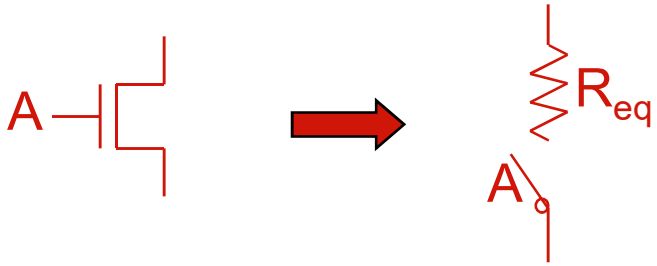
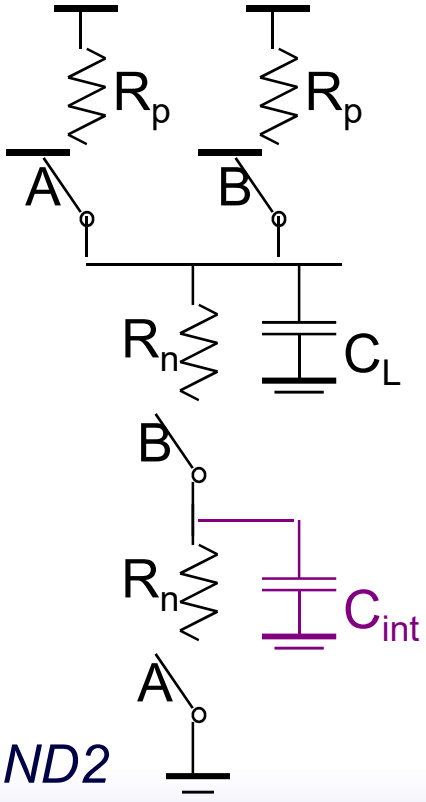
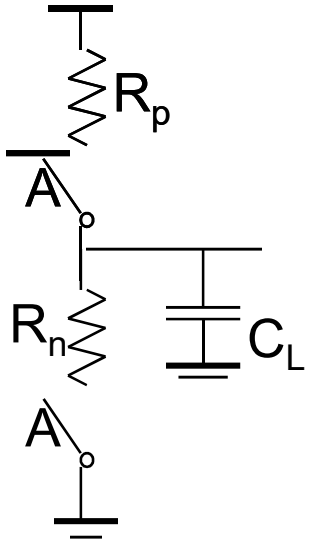
Switch Delay Model

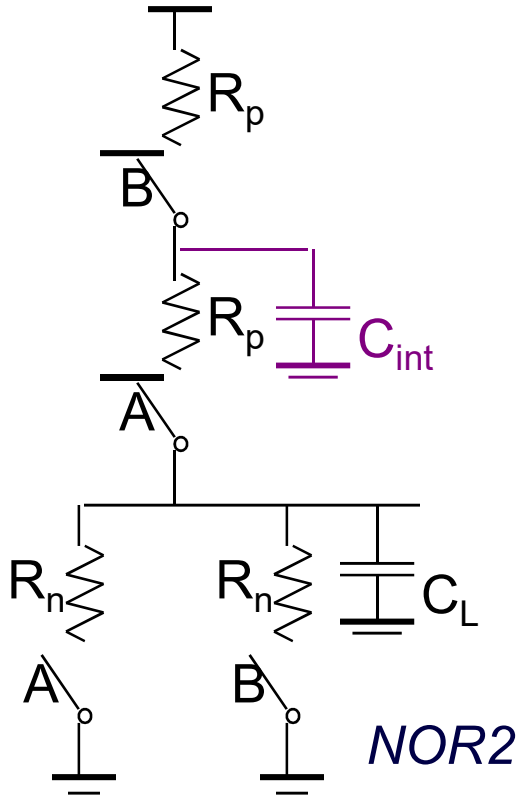




NAND2

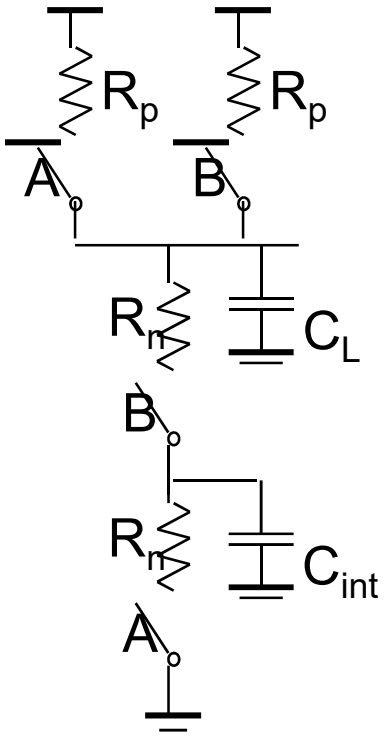


INV



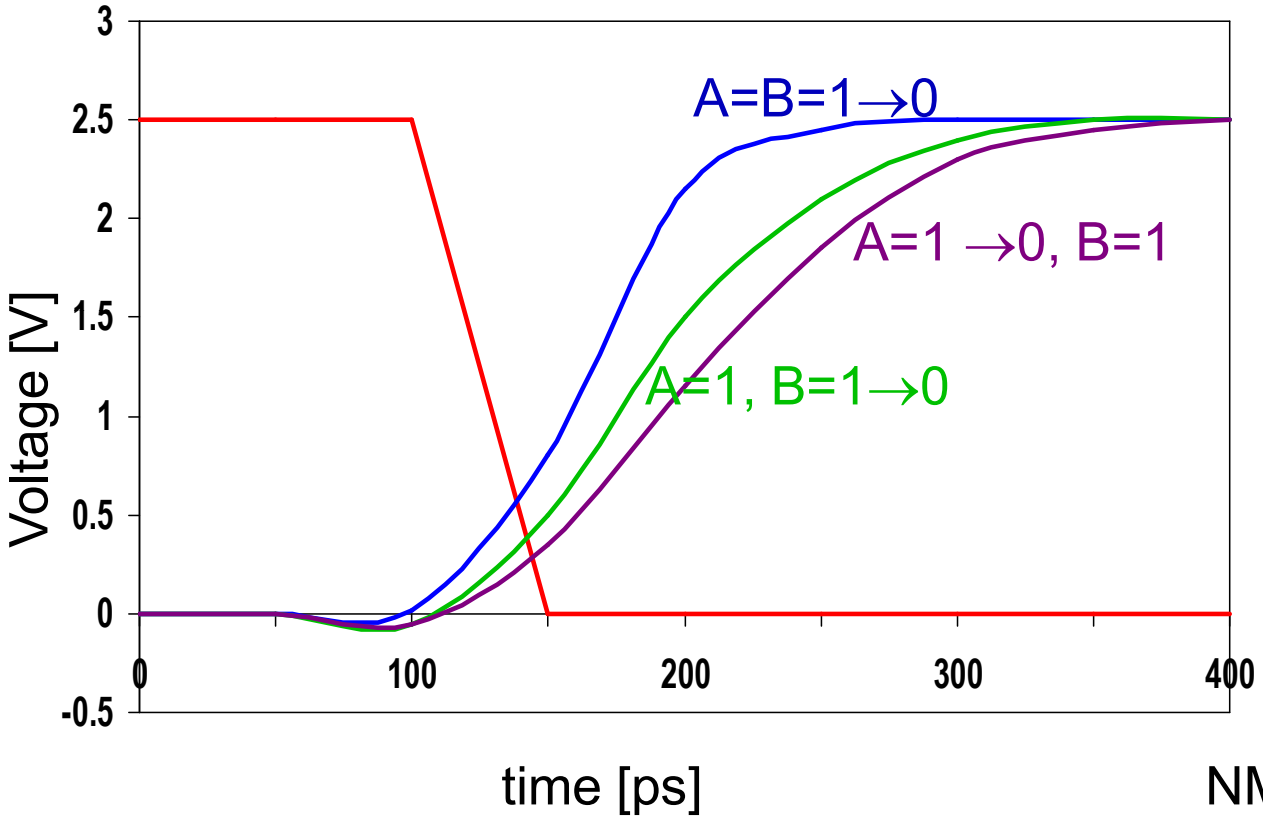
NOR2

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

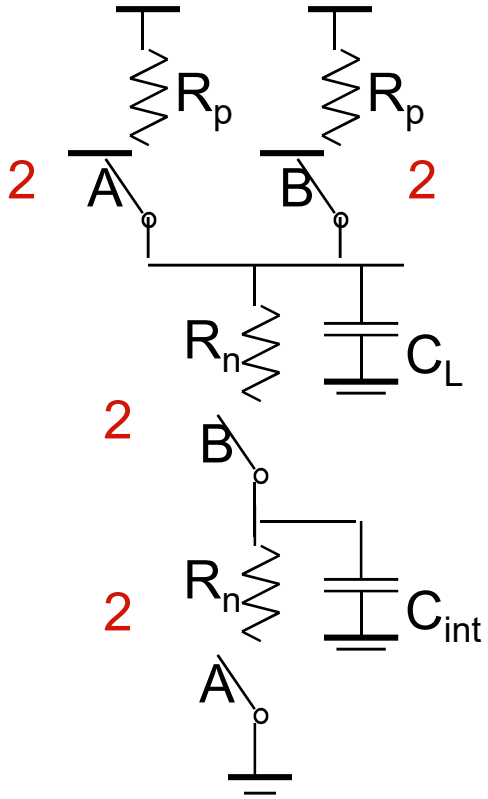
Delay Dependence on Input Patterns

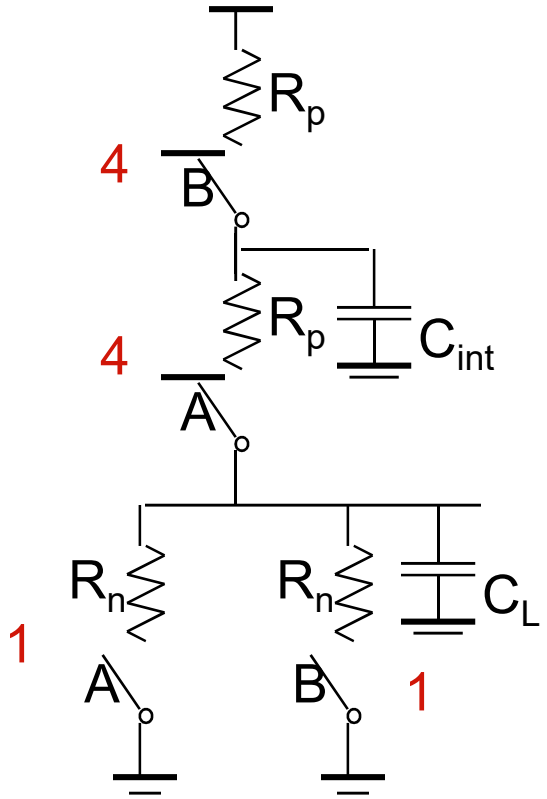


Input Data Pattern	Delay (psec)
A=B=0→1	67
A=1, B=0→1	64
A= 0→1, B=1	61
A=B=1→0	45
A=1, B=1→0	80
A= 1→0, B=1	81

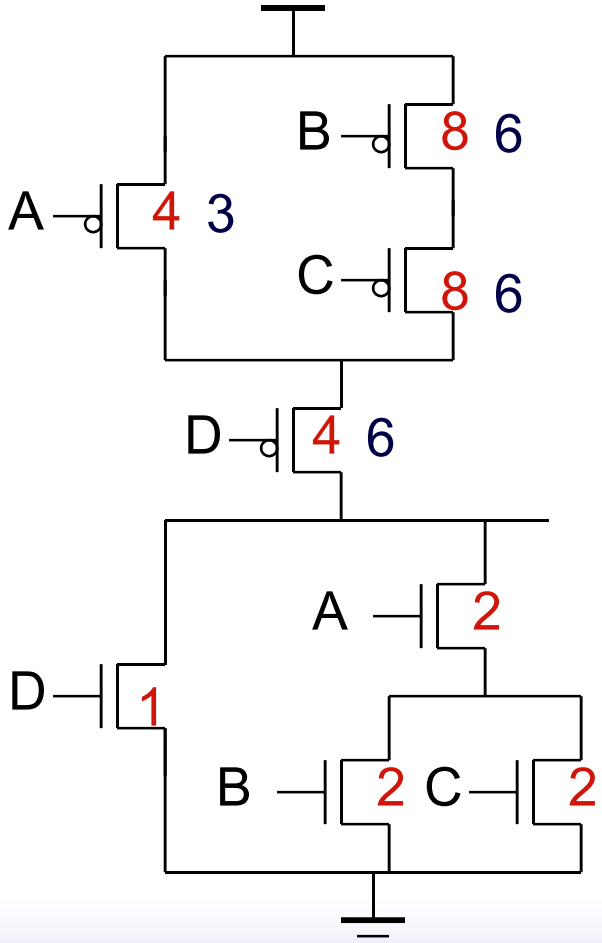
NMOS = 0.5μm/0.25 μm
PMOS = 0.75μm/0.25 μm
C_L = 100 fF

