Lecture 4: Implementing Logic in CMOS

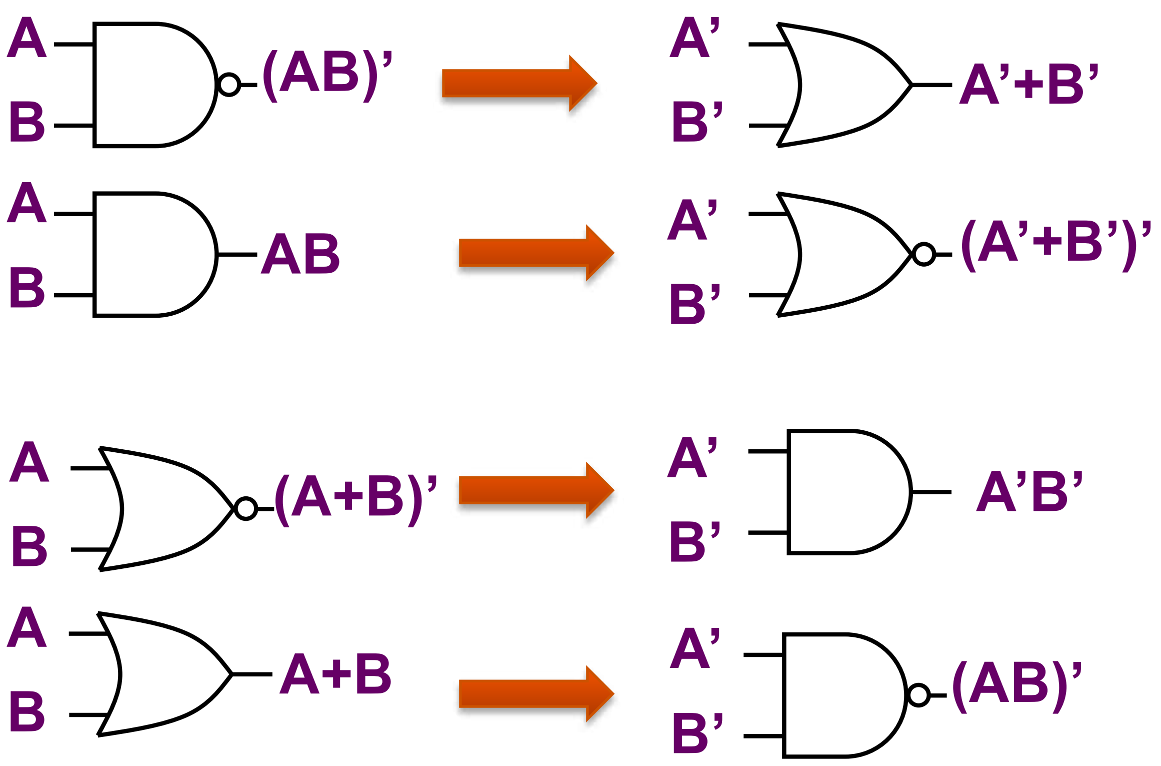
Mark McDermott
Electrical and Computer Engineering
The University of Texas at Austin

Review of DeMorgan’s Theorem

Recall that:

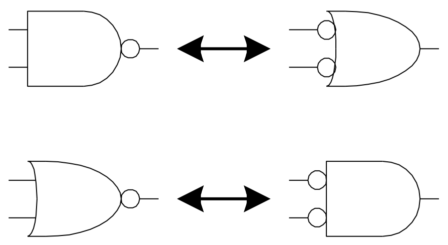
(AB)’ = A’+B’ and AB = (A’ + B’)’

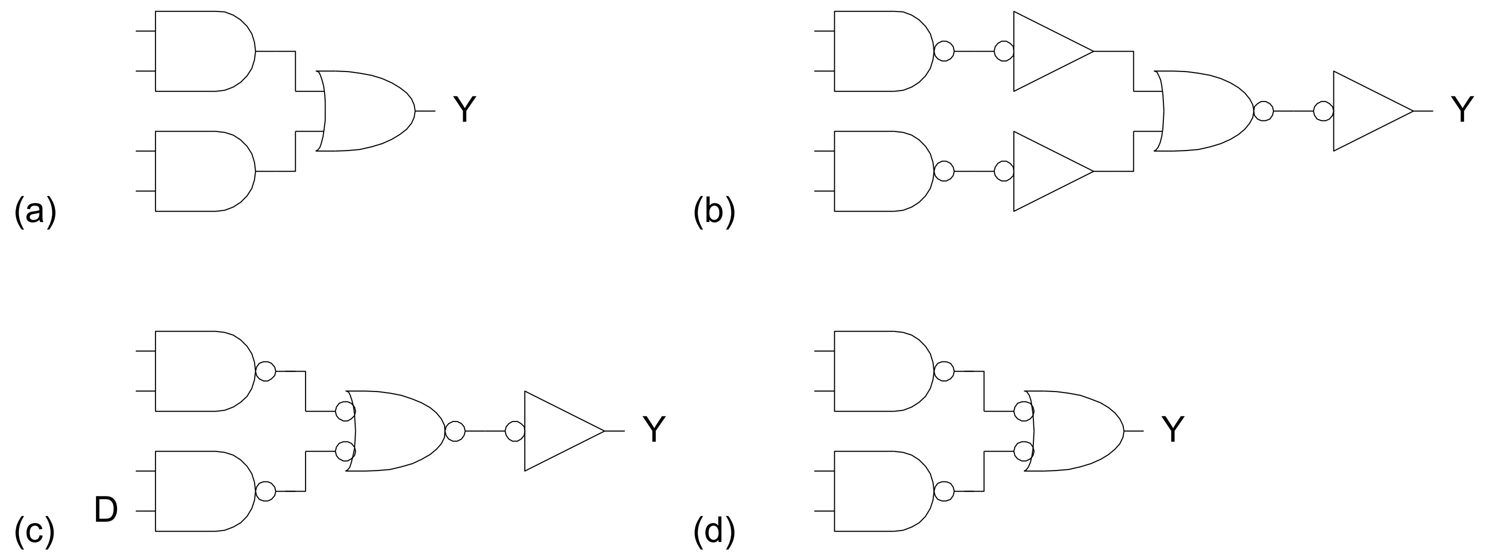
(A+B)’ = A’B’ and A+B = (A’B’)’



Bubble Pushing

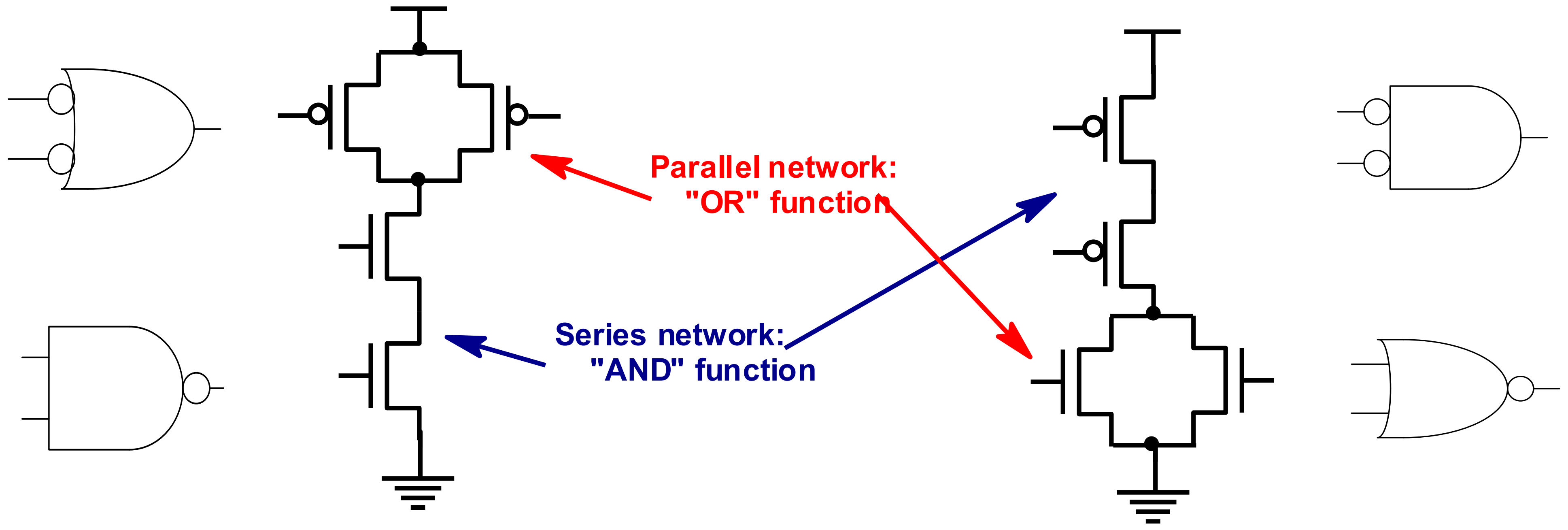
- Start with network of AND / OR gates
- Convert to NAND / NOR + inverters
- Push bubbles around to simplify logic





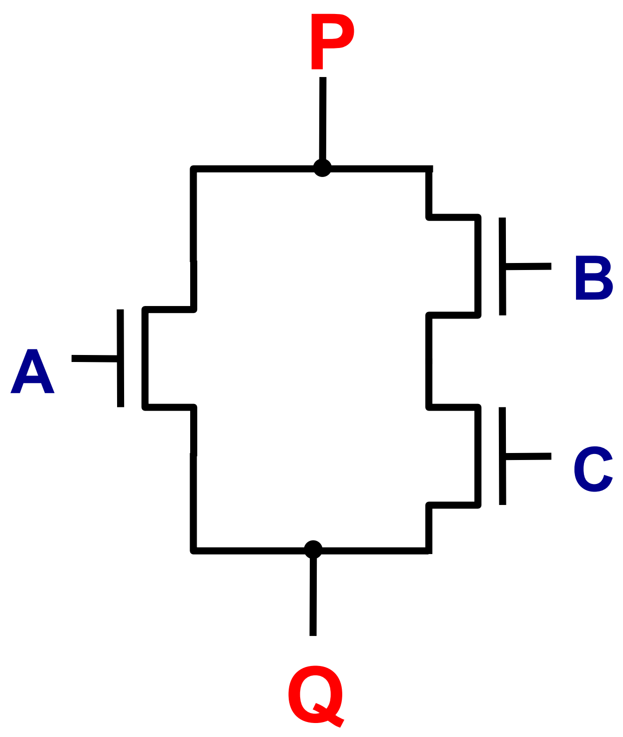
Static CMOS Circuits

- **N and P channel networks implement logic functions**
  - Each network connected between Output and VDD or VSS



