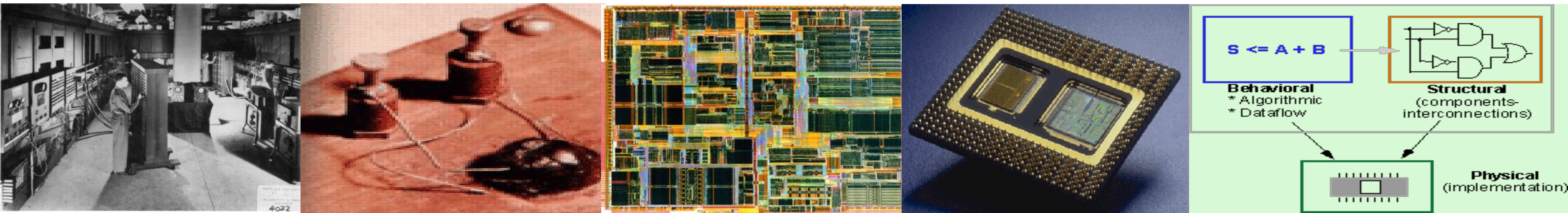
CPE 690: Introduction to VLSI Design

Lecture 11
Combinational Logic

Bryan Ackland
Department of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, NJ 07030

Adapted from Lecture Notes, David Mahoney Harris CMOS VLSI Design



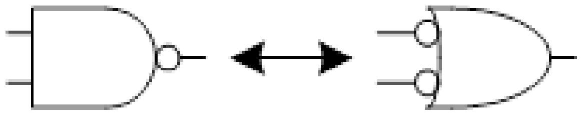
Combinational Logic Families

- Combinational circuits: outputs depend only on current inputs (no memory)
- CMOS Combinational Circuit Families:
  - Static (compound) gates
  - Ratio’ed CMOS gates
  - Dynamic CMOS gates
  - Pass Transistor Logic
- Compound Gates: complimentary N and P networks that ensure gate is always driven high or low (but not both)
- Techniques to optimize compound gates
  - Bubble pushing
  - Input ordering
  - Asymmetric gates
  - Skewed gates

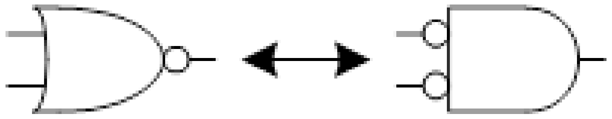
Bubble Pushing

- Logic traditionally expressed in terms of AND & OR
- CMOS compound gates are always inverting
  - NAND, NOR, INV etc.
- "Push bubbles" around to reformat logic expressions in form amenable to CMOS compound gates
- DeMorgan's Law:

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

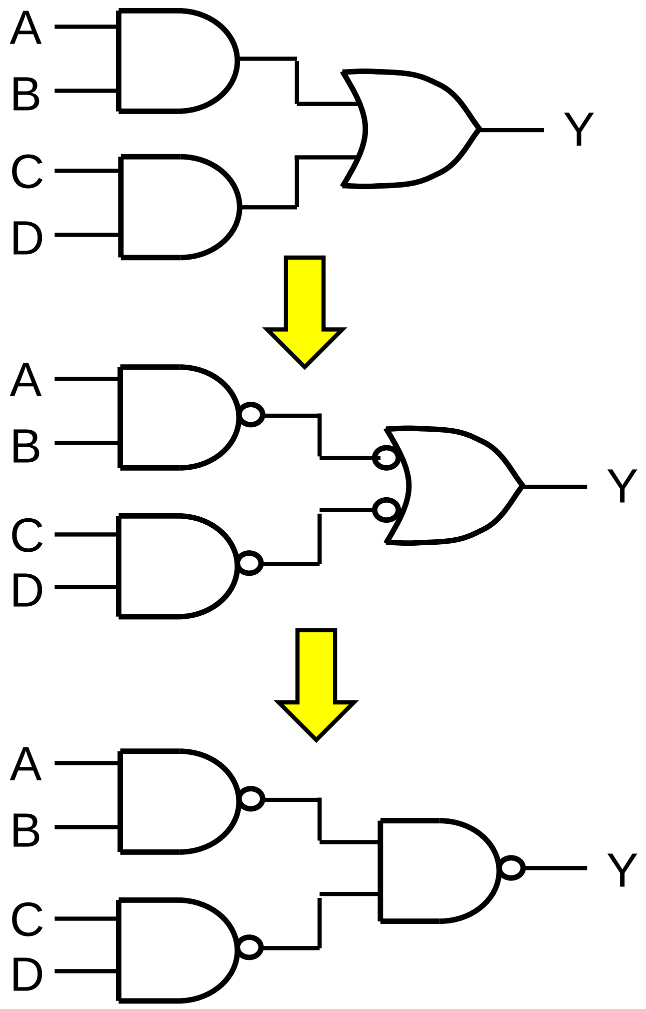


$$\overline{A+B} = \bar{A}.\bar{B}$$



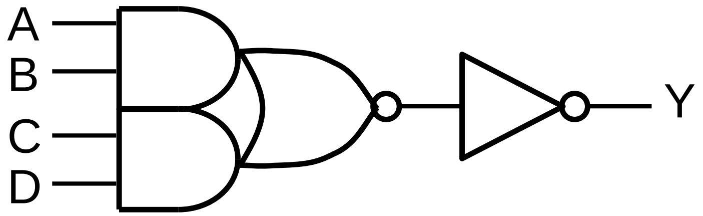
Example: AOI22

- $Y = A.B + C.D$
  - used frequently as 2-input multiplexer: $Y = \bar{S}.I_0 + S.I_1$

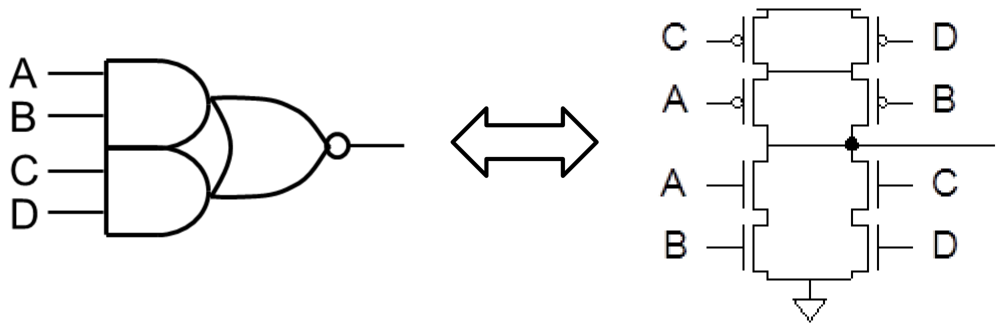


Example: Alternate Solution

- *Y = A.B + C.D*
- Implement as a single-stage compound gate plus inverter:



where

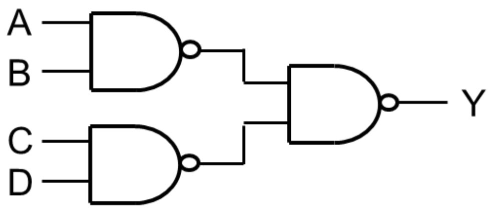


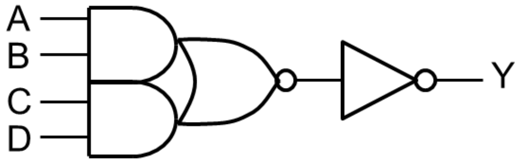
- *Which implementation is better?*

Example: Compare delays

- Suppose the *Y=A.B + C.D* function must drive a load of 160 units of capacitance and is limited to a C_{in} of 16 units of capacitance on each input.

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





$$P = 2 + 2 = 4$$

$$G = {}^{4}/_{3} \times {}^{4}/_{3} = {}^{16}/_{9}$$

$$F = G.B.H = {}^{160}/_{9}$$

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

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

$$P = 4 + 1 = 5$$

$$G = 2 \times 1 = 2$$

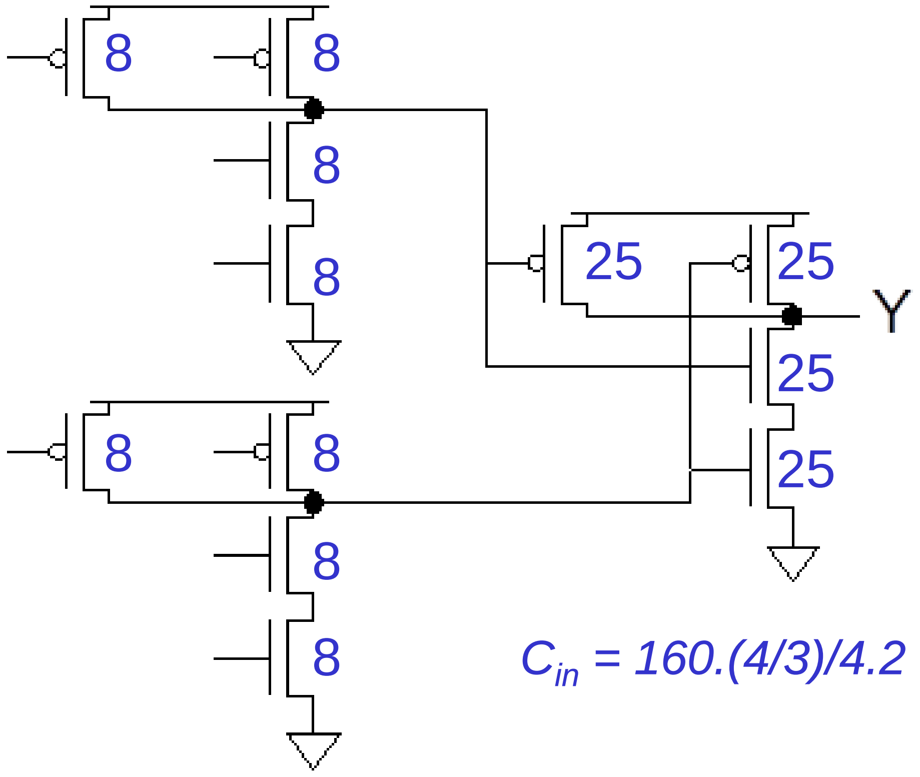
$$F = G.B.H = 20$$

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

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

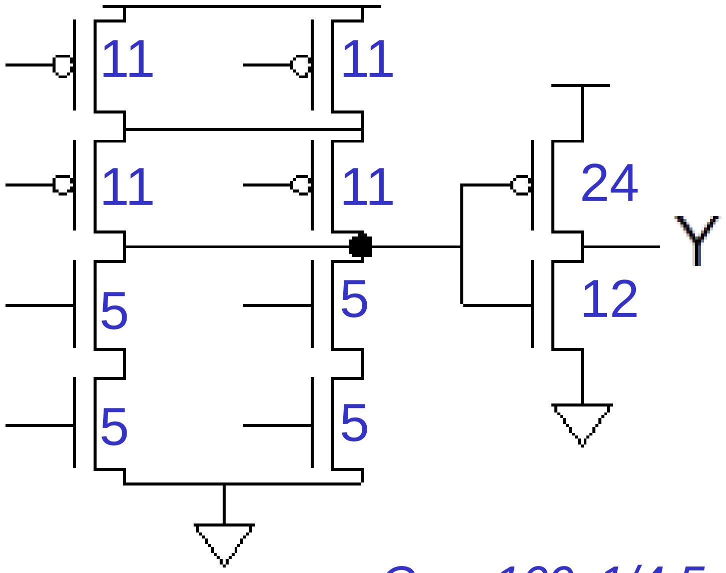
Example: Determine Transistor Sizes

C_{in} = 50.(4/3)/4.2 = 16



C_{in} = 160.(4/3)/4.2 = 50

C_{in} = 36x2/4.5 = 16



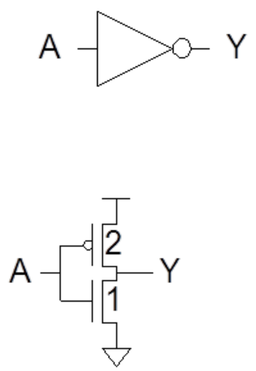
C_{in} = 160x1/4.5 = 36

Logical Effort of Compound Gates

- In general, logical effort of compound gate depends on which input path is passing through:

unit inverter

$$Y = \overline{A}$$

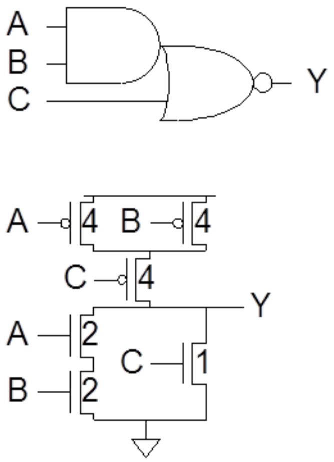


$g_A = 3/3$

$p = 3/3$

AOI21

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



$g_A = 6/3$

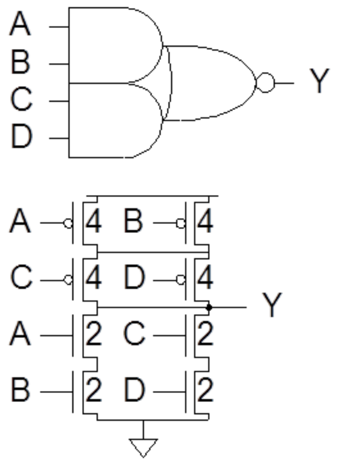
$g_B = 6/3$

$g_C = 5/3$

$p = 7/3$

AOI22

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



$g_A = 6/3$

$g_B = 6/3$

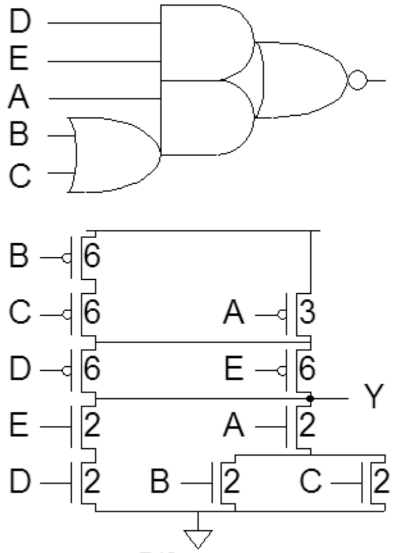
$g_C = 6/3$

$g_D = 6/3$

$p = 12/3$

Complex AOI

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



$g_A = 5/3$

$g_B = 8/3$

$g_C = 8/3$

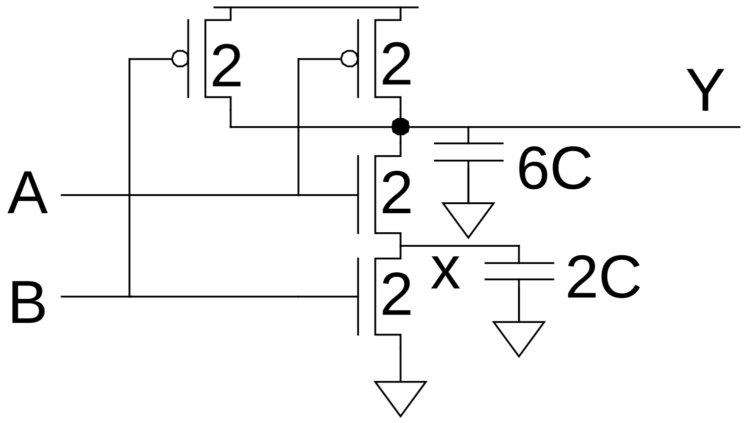
$g_D = 8/3$

$g_E = 8/3$

$p = 16/3$

Input Ordering Delay

- When using *logical effort*, our parasitic delay model only accounted for *capacitance* on output node
- Recall that *Elmore delay* allows us to account for *capacitance* on intermediate nodes
  - then nominally symmetric gates (NAND, NOR etc) will show different parasitic delays at different inputs
- Calculate NAND2 parasitic (Elmore) delay for Y falling
  - If B arrives latest? $p = (R/2)(2C)+R(6C) = 7RC = \mathbf{2.33\tau}$
  - If A arrives latest? $p = R(6C) = 6RC = \mathbf{2\tau}$

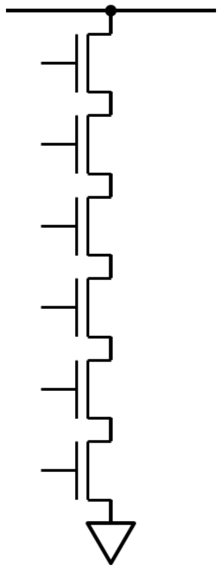


Inner vs. Outer Inputs

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

- Effect is more pronounced with higher fan-in gates
- *e.g.,* a NAND6 has parasitic delays:

6τ	(innermost)
7.7τ	
9τ	
10τ	
10.7τ	
11τ	(outermost)

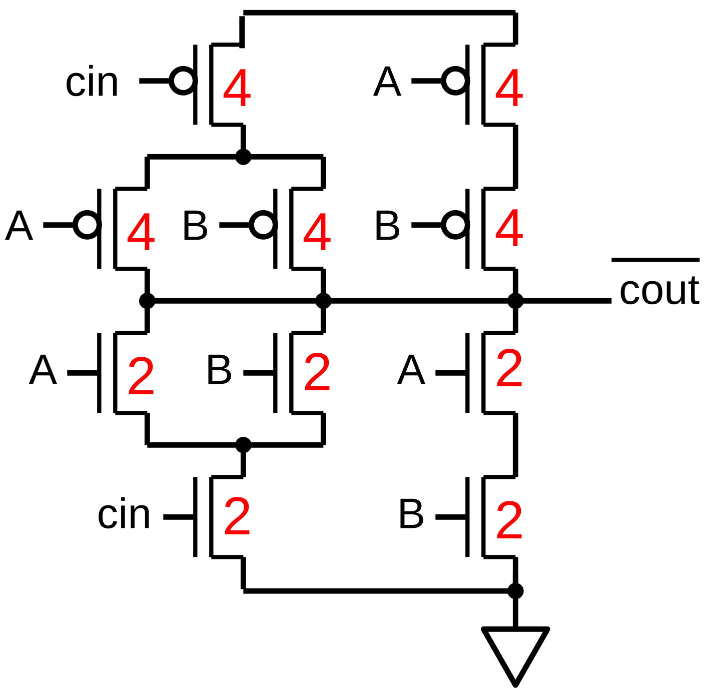


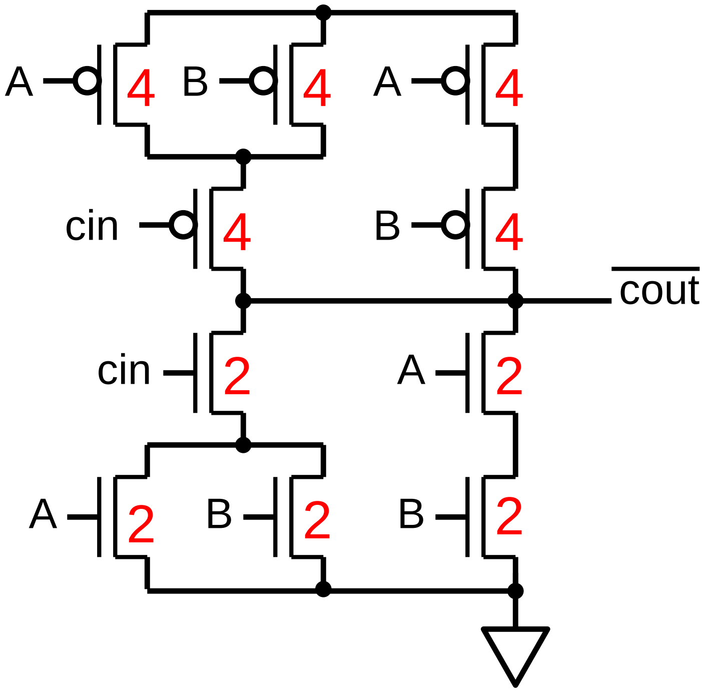
almost 2:1 variation!

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

Example: Carry Ripple Delay

- Need to minimize delay from *cin* to *cout*
- $\overline{cout} = \overline{a.b + cin(a+b)}$





$p_d = (R/2)(6C)+R(18C)$

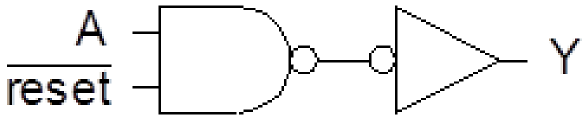
$= 21RC = \mathbf{7\tau}$

$p_u = 24RC = \mathbf{8\tau}$

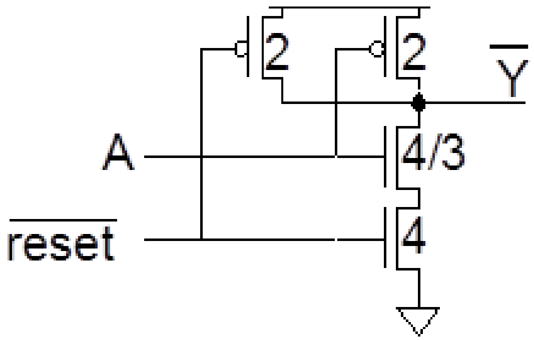
$p_d = p_u = R(12C) = 12RC = \mathbf{4\tau}$

Asymmetric Gates

- In addition to choosing innermost gate, we can change relative size of inner and outer transistors
- Ex: suppose input A of a NAND gate is most critical



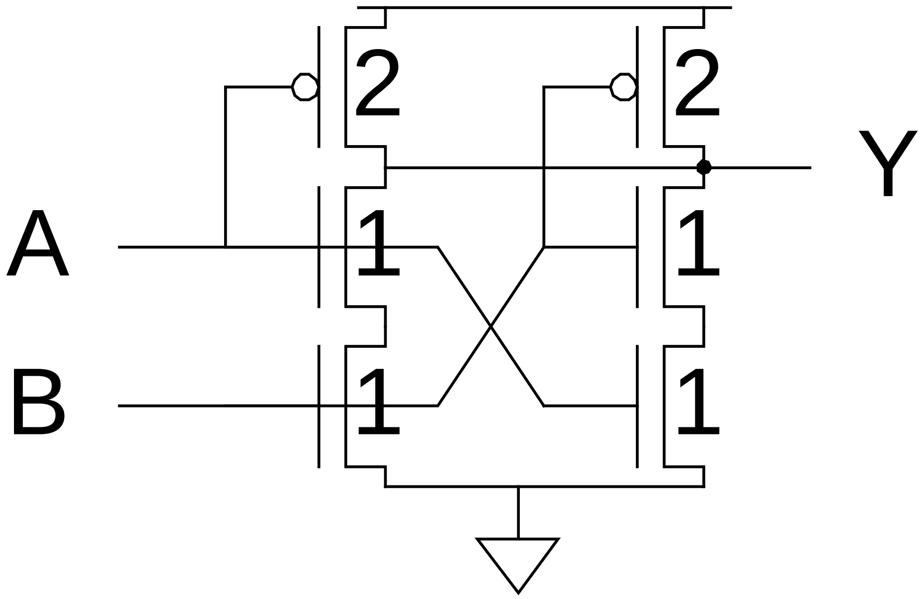
- Use smaller transistor on A (less capacitance)
  - Boost size of noncritical input so total resistance is same



