

Combinational Circuit Design

Dr. Bassam Jamil

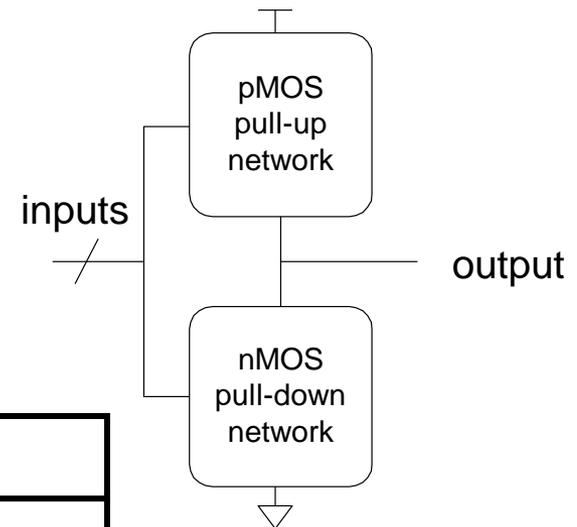
**Adopted from slides of the
textbook**

Topics

- ❑ CMOS Gate
 - Complementary N and P networks
 - Complex Gates
 - Pass transistor and transmission gates
 - Tristates, multiplexers

Complementary CMOS

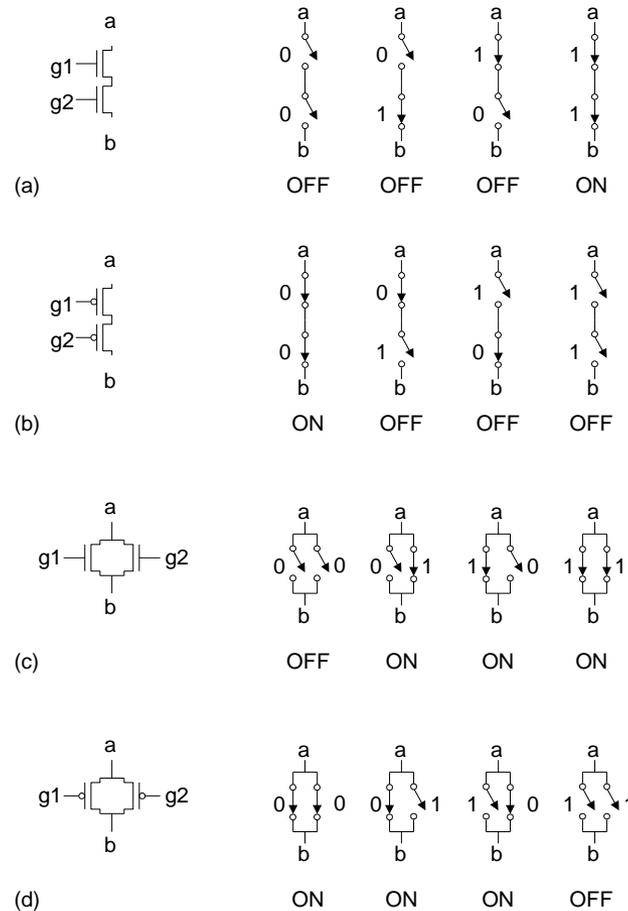
- Complementary CMOS logic gates
 - nMOS *pull-down network*
 - pMOS *pull-up network*
 - a.k.a. static CMOS



	Pull-up OFF	Pull-up ON
Pull-down OFF	Z (float)	1
Pull-down ON	0	X (crowbar)

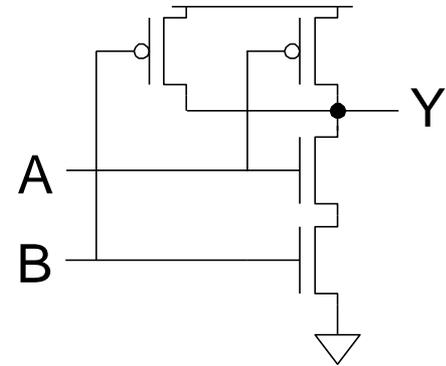
Series and Parallel

- nMOS: 1 = ON
- pMOS: 0 = ON
- *Series*: both must be ON
- *Parallel*: either can be ON



Conduction Complement

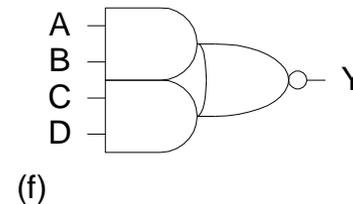
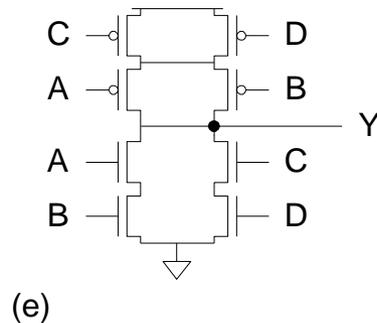
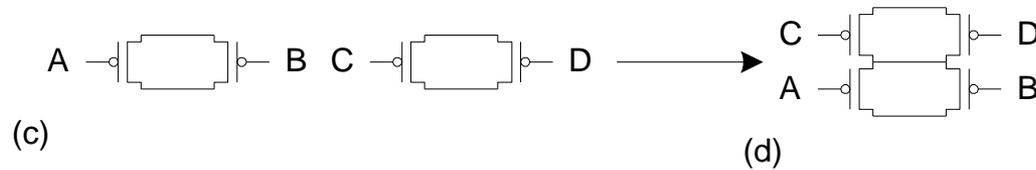
- ❑ Complementary CMOS gates always produce 0 or 1
- ❑ Ex: NAND gate
 - Series nMOS: $Y=0$ when both inputs are 1
 - Thus $Y=1$ when either input is 0
 - Requires parallel pMOS
- ❑ Rule of *Conduction Complements*
 - Pull-up network is complement of pull-down
 - Parallel \rightarrow series, series \rightarrow parallel



Compound Gates

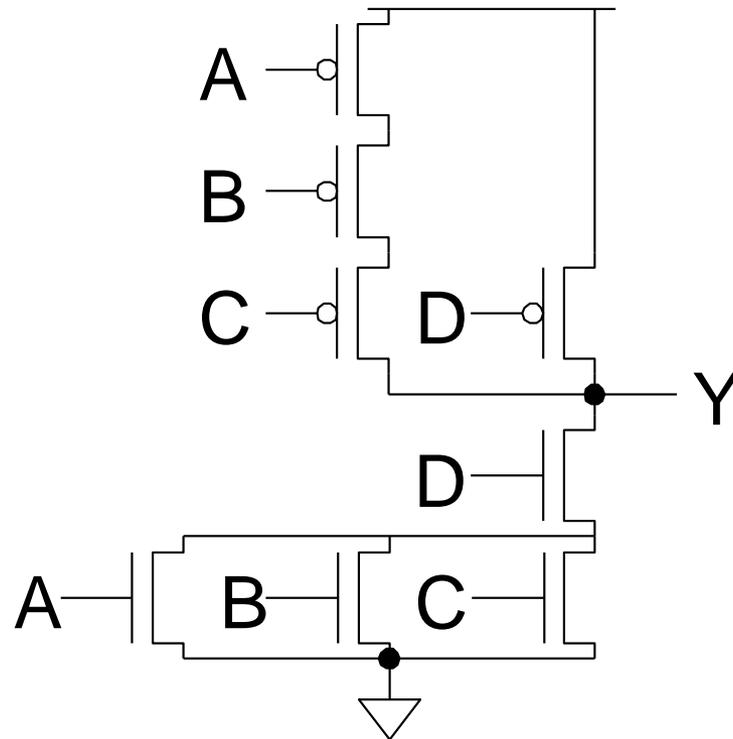
❑ *Compound gates can do any inverting function*

❑ Ex: $Y = (AB + CD)'$



Example: O3AI

$$\square Y = ((A + B + C) D)'$$

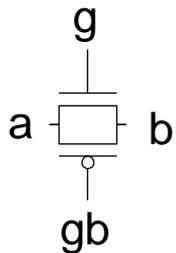


Signal Strength

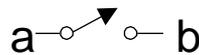
- ❑ *Strength* of signal
 - How close it approximates ideal voltage source
- ❑ V_{DD} and GND rails are strongest 1 and 0
- ❑ nMOS pass strong 0
 - But degraded or weak 1
- ❑ pMOS pass strong 1
 - But degraded or weak 0
- ❑ Thus nMOS are best for pull-down network

Transmission Gates

- ❑ Pass transistors produce degraded outputs
- ❑ *Transmission gates* pass both 0 and 1 well



$$g = 0, gb = 1$$



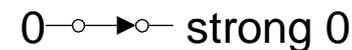
$$g = 1, gb = 0$$



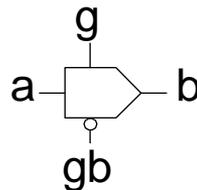
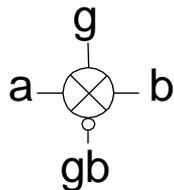
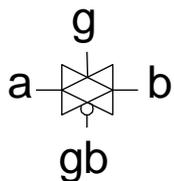
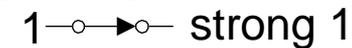
Input

