Digital Integrated Circuit Design
CMPE 530/630

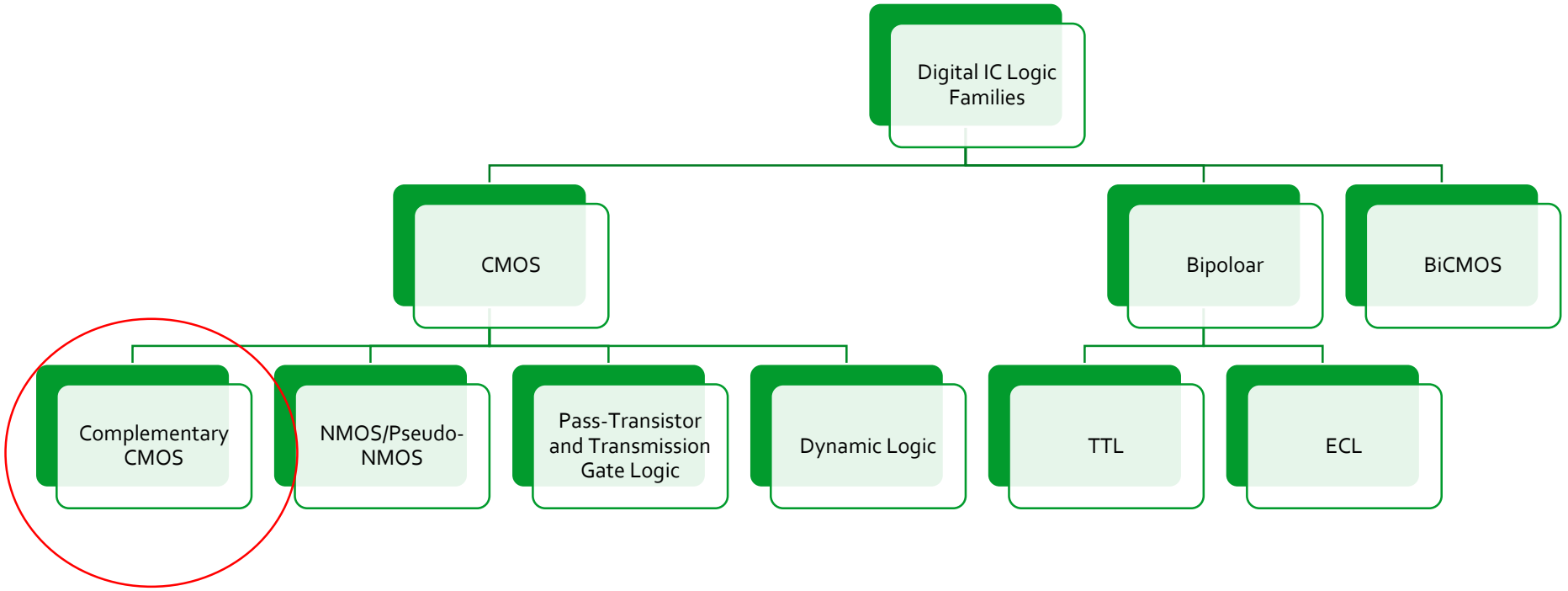
Combinational Circuit Design

Dr. Cory Merkel

Learning Objectives

- Discuss the pros and cons of different combinational circuit design approaches and logic families
- Be able to estimate and compare the logical effort, parasitic delay, and overall delay of different design approaches
- Identify some of the common pitfalls encountered by designers when designing combinational circuits

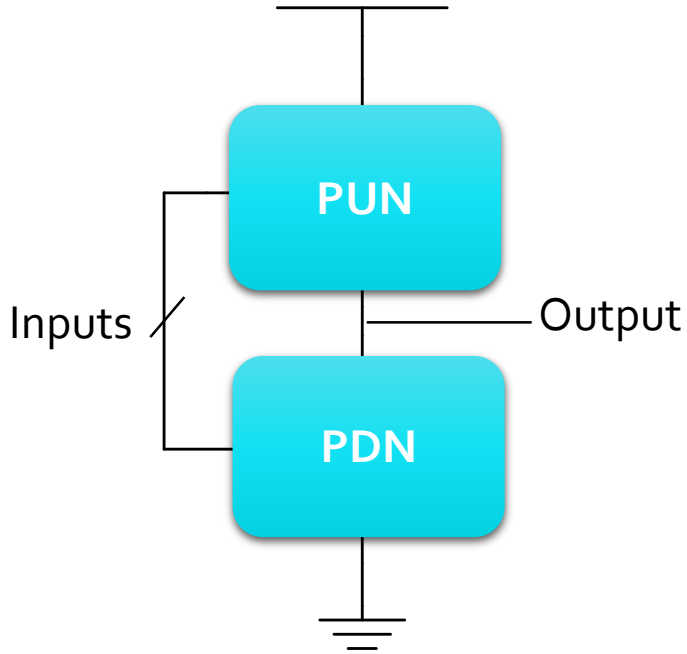
Logic Styles/Families



Primary focus in this course.

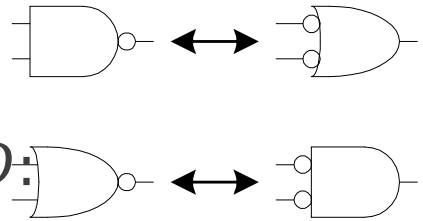
Complementary/Static CMOS

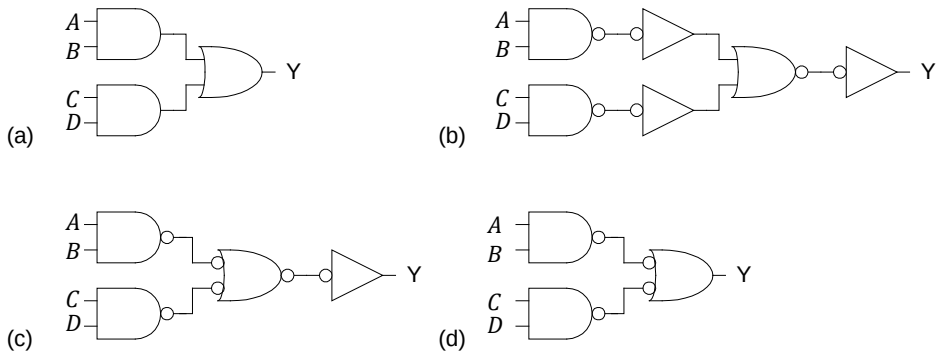
- Pros
  - Good noise margins
  - Fast
  - Low power
  - Good variation tolerance
  - Easy to design
  - Most widely-used style for combinational logic
- Cons
  - Not always the fastest or most area-efficient
- There are a few ways we can optimize static CMOS circuits



Complementary/Static CMOS

- Bubble pushing
  - Idea: Convert AND-OR logic to NAND-NOR logic using DeMorgan's Law
    - Reduces transistor count
- $\overline{AB} = \overline{A} + \overline{B}$
- $\overline{A + B} = \overline{A}\,\overline{B}$
- Consider the logic function $Y = AB + CD$:



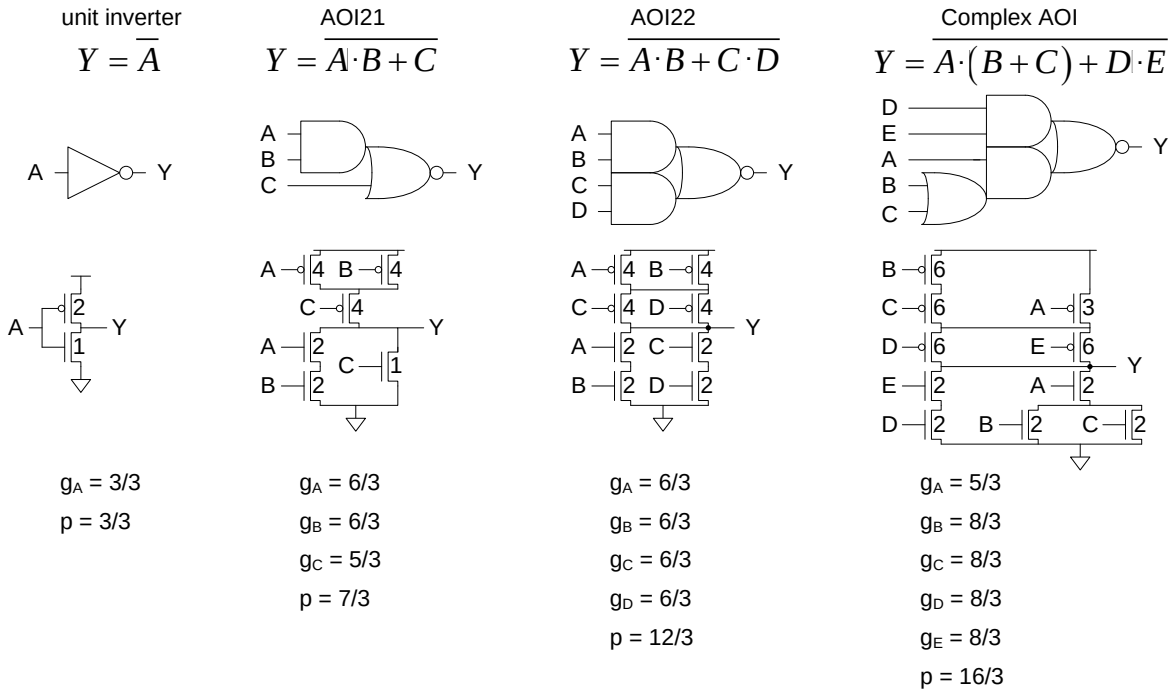


(a) Has 18 transistors

(d) Has 12 transistors

Complementary/Static CMOS

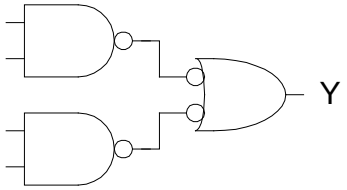
- Compound gates
  - Instead of using multiple NAND/NOR/Inverter gates, we can combine a complex logic function into a single gate
  - Whereas NAND and NOR have the same logical effort for every input, compound gates can have different logical efforts for each input:



Complementary/Static CMOS

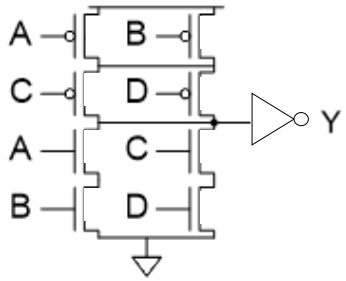
- Which is faster? Network of NAND/NOR or compound gate?
- $Y = AB + CD$
  - Assume each input has 20 units of capacitance and the output drives 100 units of capacitance

NAND/NOR



$$H = \frac{100}{20} = 5$$
$$B = 1$$
$$G = \frac{4}{3} \times \frac{4}{3} = \frac{16}{9}$$
$$P = 2 + 2 = 4$$
$$F = GBH = \frac{80}{9}$$
$$\hat{f} = F^{1/N} = 3$$
$$D = N\hat{f} + P = 10$$
$$C_{in2} = \frac{C_{out2} \times g_2}{\hat{f}} = \frac{100 \times 4/3}{3} = 44$$

Compound

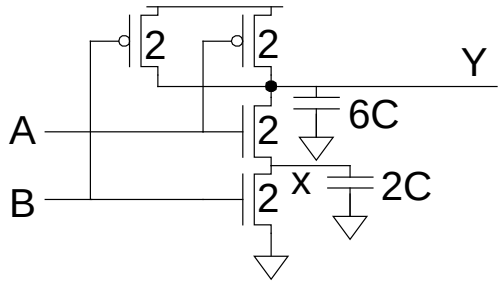


$$H = \frac{100}{20} = 5$$
$$B = 1$$
$$G = \frac{6}{3} \times 1 = 2$$
$$P = 12/3 + 1 = 5$$
$$F = GBH = 10$$
$$\hat{f} = F^{1/N} = 3.2$$
$$D = N\hat{f} + P = 11.3$$
$$C_{in2} = \frac{C_{out2} \times g_2}{\hat{f}} = \frac{100 \times 4/3}{3} = 31$$

Compound gates aren't necessarily fastest!

Complementary/Static CMOS

- Gate input ordering
  - Different inputs/transitions have different logical efforts and parasitic delays associated with them
  - Even simple gates like NAND have slight differences in g and p for different inputs:
- Case 1: $A = 1, B: 0 \rightarrow 1$
  - Node x is initially at $V_{dd} - V_{tn}$
  - $t_{pf} = \left(\frac{R}{2}\right) 2C + R(6C) = 2.33\tau$
- Case 2: $B = 1, A: 0 \rightarrow 1$
  - Node x is initially at o
  - $t_{pf} = R(6C) = 2\tau$



Complementary/Static CMOS

» Logical effort and parasitic delay of NAND gates in IBM 65 nm process:
  - Found by extracting slope and intercept of $d = gh + p$

# of inputs	Input	Rising Logical Effort g_u	Falling Logical Effort g_u	Average Logical Effort g	Rising Parasitic Delay p_u	Falling Parasitic Delay p_d	Average Parasitic Delay p
2	A	1.40	1.12	**1.26**	2.46	2.48	**2.47**
	B	1.31	1.16	**1.24**	1.97	1.82	**1.89**
3	A	1.76	1.27	**1.51**	4.77	4.10	**4.44**
	B	1.73	1.32	**1.52**	3.93	3.60	**3.77**
	C	1.59	1.38	**1.48**	3.05	2.43	**2.74**
4	A	2.15	1.42	**1.78**	7.63	5.94	**6.79**
	B	2.09	1.48	**1.78**	6.67	5.37	**6.02**
	C	2.08	1.53	**1.80**	5.32	4.51	**4.91**
	D	1.90	1.59	**1.75**	4.04	2.93	**3.49**

» *Outer input* – Closer to supply rail

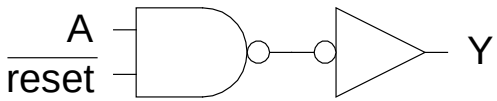
» *Inner input* – Closer to output

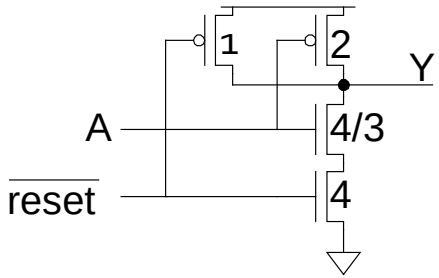
» Parasitic delay is lowest when inner input switches last

» Signals that arrive 'late' should be connected to inner inputs to minimize delay

Complementary/Static CMOS

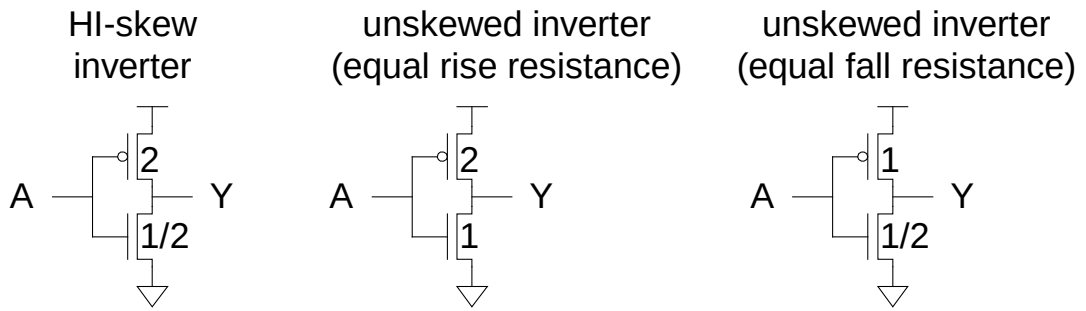
- Asymmetric gates
  - Idea:
    - 1.) Connect critical input(s) to inner input
    - 2.) In series networks, reduce width of critical input to reduce logical effort
    - 3.) Increase width of other transistors in series to maintain overall resistance
    - 4.) Reduce width of non-critical parallel transistors to reduce parasitic capacitance
- Example:
  - Logical effort is 10/9 for A, which is better than the typical NAND g of 4/3
  - Reset input logical effort is higher (5/3)