Transistor Sizing



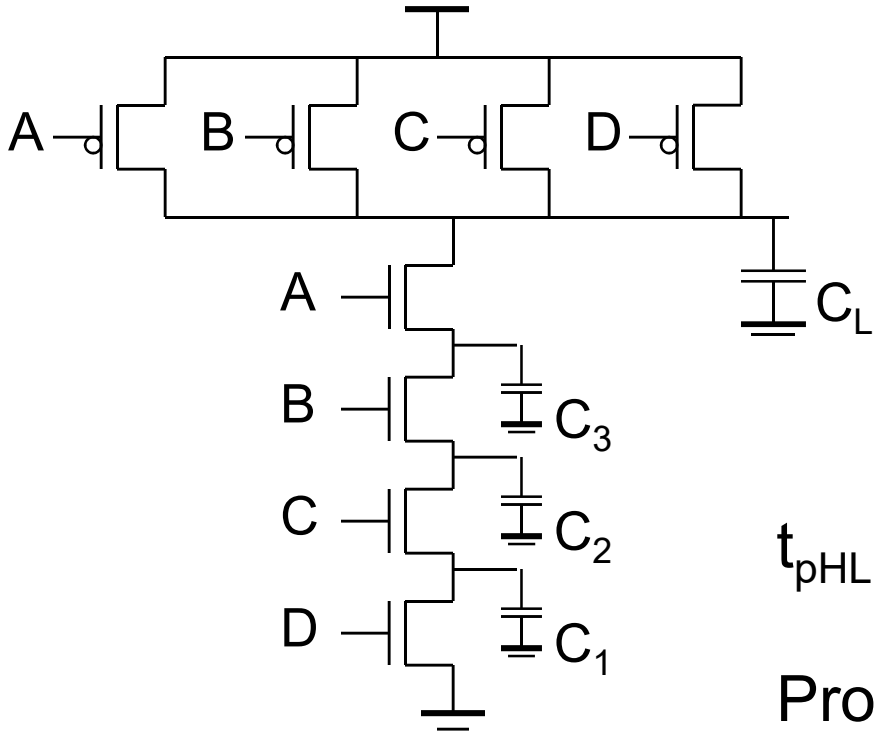


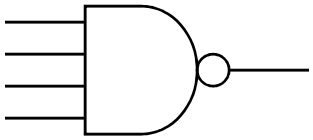
Transistor Sizing a Complex CMOS Gate



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

Fan-In Considerations



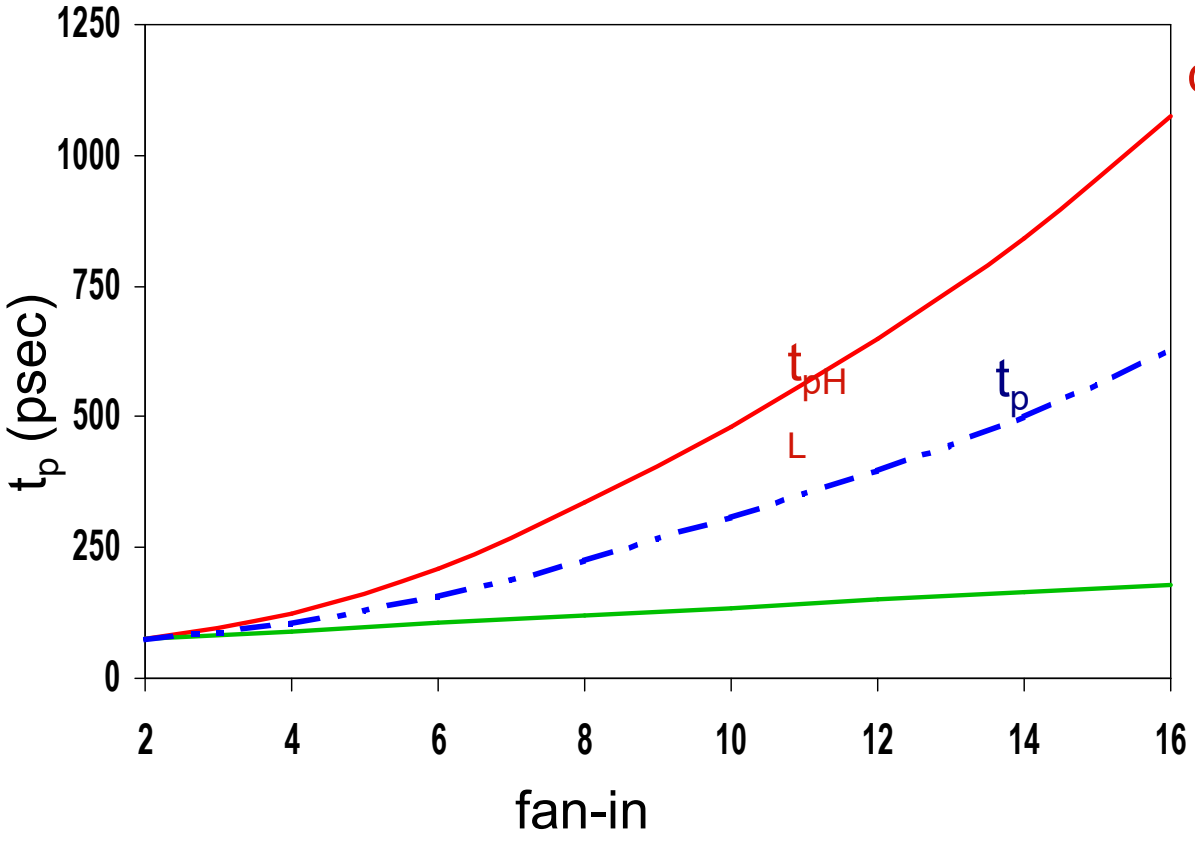


Distributed RC model
(Elmore delay)

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

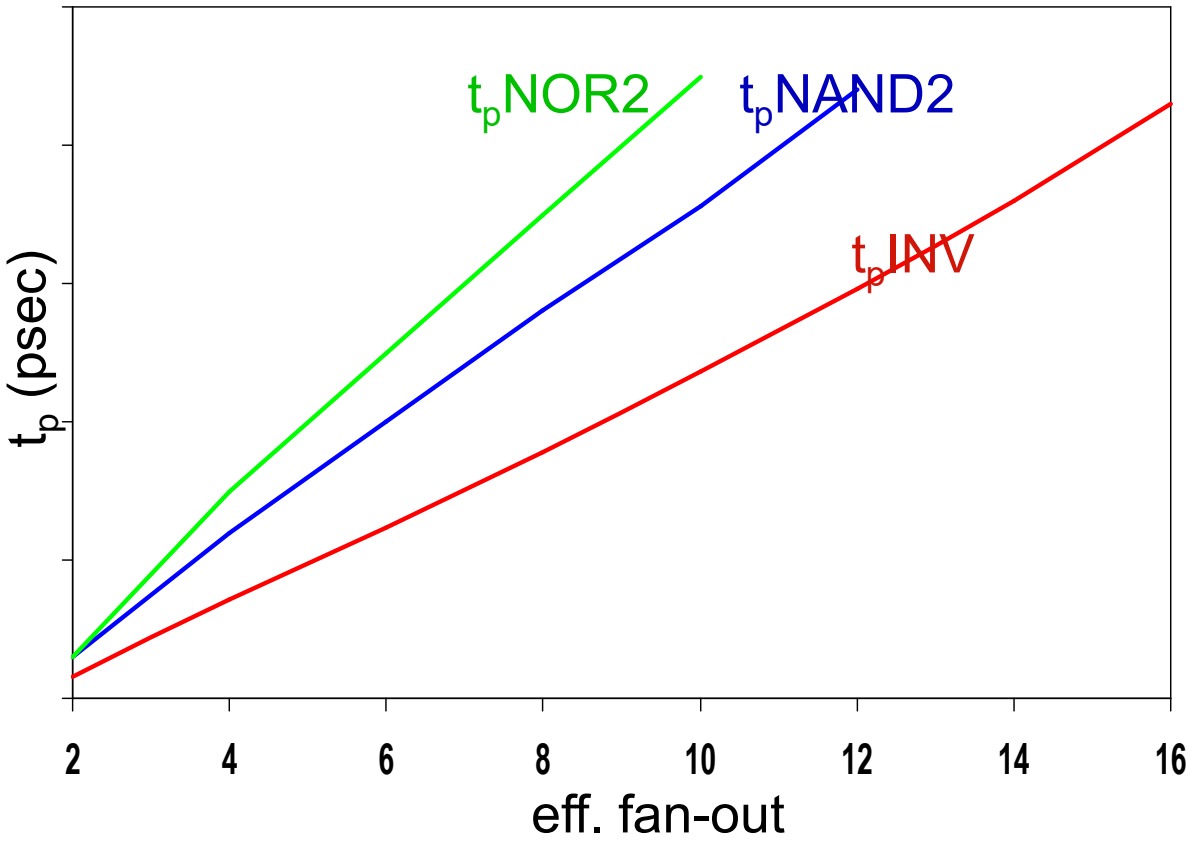
Propagation delay deteriorates rapidly as a function of fan-in – quadratically in the worst case.

t_p as a Function of Fan-In



Gates with a fan-in greater than 4 should be avoided.

t_p as a Function of Fan-Out



All gates have the same drive current.

Slope is a function of "driving strength"

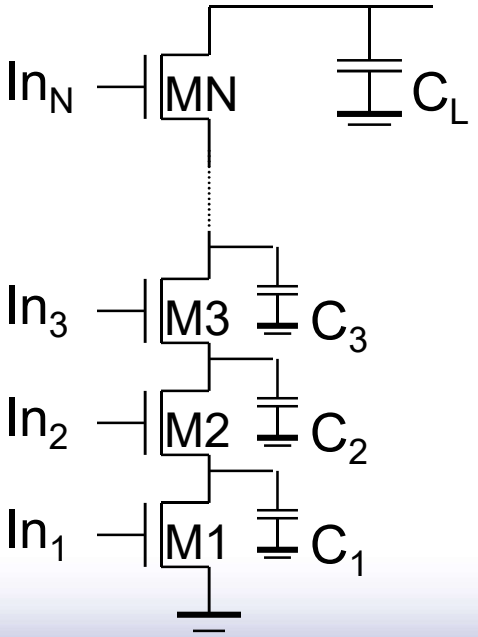
t_p as a Function of Fan-In and Fan-Out

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

$$t_p = a_1 FI + a_2 FI^2 + a_3 FO$$

Fast Complex Gates: Design Technique 1

- Transistor sizing
  - as long as fan-out capacitance dominates
- Progressive sizing



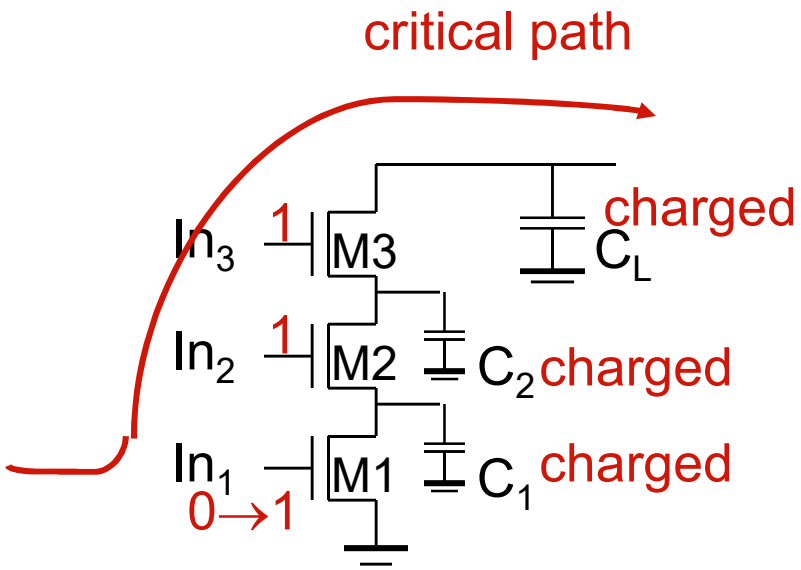
Distributed RC line

M1 > M2 > M3 > … > MN
(the fet closest to the output is the smallest)

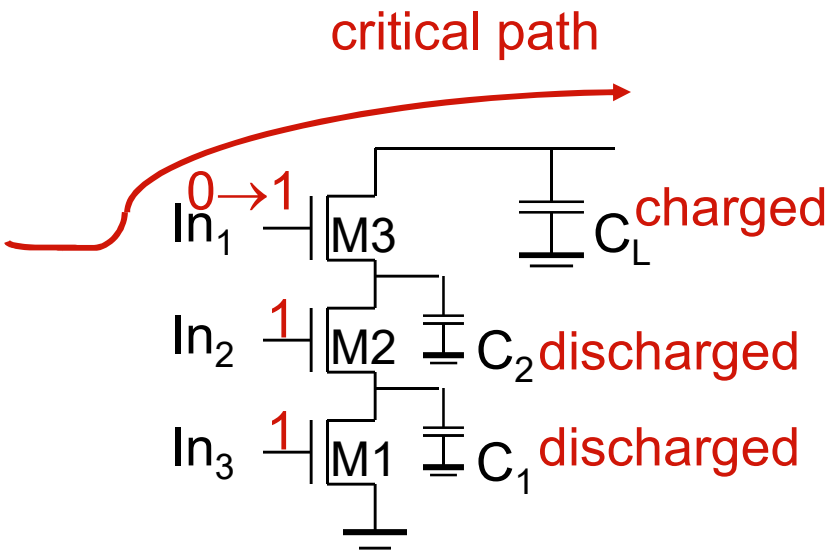
Can reduce delay by more than 20%; decreasing gains as technology shrinks