- $g_A = 10/9$ (*normally NAND2 is* 4/3)
- $p_A = 16/9$ *(normally 2)*
- $g_{reset} = (6/3) = 2$, $p_{reset} = 19/9$
- $g_{avg} = (g_A + g_{reset})/2 = 14/9$ (*normally 12*/9)
- As asymmetry increases, g ⇨ 1 on critical input
  - at expense of much greater delay on non-critical input

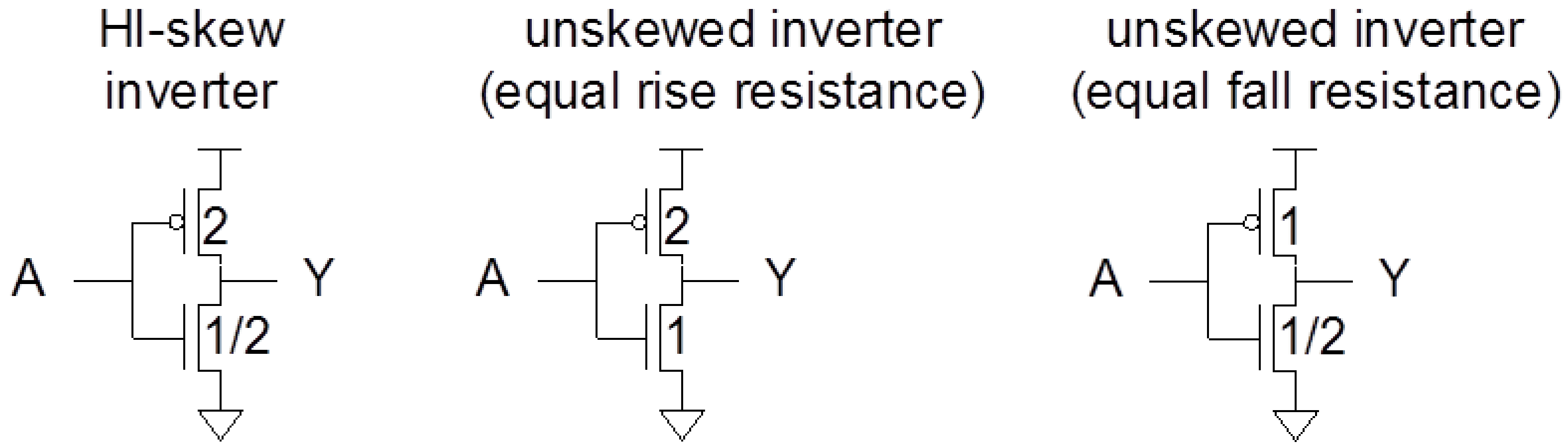
Symmetric Gates

- Can we build a perfectly symmetric gate?



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 un-skewed inverter with same effective resistance on that edge.
  - $g_u = 2.5 / 3 = 5/6$
  - $g_d = 2.5 / 1.5 = 5/3$

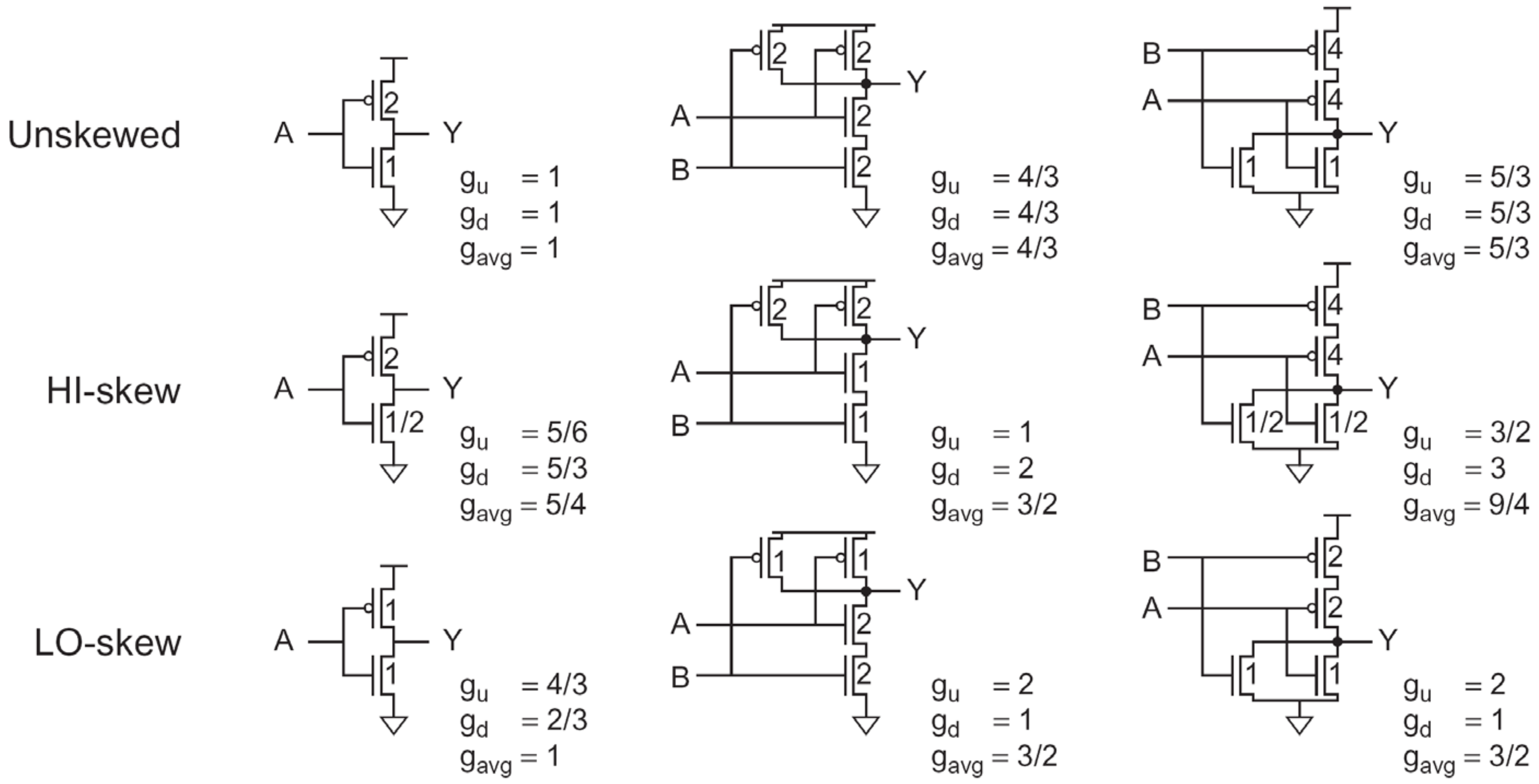
HI and LO Skew

- Define: 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 un-skewed 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)

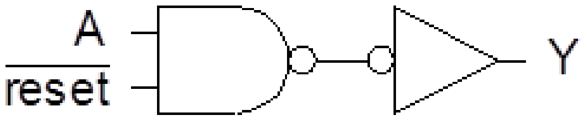
- Reduced logical effort in the favored direction
  - at expense of larger logical effort in the other direction
  - also reduced noise margin

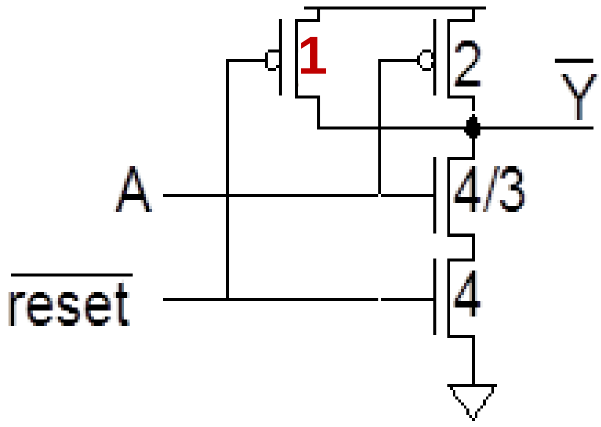
HI and LO Skew



Asymmetric + Skew

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

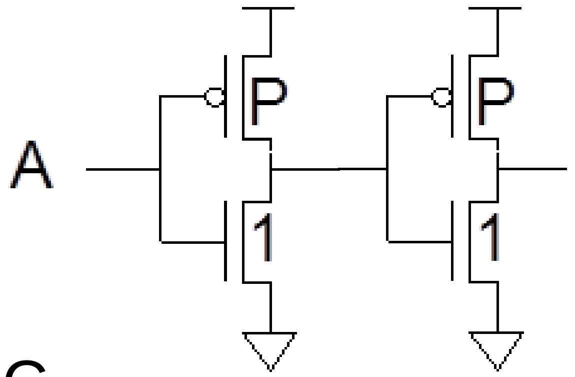




- $g_A = 10/9$ (*normally NAND2 is* 4/3)
- $p_A = 13/9$ *(normally 2)*
- $g_{reset} = (5/1.5) = 10/3$
- $g_{avg} = (g_A + g_{reset})/2 = 20/9$ (*normally 12*/9)

What is best nominal P/N ratio?

