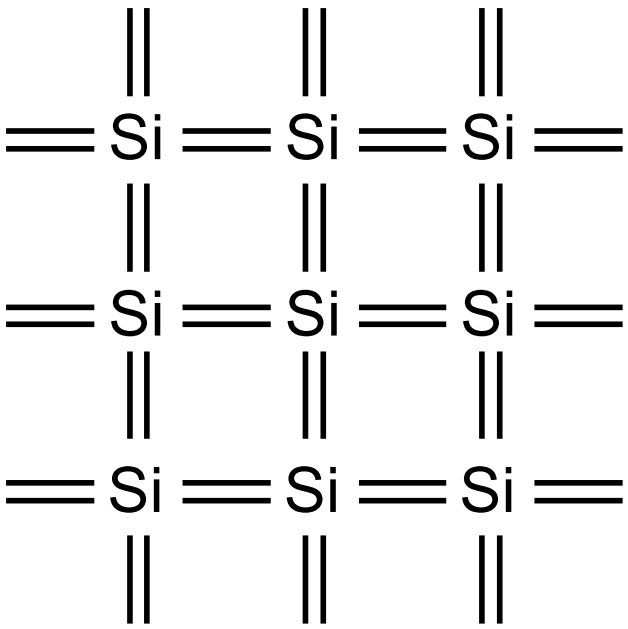
ECE1388 VLSI Design Methodology

Lecture 1: Review of CMOS Devices and Circuits

Adapted from Weste, Harris CMOS VLSI Design

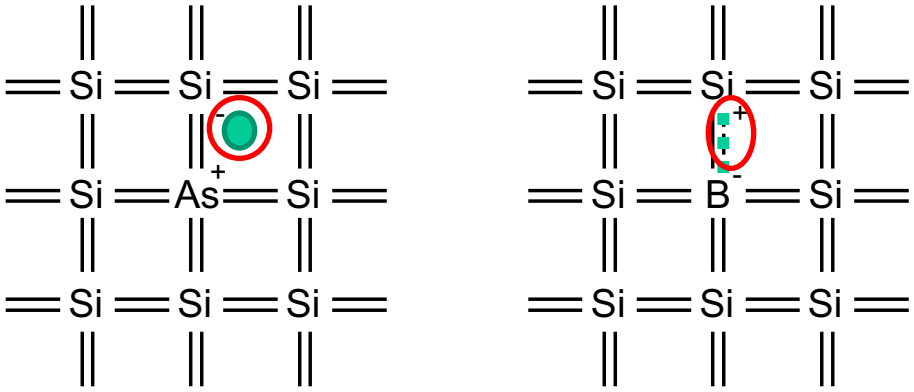
Silicon Lattice

- Transistors are built on a silicon substrate
- Silicon is a Group IV material
- Forms crystal lattice with bonds to four neighbors



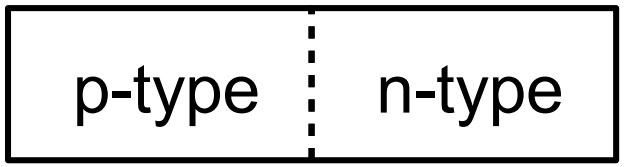
Dopants

- Silicon is a semiconductor
- Pure silicon has no free carriers and conducts poorly
- Adding dopants increases the conductivity
- Group V: extra electron (n-type, eg Arsenic)
- Group III: missing electron, called hole (p-type, eg Boron)

