






# VLSI Design



## Circuits & Layout

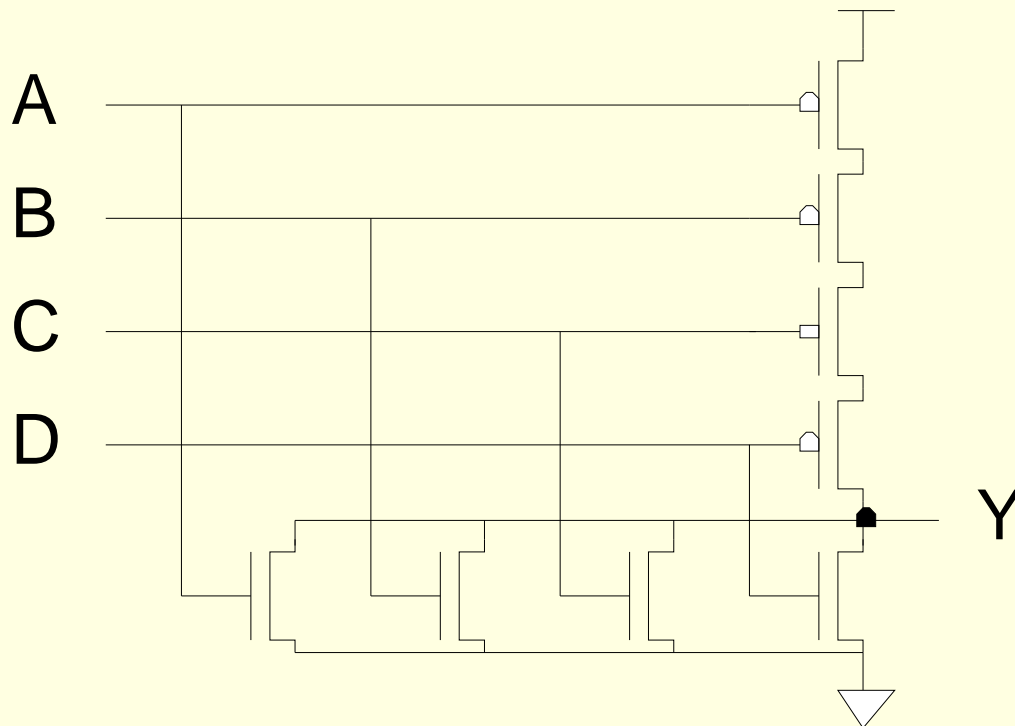
# Outline

---

- CMOS Gate Design
- Pass Transistors
- CMOS Latches & Flip-Flops
- Standard Cell Layouts
- Stick Diagrams

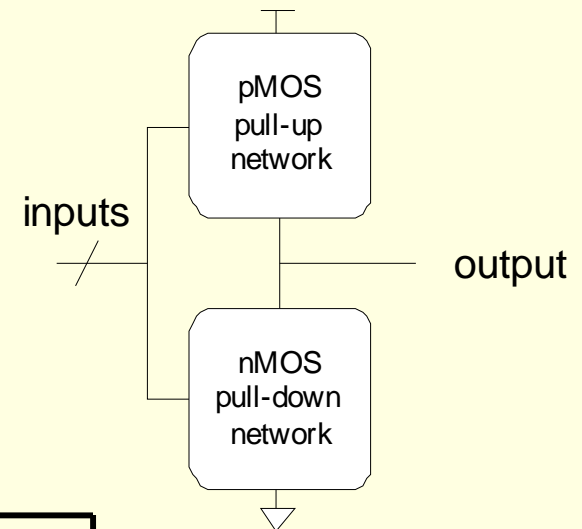
# CMOS Gate Design

- A 4-input CMOS NOR gate



# Complementary CMOS

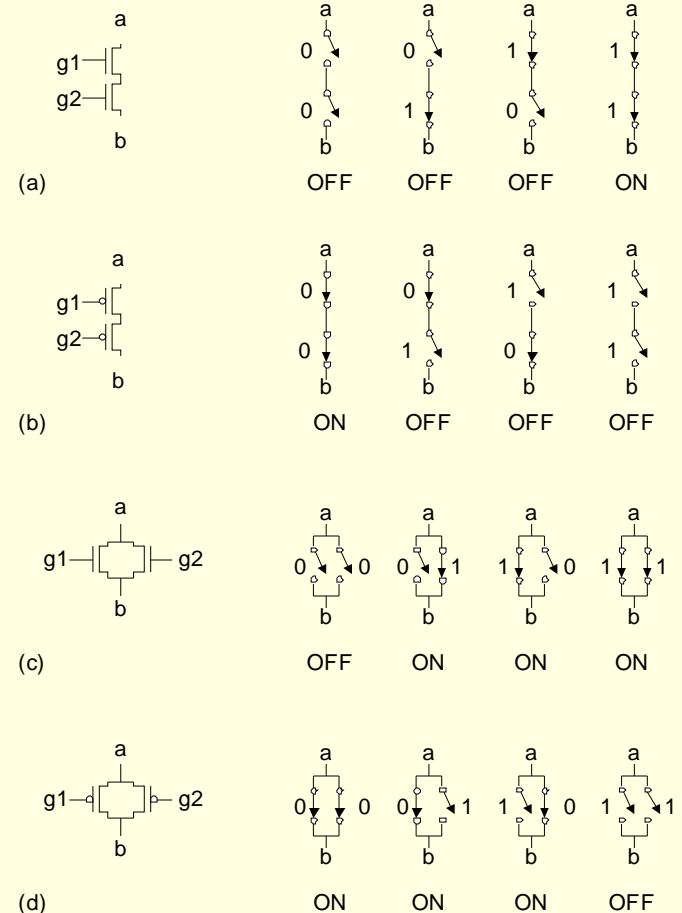
- Complementary CMOS logic gates
  - nMOS *pull-down network*
  - pMOS *pull-up network*
  - a.k.a. static CMOS



	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

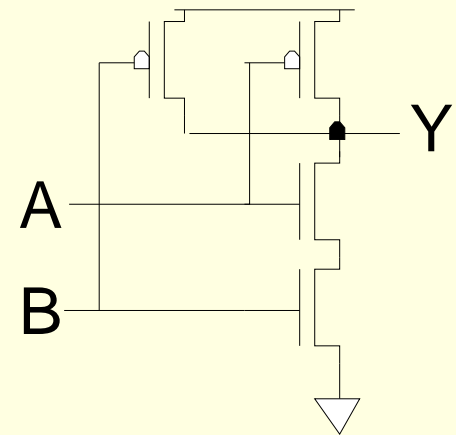


# Conduction Complement

- 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*

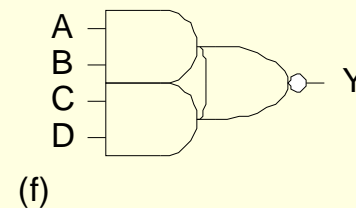
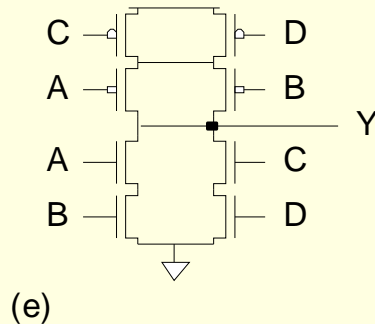
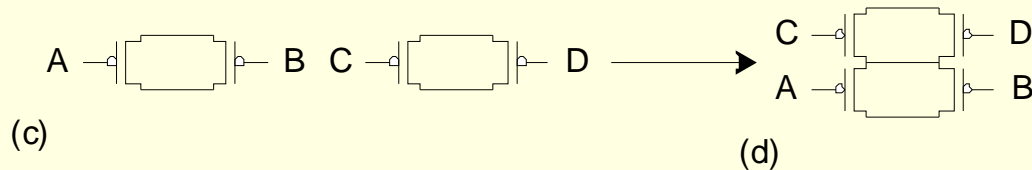
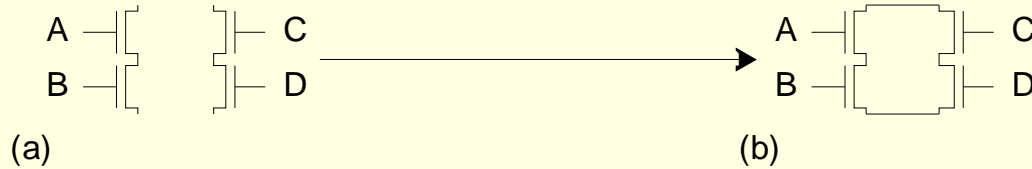
- Pull-up network is **complement** of pull-down
- Parallel  $\rightarrow$  series, series  $\rightarrow$  parallel

# Compound Gates

- *Compound gates* can do any inverting function

- Ex: AND-AND-OR-INV (AOI22)