Fast Complex Gates: Design Technique 2

- Transistor ordering



delay determined by time to discharge C_L, C_1 and C_2

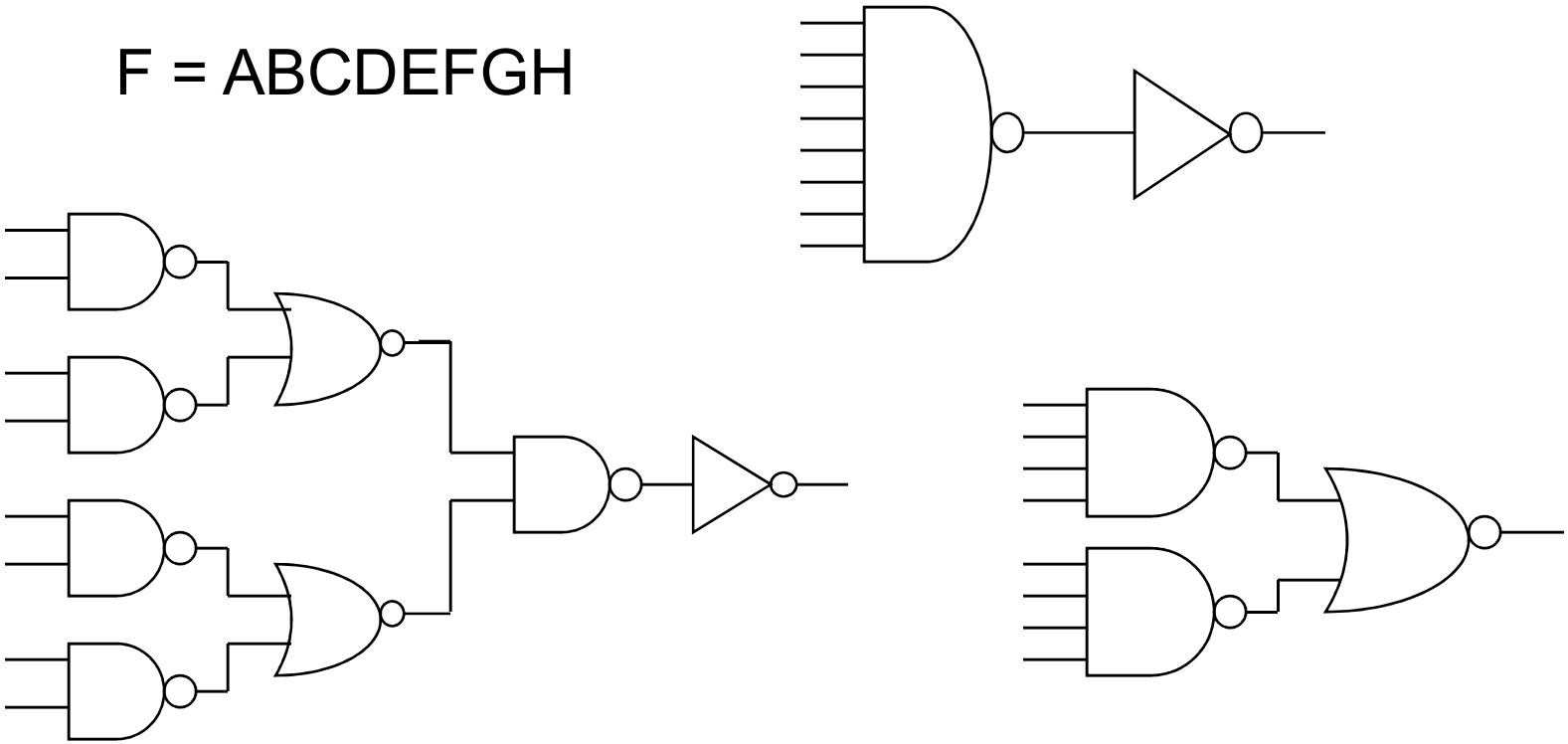


delay determined by time to discharge C_L

Fast Complex Gates: Design Technique 3

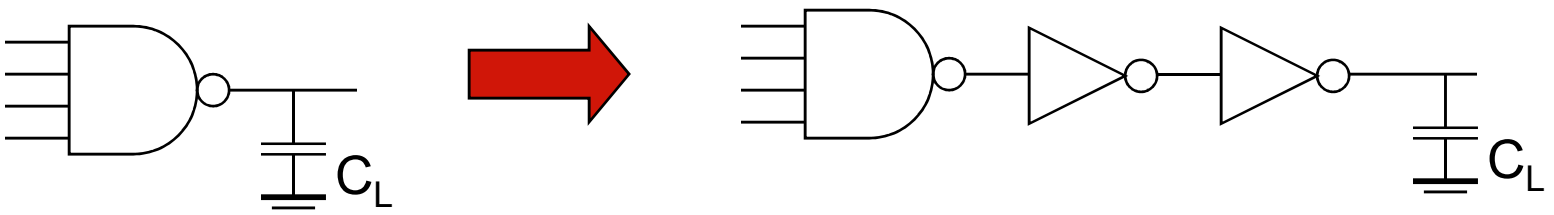
- Alternative logic structures

F = ABCDEFGH



Fast Complex Gates: Design Technique 4

- Isolating fan-in from fan-out using buffer insertion



Fast Complex Gates: Design Technique 5

- Reducing the voltage swing

$$t_{pHL} = 0.69\ (3/4\ (C_L\ V_{DD})/\ I_{DSATn}\ )$$

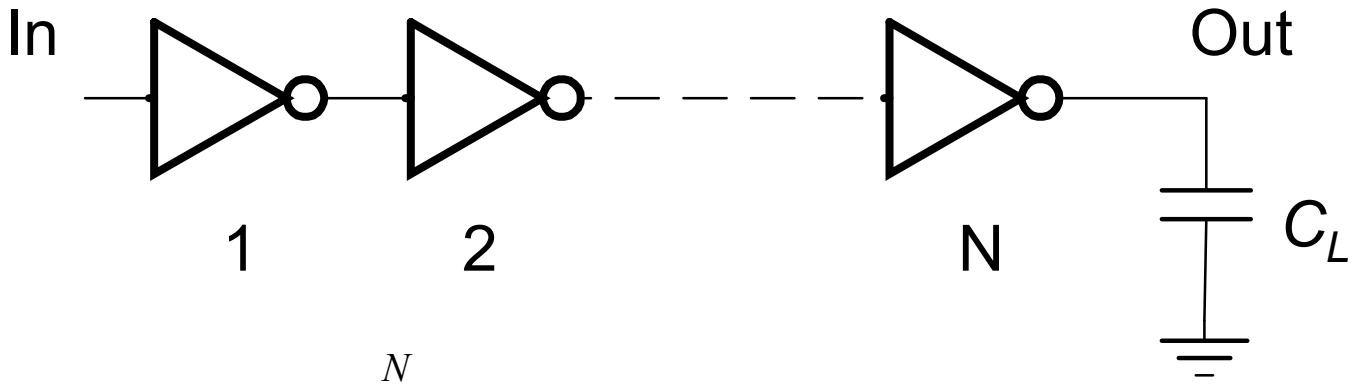
$$= 0.69\ (3/4\ (C_L\ V_{swing})/\ I_{DSATn}\ )$$

  - linear reduction in delay
  - also reduces power consumption
- But the following gate is much slower!
- Or requires use of "sense amplifiers" on the receiving end to restore the signal level (memory design)

Sizing Logic Paths for Speed

- Frequently, input capacitance of a logic path is constrained
- Logic also has to drive some capacitance
- Example: ALU load in an Intel's microprocessor is 0.5pF
- How do we size the ALU datapath to achieve maximum speed?
- We have already solved this for the inverter chain – can we generalize it for any type of logic?

Buffer Example



$$Delay = \sum_{i=1}^{N} (p_i + g_i \cdot f_i)$$

(in units of τ_{inv})

For given N: $C_{i+1}/C_i = C_i/C_{i-1}$
To find N: $C_{i+1}/C_i \sim 4$
How to generalize this to any logic path?

Logical Effort

$$Delay = k \cdot R_{unit} C_{unit} \left(1 + \frac{C_L}{\gamma C_{in}} \right)$$

$$= \tau \left(p + g \cdot f \right)$$

p – intrinsic delay ($3kR_{unit}C_{unit}\gamma$) - gate parameter $\neq$ f(W)
g – logical effort ($kR_{unit}C_{unit}$) – gate parameter $\neq$ f(W)
f – effective fanout

Normalize everything to an inverter:
g_{inv} =1, p_{inv} = 1

Divide everything by τ_{inv}
(everything is measured in unit delays τ_{inv})
Assume γ = 1.

Delay in a Logic Gate

Gate delay:

$$d = h + p$$

effort delay — **intrinsic delay**

Effort delay:

$$h = g\,f$$

logical effort — **effective fanout = C_{out}/C_{in}**

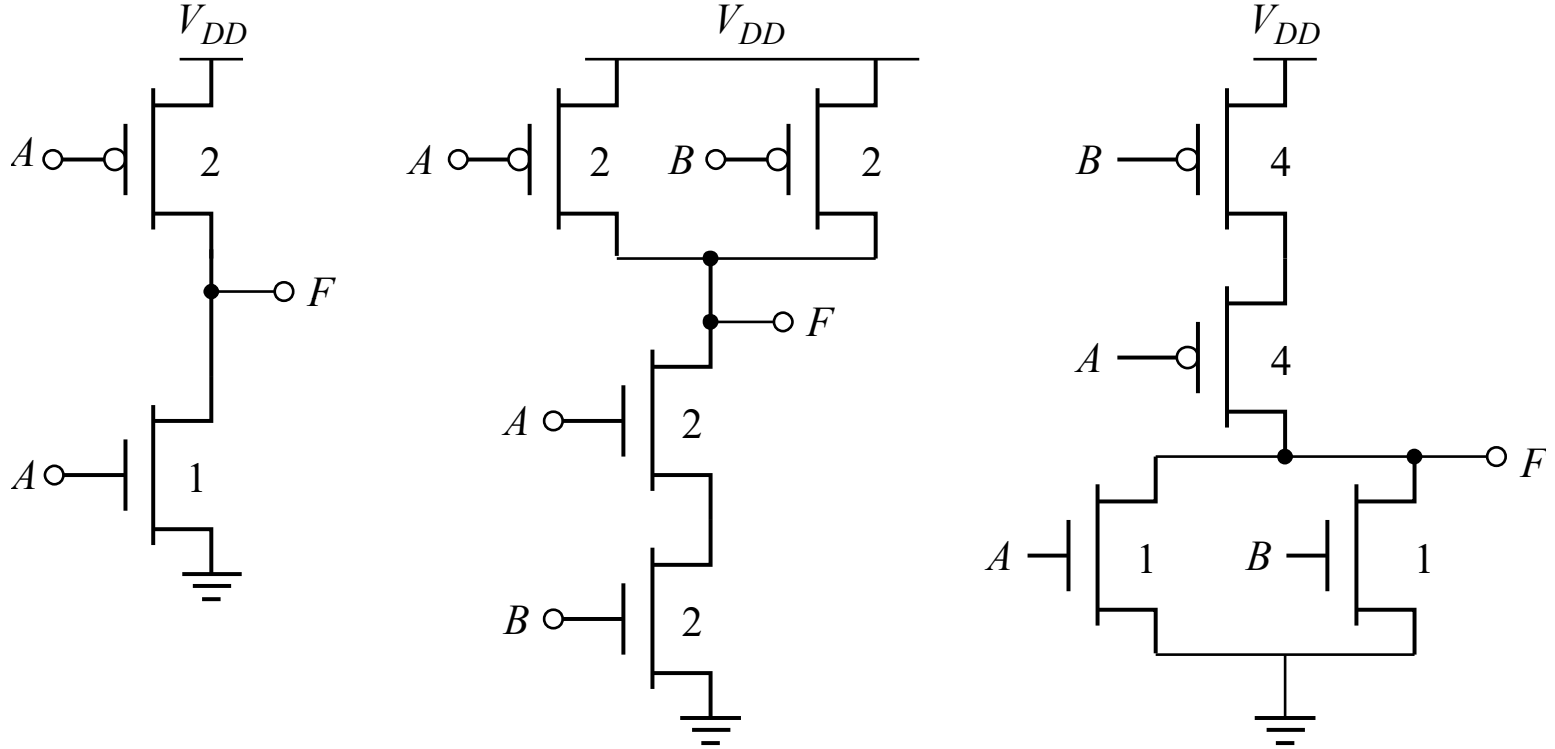
Logical effort is a function of topology, independent of sizing
Effective fanout (electrical effort) is a function of load/gate size

Logical Effort

- Inverter has the smallest logical effort and intrinsic delay of all static CMOS gates
- Logical effort of a gate presents the ratio of its input capacitance to the inverter capacitance when sized to deliver the same current
- Logical effort increases with the gate complexity

Logical Effort

Logical effort is the ratio of input capacitance of a gate to the input capacitance of an inverter with the same output current