Output

$$g = 1, gb = 0$$



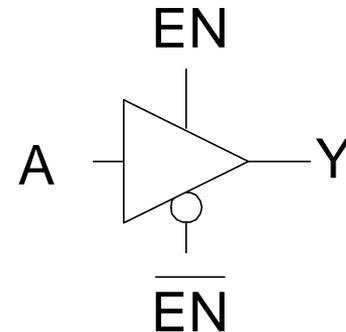
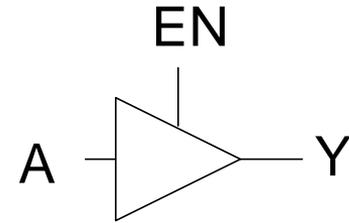
$$g = 1, gb = 0$$



Tristates

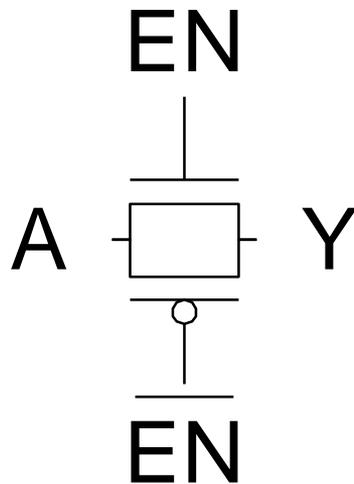
- ❑ *Tristate buffer* produces Z when not enabled

EN	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1



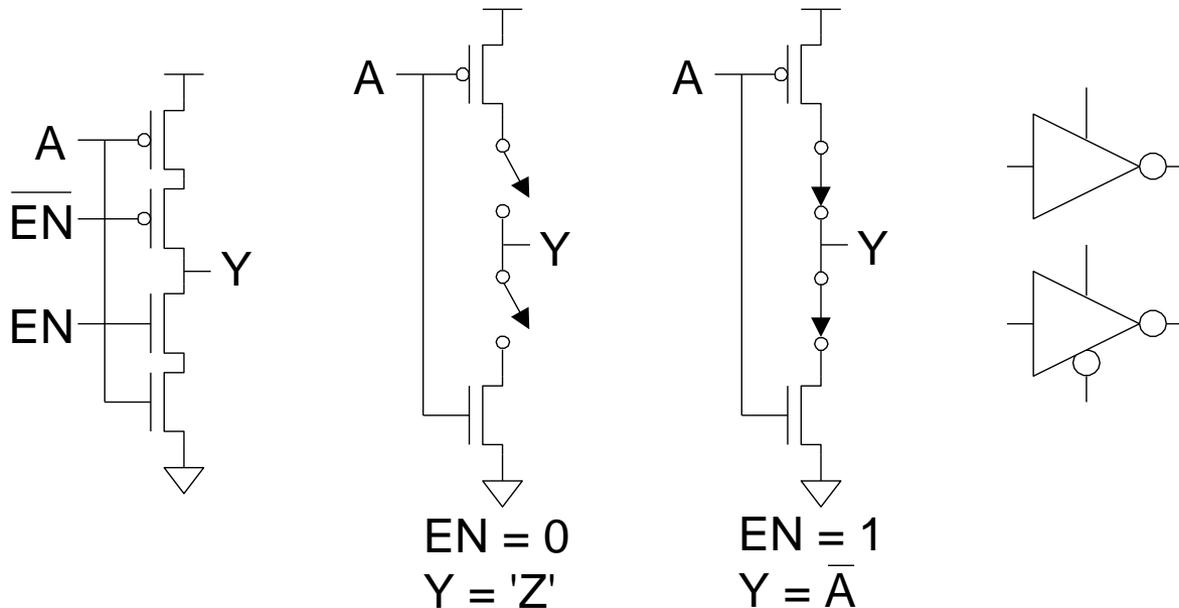
Nonrestoring Tristate

- ❑ Transmission gate acts as tristate buffer
 - Only two transistors
 - But *nonrestoring*
 - Noise on A is passed on to Y



Tristate Inverter

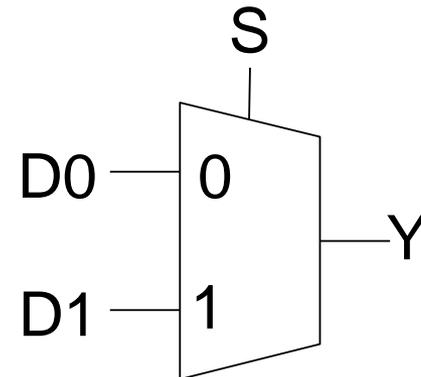
- ❑ Tristate inverter produces restored output
 - Violates conduction complement rule
 - Because we want a Z output



Multiplexers

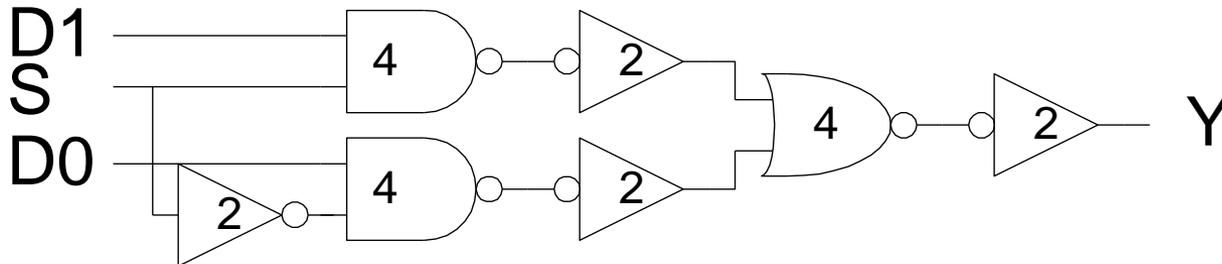
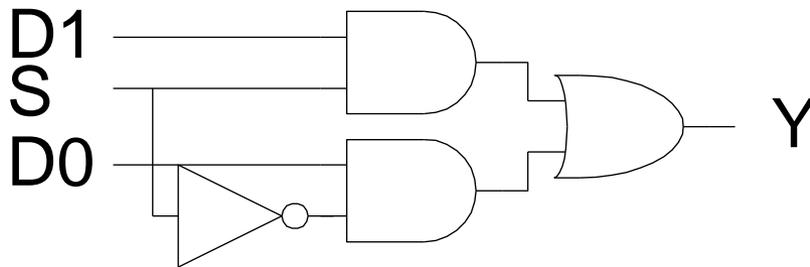
- 2:1 multiplexer chooses between two inputs

S	D1	D0	Y
0	X	0	0
0	X	1	1
1	0	X	0
1	1	X	1



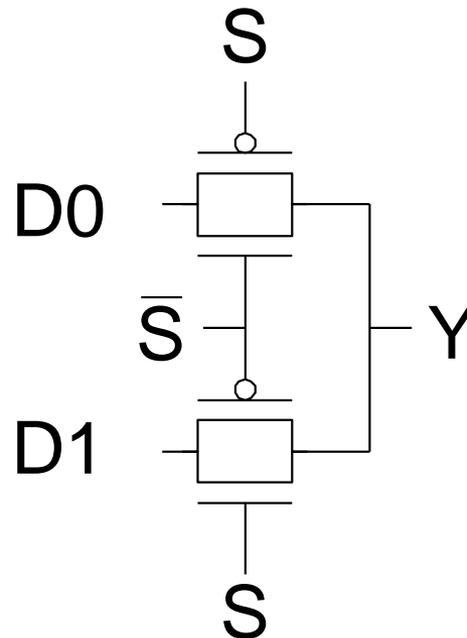
Gate-Level Mux Design

- ❑ $Y = SD_1 + \bar{S}D_0$ (too many transistors)
- ❑ How many transistors are needed? 20