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



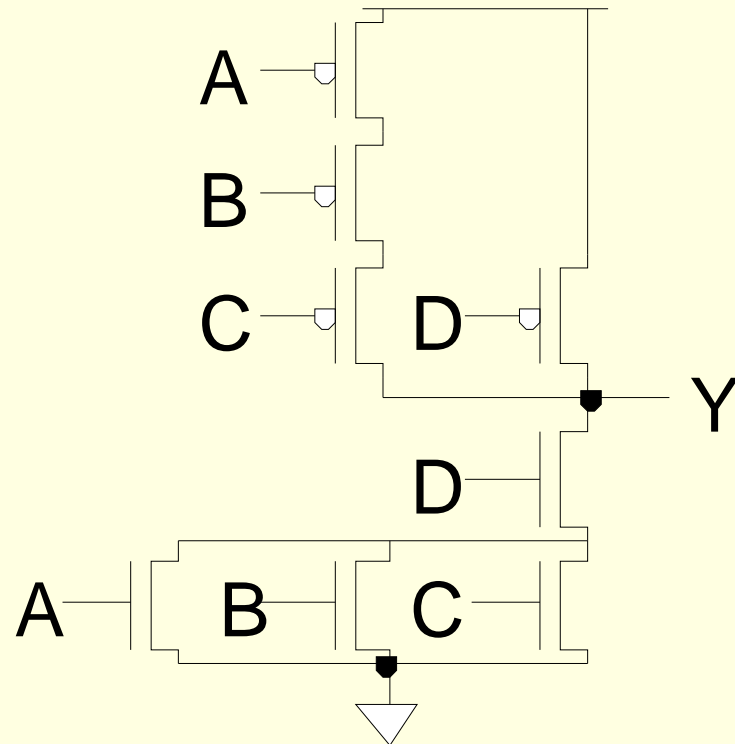
# Example: O3AI

---

- $Y = \overline{(A + B + C)} \bullet D$

# Example: O3AI

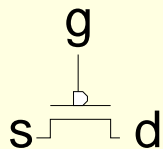
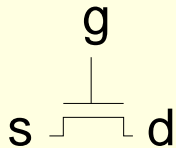
■ 
$$Y = \overline{(A + B + C)} \bullet D$$



# Pass Transistors

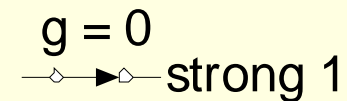
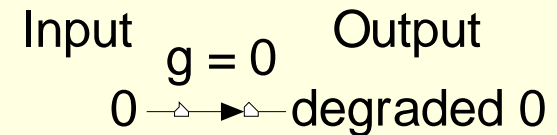
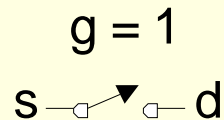
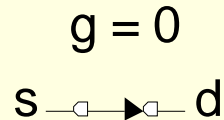
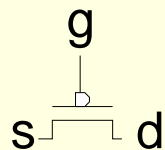
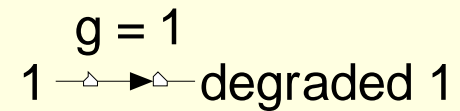
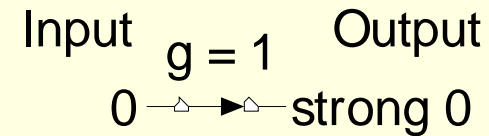
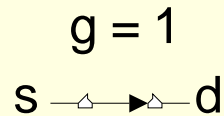
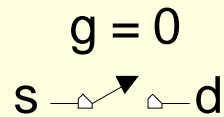
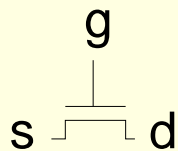
---

- Transistors can be used as switches



# Pass Transistors

- Transistors can be used as switches



# 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
- Thus PMOS are best for pull-up network

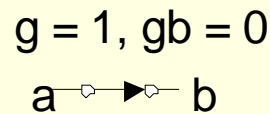
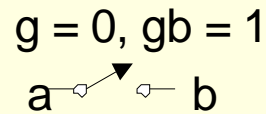
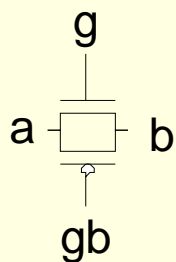
# Transmission Gates

---

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

# Transmission Gates

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

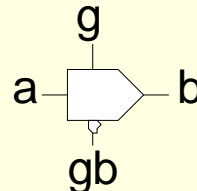
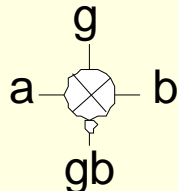
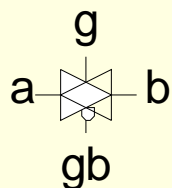


Input

Output

$g = 1, gb = 0$   
 $0 \rightarrow$  strong 0

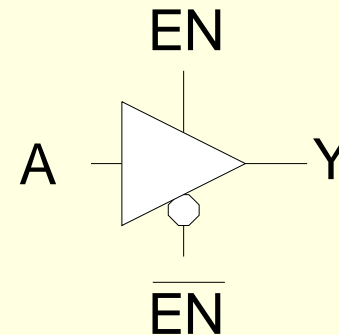
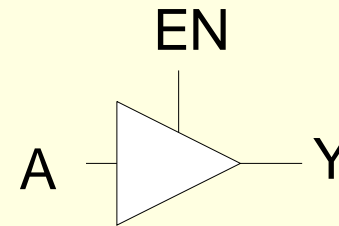
$g = 1, gb = 0$   
 $1 \rightarrow$  strong 1



# Tristates

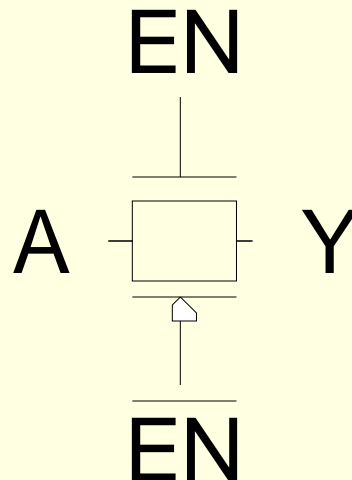
- *Tristate buffer* produces Z when not enabled

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



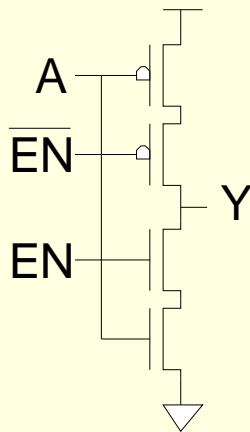
# Nonrestoring Tristate

- Transmission gate acts as tristate buffer
  - Only two transistors
  - But *nonrestoring*
    - Noise on A is passed on to Y (after several stages, the noise may degrade the signal beyond recognition)



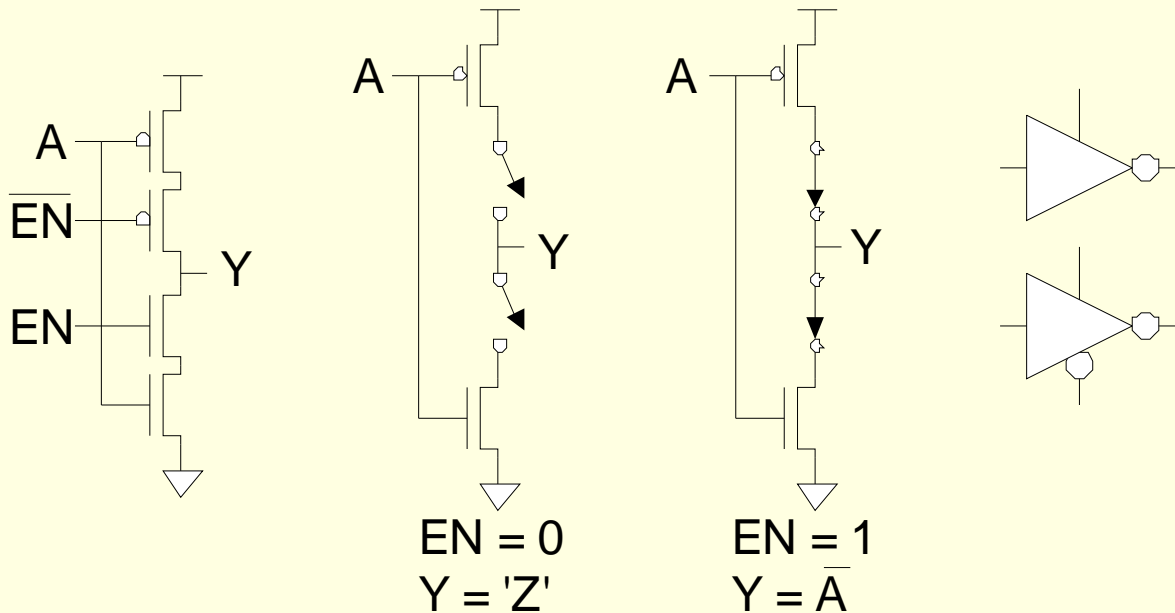
# Tristate Inverter

- Tristate inverter produces restored output
- Note however that the Tristate buffer
  - ignores the conduction complement rule because we want a Z output



