CAL POLY
SAN LUIS OBISPO

EE 431: COMPUTER-AIDED DESIGN OF VLSI DEVICES

Combinational Circuits

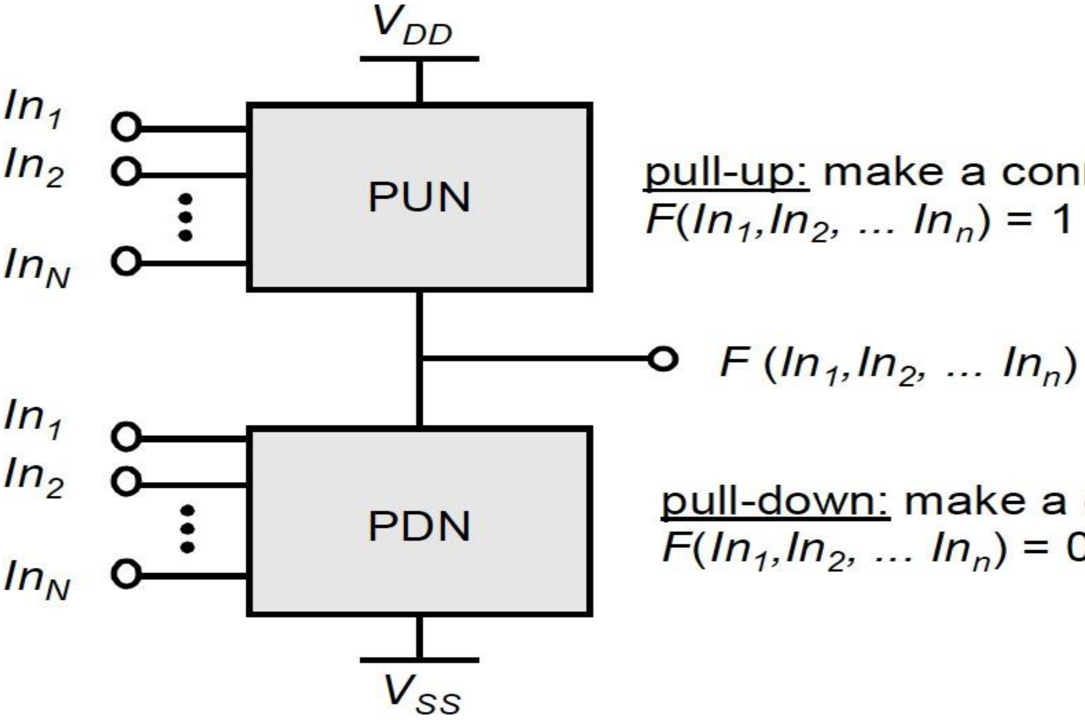
Nishith N. Chakraborty

October, 2024

Adapted from: Dr. Garrett Rose, Dr. Ahmedullah aziz University of Tennessee

CMOS Static Gate Basics

COMPLEMENTARY MOS



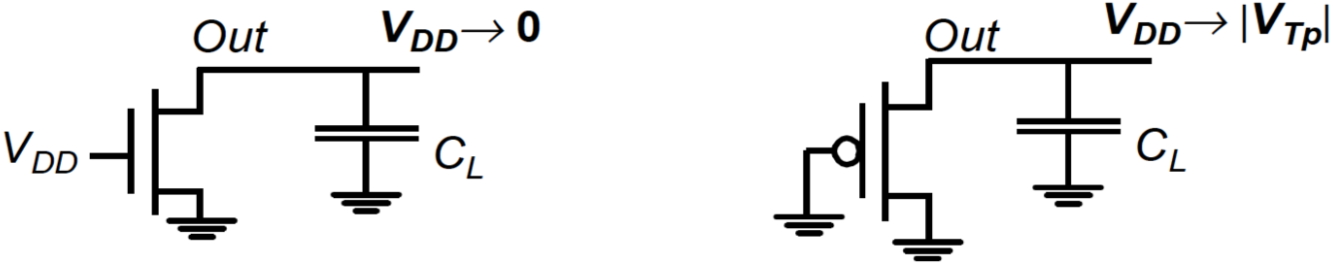
pull-up: make a connection from V_{DD} to F when $F(In_1, In_2, \ldots\ In_n) = 1$

pull-down: make a connection from V_{DD} to V_{ss} when $F(In_1, In_2, \ldots\ In_n) = 0$

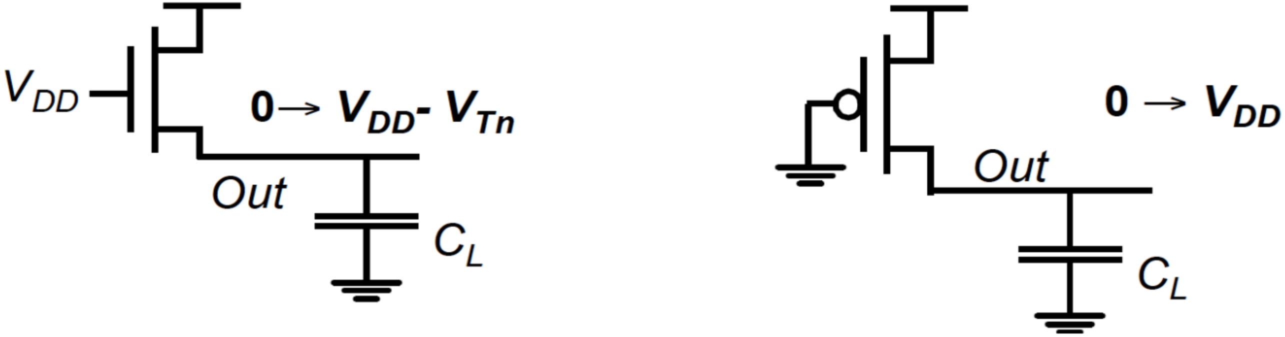
CHOOSING PDN AND PUN

PDN → NMOS

PUN → PMOS



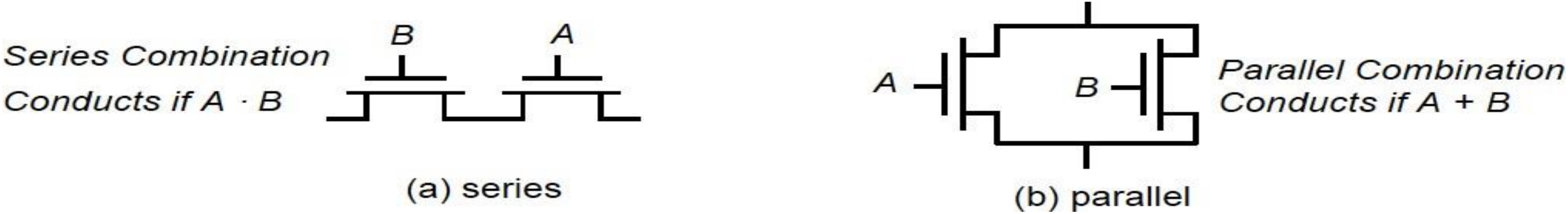
Pulling Down a node with NMOS and PMOS transistors



Pulling Up a node with NMOS and PMOS transistors

LOGIC RULES

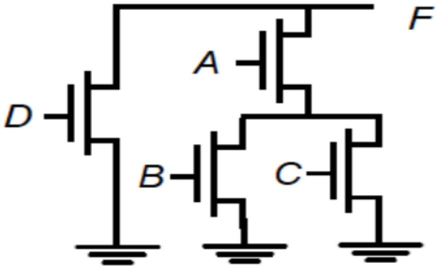
- NMOS devices connected in series corresponds to an (N)AND function. Similarly, NMOS transistors connected in parallel represent an (N)OR function. Similarly, construction rules for PMOS networks can be formulated.
- The pull-up and pull-down networks of a complementary CMOS structure are dual networks. This means that a parallel connection of transistors in the pull-up network corresponds to a series connection of the corresponding devices in the pull-down network, and vice versa
- Therefore, to construct a CMOS gate, one of the networks (e.g., PDN) is implemented using combinations of series and parallel devices. The other network (i.e., PUN) is obtained using duality principle by walking the hierarchy, replacing series sub-nets with parallel sub-nets, and parallel sub-nets with series sub-nets. The complete CMOS gate is constructed by combining the PDN with the PUN.



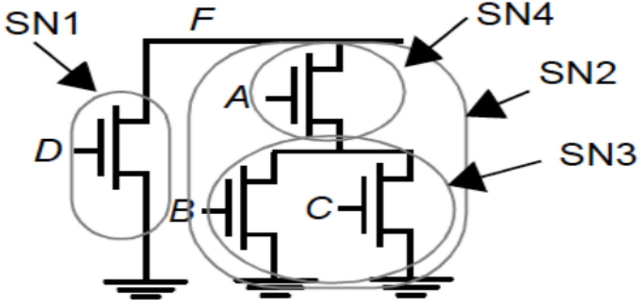
(a) series

(b) parallel

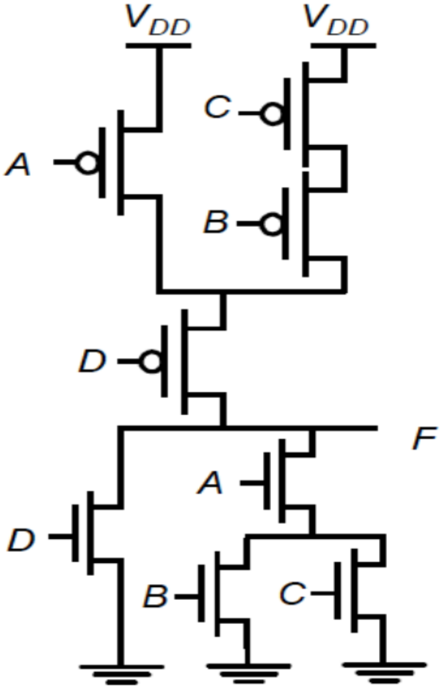
IDENTIFYING SUB-NETS ($\overline{F=D+A(B+C)}$)