Duality in CMOS Circuits

- N and P networks must implement complementary functions
- Duality is sufficient for correct operation

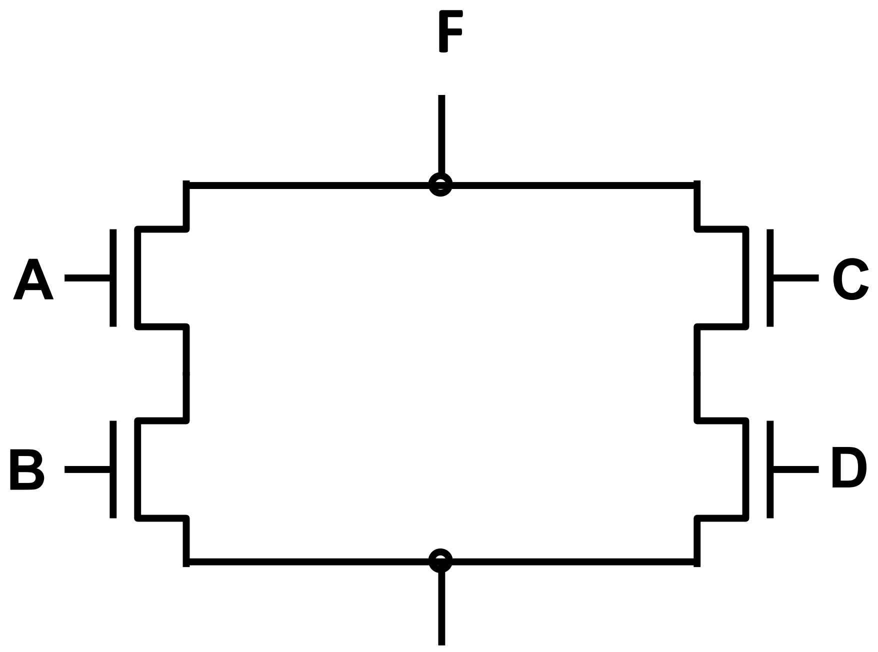


What are the values of A, B and C which will produce a connection between P and Q

A + B * C

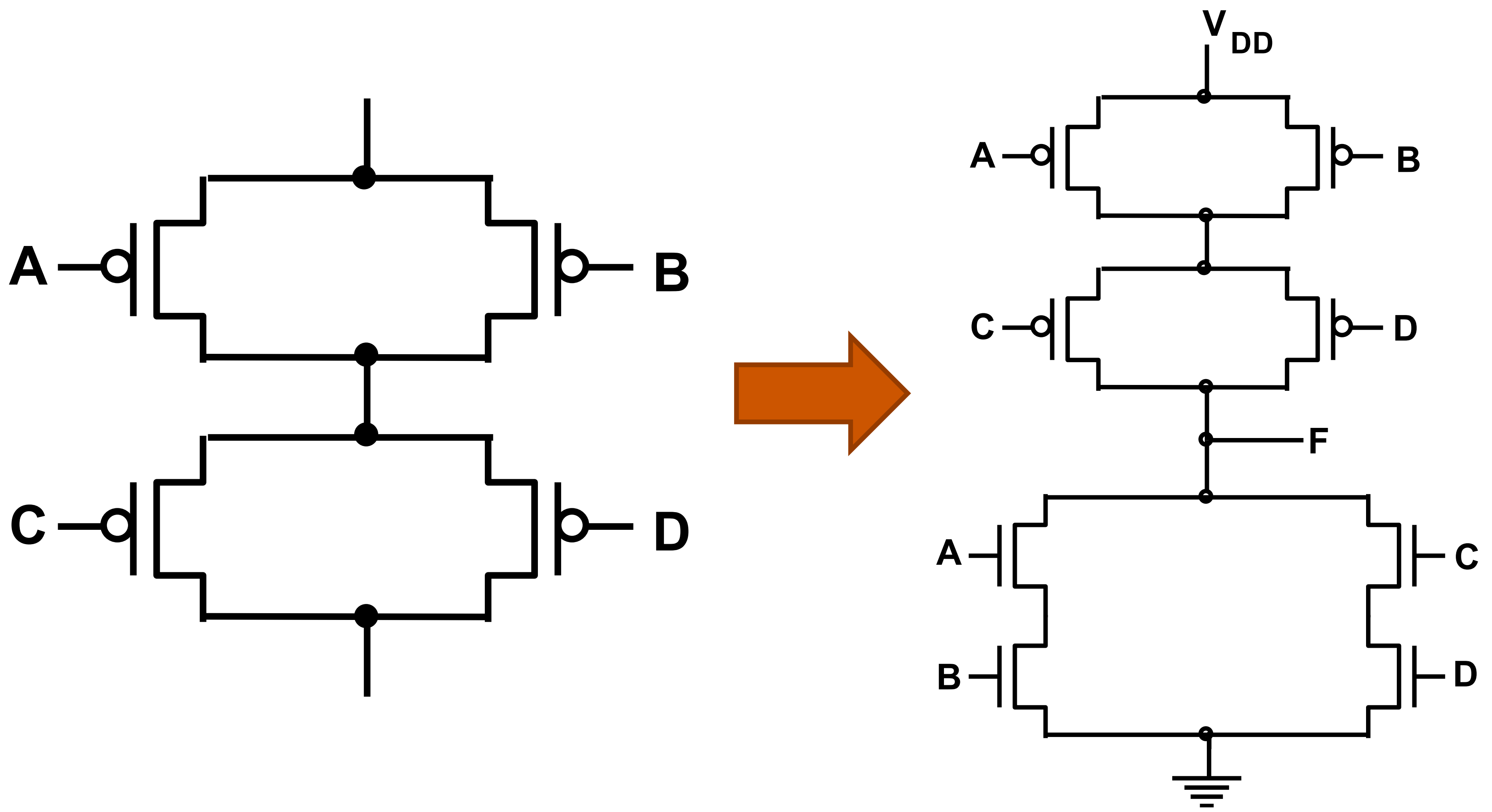
Constructing Complex Gates

- **Example: F = $\overline{(A * B) + (C * D)}$**
  - Take un-inverted function F = (AB + CD) and derive N-network
  - Identify AND, OR components; F is OR of AB,CD
  - Make connections of transistors
    - AND , Series connection, OR , Parallel