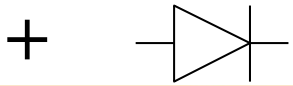


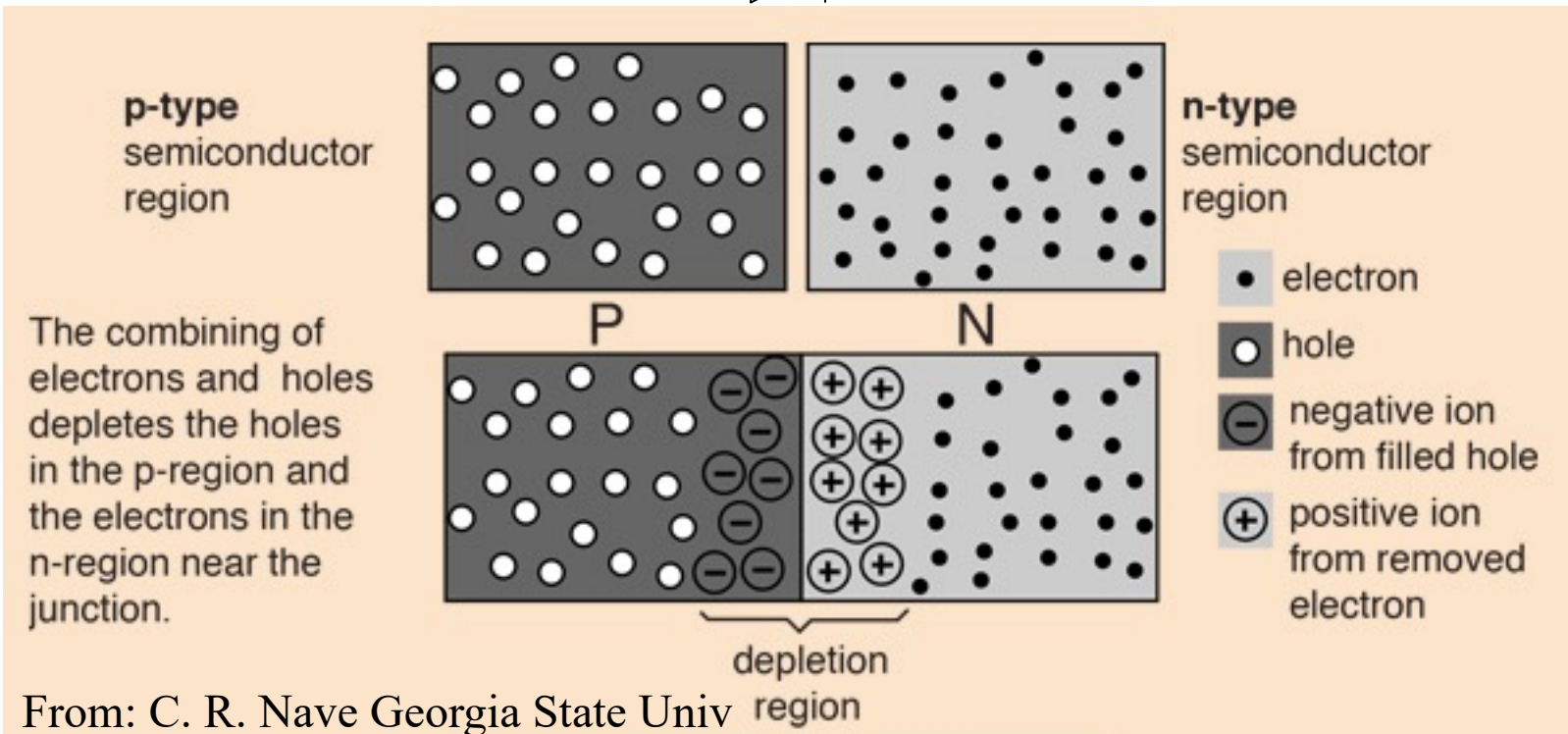
p-n Junctions

- A junction between p-type and n-type semiconductor forms a diode.
- Current flows only in one direction