(a) pull-down network

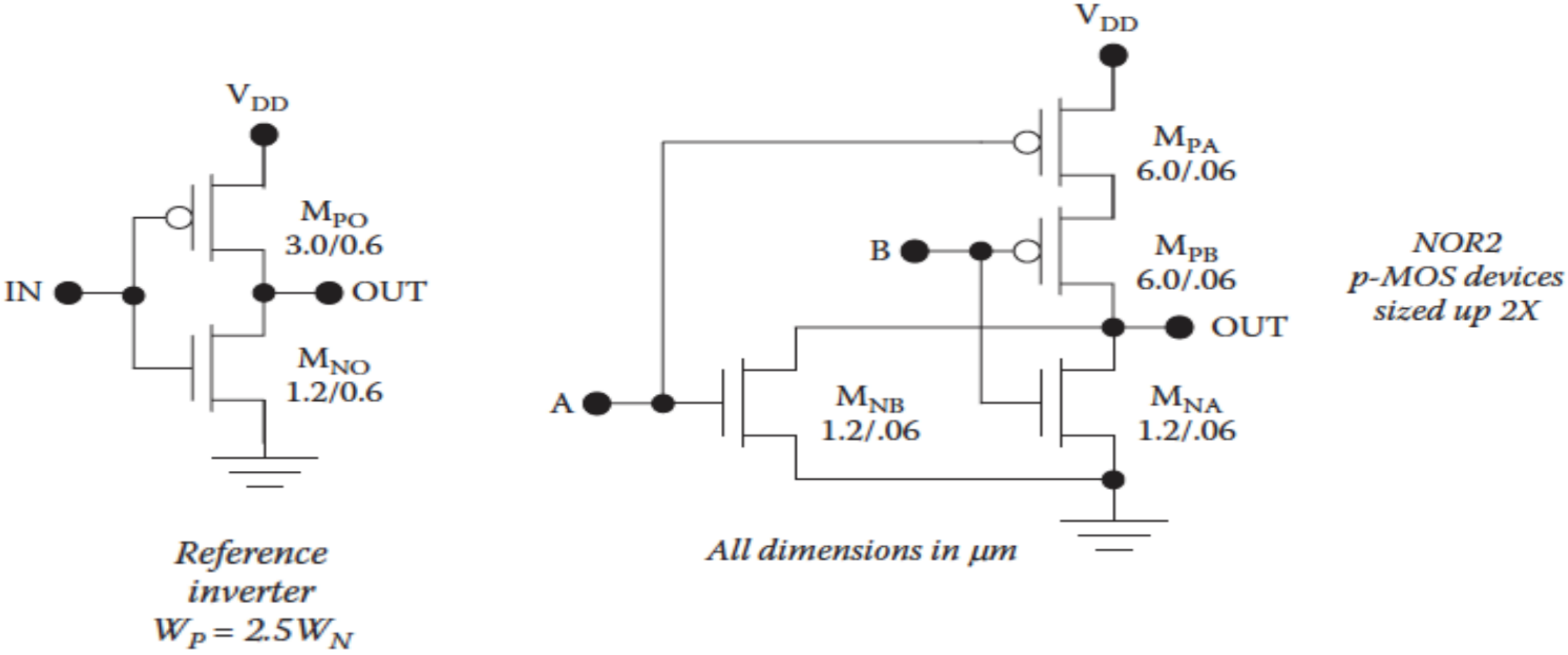


(b) Deriving the pull-up network hierarchically by identifying sub-nets



(c) complete gate

TRANSISTOR SIZING

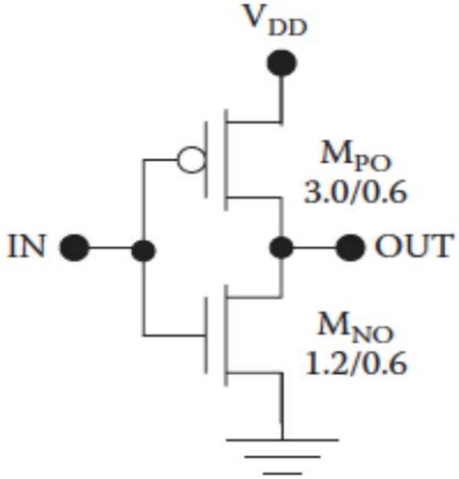


Reference inverter
$W_P = 2.5W_N$

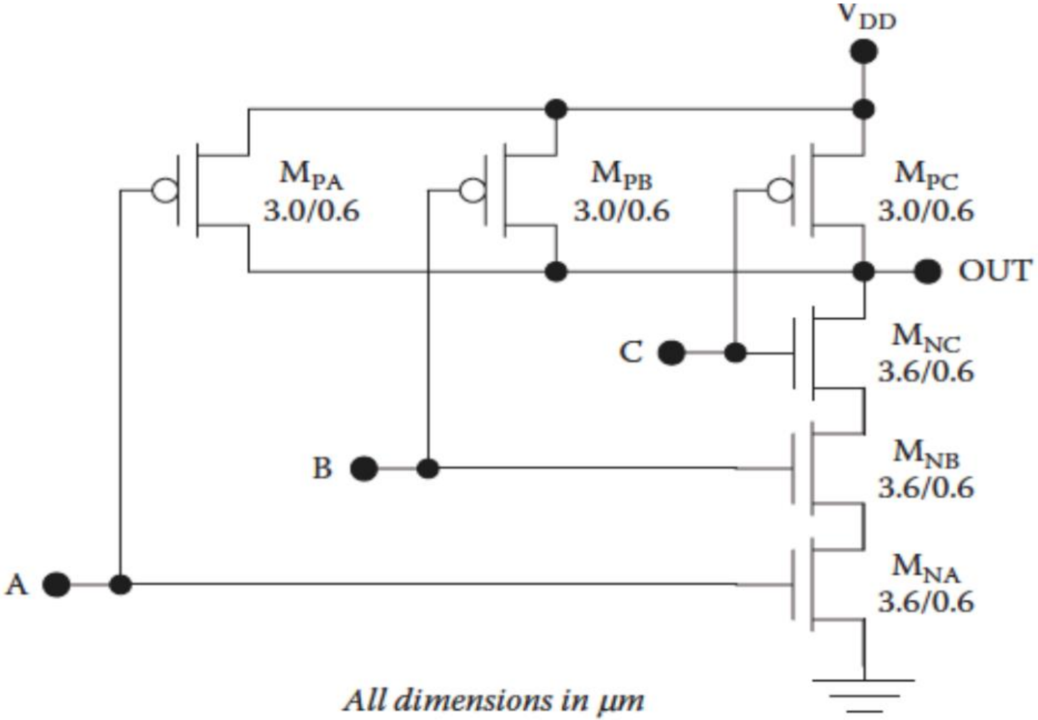
All dimensions in μm

NOR2
p-MOS devices
sized up 2X

TRANSISTOR SIZING



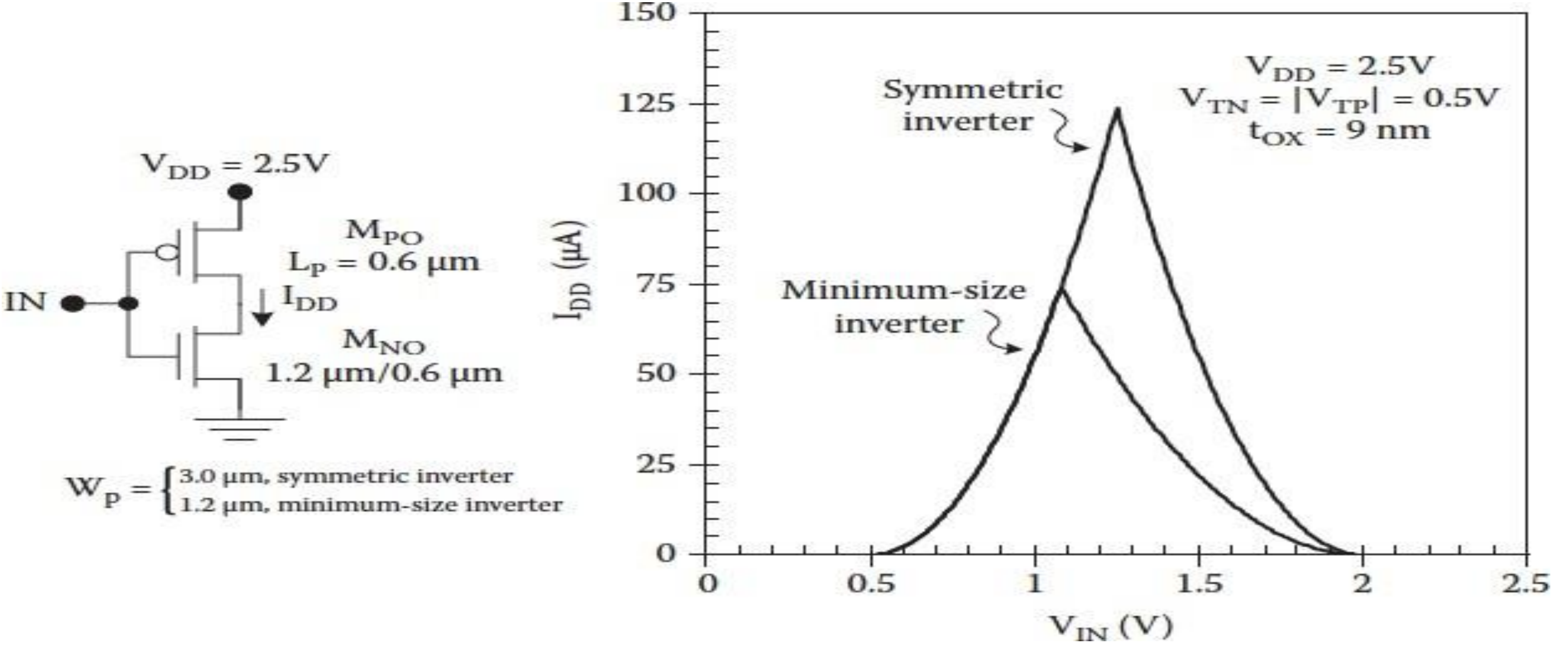
Reference inverter
$W_P = 2.5W_N$



All dimensions in μm

NAND3
n-MOS device
sized up 3X

TRANSISTOR SIZING



Bubble Pushing and Compound Gates

EXAMPLE 1

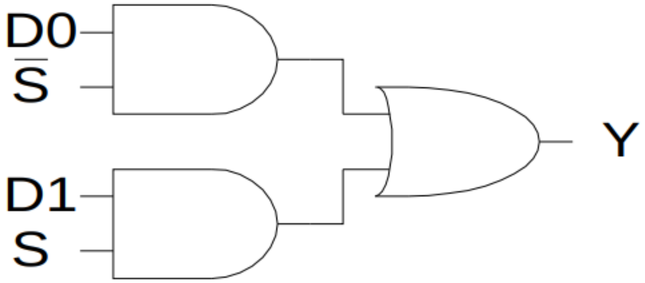
```
module mux(input  s, d0, d1, output y);
assign y = s ? d1 : d0;
Endmodule
```

1) Sketch a design using AND, OR, and NOT gates.

EXAMPLE 1

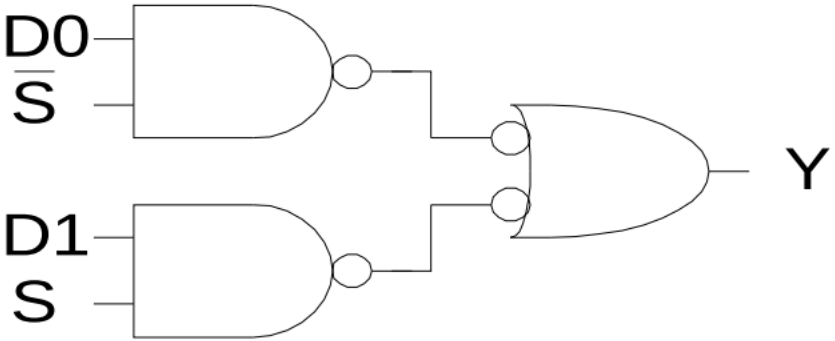
```
module mux(input  s, d0, d1, output y);
assign y = s ? d1 : d0;
Endmodule
```

1) Sketch a design using AND, OR, and NOT gates.



EXAMPLE 2

2) Sketch a design using NAND, NOR, and NOT gates. Assume ~S is available.

