3. Implementing Logic in CMOS

Jacob Abraham

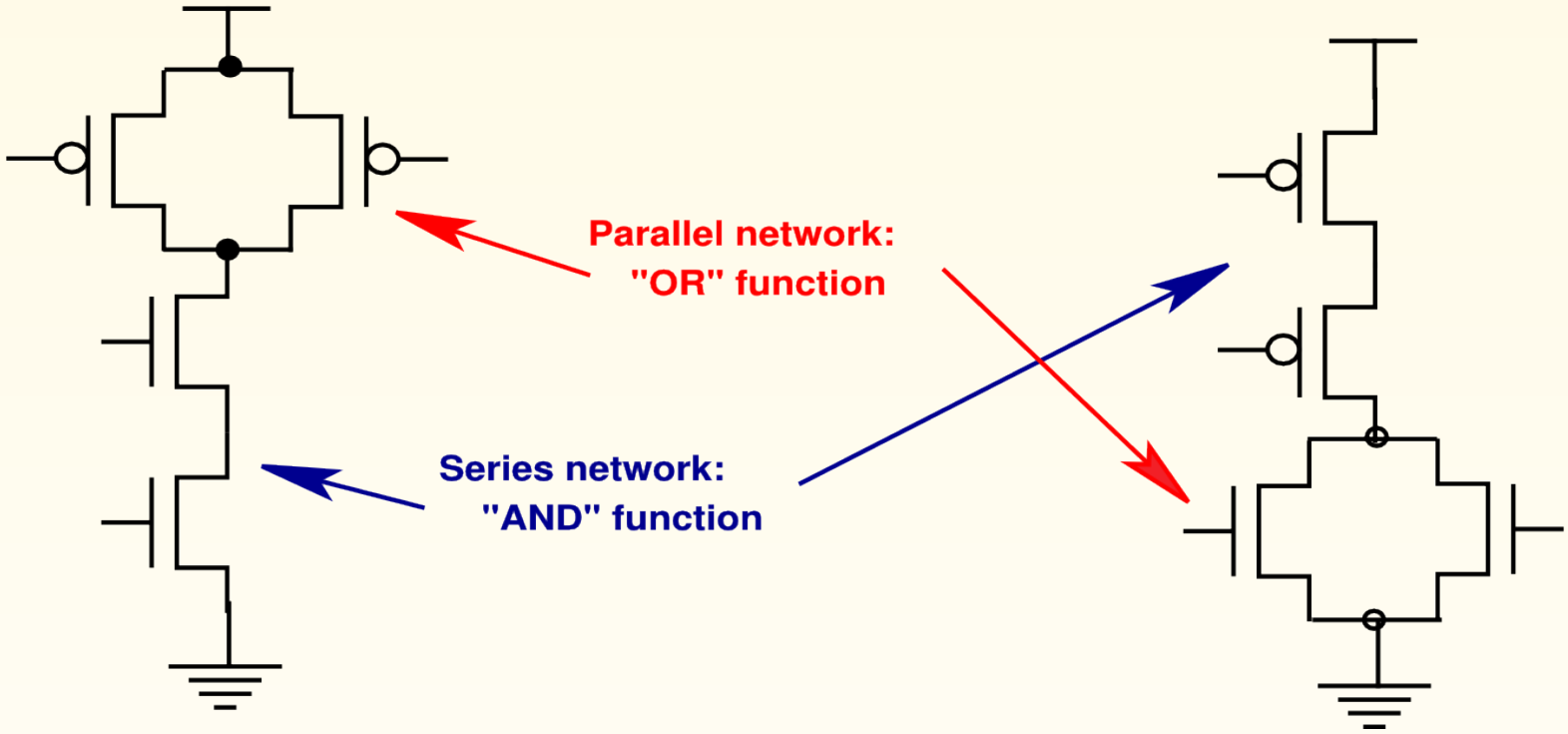
Department of Electrical and Computer Engineering
The University of Texas at Austin

VLSI Design
Fall 2020

September 3, 2020

N- and P-channel Networks

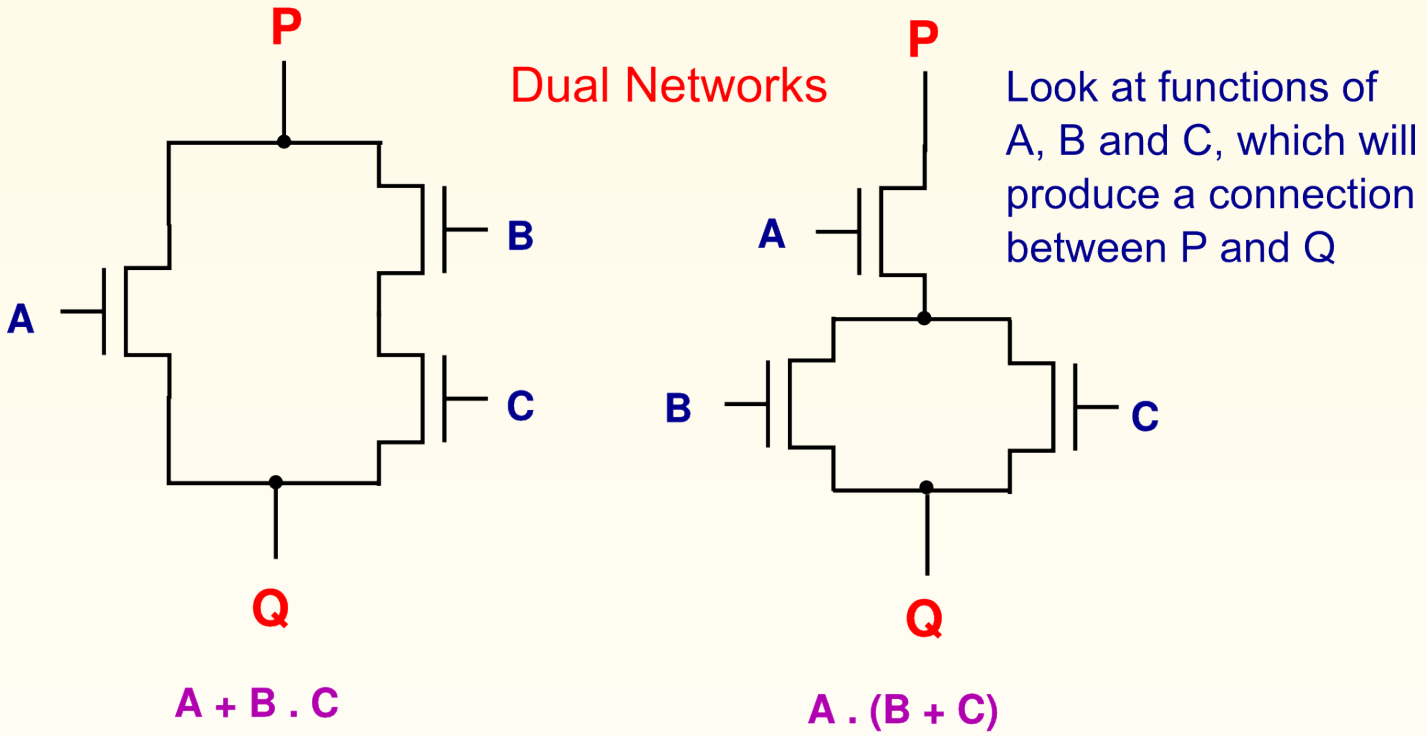
- N- and P-channel networks implement logic functions
  - Each network connected between **Output** and V_{DD} or V_{SS}
  - Function defines path between the terminals



Duality in CMOS Networks

Straightforward way of constructing static CMOS circuits is to implement dual N- and P- networks

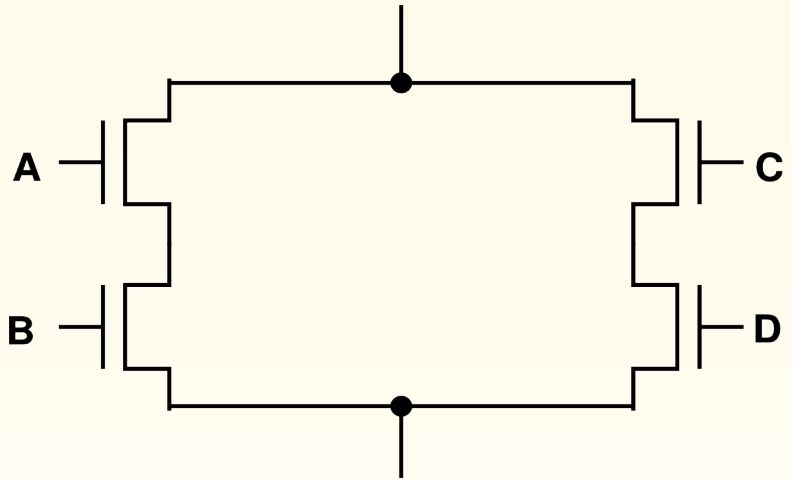
- N- and P- networks must implement **complementary** functions
- Duality **sufficient** for correct operation (but not necessary)



Constructing Complex Gates

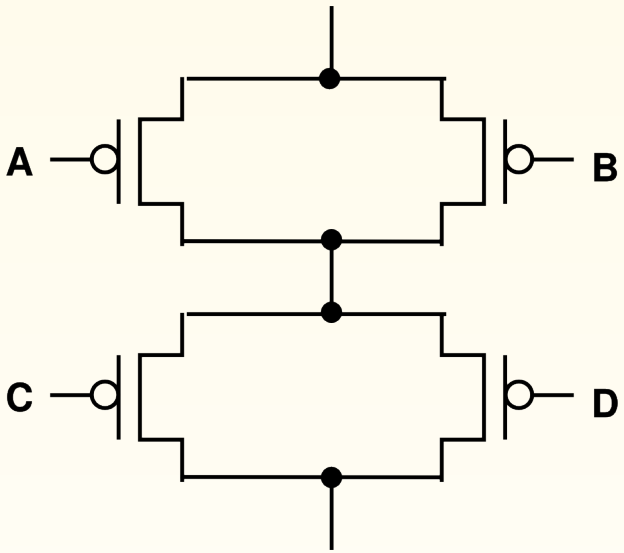
Example: $F = \overline{(A \cdot B) + (C \cdot D)}$

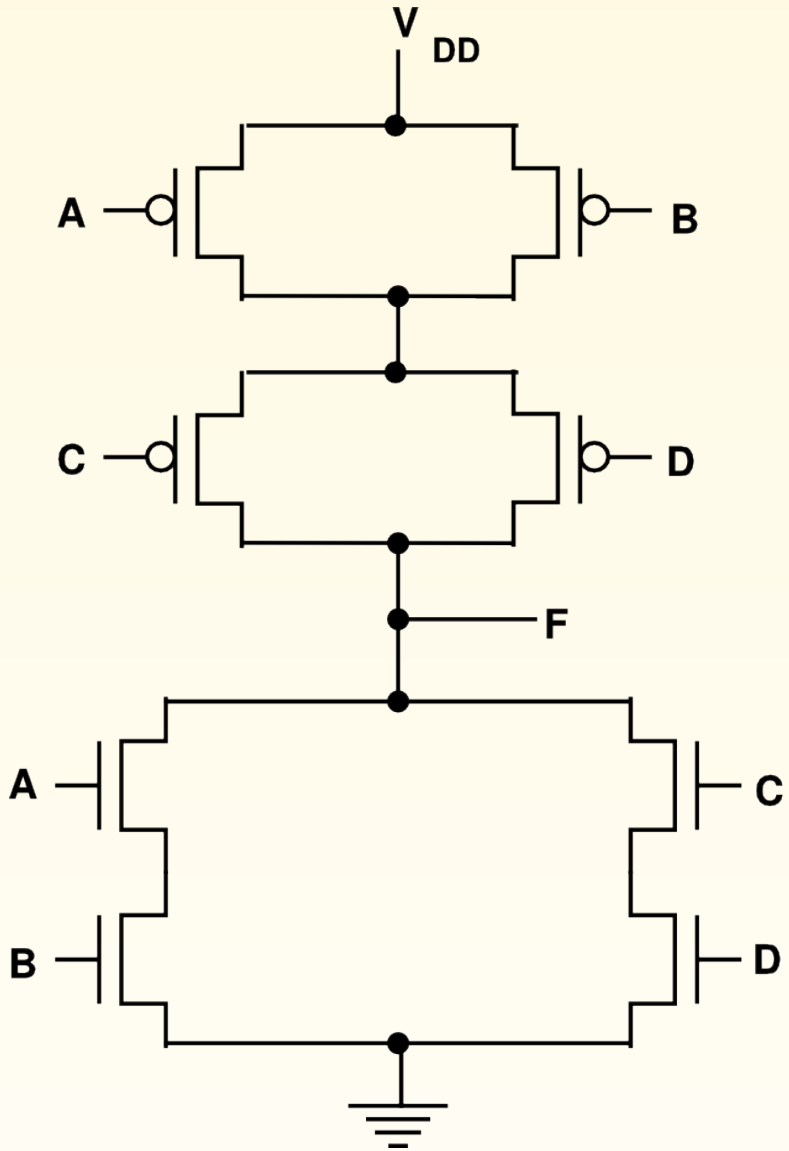
1. Take uninverted function $F = (A \cdot B) + (C \cdot D)$ and derive N-network
2. Identify AND, OR components: F is OR of AB, CD
3. Make connections of transistors