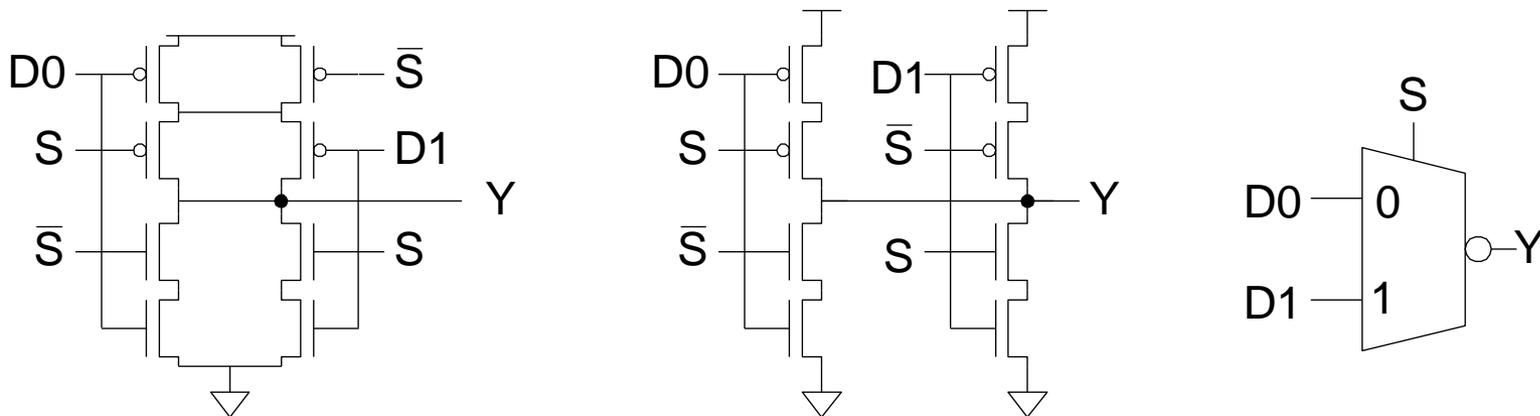
Transmission Gate Mux

- ❑ Nonrestoring mux uses two transmission gates
 - Only 4 transistors



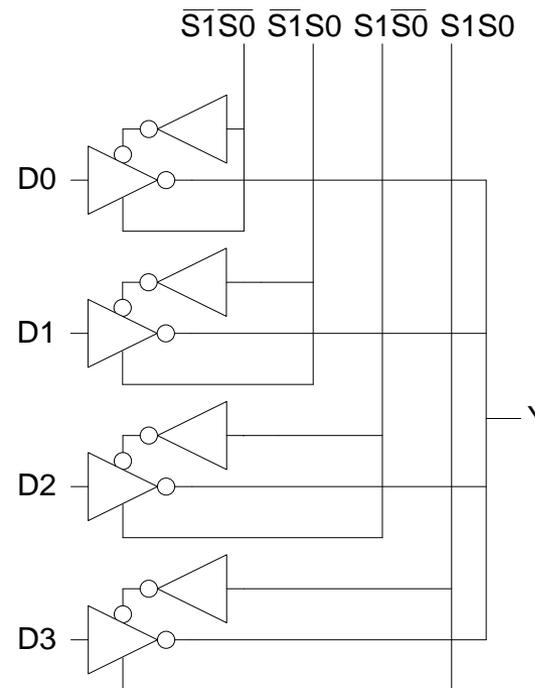
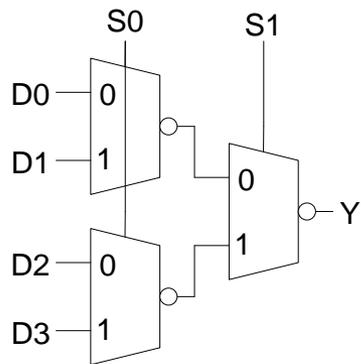
Inverting Mux

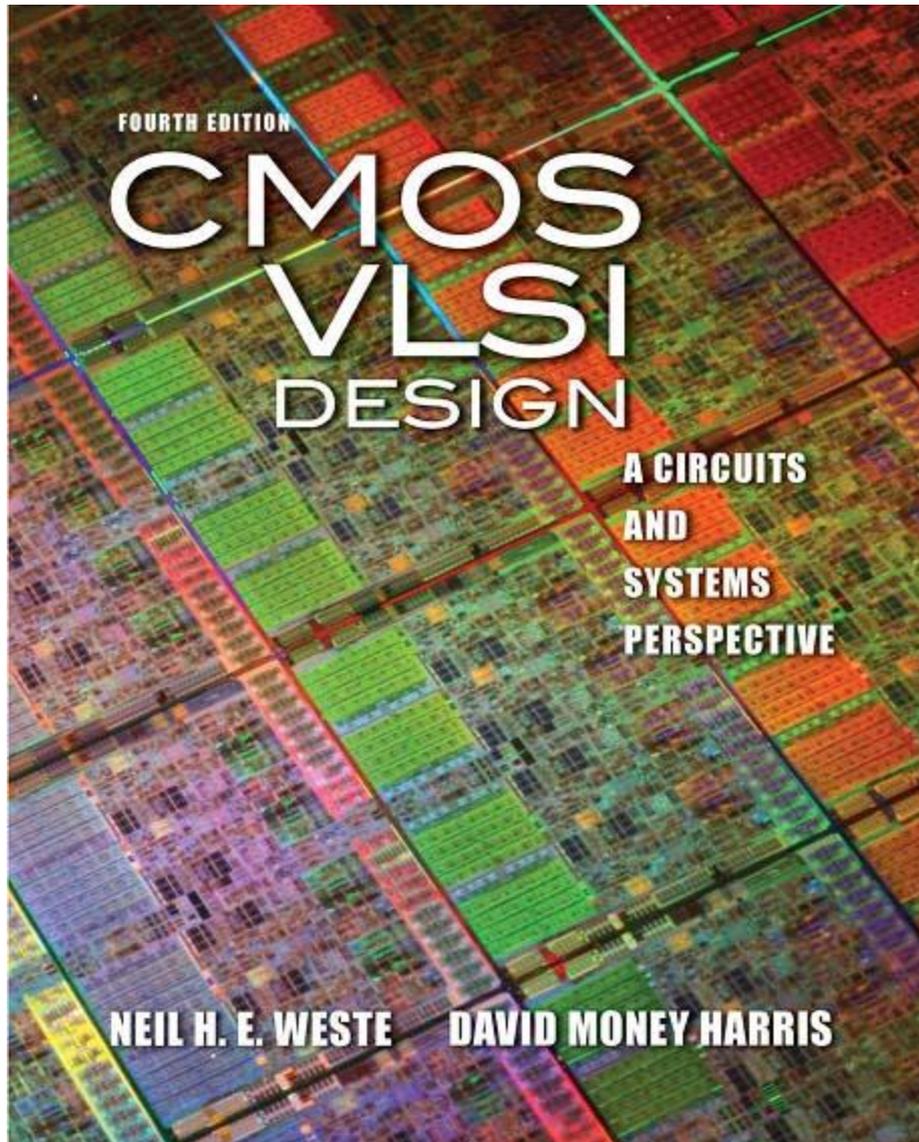
- ❑ Inverting multiplexer
 - Use compound AOI22
 - Or pair of tristate inverters
 - Essentially the same thing
- ❑ Noninverting multiplexer adds an inverter



4:1 Multiplexer

- ❑ 4:1 mux chooses one of 4 inputs using two selects
 - Two levels of 2:1 muxes
 - Or four tristates





Chapter 9: Combinational Circuit Design

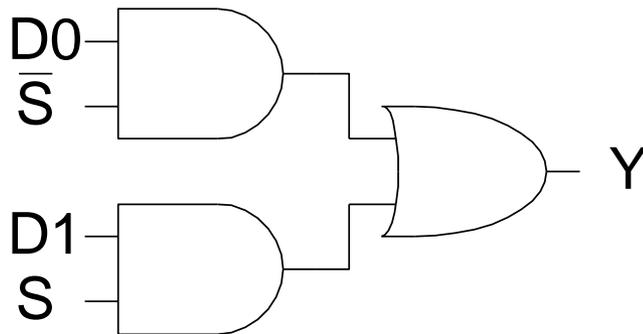
Outline

- Bubble Pushing
- Compound Gates
- Logical Effort Example
- Input Ordering
- Asymmetric Gates* (**Read**)
- Skewed Gates* (**Read**)
- Best P/N ratio

Example 1

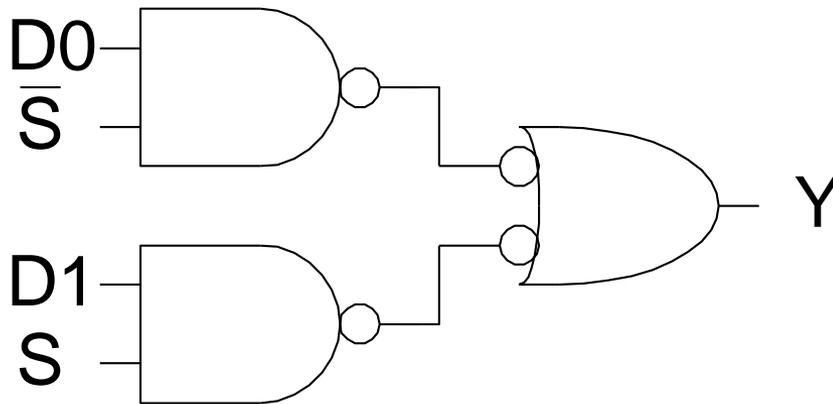
```
module mux(input s, d0, d1,  
           output y);  
  
    assign y = s ? d1 : d0;  
endmodule
```