Inverter

***g* = 1**

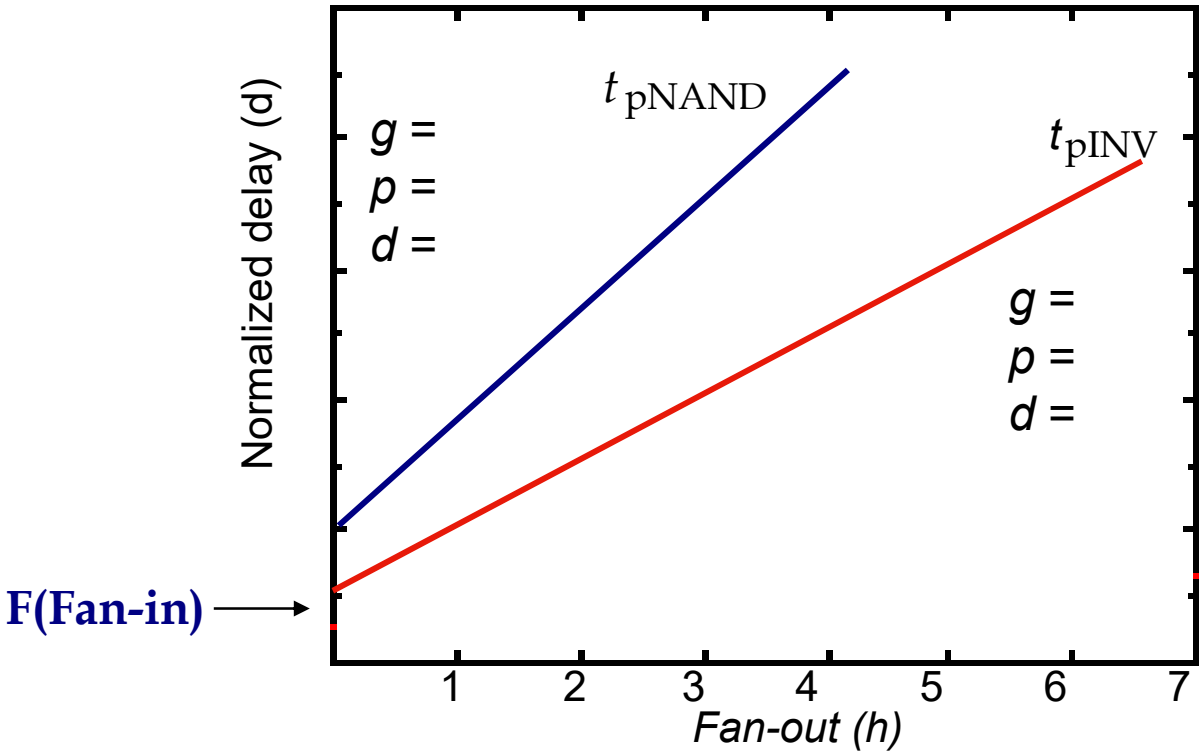
2-input NAND

***g* = 4/3**

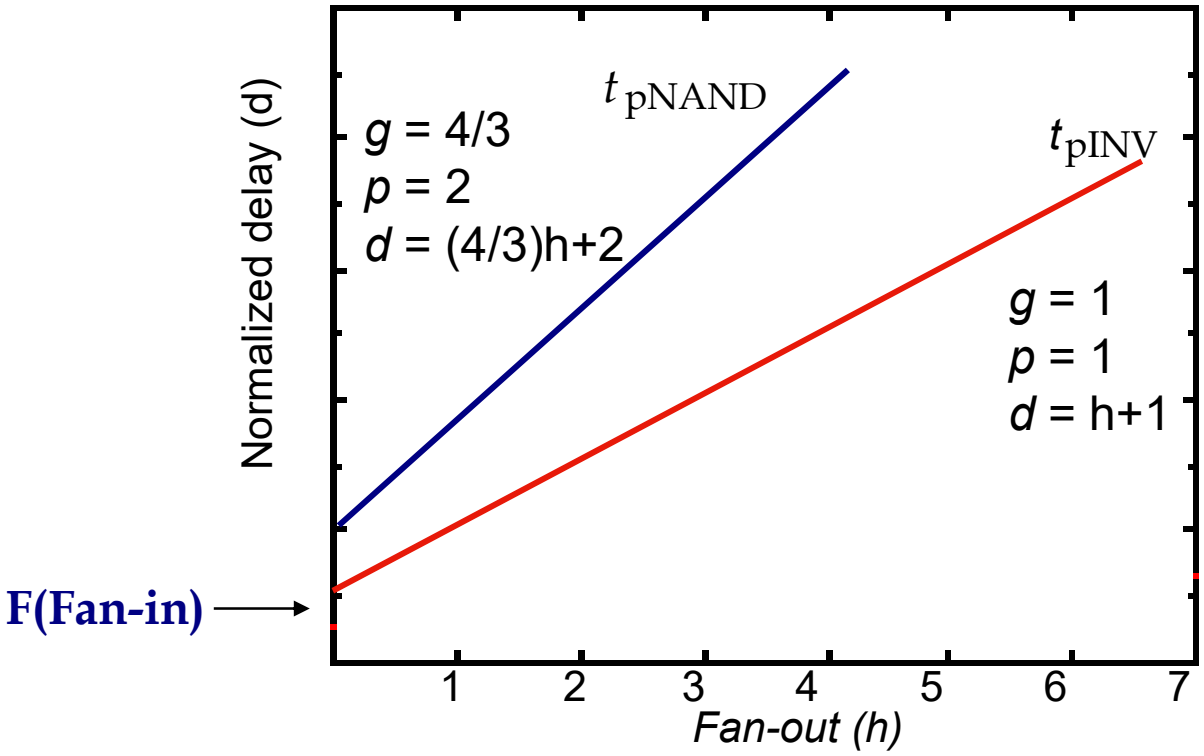
2-input NOR

***g* = 5/3**

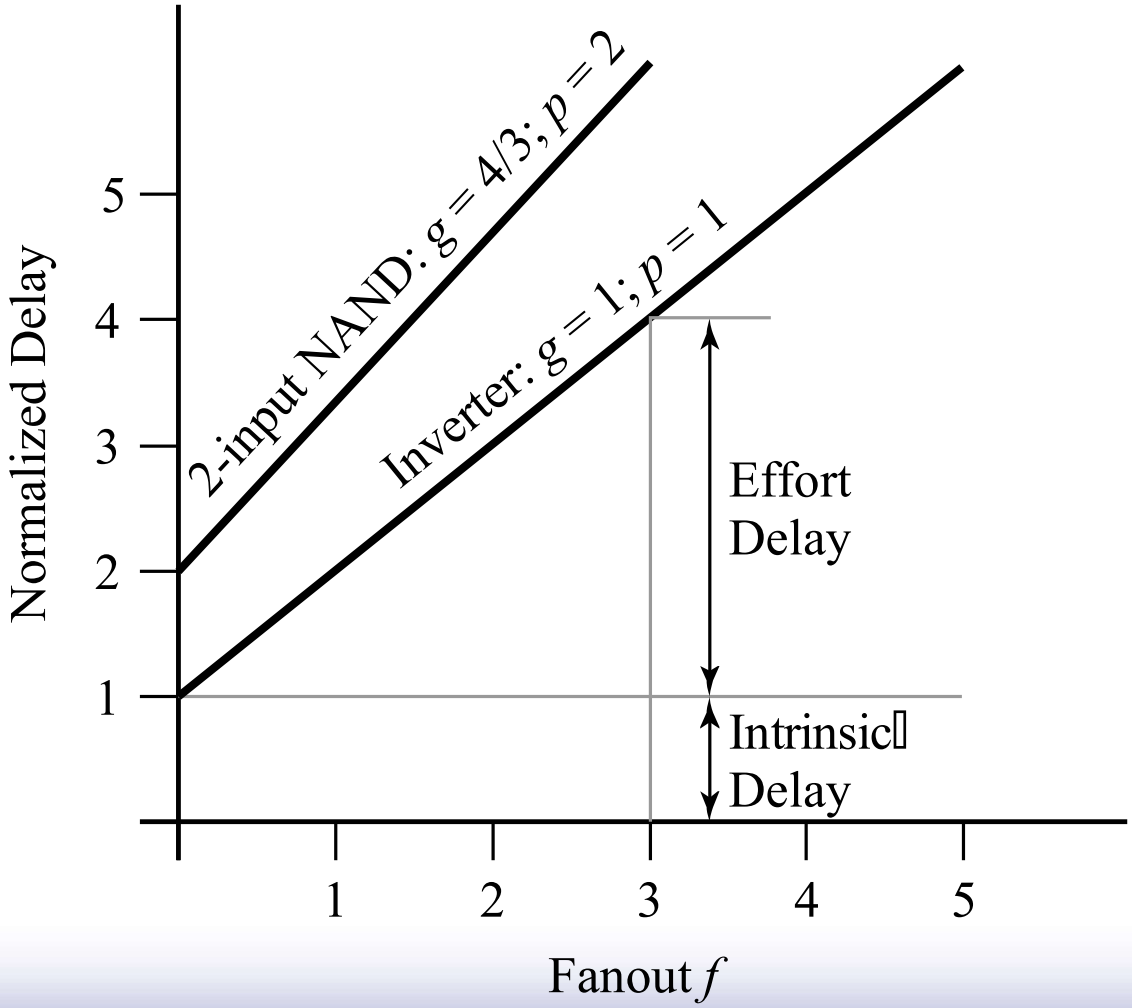
Logical Effort of Gates



Logical Effort of Gates



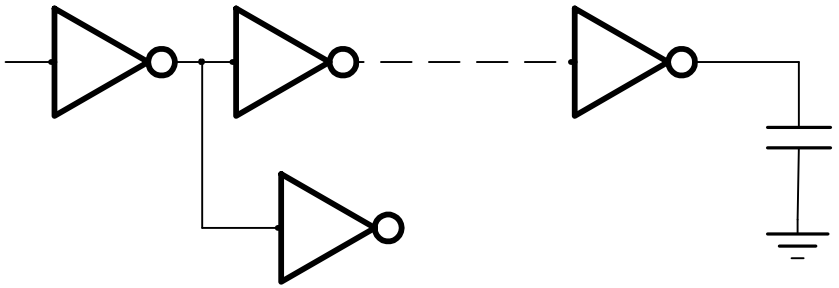
Logical Effort of Gates



Add Branching Effort

Branching effort:

$$b = \frac{C_{on-path} + C_{off-path}}{C_{on-path}}$$



Multistage Networks

$$Delay = \sum_{i=1}^{N} \left(p_i + g_i \cdot f_i \right)$$

Stage effort: $h_i = g_i f_i$

Path electrical effort: $F = C_{out}/C_{in}$

Path logical effort: $G = g_1 g_2 \ldots g_N$

Branching effort: $B = b_1 b_2 \ldots b_N$

Path effort: $H = GFB$

Path delay $D = \Sigma d_i = \Sigma p_i + \Sigma h_i$

Optimum Effort per Stage

When each stage bears the same effort:

$$h^N = H$$

$$h = \sqrt[N]{H}$$

Stage efforts: $g_1 f_1 = g_2 f_2 = \ldots = g_N f_N$

Effective fanout of each stage: $f_i = h / g_i$

Minimum path delay

$$\hat{D} = \sum \left(g_i f_i + p_i \right) = NH^{1/N} + P$$

Optimal Number of Stages

For a given load,
and given input capacitance of the first gate
Find optimal number of stages and optimal sizing

$$D = NH^{1/N} + Np_{inv}$$

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

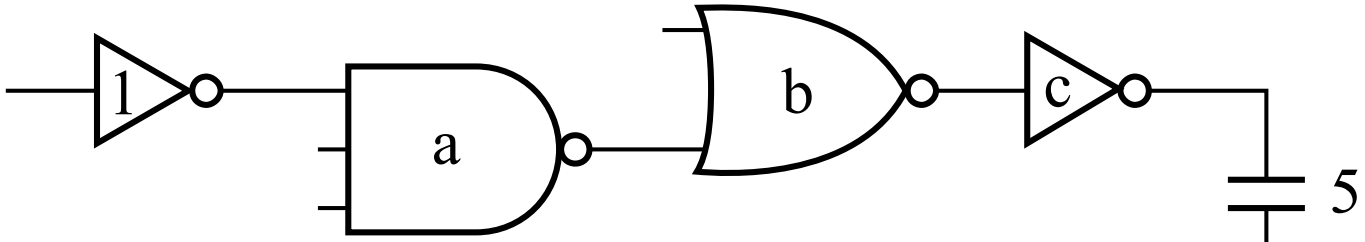
Substitute 'best stage effort' $h = H^{1/\hat{N}}$

Logical Effort

Gate Type	Number of Inputs			
	1	2	3	n
Inverter	1			
NAND		4/3	5/3	(n + 2)/3
NOR		5/3	7/3	(2n + 1)/3
Multiplexer		2	2	2
XOR		4	12	

From Sutherland, Sproull

Example: Optimize Path



$g = 1$	$g = 5/3$	$g = 5/3$	$g = 1$
$f = a$	$f = b/a$	$f = c/b$	$f = 5/c$

Effective fanout, F =

G =

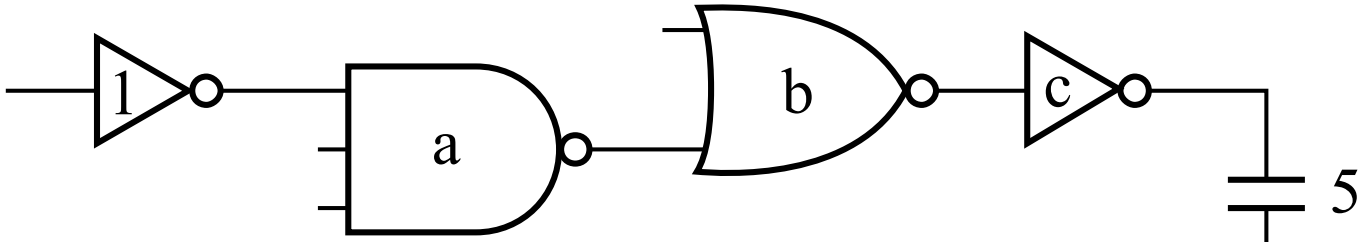
H =

h =

a =

b =

Example: Optimize Path



$g = 1$	$g = 5/3$	$g = 5/3$	$g = 1$
$f = a$	$f = b/a$	$f = c/b$	$f = 5/c$

Effective fanout, F = 5

$G = 25/9$

$H = 125/9 = 13.9$

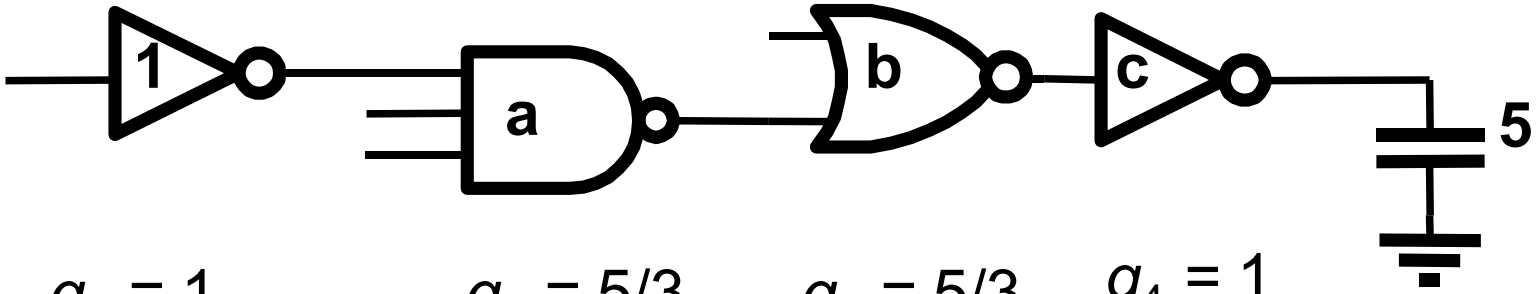
$h = 1.93$

$a = 1.93$

$b = ha/g_2 = 2.23$

$c = hb/g_3 = 5g_4/f = 2.59$

Example: Optimize Path



$g_1 = 1$ $g_2 = 5/3$ $g_3 = 5/3$ $g_4 = 1$

Effective fanout, H = 5

$G = 25/9$

$F = 125/9 = 13.9$

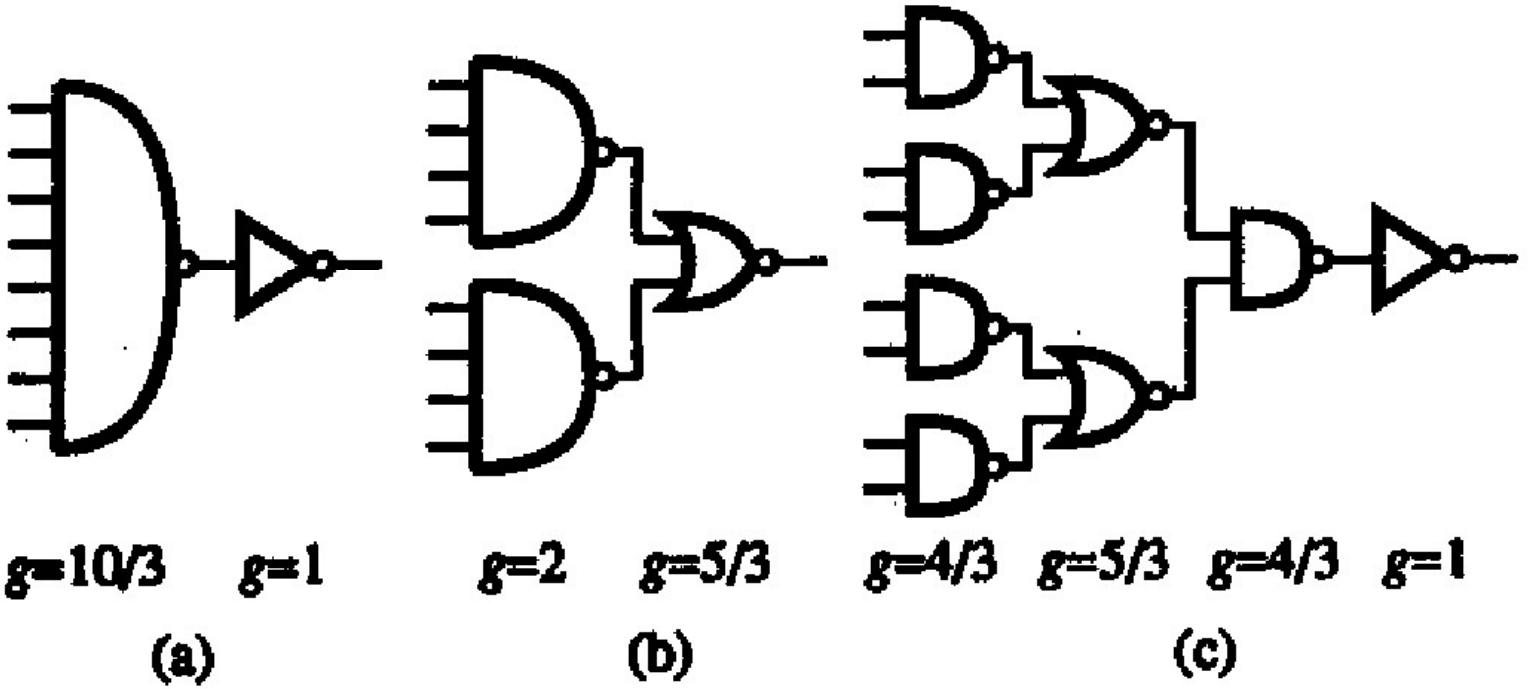
$f = 1.93$

$a = 1.93$

$b = fa/g_2 = 2.23$

$c = fb/g_3 = 5g_4/f = 2.59$

Example – 8-input AND



Method of Logical Effort

- Compute the path effort: $F = GBH$
- Find the best number of stages $N \sim \log_4 F$
- Compute the stage effort $f = F^{1/N}$
- Sketch the path with this number of stages
- Work either from either end, find sizes:
  $C_{in} = C_{out} * g/f$

Reference: Sutherland, Sproull, Harris, "Logical Effort, Morgan-Kaufmann 1999.