anode cathode

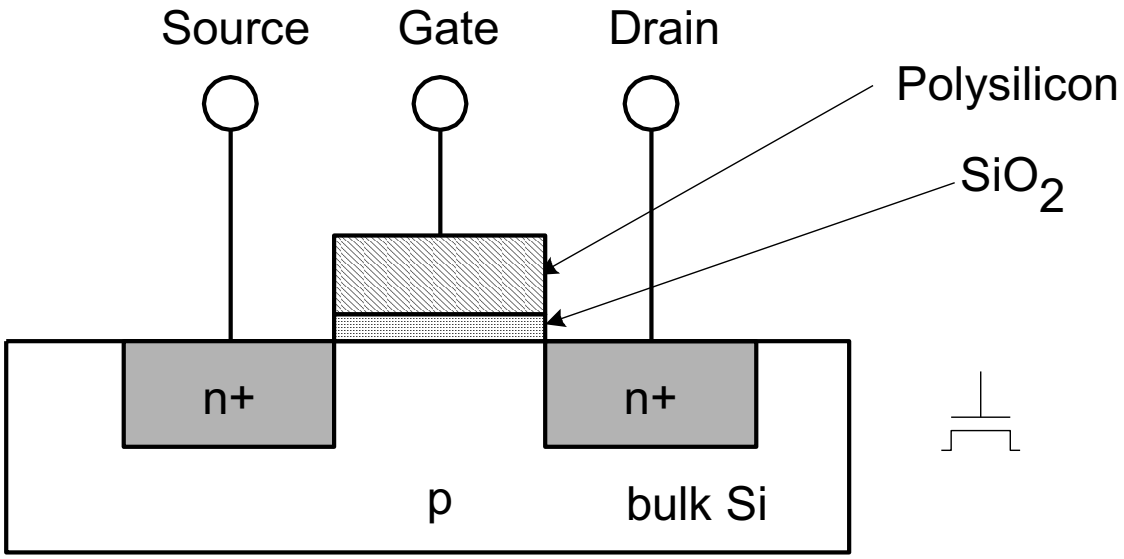




From: C. R. Nave Georgia State Univ

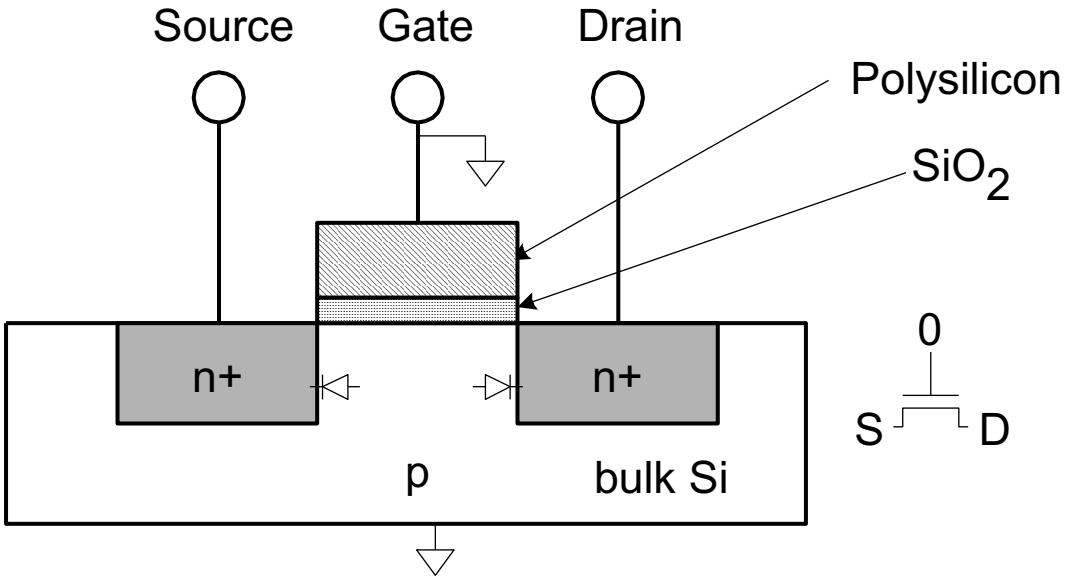
nMOS Transistor

- Four terminals: gate, source, drain, body (aka bulk)
- Gate – oxide – body stack looks like a **capacitor**
  - Gate and body are conductors
  - SiO_2 (oxide) is a very good insulator
  - Called metal – oxide – semiconductor (MOS) capacitor
  - Even though gate is no longer made of metal



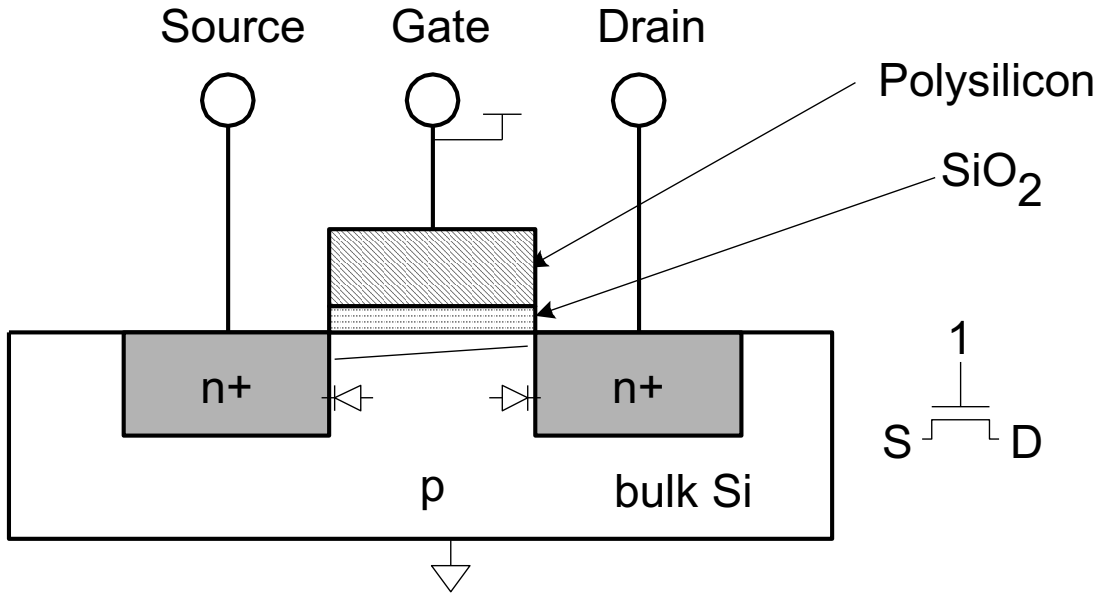
nMOS Operation

- Body is commonly tied to ground (0 V)
- When the gate is at a low voltage:
  - P-type body is at low voltage, there is no ‘channel’
  - Source-body and drain-body diodes are OFF
  - No current flows, transistor is OFF