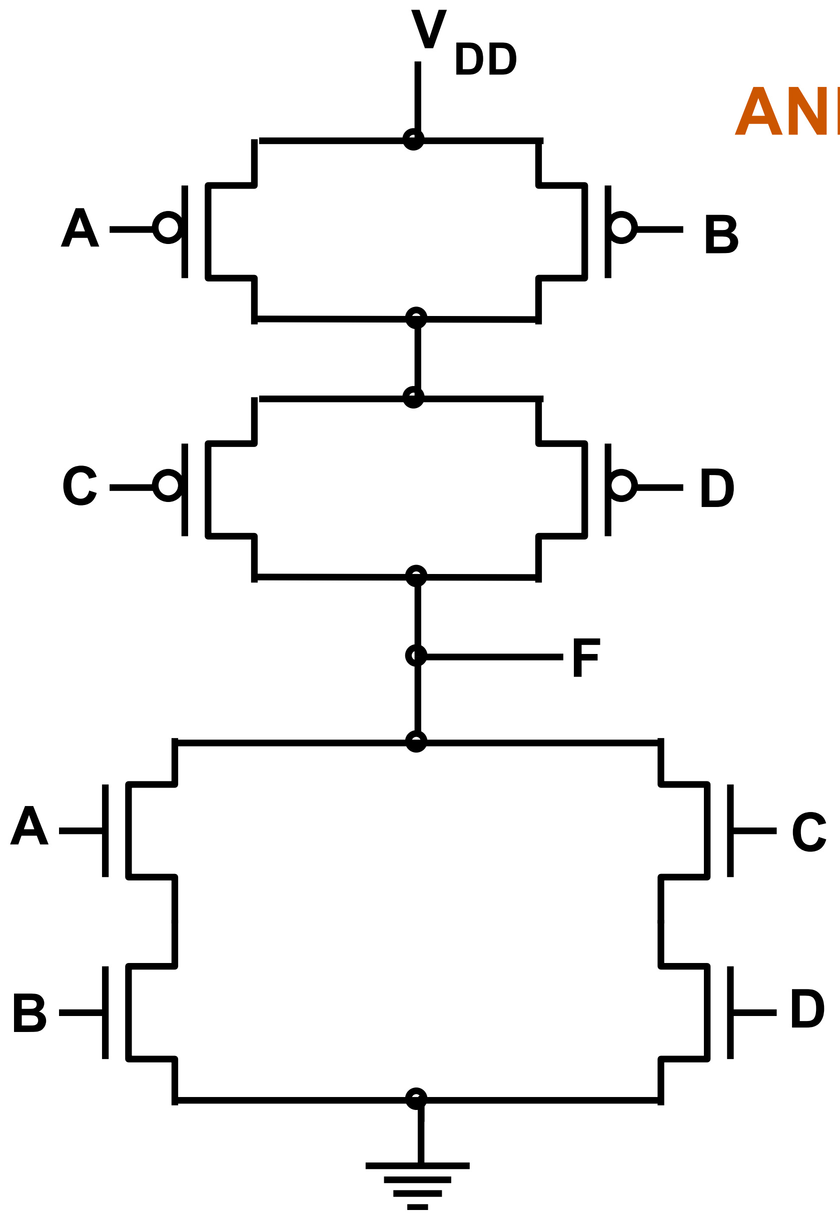


Construction of Complex Gates, Cont’d

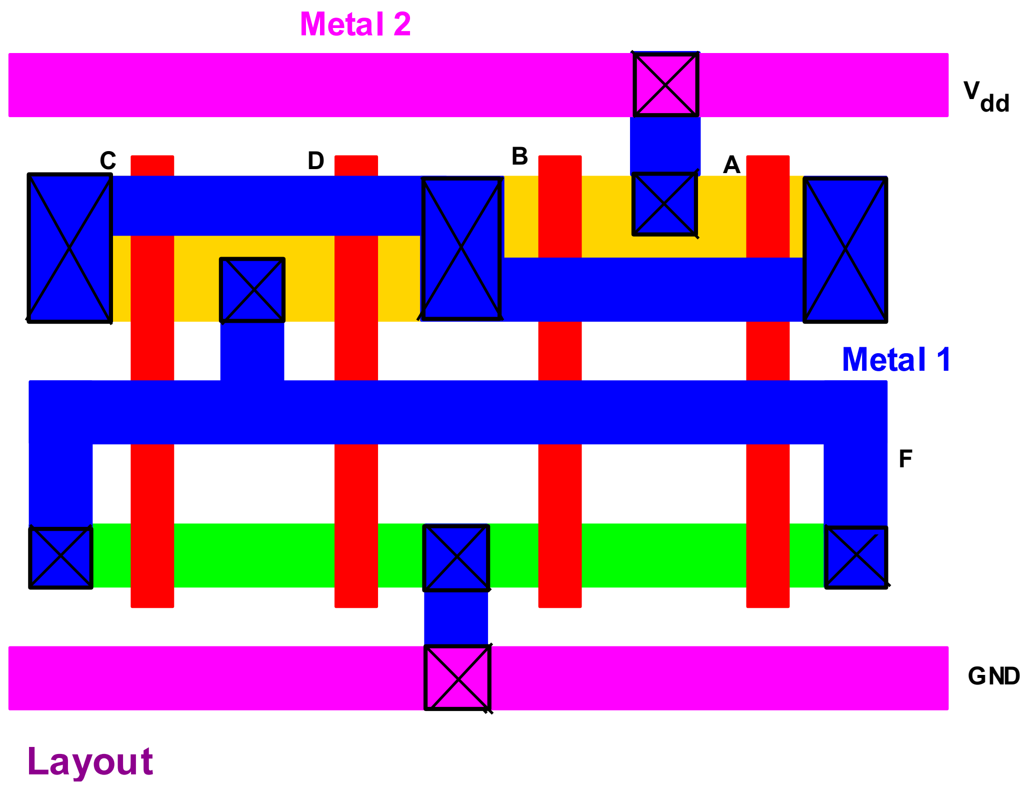
- Construct P-network by taking complement of N-expression (AB +CD), which gives the expression, $(\overline{A} + \overline{B}) * (\overline{C} + \overline{D})$
- Combine P and N circuits