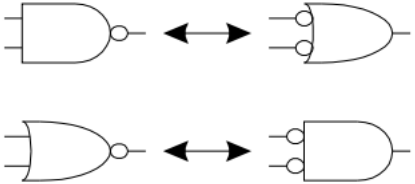


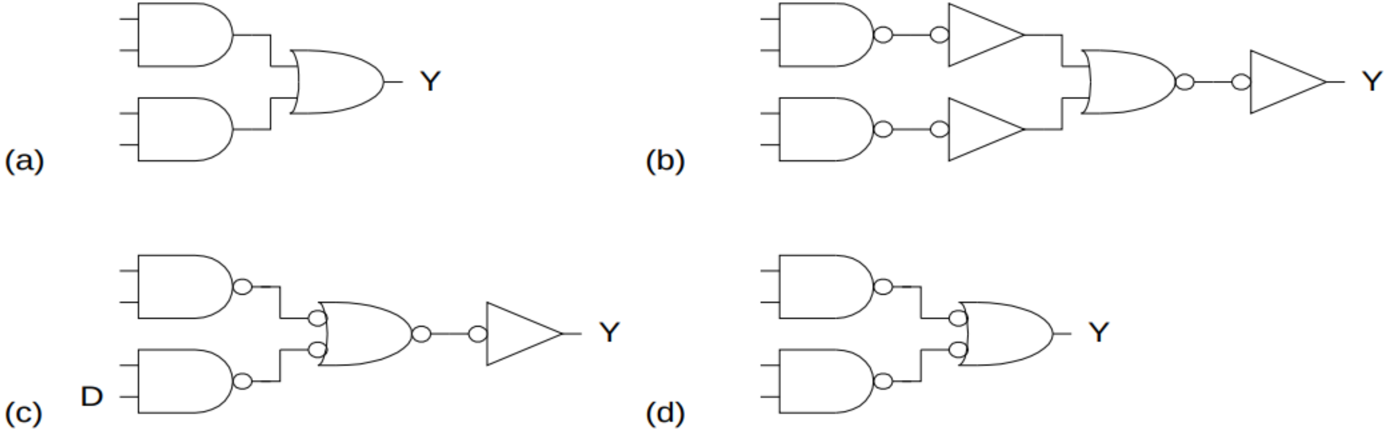
BUBBLE PUSHING

- Start with network of AND / OR gates
- Convert to NAND / NOR + inverters
- Push bubbles around to simplify logic
    - ➢ Remember DeMorgan’s Law

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

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



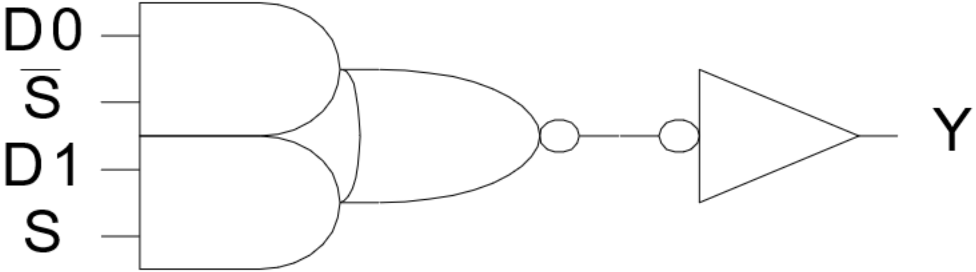


EXAMPLE 3

3) Sketch a design using one compound gate and one NOT gate. Assume ~S is available.

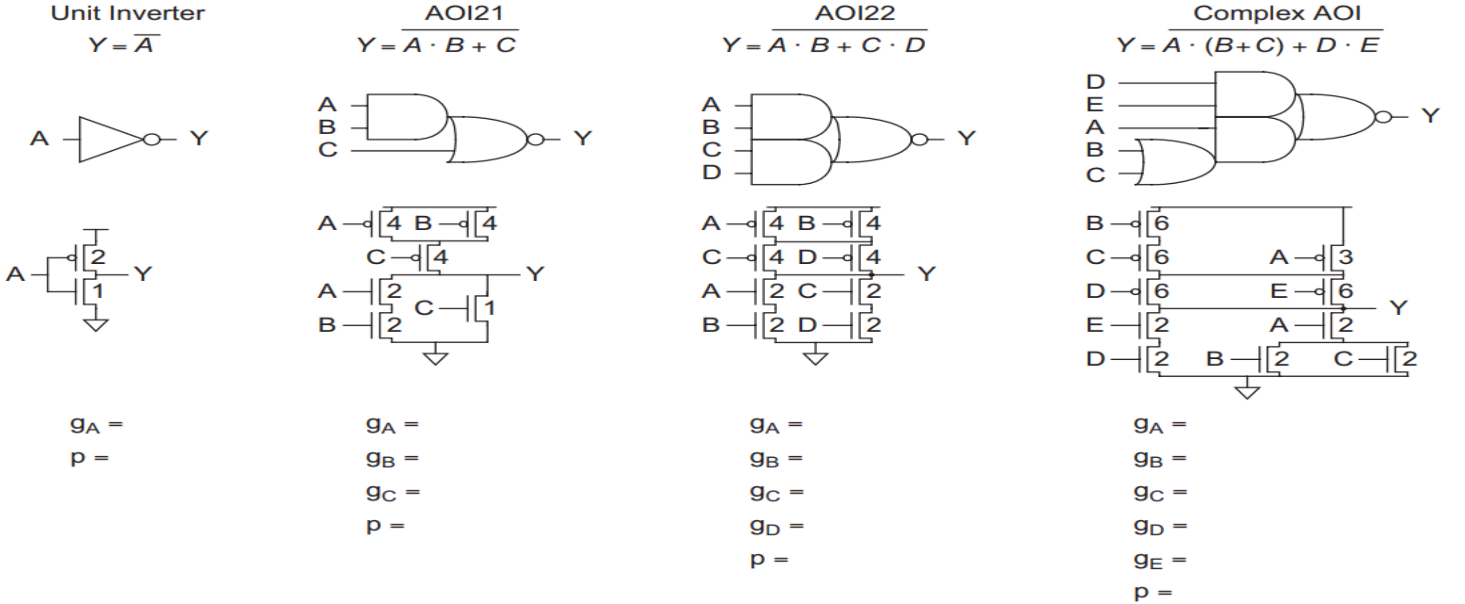
EXAMPLE 3

3) Sketch a design using one compound gate and one NOT gate. Assume ~S is available.



COMPOUND GATES

- Logical effort of compound gates:



Unit Inverter

$Y = \overline{A}$

$g_A =$

$p =$

AOI21

$Y = \overline{A \cdot B + C}$

$g_A =$

$g_B =$

$g_C =$

$p =$

AOI22

$Y = \overline{A \cdot B + C \cdot D}$

$g_A =$

$g_B =$

$g_C =$

$g_D =$

$p =$

Complex AOI

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

$g_A =$

$g_B =$

$g_C =$

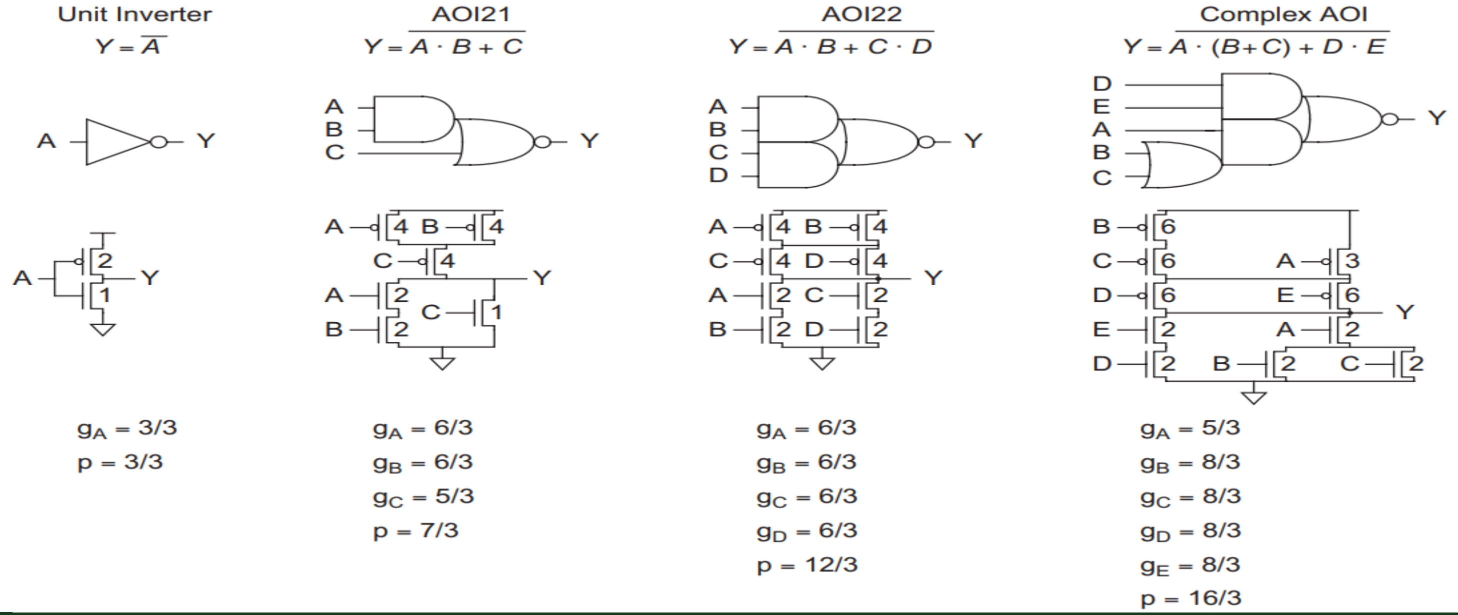
$g_D =$

$g_E =$

$p =$

COMPOUND GATES

- Logical effort of compound gates:



Unit Inverter
$Y = \overline{A}$

$g_A = 3/3$
$p = 3/3$

AOI21
$Y = \overline{A \cdot B + C}$

$g_A = 6/3$
$g_B = 6/3$
$g_C = 5/3$
$p = 7/3$

AOI22
$Y = \overline{A \cdot B + C \cdot D}$

$g_A = 6/3$
$g_B = 6/3$
$g_C = 6/3$
$g_D = 6/3$
$p = 12/3$

Complex AOI
$Y = \overline{A \cdot (B + C) + D \cdot E}$

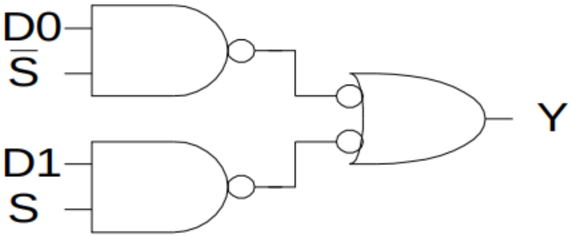
$g_A = 5/3$
$g_B = 8/3$
$g_C = 8/3$
$g_D = 8/3$
$g_E = 8/3$
$p = 16/3$

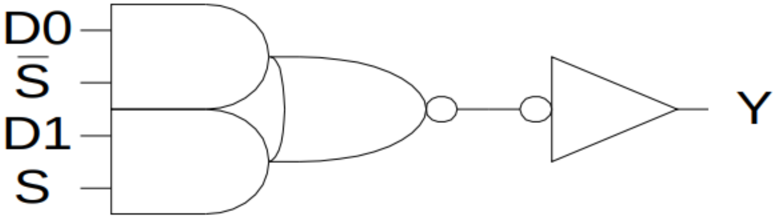
EXAMPLE 4

- The multiplexer has a maximum input capacitance of 16 units on each input. It must drive a load of 160 units. Estimate the delay of the NAND and compound gate designs.

EXAMPLE 4

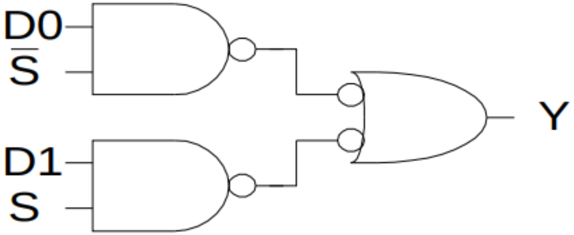
- The multiplexer has a maximum input capacitance of 16 units on each input. It must drive a load of 160 units. Estimate the delay of the NAND and compound gate designs.