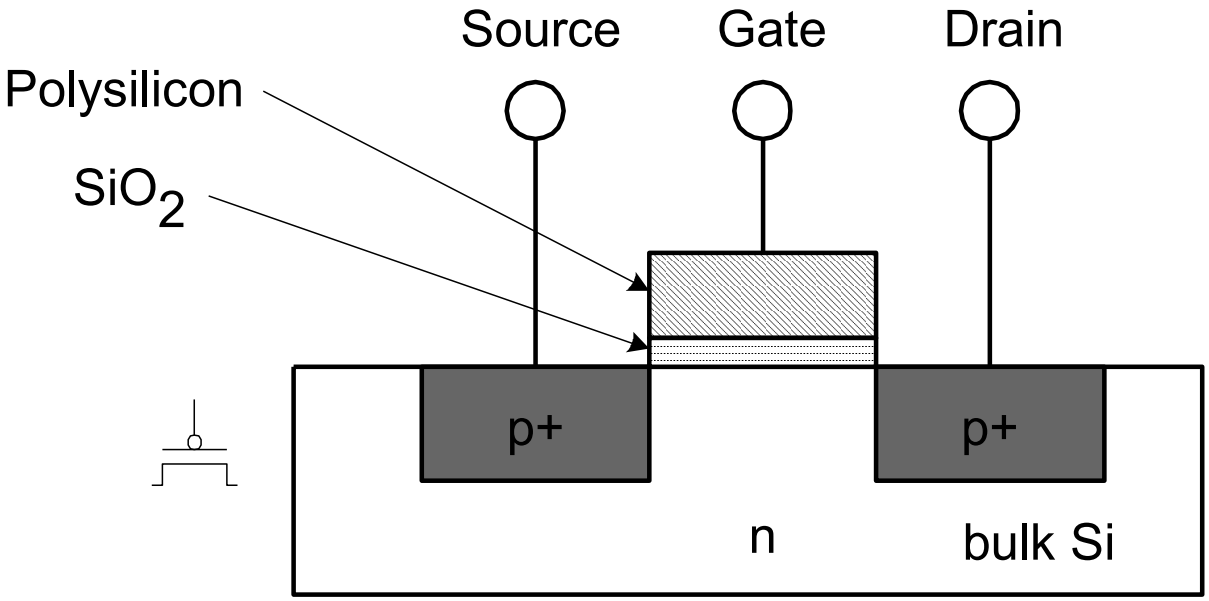
nMOS Operation Cont.

- When the gate is at a high voltage:
  - Positive charge on gate of MOS capacitor
  - Negative charge attracted to body
  - Inverts a channel under gate to n-type
  - Now current can flow through n-type silicon from source through channel to drain, transistor is ON

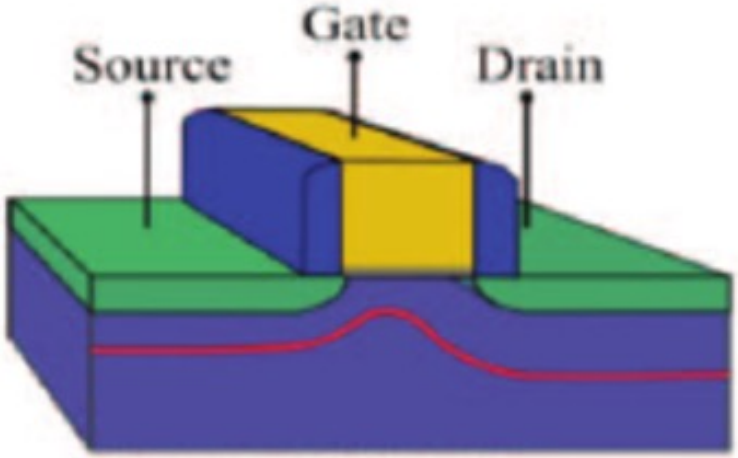


pMOS Transistor

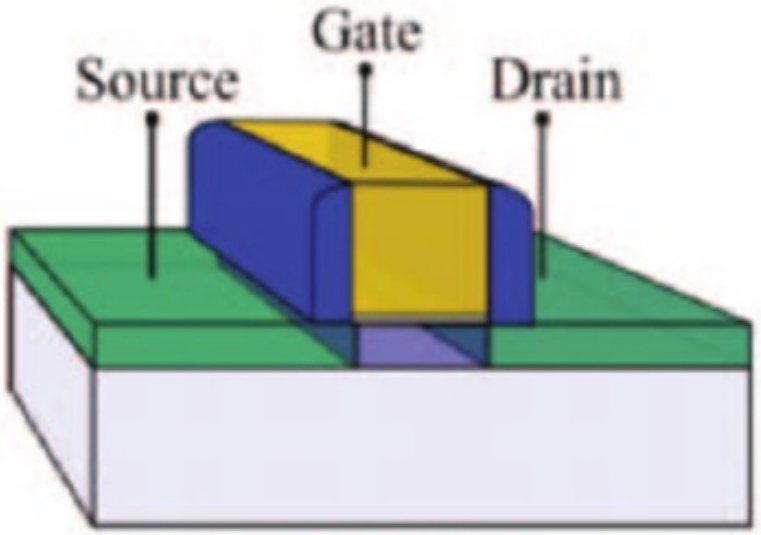
- Similar, but doping and voltages reversed
  - Body tied to high voltage (V_{DD})
  - Gate low: transistor ON
  - Gate high: transistor OFF
  - Bubble indicates inverted behavior