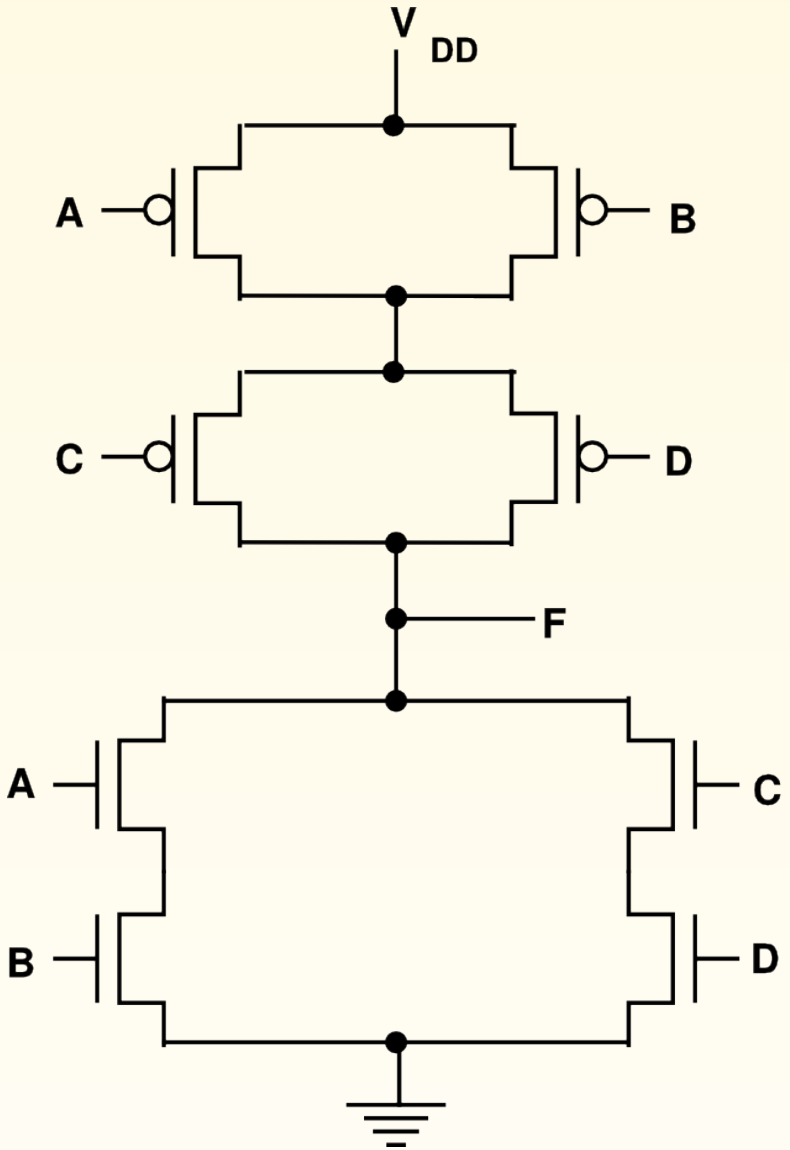
Construction of Complex Gates, Cont'd

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

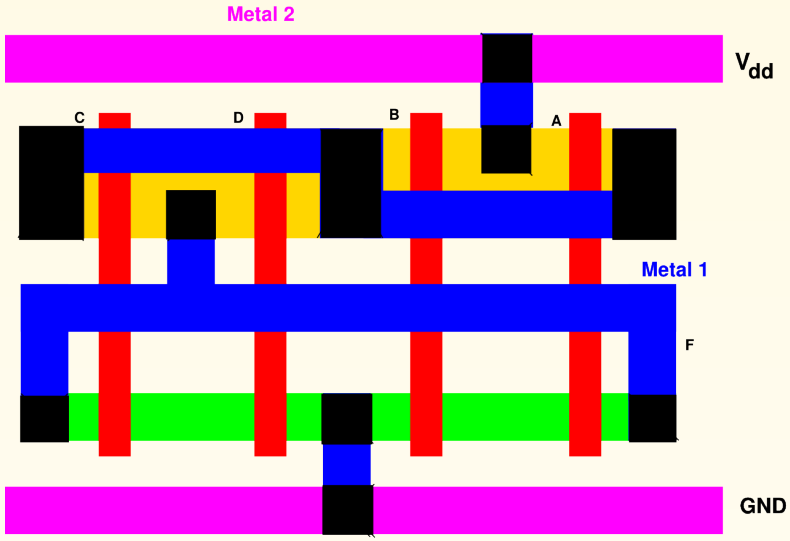




Layout of Complex Gate



AND-OR-INVERT (AOI) gate



Note: Arbitrary shapes are not allowed in some nanoscale design rules

Example of Compound Gate

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

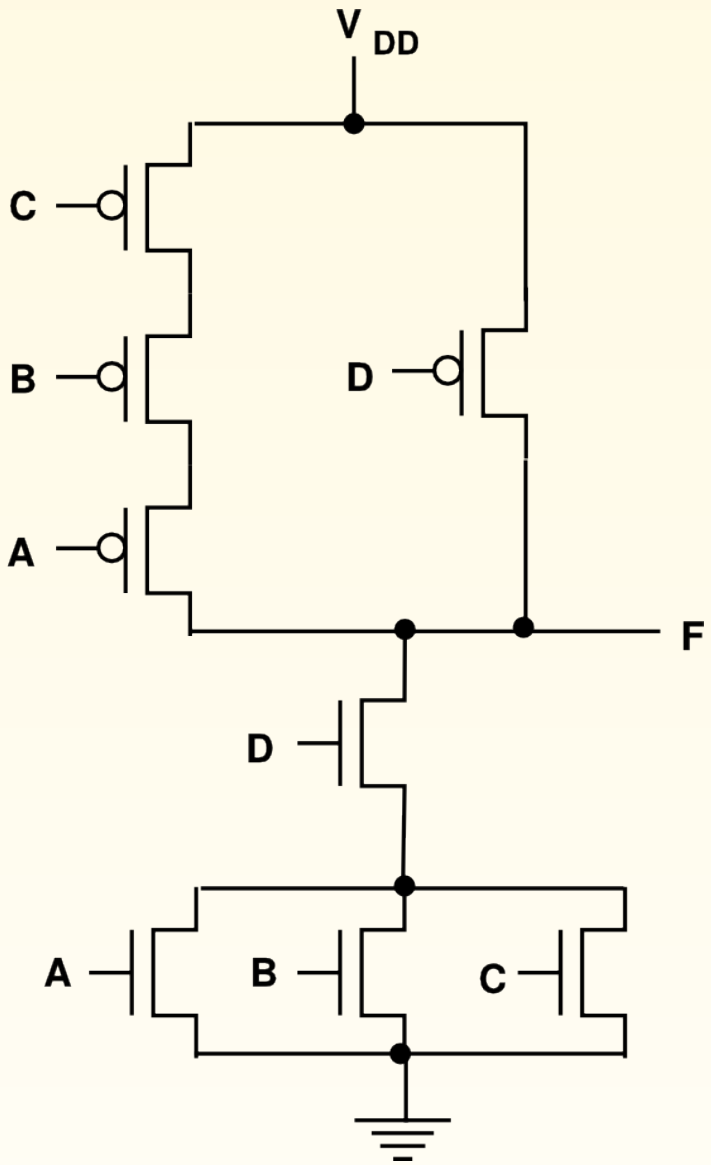
Note:

N- and P- graphs are duals of each other

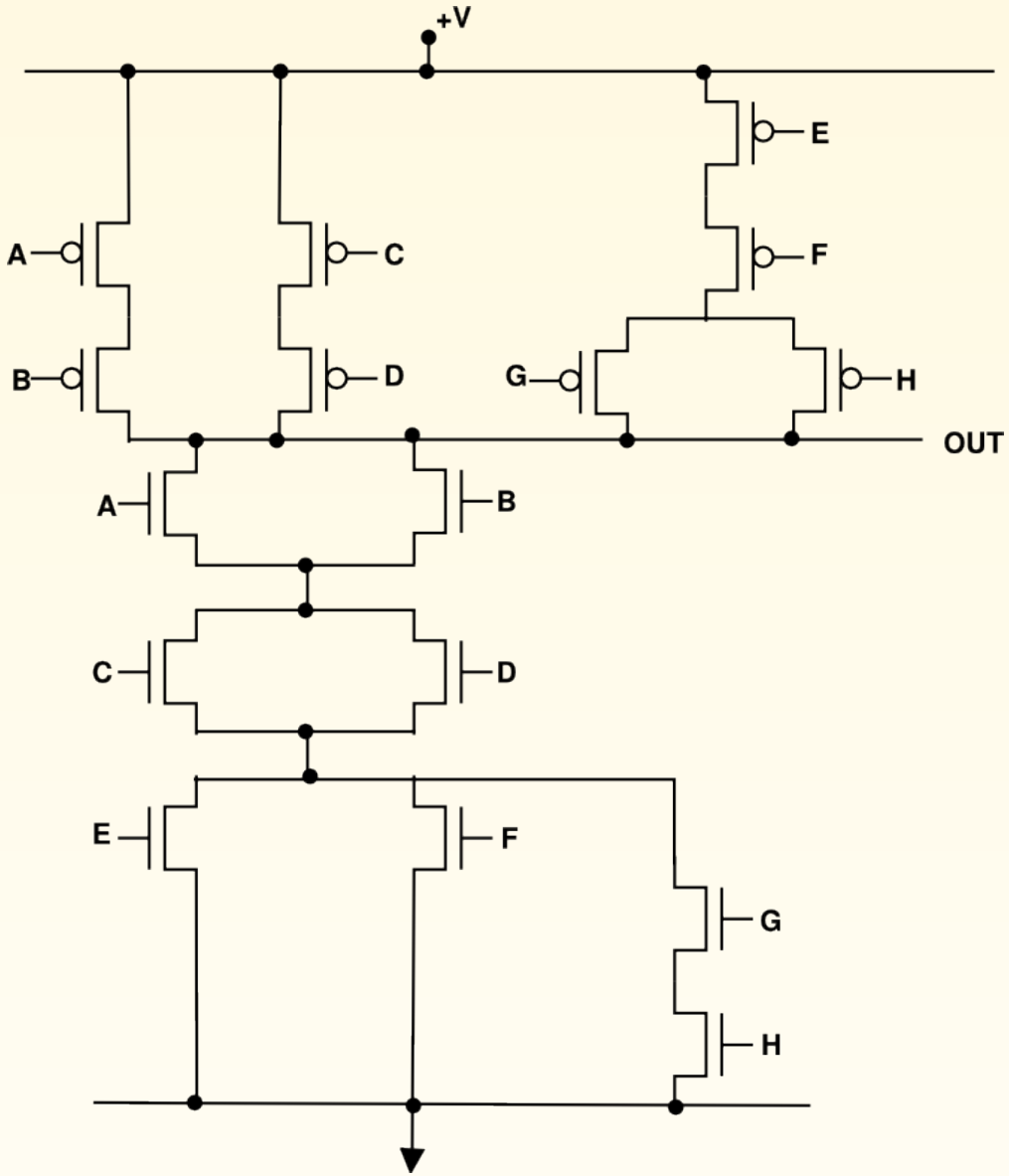
In this case, the function is the complement of the switching function between F and GND

Question: Does it make any difference to the function if the transistor with input D is connected between the parallel A, B, C, transistors and GND?

What about the electrical behavior?