H = 160 / 16 = 10
B = 1
N = 2

NAND SOLUTION



NAND SOLUTION

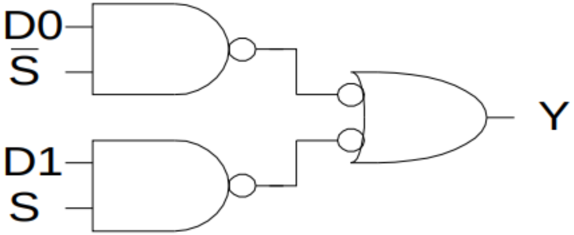
$$P = 2 + 2 = 4$$

$$G = (4/3)\ (4/3) = 16/9$$

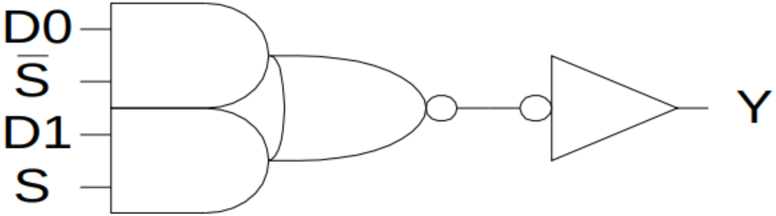
$$F = GBH = 160/9$$

$$\hat{f} = \sqrt[N]{F} = 4.2$$

$$D = N\hat{f} + P = 12.4\tau$$



COMPOUND SOLUTION



COMPOUND SOLUTION

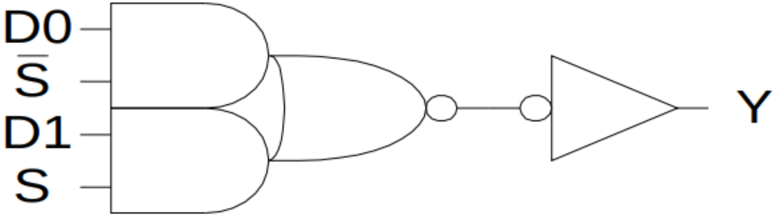
$$P = 4 + 1 = 5$$

$$G = (6/3)\ (1) = 2$$

$$F = GBH = 20$$

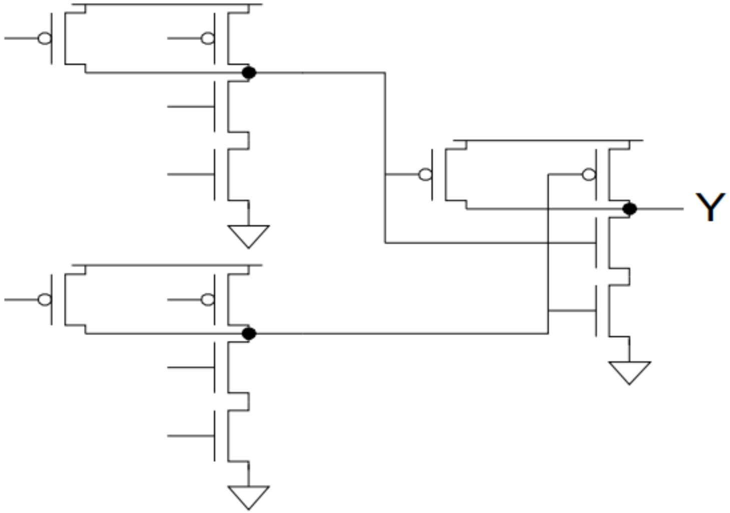
$$\hat{f} = \sqrt[N]{F} = 4.5$$

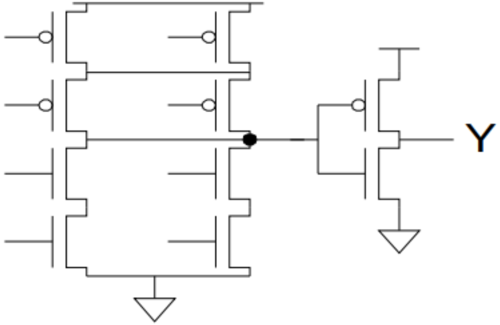
$$D = N\hat{f} + P = 14\tau$$



EXAMPLE 5

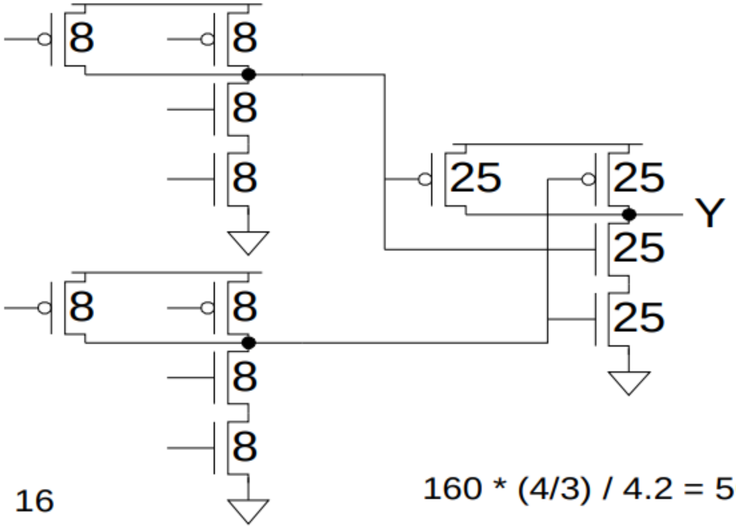
Annotate your designs with transistor sizes that achieve this delay.





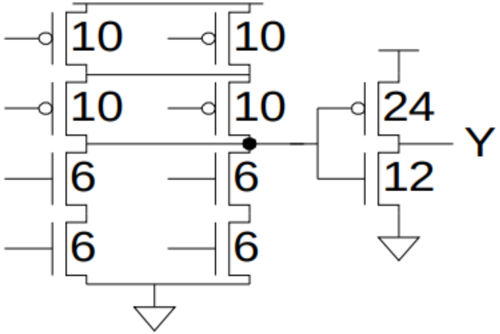
EXAMPLE 5

Annotate your designs with transistor sizes that achieve this delay.



16

160 * (4/3) / 4.2 = 50



16

160 * 1 / 4.5 = 36

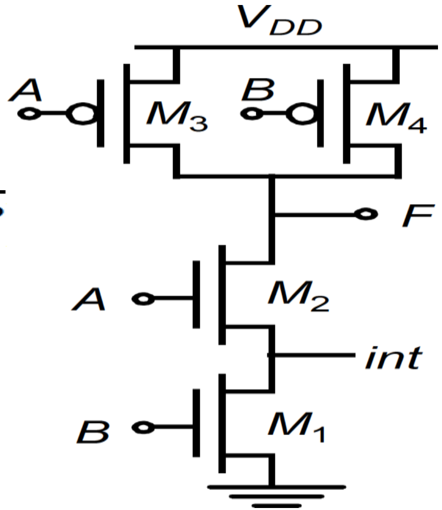
Static and Dynamic Properties

TWO-INPUT NAND GATE: STATIC PROPERTY

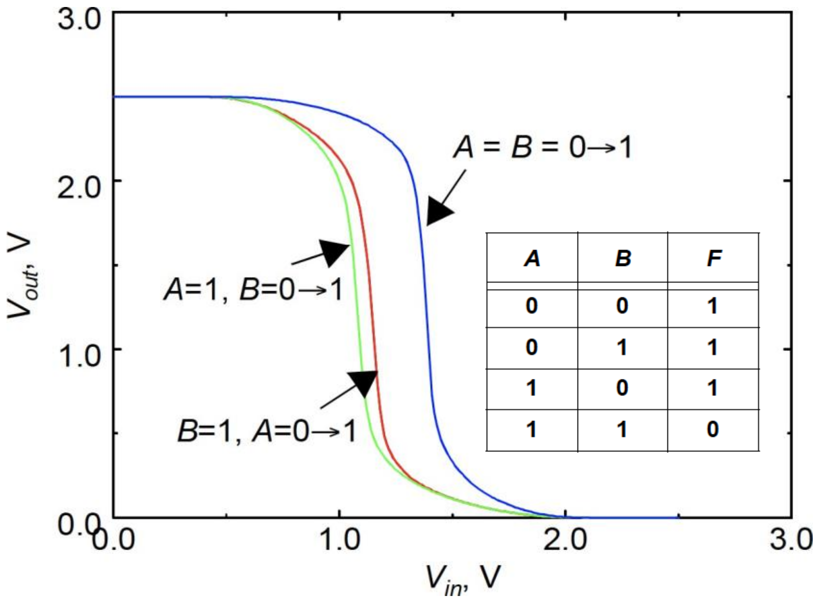
$$V_{Tn2} = V_{tn0} + \gamma((\sqrt{|2\phi_f| + V_{int}}) - \sqrt{|2\phi_f|})$$