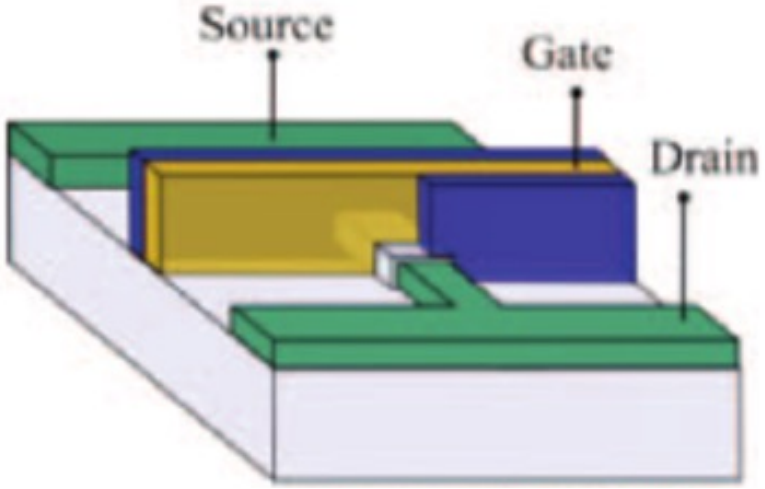
Modern Transistors



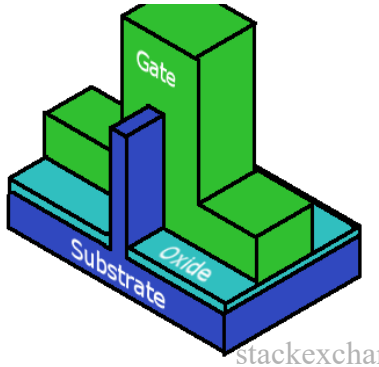
Bulk MOSFET (planar MOSFET)



UTB-SOI MOSFET (ultra-thin body silicon-on-insulator MOSFET)