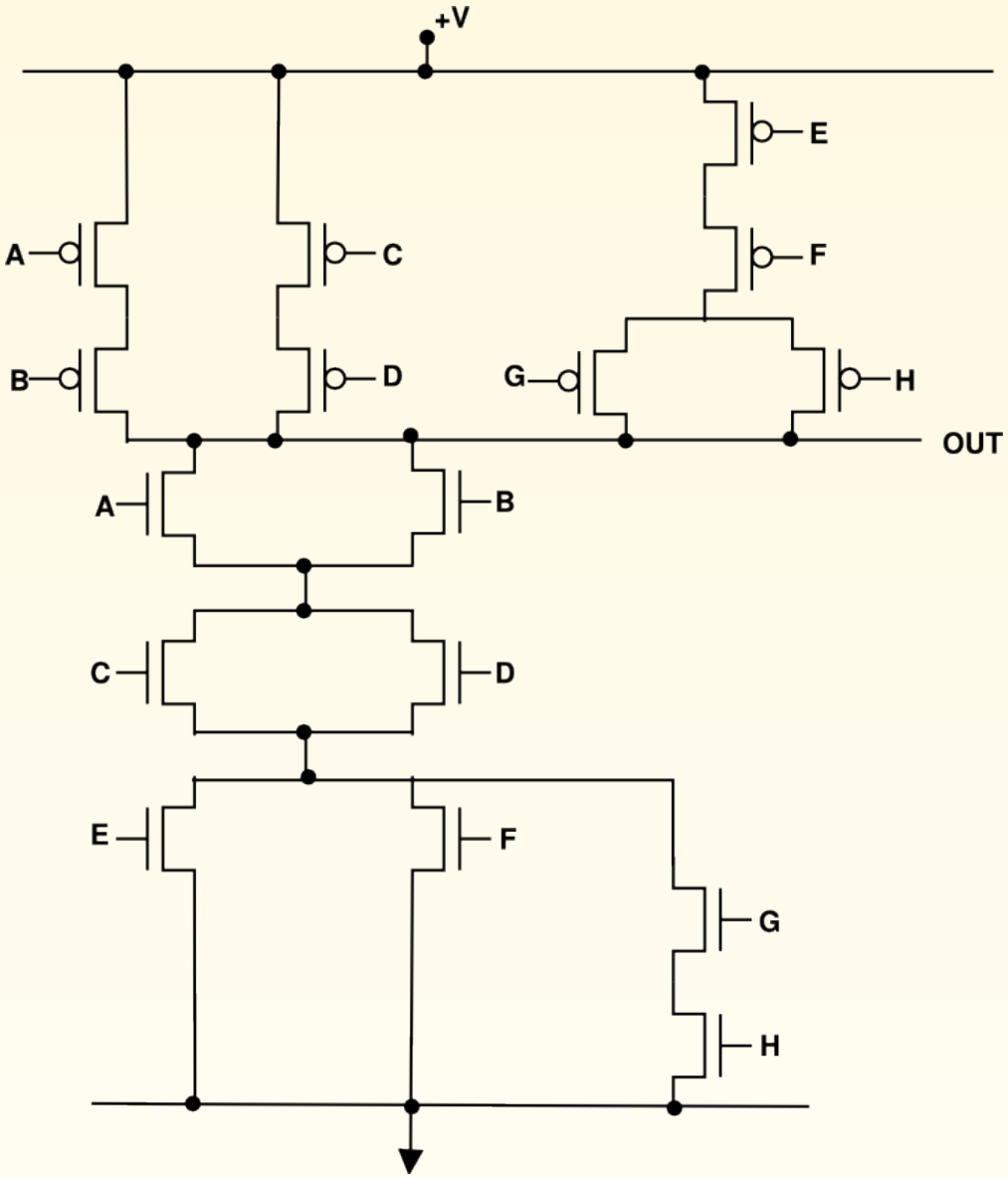


Example of More Complex Gate



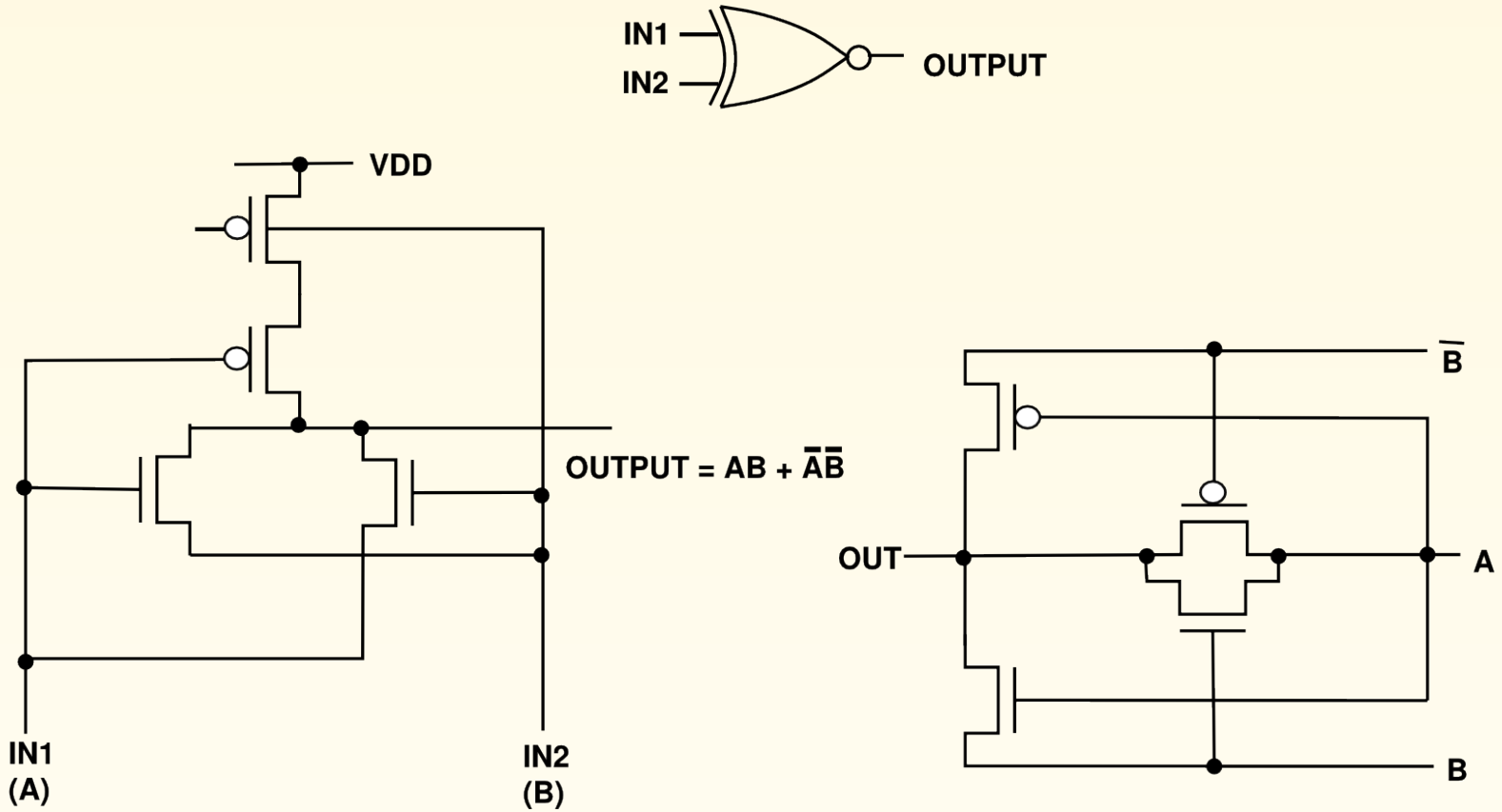
$$OUT = \overline{(A+B)\cdot(C+D)\cdot(E+F+(G\cdot H))}$$

Example of More Complex Gate



$$OUT = \overline{(A + B) \cdot (C + D) \cdot (E + F + (G \cdot H))}$$

Exclusive-NOR Gate in CMOS



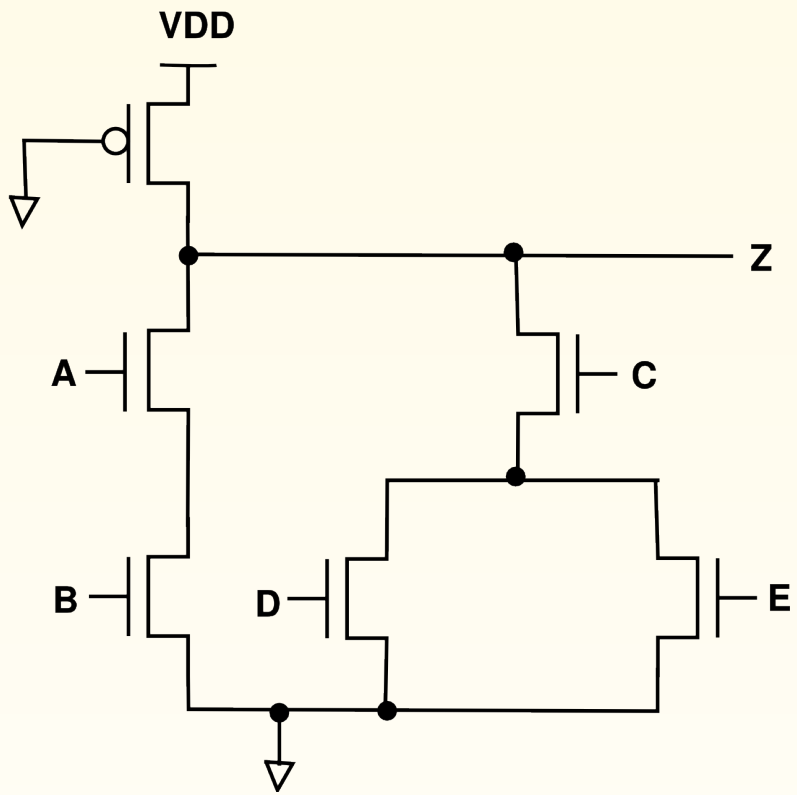
Note: designs such as these should be checked very carefully for correct behavior using circuit simulation

Pseudo nMOS Logic

Based on the old NMOS technology where a "depletion" transistor was used as a pullup resistor

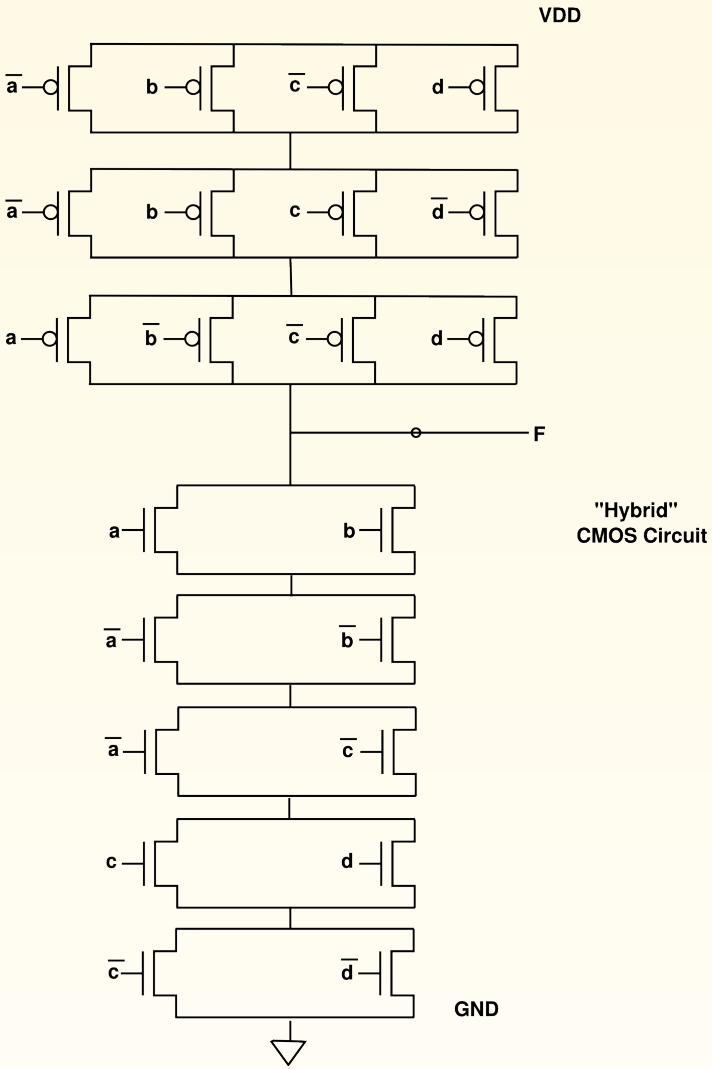
What happens when there is no path from Z to ground (i.e., Z = 1)?

What happens when there is a path from Z to ground (i.e., Z = 0)?



Duality is Not Necessary for a CMOS Structure

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

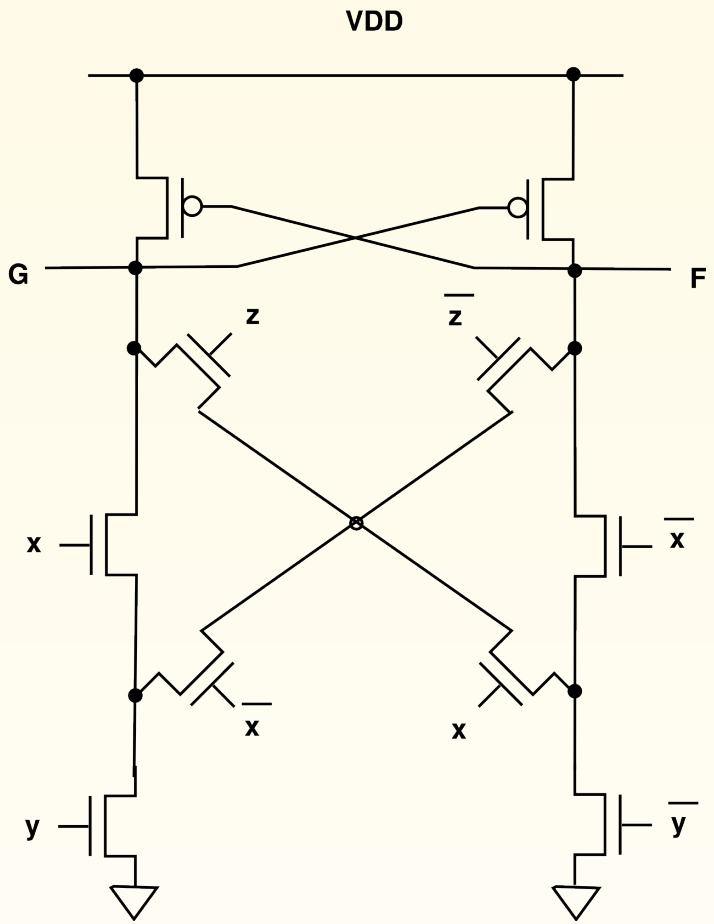


$$F = a \cdot b + \overline{a} \cdot \overline{b} + a \cdot c + c \cdot d + \overline{c} \cdot \overline{d}$$

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

Example of Another Complex CMOS Gate

This circuit does not have a pMOS network – just one transistor for each function; it will work only if F and G are complements of each other. Why?



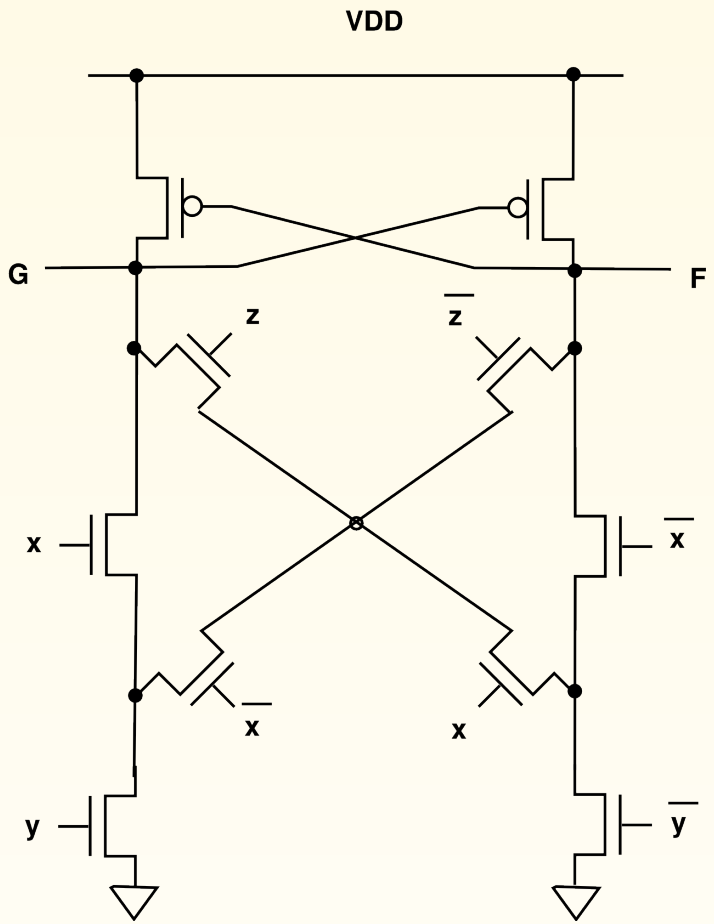
Can evaluate the voltages at F and G ($\{0, V_{DD}\}$) for each value of $x, y,$ and z

$$F = x \cdot \overline{y} \cdot z + \overline{x} \cdot y \cdot z + x \cdot y \cdot \overline{z} + x \cdot y \cdot z$$

$$G = \overline{x} \cdot y \cdot \overline{z} + x \cdot \overline{y} \cdot \overline{z} + \overline{x} \cdot \overline{y} \cdot z + \overline{x} \cdot \overline{y} \cdot \overline{z}$$

Example of Another Complex CMOS Gate

This circuit does not have a pMOS network – just one transistor for each function; it will work only if F and G are complements of each other. Why?