# Tristate Inverter

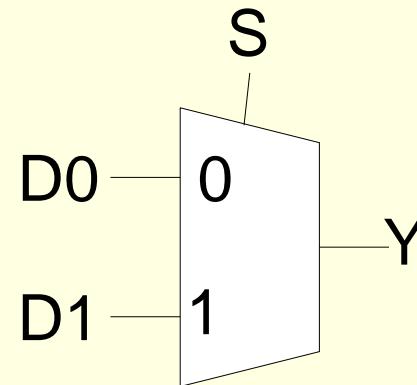
- Tristate inverter produces restored output
- Note however that the Tristate buffer
  - ignores the conduction complement rule because we want a Z output



# Multiplexers

- 2:1 *multiplexer* chooses between two inputs

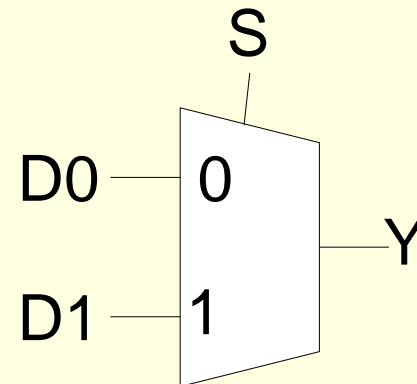
S	D1	D0	Y
0	X	0	
0	X	1	
1	0	X	
1	1	X	



# 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



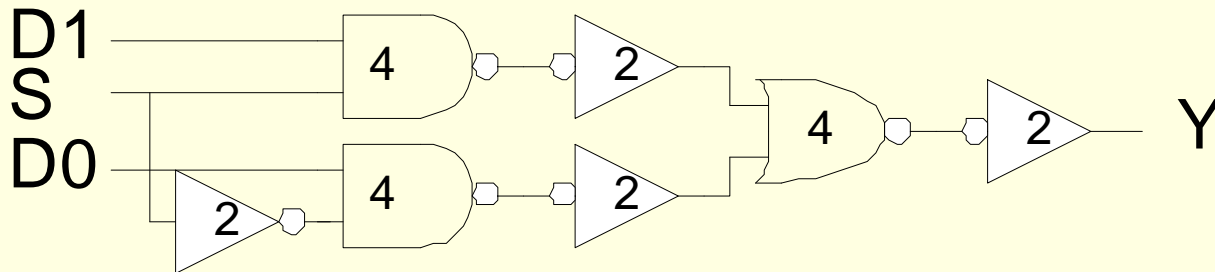
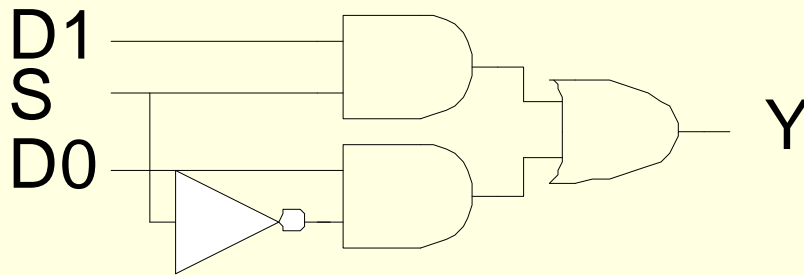
# Gate-Level Mux Design

---

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

# Gate-Level Mux Design

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



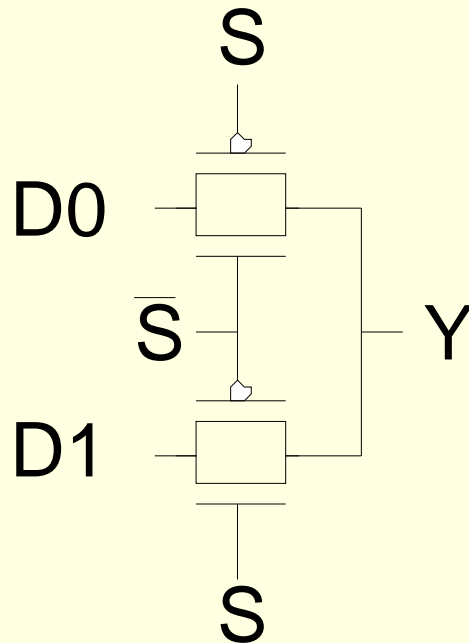
# Transmission Gate Mux

---

- Nonrestoring mux uses two transmission gates

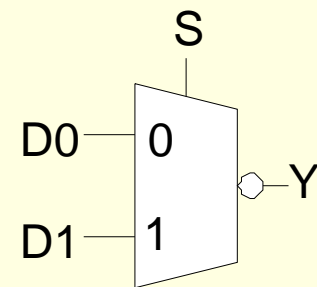
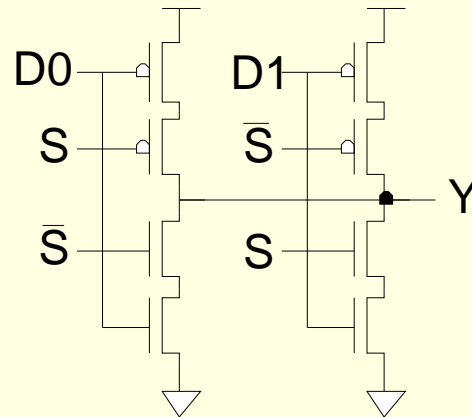
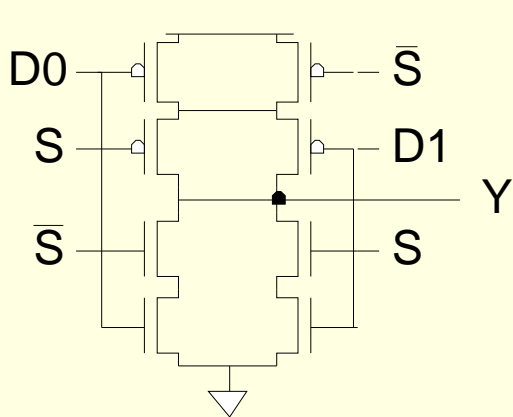
# Transmission Gate Mux

- 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



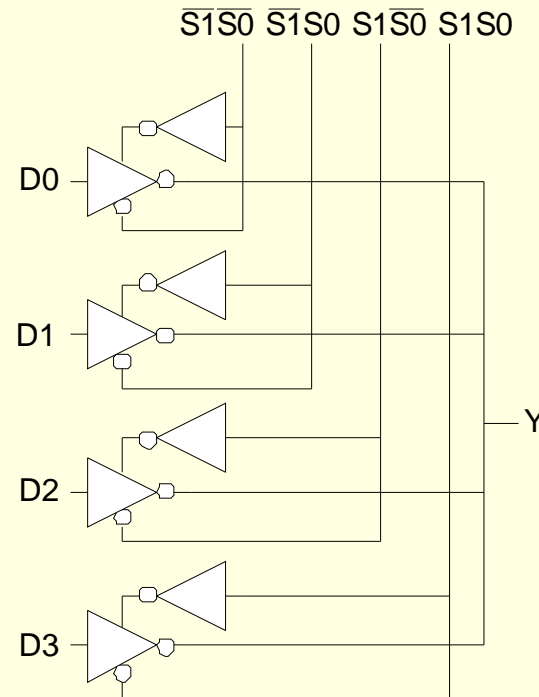
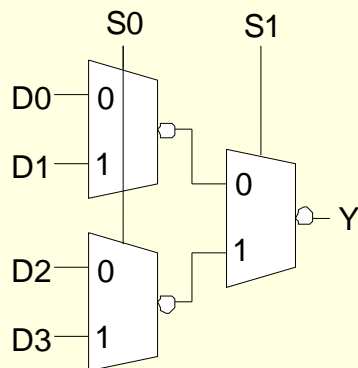
# 4:1 Multiplexer

