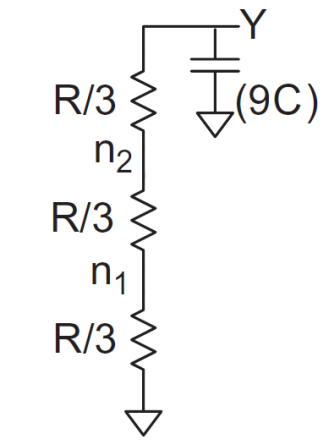
Output not loaded

worst case falling **worst case rising**

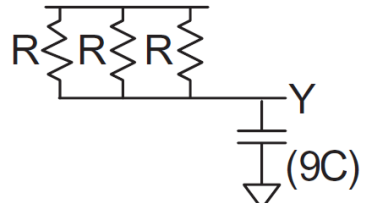


$t_{cdf}=\ln 2[(9C)(R/3+R/3+R/3)]=\ln 2(9RC)$

$t_{cdr}=\ln 2[(9C)(R/3)]=\ln 2(3RC)$



Falling



Rising

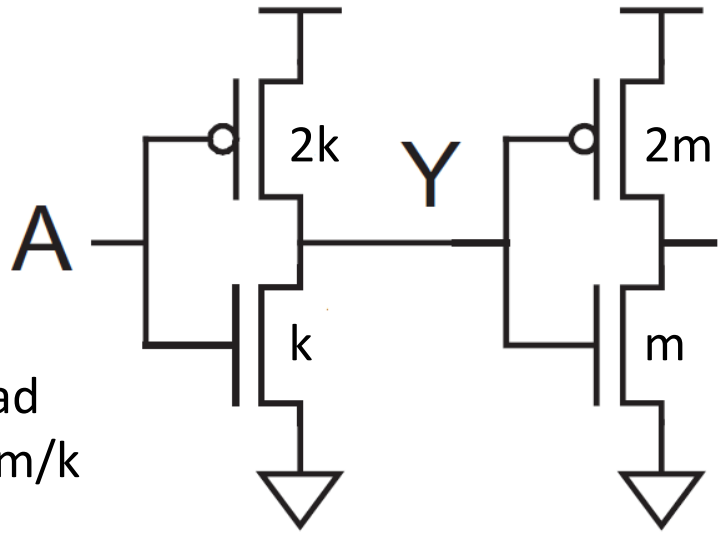
Propagation Delay of an Inverter

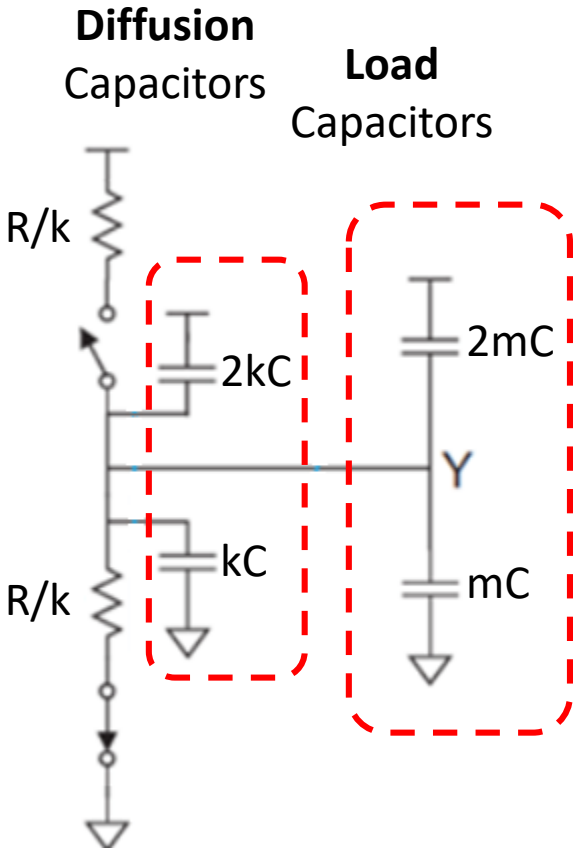
- Inverter driving *m* identical unit inverters

$t_{pdf} = t_{pdr} = \ln 2(3k+3m)(C)(R/k)$
$= \ln 2(3+3m/k)(RC)$

fanout or *electrical effort h*=ratio between load capacitance 3m and input capacitance 3k: h=m/k