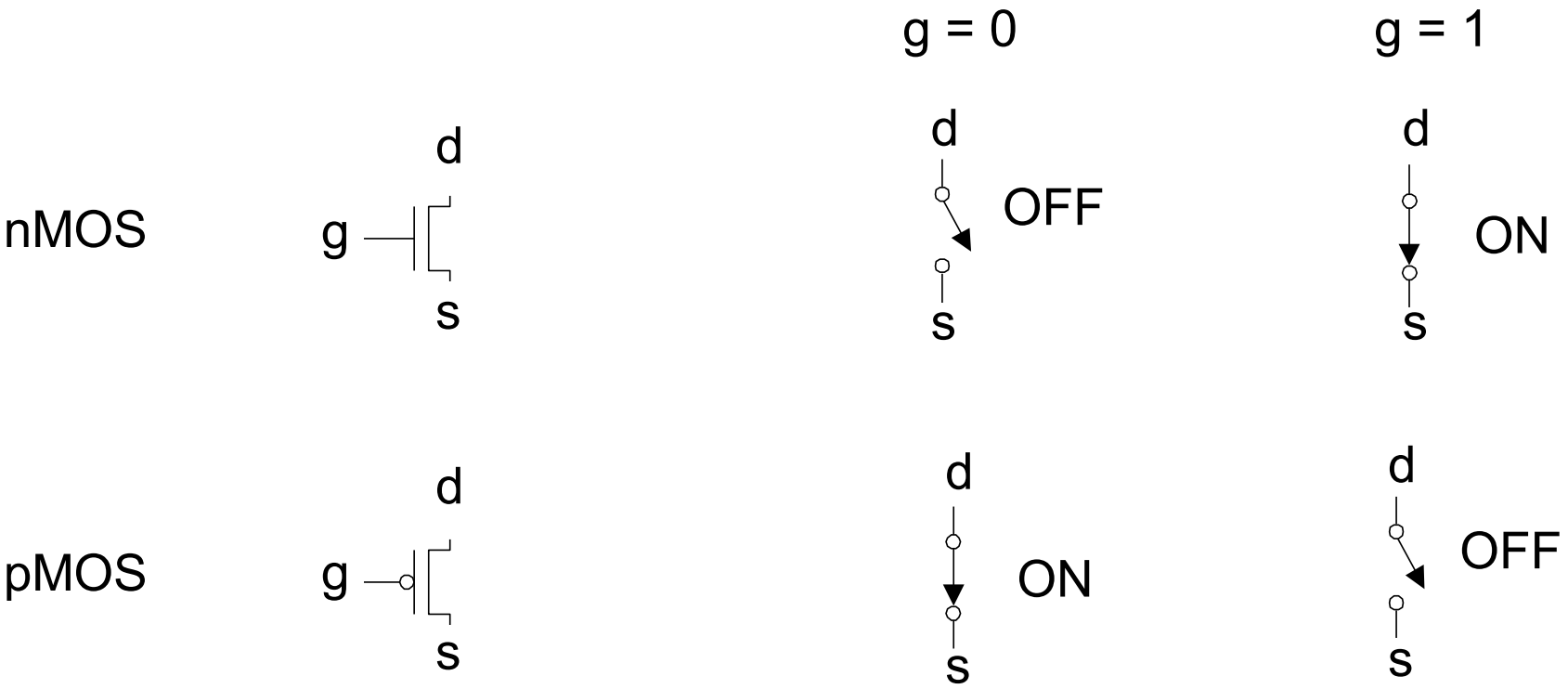
FinFet (Tri-gate MOSFET) on SOI



FinFet (Tri-gate MOSFET) on bulk

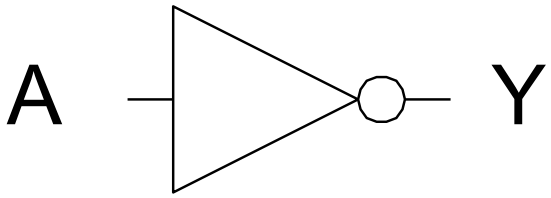
Transistors as Switches

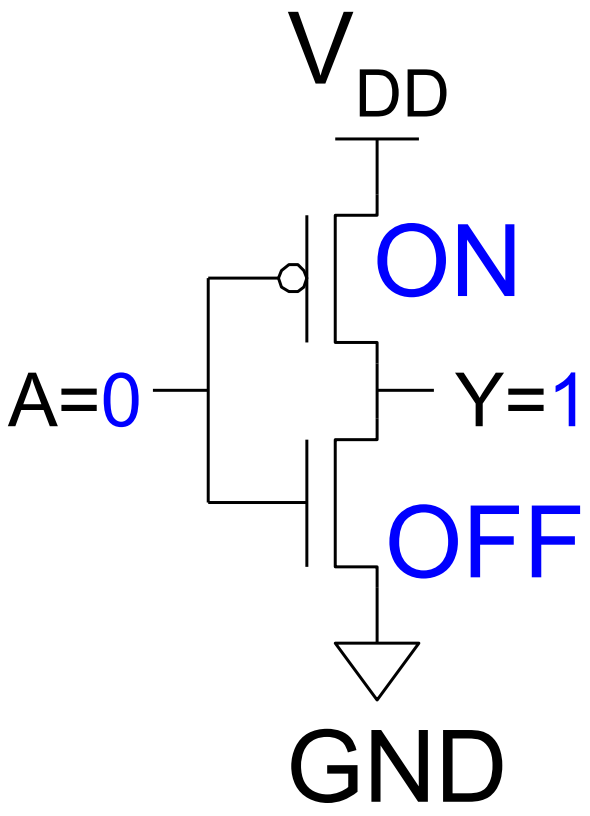
- We can view MOS transistors as electrically controlled switches
- Voltage at gate controls path from source to drain