Layout of Complex Gate

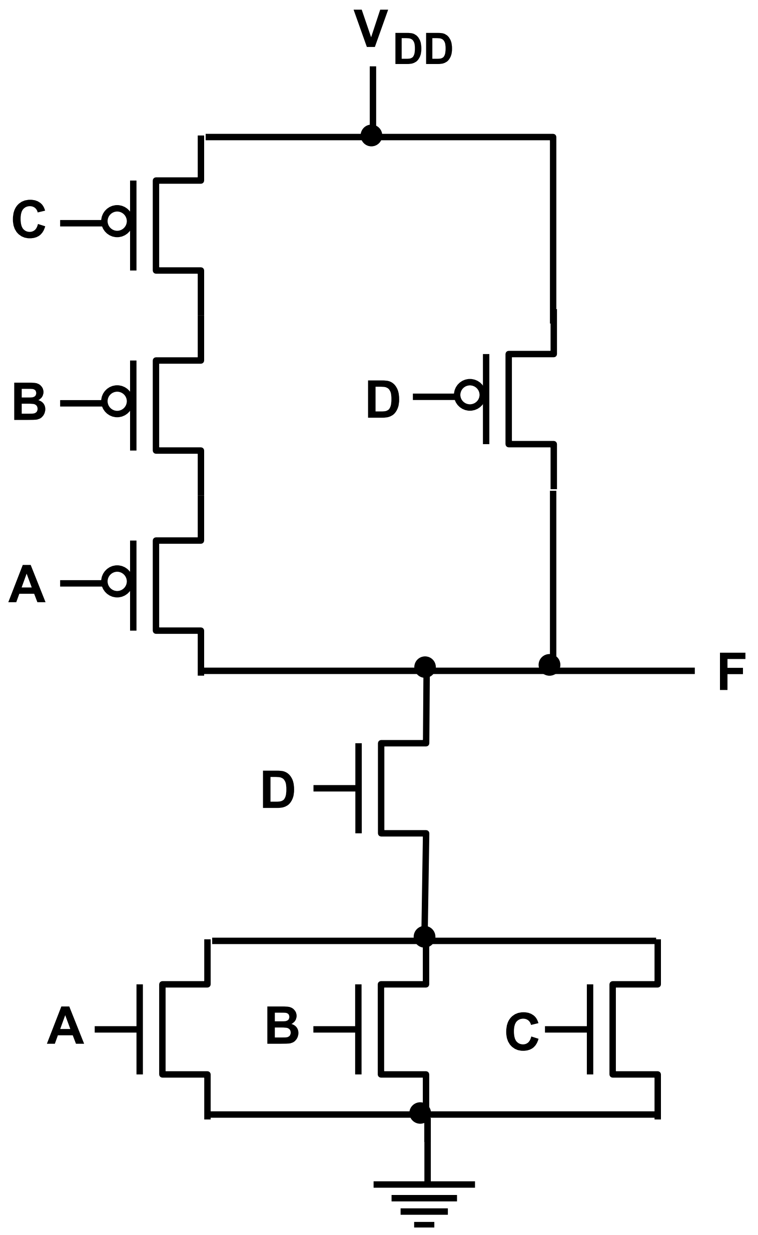


AND-OR-INVERT (AOI) gate

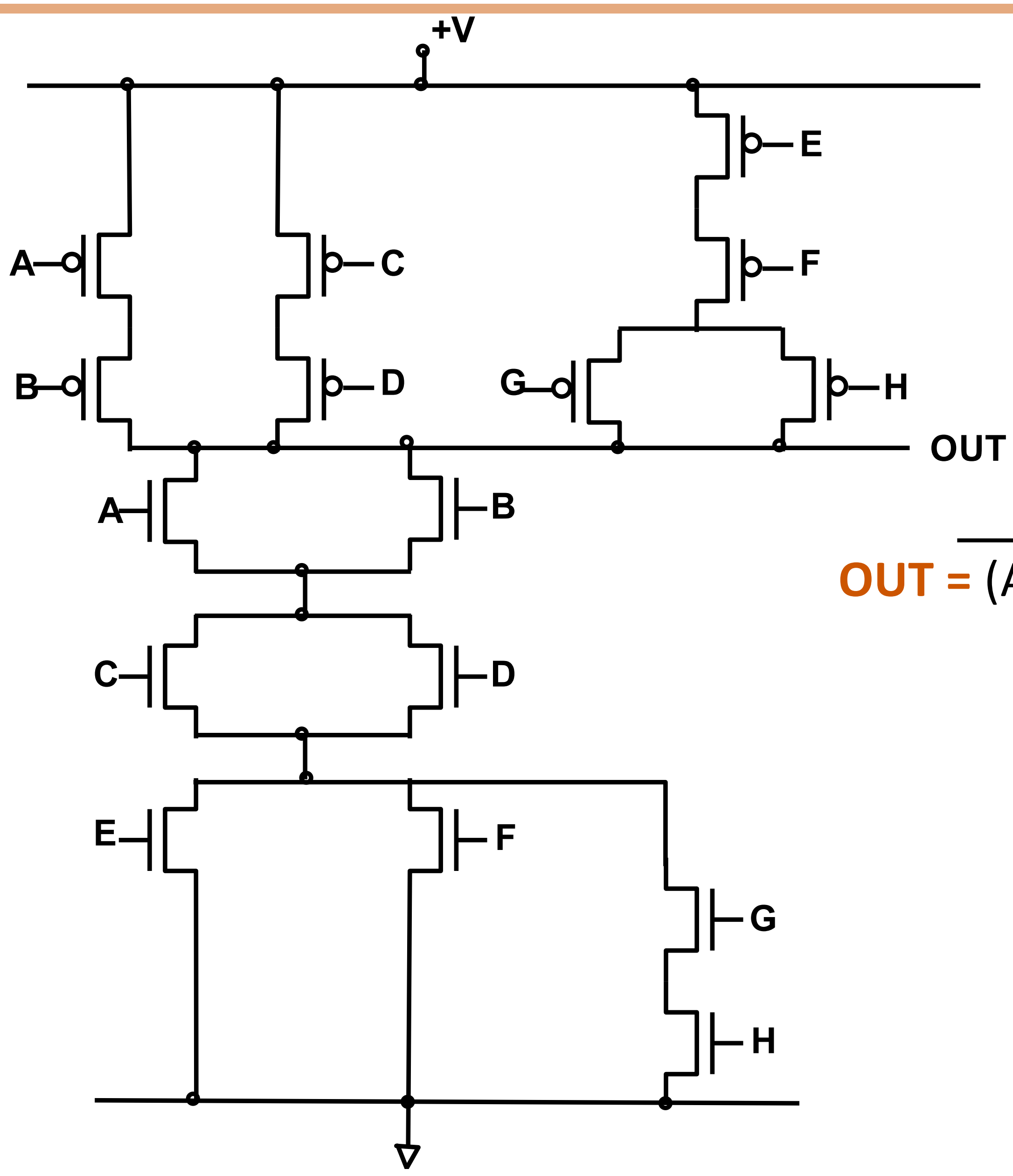


Example of Compound Gate

$$F = \overline{(A + B + C) * D}$$

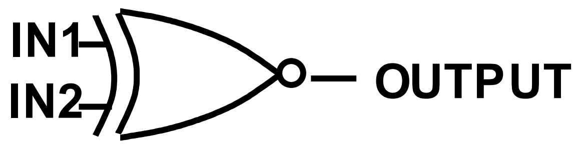


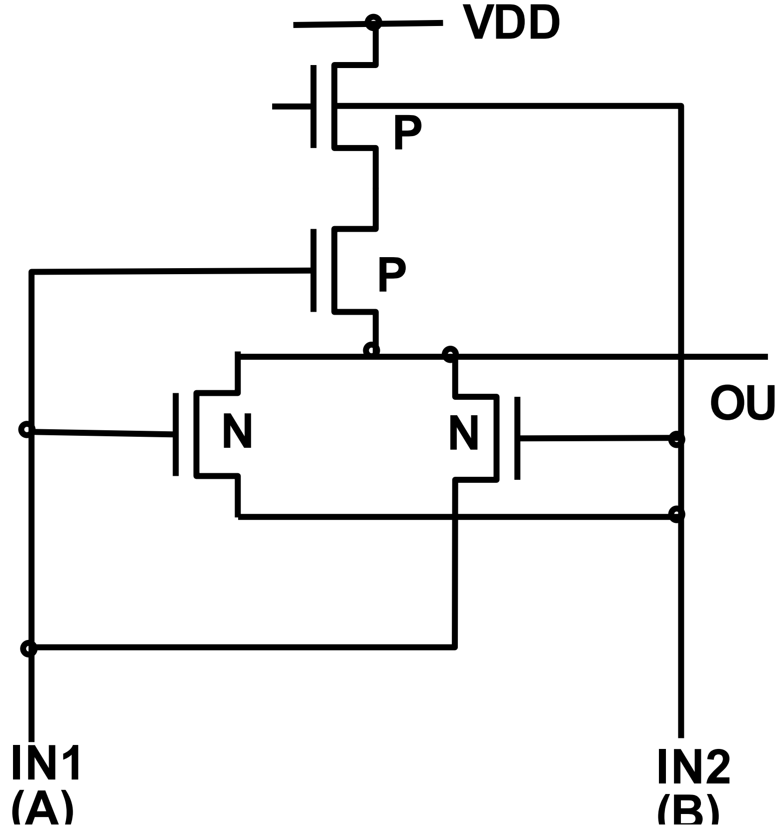
Example of More Complex Gate



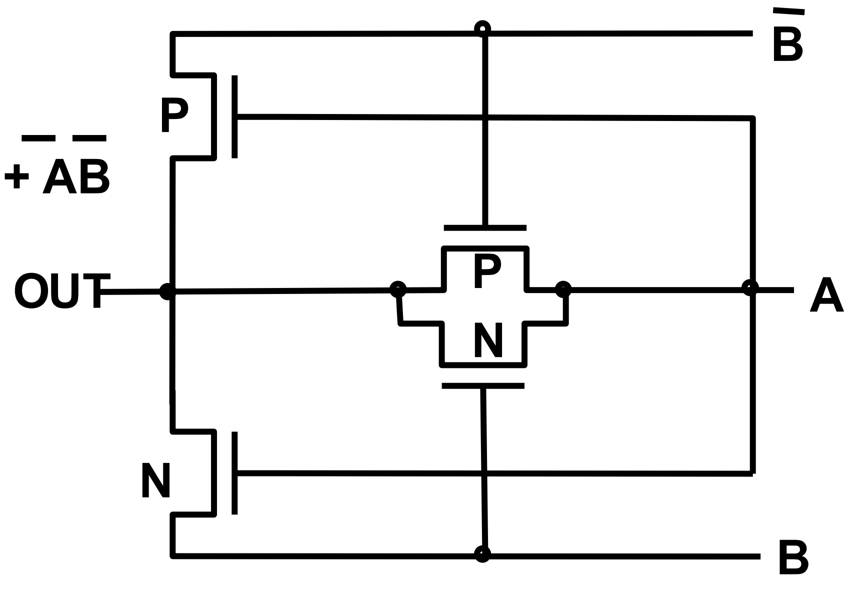
OUT = $\overline{(A+B)*(C+D)*(E+F+GH)}$

Exclusive-NOR Gate in CMOS

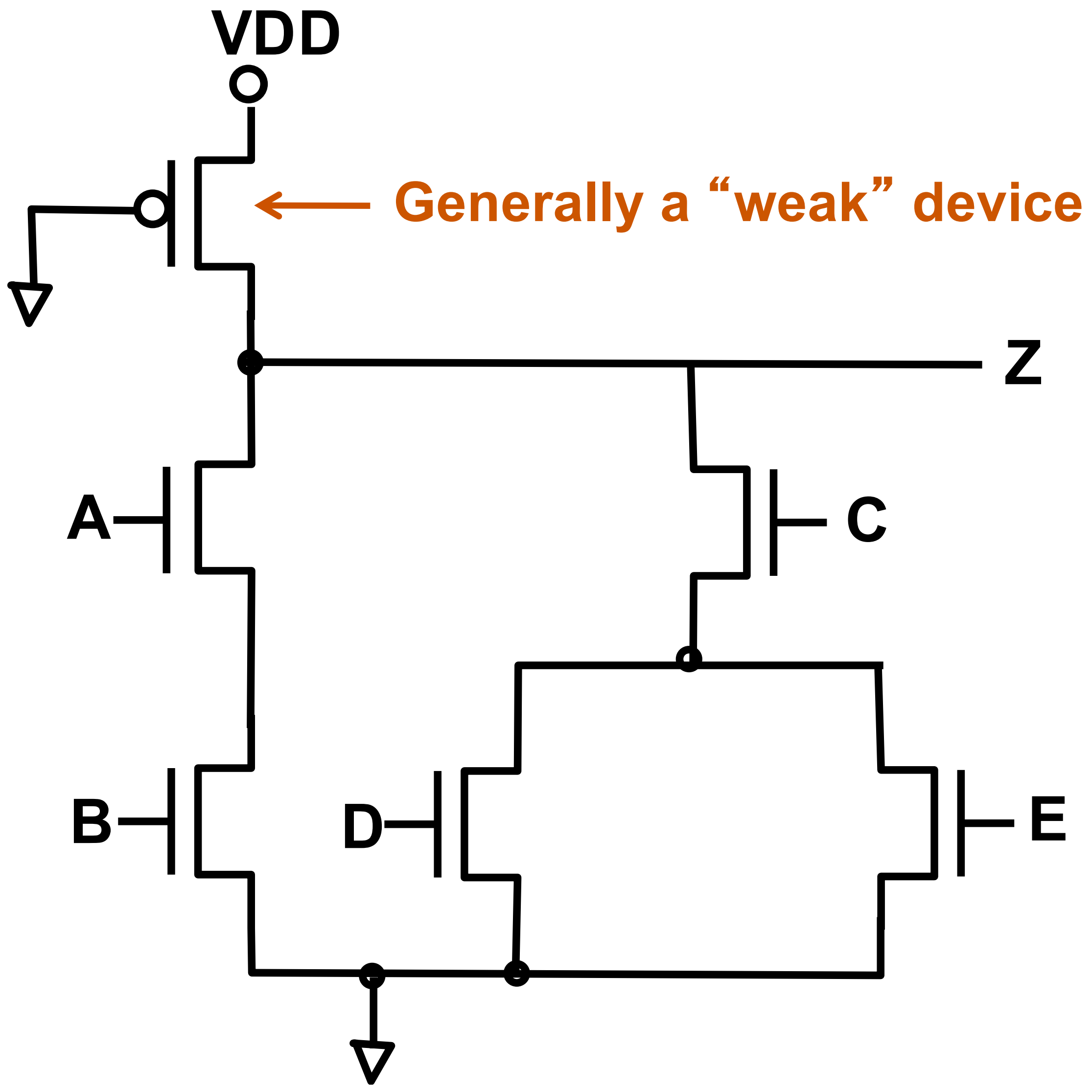




$OUTPUT = AB + \bar{A}\bar{B}$

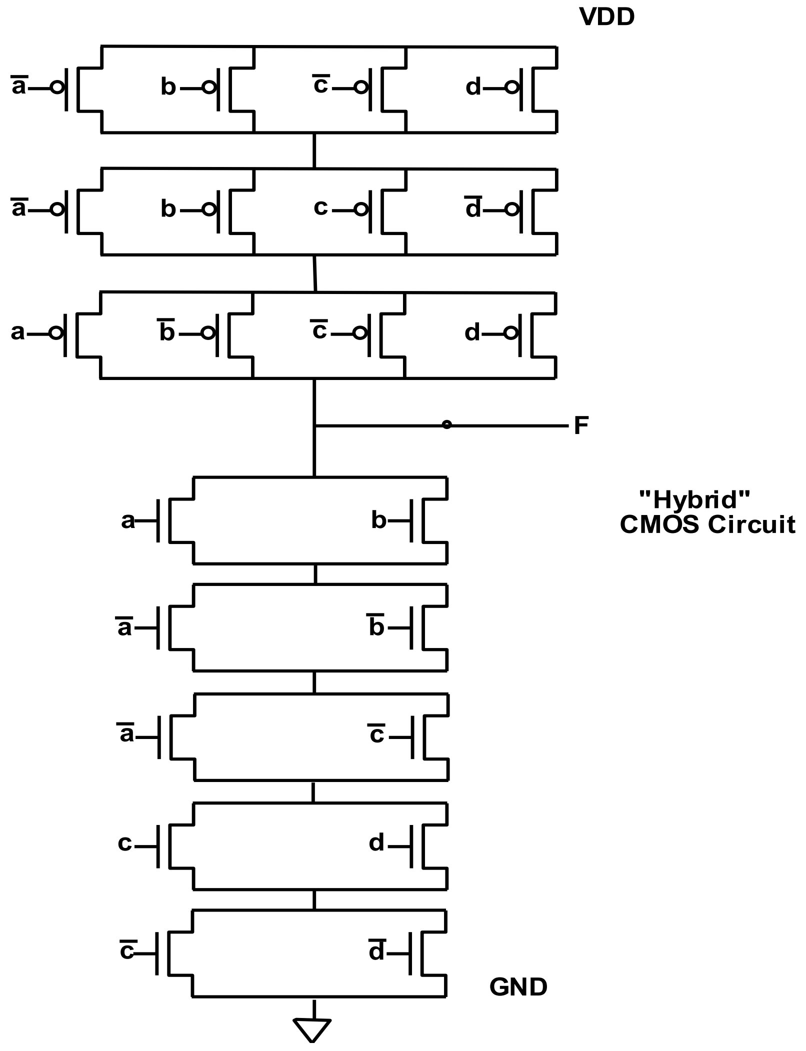


Pseudo nMOS Logic



Duality is not Necessary