$t_{pdf} = t_{pdr} = \ln 2(3k+3m)(C)(R/k)$
$= \ln 2(1+h)(3RC) = (1+h)\tau$ with $\tau = \ln 2 * 3RC$





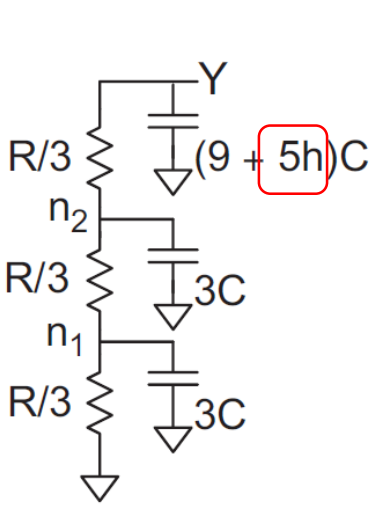
Propagation Delay of Complementary CMOS Gates

NAND3 gate driving h identical NAND3 gates

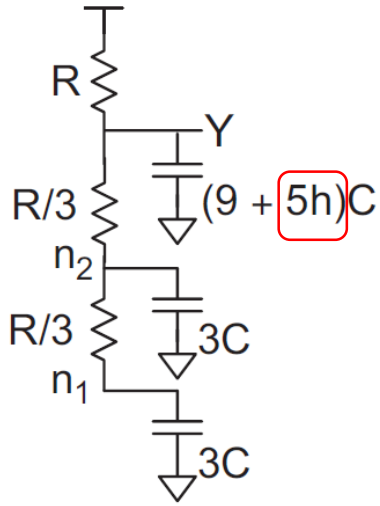
worst case:

t_{pdf}=ln2[(3C)(R/3)+(3C)(R/3+R/3)+(9C+5hC)(R/3+R/3+R/3)]
=ln2(12+5h)RC =ln2(4+5/3h)3RC = (4+5/3h)τ

t_{pdr}=ln2[(3C)(R)+(3C)(R)+(9C+5hC)(R)]=
ln2(15+5h)RC= (5+5/3h) τ



Falling



Rising

Propagation Delay of Complementary CMOS Gates

NAND3 gate driving h identical NAND3 gates

Normalized delay $t_{pd}/\tau = d = (p + f)$ consists of two components:

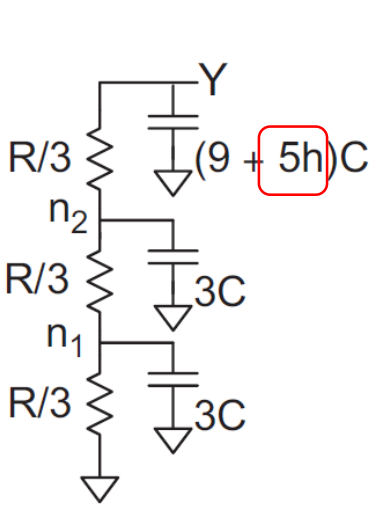
1. p: *parasitic delay*
   - time for a gate to drive its own internal diffusion capacitance
   - ideally independent of the gate size, increasing the width of the transistors decreases the resistance but increases the capacitance
2. f: effort delay, it depends on:
   - *electrical effort h* , ratio of external load capacitance to input capacitance and thus changes with transistor widths
   - *logical effort* the gate *g*, ratio of the input capacitance of the gate to the input capacitance of an inverter that can deliver the same output current