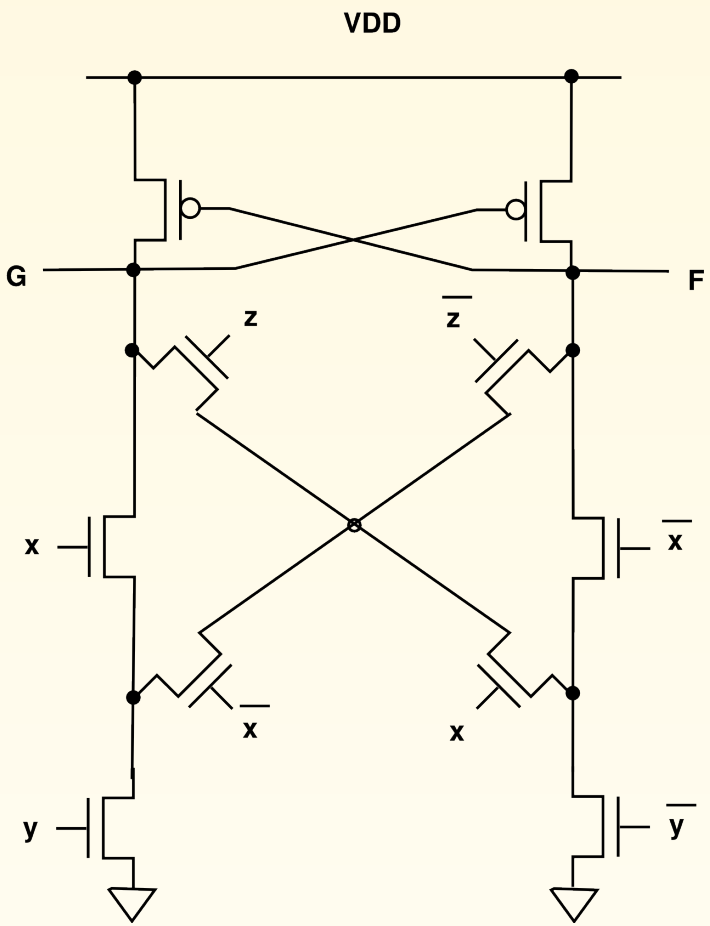


Can evaluate the voltages at F and G ($\{0, V_{DD}\}$) for each value of $x, y,$ and z

$$F = x \cdot \overline{y} \cdot z + \overline{x} \cdot y \cdot z + x \cdot y \cdot \overline{z} + x \cdot y \cdot z$$

$$G = \overline{x} \cdot y \cdot \overline{z} + x \cdot \overline{y} \cdot \overline{z} + \overline{x} \cdot \overline{y} \cdot z + \overline{x} \cdot \overline{y} \cdot \overline{z}$$

Example of Another Complex CMOS Gate, Cont'd



Can also follow every path from F and G to GND and identify values of $x, y,$ and z which will enable the path to be enabled.

$$\overline{F} = \overline{x} \cdot \overline{y} + x \cdot \overline{y} \cdot \overline{z} + \overline{x} \cdot y \cdot \overline{z}$$

$$F = (x + y) \cdot (\overline{x} + y + z) \cdot (x + \overline{y} + z)$$

$$= (y + x \cdot z) \cdot (x + \overline{y} + z)$$

$$= x \cdot y + y \cdot z + x \cdot z$$

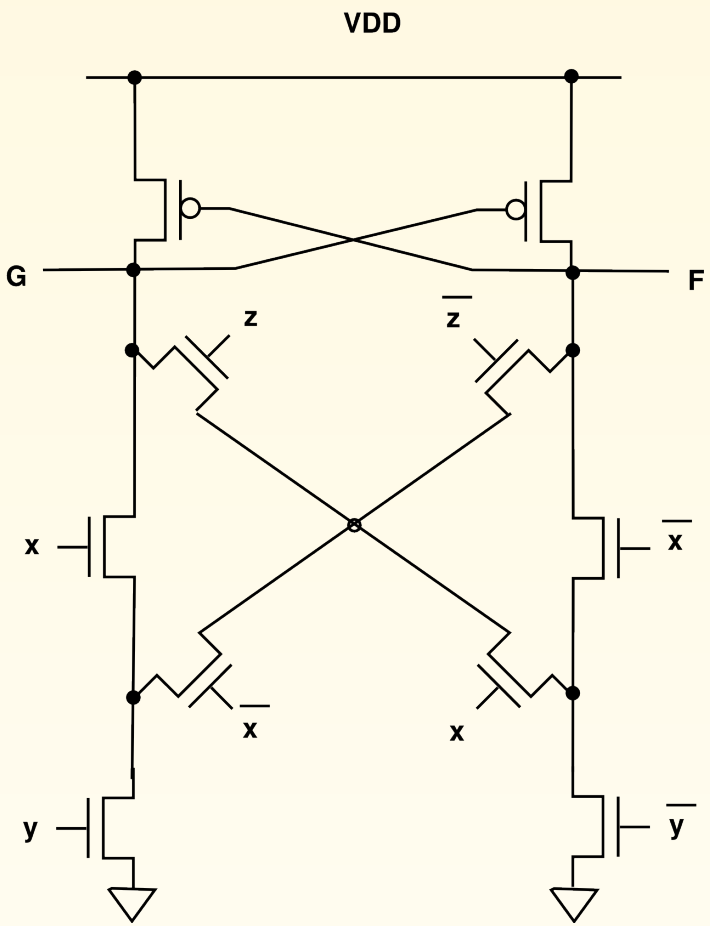
$$G = \overline{x} \cdot y \cdot \overline{z} + x \cdot \overline{y} \cdot \overline{z} + \overline{x} \cdot \overline{y}$$

$$= \overline{x} \cdot \overline{y} + \overline{x} \cdot \overline{z} + \overline{y} \cdot \overline{z}$$

Can you describe the functions in simple terms?

(Hint: look at the number of input variables which are true (or false) when the output is 1.)

Example of Another Complex CMOS Gate, Cont'd



Can also follow every path from F and G to GND and identify values of $x, y,$ and z which will enable the path to be enabled.

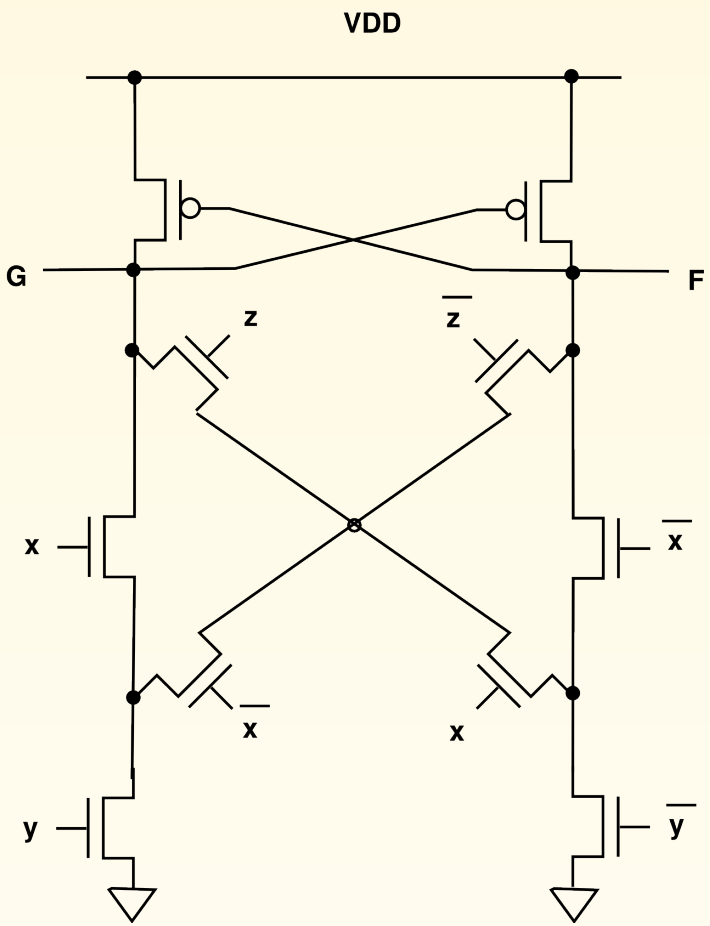
$$\overline{F} = \overline{x} \cdot \overline{y} + x \cdot \overline{y} \cdot \overline{z} + \overline{x} \cdot y \cdot \overline{z}$$

$$\begin{aligned} F &= (x + y) \cdot (\overline{x} + y + z) \cdot (x + \overline{y} + z) \\ &= (y + x \cdot z) \cdot (x + \overline{y} + z) \\ &= x \cdot y + y \cdot z + x \cdot z \end{aligned}$$

$$\begin{aligned} G &= \overline{x} \cdot y \cdot \overline{z} + x \cdot \overline{y} \cdot \overline{z} + \overline{x} \cdot \overline{y} \\ &= \overline{x} \cdot \overline{y} + \overline{x} \cdot \overline{z} + \overline{y} \cdot \overline{z} \end{aligned}$$

Can you describe the functions in simple terms?
(Hint: look at the number of input variables which are true (or false) when the output is 1.)

Example of Another Complex CMOS Gate, Cont'd



Can also follow every path from F and G to GND and identify values of $x, y,$ and z which will enable the path to be enabled.

$$\overline{F} = \overline{x} \cdot \overline{y} + x \cdot \overline{y} \cdot \overline{z} + \overline{x} \cdot y \cdot \overline{z}$$

$$F = (x + y) \cdot (\overline{x} + y + z) \cdot (x + \overline{y} + z)$$

$$= (y + x \cdot z) \cdot (x + \overline{y} + z)$$

$$= x \cdot y + y \cdot z + x \cdot z$$

$$G = \overline{x} \cdot y \cdot \overline{z} + x \cdot \overline{y} \cdot \overline{z} + \overline{x} \cdot \overline{y}$$

$$= \overline{x} \cdot \overline{y} + \overline{x} \cdot \overline{z} + \overline{y} \cdot \overline{z}$$

Can you describe the functions in simple terms?

(Hint: look at the number of input variables which are true (or false) when the output is 1.)

Signal Strength

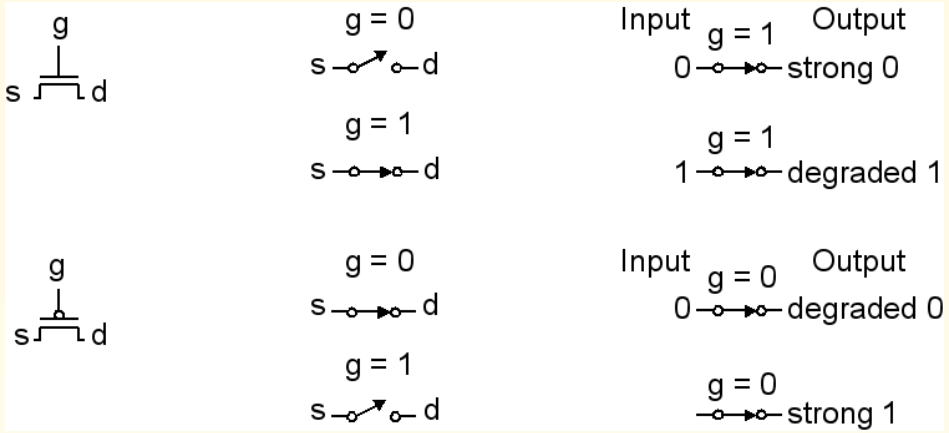
Voltages represent digital logic values

- **Strength** of signal:
  - How close it approximates ideal voltage
- V_{DD} and GND rails are strongest 1 and 0
- nMOS transistors pass a strong 0
  - But degraded or weak 1
- pMOS transistors pass a strong 1
  - But degraded or weak 0

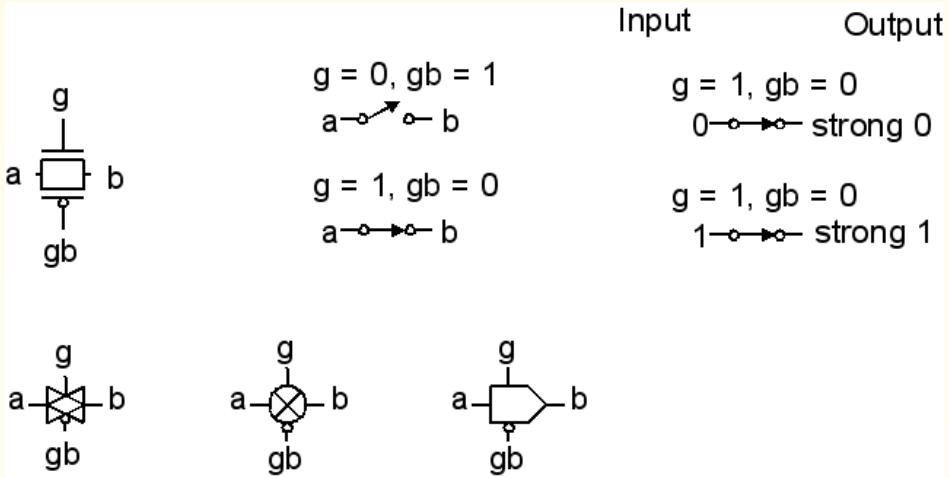
Therefore, nMOS transistors are best for the "pull-down" network

Pass Transistors and Transmission Gates

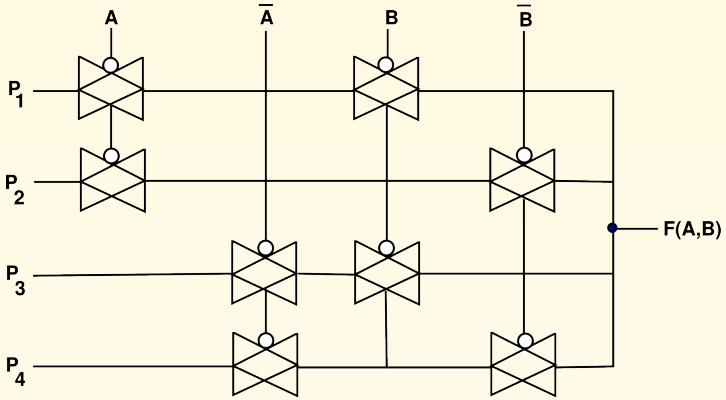
Transistors can be used as switches; however, they could produce degraded outputs