---

- 4:1 mux chooses one of 4 inputs using two selects

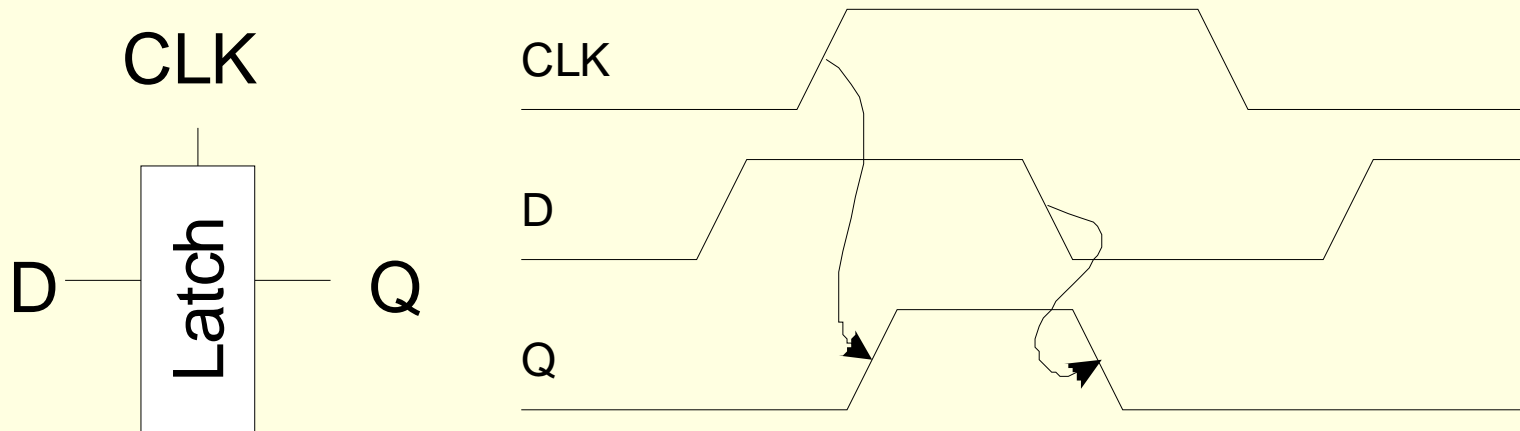
# 4:1 Multiplexer

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



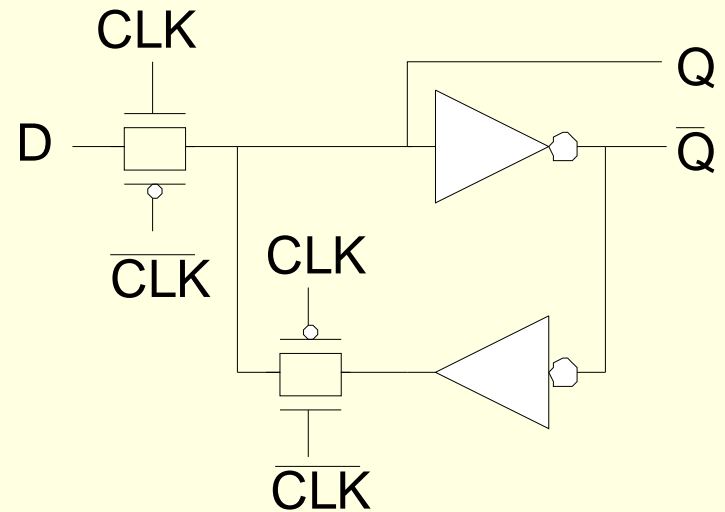
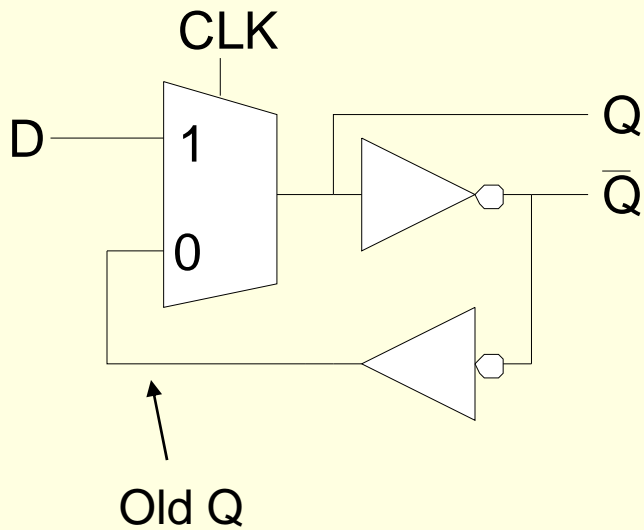
# D Latch

- When  $CLK = 1$ , latch is *transparent*
  - Q follows D (a buffer with a **D**elay)
- When  $CLK = 0$ , the latch is *opaque*
  - Q holds its last value independent of D
- a.k.a. *transparent latch* or *level-sensitive latch*

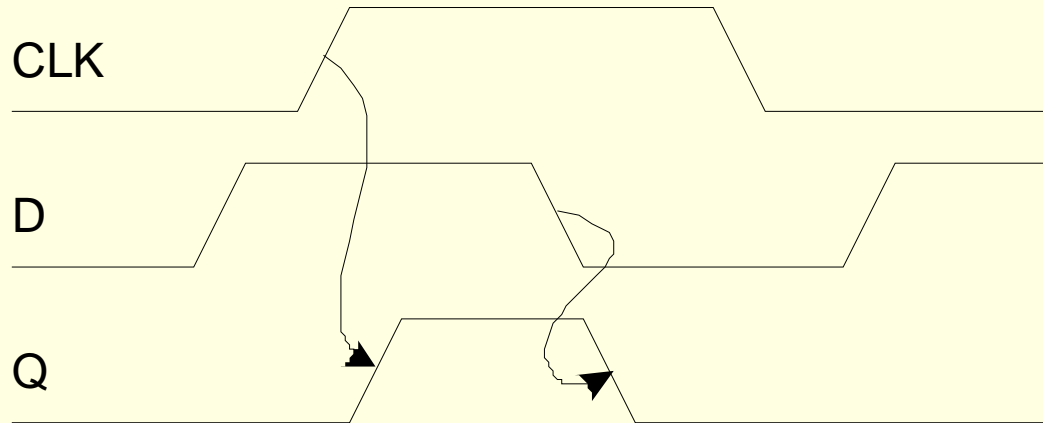
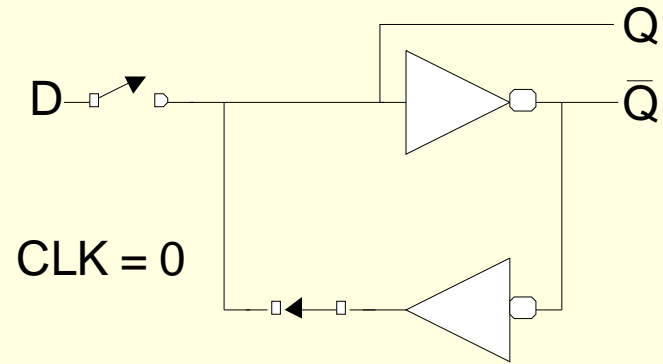
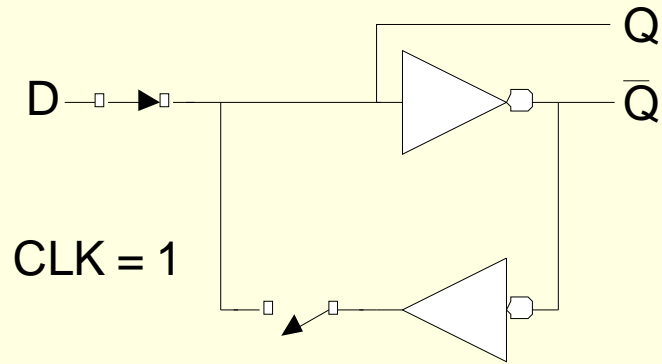