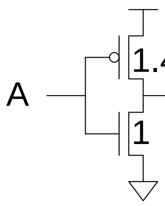
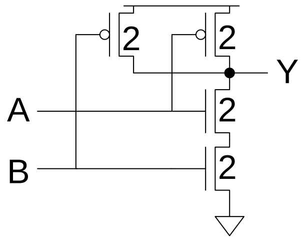
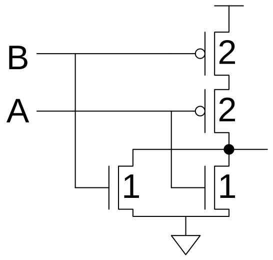
- We normally set P/N ratio for equal rise and fall resistance ($\mu = \mu_n/\mu_p$ = 2-3 for an inverter).
- Alternative: choose ratio for least average delay
- Ex: Calculate delay of inverter driving identical inverter



  - $t_{pdf} = 2.(P+1).RC$
  - $t_{pdr} = 2.(P+1)(\mu/P).RC$
  - $t_{pd} = 2.RC.(P+1).(1+\mu/P)/2 = RC.(P + 1 + \mu + \mu/P)$
  - Least delay when $dt_{pd} / dP = RC.(1- \mu/P^2) = 0$
  - when $P = \sqrt{\mu}$

P/N Ratios

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

	Inverter	NAND2	NOR2
fastest P/N ratio			
	$g_u = 1.14$	$g_u = 4/3$	$g_u = 2$
	$g_d = 0.80$	$g_d = 4/3$	$g_d = 1$
	$g_{avg} = 0.97$	$g_{avg} = 4/3$	$g_{avg} = 3/2$

Observations

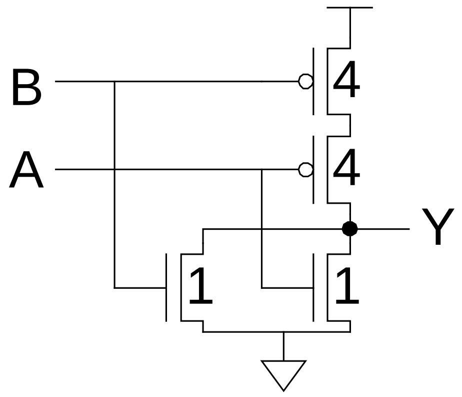
- For speed:
  - NAND vs. NOR
  - Many simple stages vs. fewer high fan-in stages
  - Latest-arriving input

- For area and power:
  - Many simple stages vs. fewer high fan-in stages

- P/N ratio should be chosen on the basis of area & power, not average delay
  - In most standard cell libraries, the pitch of the cell influences P/N ratio of individual gates

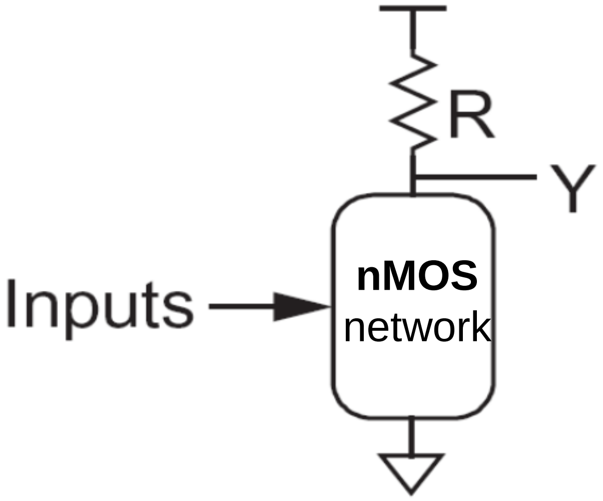
Beyond Static CMOS

- What makes a circuit fast?
  - $I = C\, dV/dt$ -> $t_{pd} \propto (C/I)\, \Delta V$
  - low capacitance
  - high current
  - small swing
- Logical effort is proportional to C/I
- pMOS transistors are the enemy!
  - High capacitance for a given current
- Can we take the pMOS capacitance off the input?
- Various circuit families try to do this…

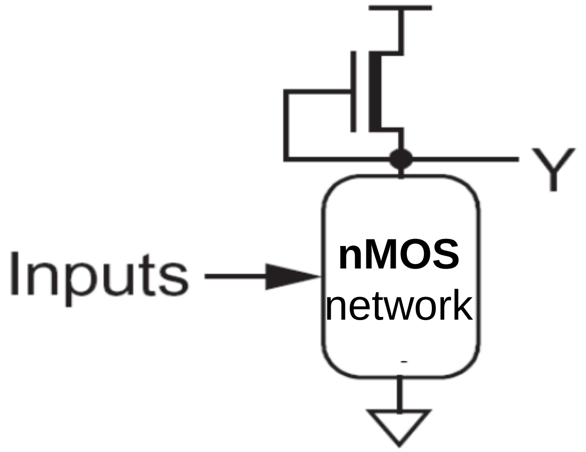


Ratio’d Circuits

- Ratio’s circuits use a passive pullup instead of active pMOS devices.
  - when nMOS network is not conducting, output is high
  - when nMOS network is conducting, it is stronger than R and pulls output low
  - resistors are impractically large



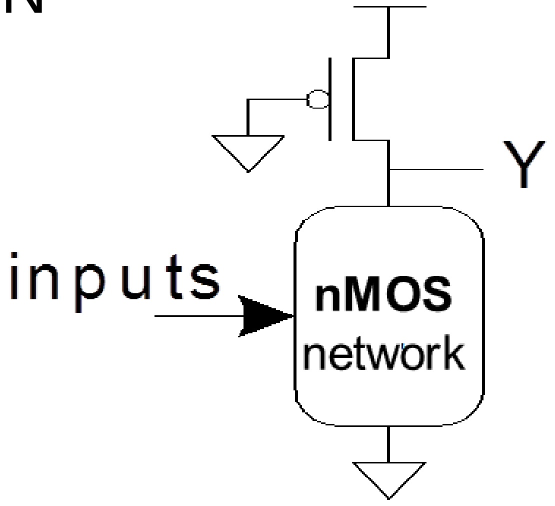
- Before CMOS, nMOS logic families used depletion device as passive load
  - depletion transistor has $V_T<0$



- Unlike complimentary CMOS, ratio of transistor sizes must be carefully chosen to ensure correct operation

Psuedo-nMOS

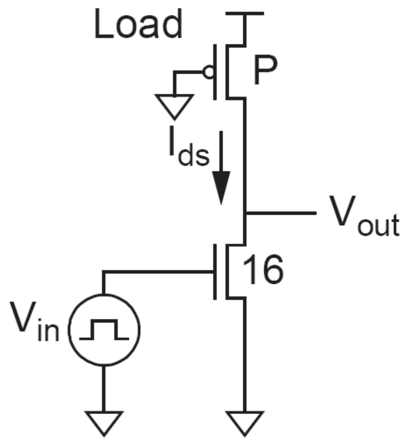
- In CMOS, use a single pMOS transistor as load
- pMOS gate grounded so its always ON
  - *ratio* issue

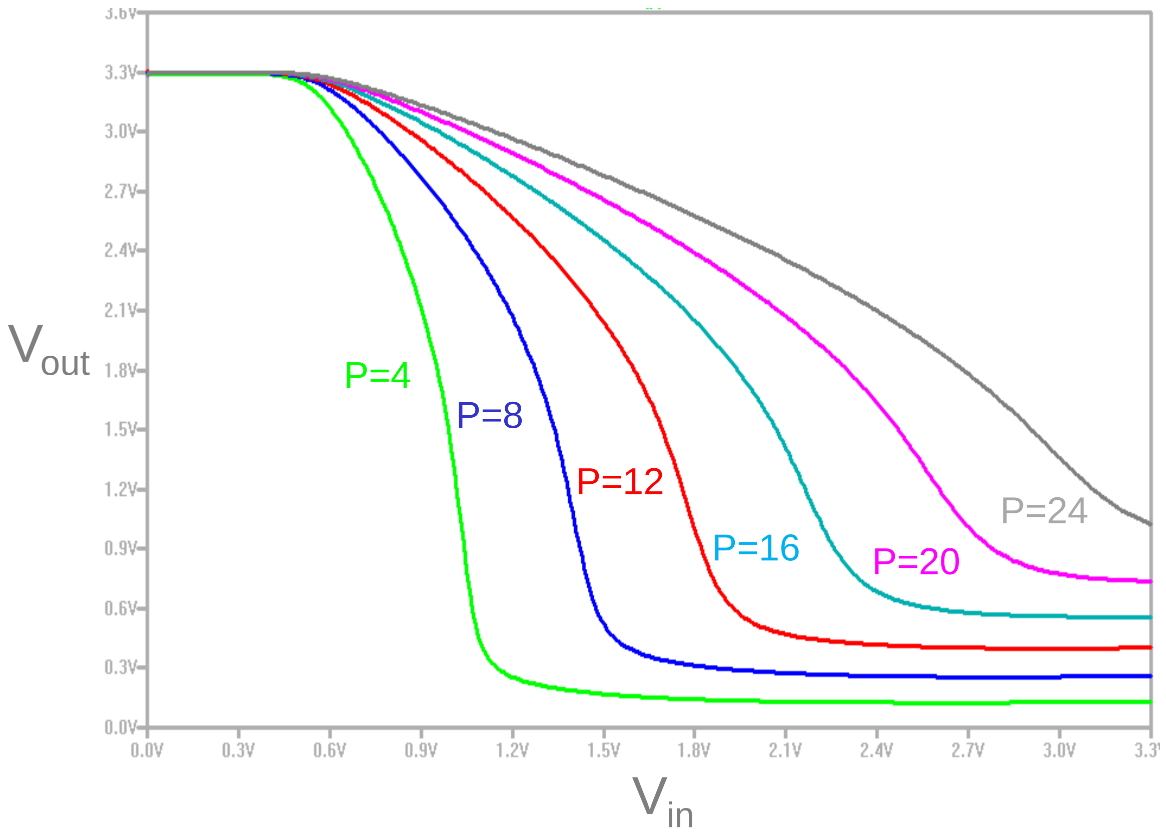


- What size should the pMOS be?
  - If too large, will slow 1→0 transition and gate may not pull down properly
  - If too small, will slow 0→1 transition

Psuedo-nMOS

- Use SPICE to try out different ratios:

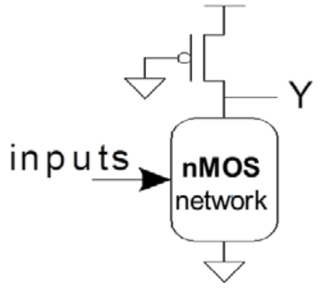




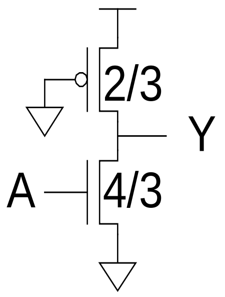
- Make pMOS about ¼ strength of pulldown network
  - compromise between speed & noise margin

Psuedo-nMOS Performance

- Logical effort is ratio of input capacitance of gate to that of standard complimentary inverter that delivers same current
- Parasitic delay is ratio of output capacitance compared to standard inverter delivering same current
- Remember that on pull-down: pMOS fights nMOS

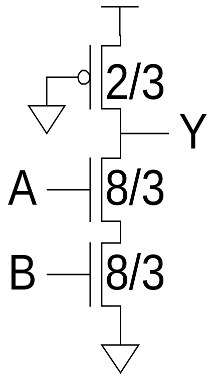


Inverter



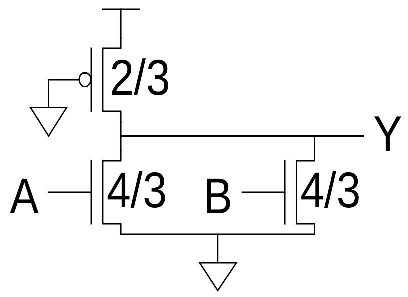
$g_u = 4/3$
$g_d = 4/9$
$g_{avg} = 8/9$
$p_u = 6/3$
$p_d = 6/9$
$p_{avg} = 12/9$

NAND2



$g_u = 8/3$
$g_d = 8/9$
$g_{avg} = 16/9$
$p_u = 10/3$
$p_d = 10/9$
$p_{avg} = 20/9$

NOR2



$g_u = 4/3$
$g_d = 4/9$
$g_{avg} = 8/9$
$p_u = 10/3$
$p_d = 10/9$
$p_{avg} = 20/9$