$$V_{Tn1} = V_{tn0}$$

$$F = \overline{A \cdot B}$$

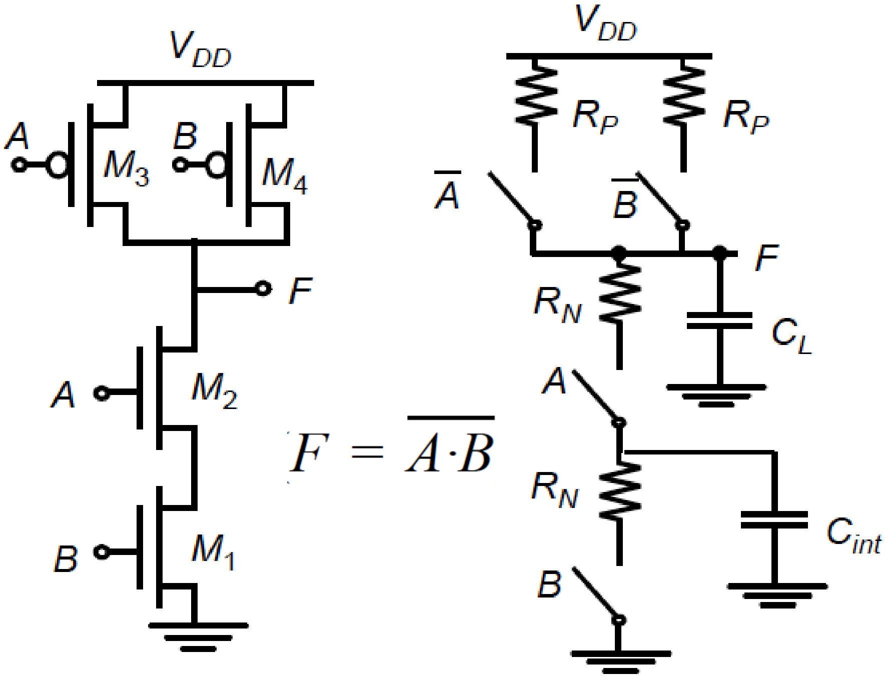


Data dependent VTC



A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

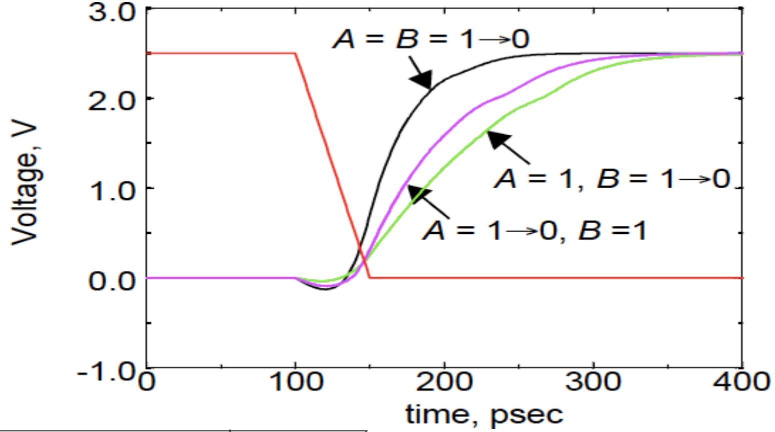
TWO-INPUT NAND GATE: DYNAMIC PROPERTY



$$F = \overline{A \cdot B}$$

(a) Two-input NAND

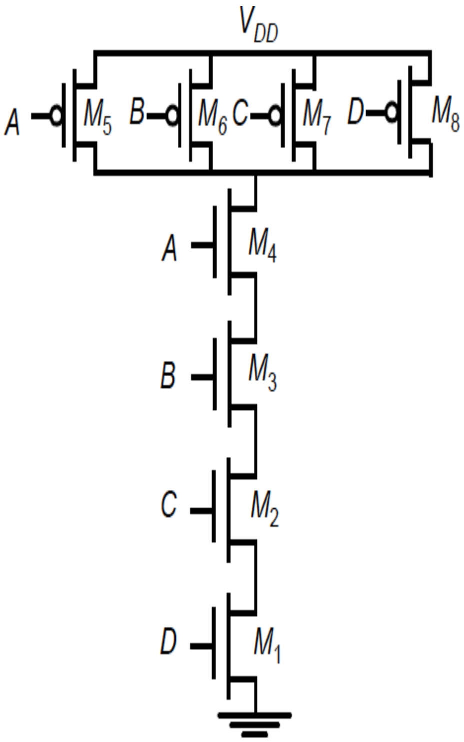
(b) RC equivalent model

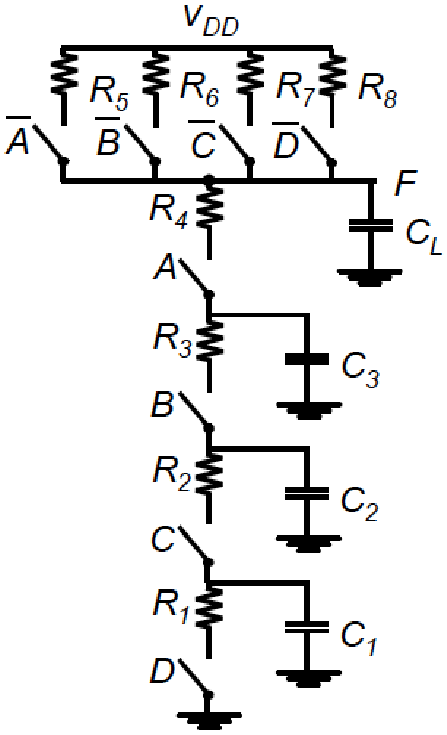


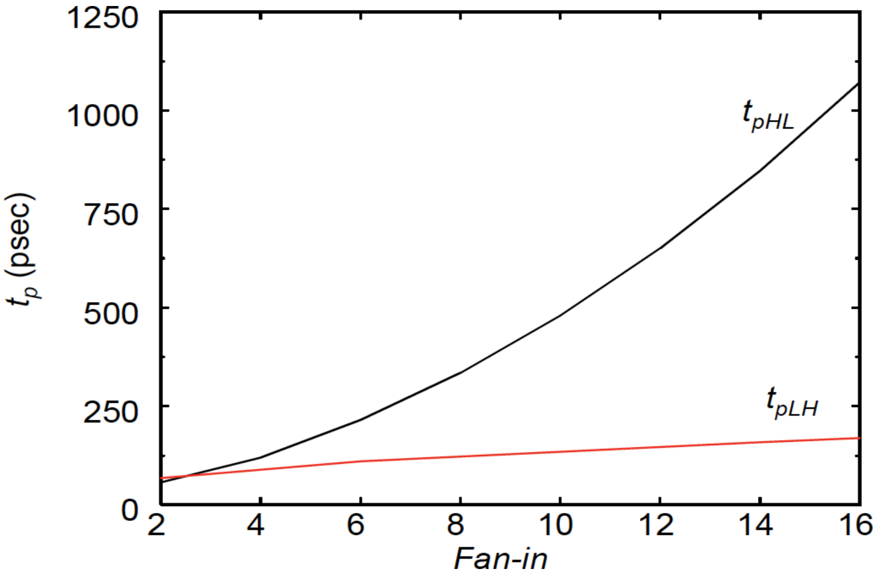
Input Data Pattern	Delay (psec)
A = B= 0→1	69
A = 1, B= 0→1	62
A= 0→1, B = 1	50
A=B=1→0	35
A=1, B = 1→0	76
A= 1→0, B = 1	57

The delay is dependent on the input pattern

FOUR-INPUT NAND GATE: DYNAMIC PROPERTY

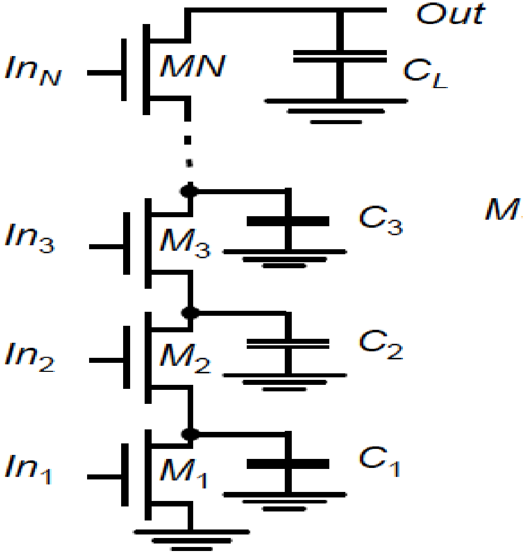




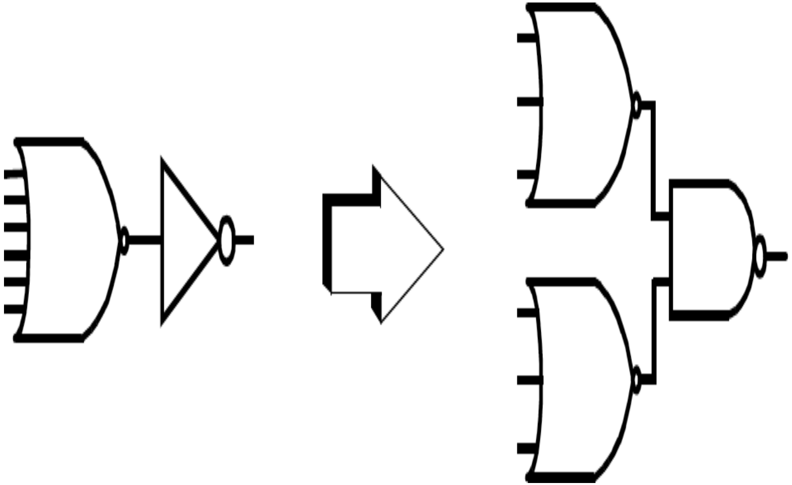


Propagation delay of CMOS NAND gate as a function of fan-in. A fan-out of one inverter is assumed, and all pull-down transistors are minimal size

PROGRESSIVE SIZING & LOGIC RECONSTRUCTION



$M_1 > M_2 > M_3 > M_N$

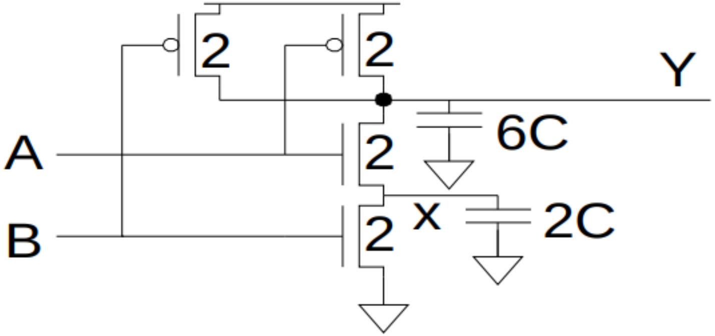


Progressive sizing of transistors in large transistor chains copes with the extra load of internal capacitances.

Logic reconstruction can reduce the gate fan-in

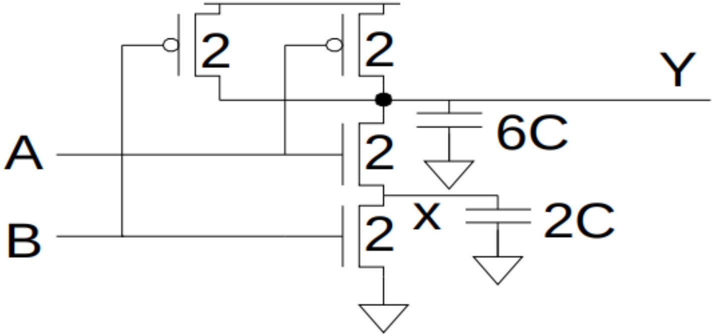
INPUT ORDER

- Our parasitic delay model was too simple
  - Calculate parasitic delay for Y falling
    - If A arrives latest?
    - If B arrives latest?



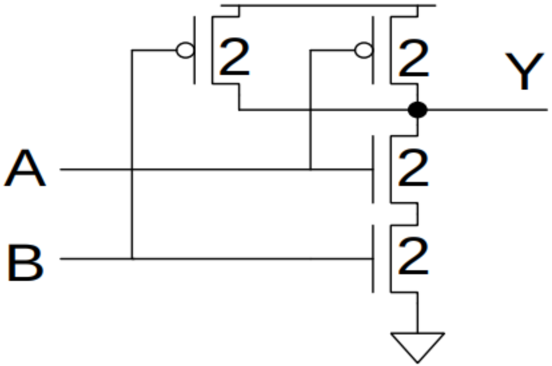
INPUT ORDER

- Our parasitic delay model was too simple
  - Calculate parasitic delay for Y falling
    - If A arrives latest? 2τ
    - If B arrives latest? 2.33τ



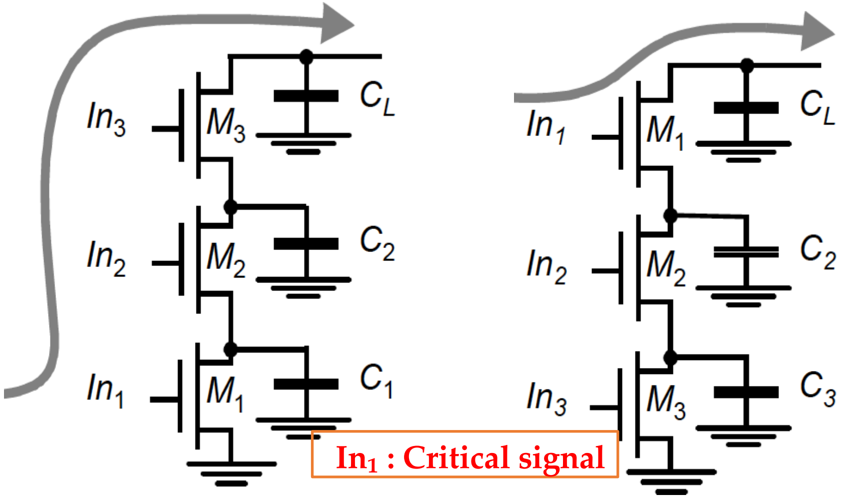
INNER & OUTER INPUTS

- Outer input is closest to rail (B)
- Inner input is closest to output (A)