# D Latch Design

- Multiplexer chooses D or old Q

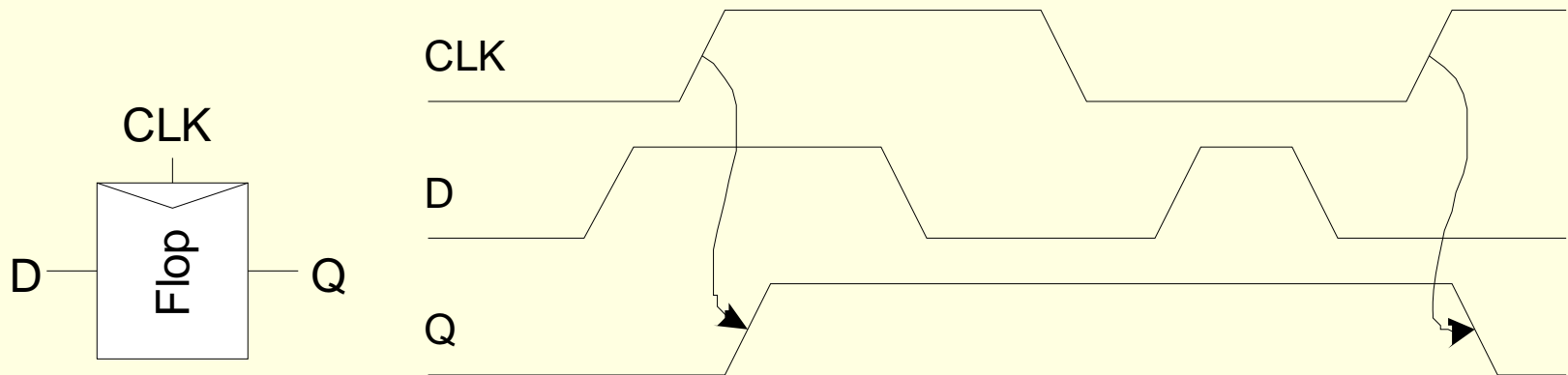


# D Latch Operation



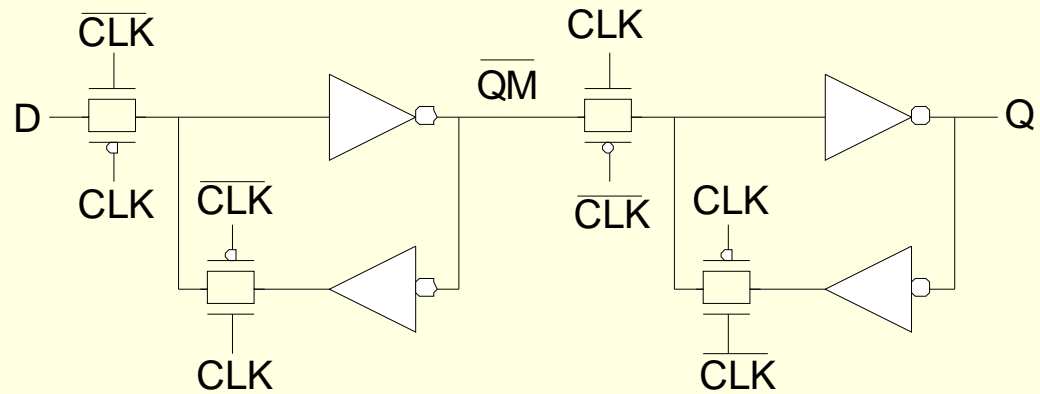
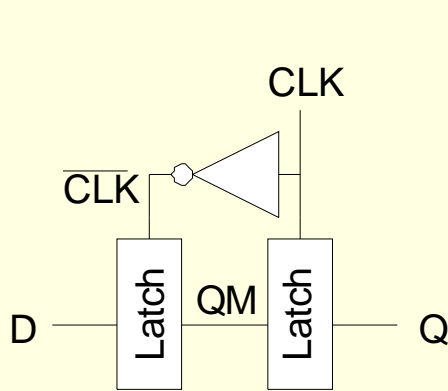
# D Flip-flop

- When CLK rises, D is copied to Q
- At all other times, Q holds its value
- a.k.a. *positive edge-triggered flip-flop, master-slave flip-flop*