$t_{pdf}/\tau = (4+5/3h)$

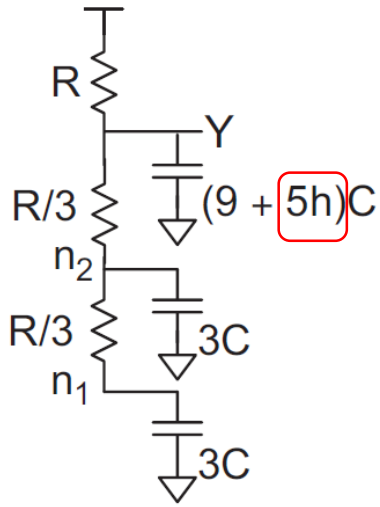
$d = (p+gh)$

$t_{pdr}/\tau = (5+5/3h)$

NAND3: parasitic delay of 5 and a logical effort of 5/3



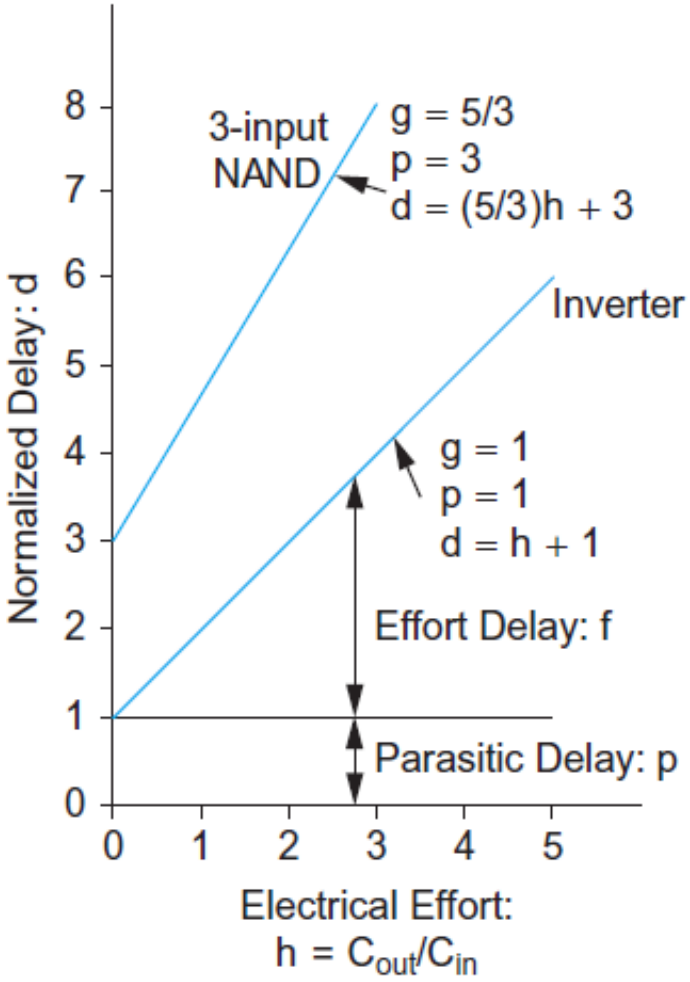
Falling



Rising

Linear Delay Model

- Linear delay model of a logic gate d = (p+gh) plotted as a function of the fanout (electrical effort h) for an inverter and for a NAND3 gate
- The slope of the line is the *logical effort* g, its intercept is the *parasitic delay p*
- We can adjust the delay by
  - adjusting the effective fanout (by transistor sizing)
  - choosing a logic gate with a different logical effort



Common Gates Parameters

Parasitic Delay

Gate Type	Number of Inputs				
	1	2	3	4	*n*
inverter	1				
NAND		2	3	4	*n*
NOR		2	3	4	*n*
tristate, multiplexer	2	4	6	8	2*n*

Logical Effort

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	$(2n+1)/3$
tristate, multiplexer	2	2	2	2	2
XOR, XNOR		4, 4	6, 12, 6	8, 16, 16, 8	

- NANDs are better than NORs
- Gates with few inputs are better than gates with many

Delay of a Multistage Network

- How to choose the fastest circuit topology and gate sizes for a specific logic function ?
- How to estimate the delay of the design?
- **Logical Effort method** for best topology and number of stages of logic for a function

Delay of a Multistage Network Logical Effort Method

- *G : path logical effort* $G = \prod g_i$
- *H : path electrical effort path effort F* $H = \frac{C_{out(path)}}{C_{in(path)}}$