Complementary/Static CMOS

- Skewed gates
  - Used when one *transition* (rising or falling) is more important than the other
  - HI-skew – Favors rising output
    - Decrease size of PDN transistors
  - LO-skew – Favors falling output
    - Decrease size of PUN transistors
  - Degree of skewing = β_f/β_s, where β_f and β_s are the fast and slow transitions, respectively
  - Noise margin are affected
- Example:

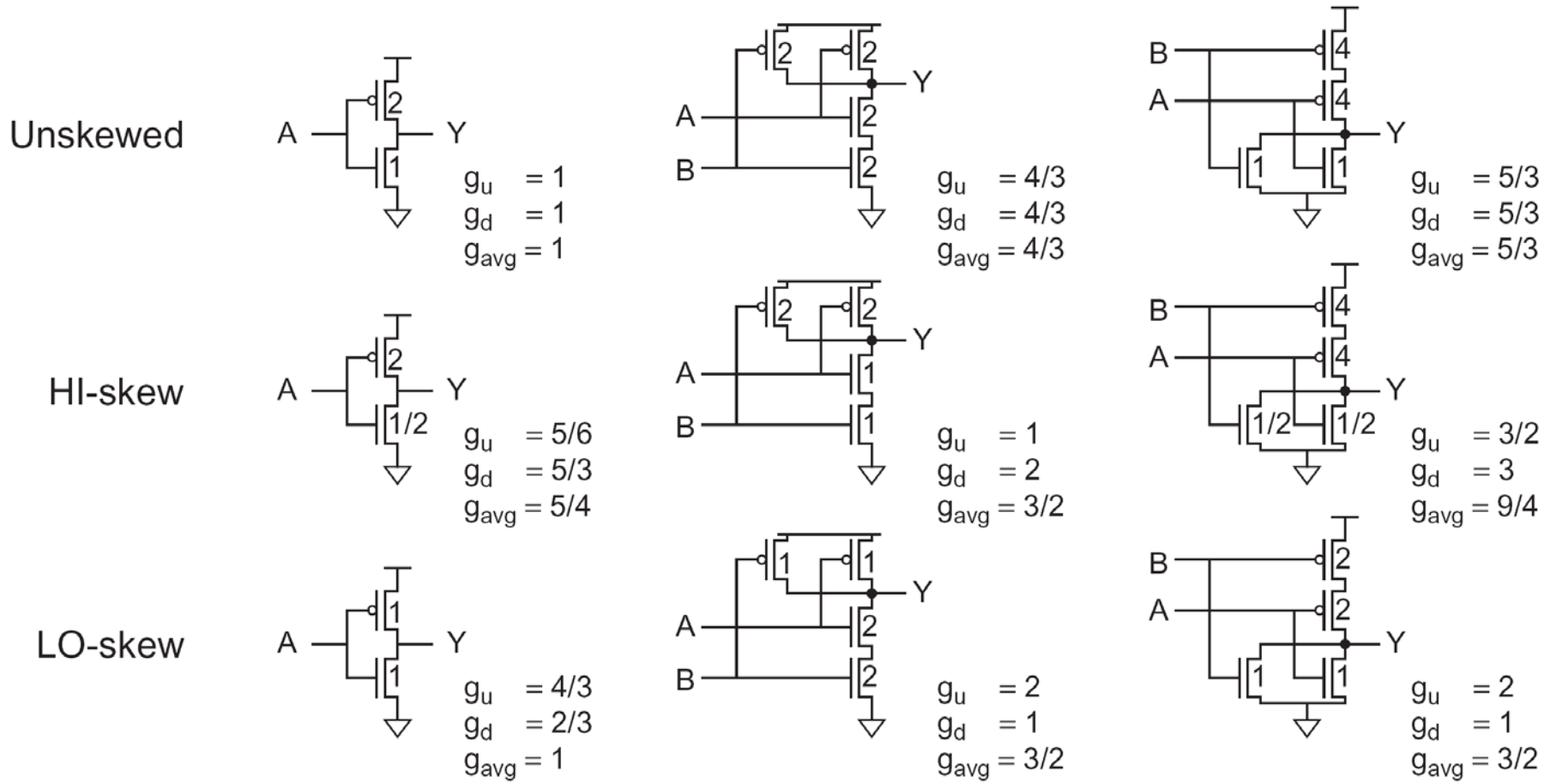


Logical effort for rising transition: $g_u = \frac{2.5}{3} = 5/6$

Logical effort for falling transition: $g_d = \frac{2.5}{1.5} = 5/3$

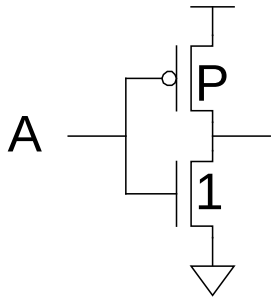
Complementary/Static CMOS

» More examples of skewed gates:



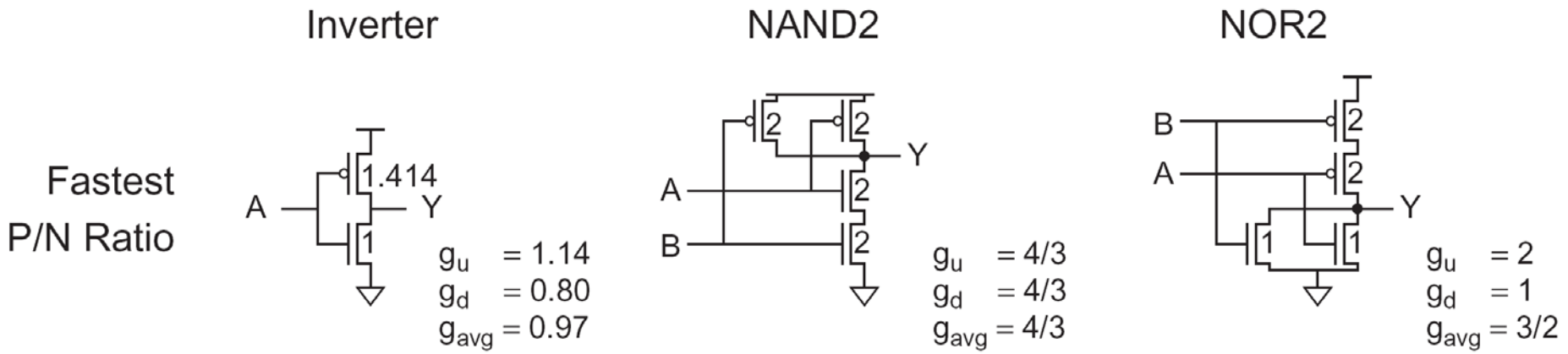
Complementary/Static CMOS

- What is the best ratio of P and N sizes?
  - So far we have chosen based on equal rise/fall delay
- We could optimize by choosing based on best *average* delay
- Delay of FO1 inverter:
  - $\mu \equiv \mu_n/\mu_p$
  - $t_{pdf} = (P + 1)$
  - $t_{pdr} = (P + 1)(\mu/P)$
  - $t_{pd} = (P + 1)(1 + \mu/P)/2 = (P + 1 + \mu + \mu/P)/2$
  - $\mathrm{d}t_{pd} / \mathrm{d}P = (1 - \mu/P^2)/2 = 0$
  - Least average delay for $P = \sqrt{\mu}$
- Delay not affected much, but power and area are significantly reduced
- Cons: Noise margins, asymmetric transitions



Complementary/Static CMOS

» CMOS gates with optimal P/N ratio:

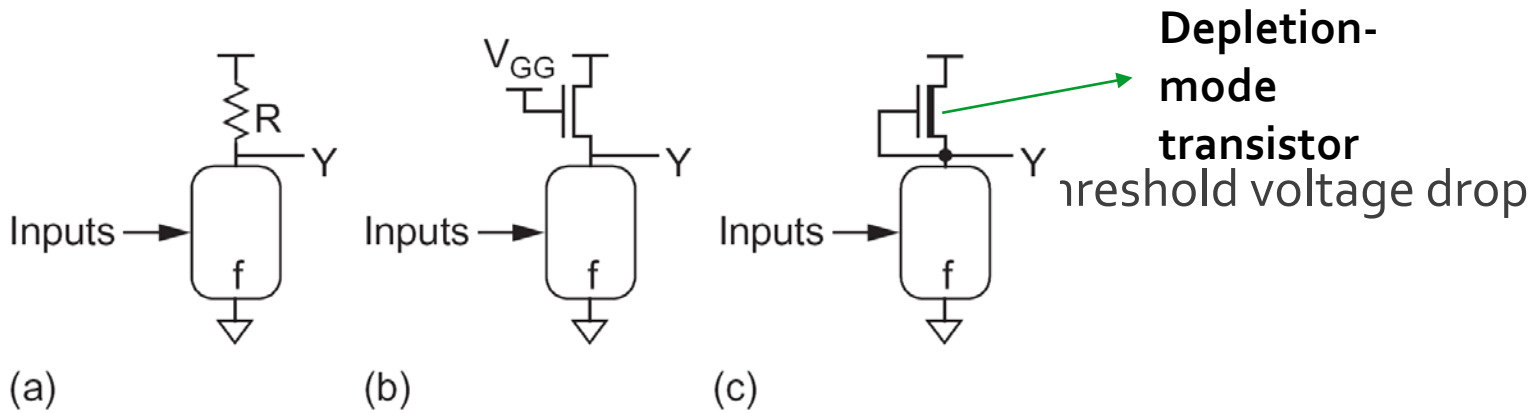


» NOR structures affected more

Ratioed Circuits

- NMOS logic
  - Primary approach in 70's and 80's before CMOS matured
  - Static load → Constant static power consumption for high output
    - PUN/PDN ratio must be carefully set for proper rise/fall times, noise margins, etc.
  - CMOS eventually displaced NMOS logic due to high static power consumption

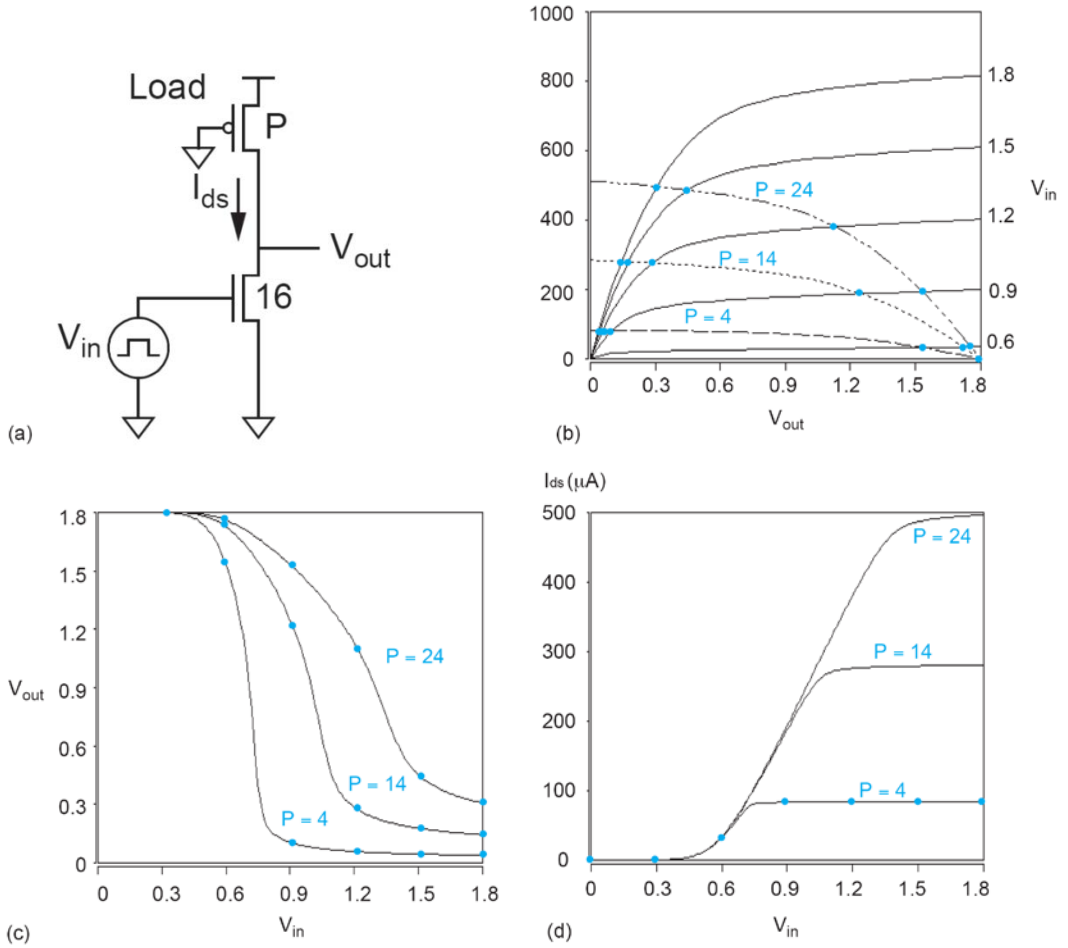
- (a) and (c) use components exacerbated by body effect



…reshold voltage drop

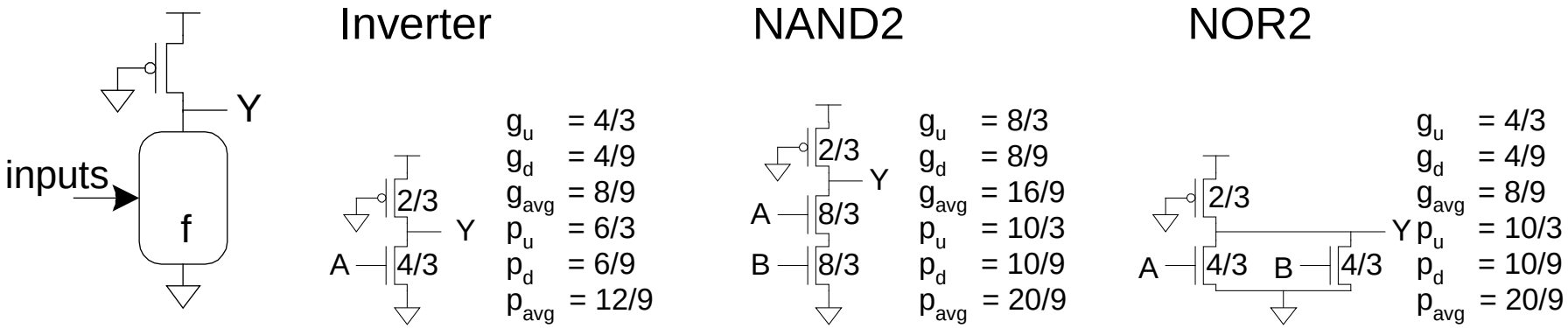
Ratioed Circuit

- Pseudo-nMOS
  - Motivation, remove large PMOS transistors in PUN to reduce delay/power
- Use grounded PMOS as load
  - Size affects speed, noise margins, power
- Best size is process-dependent
  - Usually ~1/3-1/6 of minimum NMOS size



Ratioed Circuits

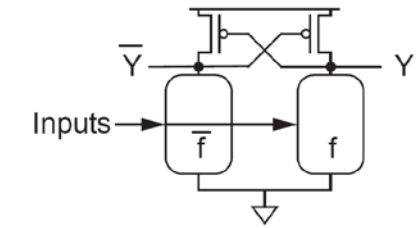
- Pseudo-nMOS gates:
  - PMOS sizes are selected to be about ¼ strength of NMOS
  - PMOS produces current of $I/3$, NMOS produces $4I/3$



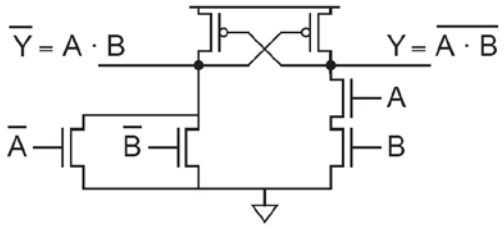
- Remember, logical effort and parasitic delay are ratios of input/parasitic capacitance to those of a unit inverter *that can provide the same output current*
- Pseudo-nMOS is slower for NAND structures and faster for NOR structures