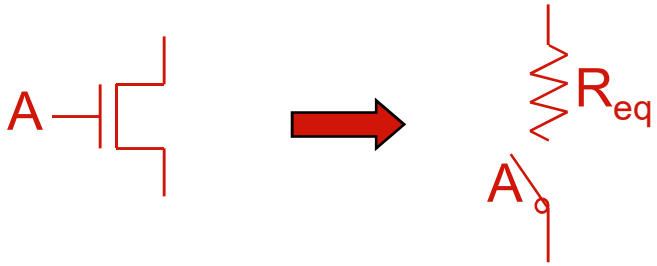
Summary

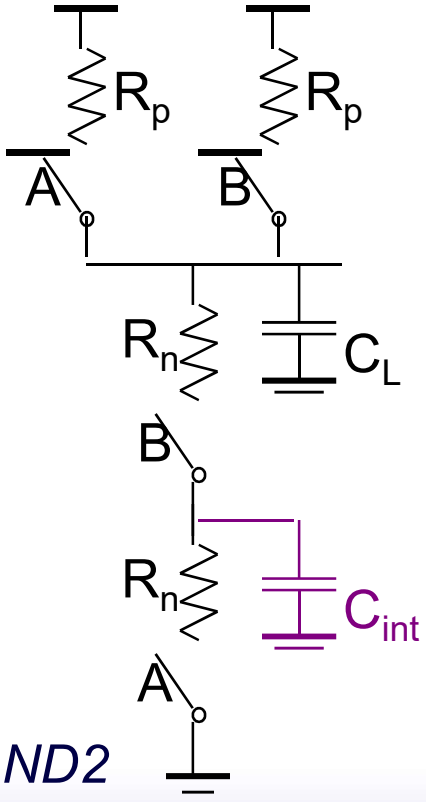
Table 4: Key Definitions of Logical Effort

Term	Stage expression	Path expression
Logical effort	g (seeTable 1)	$G = \prod g_i$
Electrical effort	$h = \frac{C_{out}}{C_{in}}$	$H = \frac{C_{out\ (path)}}{C_{in\ (path)}}$
Branching effort	n/a	$B = \prod b_i$
Effort	$f = gh$	$F = GBH$
Effort delay	f	$D_F = \sum f_i$
Number of stages	1	N
Parasitic delay	p (seeTable 2)	$P = \sum p_i$
Delay	$d = f + p$	$D = D_F + P$

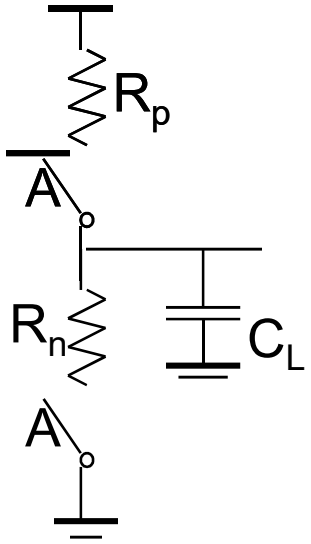
Sutherland,
Sproull
Harris

Switch Delay Model

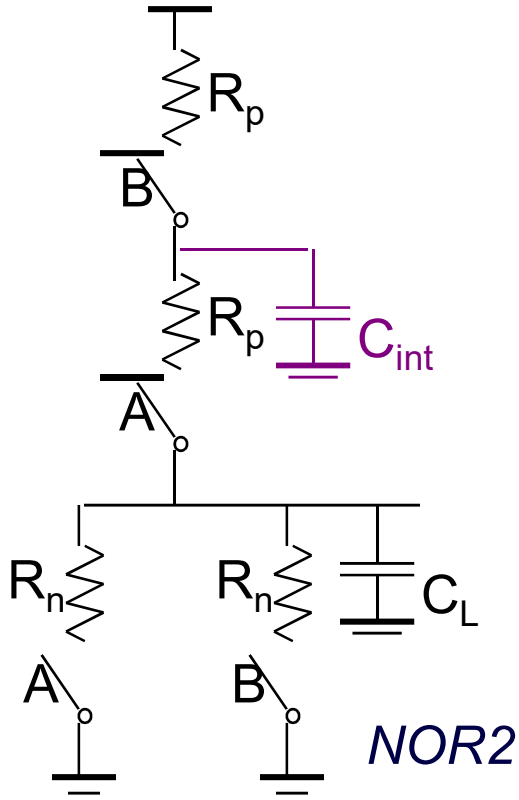




NAND2

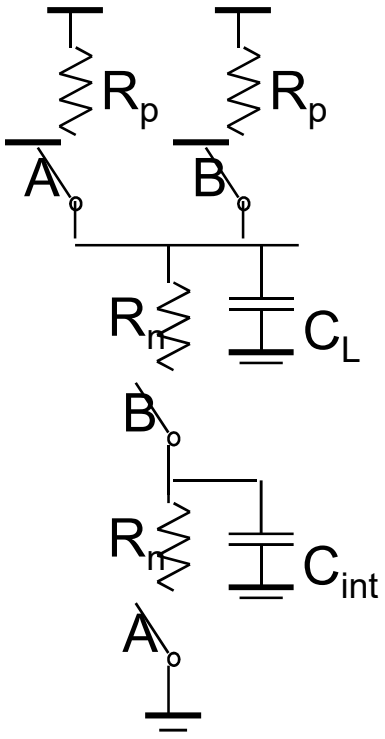


INV



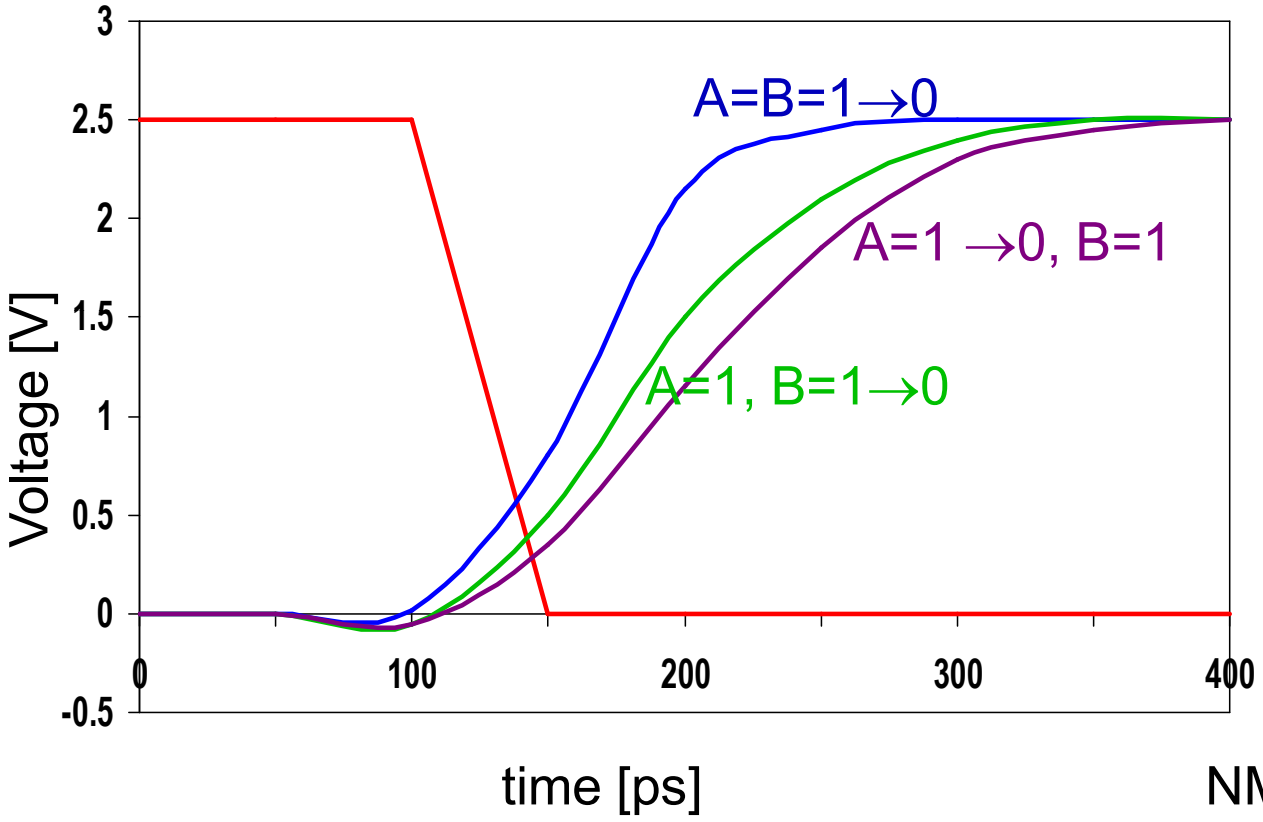
NOR2

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

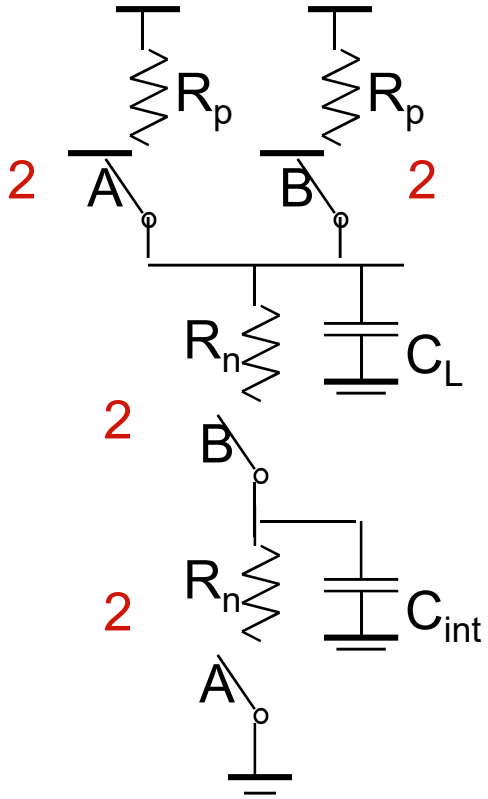
Delay Dependence on Input Patterns

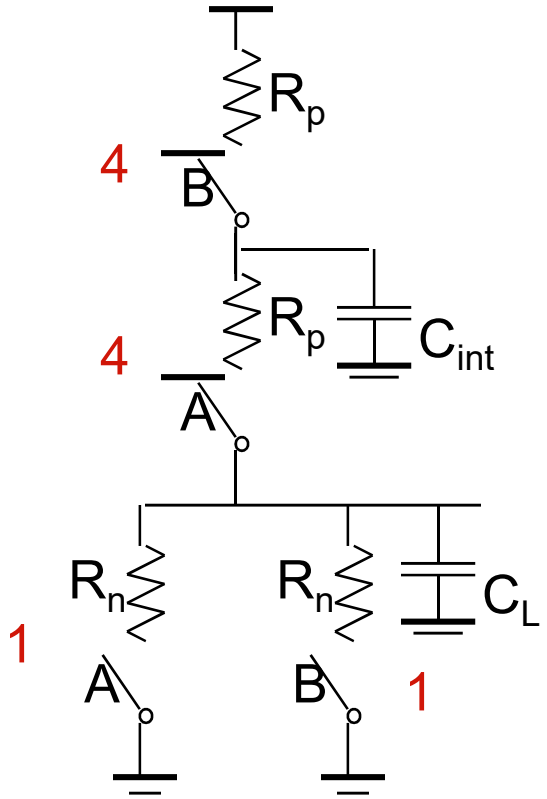


Input Data Pattern	Delay (psec)
A=B=0→1	67
A=1, B=0→1	64
A= 0→1, B=1	61
A=B=1→0	45
A=1, B=1→0	80
A= 1→0, B=1	81

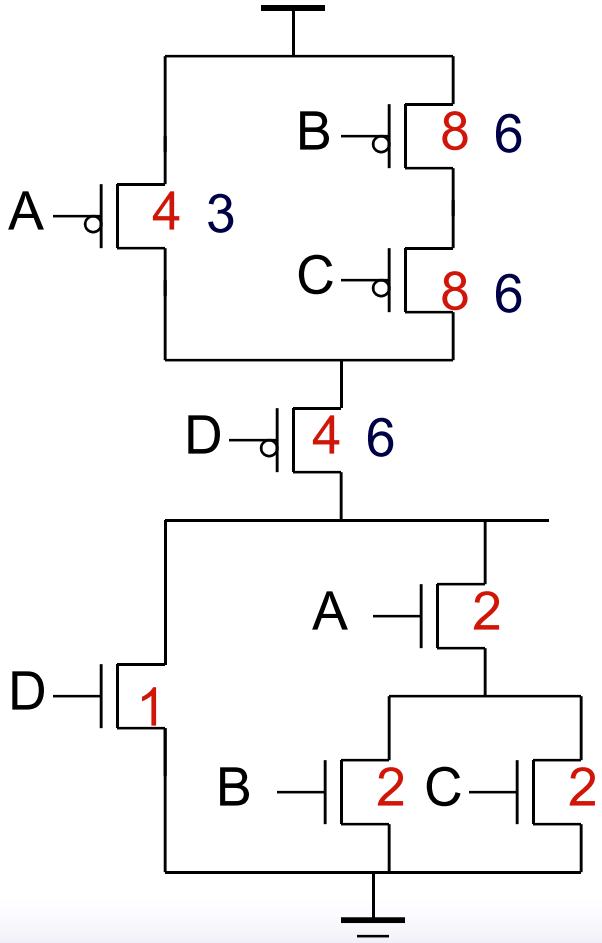
NMOS = 0.5μm/0.25 μm
PMOS = 0.75μm/0.25 μm
C_L = 100 fF