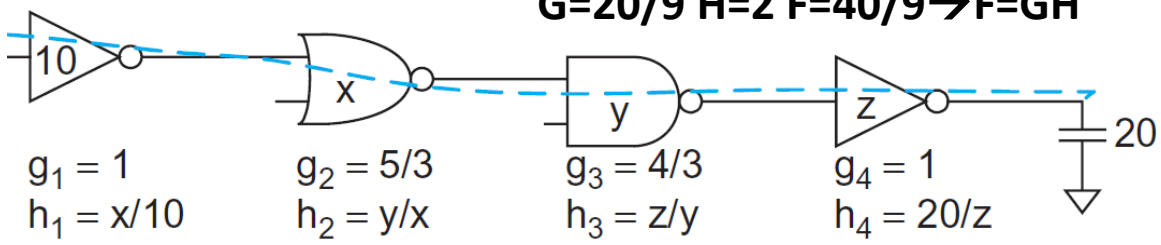
$F = \prod f_i = \prod g_i h_i$

Is *F=GH? NO* in branching paths

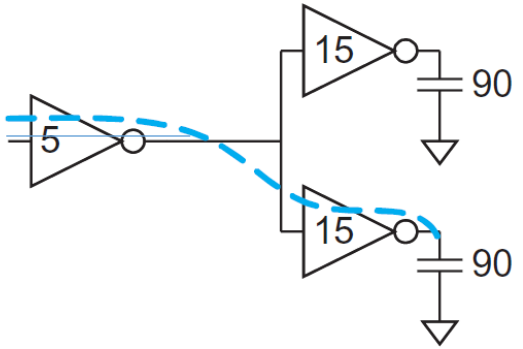
- *b : branching effort* $b = \frac{C_{onpath} + C_{offpath}}{C_{onpath}}$
- B *path branching effort* $B = \prod b_i$

$F = GBH$

G=20/9 H=2 F=40/9→F=GH



Multistage logic network



G=1 H=90/5=18
F = f1f2 = g1h1g2h2=1×6×1×6=36
F≠GH

Network with two-way branch

Delay of a Multistage Network

- *path delay D* $D = \sum d_i = D_F + P$
- *path effort delay D_F* $D_F = \sum f_i$
- *path parasitic delay P* $P = \sum p_i$

- The product of the stage efforts is *F*, independent of gate sizes
- The path effort delay D_F is the sum of the stage efforts
- The sum of a set of numbers whose product is constant is minimized by choosing all the numbers to be equal
- The path delay is minimised when each stage sustains the same effort

Delay of a Multistage Network

- If a path has *N* stages and each sustains the same effort, that effort must be:

$$\hat{f} = g_i h_i = F^{1/N}$$

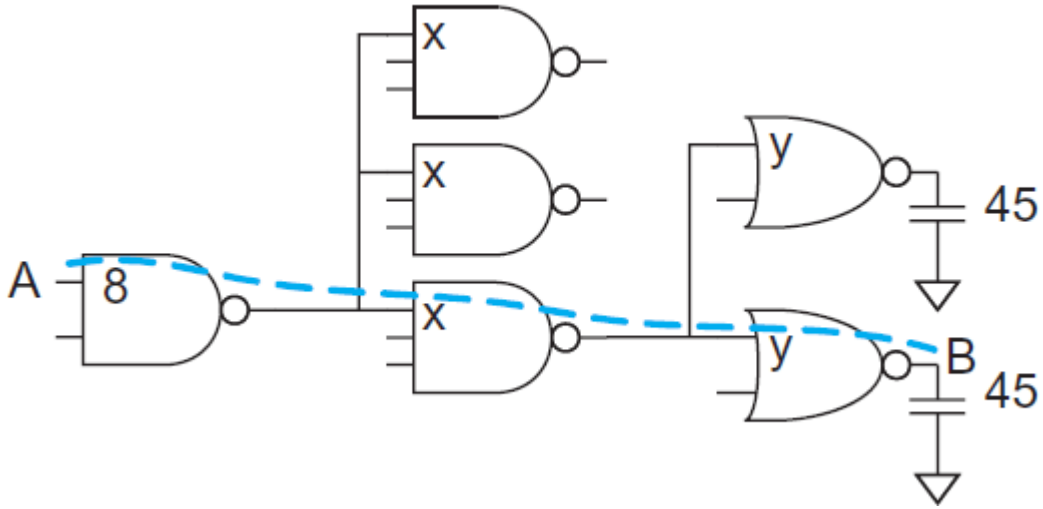
- The minimum possible delay of an *N*-stage path with path effort *F* and path parasitic delay *P* is:

$$D = NF^{1/N} + P$$

- The minimum delay of the path can be estimated knowing only the number of stages *N*, path effort *F*, and parasitic delays *P* without the need to assign transistor sizes