# D Flip-flop Design

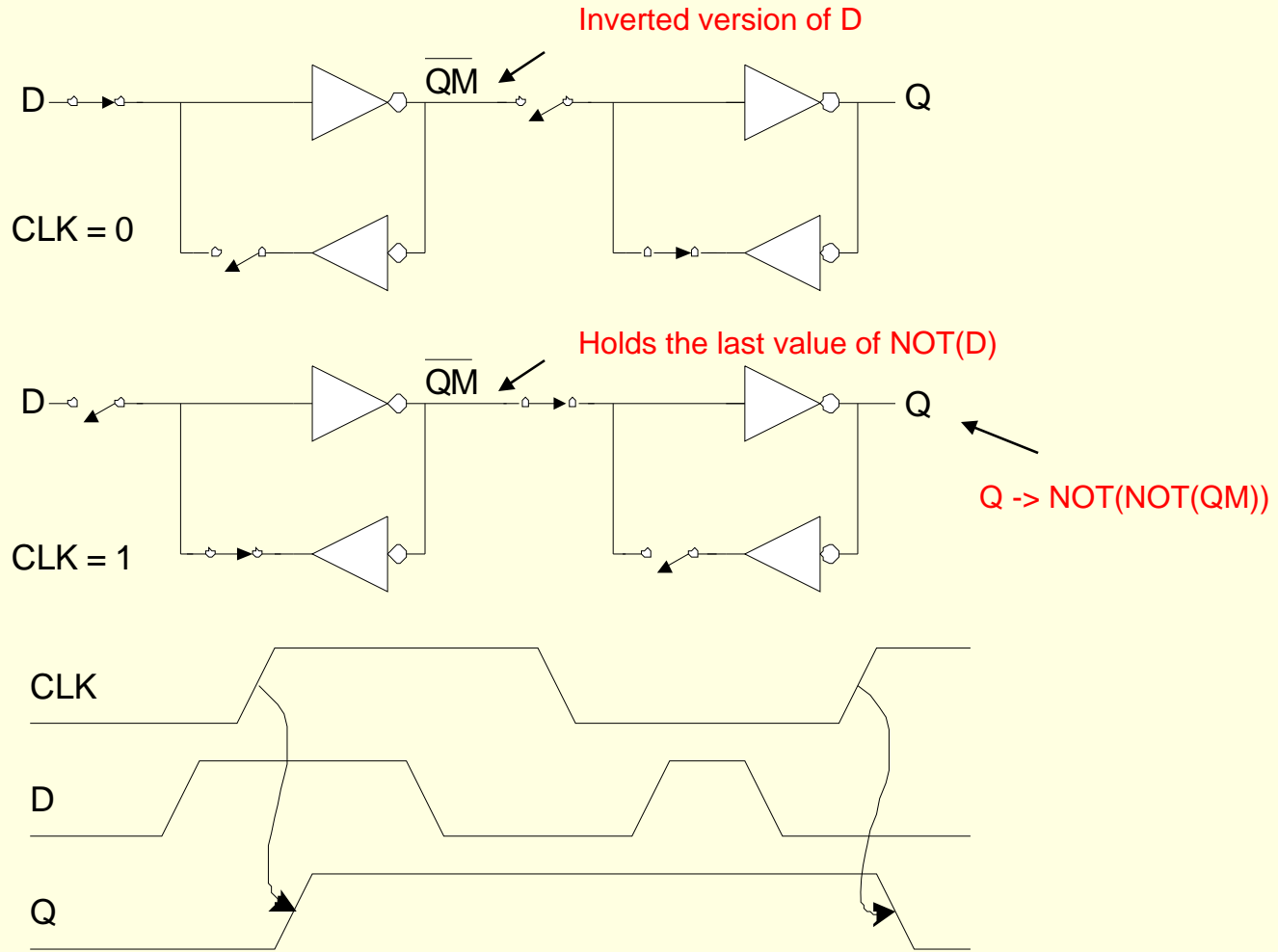
- Built from master and slave D latches



A "negative level-sensitive" latch

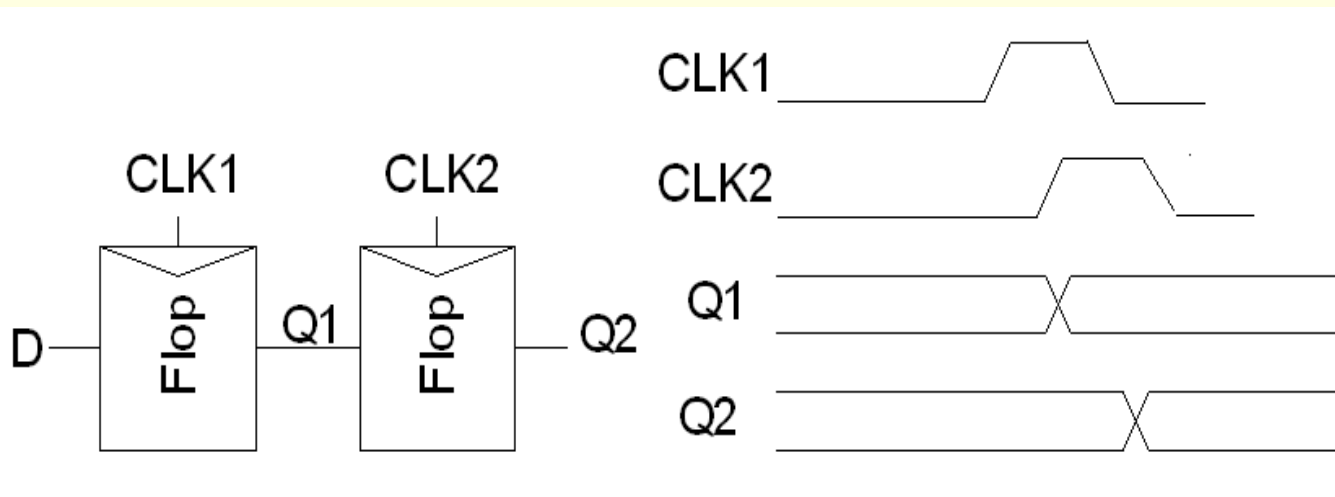
A "positive level-sensitive" latch

# D Flip-flop Operation



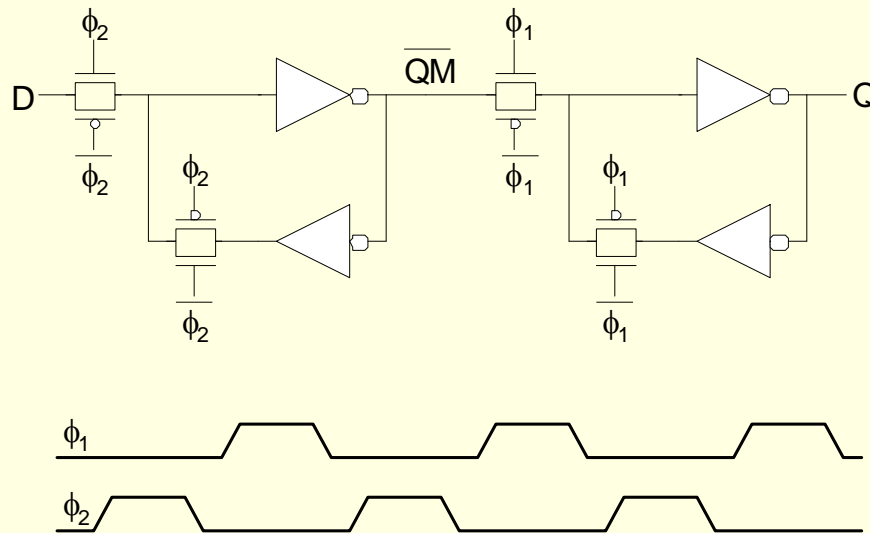
# Race Condition

- Back-to-back flops can malfunction from clock skew
  - Second flip-flop fires Early
  - Sees first flip-flop change and captures its result
  - Called *hold-time failure* or *race condition*



# Nonoverlapping Clocks

- Nonoverlapping clocks can prevent races
  - As long as nonoverlap exceeds clock skew
- Good for safe design
  - Industry manages skew more carefully instead

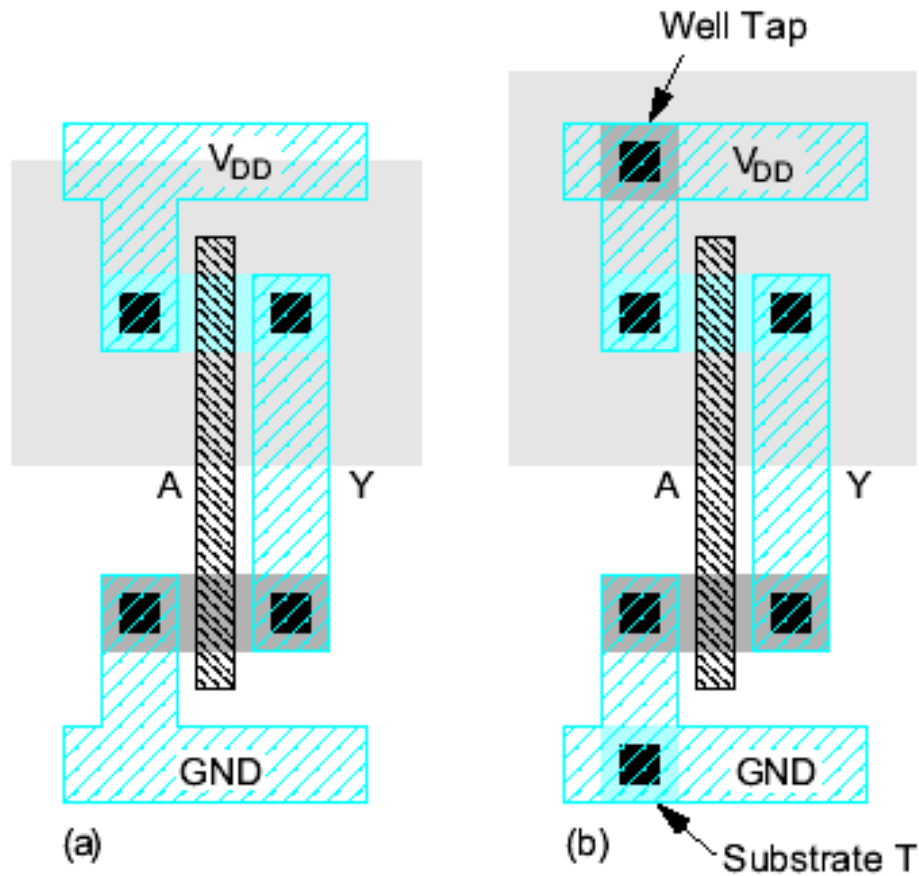


# Gate Layout

---

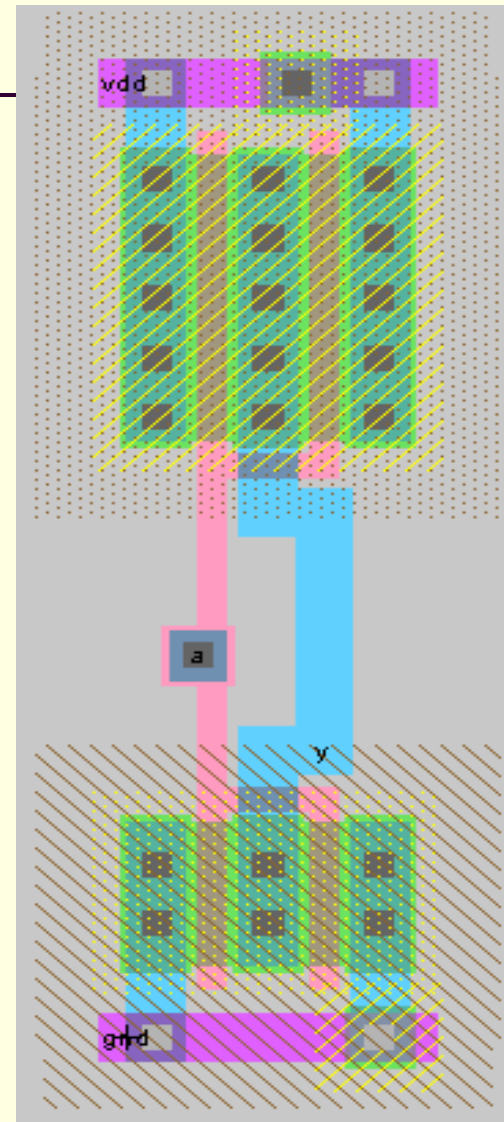
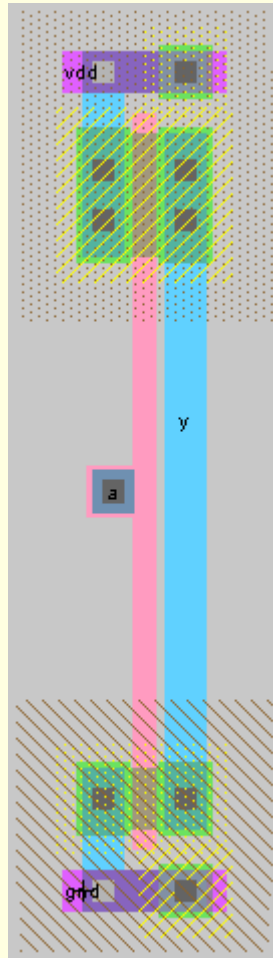
- Layout can be very time consuming
  - Design gates to fit together nicely
  - Build a library of standard cells
  - Must follow a technology rule
  
- Standard cell design methodology
  - $V_{DD}$  and GND should abut (standard height)
  - Adjacent gates should satisfy design rules
  - nMOS at bottom and pMOS at top
  - All gates include well and substrate contacts