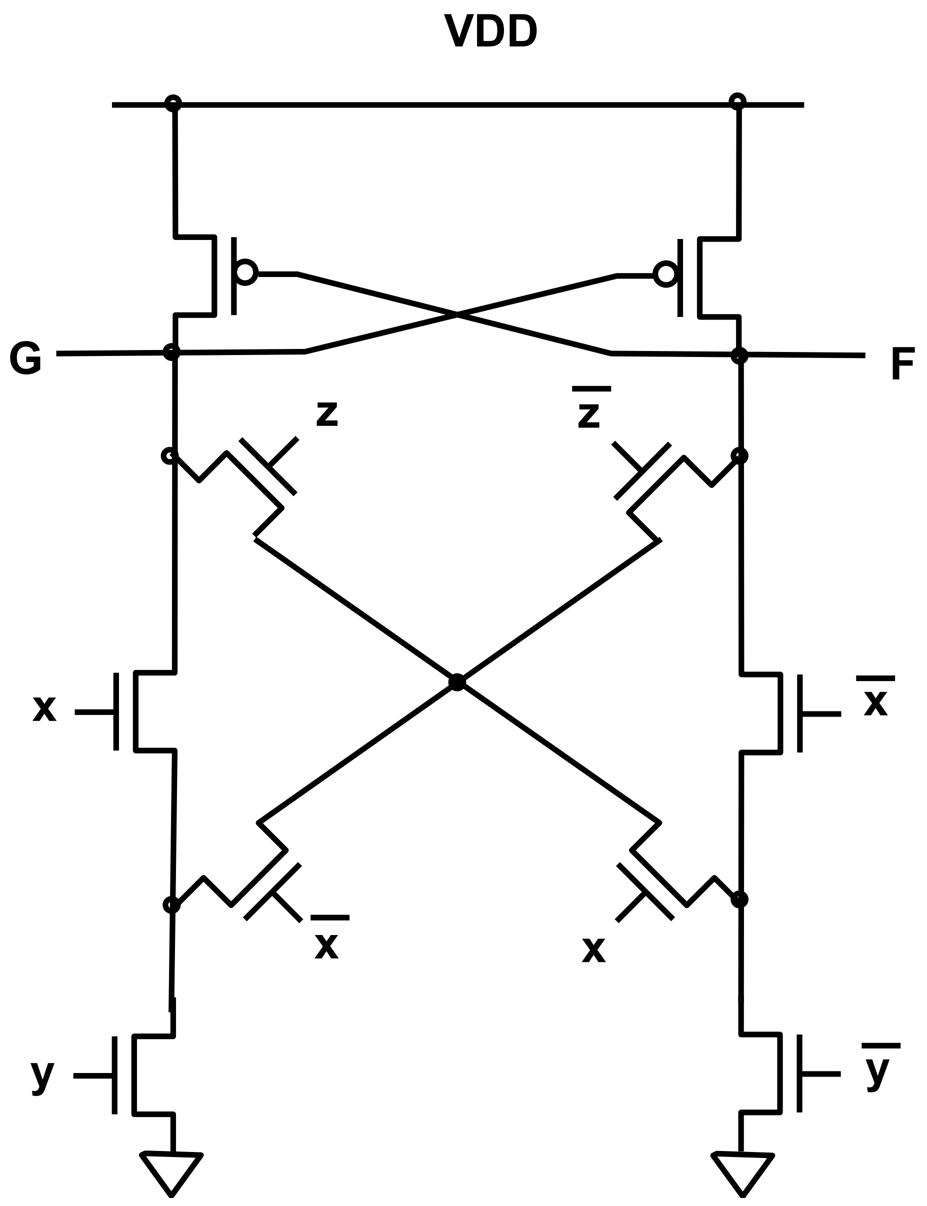
- Functions realized by N and P networks must be complementary, and one of them must conduct for every input combination



F = ab + a' b' + a'c' + cd + c' d'

The N and P networks are NOT duals, but the switching functions they implement are complementary

Example of "Dual Rail" Complex CMOS Gate



F =

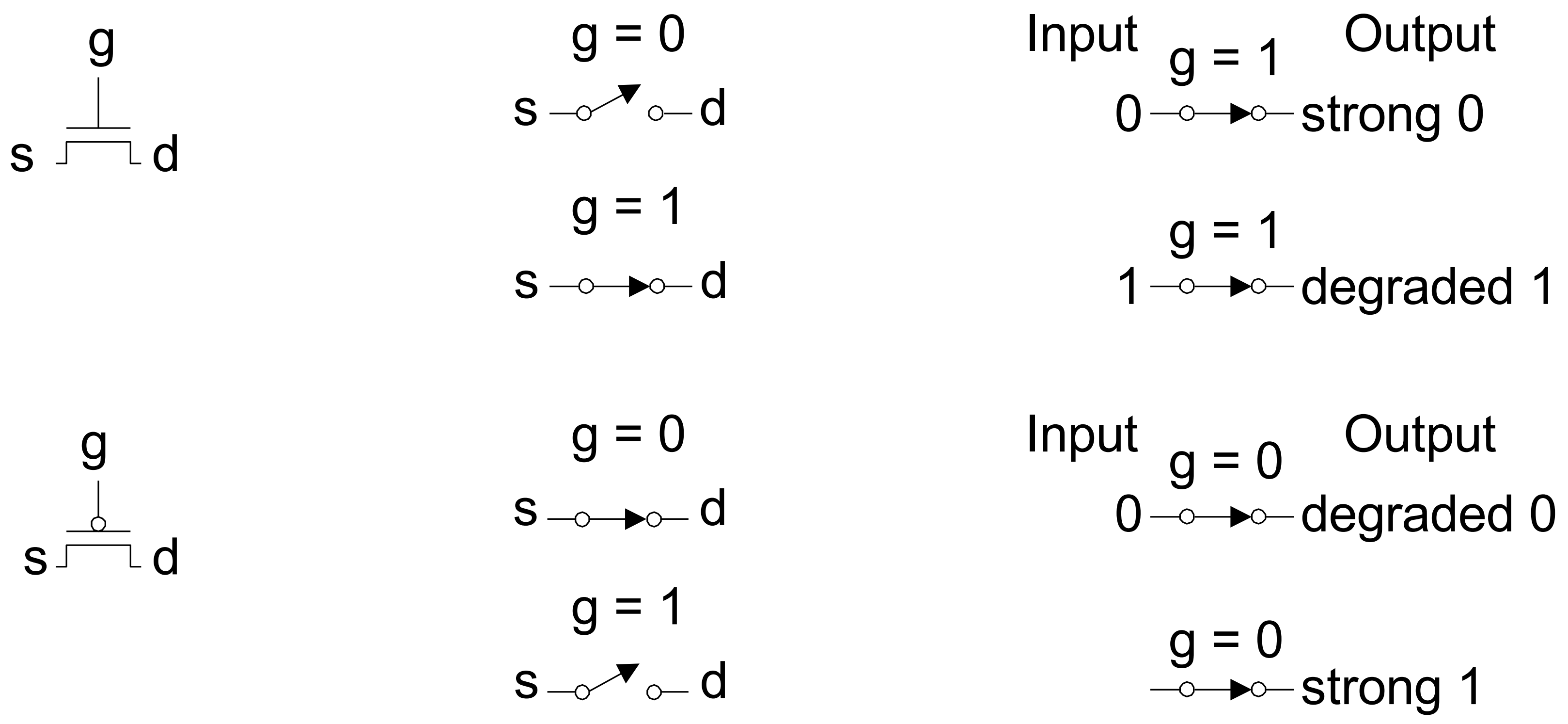
G =

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

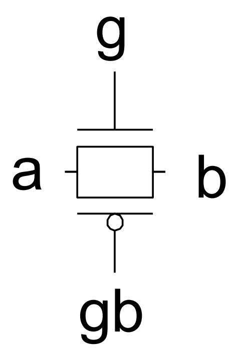
Pass Transistors

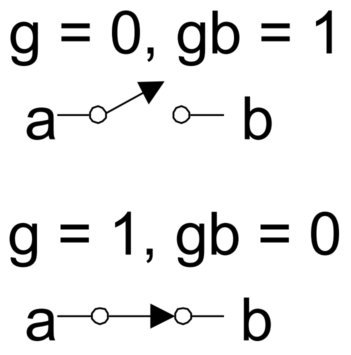
- **Transistors can be used as switches**