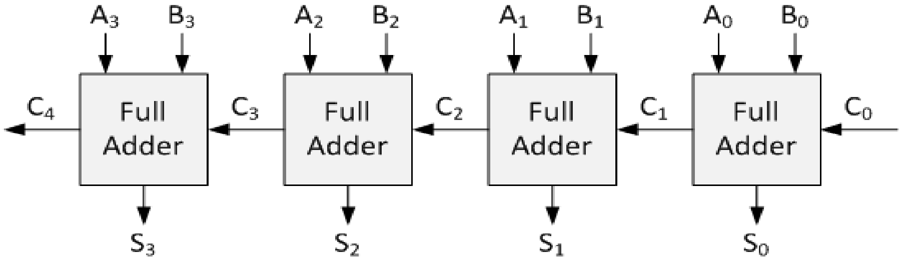


- If input arrival time is known
  - Connect latest input to inner terminal

INPUT RE-ORDERING



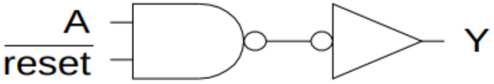
In_1 : Critical signal

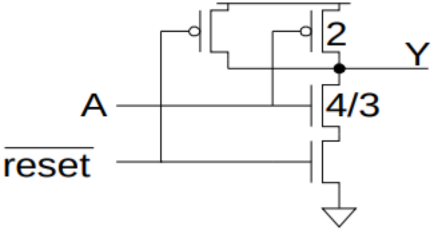


Some signals in complex combinational logic blocks might be more critical than others. Not all inputs of a gate arrive at the same time (due, for instance, to the propagation delays of the preceding logical gates). An input signal to a gate is called critical if it is the last signal of all inputs to assume a stable value. The path through the logic which determines the ultimate speed of the structure is called the critical path. Putting the critical-path transistors closer to the output of the gate can result in a speedup.

ASYMMETRIC GATES

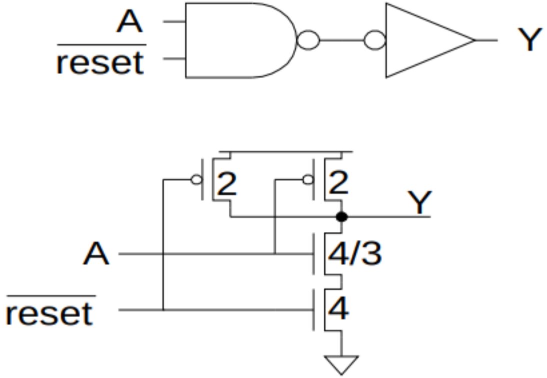
- Asymmetric gates favor one input over another
- Ex: suppose input A of a NAND gate is most critical
  - Use smaller transistor on A (less capacitance)
  - Boost size of noncritical input
  - So total resistance is same
- g_A =
- g_B =
- $g_{total} = g_A + g_B$ =
- Asymmetric gate approaches g = 1 on critical input
- But total logical effort goes up





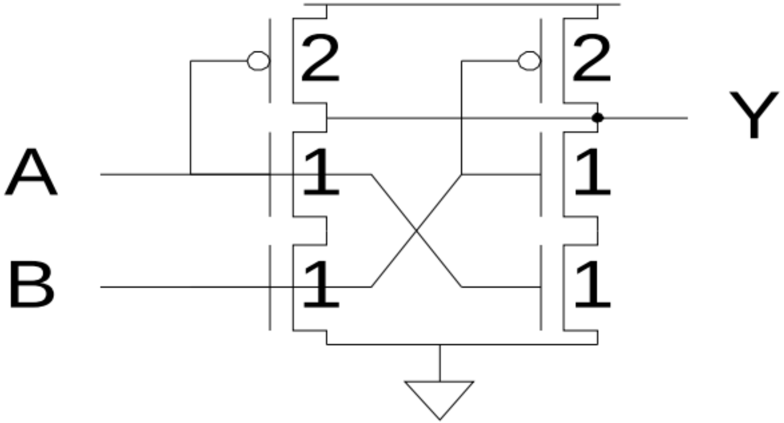
ASYMMETRIC GATES

- Asymmetric gates favor one input over another
- Ex: suppose input A of a NAND gate is most critical
  - ➢ Use smaller transistor on A (less capacitance)
  - ➢ Boost size of noncritical input
  - ➢ So total resistance is same
- $g_A = 10/9$
- $g_B = 2$
- $g_{total} = g_A + g_B = 28/9$
- Asymmetric gate approaches g = 1 on critical input
- But total logical effort goes up



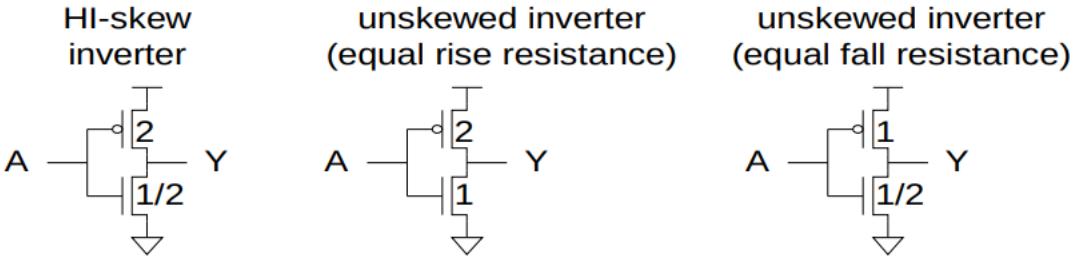
SYMMETRIC GATES

- Inputs can be made perfectly symmetric



SKEWED GATES

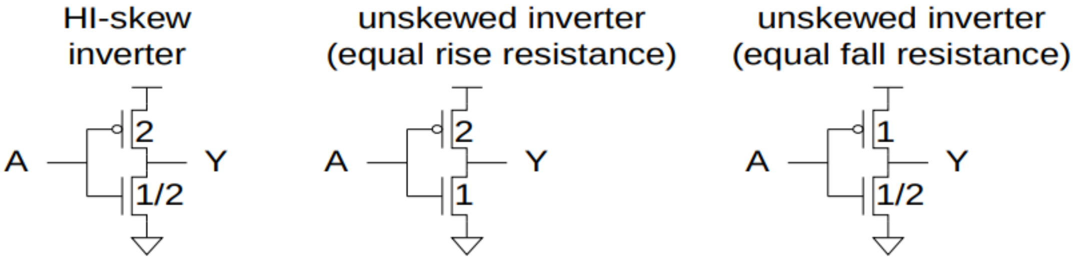
- Skewed gates favor one edge over another
- Ex: suppose rising output of inverter is most critical
  - Downsize noncritical nMOS transistor


HI-skew inverter
unskewed inverter (equal rise resistance)
unskewed inverter (equal fall resistance)
A
2
1/2
Y
A
2
1
Y
A
1
1/2
Y

- Calculate logical effort by comparing to unskewed inverter with same effective resistance on that edge.
  - g_u =
  - g_d =

SKEWED GATES

- Skewed gates favor one edge over another
- Ex: suppose rising output of inverter is most critical
  - ➢ Downsize noncritical nMOS transistor


HI-skew inverter
unskewed inverter (equal rise resistance)
unskewed inverter (equal fall resistance)
A
Y
2
1/2
A
Y
2
1
A
Y
1
1/2

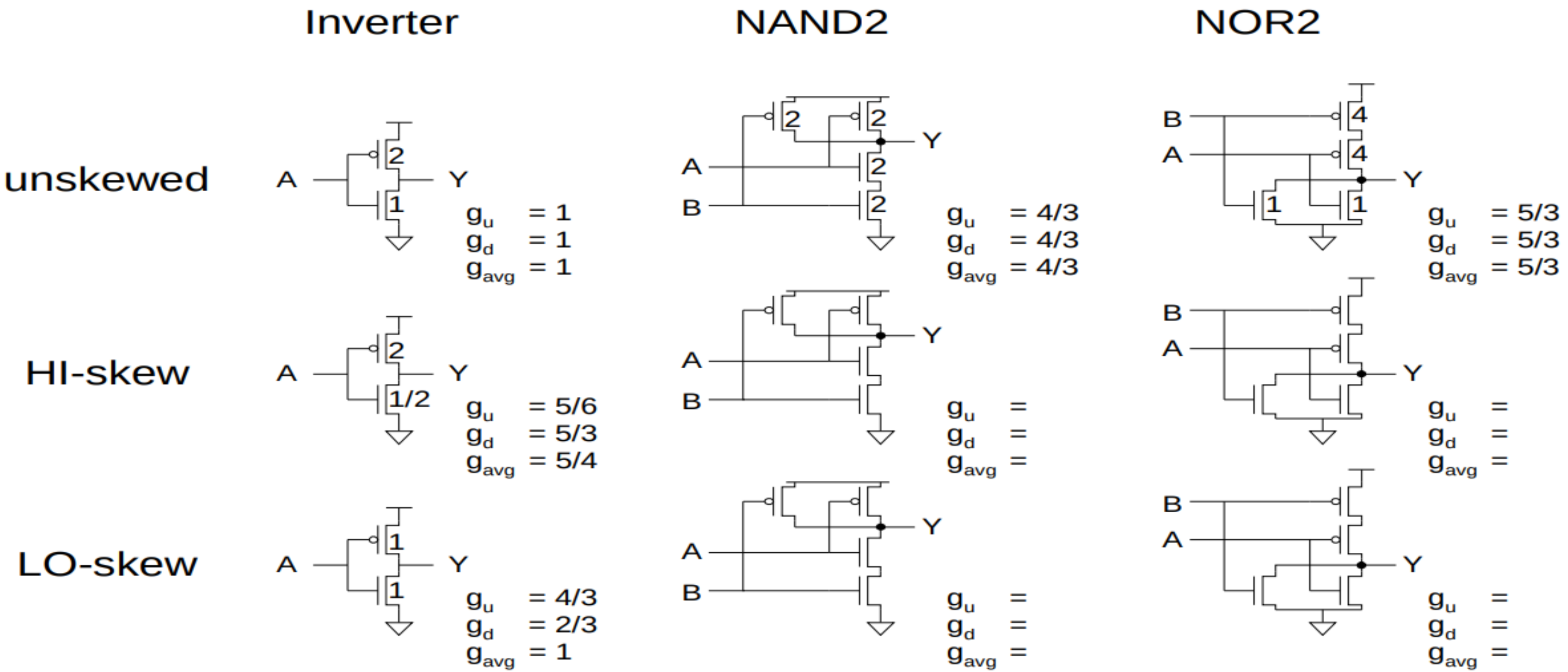
- Calculate logical effort by comparing to unskewed inverter with same effective resistance on that edge.
  - ➢ g_u = 2.5 / 3 = 5/6
  - ➢ g_d = 2.5 / 1.5 = 5/3

HI- AND LO-SKEW

- Def: Logical effort of a skewed gate for a particular transition is the ratio of the input capacitance of that gate to the input capacitance of an unskewed inverter delivering the same output current for the same transition.
- Skewed gates reduce size of noncritical transistors
  - HI-skew gates favor rising output (small nMOS)
  - LO-skew gates favor falling output (small pMOS)
- Logical effort is smaller for favored direction
- But larger for the other direction