Transistor Sizing



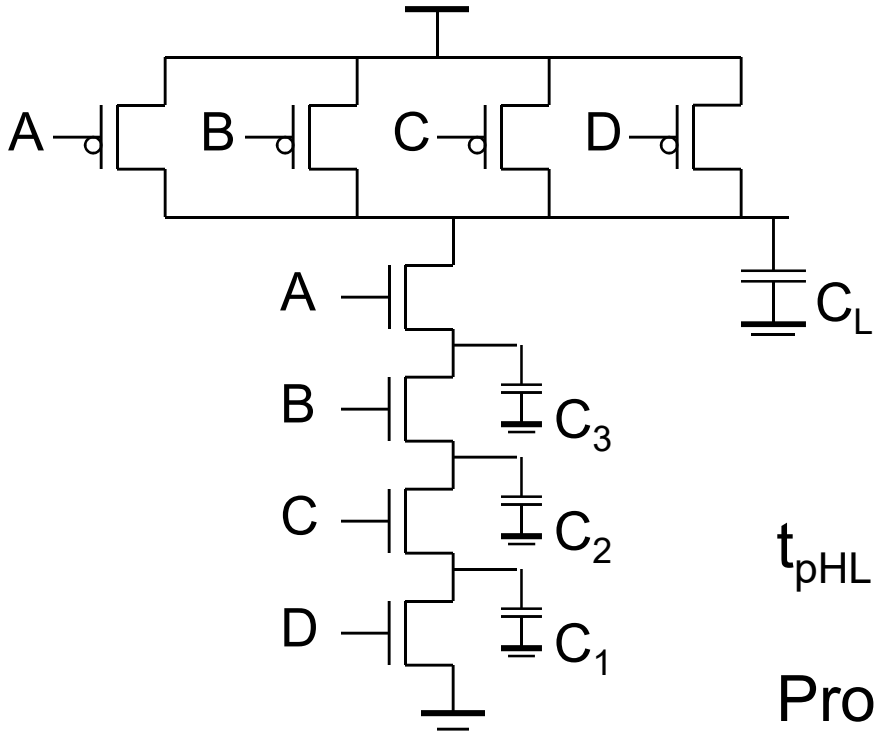


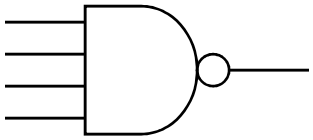
Transistor Sizing a Complex CMOS Gate



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

Fan-In Considerations



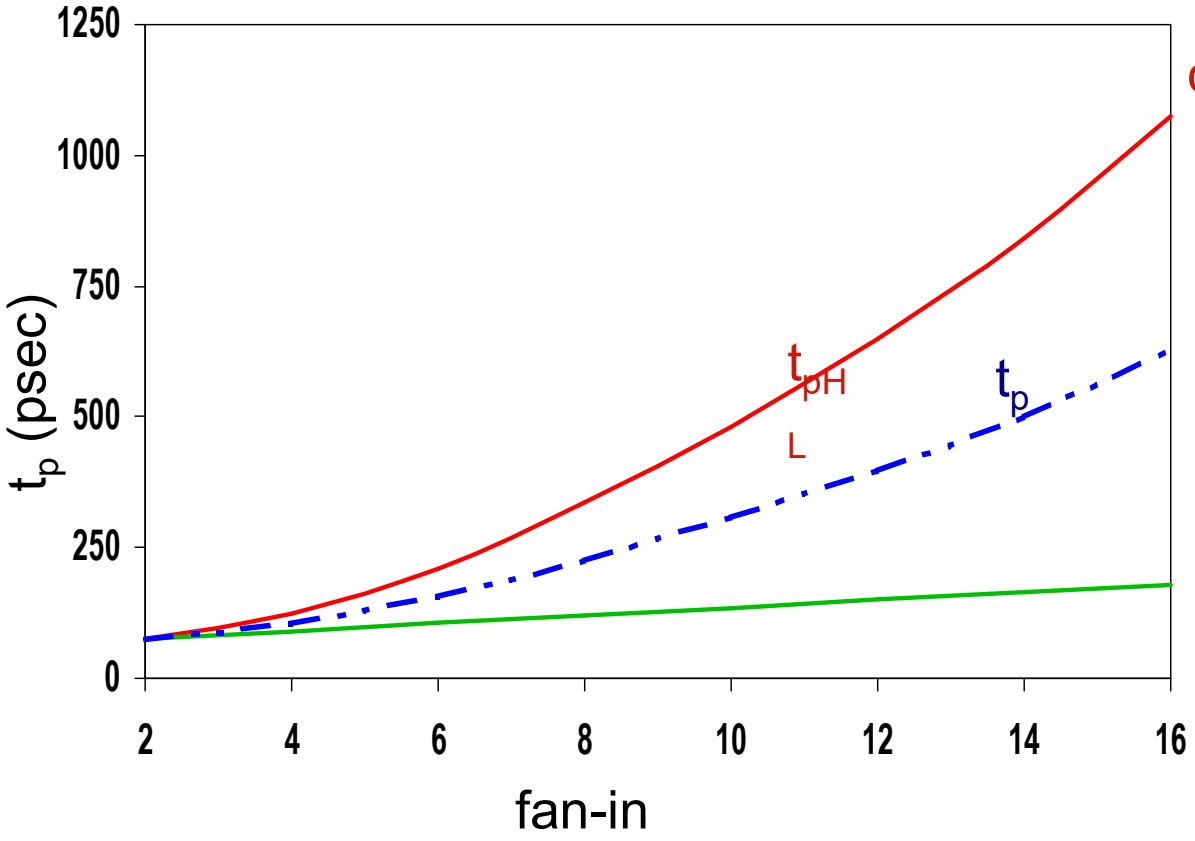


Distributed RC model
(Elmore delay)

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

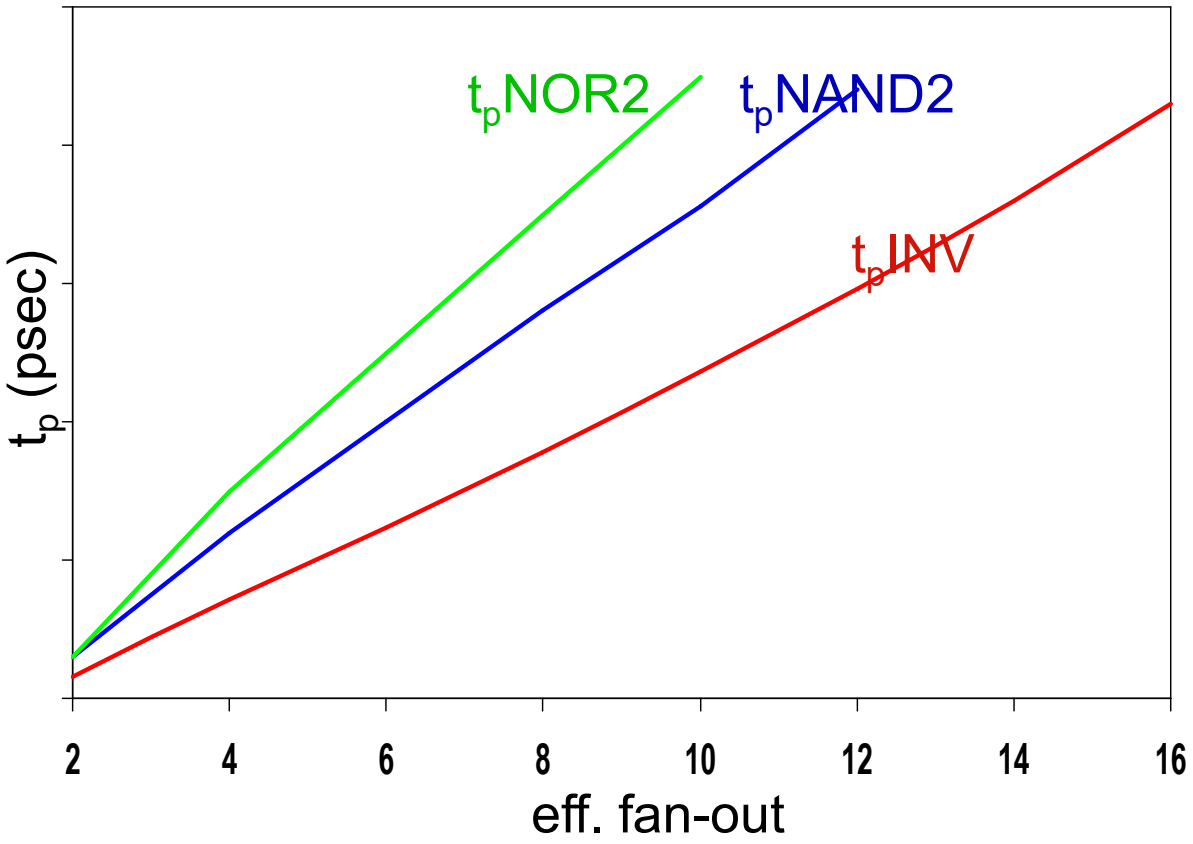
Propagation delay deteriorates rapidly as a function of fan-in – quadratically in the worst case.

t_p as a Function of Fan-In



Gates with a fan-in greater than 4 should be avoided.

t_p as a Function of Fan-Out



All gates have the same drive current.

Slope is a function of "driving strength"

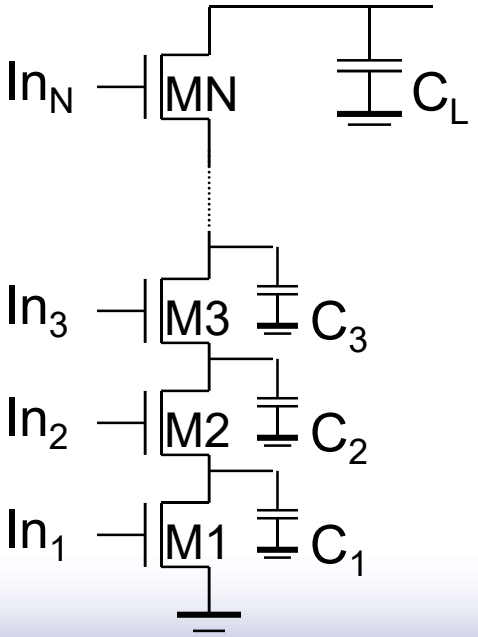
t_p as a Function of Fan-In and Fan-Out

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

$$t_p = a_1 FI + a_2 FI^2 + a_3 FO$$

Fast Complex Gates: Design Technique 1

- Transistor sizing
  - as long as fan-out capacitance dominates
- Progressive sizing



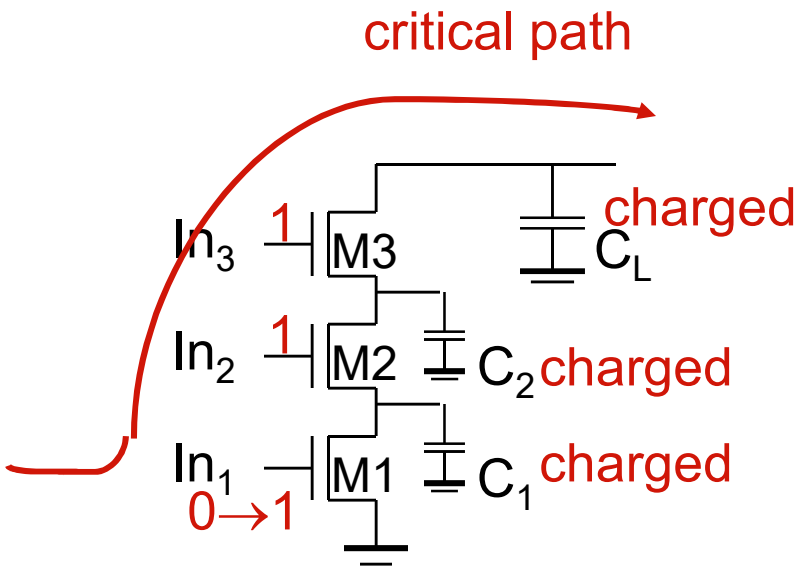
Distributed RC line

M1 > M2 > M3 > … > MN
(the fet closest to the output is the smallest)

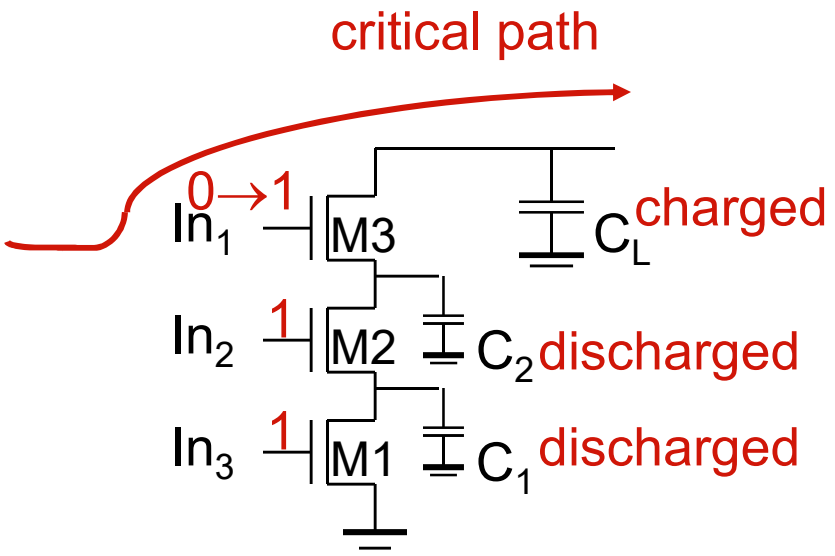
Can reduce delay by more than 20%; decreasing gains as technology shrinks