Cascode Voltage Switch Logic

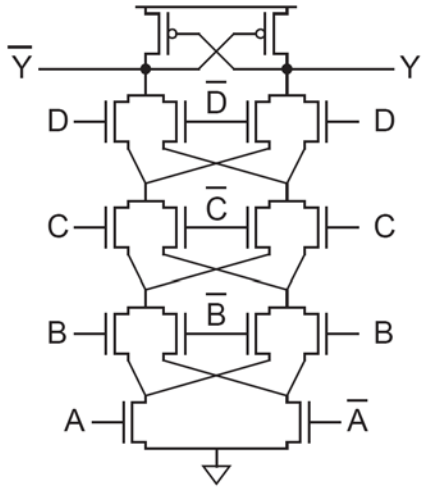
» CVSL
  - Advantage of ratioed logic without static power consumption



(a)



(b)



(c)

» Disadvantages:
  - Need true and complement of all inputs
  - Still usually slower than static CMOS

Dynamic Logic

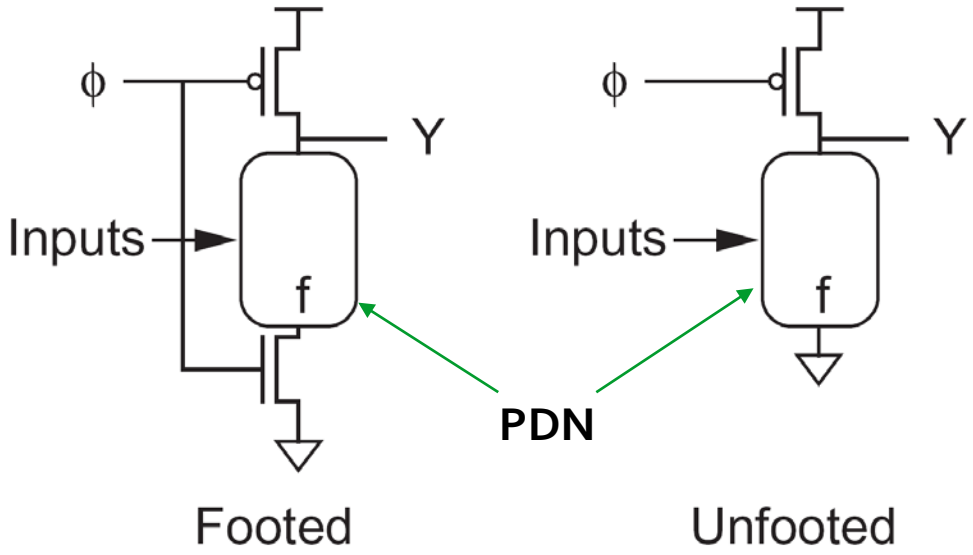
» Idea: Advantage of ratioed logic (i.e. reduced input cap. from PMOS) without:
- Contention during falling transition
- Slow rise times due to high PUN resistance

» Accomplished by clocking the PUN:

1.) Precharge phase:
- $\phi = 0$
- Output charges to V_{dd}

2.) Evaluate phase:
- $\phi = 1$
- Output remains or falls depending on inputs

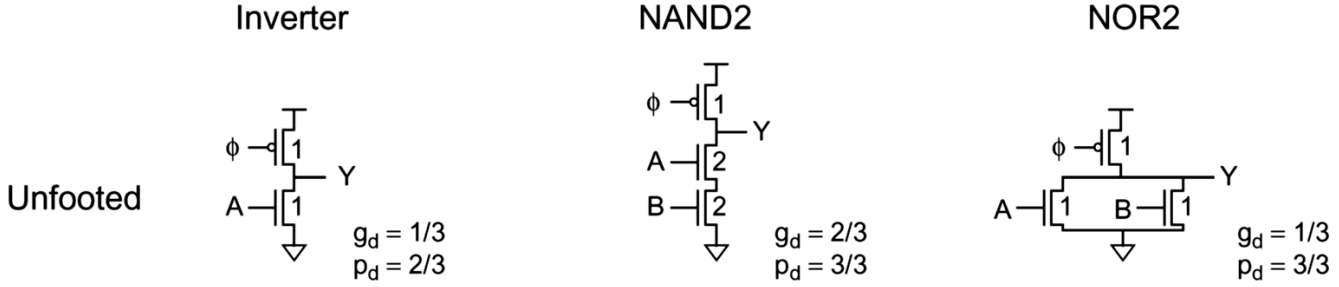
» Fastest commonly-used circuit family



Footer transistor added when input can't be guaranteed to be o during pre-charge.

Dynamic Logic

- Precharge transistor is sized for $2 \times R$
  - Reduces capacitive load on clock
  - Reduces parasitic delay
- Reduced logical effort and parasitic delay compared to static CMOS and ratioed logic because:
  - Smaller PUN **AND** no contention during transition

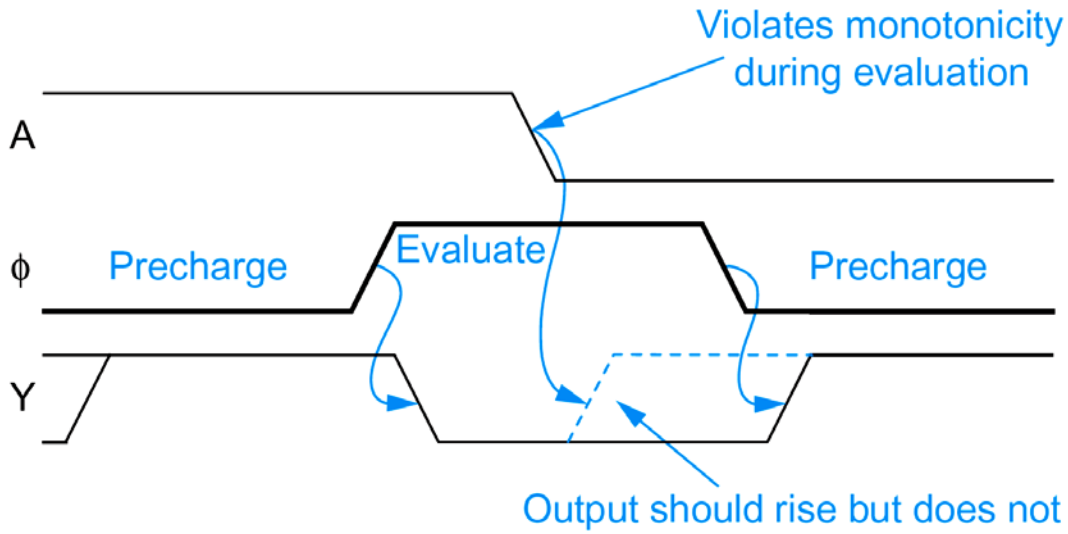


Static CMOS	$g_d = 1$ $p_d = 1$	$g_d = 4/3$ $p_d = 2$	$g_d = 5/3$ $p_d = 2$

Dynamic Logic

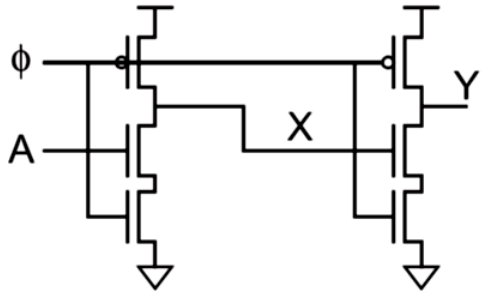
» Monotonicity problem
  - During evaluation, the inputs to a dynamic logic gate must be *monotonically rising*
    - 0→0, 0→1, 1→1 are ok, 1→0 is not!