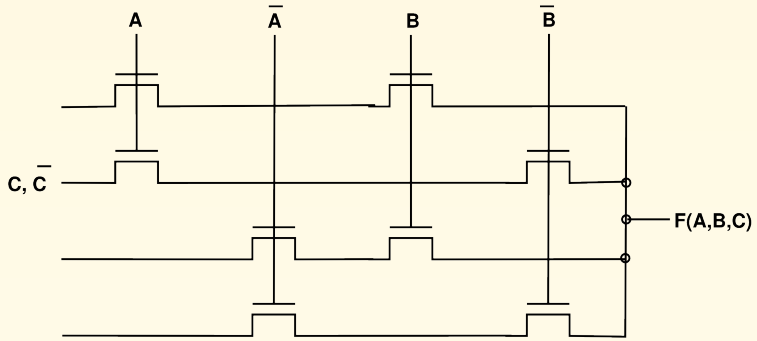


Transmission gates pass both 0 and 1 well



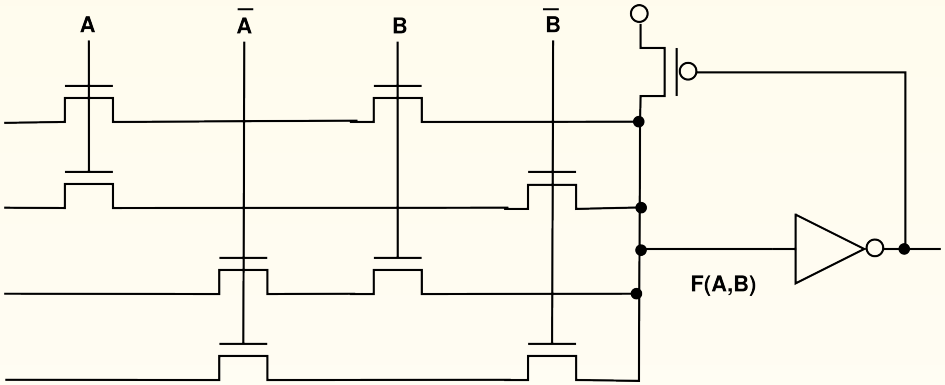
Pass Transistor Logic





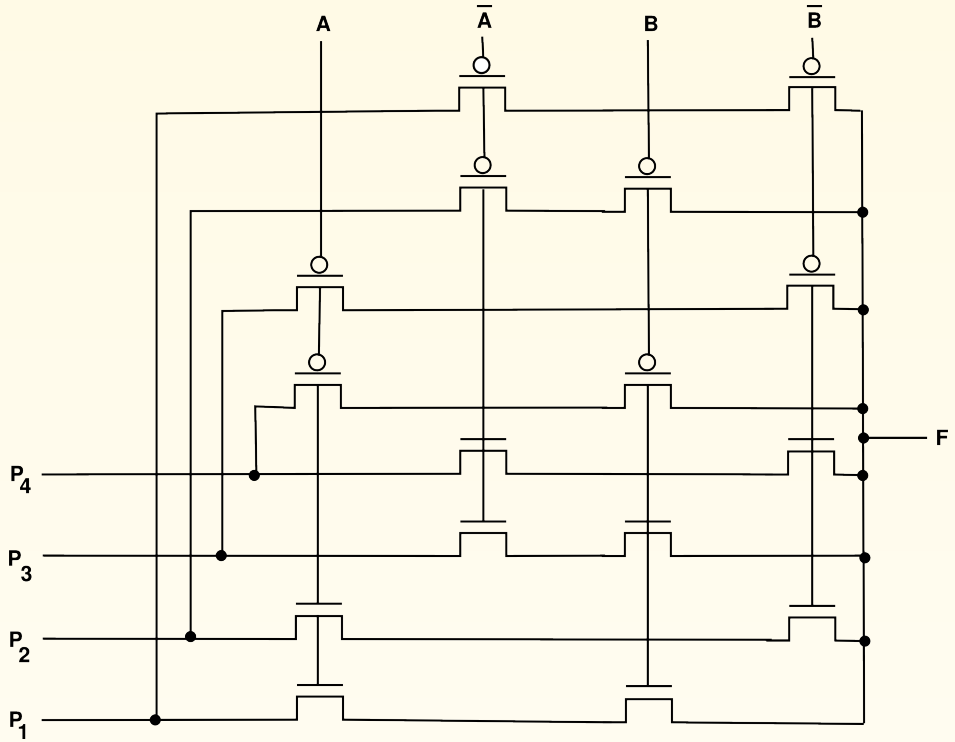
"Pull-Up" Circuit

- Used to restore degraded logic 1 from output of nMOS pass transistor



Pass Transistor Logic – Better Layout

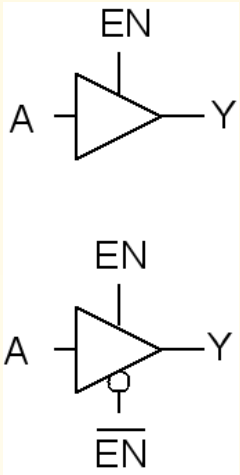
Group similar transistors, so they can be in the same well



Tristates

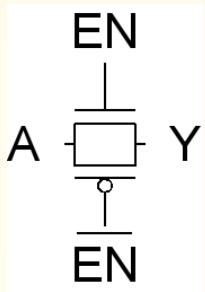
Tristate Buffer produces **Z** (high impedance) when not enabled

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



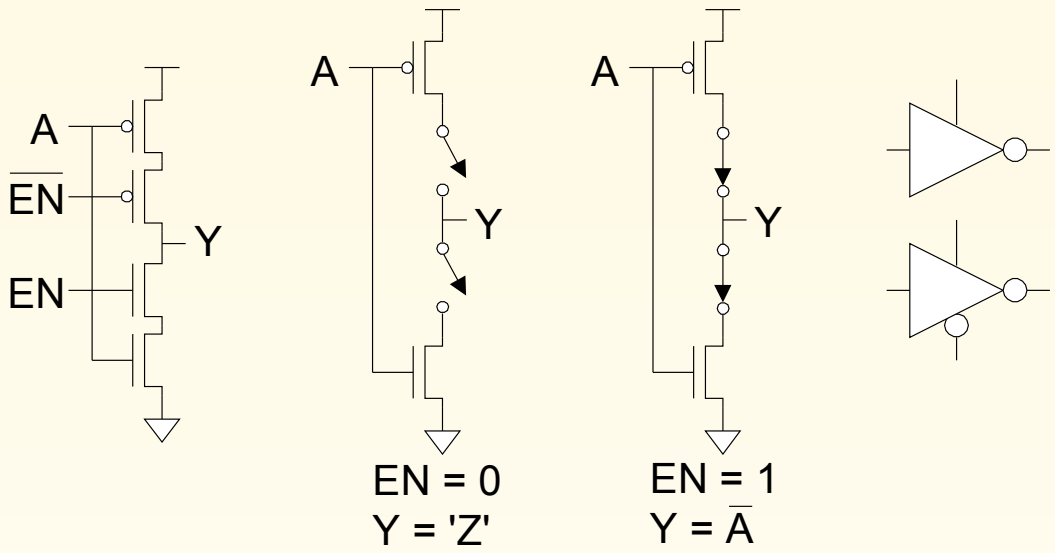
Non-Restoring Tristate

- Transmission gate acts as a tristate buffer
- Only two transistors, but **nonrestoring**
- Noise on A is passed to Y



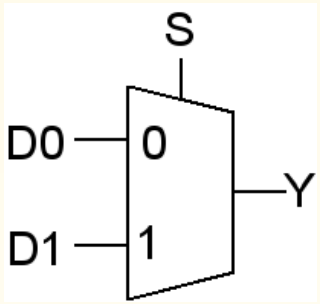
Tristate Inverter

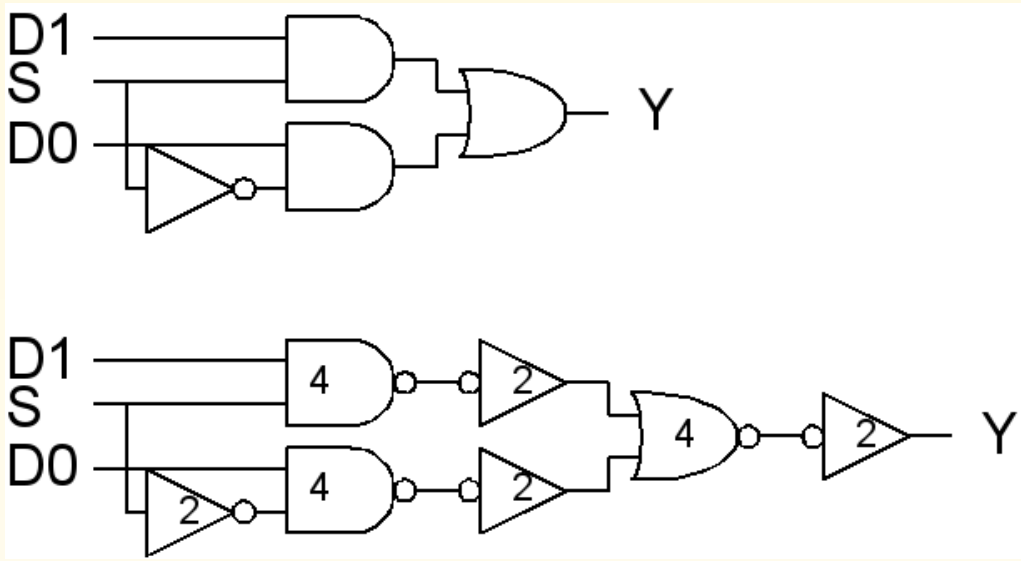
Tristate inverter produces restored output, but complements signal



Multiplexers

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



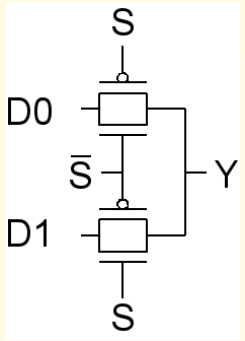


How many transistors are needed?
(The better design uses 3 NAND gates and 1 inverter)

Transmission Gate MUX

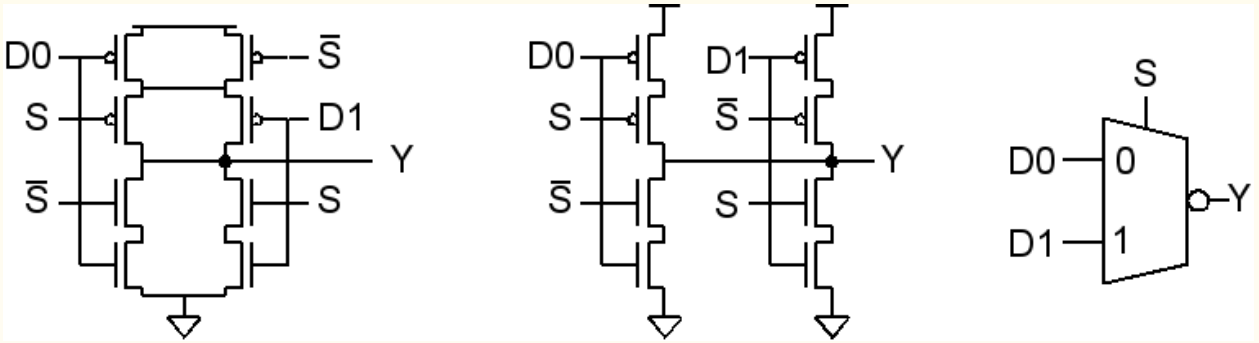
Nonrestoring MUX

- Uses two transmission gates $\Longrightarrow$ only 4 transistors



Inverting MUX – adds an inverter

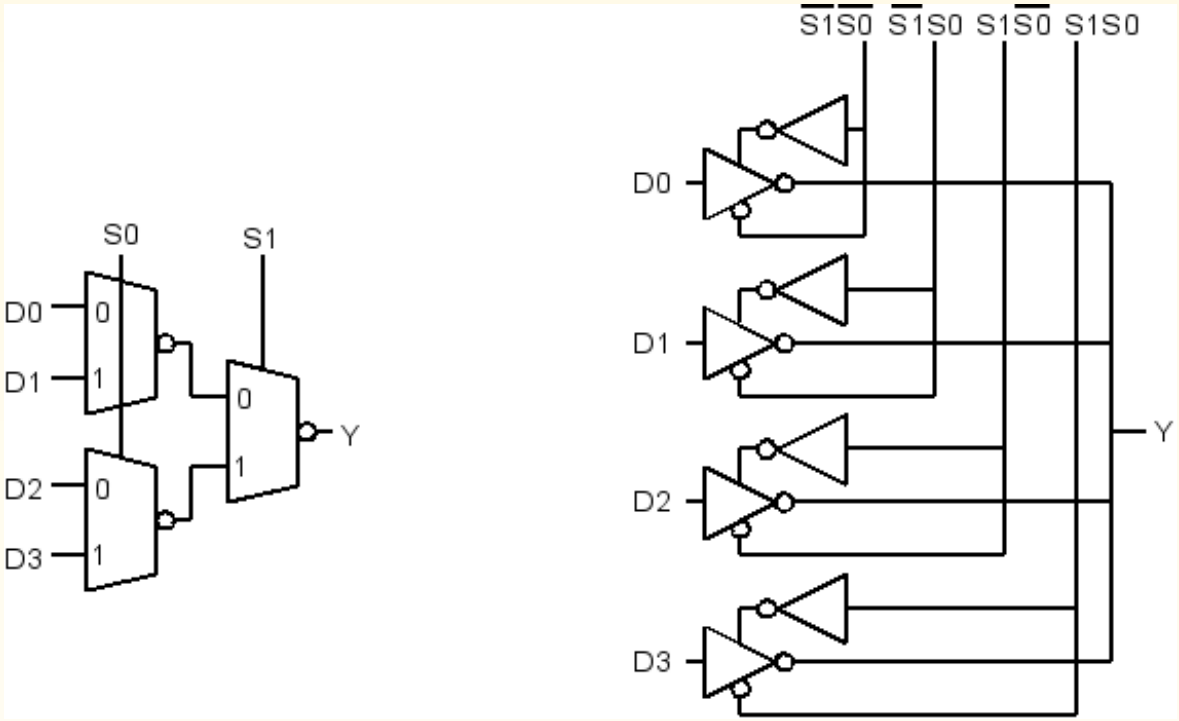
- Uses compound gate AOI22
- Alternatively, a pair of tristate inverters (same thing)



4:1 Multiplexer

A 4:1 MUX chooses one of 4 inputs using two selects

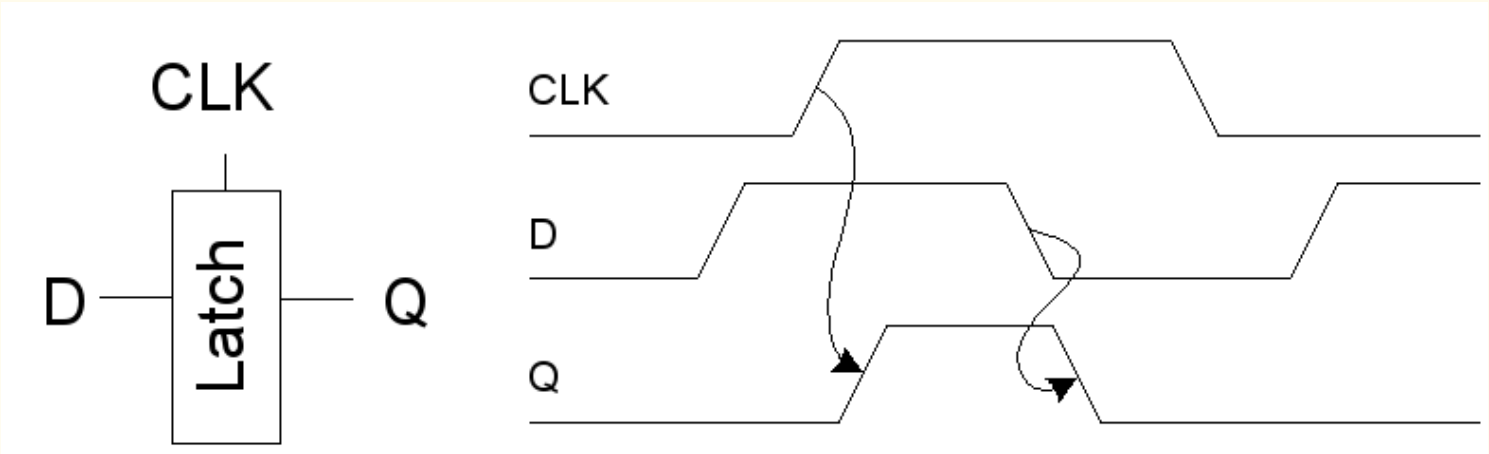
- Two levels of 2:1 MUXes
- Alternatively, four tristates