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



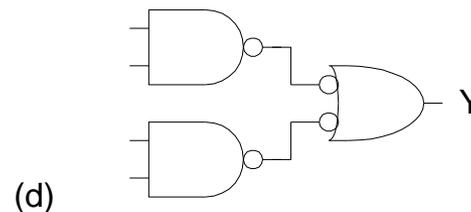
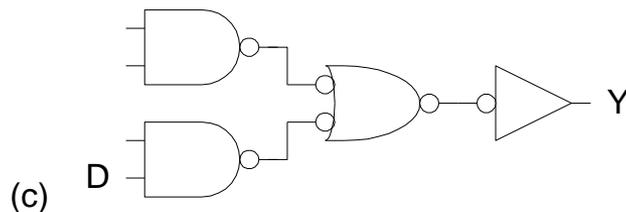
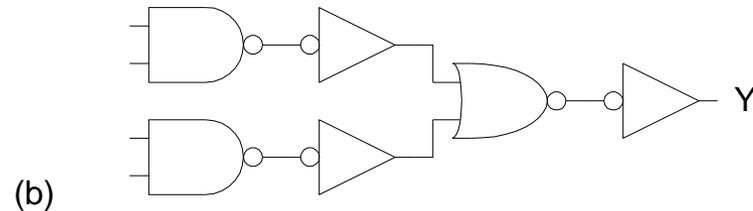
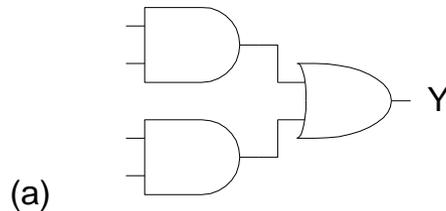
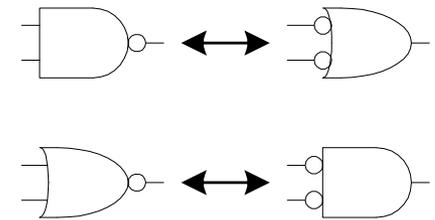
Example 2

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



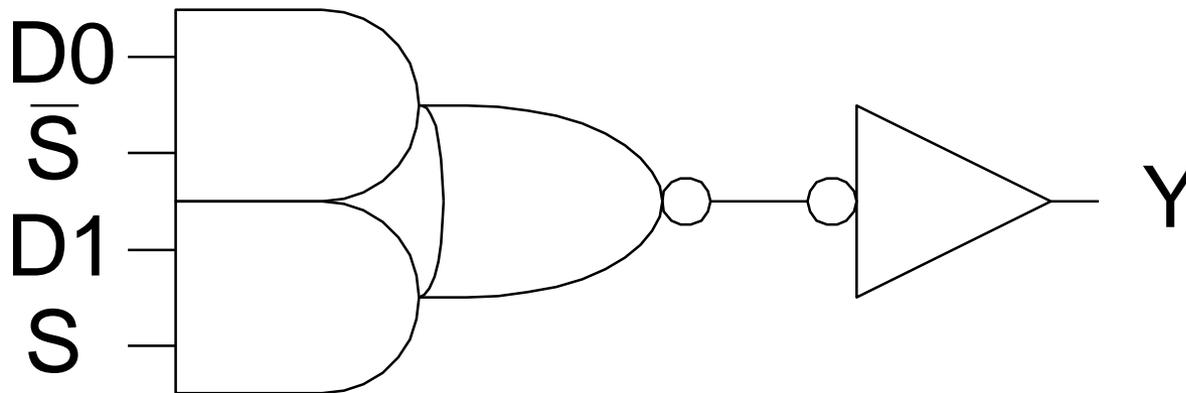
Bubble Pushing

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



Example 3

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

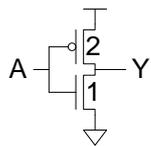
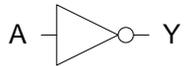


Compound Gates

Logical Effort of compound gates

unit inverter

$$Y = \overline{A}$$

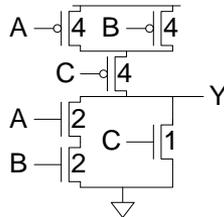
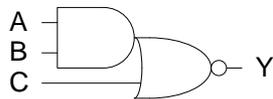


$$g_A = 3/3$$

$$p = 3/3$$

AOI21

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



$$g_A = 6/3$$

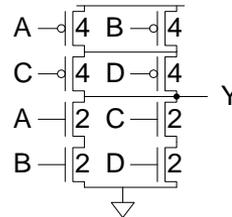
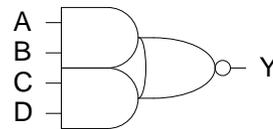
$$g_B = 6/3$$

$$g_C = 5/3$$

$$p = 7/3$$

AOI22

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



$$g_A = 6/3$$

$$g_B = 6/3$$

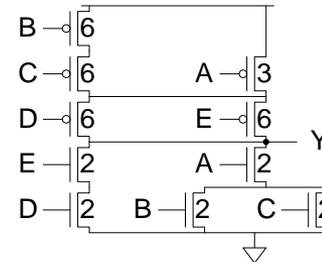
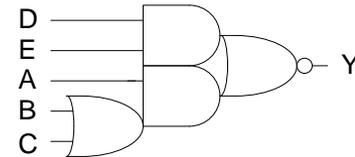
$$g_C = 6/3$$

$$g_D = 6/3$$

$$p = 12/3$$

Complex AOI

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



$$g_A = 5/3$$

$$g_B = 8/3$$

$$g_C = 8/3$$

$$g_D = 8/3$$

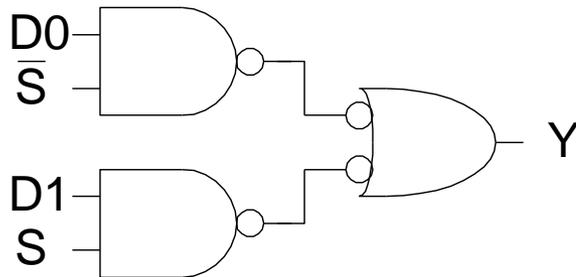
$$g_E = 8/3$$

$$p = 16/3$$

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 two designs.

$$H = 160 / 16 = 10 \quad B = 1 \quad N = 2$$



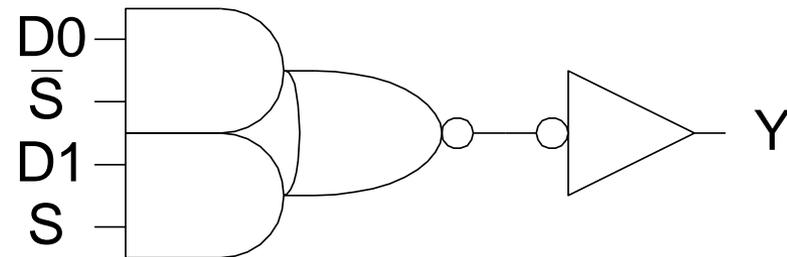
$$P = 2 + 2 = 4$$

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

$$F = GBH = 160/9$$

$$\hat{f} = \sqrt[3]{F} = 4.2$$

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



$$P = 4 + 1 = 5$$

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

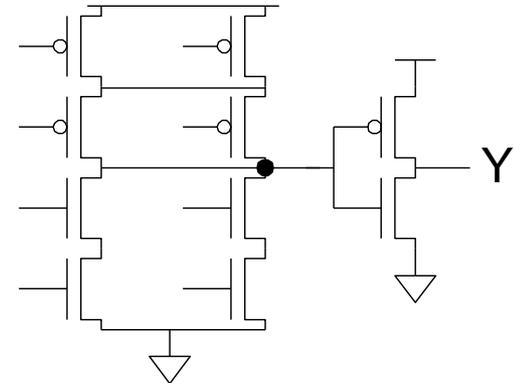
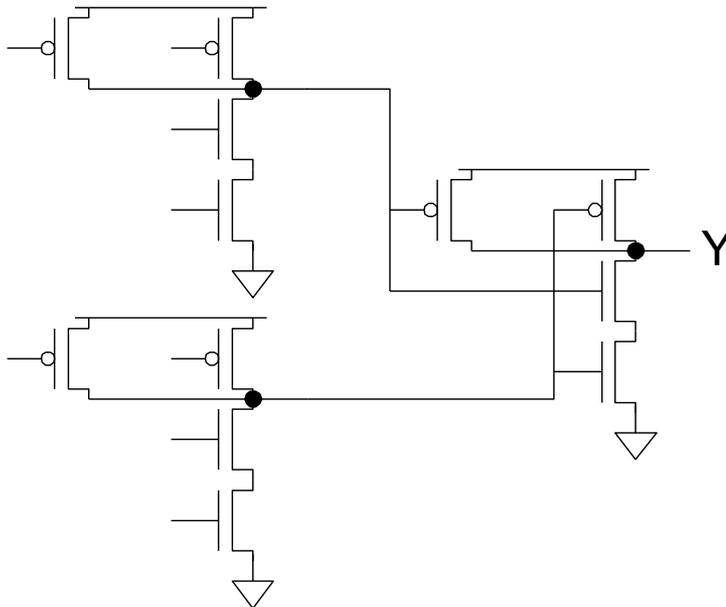
$$F = GBH = 20$$

$$\hat{f} = \sqrt[3]{F} = 4.5$$

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

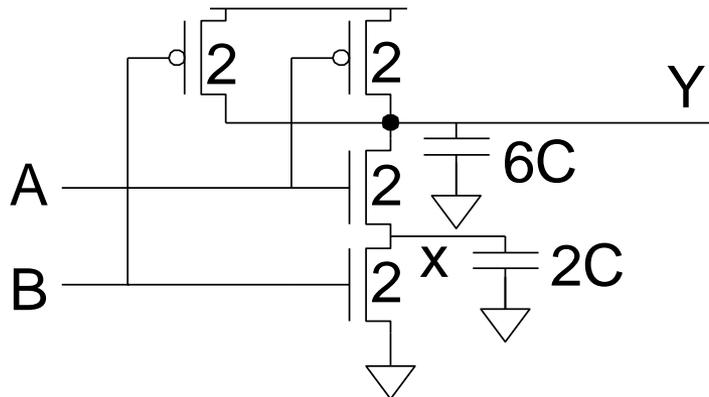
Example 5

- Annotate your designs with transistor sizes that achieve this delay.