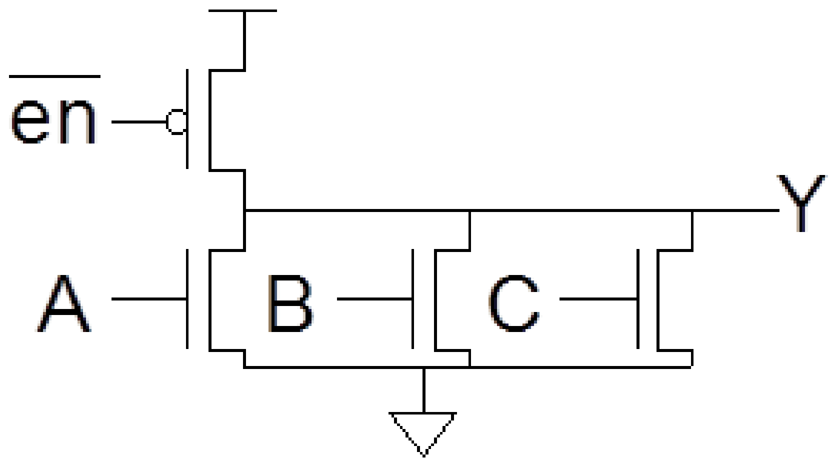
- Best suited to large fan-in NOR networks

Psuedo-nMOS Power

- Pseudo-nMOS draws power whenever Y = 0
  - Called static power P = $I_{DD}V_{DD}$
  - A few hundred μA / gate * 1M gates is a problem
  - Explains why nMOS went extinct

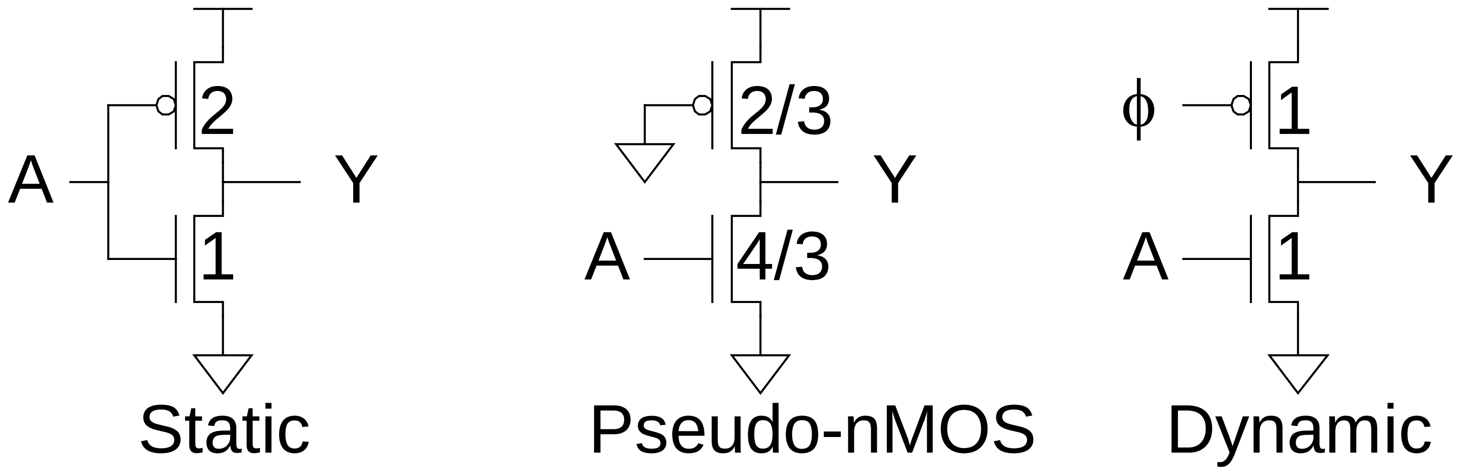
- Use pseudo-nMOS sparingly for wide NORs

- Turn off pMOS when not in use



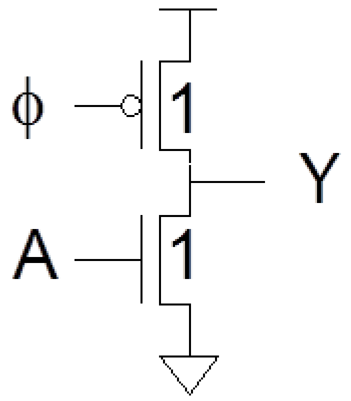
Dynamic Logic

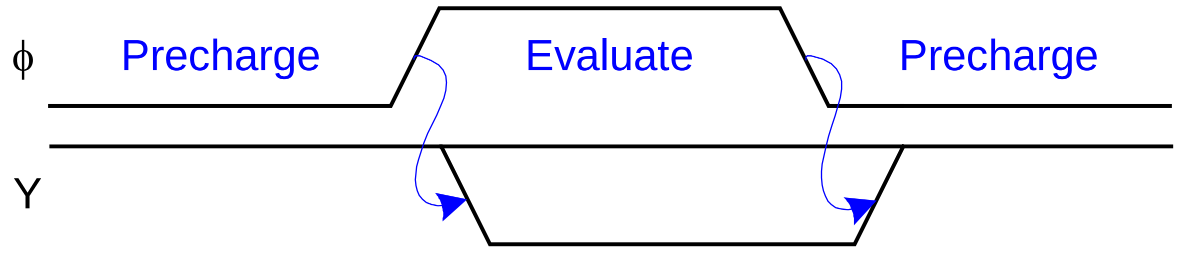
- Ratio'd circuits reduce input capacitance by replacing pMOS pullup tree with a single static load
  - slow rising transitions
  - contention on falling transitions
  - static power dissipation
  - non-zero V_{OL} (reduced noise margin)
- *Dynamic* gates use a clocked pMOS pullup



Dynamic Logic Phases

- *Dynamic* gates operate in two phases: precharge and evaluate
- During pre-charge phase (ϕ=0), the output Y is initialized high
- During the evaluate phase (ϕ=1), Y is conditionally discharged low, depending on the value of input A

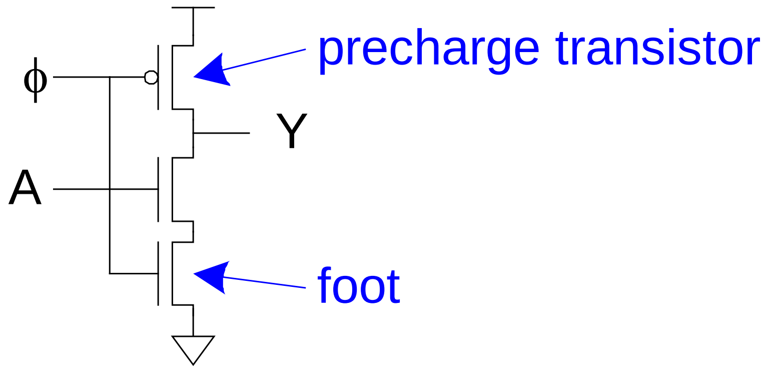


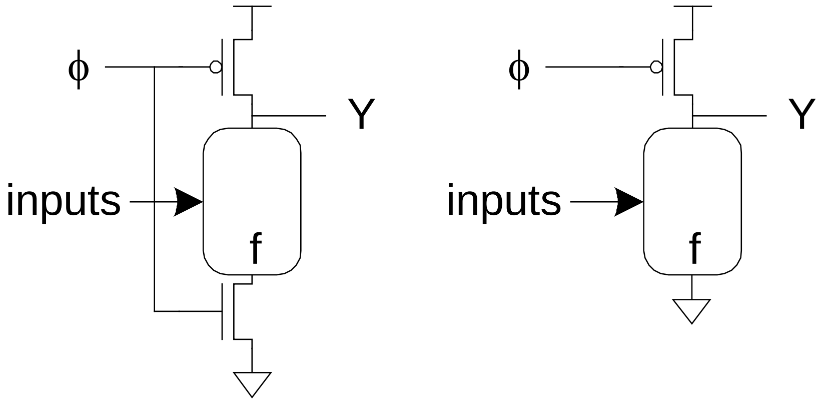


- What happens if A=1 during precharge?

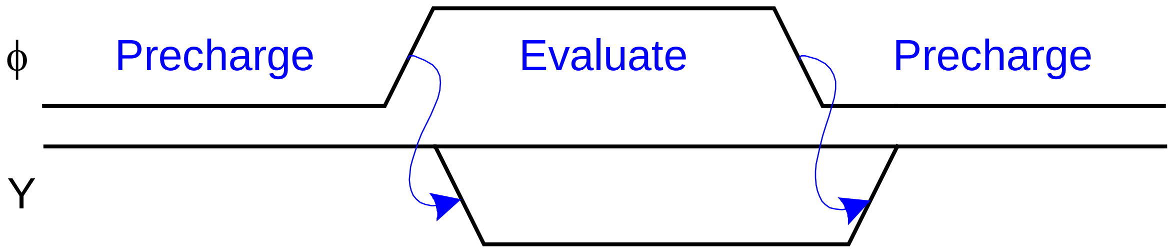
The Foot

- Introduce series nMOS evaluation transistor called foot
  - eliminates contention during precharge

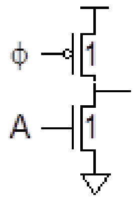
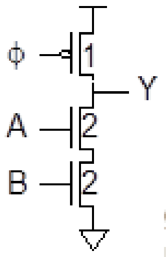
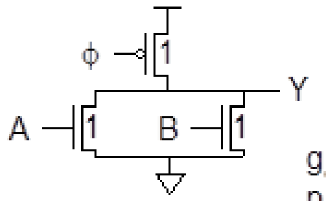
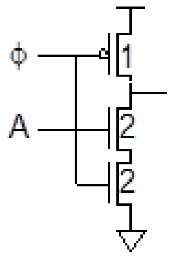
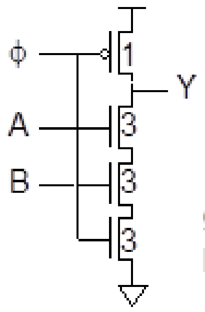
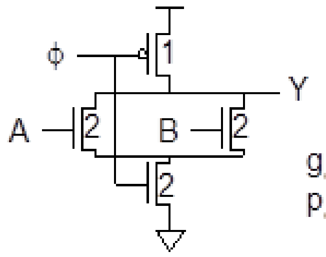




footed unfooted

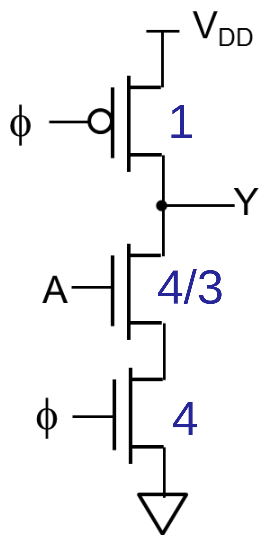


Logical Effort of Dynamic Gates

	Inverter	NAND2	NOR2
unfooted	 $g_d = 1/3$ $p_d = 2/3$	 $g_d = 2/3$ $p_d = 3/3$	 $g_d = 1/3$ $p_d = 3/3$
footed	 $g_d = 2/3$ $p_d = 3/3$	 $g_d = 3/3$ $p_d = 4/3$	 $g_d = 2/3$ $p_d = 5/3$
compare to static:	*g=1* *p=1*	*g=4/3* *p=2*	*g=5/3* *p=2*