D Latch

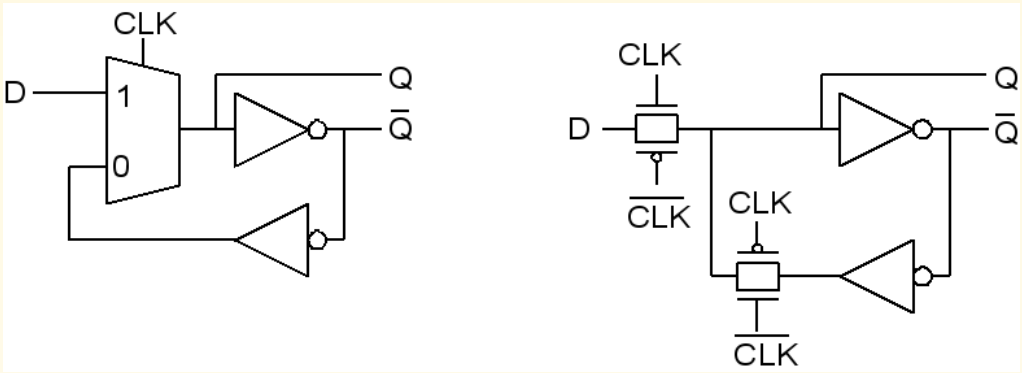
Basic Memory Element

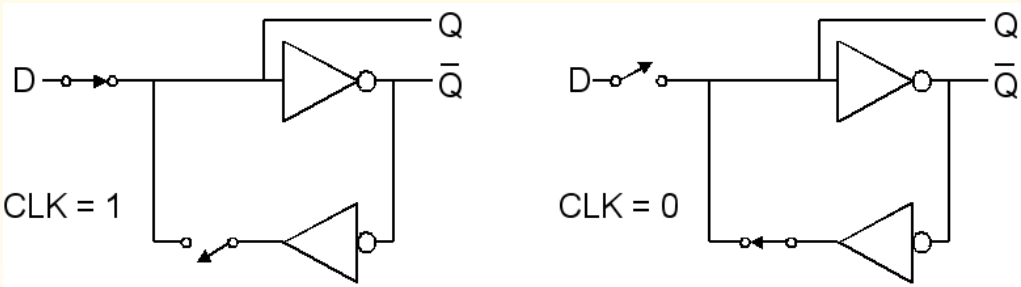
- When CLK = 1, latch is transparent
  - D flows through to Q like a buffer
- When CLK = 0, the latch is opaque
  - Q holds its old value independent of D
- a.k.a., **transparent latch** or **level-sensitive latch**



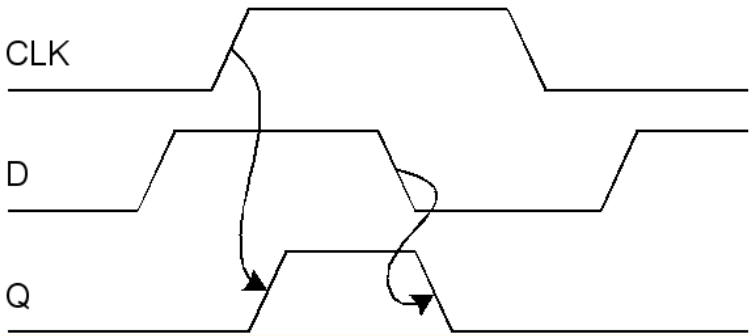
D Latch, Cont'd

D Latch Design: MUX chooses between D and old Q





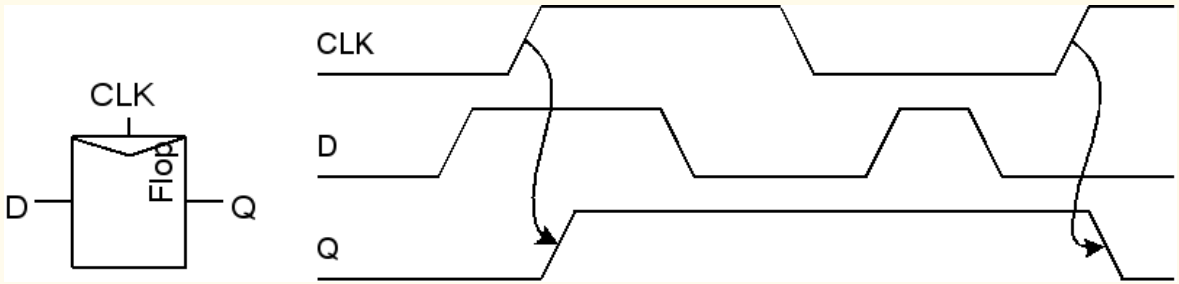
D Latch Operation

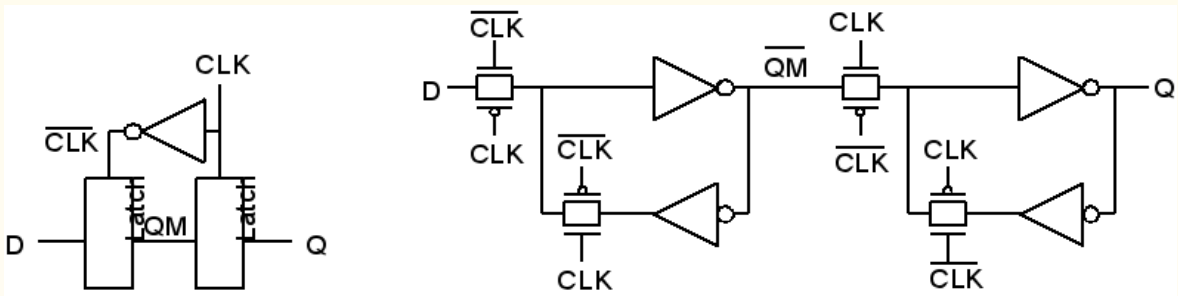


D Flip-Flop (D-Flop)

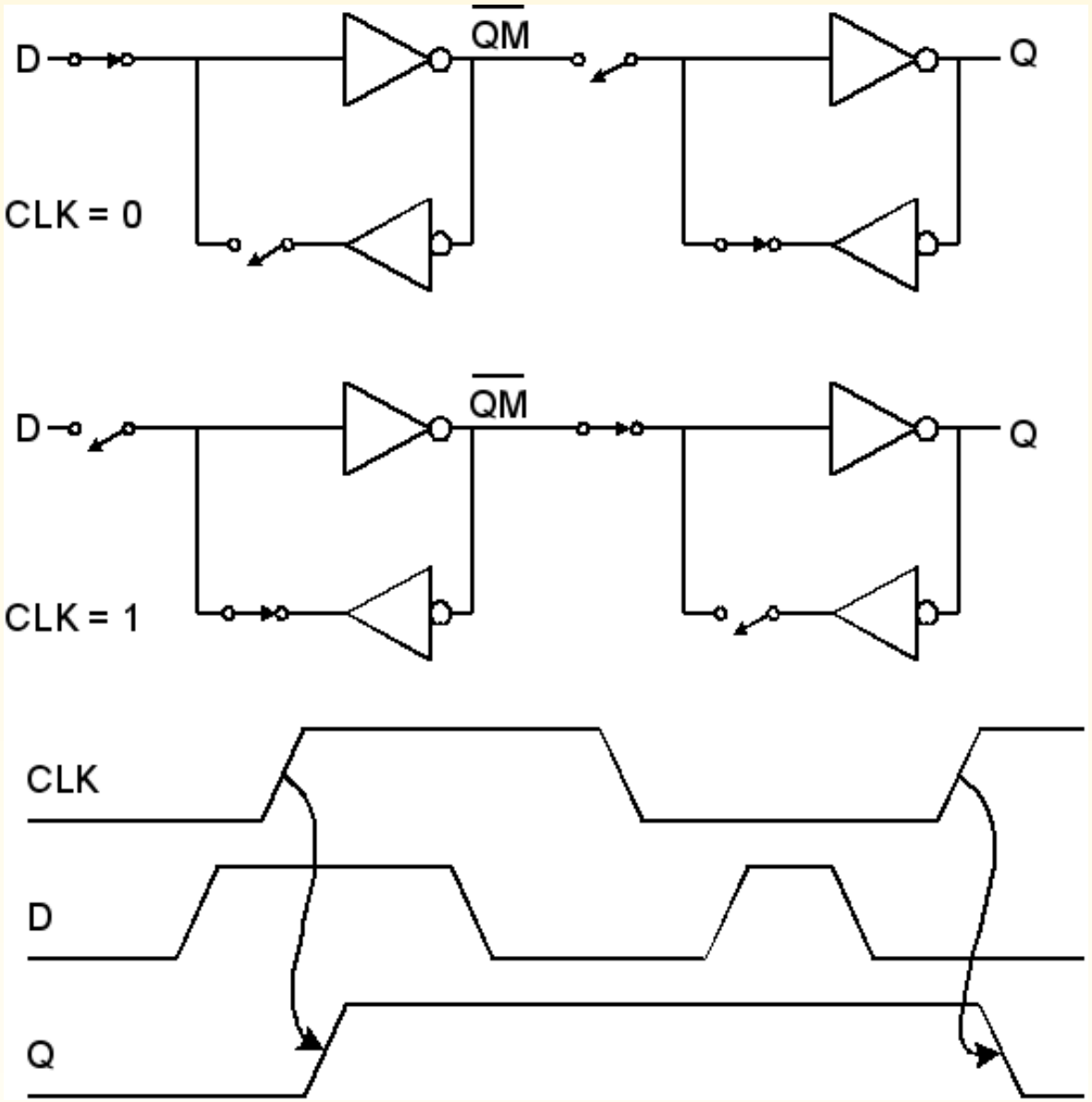
Another common storage element

- When CLK rises, D is copied to Q
- At all other times, Q holds its value
- **positive edge-triggered flip-flop** or **master-slave flip-flop**
- Built from "master" and "slave" D latches





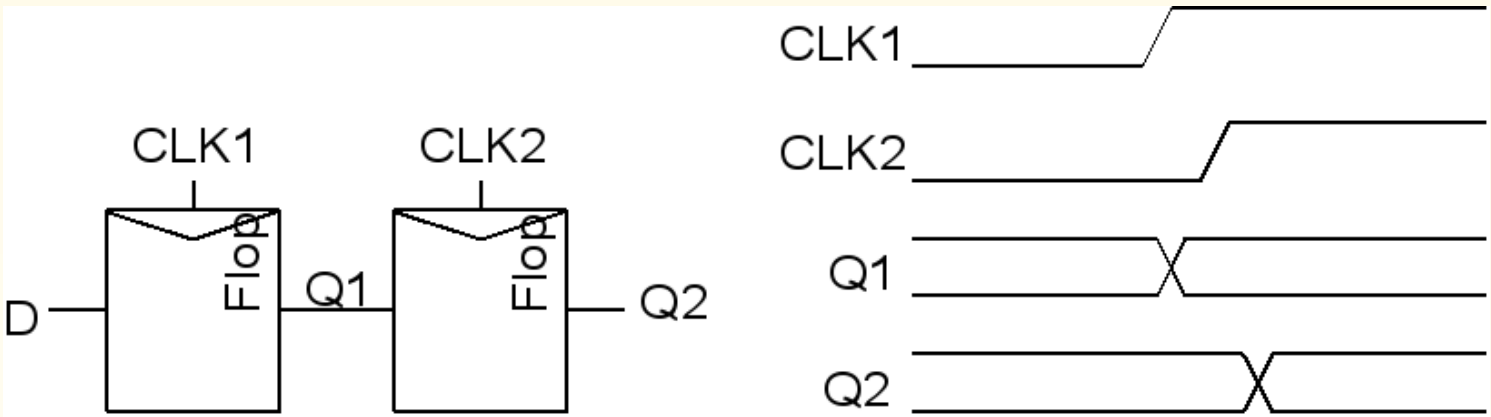
D Flip-Flop Operation



Race Condition – Hold Time Failure

Back-to-back flops can malfunction from clock skew

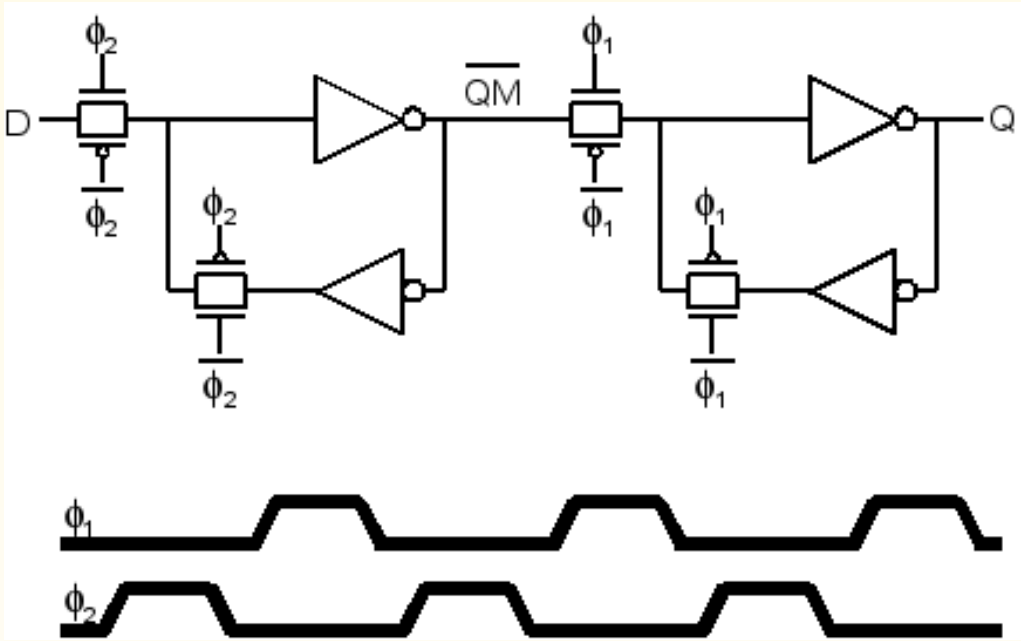
- Second flip-flop fires late
- Sees first flip-flop change and captures its result
- Called **hold-time failure** or **race condition**