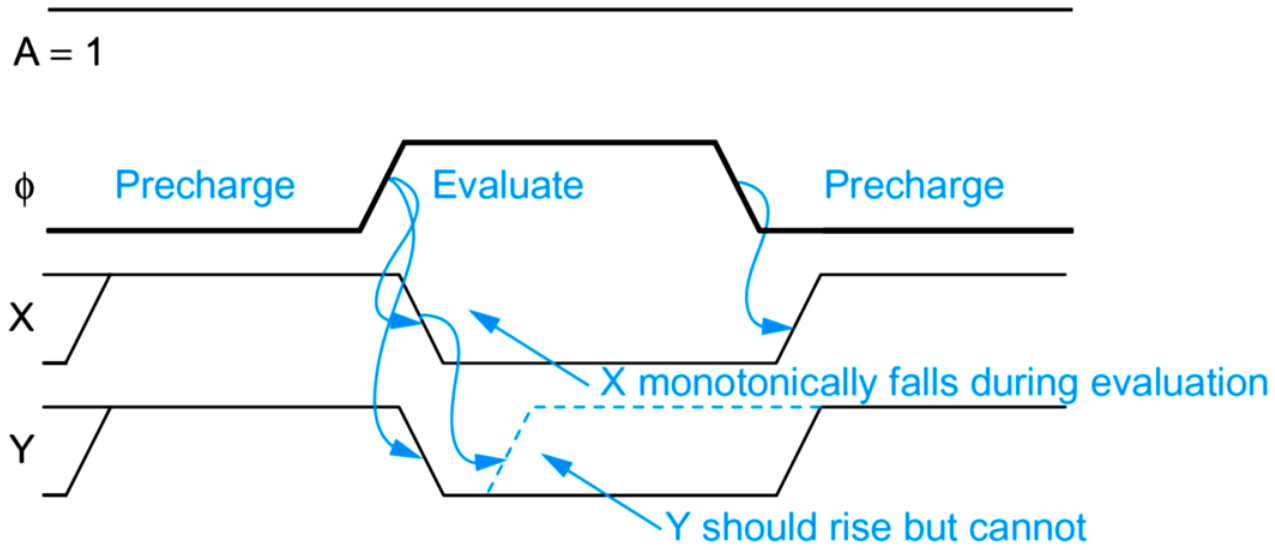
» Example:
  - Dynamic logic inverter violating monotonicity:



Dynamic Logic

- Monotonicity can be violated when domino gates are connected
- Example:
  - This dynamic logic buffer will not work correctly

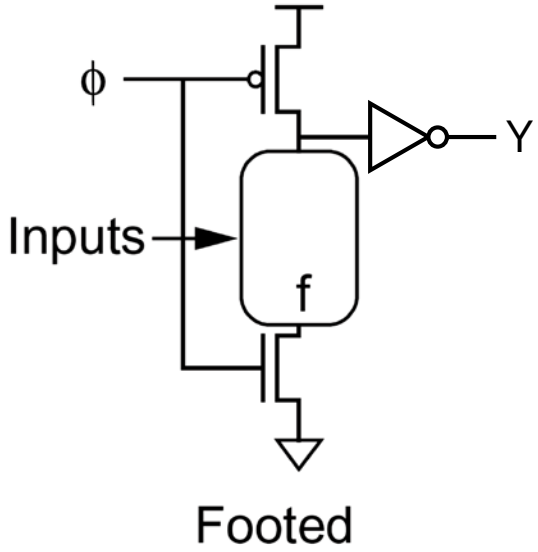




Domino Logic

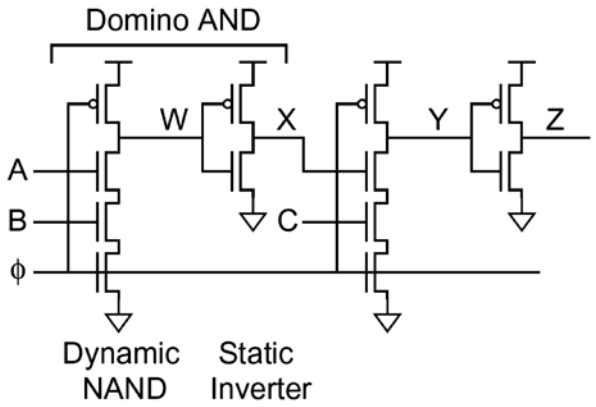
- Domino logic solves the monotonicity problem by forcing every gate output to be monotonically rising
  - Achieved by adding a static CMOS inverter

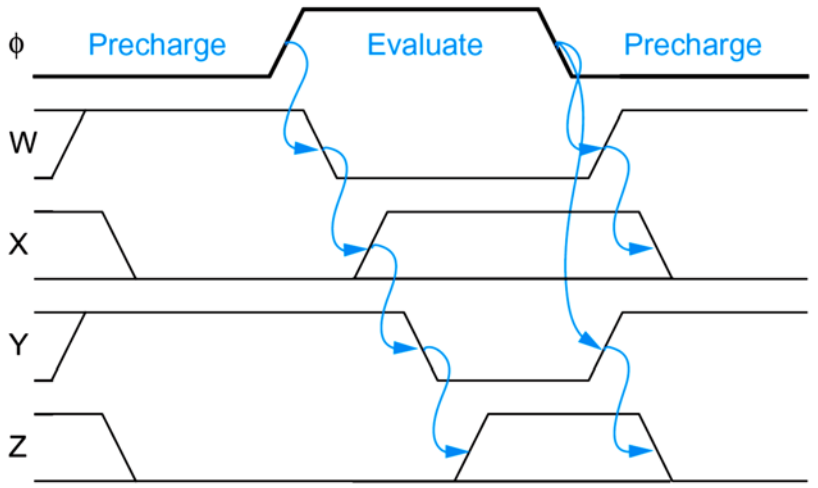
- Inverter is usually HI-skew since rising transition is most important
  - Falling transition of a series of gates occurs in parallel, while the rising transition occurs sequentially

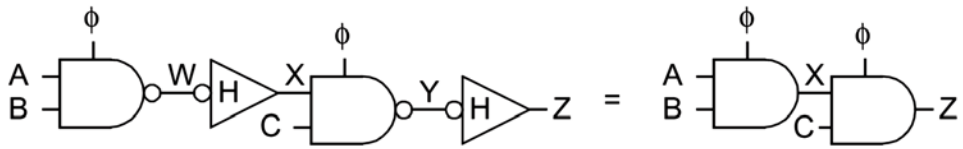


Domino Logic

» Example:







» During pre-charge, outputs are set up like dominos, and then fall, one-by-one

Domino Logic

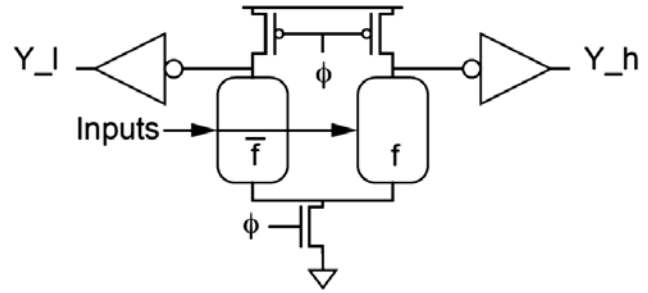
- Notice that domino logic can only implement non-inverting functions
  - e.g. AND, OR, buffer
- We can't directly implement inverting functions with domino logic
  - e.g. NAND, NOR, XOR, NOT
- Three techniques to overcome this:
  - Use static logic for inverting sub-functions
  - Directly cascade gates without CMOS inverter and delay clock to later stages
  - Use dual-rail domino logic