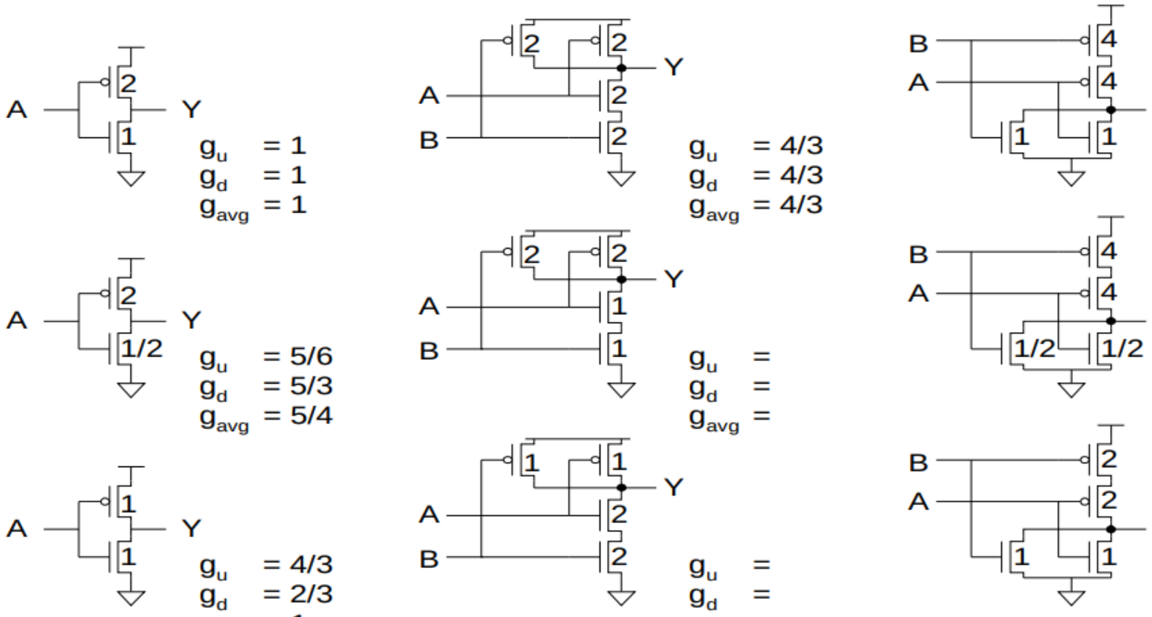
CAL POLY
SAN LUIS OBISPO

CATALOG OF SKEWED GATES



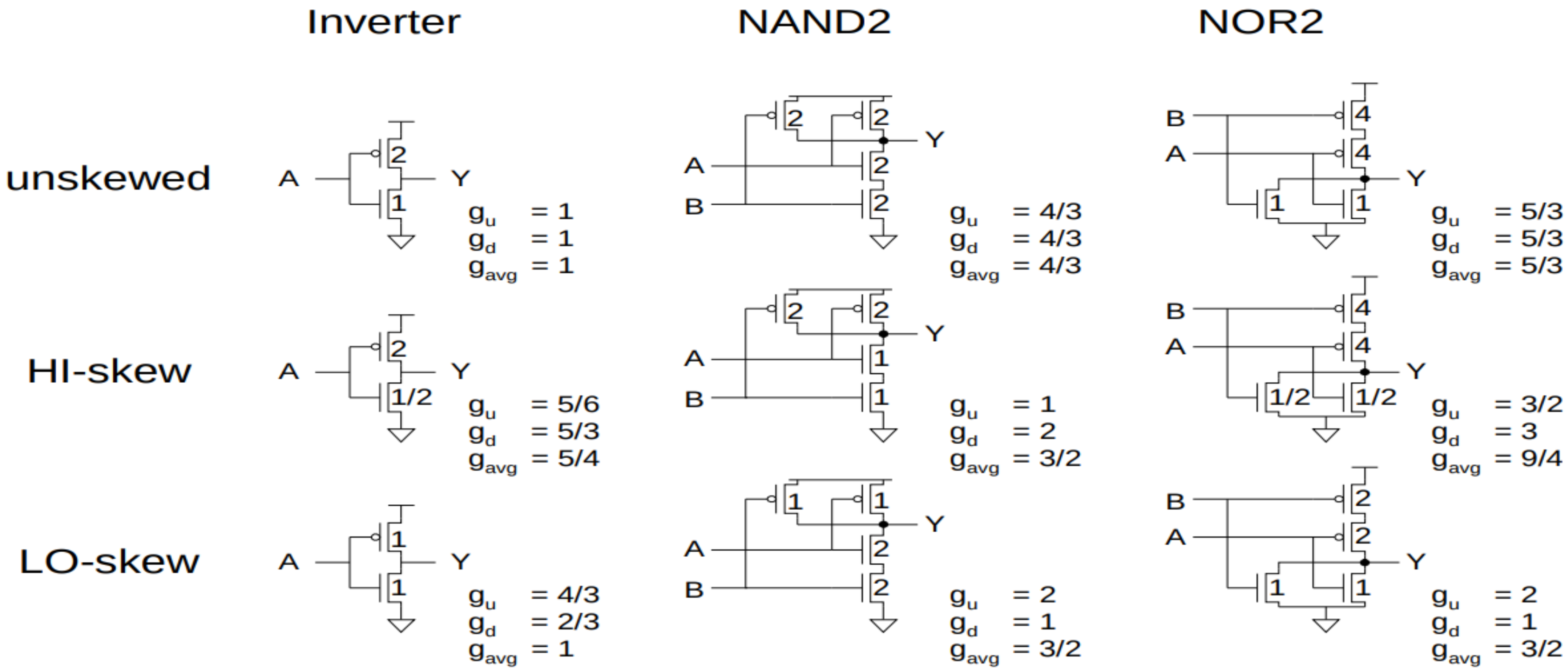
	Inverter	NAND2	NOR2
unskewed	g_u = 1 g_d = 1 g_{avg} = 1	g_u = 4/3 g_d = 4/3 g_{avg} = 4/3	g_u = 5/3 g_d = 5/3 g_{avg} = 5/3
HI-skew	g_u = 5/6 g_d = 5/3 g_{avg} = 5/4	g_u = g_d = g_{avg} =	g_u = g_d = g_{avg} =
LO-skew	g_u = 4/3 g_d = 2/3 g_{avg} = 1	g_u = g_d = g_{avg} =	g_u = g_d = g_{avg} =

CATALOG OF SKEWED GATES



	Inverter	NAND2	NOR2
unskewed	g_u = 1 g_d = 1 g_{avg} = 1	g_u = 4/3 g_d = 4/3 g_{avg} = 4/3	g_u = 5/3 g_d = 5/3 g_{avg} = 5/3
HI-skew	g_u = 5/6 g_d = 5/3 g_{avg} = 5/4	g_u = g_d = g_{avg} =	g_u = g_d = g_{avg} =
LO-skew	g_u = 4/3 g_d = 2/3 g_{avg} = 1	g_u = g_d = g_{avg} =	g_u = g_d = g_{avg} =

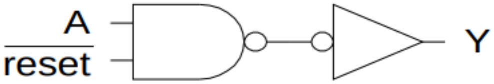
CATALOG OF SKEWED GATES

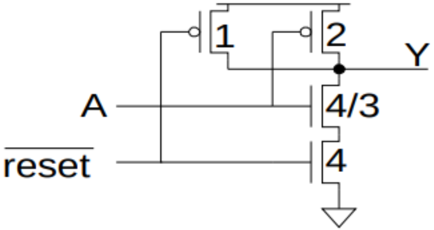


	Inverter	NAND2	NOR2
unskewed	$g_u = 1$ $g_d = 1$ $g_{avg} = 1$	$g_u = 4/3$ $g_d = 4/3$ $g_{avg} = 4/3$	$g_u = 5/3$ $g_d = 5/3$ $g_{avg} = 5/3$
HI-skew	$g_u = 5/6$ $g_d = 5/3$ $g_{avg} = 5/4$	$g_u = 1$ $g_d = 2$ $g_{avg} = 3/2$	$g_u = 3/2$ $g_d = 3$ $g_{avg} = 9/4$
LO-skew	$g_u = 4/3$ $g_d = 2/3$ $g_{avg} = 1$	$g_u = 2$ $g_d = 1$ $g_{avg} = 3/2$	$g_u = 2$ $g_d = 1$ $g_{avg} = 3/2$

ASYMMETRIC SKEW

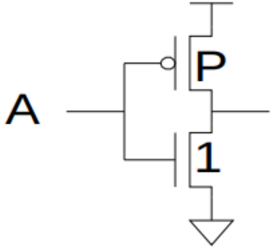
- Combine asymmetric and skewed gates
  - Downsize noncritical transistor on unimportant input
  - Reduces parasitic delay for critical input





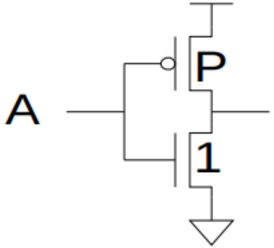
BEST P/N RATIO

- We have selected P/N ratio for unit rise and fall resistance (μ = 2-3 for an inverter).
- Alternative: choose ratio for least average delay
- Ex: inverter
    - ➢ Delay driving identical inverter
    - ➢ t_{pdf} =
    - ➢ t_{pdr} =
    - ➢ $t_{pd,m}$ =
    - ➢ Differentiate $t_{pd,m}$ w.r.t. P
    - ➢ Least delay for P =



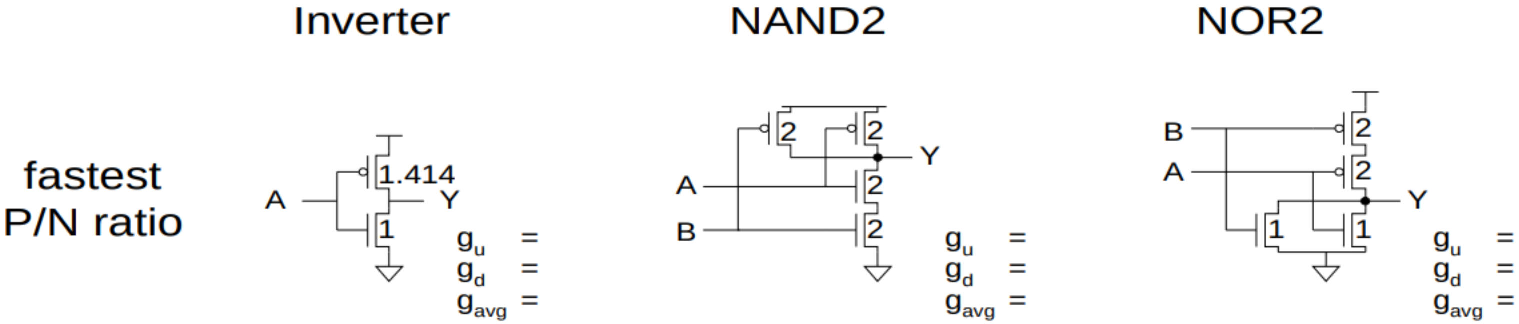
BEST P/N RATIO

- We have selected P/N ratio for unit rise and fall resistance (μ = 2-3 for an inverter).
- Alternative: choose ratio for least average delay
- Ex: inverter
  - ➢ Delay driving identical inverter
  - ➢ t_{pdf} = (P+1)
  - ➢ t_{pdr} = (P+1)(μ/P)
  - ➢ $t_{pd,m}$ = (P+1)(1+μ/P)/2 = (P + 1 + μ + μ/P)/2
  - ➢ Differentiate $t_{pd,m}$ w.r.t. P
  - ➢ Least delay for P = √μ