Delay of a Multistage Network: Exercise

- Estimate the minimum delay of the path from A to B
- choose transistor sizes to achieve this delay.
- The initial NAND2 gate may present a load of 8 of transistor width on the input and the output load is equivalent to 45 of transistor width



Delay of a Multistage Network: Exercise

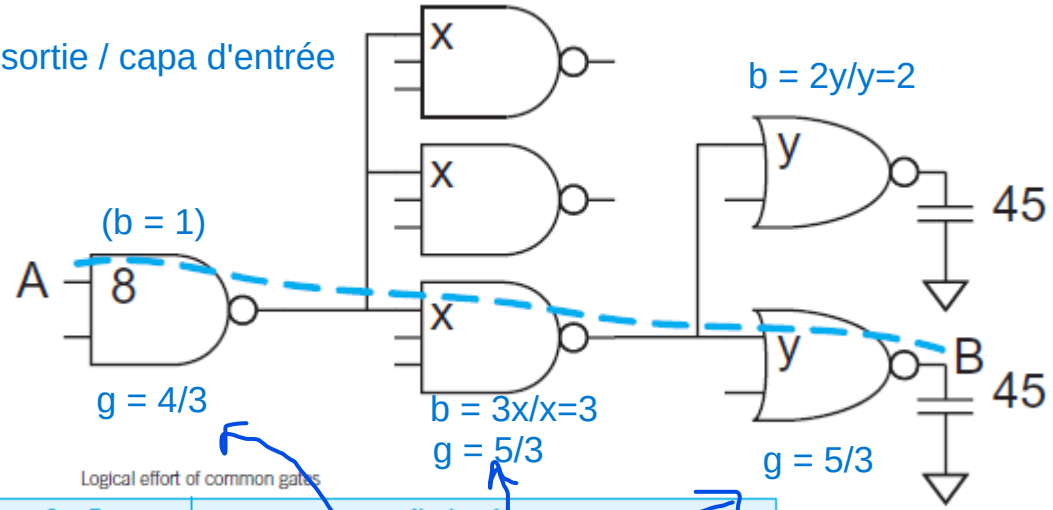
multiplication de tous les g

- *path logical effort* G=(4/3)×(5/3)×(5/3)=100/27
- *path electrical effort H* = 45/8
- *path branching effort B*=3×2=6
- *path effort F=BGH*=125
- *best stage effort* (N=3) $f = \sqrt[3]{125} = 5$
- *path parasitic delay P*=2+3+2=7
- minimum path delay *D*= 3×5+7= 22 in units of τ

Parasitic delay of common gates

Gate Type	Number of Inputs				
	1	2	3	4	*n*
inverter	1				
NAND		2	3	4	*n*
NOR		2	3	4	*n*
tristate, multiplexer	2	4	6	8	2*n*

capa de sortie / capa d'entrée



Logical effort of common gates

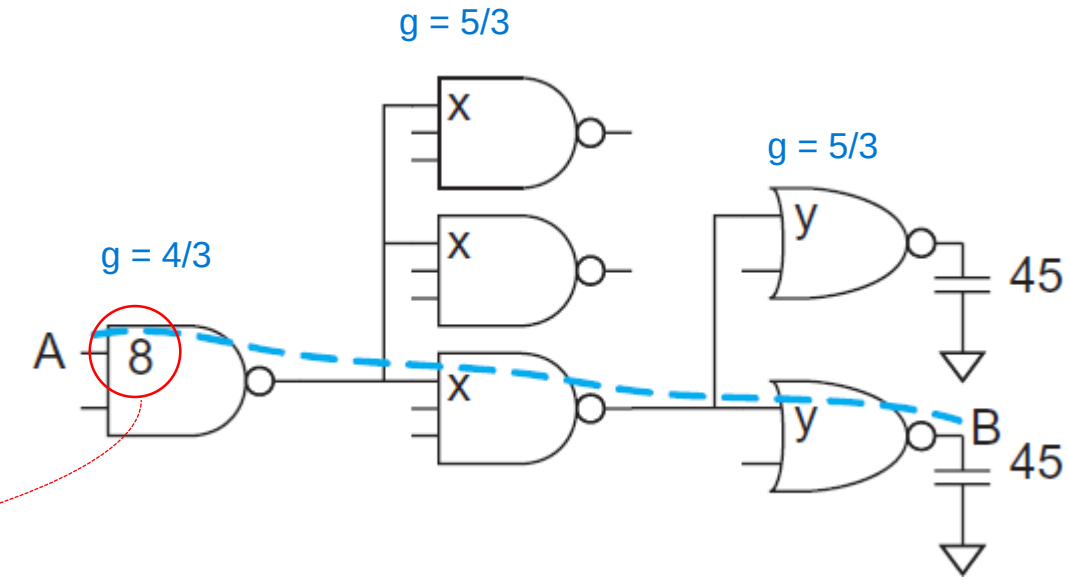
Gate Type	Number of Inputs				
	1	2	3	4	*n*
inverter	1				
NAND		4/3	5/3	6/3	(*n* + 2)/3
NOR		5/3	7/3	9/3	(2*n* + 1)/3
tristate, multiplexer	2	2	2	2	2
XOR, XNOR		4, 4	6, 12, 6	8, 16, 16, 8	