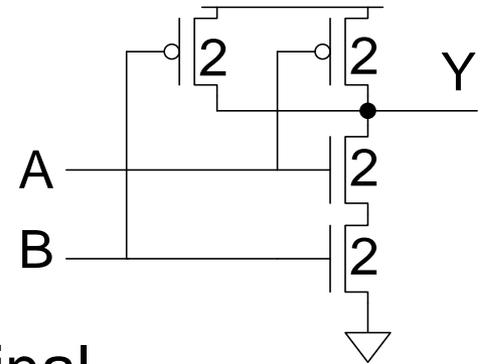
Input Order

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



Inner & Outer Inputs

- ❑ *Inner* input is closest to output (A)
- ❑ *Outer* input is closest to rail (B)
- ❑ If input arrival time is known
 - Connect latest input to inner terminal



Best P/N Ratio

- ❑ We have selected P/N ratio for unit rise and fall resistance ($\mu = 2-3$ for an inverter).
- ❑ Alternative: choose ratio for least average delay
- ❑ Ex: inverter

- Delay driving identical inverter

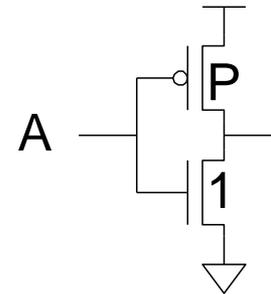
- $t_{pdf} = (P+1)$

- $t_{pdr} = (P+1)(\mu/P)$

- $t_{pd} = (P+1)(1+\mu/P)/2 = (P + 1 + \mu + \mu/P)/2$

- $dt_{pd} / dP = (1 - \mu/P^2)/2 = 0$

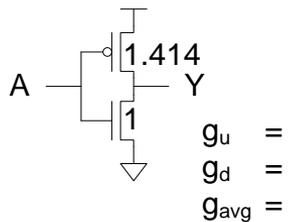
- Least delay for $P = \sqrt{\mu}$



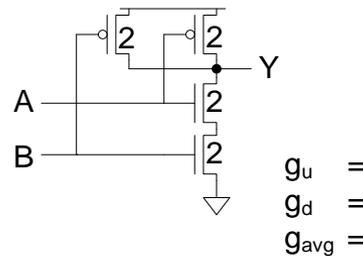
P/N Ratios

- 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

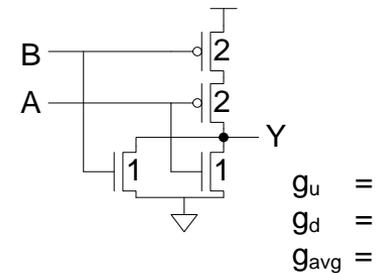
Inverter



NAND2

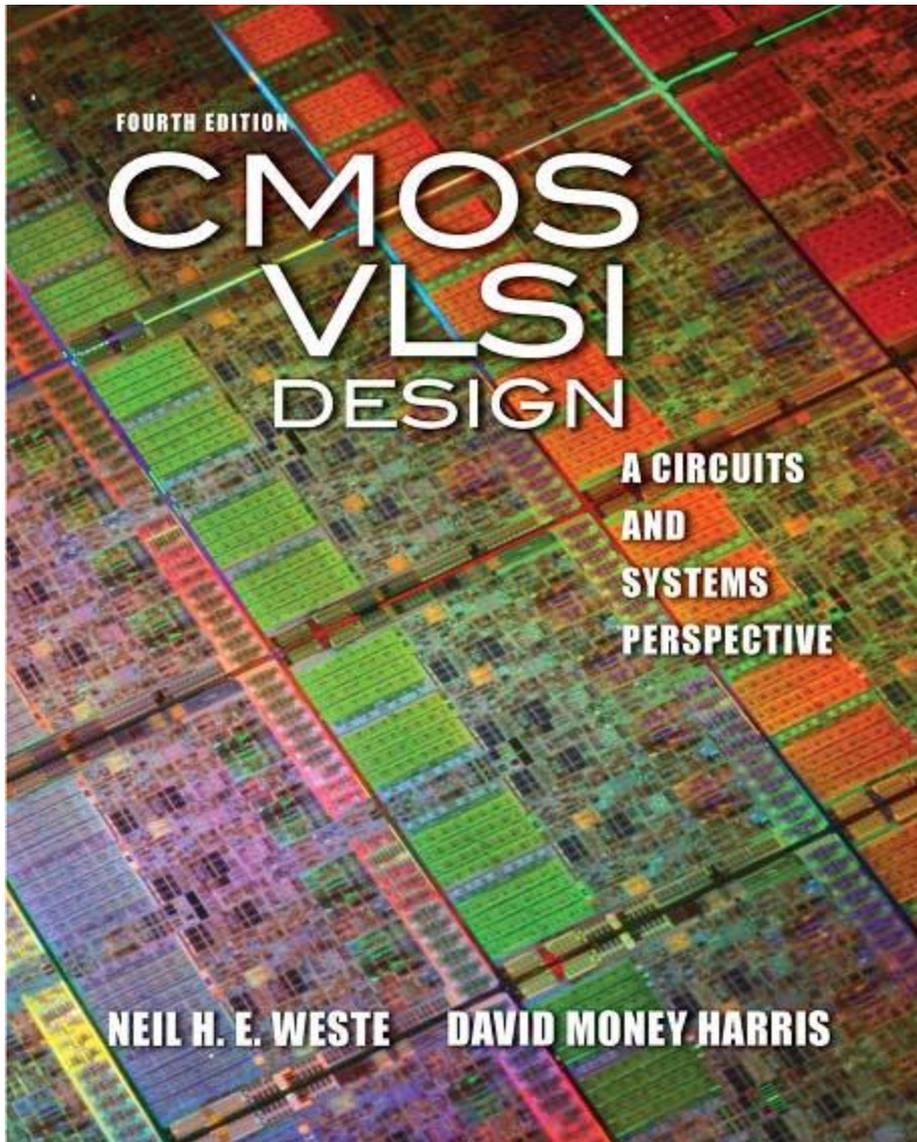


NOR2



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



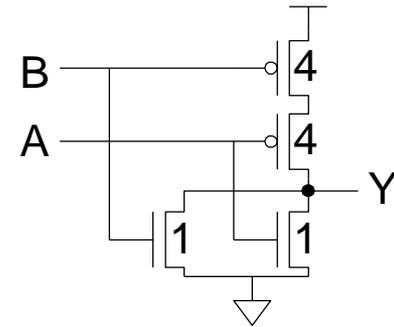
Lecture 10: Circuit Families

Outline

- Pseudo-nMOS Logic
- Dynamic Logic
- Pass Transistor Logic

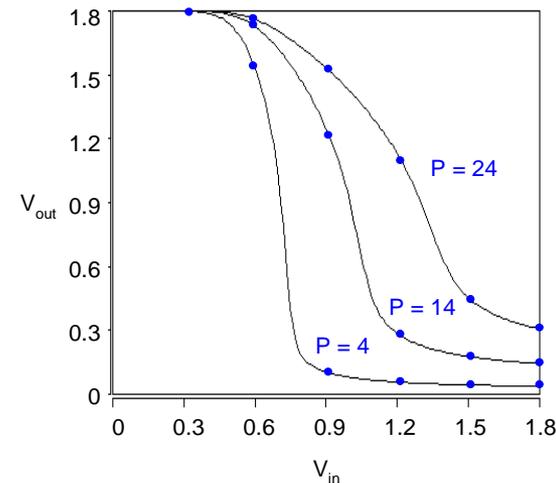
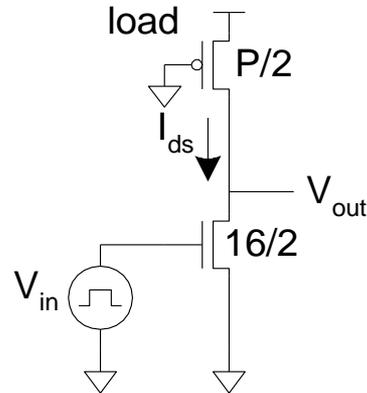
Introduction

- ❑ What makes a circuit fast?
 - $I = C \, dV/dt \quad \rightarrow \quad t_{pd} \propto (C/I) \, \Delta V$
 - low capacitance
 - high current
 - small swing
- ❑ Logical effort is proportional to C/I
- ❑ pMOS 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...