Fast Complex Gates: Design Technique 2

- Transistor ordering



delay determined by time to discharge C_L, C_1 and C_2

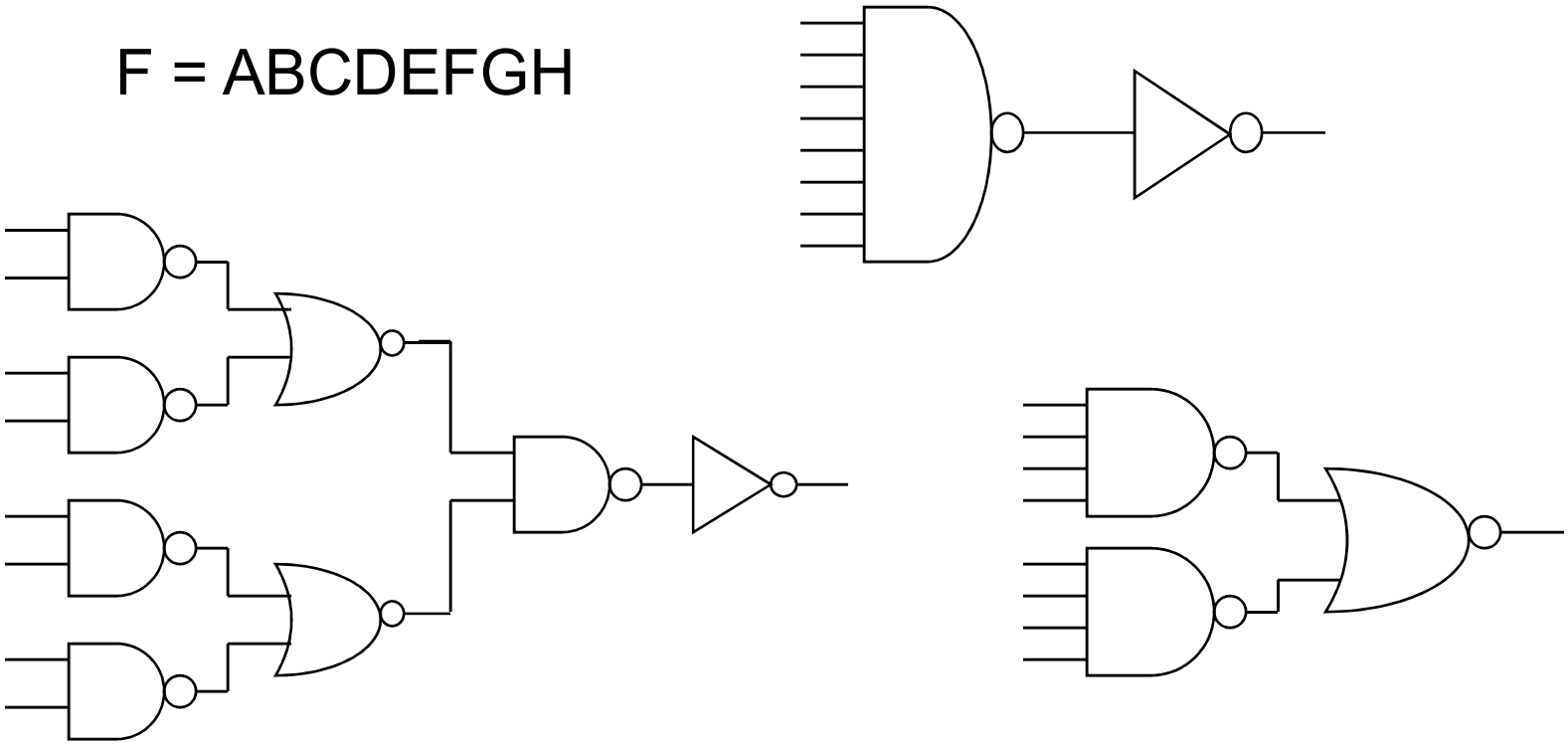


delay determined by time to discharge C_L

Fast Complex Gates: Design Technique 3

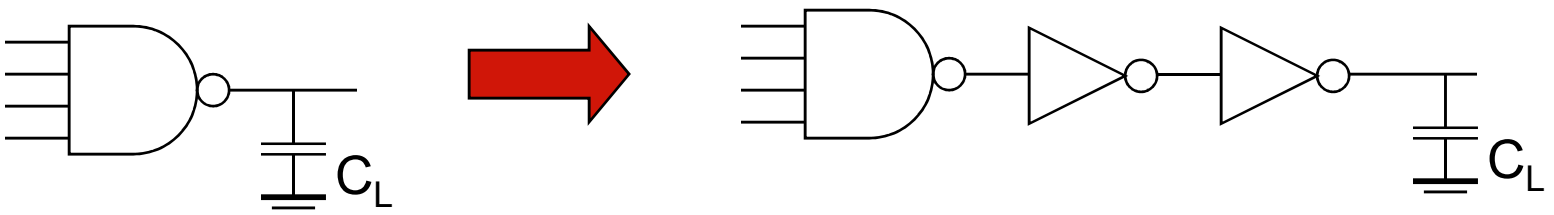
- Alternative logic structures

F = ABCDEFGH



Fast Complex Gates: Design Technique 4

- Isolating fan-in from fan-out using buffer insertion



Fast Complex Gates: Design Technique 5

- Reducing the voltage swing

$$t_{pHL} = 0.69\ (3/4\ (C_L\ V_{DD})/\ I_{DSATn}\ )$$

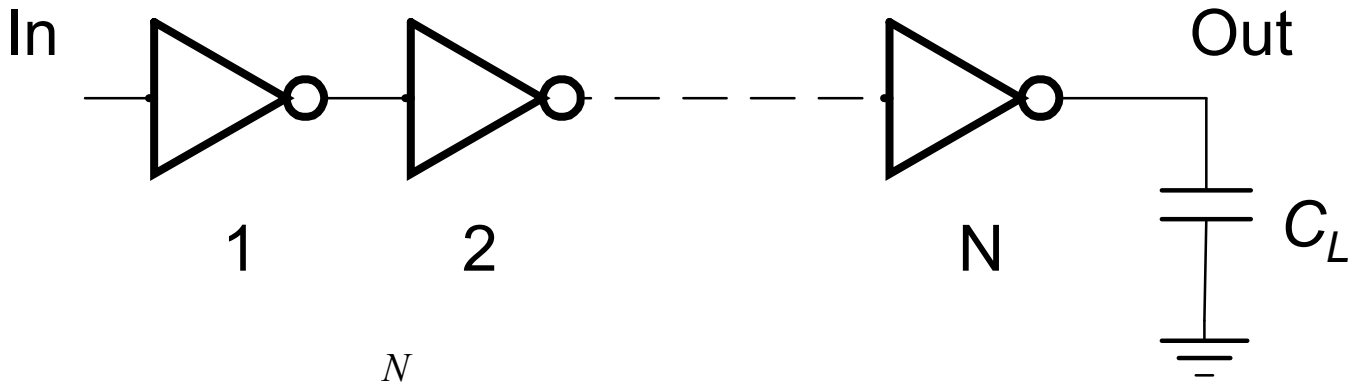
$$= 0.69\ (3/4\ (C_L\ V_{swing})/\ I_{DSATn}\ )$$

  - linear reduction in delay
  - also reduces power consumption
- But the following gate is much slower!
- Or requires use of "sense amplifiers" on the receiving end to restore the signal level (memory design)

Sizing Logic Paths for Speed

- Frequently, input capacitance of a logic path is constrained
- Logic also has to drive some capacitance
- Example: ALU load in an Intel's microprocessor is 0.5pF
- How do we size the ALU datapath to achieve maximum speed?
- We have already solved this for the inverter chain – can we generalize it for any type of logic?

Buffer Example



$$Delay = \sum_{i=1}^{N} (p_i + g_i \cdot f_i)$$

(in units of τ_{inv})

For given N: $C_{i+1}/C_i = C_i/C_{i-1}$
To find N: $C_{i+1}/C_i \sim 4$
How to generalize this to any logic path?

Logical Effort

$$Delay = k \cdot R_{unit} C_{unit} \left(1 + \frac{C_L}{\gamma C_{in}} \right)$$

$$= \tau \left(p + g \cdot f \right)$$

p – intrinsic delay ($3kR_{unit}C_{unit}\gamma$) - gate parameter $\neq$ f(W)
g – logical effort ($kR_{unit}C_{unit}$) – gate parameter $\neq$ f(W)
f – effective fanout

Normalize everything to an inverter:
g_{inv} =1, p_{inv} = 1

Divide everything by τ_{inv}
(everything is measured in unit delays τ_{inv})
Assume γ = 1.

Delay in a Logic Gate

Gate delay:

$$d = h + p$$

effort delay — **intrinsic delay**

Effort delay:

$$h = g\,f$$

logical effort — **effective fanout = C_{out}/C_{in}**

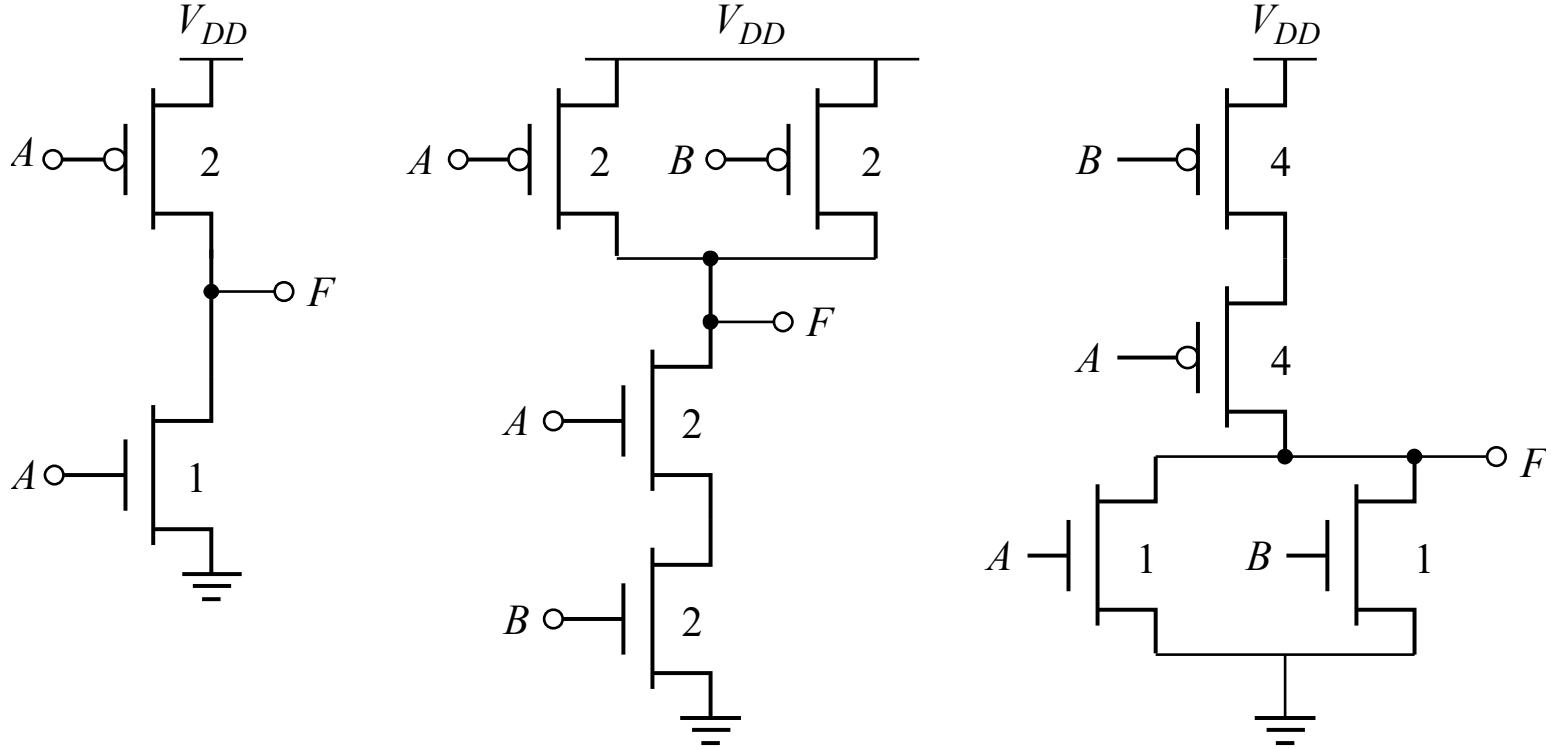
Logical effort is a function of topology, independent of sizing
Effective fanout (electrical effort) is a function of load/gate size

Logical Effort

- Inverter has the smallest logical effort and intrinsic delay of all static CMOS gates
- Logical effort of a gate presents the ratio of its input capacitance to the inverter capacitance when sized to deliver the same current
- Logical effort increases with the gate complexity

Logical Effort

Logical effort is the ratio of input capacitance of a gate to the input capacitance of an inverter with the same output current