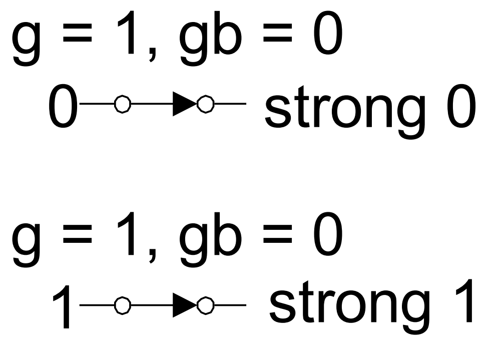


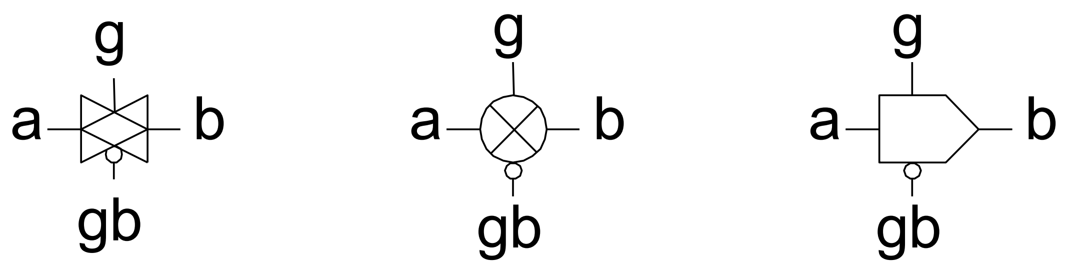
Transmission Gates

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



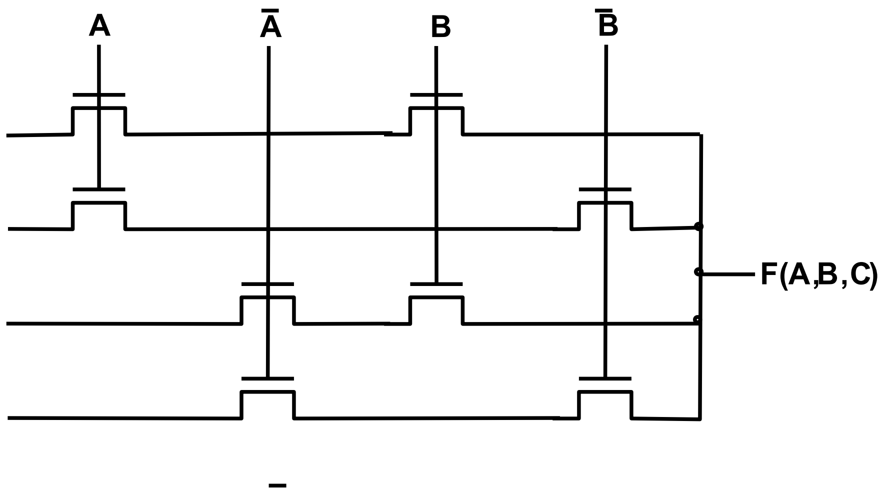


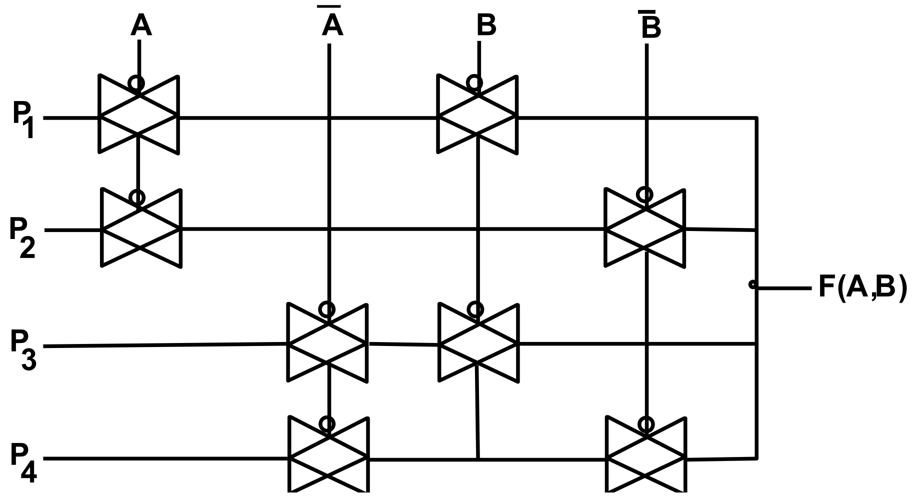




Pass Transistor Logic

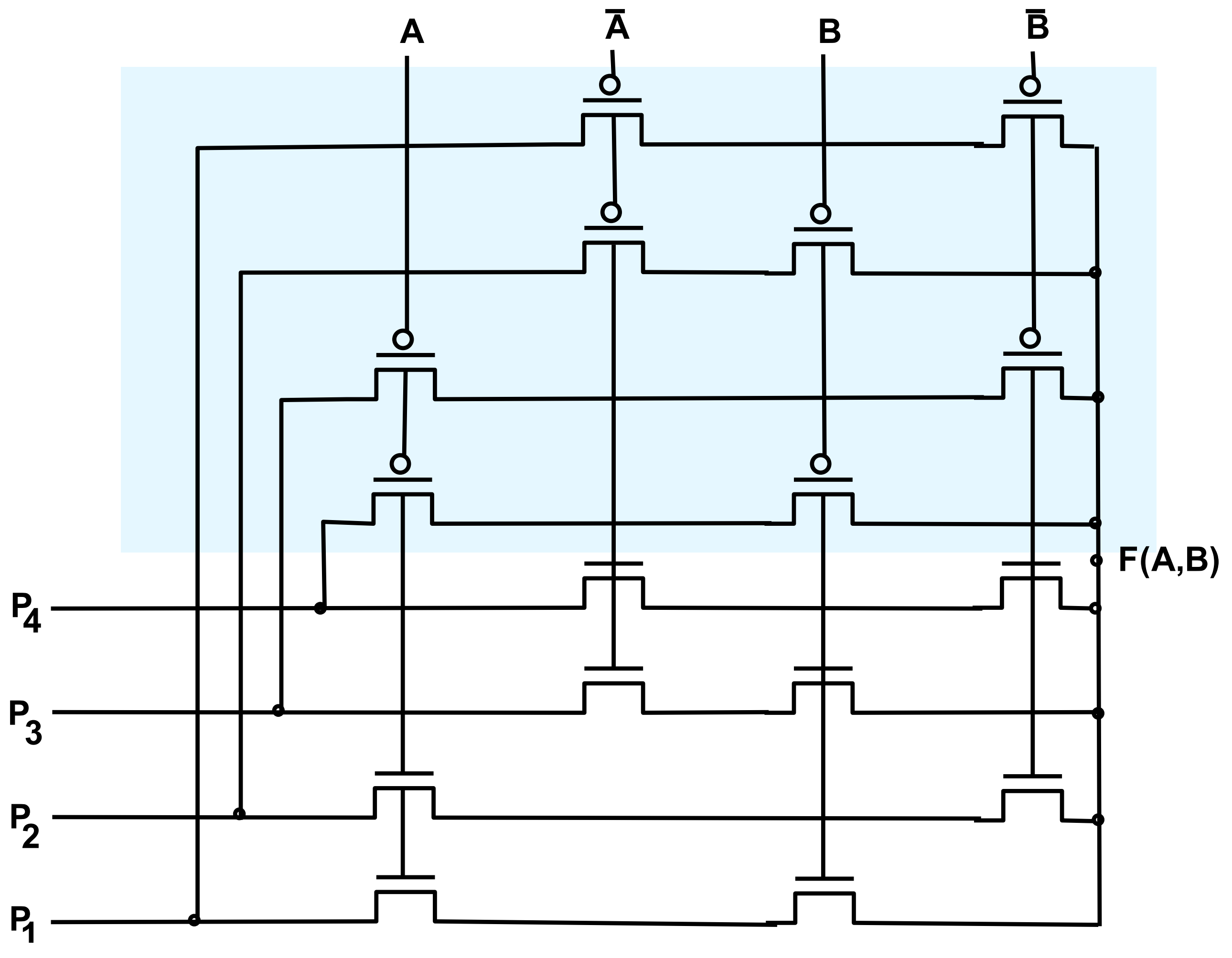
- What is the difference between the two circuits?





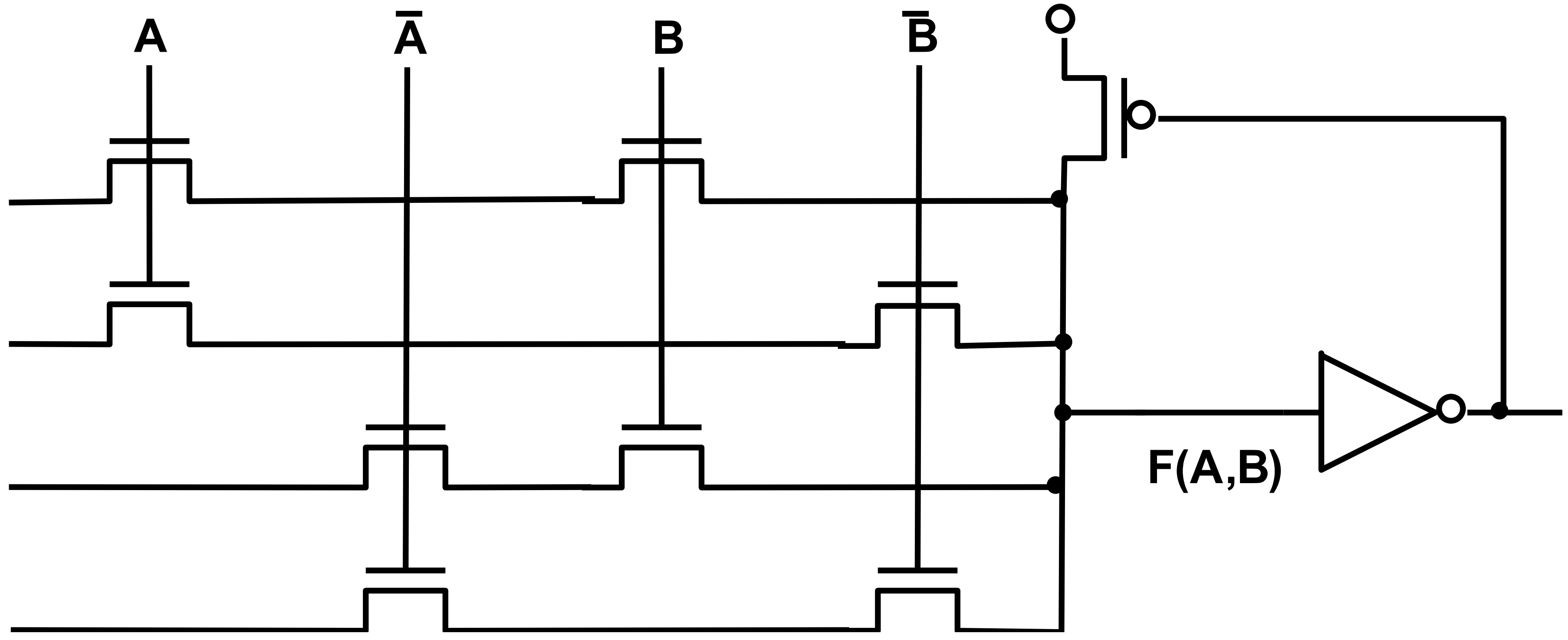
Pass Transistor Logic -- Better Layout

- Group similar transistors, so they can be in the same well



Pass Transistor Logic Pull-Up Version

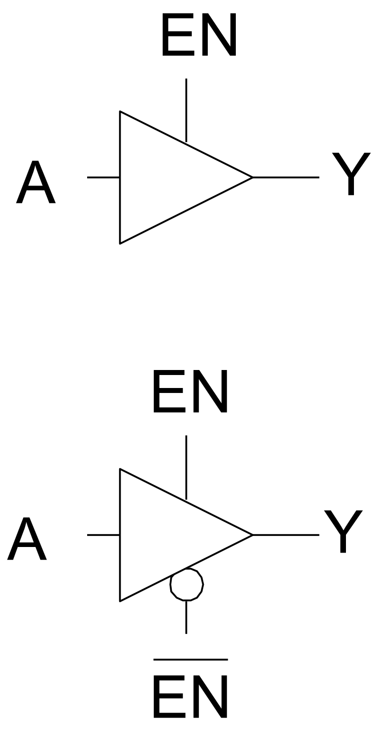
- **How do voltage levels at the output of this gate differ from that of the pass-transistor multiplexer in the previous foil?**



Tristates

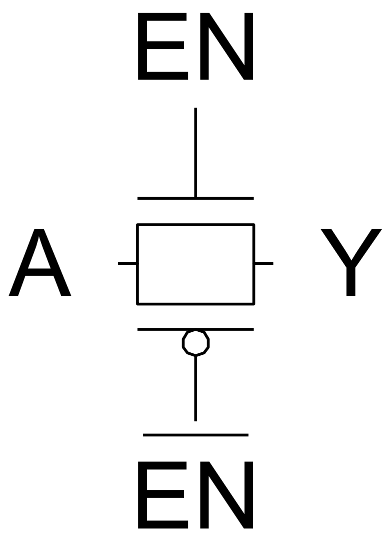
- ***Tristate buffer*** **produces Z when not enabled**