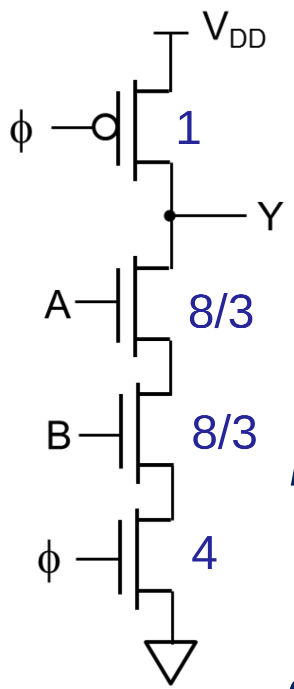
Dynamic Gates with Large (W=4) Foot

Inverter



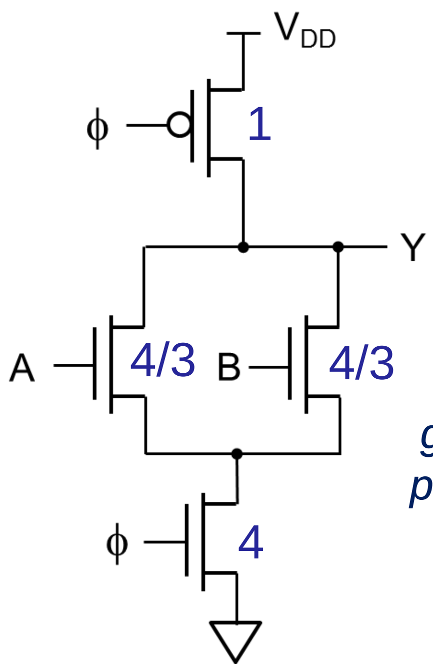
g=4/9
p=7/9

NAND2



g=8/9
p=11/9

NOR2



g=4/9
p=11/9

	Inverter	NAND2	NOR2
compare to regular foot:	*g=2/3* *p=1*	*g=1* *p=4/3*	*g=2/3* *p=5/3*
compare to static:	*g=1* *p=1*	*g=4/3* *p=2*	*g=5/3* *p=2*

- at expense of considerably increased area and power!

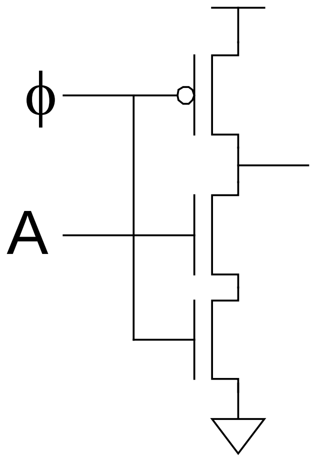
Compared to static logic

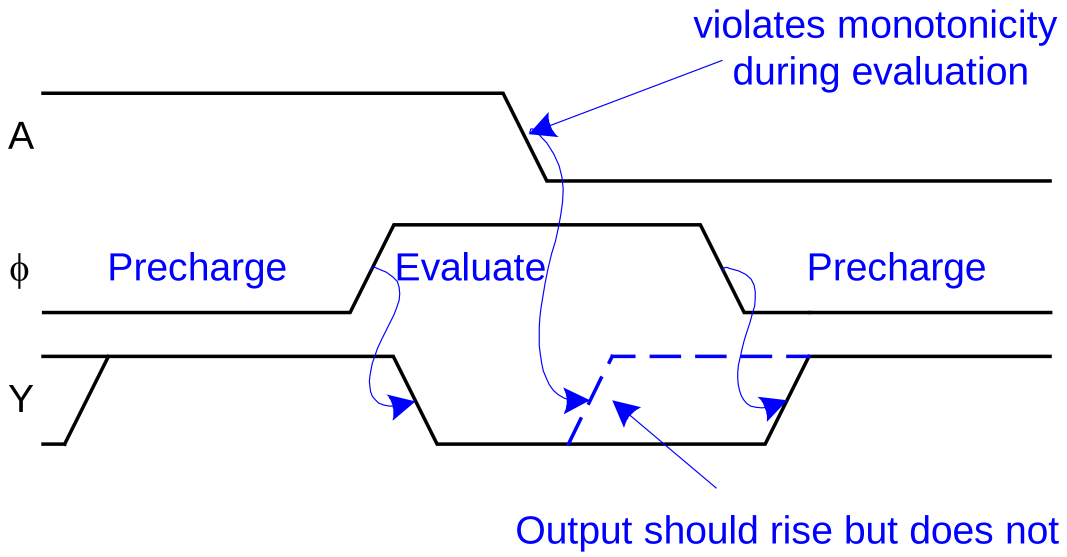
- Advantages of dynamic gates:
  - fastest commonly used circuit family
  - lower input capacitance
  - no contention during switching
  - zero static power dissipation
  - no ratio issues

- Limitations of dynamic gates:
  - precharge/evaluate paradigm
  - require careful clocking
  - consume significant dynamic power
  - reduced noise margin
  - monotonicity requirement

Monotonicity

- Dynamic gates require *monotonically rising* inputs during evaluation
  - 0 -> 0
  - 0 -> 1
  - 1 -> 1
  - but not 1 -> 0!

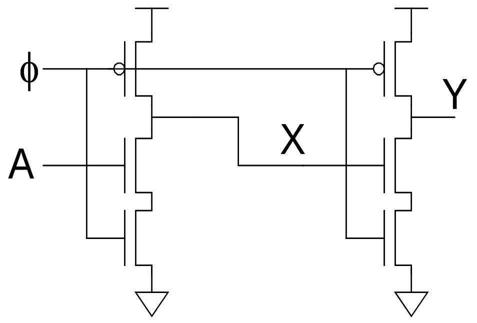


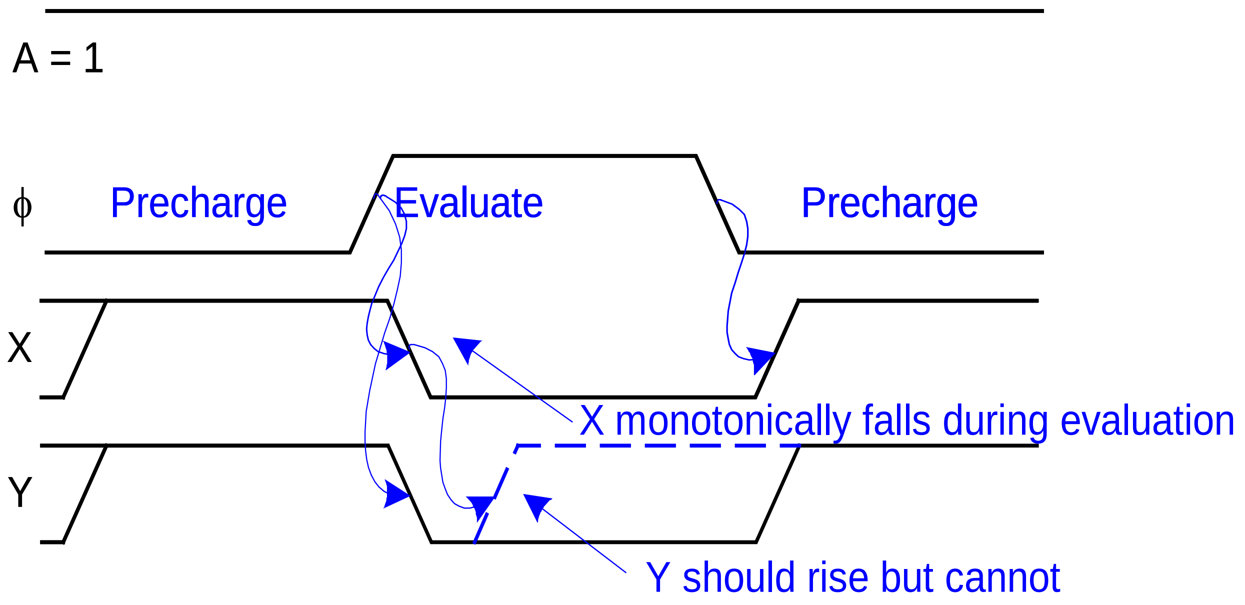


violates monotonicity during evaluation
A
ϕ
Precharge
Evaluate
Precharge
Y
Output should rise but does not

Monotonicity Woes

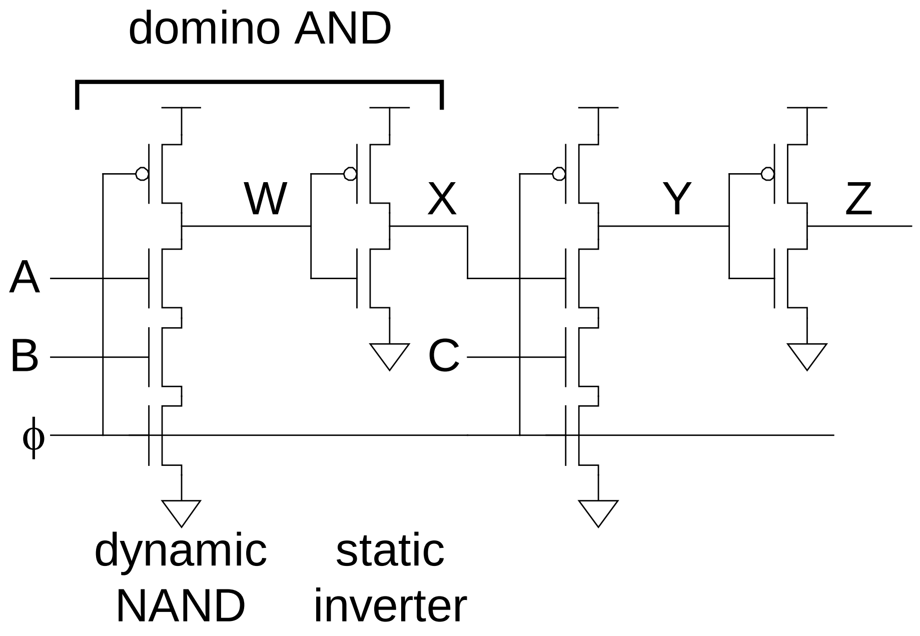
- But dynamic gates produce *monotonically falling* outputs during evaluation
- Illegal for one dynamic gate to drive another!