# Example: Inverter



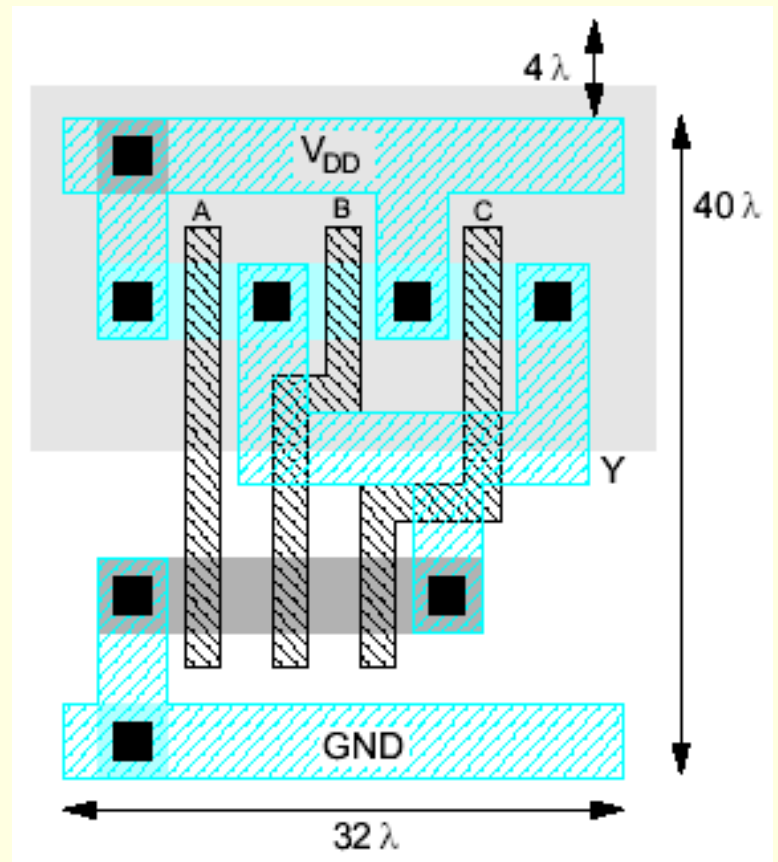
# Layout using Electric

Inverter, contd..

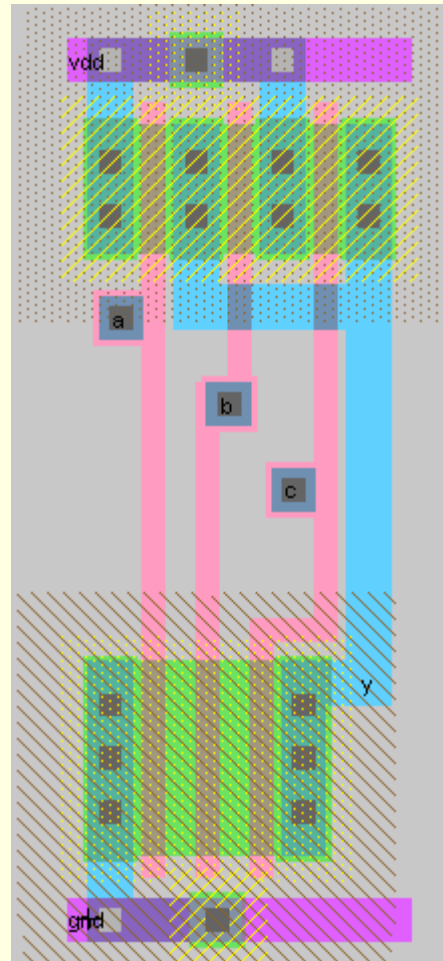


# Example: NAND3

- Horizontal N-diffusion and p-diffusion strips
- Vertical polysilicon gates
- Metal1  $V_{DD}$  rail at top
- Metal1 GND rail at bottom
- $32 \lambda$  by  $40 \lambda$

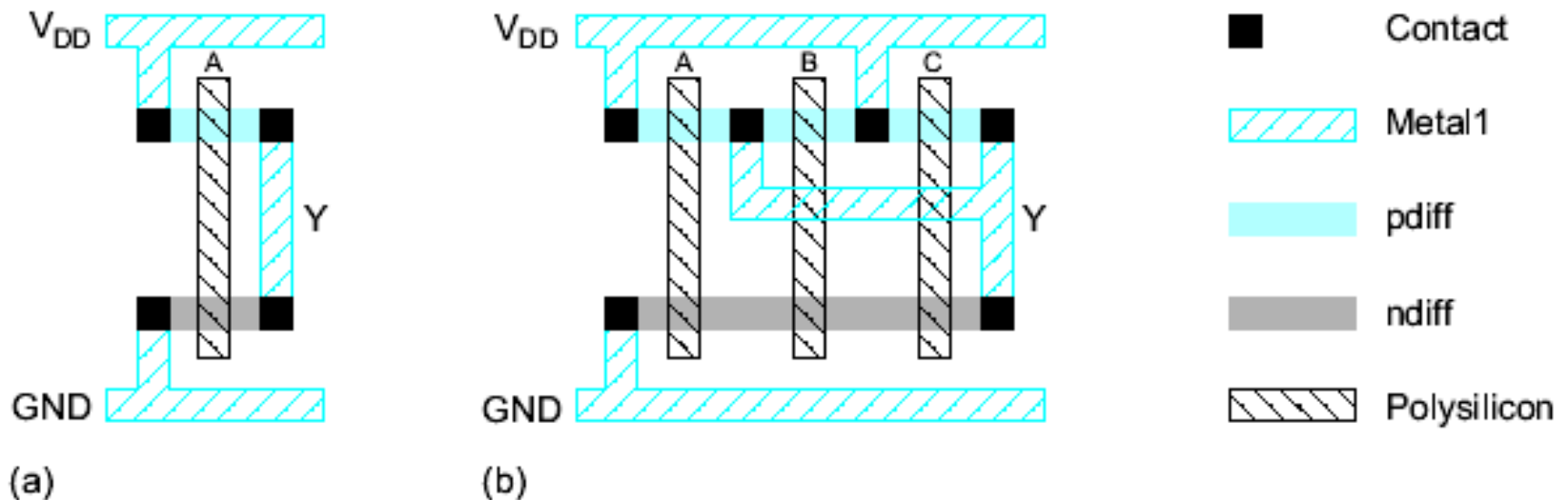


# NAND3 (using Electric), contd.



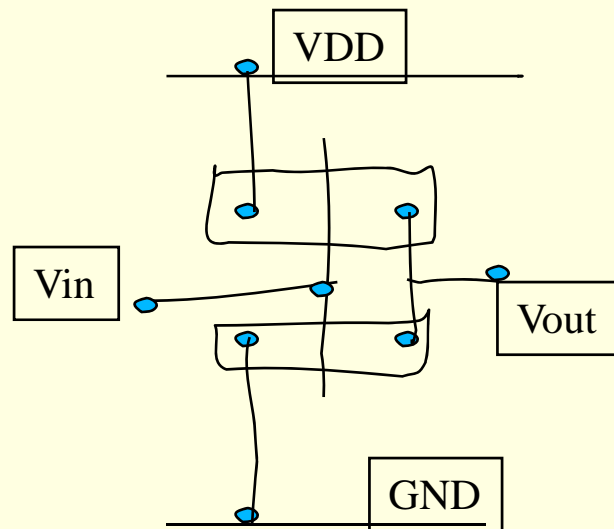
# Stick Diagrams

- *Stick diagrams* help plan layout quickly
  - Need not be to scale
  - Draw with color pencils or dry-erase markers



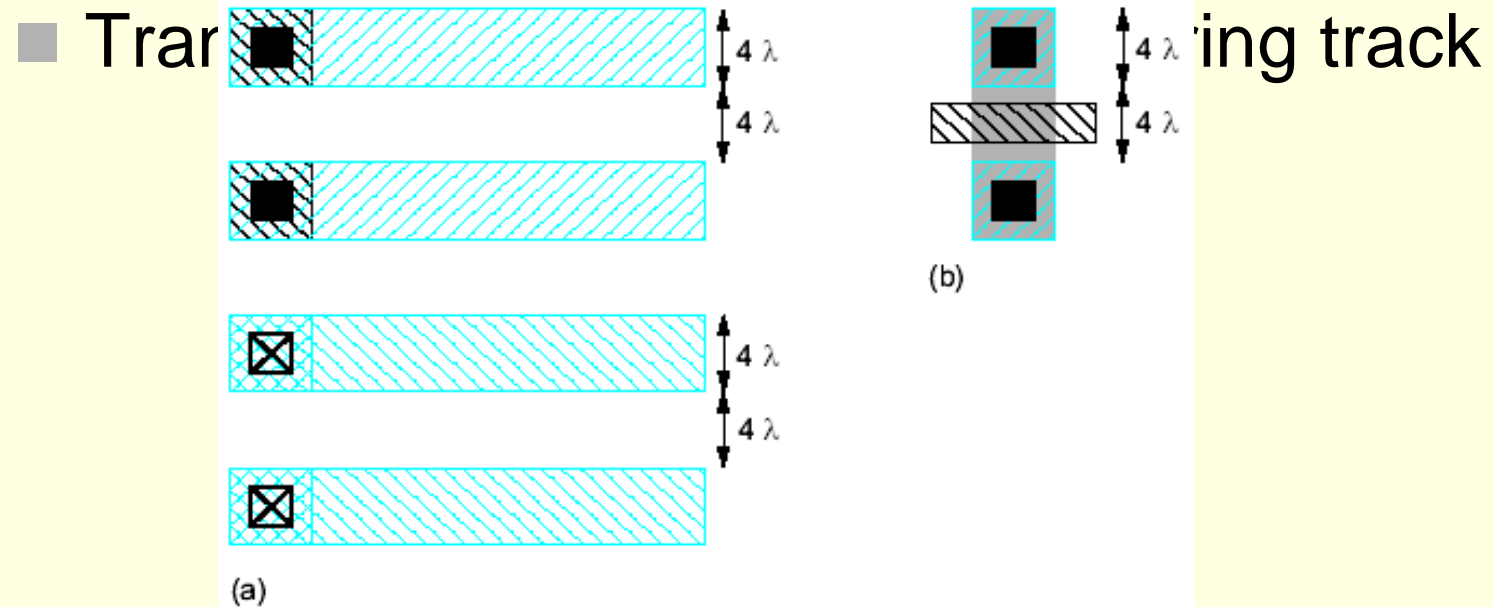
# Stick Diagrams

- *Stick diagrams* help plan layout quickly
  - Need not be to scale
  - Draw with color pencils or dry-erase markers