EN	A	Y
0	0	
0	1	
1	0	
1	1	



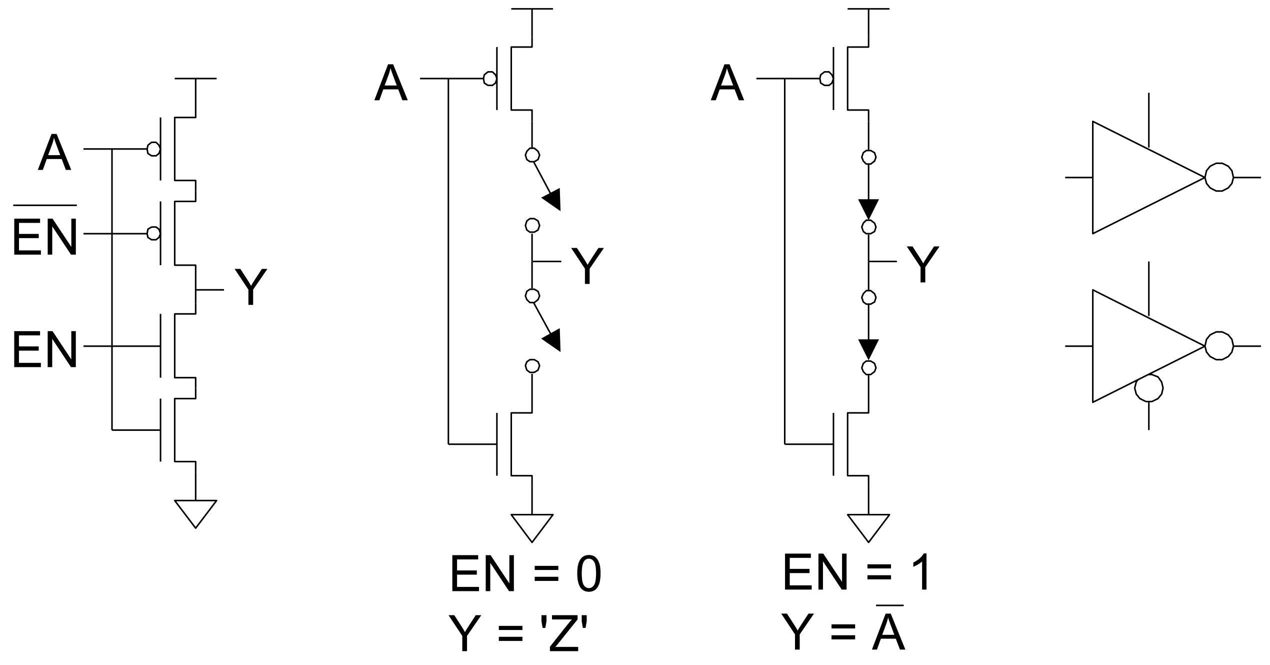
Non-restoring Tri-state®

- Transmission gate acts as Tri-state® buffer
  - Only two transistors
  - But nonrestoring
    - Noise on A is passed on to Y



Tri-state® Inverter

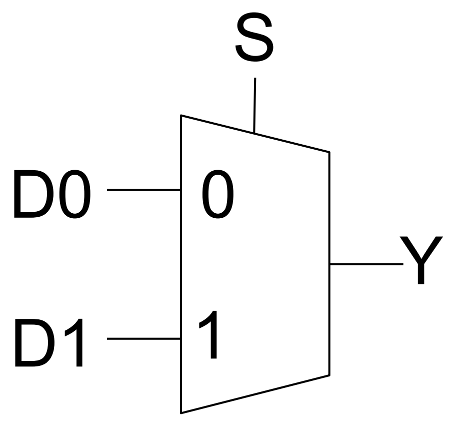
- **Tri-state® inverter produces restored output**
  - Violates conduction complement rule
  - Because we want a Z output



Multiplexers (mux)

- **2:1 *multiplexer* chooses between two inputs**

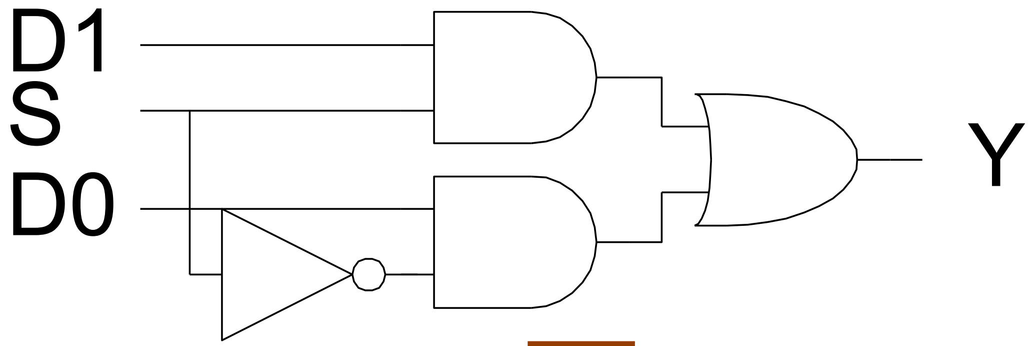
S	D1	D0	Y
0	X	0	0
0	X	1	1
1	0	X	0
1	1	X	1

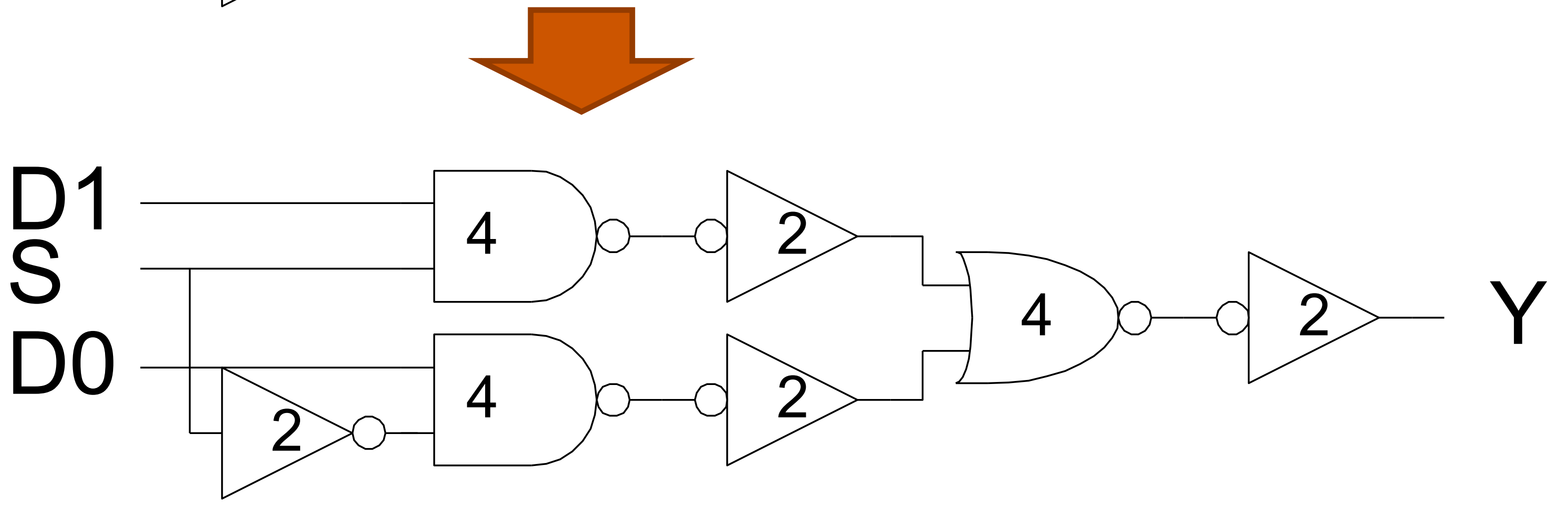


Gate-Level Mux Design

- **How many transistors are needed?**

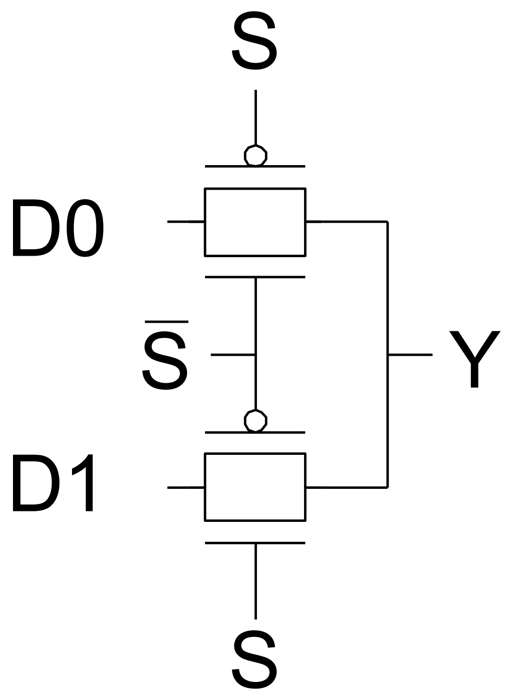
$$Y = SD_1 + \overline{S}D_0 \text{ (too many transistors)}$$ 20





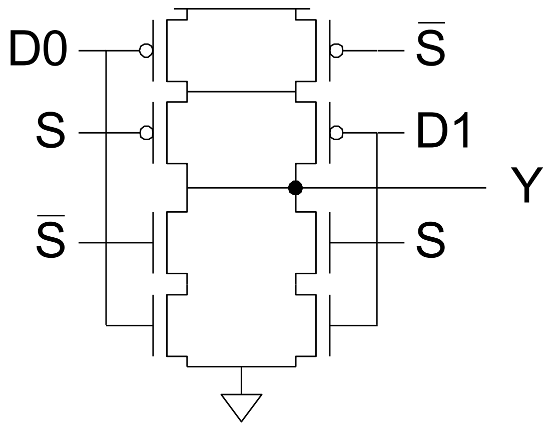
Transmission Gate Mux

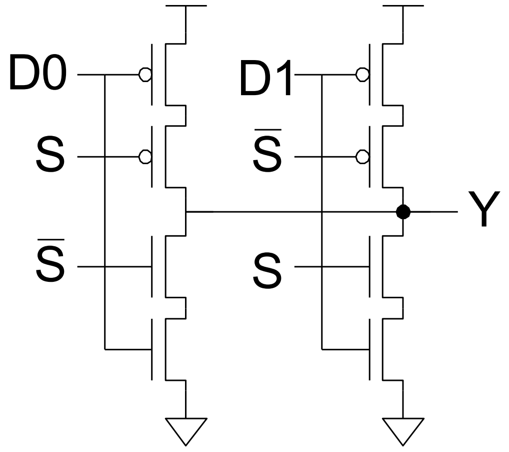
- Nonrestoring mux uses two transmission gates
  - Only 4 transistors if both of the select signals are available
  - If not then it takes 6 transistors…