Dual-Rail Domino Logic

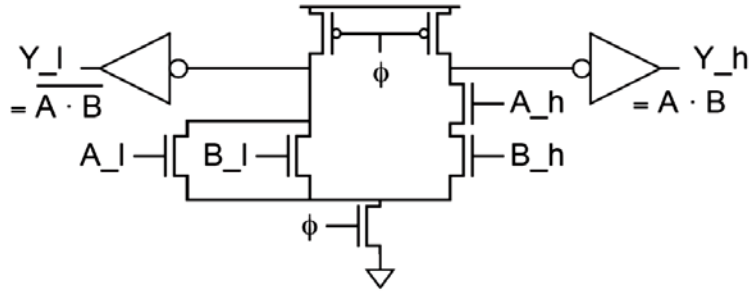
- Similar to CVSL
  - Instead of cross-coupled PMOS, they are connected to clock

sig_h	sig_l	Meaning
0	0	Precharged
0	1	'0'
1	0	'1'
1	1	Invalid

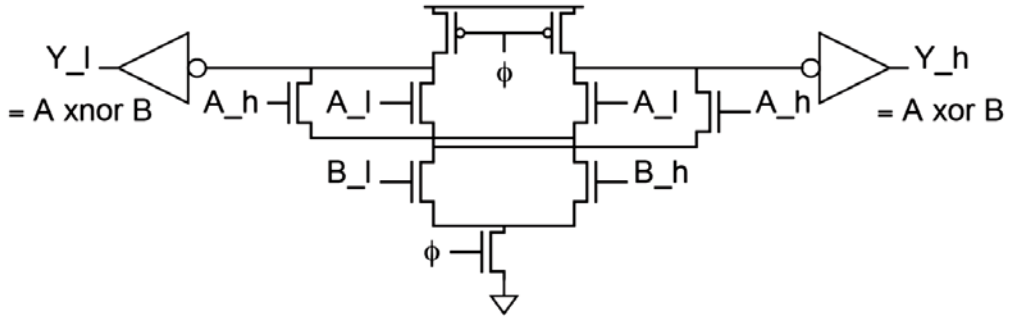
- NAND gate can be connected to outputs to detect end of computation



(a)



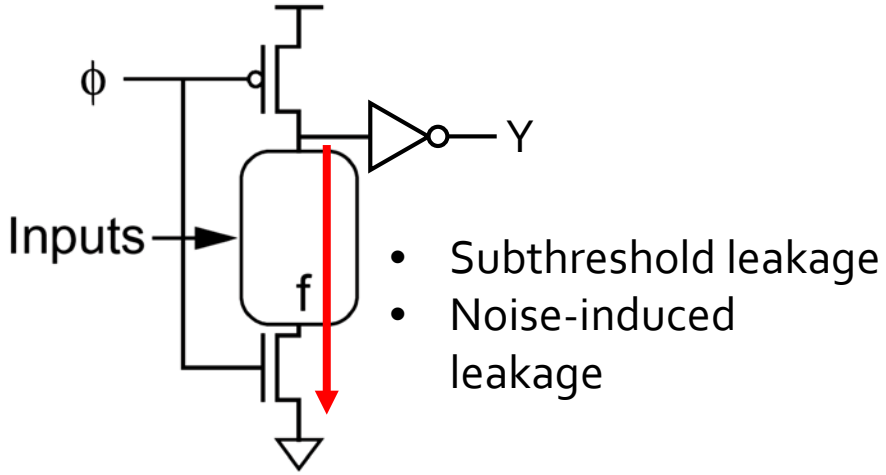
(b)



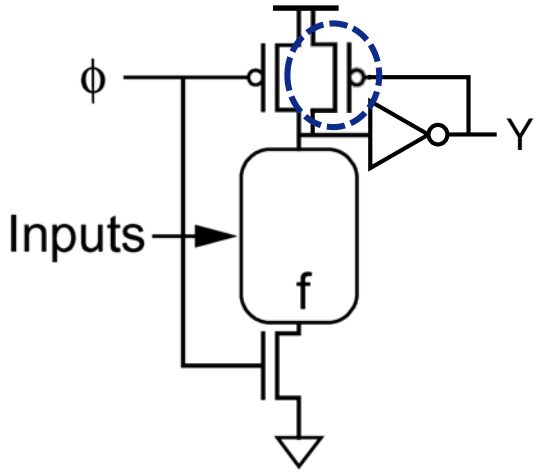
(c)

Dynamic Logic – Charge Keepers

» Due to leakage and/or noise, charge may leak from the dynamic node of a dynamic logic gate:
  - The dynamic node may fall to '0' even if the inputs would normally keep it at '1'

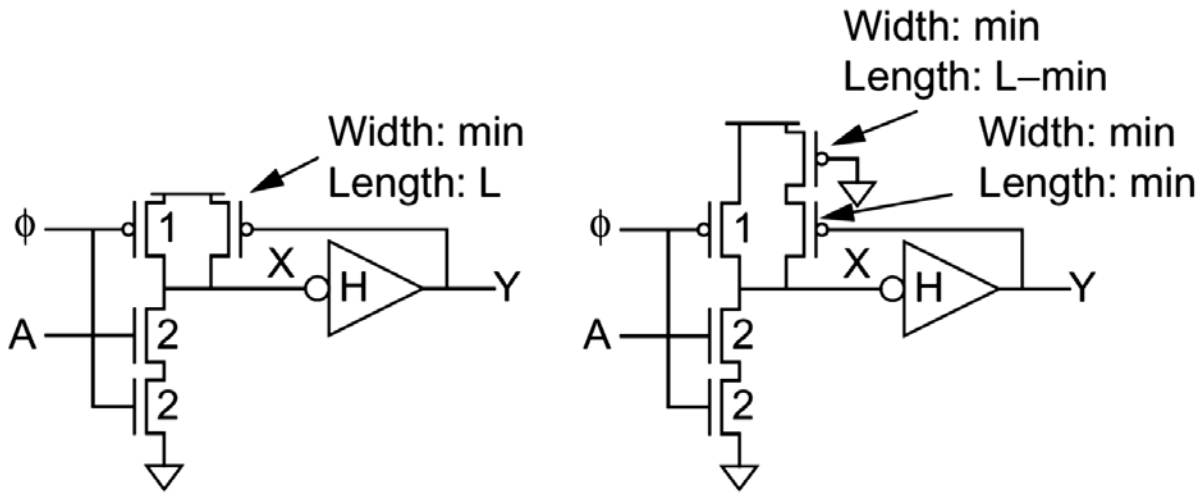


» One solution is to add a 'charge keeper'
  - Keeps charge on dynamic node unless inputs cause it to discharge



Dynamic Logic – Charge Keepers

- Charge keepers need to be sized carefully
  - Too strong
    - Large contention current, output may not be able to switch
  - Too weak
    - Can't counteract effect of leakage/noise
  - Generally, should be weaker than minimally-sizes transistor
    - Minimum width, increased length

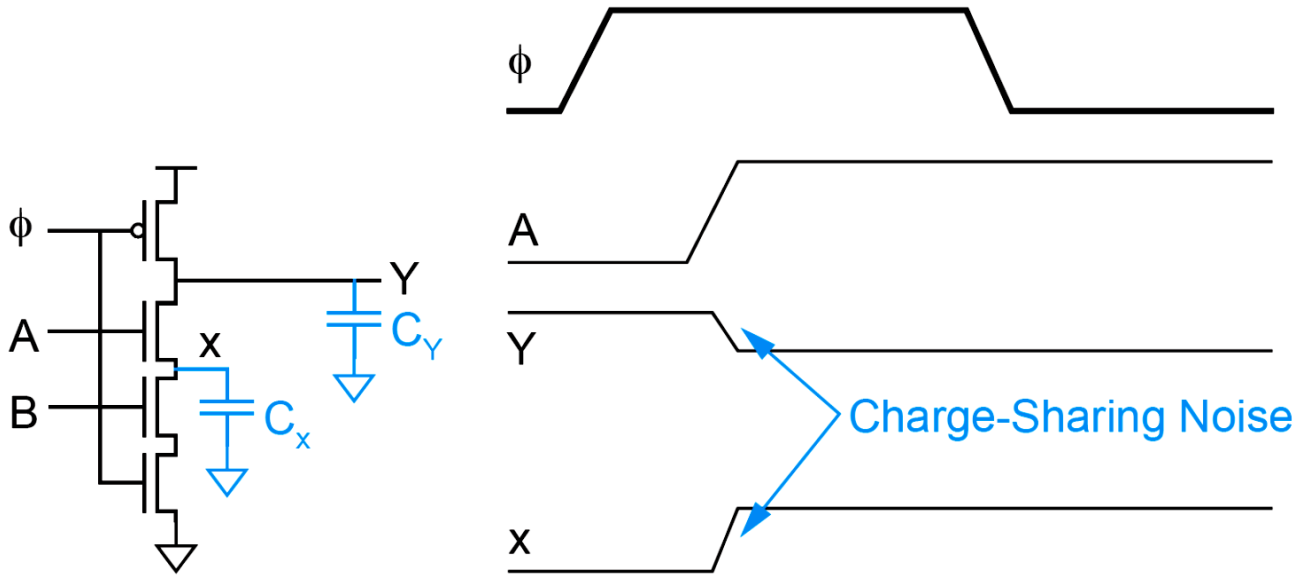


Dynamic Logic – Charge Sharing

» Consider the dynamic NAND gate below, where
- Initially node x is 0, A=B=0, Y is precharged
- During evaluation, A=1, B=0
  - Charge will move from capacitor C_y to C_x until their voltage is the same:

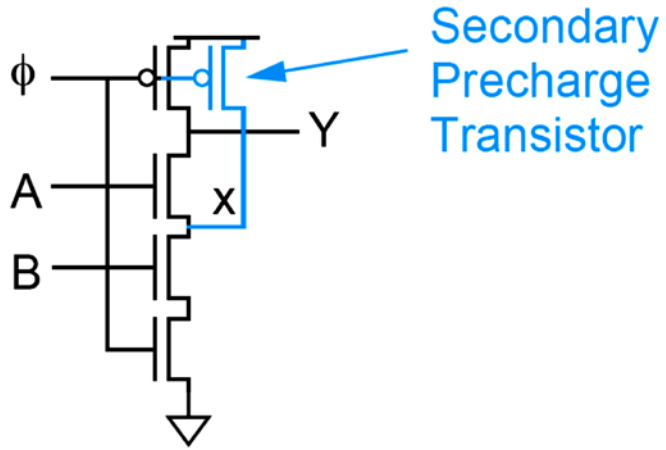
$$V_x = V_y = \frac{C_y}{C_x + C_y} V_{dd}$$

If $C_x \gg C_y$, then V_y may fall below a legal '1' value.



Dynamic Logic – Charge Sharing

» Solutions
- Make sure that inputs are high during precharge
  - May require a lot of logit
- Add 'secondary' precharge transistors to ensure all internal nodes are charges:



» Adverse effect:
- Falling transition becomes slower because more charge has to discharged to ground

Multiple Output Domino Logic

» Consider the functions

$$c_1 = g_1 + p_1 c_0$$
$$c_2 = g_2 + p_2(g_1 + p_1 c_0)$$
$$c_3 = g_3 + p_3\big(g_2 + p_2(g_1 + p_1 c_0)\big)$$
$$c_4 = g_4 + p_4(g_3 + p_3(g_2 + p_2(g_1 + p_1 c_0)))$$

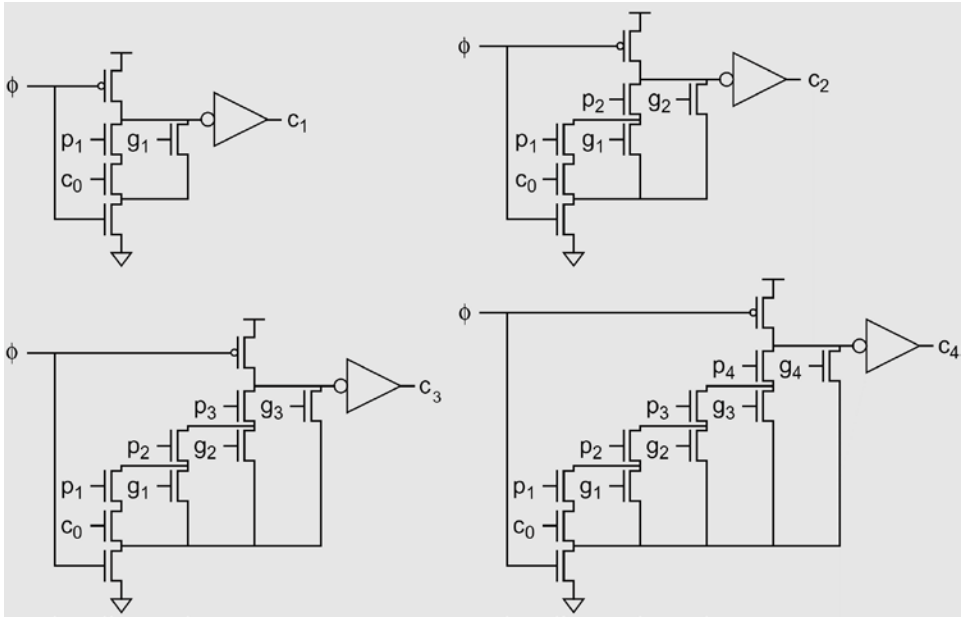
» Some functions contain "subfunctions"

- e.g. $g_1 + p_1 c_0$ is a subfunction that occurs in all of them

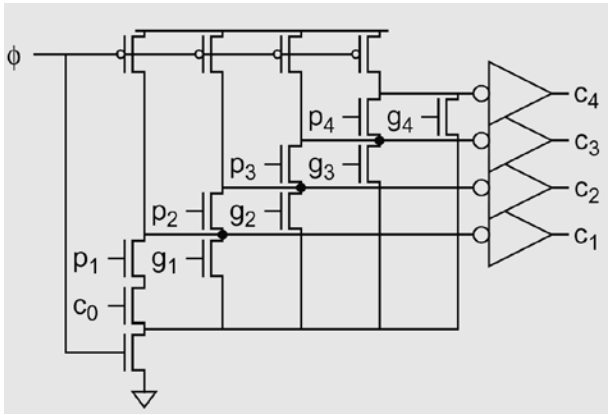
» Can we re-use subfunctions to reduce area/power?

Multiple Output Domino Logic

One domino gate for each function:



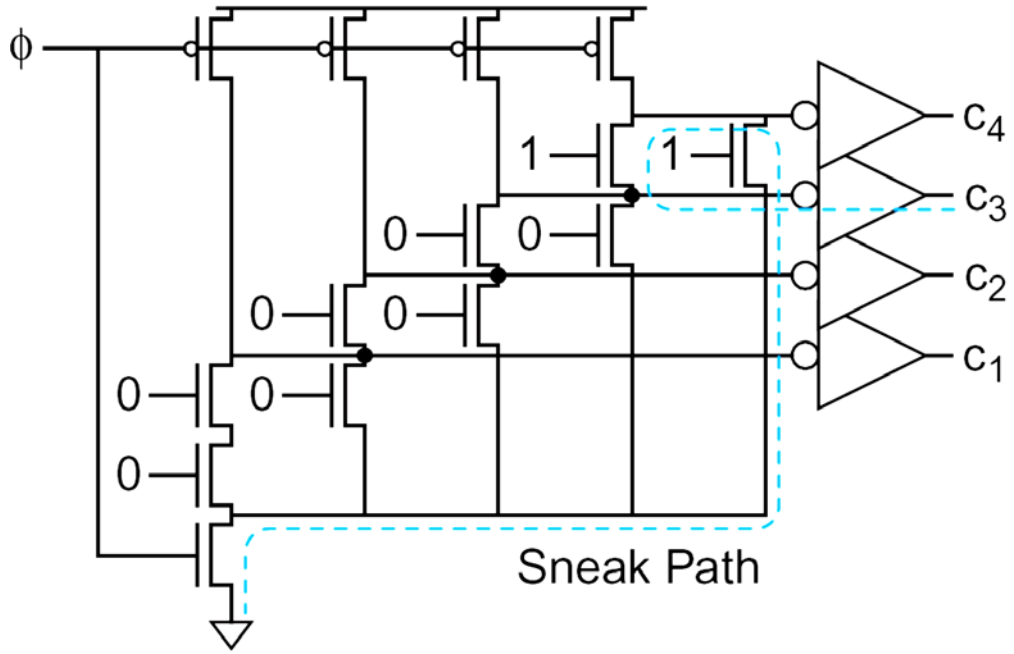
Multiple output domino logic:



- Precharge transistor for each subfunction
- "Sneak paths" can occur for some input combinations
  - Need to make some inputs mutually exclusive to avoid

Multiple Output Domino Logic

» Sneak path example:



» The two '1' inputs should be mutually exclusive to avoid the sneak path

Different Circuit Family Implementations of 2-input MUXes