Static CMOS Logic: Inverter

A	Y
0	**1**
1	0

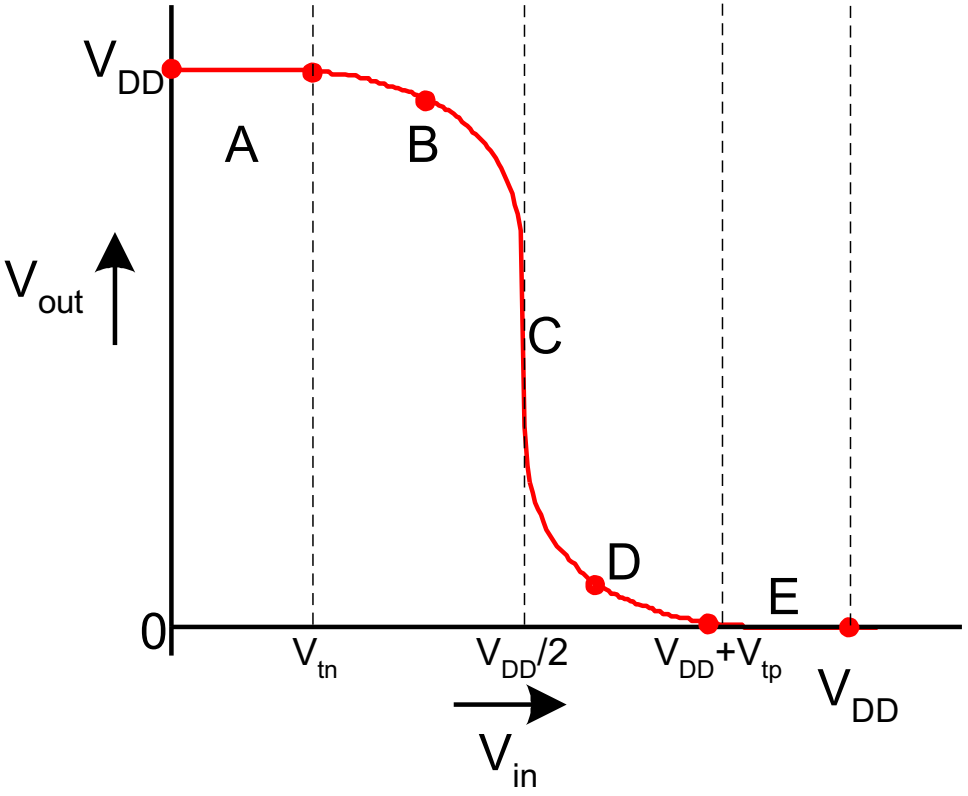




Signal Level Restoring

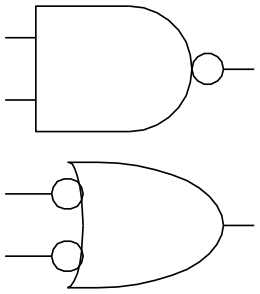
- Static CMOS logic: Inherently signal level-restoring

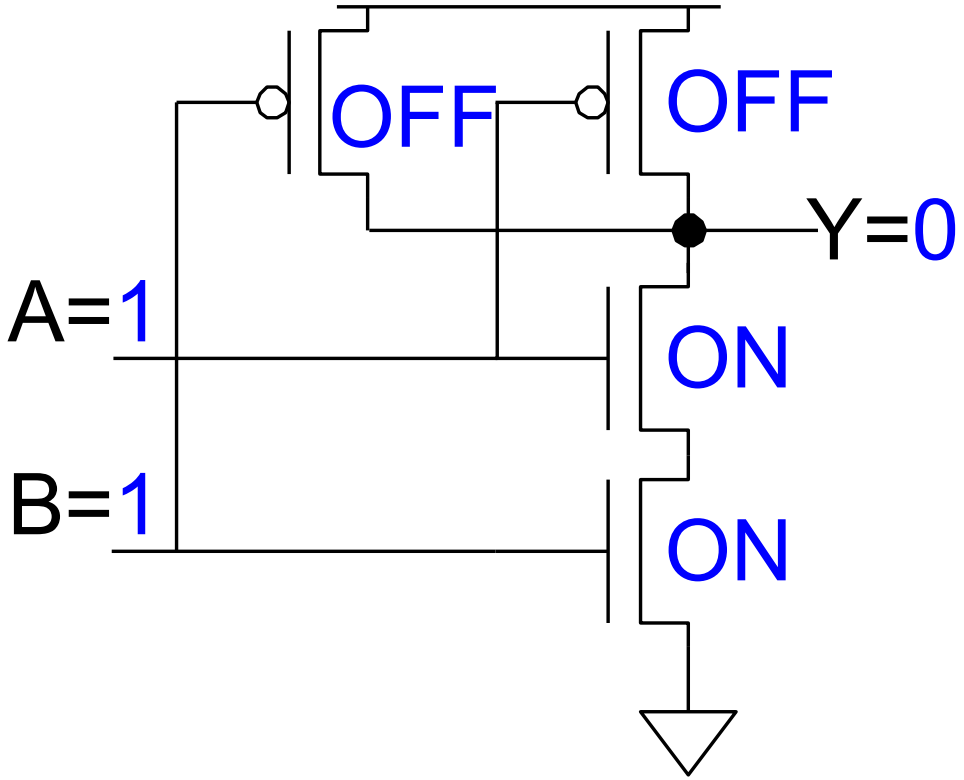
Region	nMOS	pMOS
A	Cutoff	Linear
B	Saturation	Linear
C	Saturation	Saturation
D	Linear	Saturation
E	Linear	Cutoff