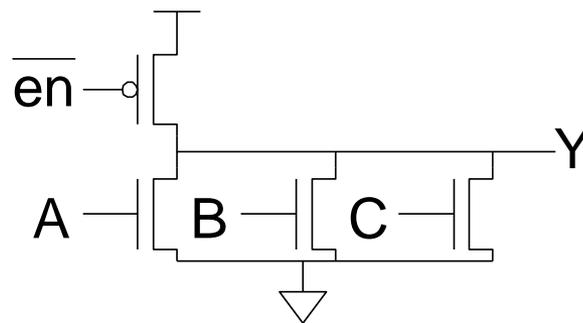
Pseudo-nMOS

- ❑ In the old days, nMOS processes had no pMOS
 - Instead, use pull-up transistor that is always ON
- ❑ In CMOS, use a pMOS that is always ON
 - *Ratio* issue
 - Make pMOS about $\frac{1}{4}$ effective strength of pulldown network



Pseudo-nMOS Power

- ❑ Pseudo-nMOS draws power whenever $Y = 0$
 - Called static power $P = I_{DD} V_{DD}$
 - A few mA / gate * 1M gates would be a problem
 - Explains why nMOS went extinct
- ❑ Use pseudo-nMOS sparingly for wide NORs
- ❑ Turn off pMOS when not in use



Ratio Example

- ❑ The chip contains a 32 word x 48 bit ROM
 - Uses pseudo-nMOS decoder and bitline pullups
 - On average, one wordline and 24 bitlines are high
- ❑ Find static power drawn by the ROM
 - $I_{\text{on-p}} = 36 \mu\text{A}$, $V_{\text{DD}} = 1.0 \text{ V}$

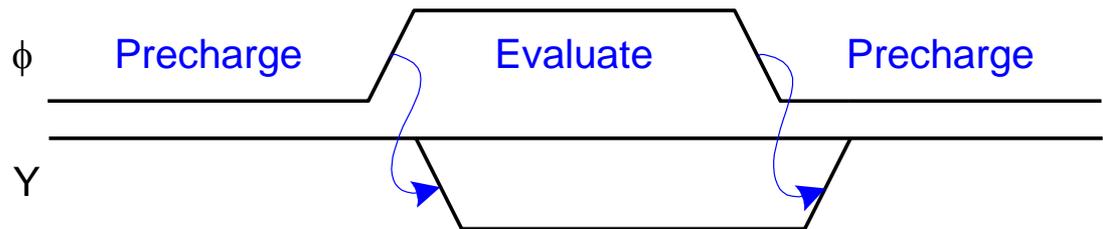
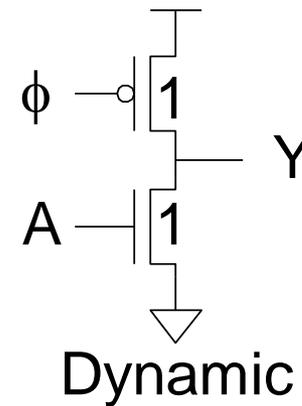
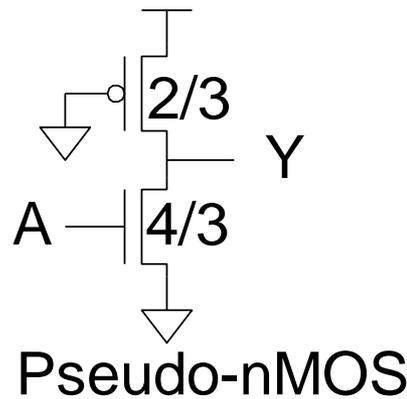
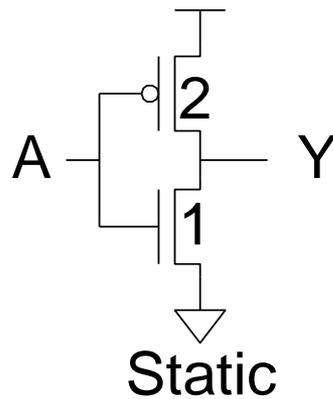
- ❑ Solution:

$$P_{\text{pull-up}} = V_{\text{DD}} I_{\text{pull-up}} = 36 \mu\text{W}$$

$$P_{\text{static}} = (31 + 24) P_{\text{pull-up}} = 1.98 \text{ mW}$$

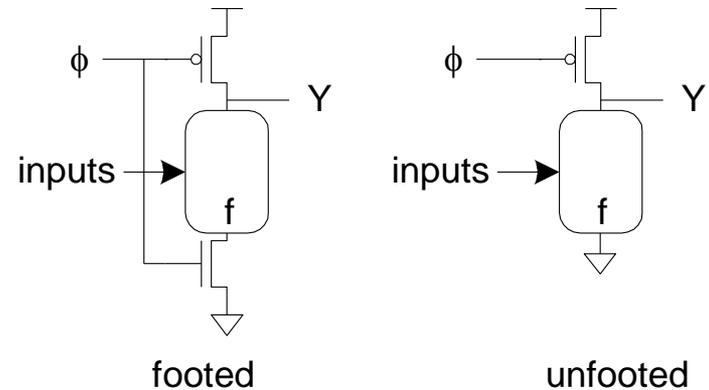
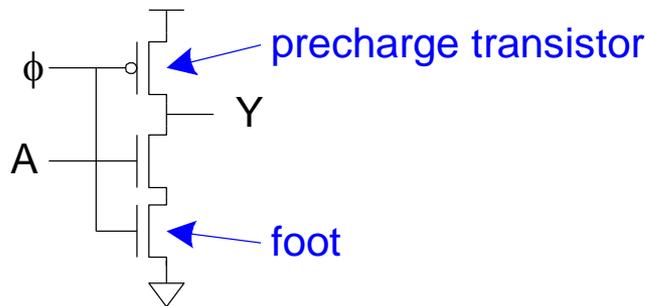
Dynamic Logic

- ❑ *Dynamic* gates uses a clocked pMOS pullup
- ❑ Two modes: *precharge* and *evaluate*



The Foot

- ❑ What if pulldown network is ON during precharge?
- ❑ Use series evaluation transistor to prevent fight.



Monotonicity

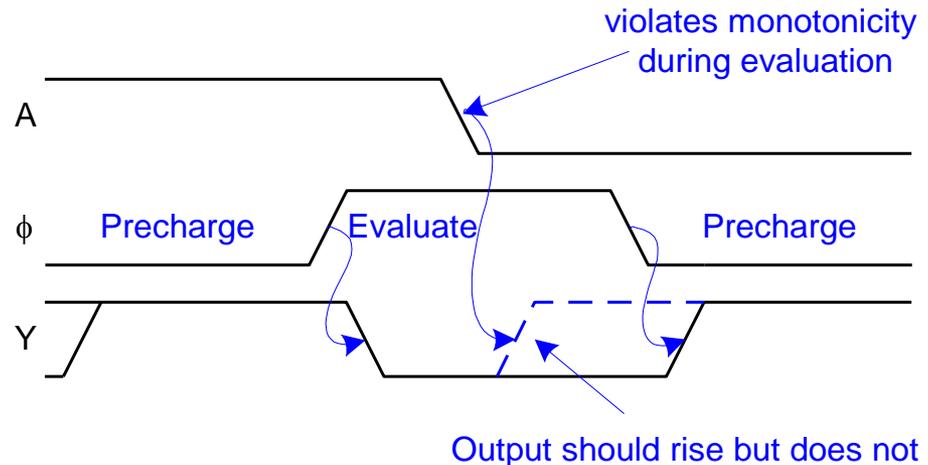
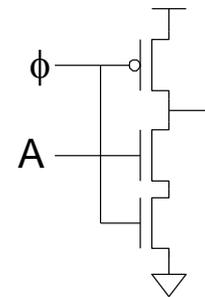
❑ Dynamic gates require *monotonically rising* inputs during evaluation

– 0 -> 0

– 0 -> 1

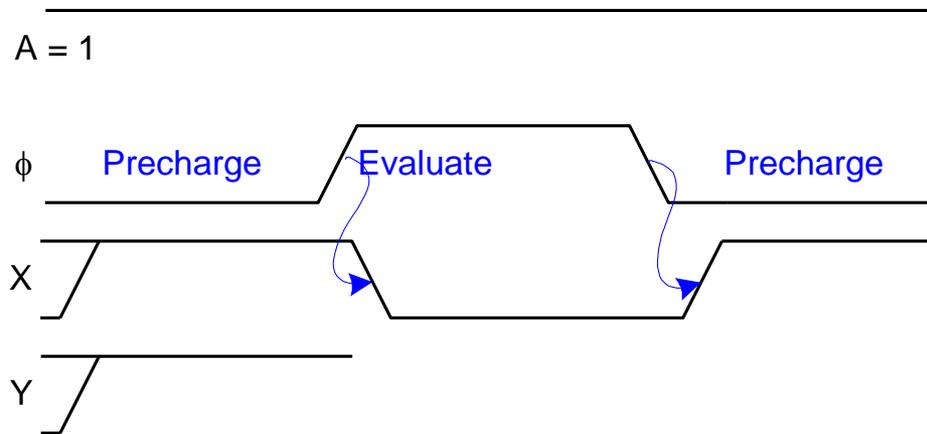
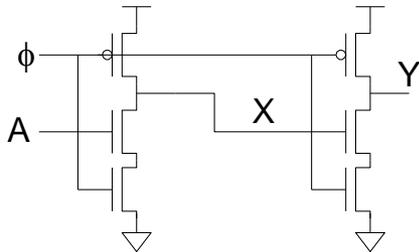
– 1 -> 1