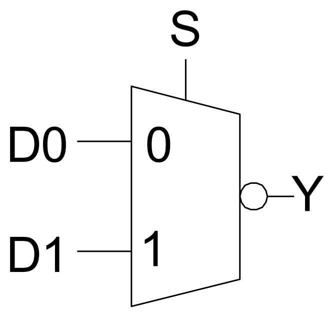


Inverting Mux

- **Inverting multiplexer**
  - Use compound AOI22
  - Or pair of tristate inverters
  - Essentially the same thing
- **Non-inverting multiplexer requires adding an inverter**

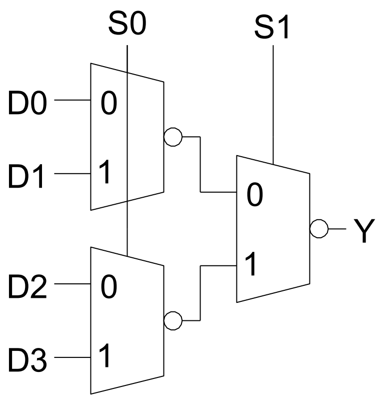




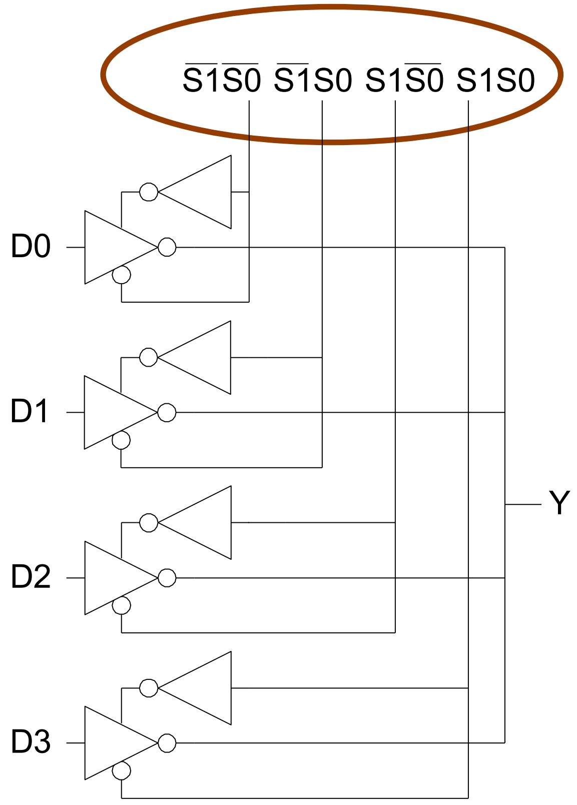


4:1 Multiplexer

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

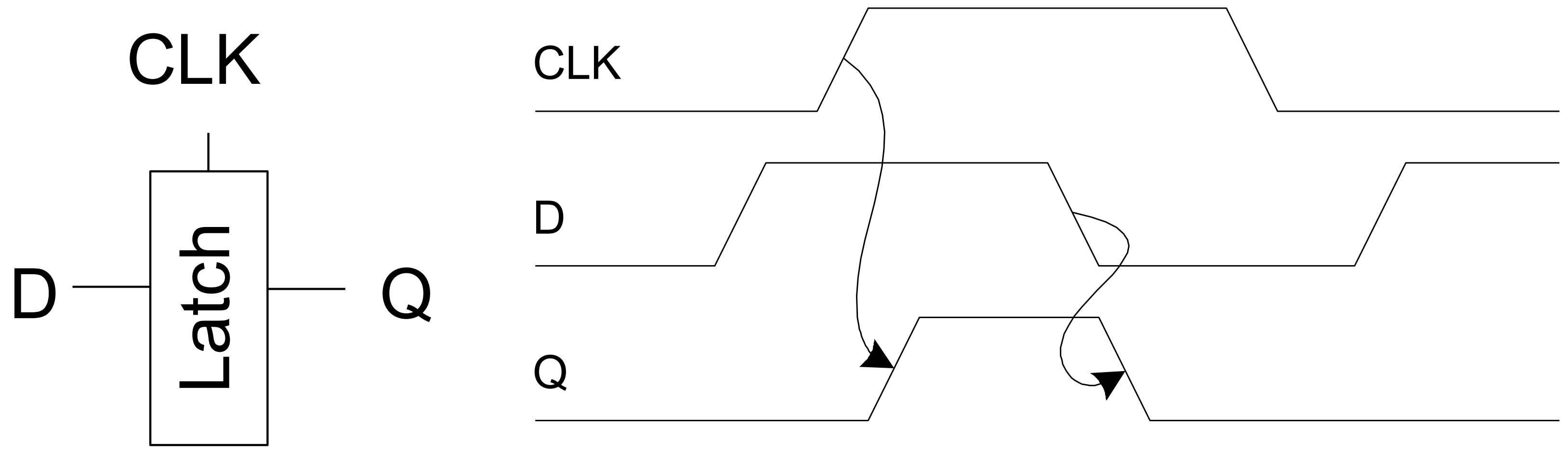


Requires pre-decoded signals


$\overline{S1}\,\overline{S0}$ $\overline{S1}S0$ $S1\overline{S0}$ $S1S0$
D0
D1
D2
D3
Y

D Latch**

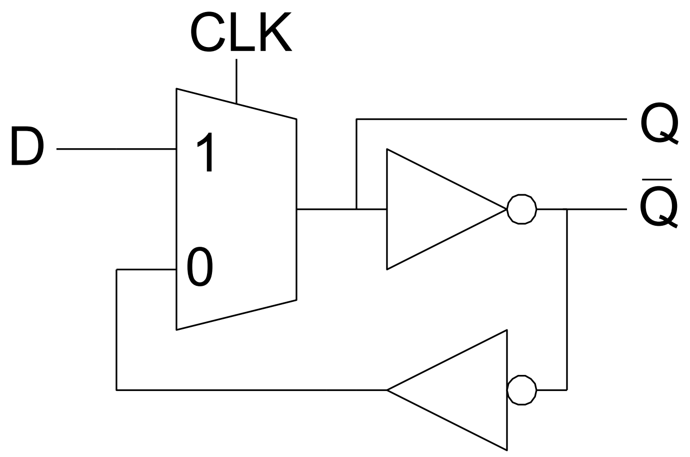
- When CLK = 1, latch is ***transparent***
  - D flows through to Q like a buffer
- When CLK = 0, the latch is ***opaque***
  - Q holds its old value independent of D

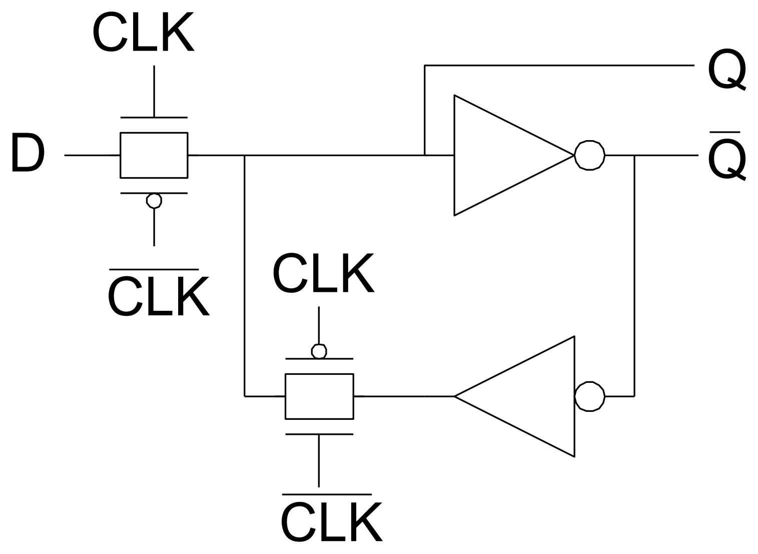


** ***transparent latch*** **or** ***level-sensitive latch***

D Latch Design

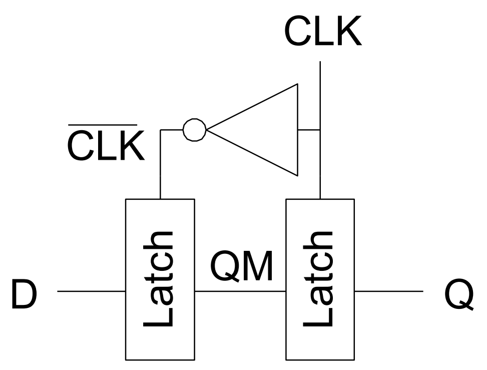
- **Multiplexer chooses D or old Q**

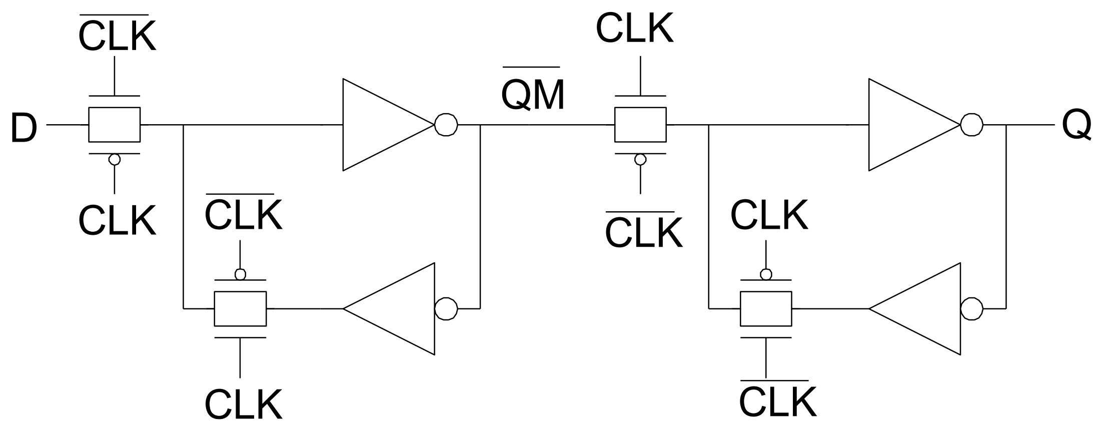




D Flip-flop Design

- **Built from master and slave D latches**





Questions?