Inverter

***g* = 1**

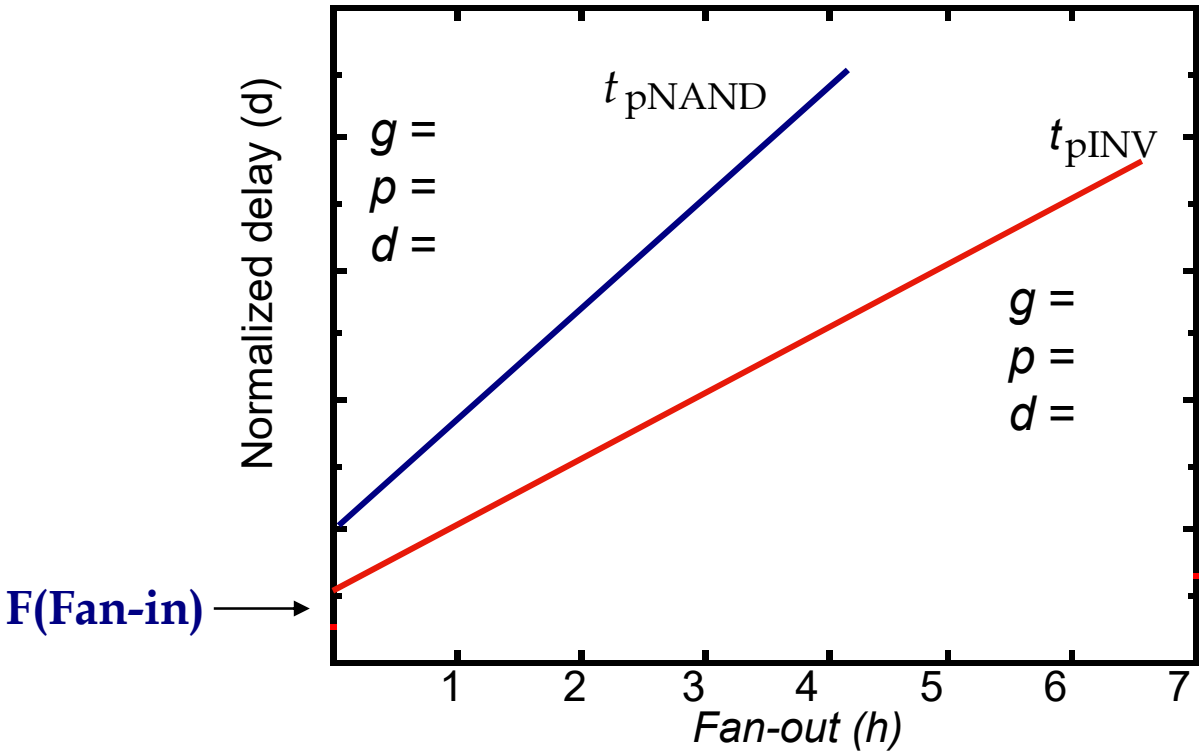
2-input NAND

***g* = 4/3**

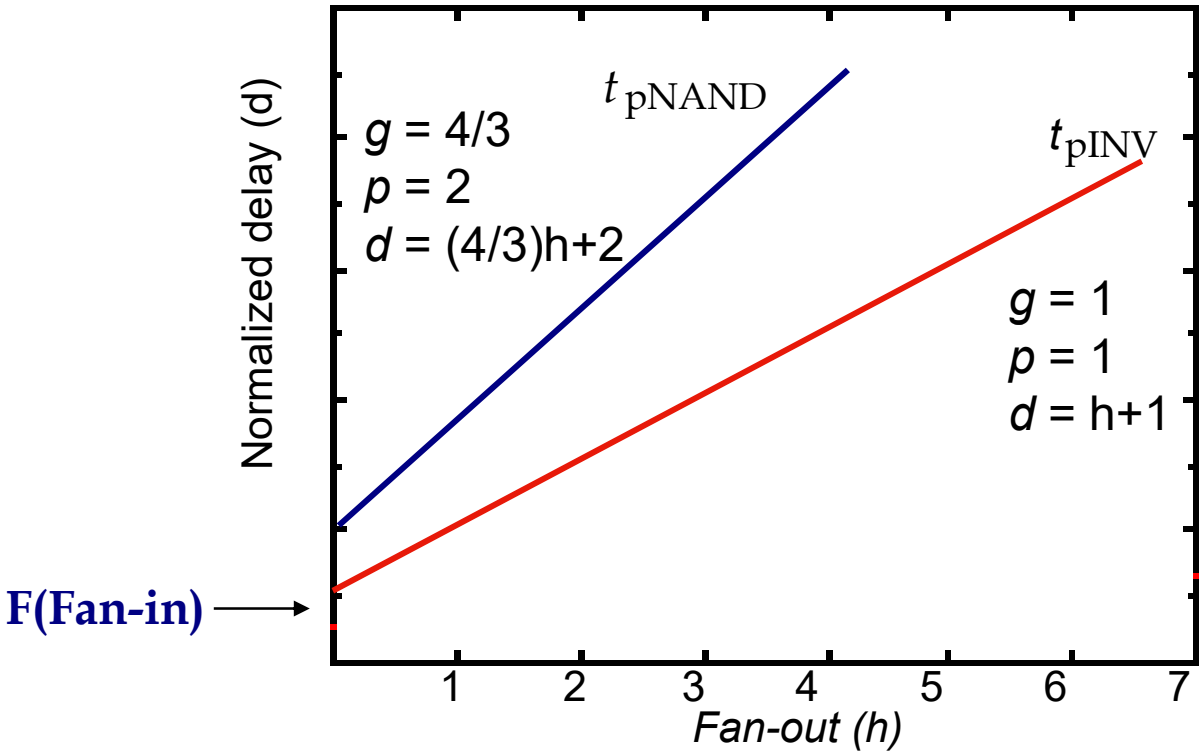
2-input NOR

***g* = 5/3**

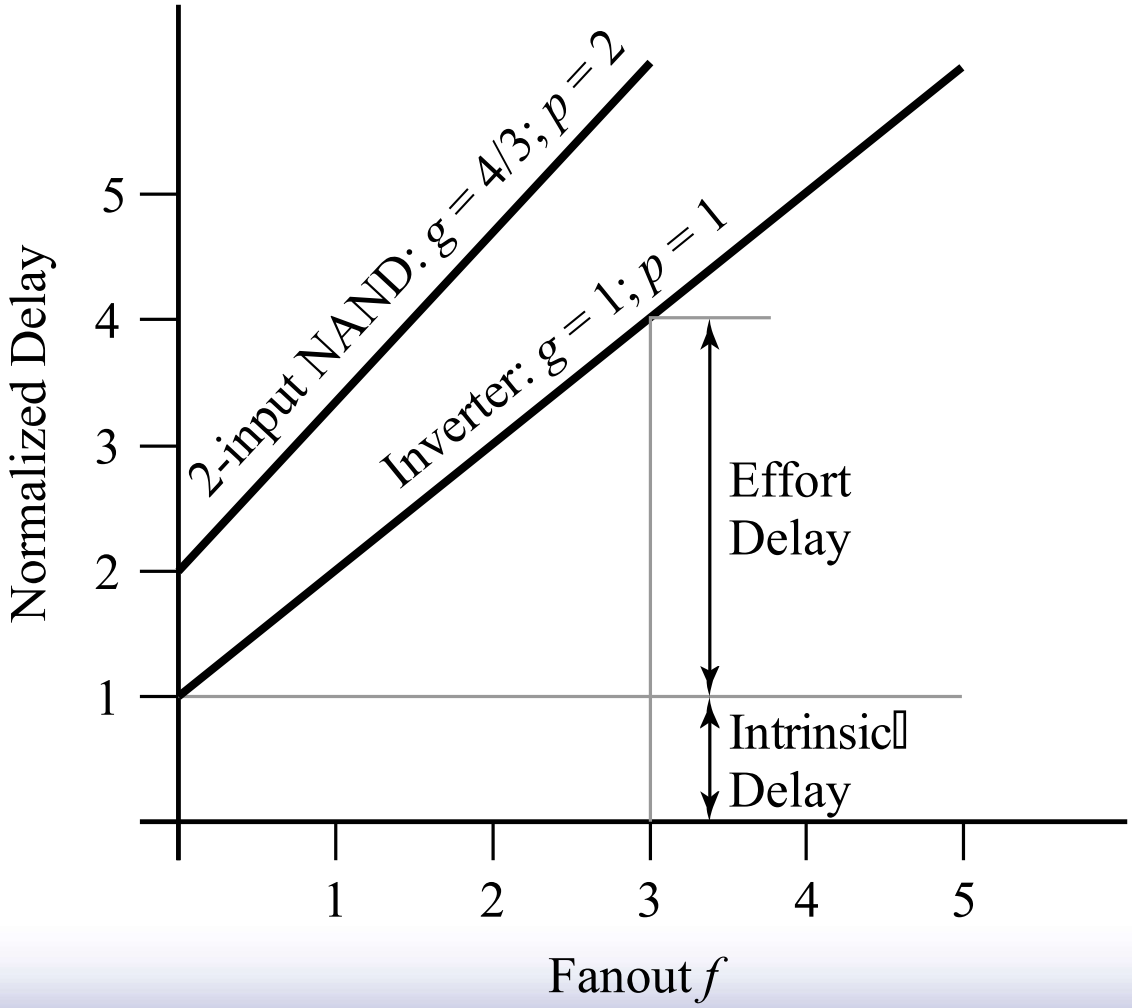
Logical Effort of Gates



Logical Effort of Gates



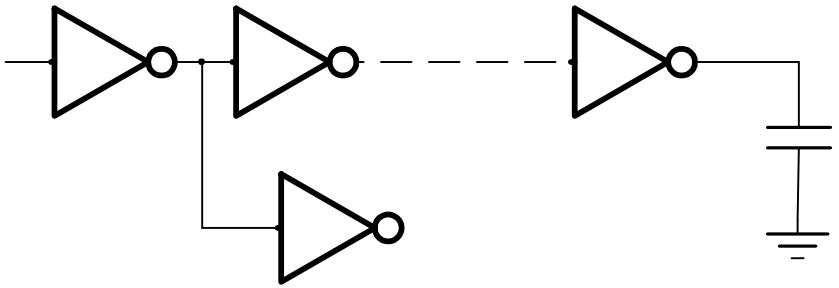
Logical Effort of Gates



Add Branching Effort

Branching effort:

$$b = \frac{C_{on-path} + C_{off-path}}{C_{on-path}}$$



Multistage Networks

$$Delay = \sum_{i=1}^{N} \left(p_i + g_i \cdot f_i \right)$$

Stage effort: $h_i = g_i f_i$

Path electrical effort: $F = C_{out}/C_{in}$

Path logical effort: $G = g_1 g_2 \ldots g_N$

Branching effort: $B = b_1 b_2 \ldots b_N$

Path effort: $H = GFB$

Path delay $D = \Sigma d_i = \Sigma p_i + \Sigma h_i$

Optimum Effort per Stage

When each stage bears the same effort:

$$h^N = H$$

$$h = \sqrt[N]{H}$$

Stage efforts: $g_1 f_1 = g_2 f_2 = \ldots = g_N f_N$

Effective fanout of each stage: $f_i = h / g_i$

Minimum path delay

$$\hat{D} = \sum \left(g_i f_i + p_i \right) = NH^{1/N} + P$$

Optimal Number of Stages

For a given load,
and given input capacitance of the first gate
Find optimal number of stages and optimal sizing

$$D = NH^{1/N} + Np_{inv}$$

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

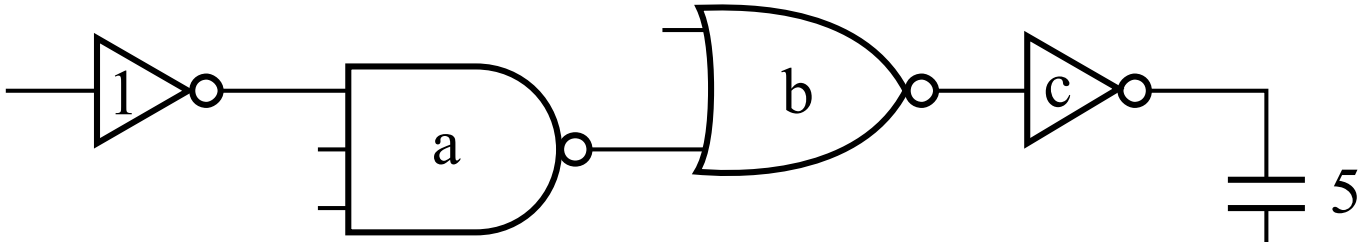
Substitute 'best stage effort' $h = H^{1/\hat{N}}$

Logical Effort

Gate Type	Number of Inputs			
	1	2	3	n
Inverter	1			
NAND		4/3	5/3	(n + 2)/3
NOR		5/3	7/3	(2n + 1)/3
Multiplexer		2	2	2
XOR		4	12	

From Sutherland, Sproull

Example: Optimize Path



$g = 1$	$g = 5/3$	$g = 5/3$	$g = 1$
$f = a$	$f = b/a$	$f = c/b$	$f = 5/c$

Effective fanout, F =

G =

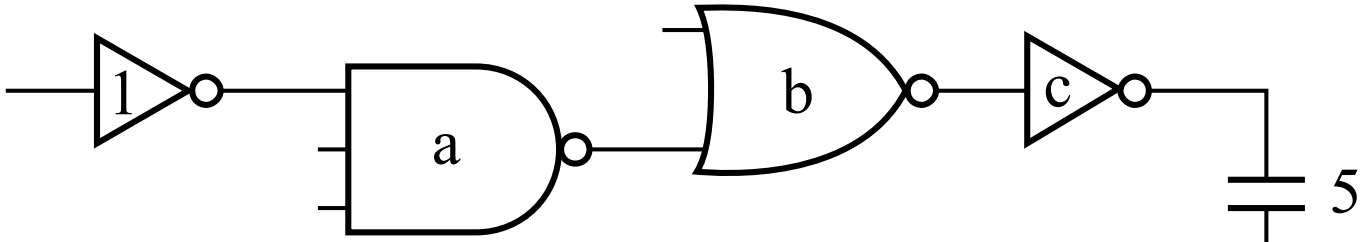
H =

h =

a =

b =

Example: Optimize Path



$g = 1$	$g = 5/3$	$g = 5/3$	$g = 1$
$f = a$	$f = b/a$	$f = c/b$	$f = 5/c$

Effective fanout, F = 5

$G = 25/9$

$H = 125/9 = 13.9$

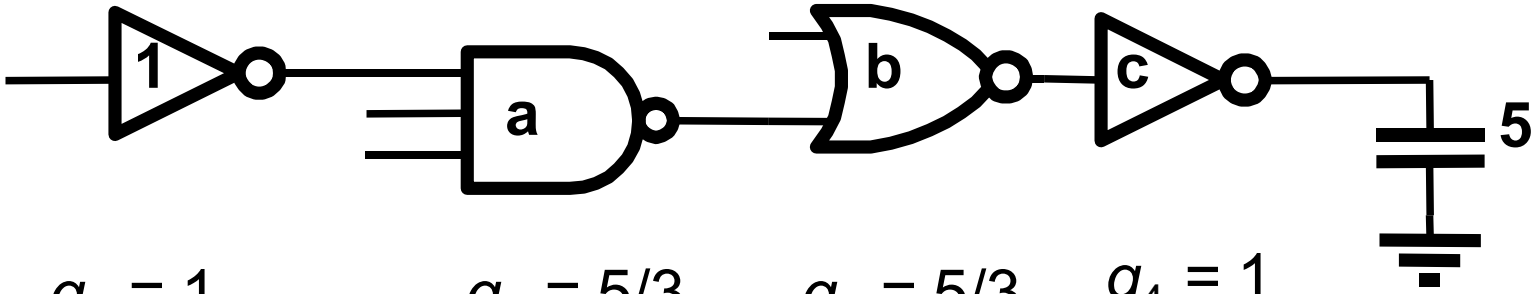
$h = 1.93$

$a = 1.93$

$b = ha/g_2 = 2.23$

$c = hb/g_3 = 5g_4/f = 2.59$

Example: Optimize Path



$g_1 = 1$ $g_2 = 5/3$ $g_3 = 5/3$ $g_4 = 1$

Effective fanout, H = 5

$G = 25/9$

$F = 125/9 = 13.9$

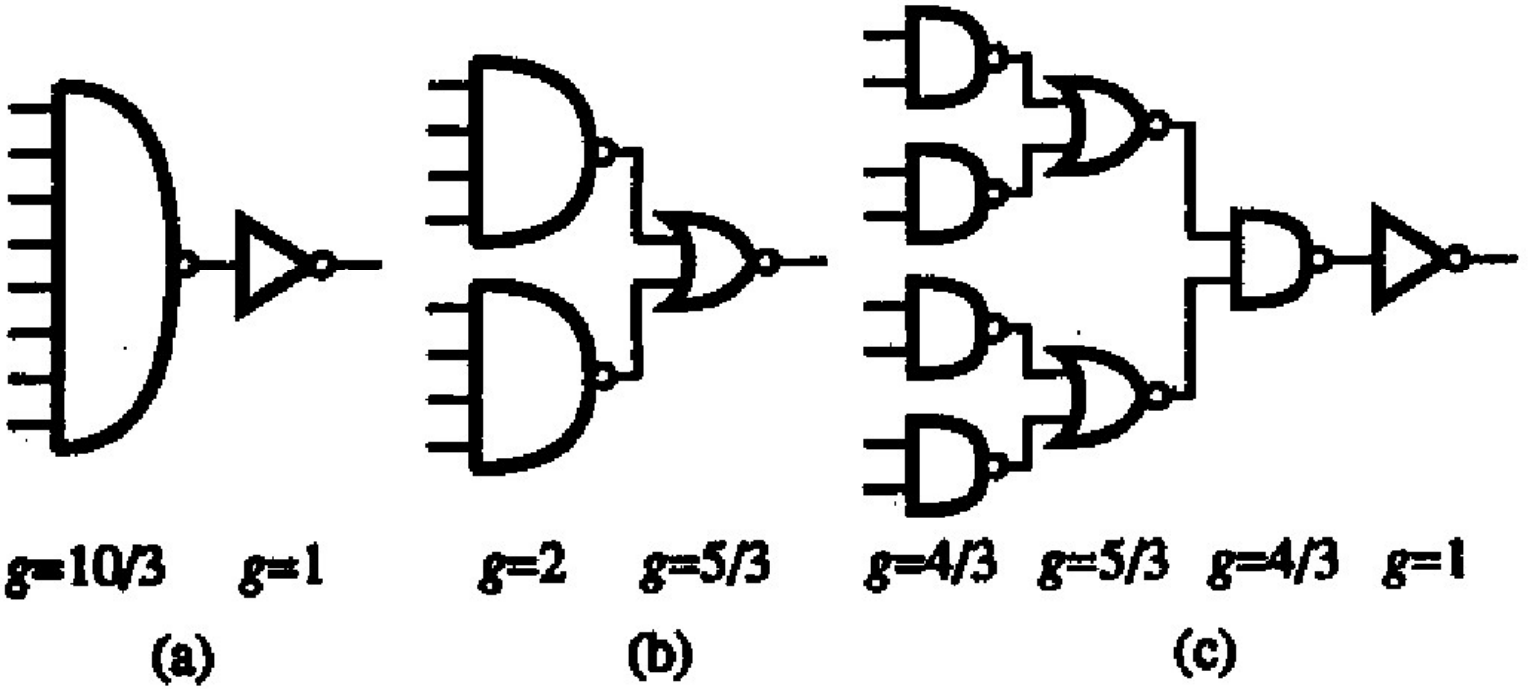
$f = 1.93$

$a = 1.93$

$b = fa/g_2 = 2.23$

$c = fb/g_3 = 5g_4/f = 2.59$

Example – 8-input AND



Method of Logical Effort

- Compute the path effort: $F = GBH$
- Find the best number of stages $N \sim \log_4 F$
- Compute the stage effort $f = F^{1/N}$
- Sketch the path with this number of stages
- Work either from either end, find sizes:
  $C_{in} = C_{out} * g/f$

Reference: Sutherland, Sproull, Harris, "Logical Effort, Morgan-Kaufmann 1999.

Summary

Table 4: Key Definitions of Logical Effort

Term	Stage expression	Path expression
Logical effort	g (seeTable 1)	$G = \prod g_i$
Electrical effort	$h = \frac{C_{out}}{C_{in}}$	$H = \frac{C_{out\ (path)}}{C_{in\ (path)}}$
Branching effort	n/a	$B = \prod b_i$
Effort	$f = gh$	$F = GBH$
Effort delay	f	$D_F = \sum f_i$
Number of stages	1	N
Parasitic delay	p (seeTable 2)	$P = \sum p_i$
Delay	$d = f + p$	$D = D_F + P$

Sutherland,
Sproull
Harris