– But not 1 -> 0



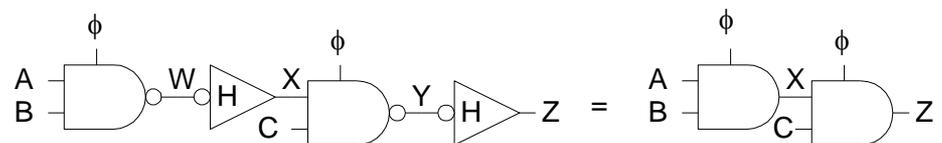
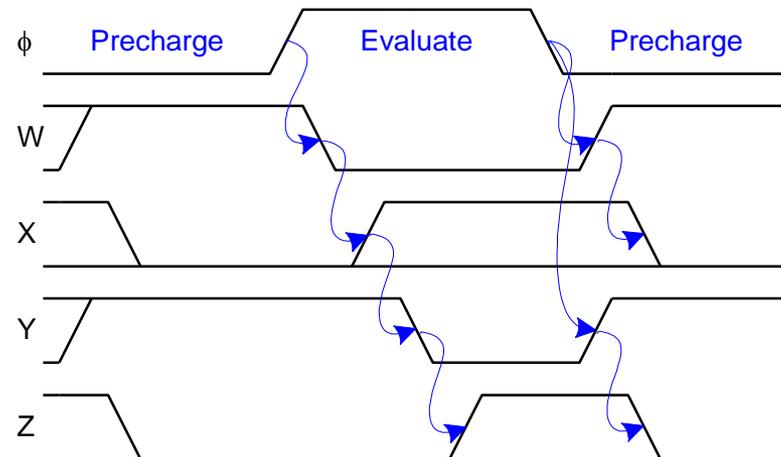
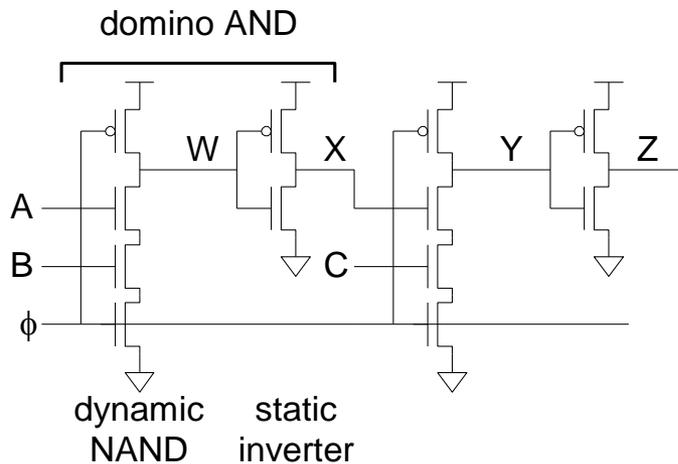
Monotonicity Woes

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



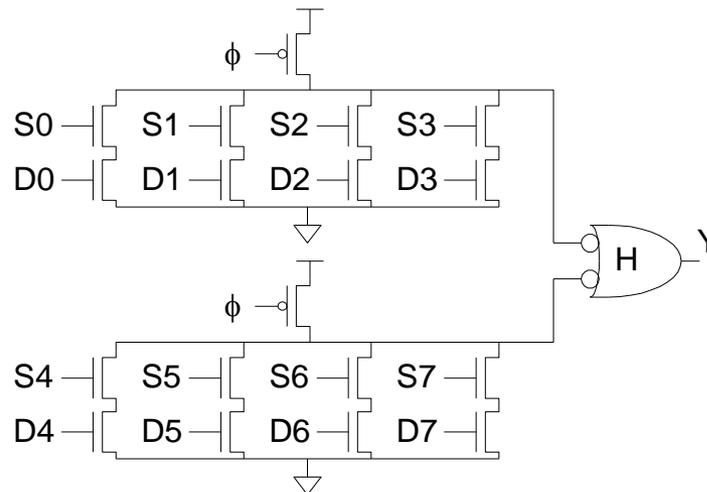
Domino Gates

- ❑ Follow dynamic stage with inverting static gate
 - Dynamic / static pair is called domino gate
 - Produces monotonic outputs



Domino Optimizations

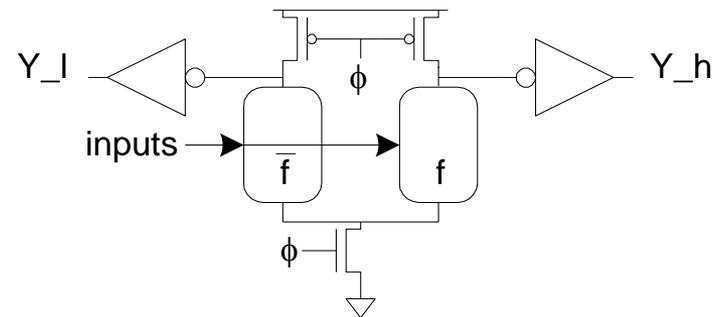
- ❑ Each domino gate triggers next one, like a string of dominos toppling over
- ❑ Gates evaluate sequentially but precharge in parallel
- ❑ Thus evaluation is more critical than precharge
- ❑ HI-skewed static stages can perform logic



Dual-Rail Domino

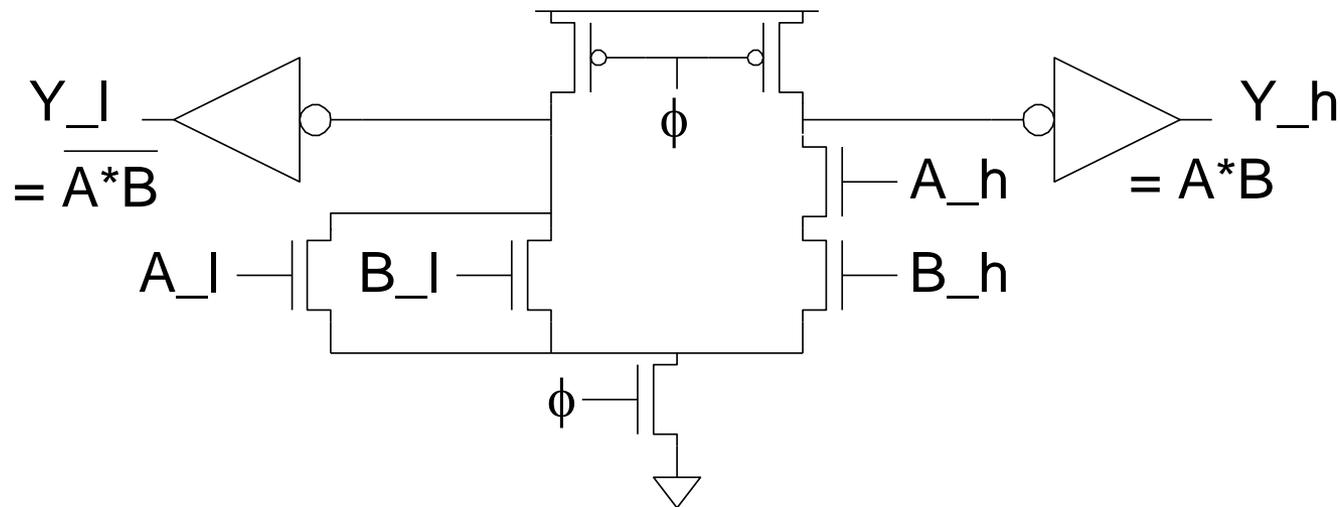
- ❑ Domino only performs noninverting functions:
 - AND, OR but not NAND, NOR, or XOR
- ❑ Dual-rail domino solves this problem
 - Takes true and complementary inputs
 - Produces true and complementary outputs

sig_h	sig_l	Meaning
0	0	Precharged
0	1	'0'
1	0	'1'
1	1	invalid



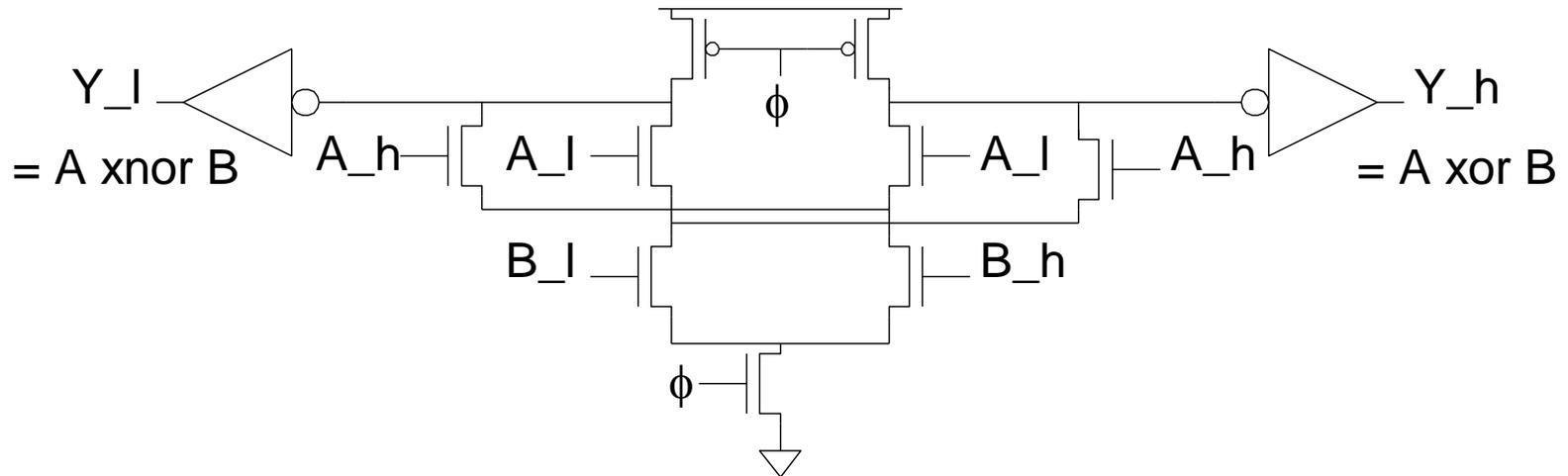
Example: AND/NAND

- ❑ Given A_h, A_l, B_h, B_l
- ❑ Compute $Y_h = AB, Y_l = \overline{AB}$
- ❑ Pulldown networks are conduction complements