Non-Overlapping Clocks

A simple way to prevent races

- This works as long as non-overlap exceeds clock skew
- Used in safe (conservative) designs
- Industry does not generally use this approach – managing skew more carefully instead



Gate Layout

Building a library of standard cells

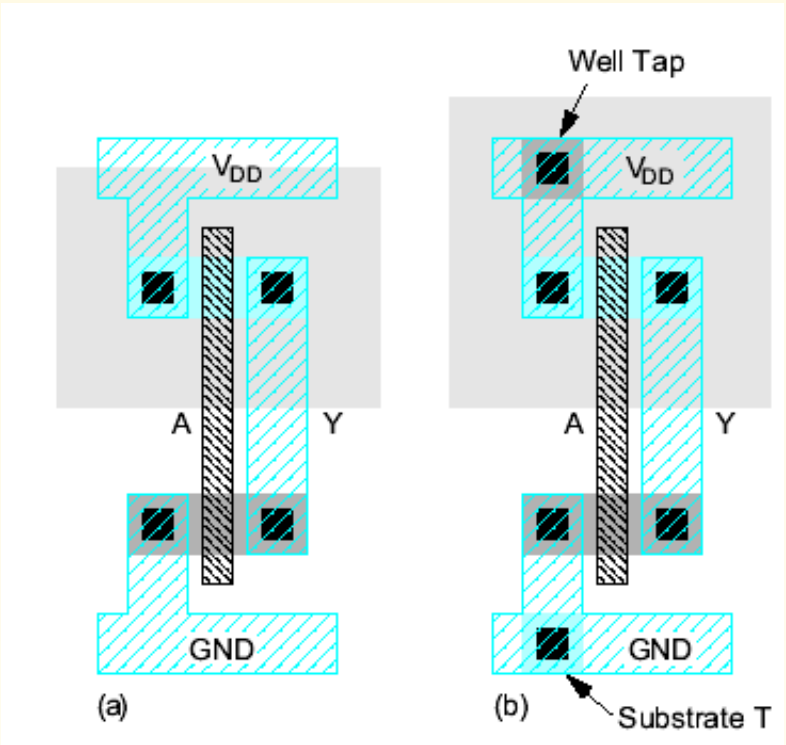
- Layout can be time consuming
- One solution is to have layouts of commonly used functions (Inverter, NAND, OR, MUX, etc.), designed to fit together very well

Standard cell design methodology

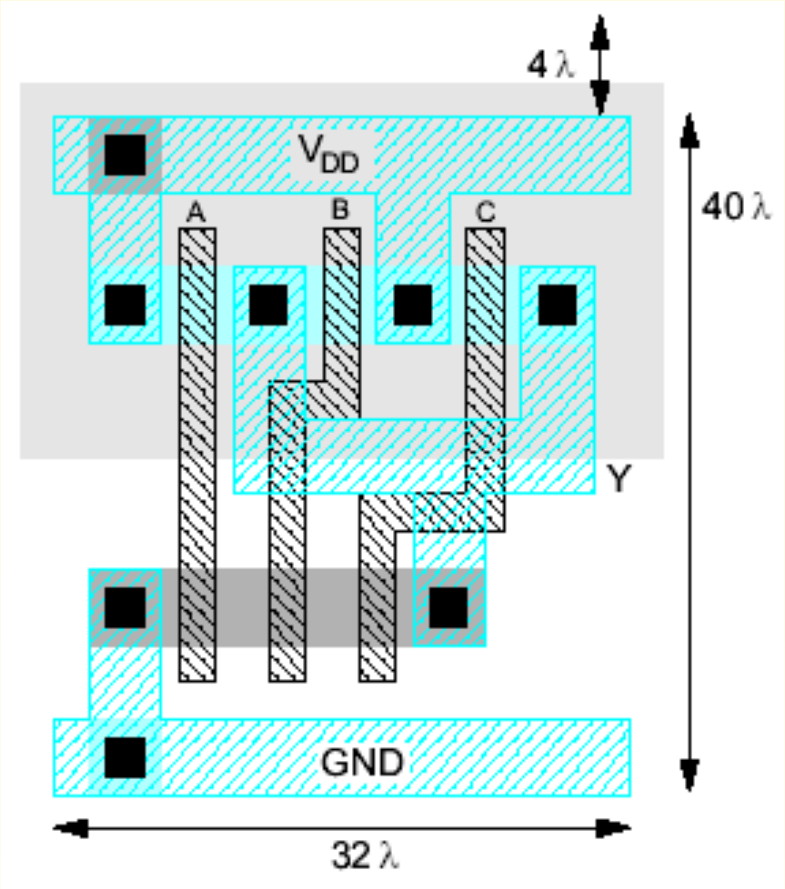
- V_{DD} and GND should abut (standard height)
- Adjacent gates should satisfy design rules
- nMOS at bottom and pMOS at top
- All gates include well and substrate contacts
- One of the large industry suppliers is ARM, others include TSMC and other foundries

Examples of Standard Cell Layout

Inverter



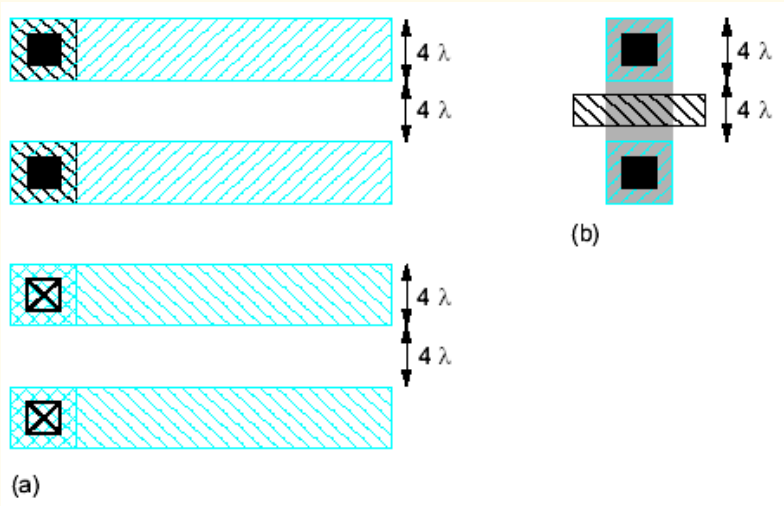
NAND3



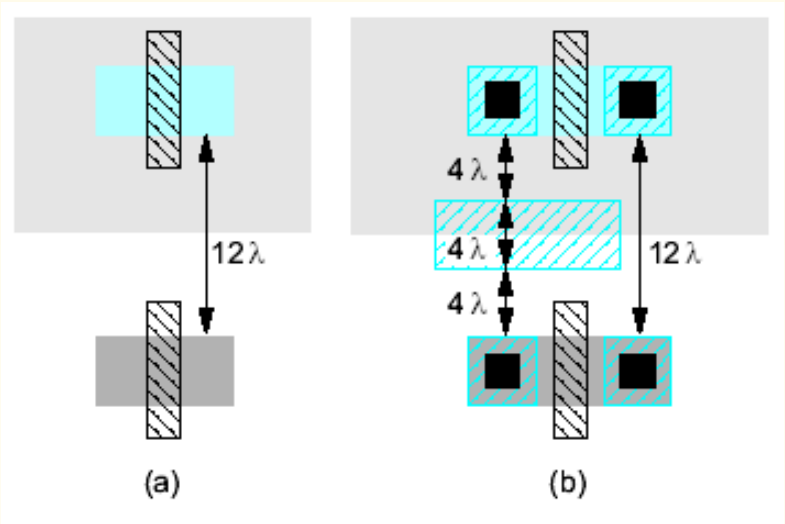
Horizontal N-diffusion and P-diffusion strips
Vertical Polysilicon gates
Metal1 V_{DD} rail at top, Metal1 GND rail at bottom
32λ by 40λ

Wiring Tracks and Well Spacing

Wiring Track is the space required for a wire
Example, 4λ width, 4λ spacing from neighbor $= 8\lambda$ pitch
Transistors also consume one wiring track

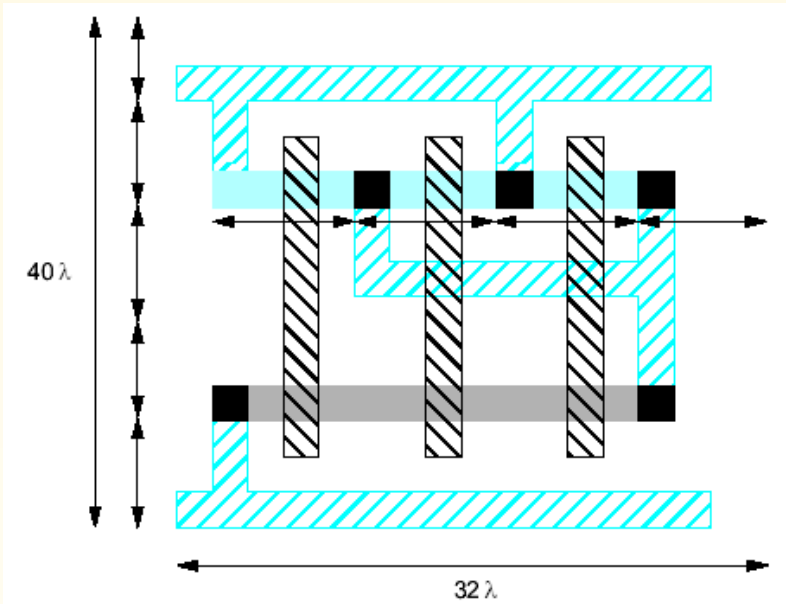


Example, **well spacing**: wells must surround transistors by 6λ
Implies 12λ between opposite transistor flavors
Leaves room for one wire track