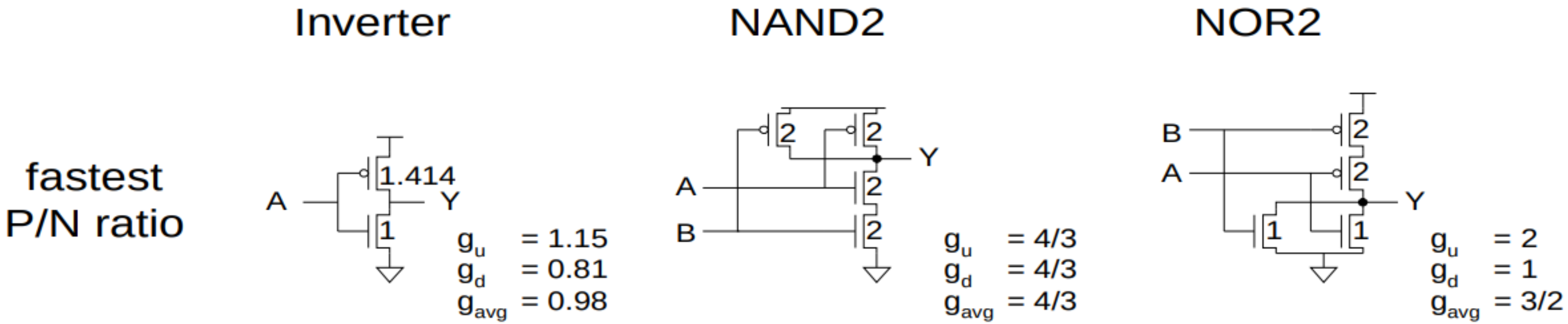
P/N RATIOS

- In general, best P/N ratio is sqrt of equal delay ratio
    - ➢ Only improves average delay slightly for inverters
    - ➢ But significantly decreases area and power



P/N RATIOS

- In general, best P/N ratio is sqrt of equal delay ratio
  - Only improves average delay slightly for inverters
  - But significantly decreases area and power



Inverter: $g_u = 1.15$, $g_d = 0.81$, $g_{avg} = 0.98$

NAND2: $g_u = 4/3$, $g_d = 4/3$, $g_{avg} = 4/3$

NOR2: $g_u = 2$, $g_d = 1$, $g_{avg} = 3/2$

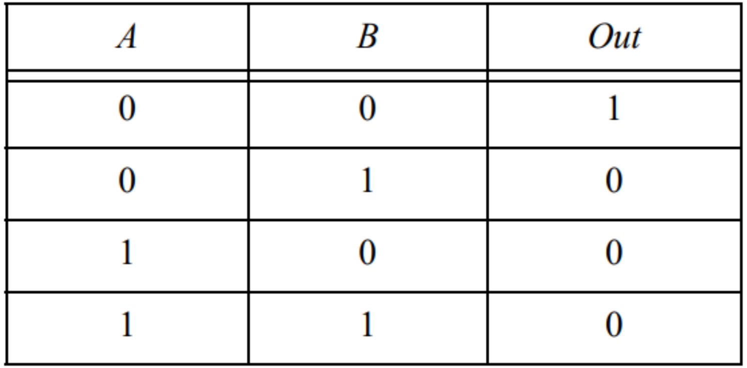
POWER CONSUMPTION IN CMOS LOGIC GATES

- Static, Dynamic and short-circuit power dissipation.
- Dynamic power consumption given by $\alpha_{0 \rightarrow 1} C_L V_{DD}^2 f$, where $\alpha_{0 \rightarrow 1}$ is the switching activity factor
- The transition activity is a strong function of the logic function being implemented.
- For static CMOS gates with statistically independent inputs, the static transition probability is the probability p_0 that the output will be in the zero state in one cycle, multiplied by the probability p_1 that the output will be in the one state in the next cycle:

$$\alpha_{0 \rightarrow 1} = p_0 \bullet p_1 = p_0 \bullet (1 - p_0)$$

2-INPUT NOR GATE SWITCHING ACTIVITY FACTOR

$$\alpha_{0 \rightarrow 1} = \frac{N_0}{2^N} \bullet \frac{N_1}{2^N} = \frac{N_0 \bullet \left(2^N - N_0\right)}{2^{2N}}$$

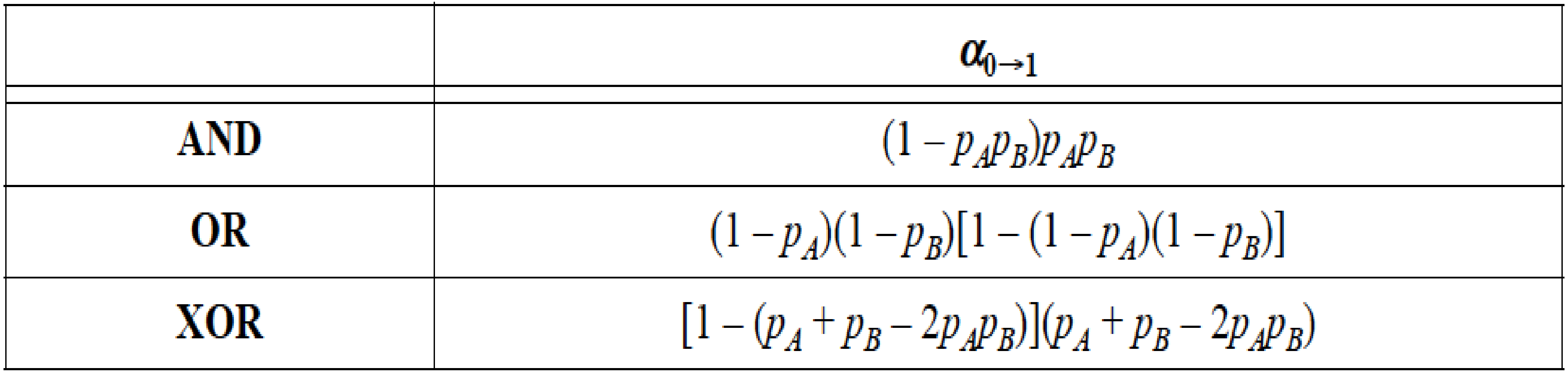


A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

$$\alpha_{0 \rightarrow 1} = \frac{N_0 \bullet \left(2^N - N_0\right)}{2^{2N}} = \frac{3 \bullet \left(2^2 - 3\right)}{2^{2 \bullet 2}} = \frac{3}{16}$$

TRANSITION PROBABILITY & SWITCHING ACTIVITY FACTOR

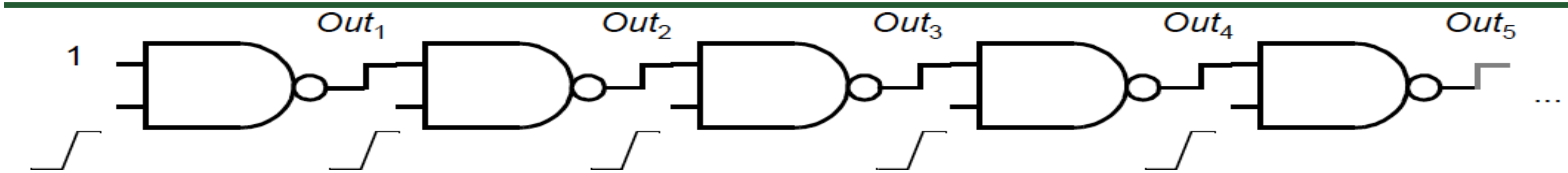
Output Transition Probabilities for Static Logic Gates

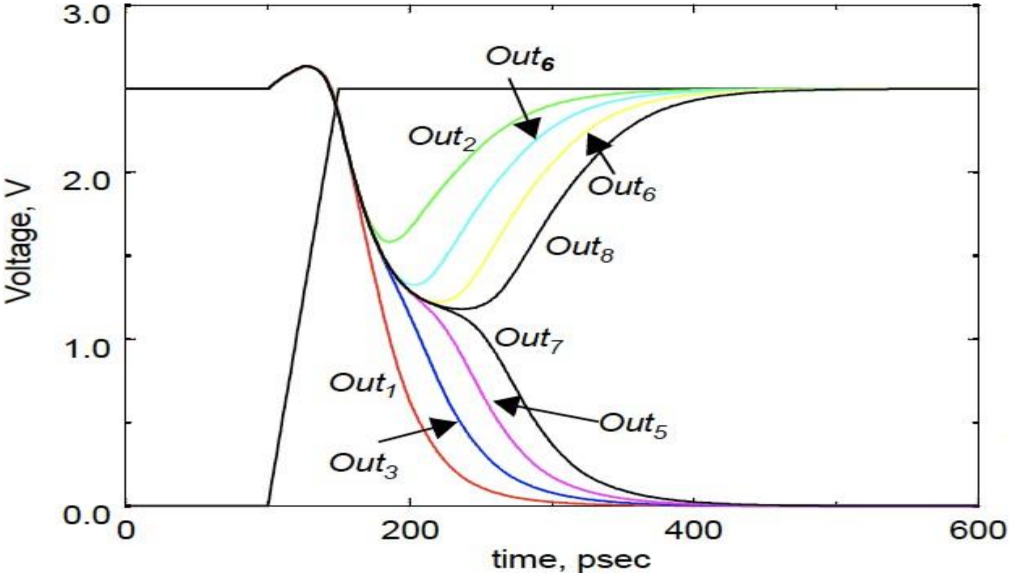


	$\alpha_{0\to1}$
AND	$(1-p_Ap_B)p_Ap_B$
OR	$(1-p_A)(1-p_B)[1-(1-p_A)(1-p_B)]$
XOR	$[1-(p_A+p_B-2p_Ap_B)](p_A+p_B-2p_Ap_B)$

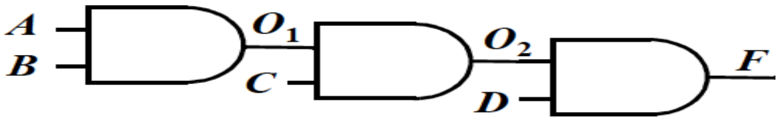
P_A and P_B are probabilities that input A and B are 1

GLITCHING IN A CHAIN OF NAND GATES

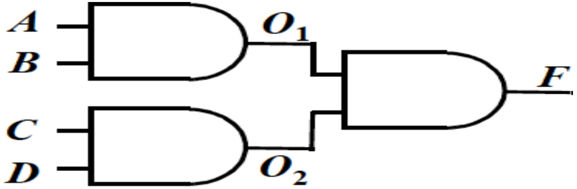




REDUCING SWITCHING ACTIVITY FACTOR



Chain structure



Tree structure

	O_1	O_2	F
p_1 (chain)	1/4	1/8	1/16
$p_0 = 1\text{-}p_1$ (chain)	3/4	7/8	15/16
$p_{0\text{->}1}$ (chain)	3/16	**7/64**	15/256
p_1 (tree)	1/4	1/4	1/16
$p_0 = 1\text{-}p_1$ (tree)	3/4	3/4	15/16
$p_{0\text{->}1}$ (tree)	3/16	**3/16**	15/256

OBSERVATIONS

- For speed:
  - NAND vs. NOR
  - Many simple stages vs. fewer high fan-in stages
  - Latest-arriving input
- For area and power:
  - Many simple stages vs. fewer high fan-in stages

CAL POLY
SAN LUIS OBISPO

Thank you!