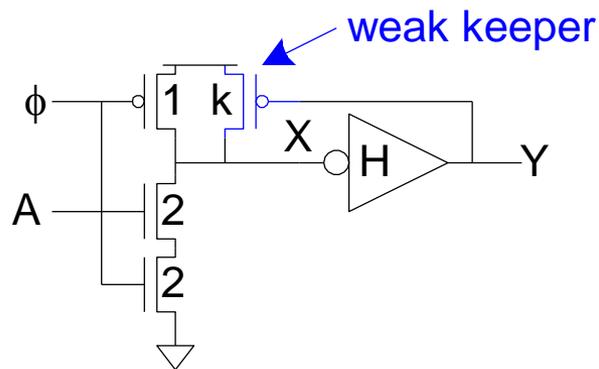
Example: XOR/XNOR

- Sometimes possible to share transistors



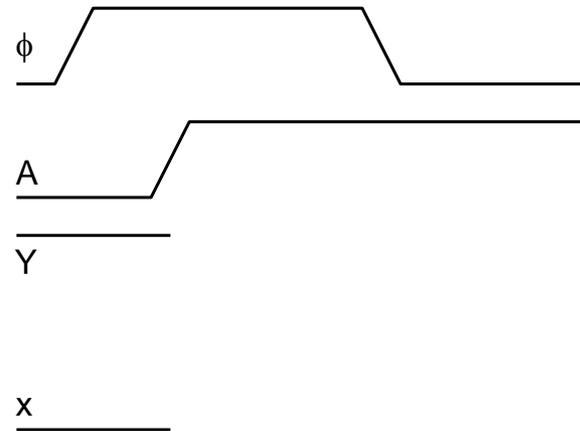
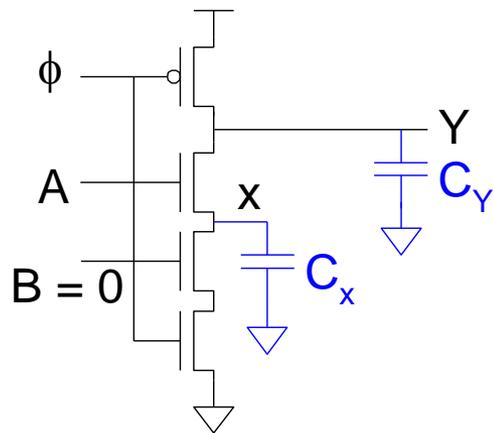
Leakage

- ❑ Dynamic node floats high during evaluation
 - Transistors are leaky ($I_{OFF} \neq 0$)
 - Dynamic value will leak away over time
 - Formerly milliseconds, now nanoseconds
- ❑ Use keeper to hold dynamic node
 - Must be weak enough not to fight evaluation



Charge Sharing

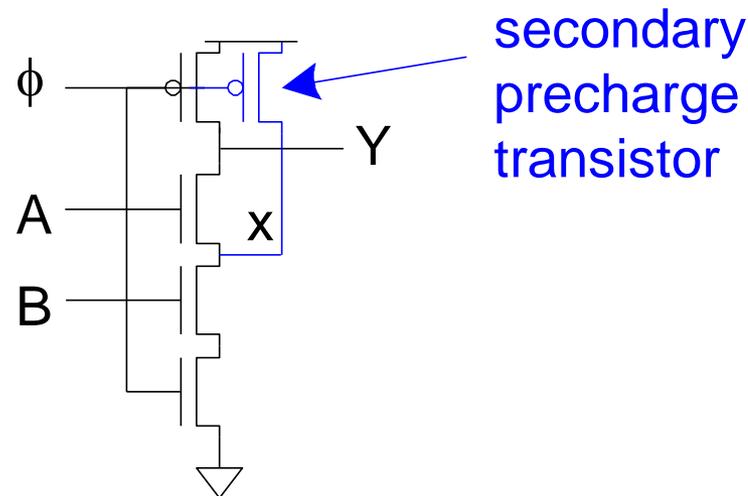
- ❑ Dynamic gates suffer from charge sharing



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

Secondary Precharge

- ❑ Solution: add secondary precharge transistors
 - Typically need to precharge every other node
- ❑ Big load capacitance C_Y helps as well



Noise Sensitivity

- ❑ Dynamic gates are very sensitive to noise
 - Inputs: $V_{IH} \approx V_{tn}$
 - Outputs: floating output susceptible noise
- ❑ Noise sources
 - Capacitive crosstalk
 - Charge sharing
 - Power supply noise
 - Feedthrough noise
 - And more!

Power

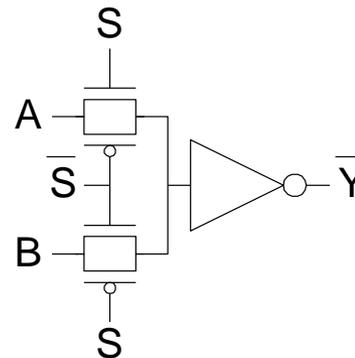
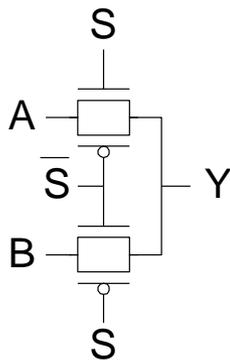
- ❑ Domino gates have high activity factors
 - Output evaluates and precharges
 - If output probability = 0.5, $\alpha = 0.5$
 - Output rises and falls on half the cycles
 - Clocked transistors have $\alpha = 1$
- ❑ Leads to very high power consumption

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 king
- ❑ Largely displaced by static CMOS now that power is the limiter
- ❑ Still used in memories for area efficiency

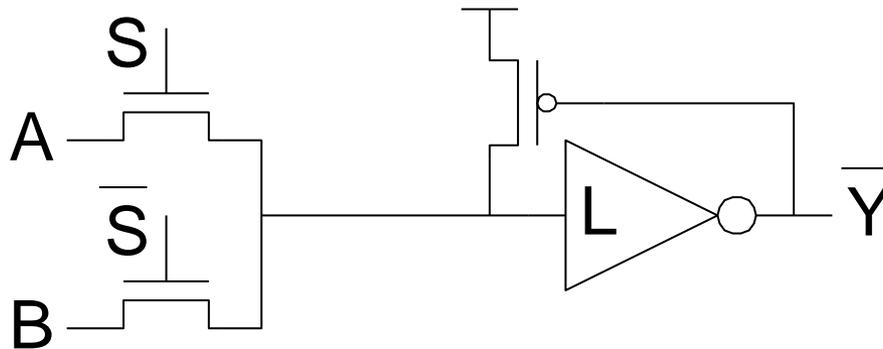
Pass Transistor Circuits

- ❑ Use pass transistors like switches to do logic
- ❑ Inputs drive diffusion terminals as well as gates
- ❑ CMOS + Transmission Gates:
 - 2-input multiplexer
 - Gates should be restoring



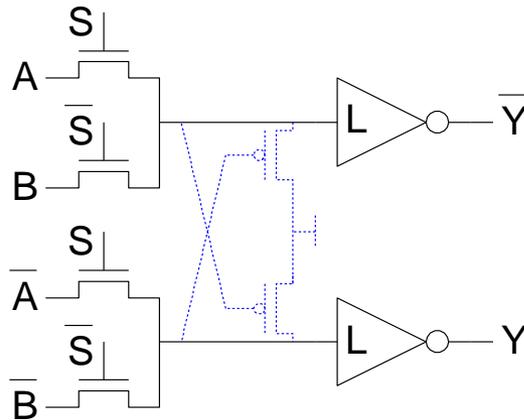
LEAP

- ❑ **LEA**n integration with **P**ass transistors
- ❑ Get rid of pMOS transistors
 - Use weak pMOS feedback to pull fully high
 - Ratio constraint



CPL

- Complementary Pass-transistor Logic
 - Dual-rail form of pass transistor logic
 - Avoids need for ratioed feedback
 - Optional cross-coupling for rail-to-rail swing



Pass Transistor Summary

- ❑ Researchers investigated pass transistor logic for general purpose applications in the 1990's
 - Benefits over static CMOS were small or negative
 - No longer generally used
- ❑ However, pass transistors still have a niche in special circuits such as memories where they offer small size and the threshold drops can be managed