Delay of a Multistage Network: Exercise

- *Using capacitance transformation* formula:

$$C_{\text{in}_i} = \frac{C_{\text{out}_i} \times g_i}{\hat{f}}$$

- $y = C_{in3} = 45 \times (5/3)/5 = 15$
- $x = C_{in2} = (15+15) \times (5/3)/5 = 10$
- $8 = (10+10+10) \times (4/3)/5$

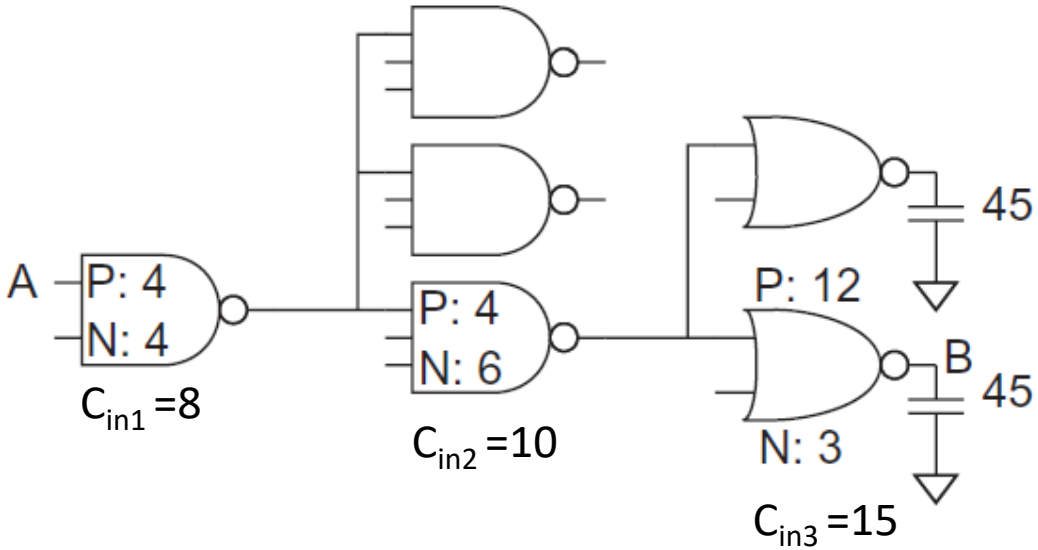


Delay of a Multistage Network: Exercise

- The transistor sizes chosen to give the desired amount of input capacitance while achieving equal rise and fall delays:

  P N

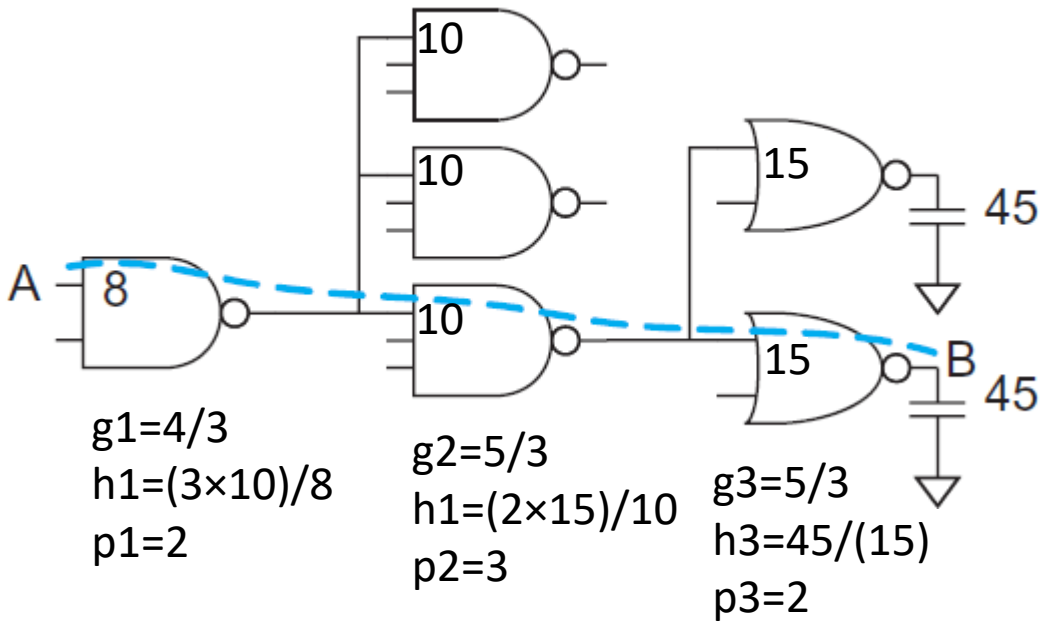
- Basic NAND2: Cin =2C+2C
- Basic NAND3: Cin =2C+3C
- Basic NOR2: Cin =4C+1C



Delay of a Multistage Network: Exercise

Path delay verification:

- D=(g1×h1+p1)+(g2×h2+p2)+(g3×h3+p3)=22 τ

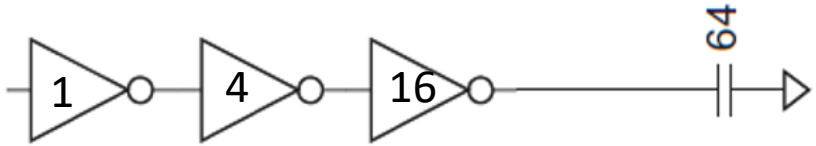


Choosing the Best Number of Stages: Exercise

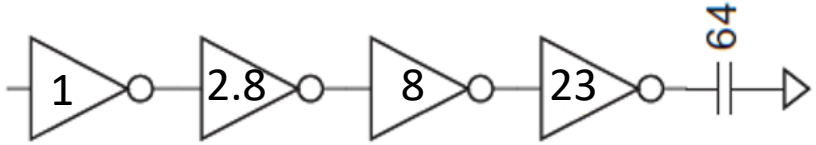
- A signal generated from a unit-sized inverter must drive 64 unit-sized inverters, calculate the optimal number of stages for minimizing the delay

N=1; h=64; p=1; g=1 → D=65

N=2; f= $\sqrt{64} = 8$; D=2*8+2=18



N=3; f= $\sqrt[3]{64} = 4$; D=3*4+3=15

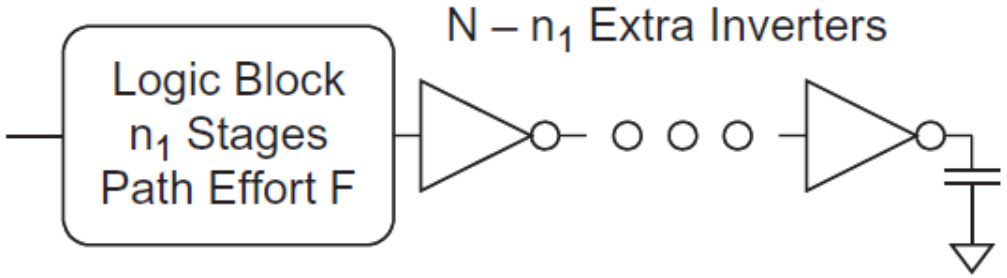


N=4; f= $\sqrt[4]{64} = 2.8$; D=4*2.8+4=15.3

- 3-stage design is the fastest
- For an even number of inversions (same polarity), the two- or four-stage designs are promising. Four-stage design is slightly faster, but the two-stage design requires significantly less area