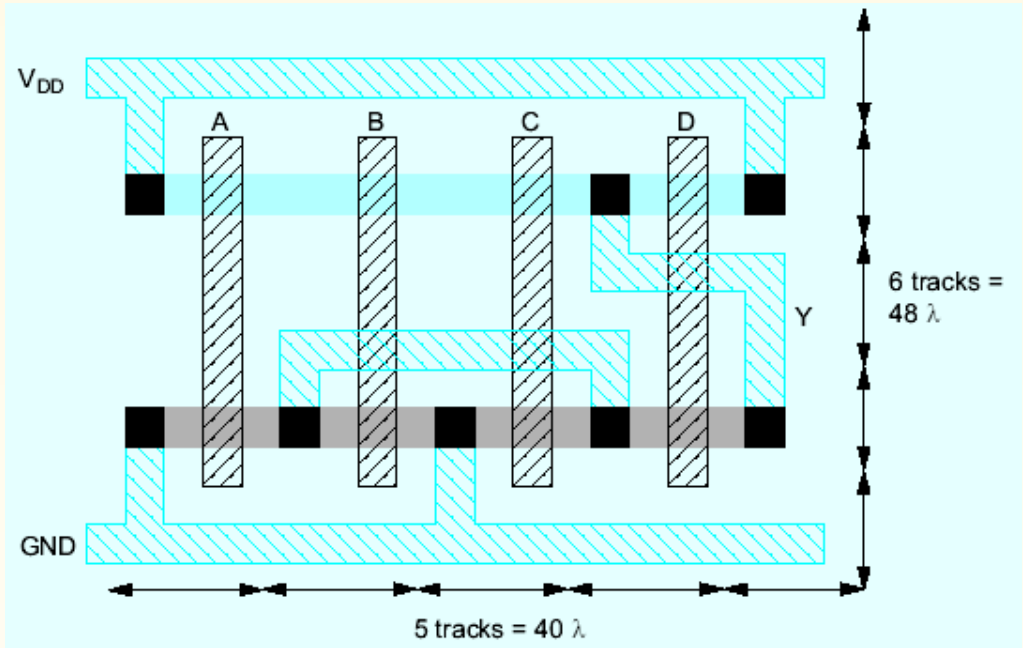


Example of Area Estimation

Estimate area by counting wiring tracks
Multiply by 8 to express in λ

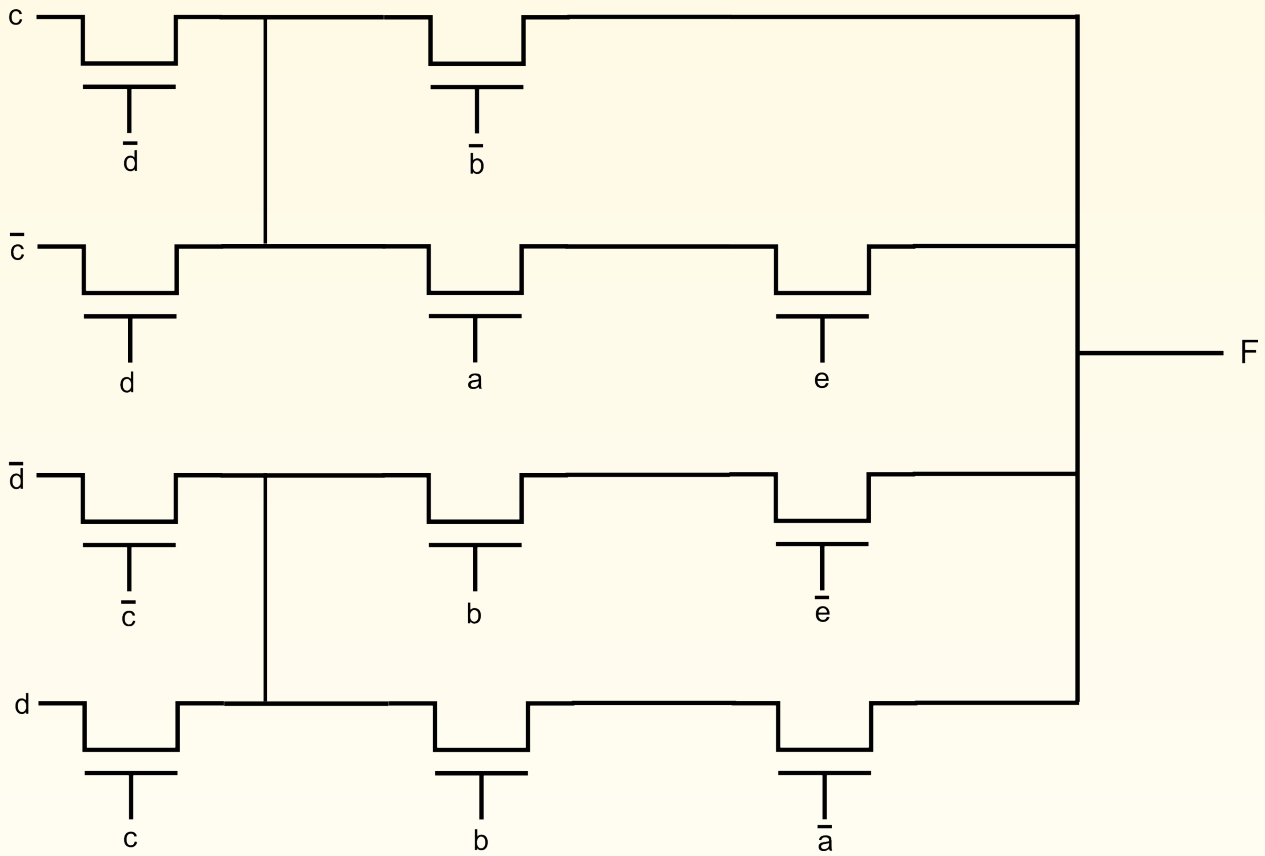


Estimating area of O3AI
Sketch a stick diagram and estimate area

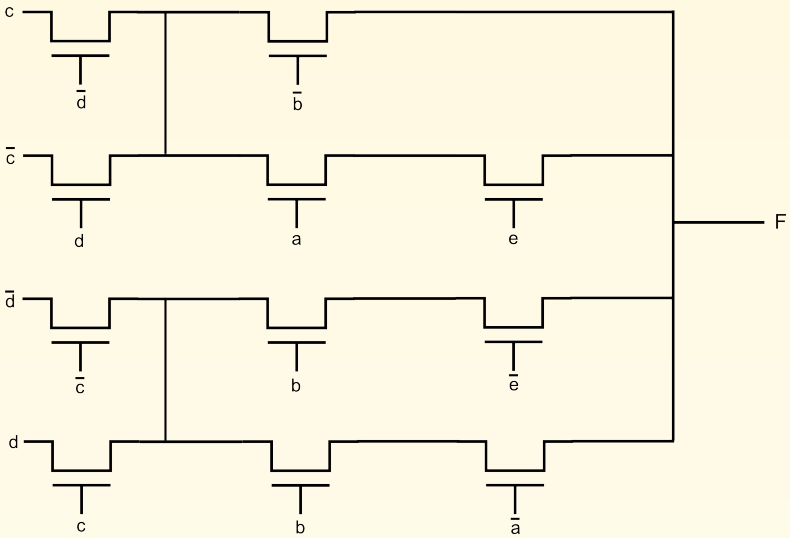


Example Circuit 1

Fill in the Karnaugh map to represent the Boolean function implemented by the pass-transistor circuit.



Example Circuit 1



F(a,b,c,d,e):

a

c

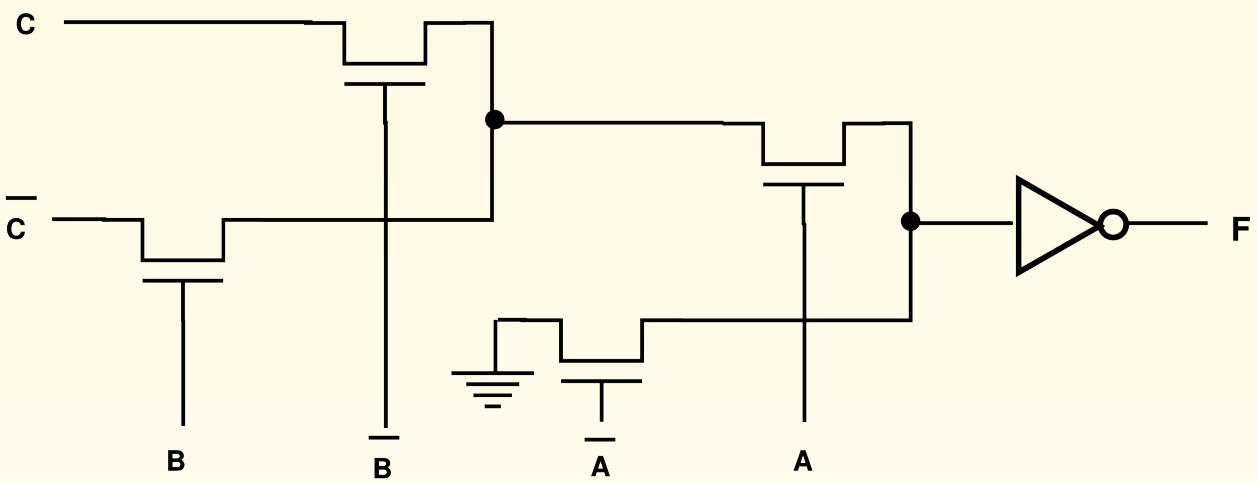
e e

b d

0 0	1 0	5 1	4 1	20 1	21 1	17 0	16 0
2 1	3 1	7 0	6 0	22 0	23 0	19 1	18 1
10 0	11 0	15 1	14 1	30 1	31 0	27 1	26 0
8 1	9 1	13 0	12 0	28 0	29 1	25 0	24 1

Example Circuit 2

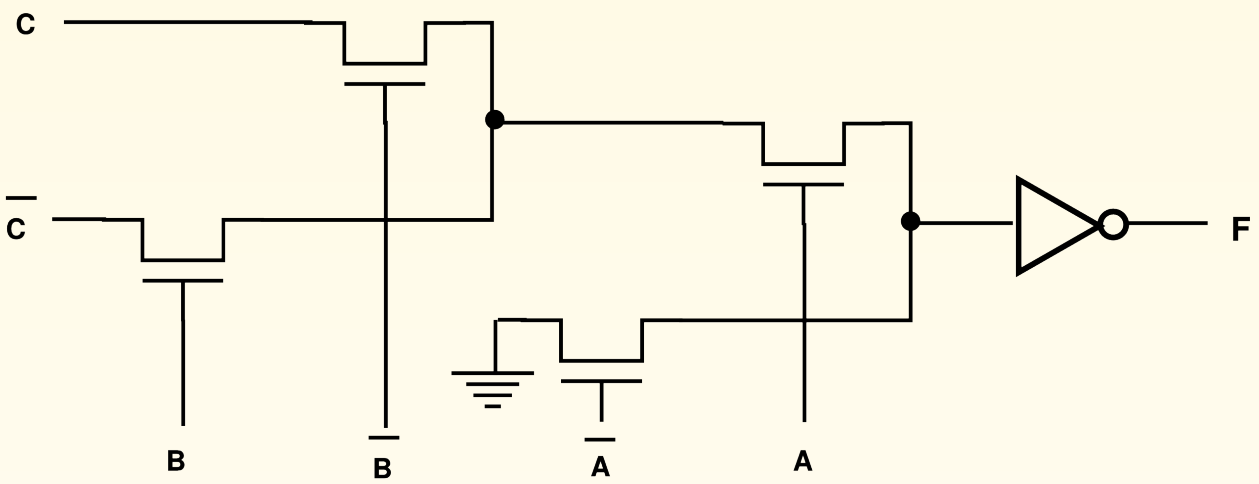
Find the function, F, implemented by the following circuit



$$\overline{A} + BC + \overline{B}\,\overline{C}$$

Example Circuit 2

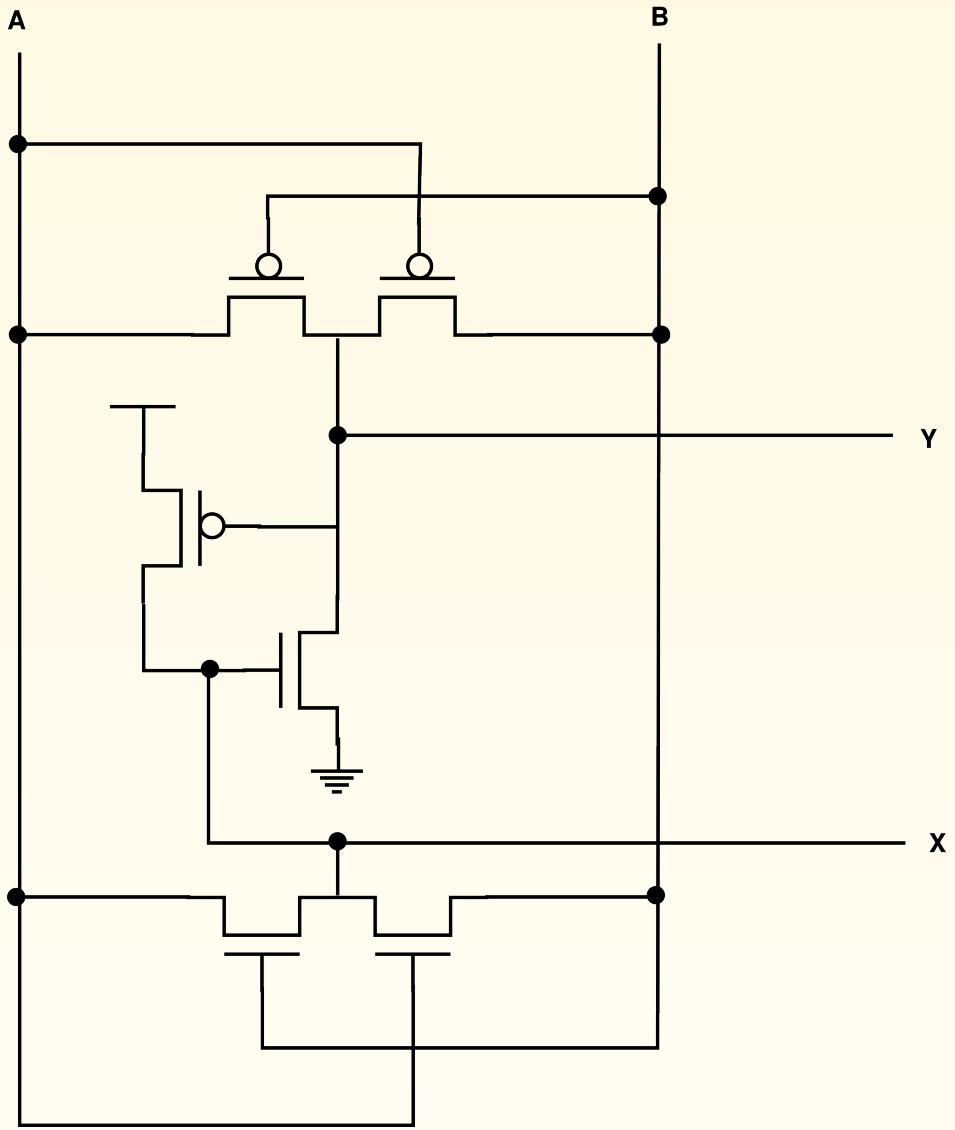
Find the function, F, implemented by the following circuit



$$\overline{A} + BC + \overline{B}\,\overline{C}$$

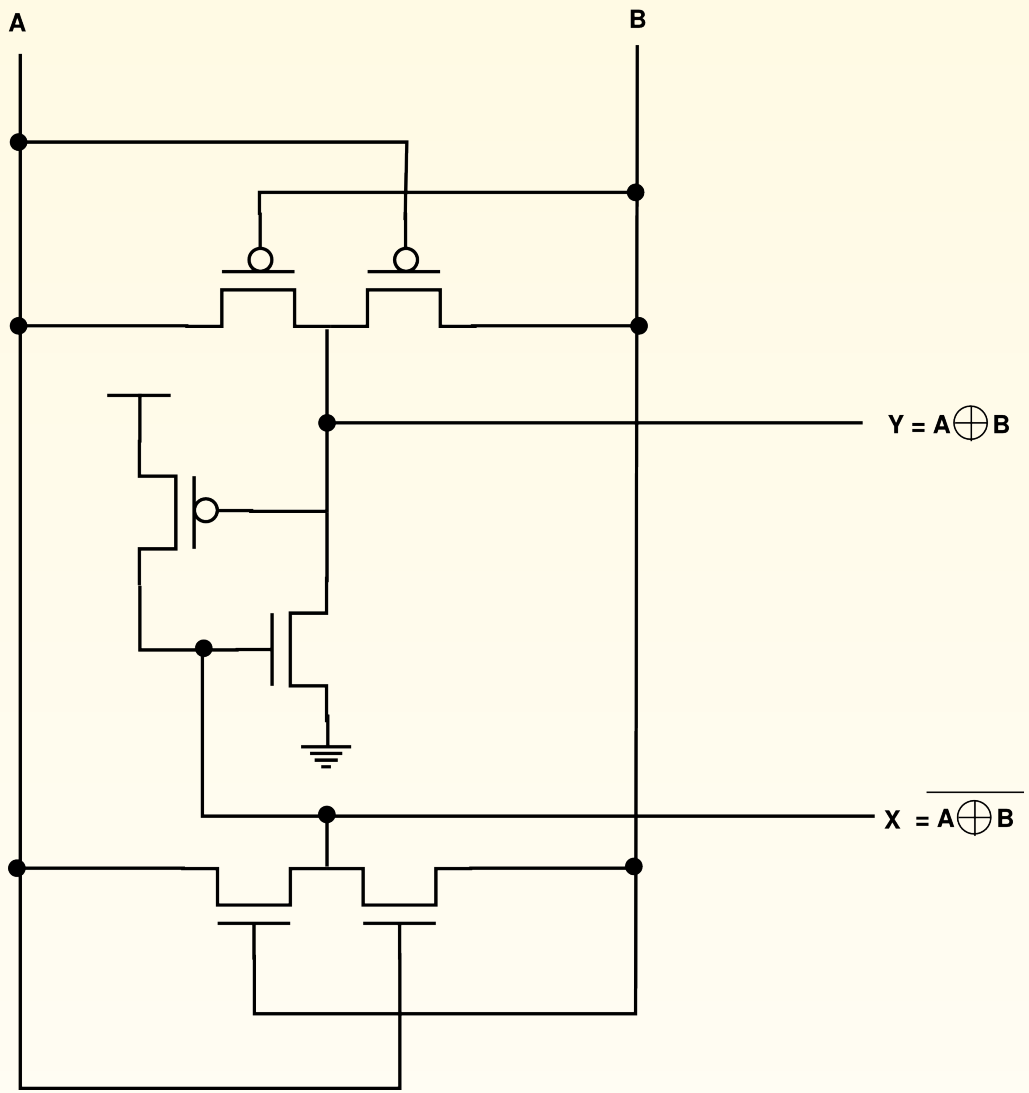
Example Circuit 3

Find the functions X and Y implemented by the following circuit



Example Circuit 3

Find the functions X and Y implemented by the following circuit



Functions to Circuits

Label the circuit so that it implements the function:
$F = a \cdot (b \cdot c + \overline{b} \cdot \overline{c})$