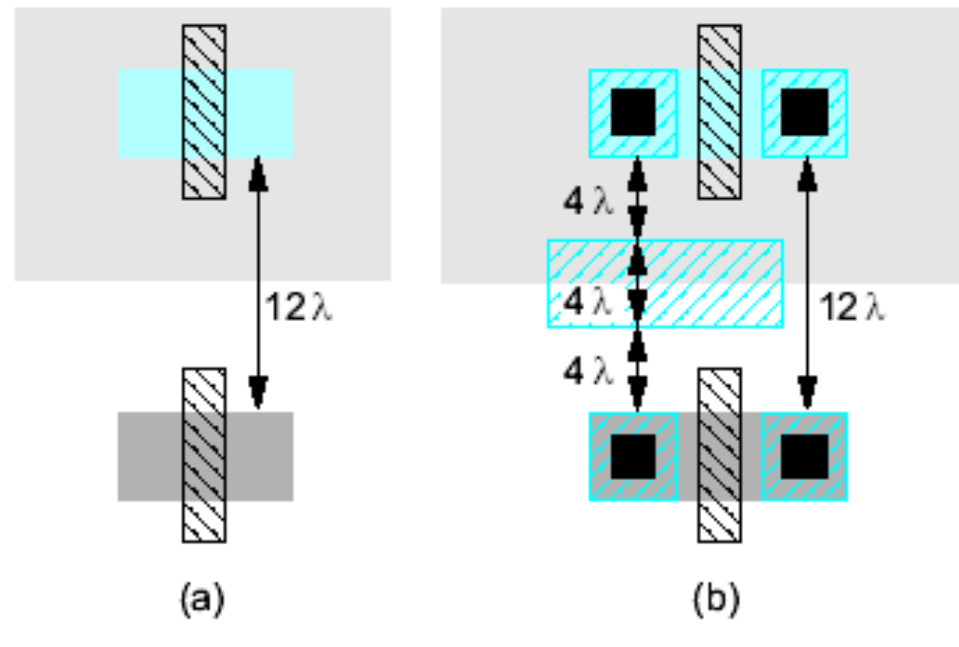
# Wiring Tracks

- A *wiring track* is the space required for a wire
  - $4 \lambda$  width,  $4 \lambda$  spacing from neighbor =  $8 \lambda$  pitch



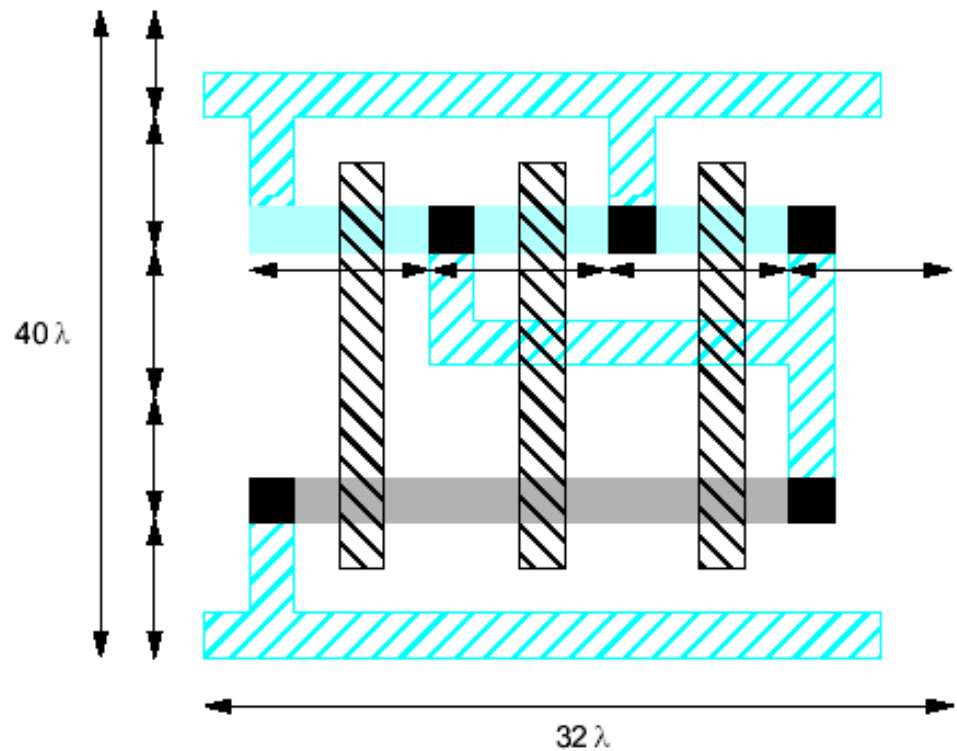
# Well spacing

- Wells must surround transistors by  $6\lambda$ 
  - Implies  $12\lambda$  between opposite transistor flavors
  - Leaves room for one wire track



# Area Estimation

- Estimate area by counting wiring tracks
  - Multiply by 8 to express in  $\lambda$



# Example: O3AI

---

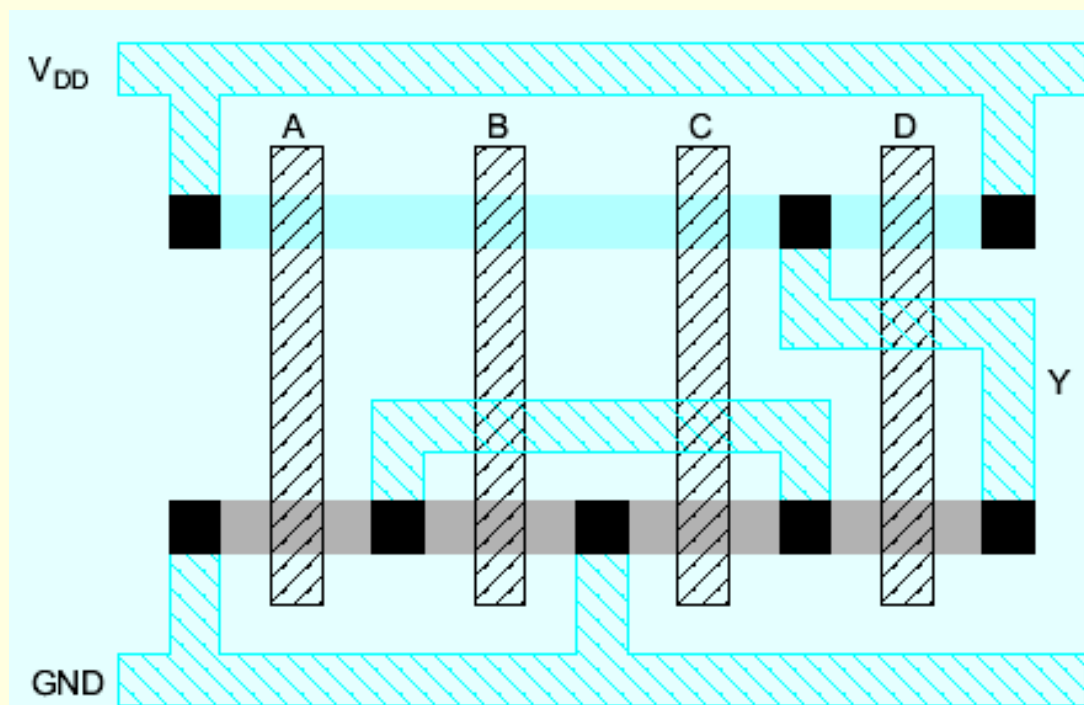
- Sketch a stick diagram for O3AI and estimate area

- $$Y = \overline{(A + B + C)} \bullet D$$

# Example: O3AI

- Sketch a stick diagram for O3AI and estimate area

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



# Example: O3AI

- Sketch a stick diagram for O3AI and estimate area

- $Y = \overline{(A + B + C)} \bullet D$