Choosing the Best Number of Stages

- In general, it is always possible to add inverters to the end of a path without changing its function (save possibly for polarity)
- how many should be added for least delay?
- Logic path has *n*1 stages and a path effort of *F*
- Consider adding *N* – *n*1 inverters to the end to bring the path to *N* stages
- The extra inverters do not change the path logical effort but do add parasitic delay
- Delay of the new path is



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

Choosing the Best Number of Stages

- Delay of the new path is

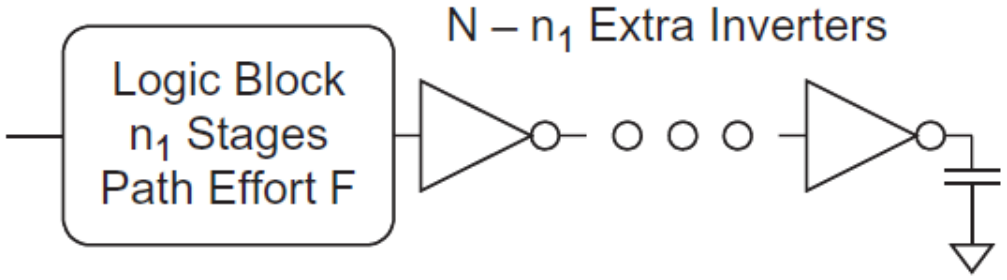
$$D = NF^{1/N} + \sum_{i=1}^{n_1} p_i + \left(N - n_1\right) p_{\text{inv}}$$

- Differentiating with respect to *N* and setting to 0 to find the best number of stages $\hat{N}$

$$\frac{\partial D}{\partial N} = -F^{1/N} \ln F^{1/N} + F^{1/N} + p_{\text{inv}} = 0$$

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

$$\hat{N} = \log_{\rho} F$$



- Solving numerically, when p_{inv} = 1, ρ= 3.5
- A path achieves least delay by using $\hat{N} = \log_{\rho} F$ stages

Summary

The method of Logical Effort is applied with the following steps:

1. Compute the path effort: $F = GBH$
2. Estimate the best number of stages: $\hat{N} = \log_4 F$
3. Sketch a path using: $\hat{N}$ stages
4. Estimate the minimum delay: $D = \hat{N}F^{1/\hat{N}} + P$
5. Determine the best stage effort: $\hat{f} = F^{1/\hat{N}}$
6. Starting at the end, work backward to find sizes: $C_{in_i} = \frac{C_{out_i} \times g_i}{\hat{f}}$

TABLE 4.5 Summary of Logical Effort notation

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

Summary

- The propagation delay of the CMOS inverter is determined by the time it takes to charge and discharge the load capacitor *CL* through the PMOS and NMOS transistors, respectively
- Getting *cl* as small as possible is crucial to the realization of high-performance CMOS circuits
- Logic gates are modelled as RC networks
- The Elmore delay model estimates the delay of the network
- The gate delay consists of a parasitic delay (accounting for the gate driving its own internal parasitic capacitance) plus an effort delay (accounting for the gate driving an external load)
- The effort delay depends on the electrical effort (the ratio of load capacitance to input capacitance, also called *fanout*) and the logical effort (which characterizes the current driving capability of the gate relative to an inverter with equal input capacitance)
- The method of logical effort builds on this linear delay model to help us quickly estimate the delay of entire paths based on the effort and parasitic delay of the path

References

- David Harris and Sarah Harris. 2007. Digital Design and Computer Architecture. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Neil Weste and David Harris. 2010. CMOS VLSI Design: A Circuits and Systems Perspective (4th. ed.). Addison-Wesley Publishing Company, USA.
- Rabaey, J. M.; Chandrakasan, A. & Nikolic, B. (2004), Digital integrated circuits- A design perspective , Prentice Hall .
- R. Jacob Baker. 2010. CMOS Circuit Design, Layout, and Simulation (3rd. ed.). Wiley-IEEE Press.
- I. Sutherland, B. Sproull, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits*, San Francisco, CA: Morgan Kaufmann, 1999.
- W. Elmore, “The transient response of damped linear networks with particular regard to wideband amplifiers,” J. Applied Physics, vol. 19, no. 1, Jan. 1948, pp. 55–63.