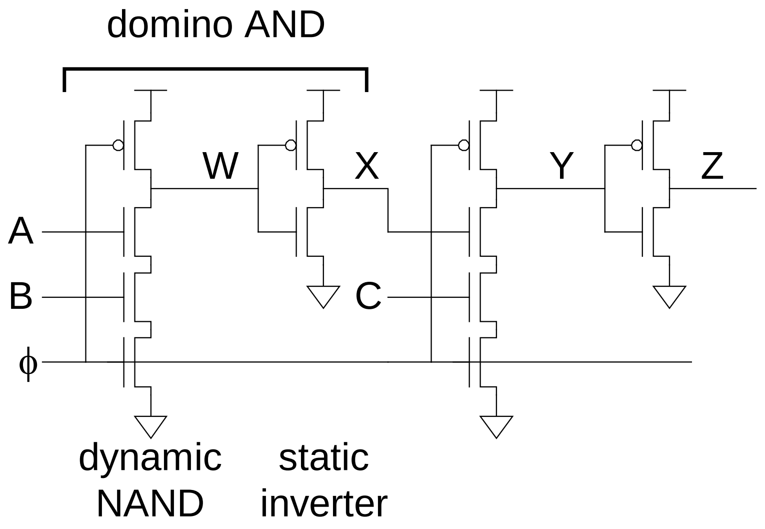
Domino Gates

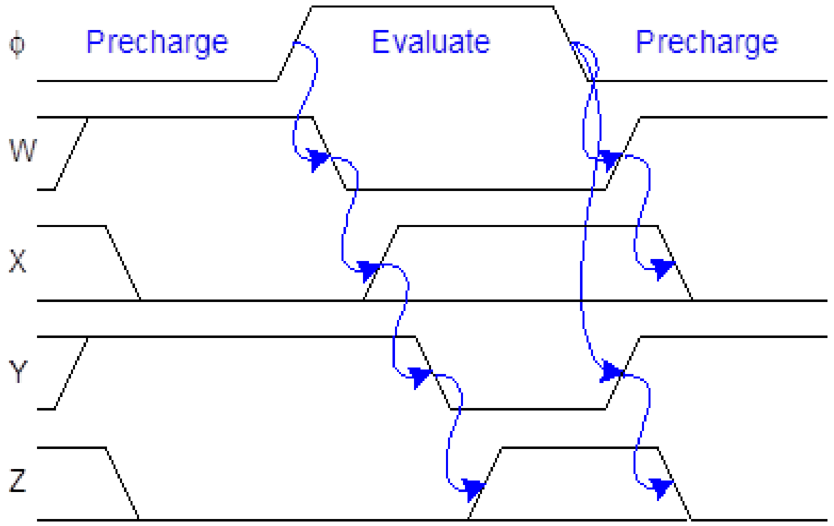
- Monotonicity problem can be solved by putting an static inverter between each dynamic gate
- Inverter output will be monotonically rising



Domino Operation

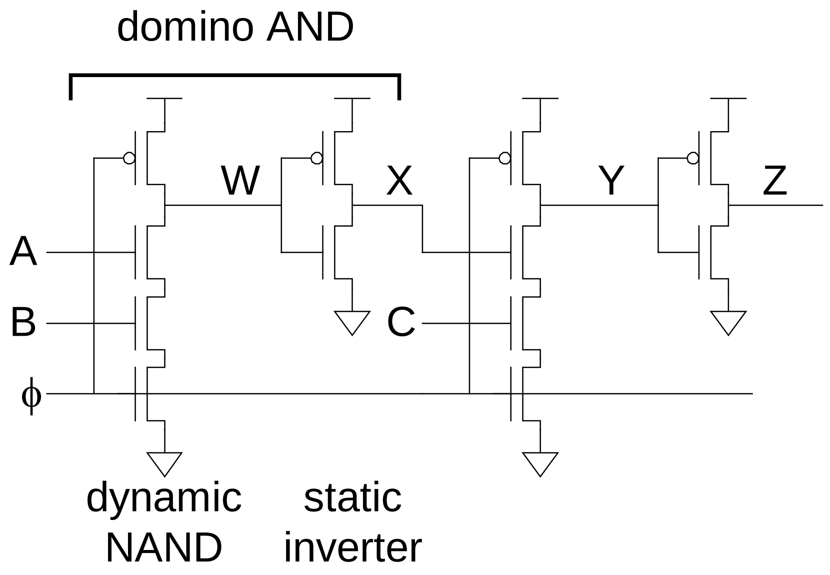
- All dynamic gates in chain are precharged in parallel
- During evaluation phase, a falling transition at the output of the first dynamic gate generates a rising transition at the output of the inverter which, in turn, is input to the second dynamic gate.
- Each domino gate triggers next one, like a string of dominos toppling over

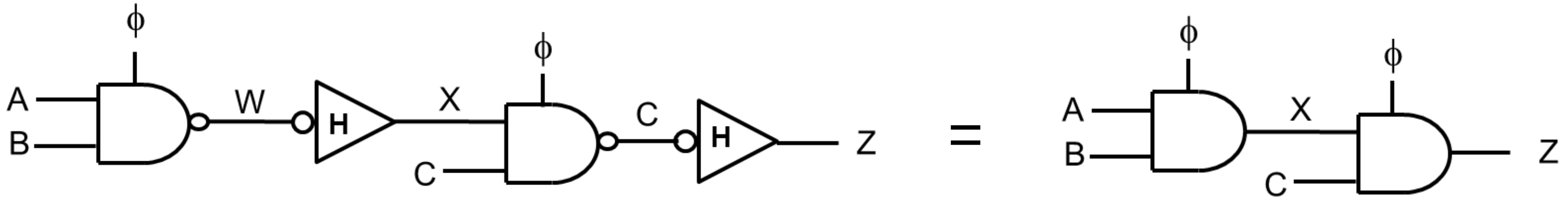




Domino Optimization

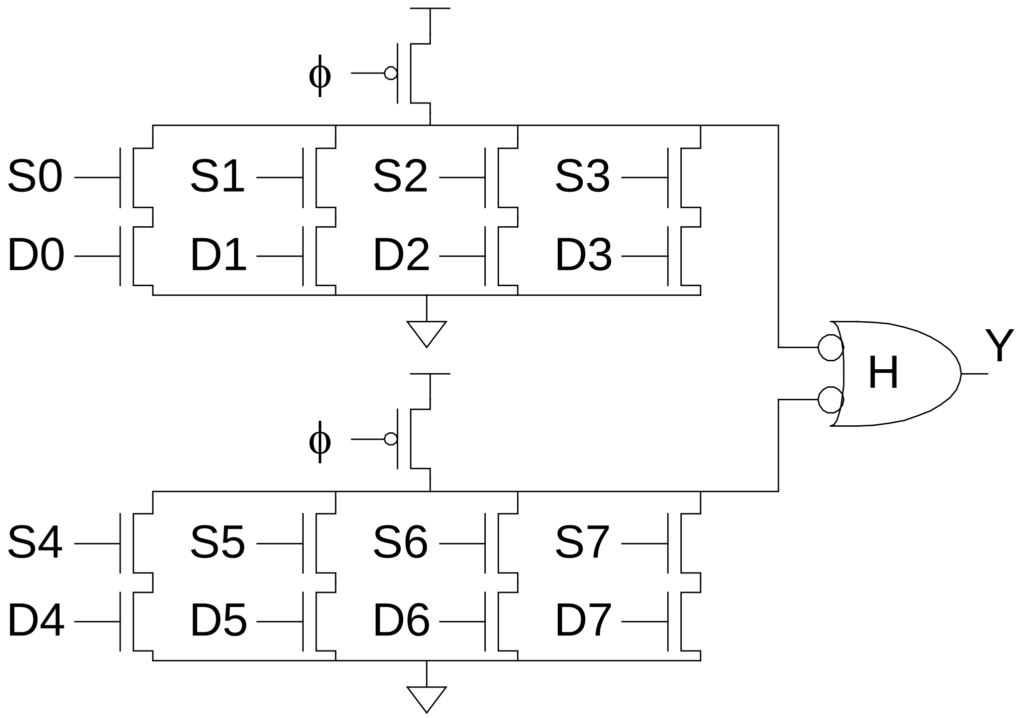
- Gates evaluate sequentially but precharge in parallel
- Thus evaluation is more critical than precharge
- Use high-skew inverter





Compound Domino

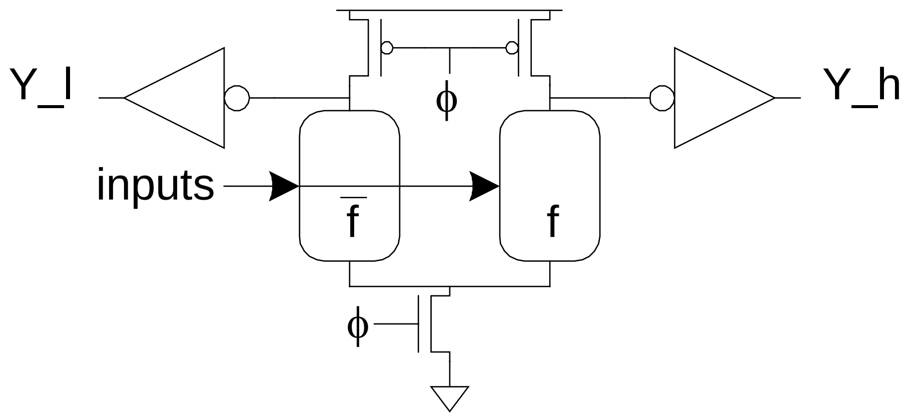
- More complex inverting (high skew) static gates can be used in place of inverter
- Example: 8 input domino multiplexer



Dual Rail Domino

- Domino only performs non-inverting functions:
  - AND, OR but not NAND, NOR, or XOR
- Dual-rail domino solves this problem
  - Takes true and complementary inputs
  - Produces true and complementary outputs

sig_h	sig_l	Meaning
0	0	Precharged
0	1	‘0’
1	0	‘1’
1	1	invalid



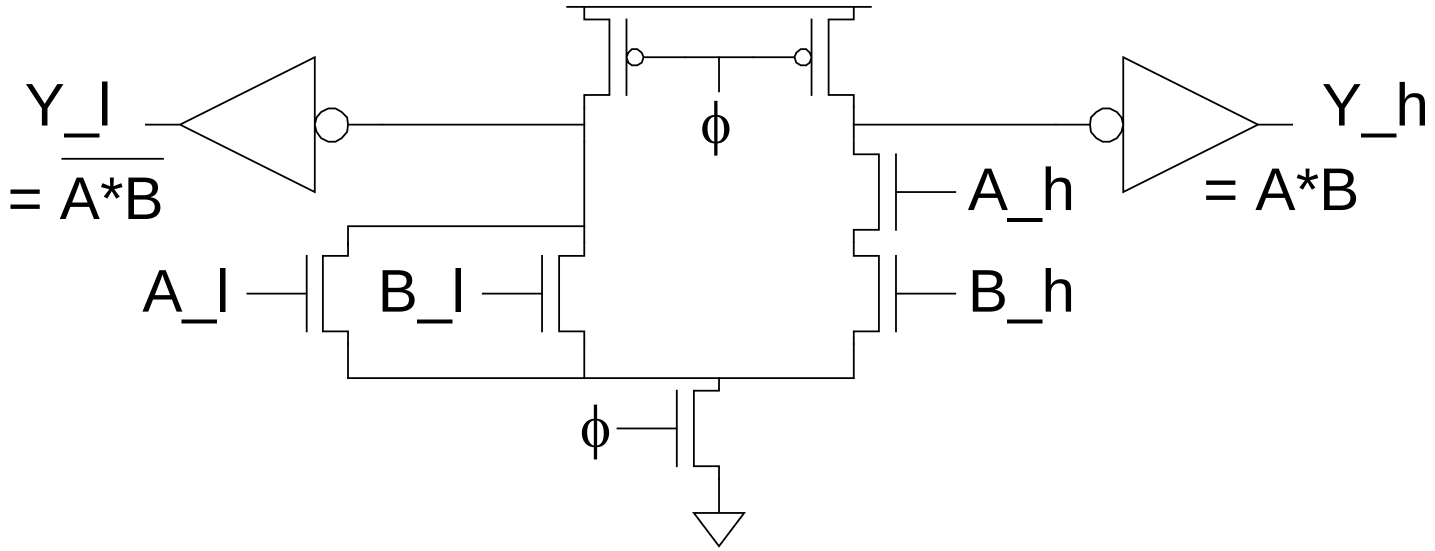
Example: AND/NAND

- *Given* A_h, A_l, B_h, B_l

  compute Y_h = A•B = A_h • B_h

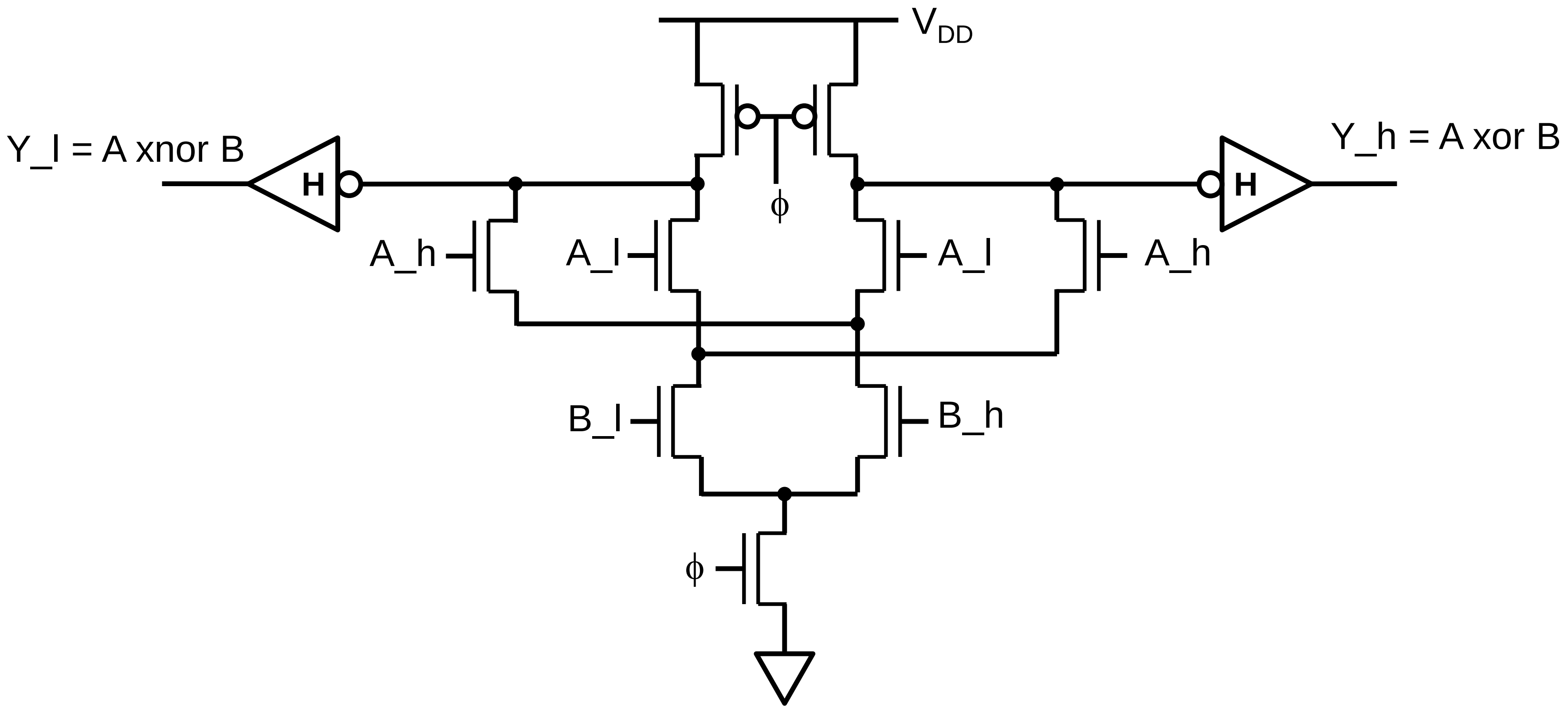
  and Y_l = $\overline{\text{A•B}}$ = $\overline{\text{A}}$ + $\overline{\text{B}}$ = A_l + B_l

- Pulldown networks are topological complements



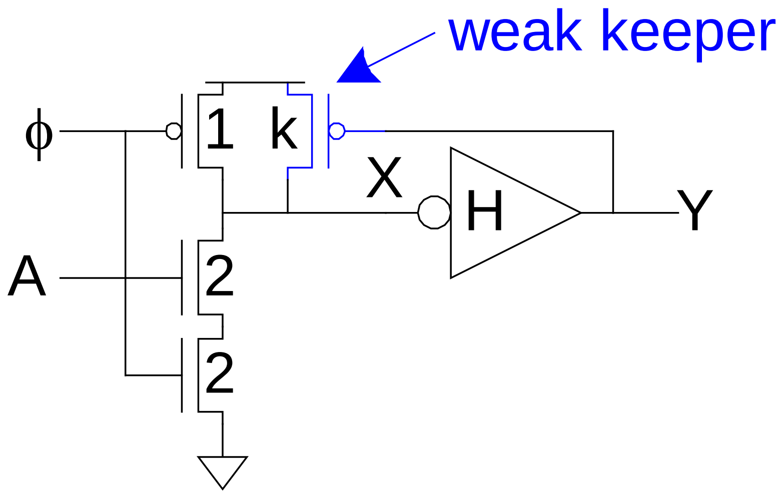
Example: XOR/XNOR

- Sometimes possible to share transistors:



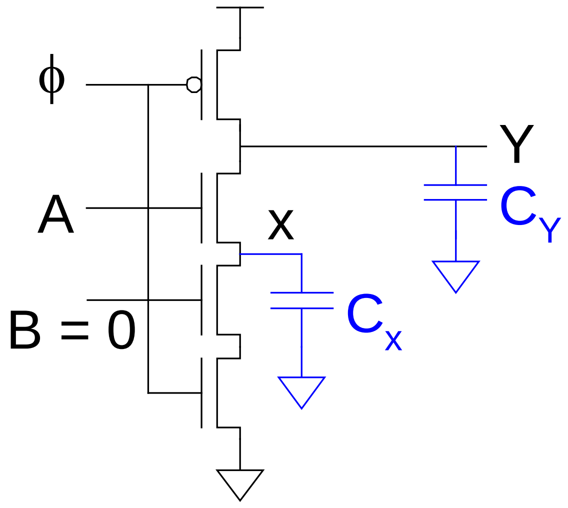
Dynamic Hazards: Leakage

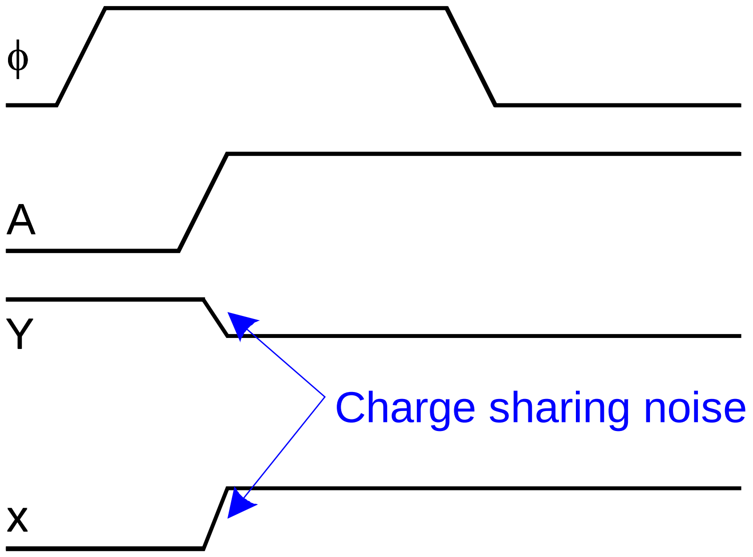
- Dynamic node is not driven high during evaluation
  - Floating node held by charge on node capacitance
  - Transistors are leaky ($I_{OFF} \neq 0$)
  - Dynamic value will leak away over time
  - Used to be miliseconds, now nanoseconds
- Use keeper to hold dynamic node
  - Must be weak enough not to fight evaluation



Dynamic Hazards: Charge Sharing

- Transitions on inner inputs can steal charge from output node:





$$V_x = \frac{C_Y}{C_x + C_Y} . V_{DD}$$

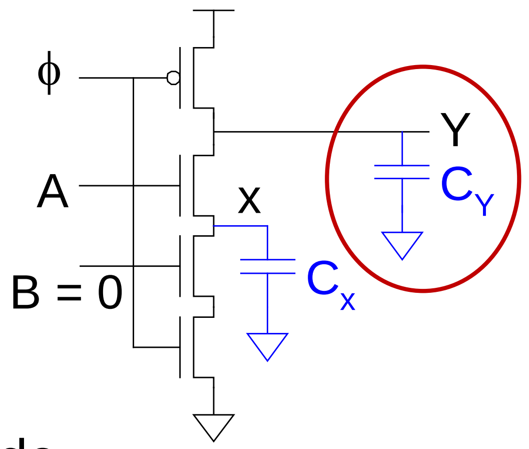
(not important)

$$\Delta V_Y = \frac{C_x}{C_x + C_Y} . V_{DD}$$

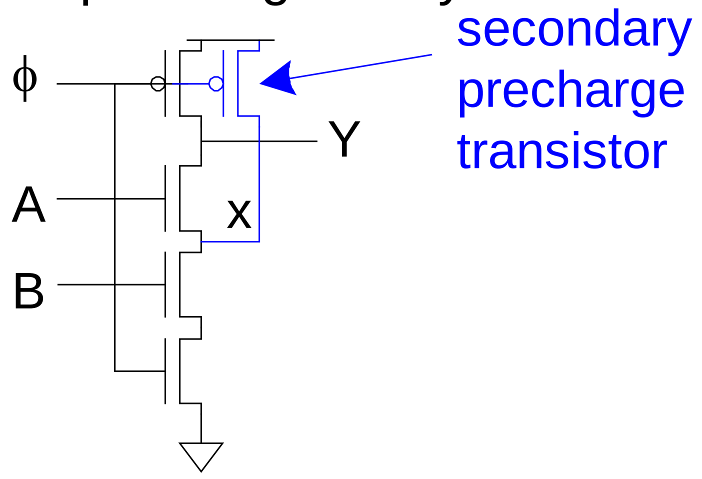
may cause output error!

Solutions to Charge Sharing

- Increase size of load capacitance C_Y
  - increases gate delay



- Add secondary precharge transistors
  - typically need to precharge every other node



- A keeper transistor can restore output if charge sharing is small

Dynamic Hazards: Noise

- Dynamic gates are very sensitive to noise
  - Inputs: $V_{IL} \approx V_{tn}$
  - Outputs: floating output susceptible noise

- Noise sources include:
  - Capacitive crosstalk
  - Charge sharing
  - Power supply noise
  - Feedthrough noise

Dynamic Hazards: Dynamic Power

- Domino gates have high activity factors
  - Gate precharges and evaluates each clock cycle
  - When output of dynamic gate remains high
    - no transitions per clock cycle
  - When output of dynamic gate is pulled low
    - 2 transitions per clock cycle
  - If output transition probability = 0.5,
    - Gate activity factor $\alpha = 0.5$
  - Also clock power dissipated in precharge and foot transistors

- Leads to very high power consumption

Domino Summary

- Domino logic is attractive for high-speed circuits
  - 1.3 – 2x faster than static CMOS
  - But many challenges:
    - Monotonicity, leakage, charge sharing, noise
- Widely used in high-performance microprocessors in 1990s when speed was primary driver
- Largely displaced by static CMOS now that power is the limiter
- Still used in memories for speed & area efficiency
  - wide NOR decoder structures