CMOS NAND Gate

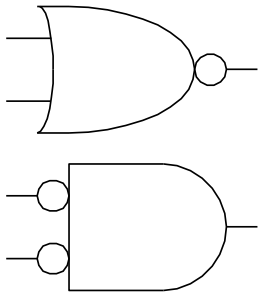
A	B	Y
0	0	1
0	1	1
1	0	1
1	**1**	**0**

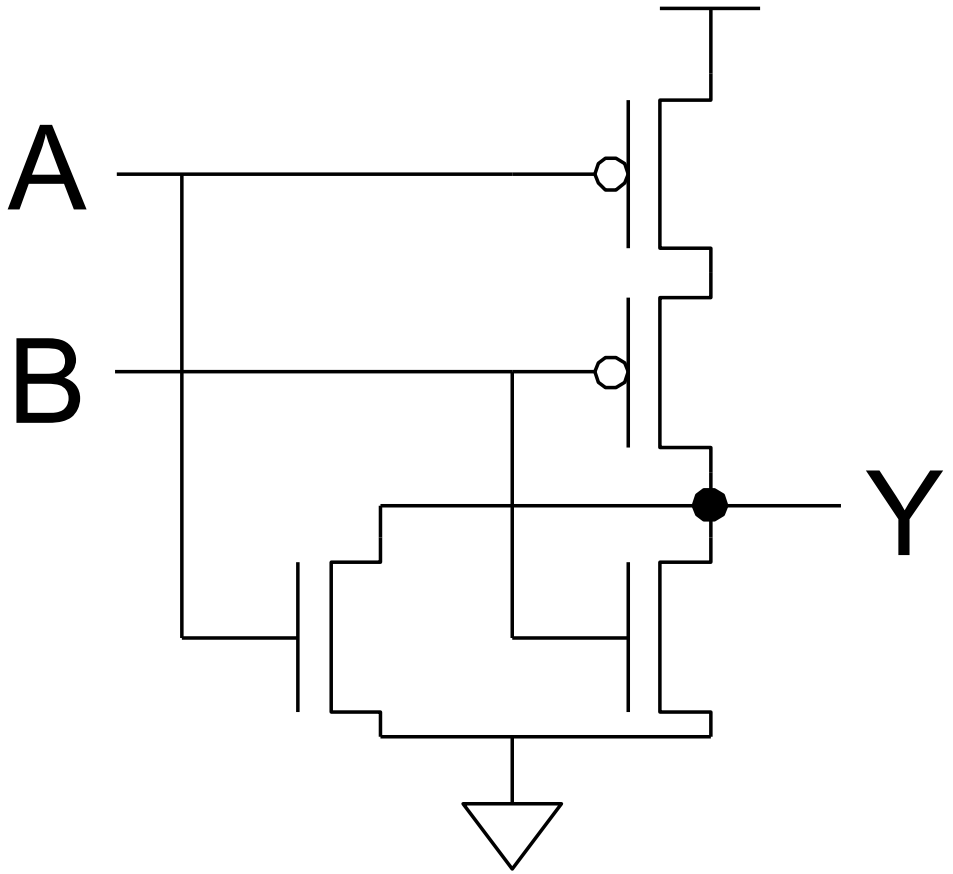




CMOS NOR Gate

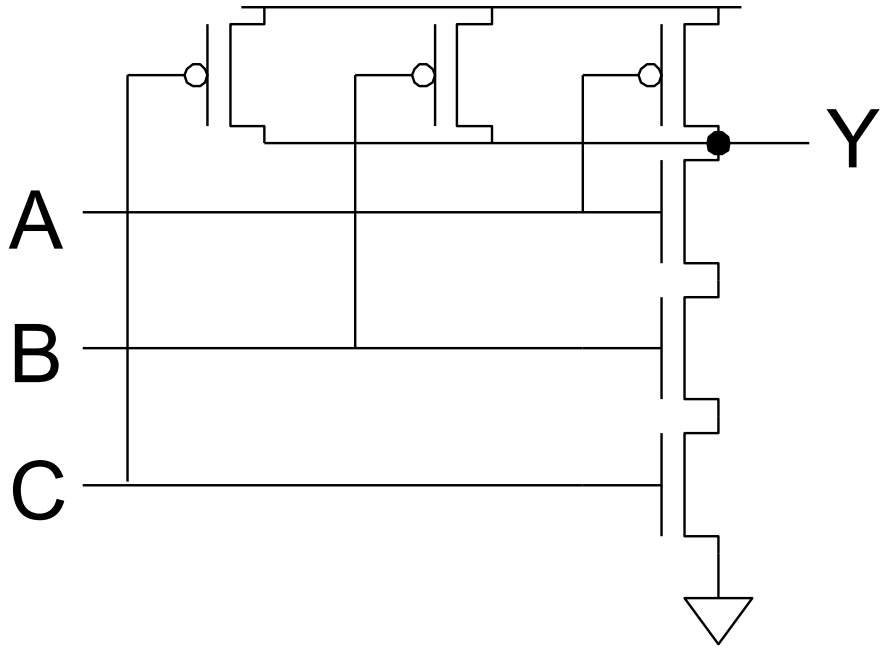
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0





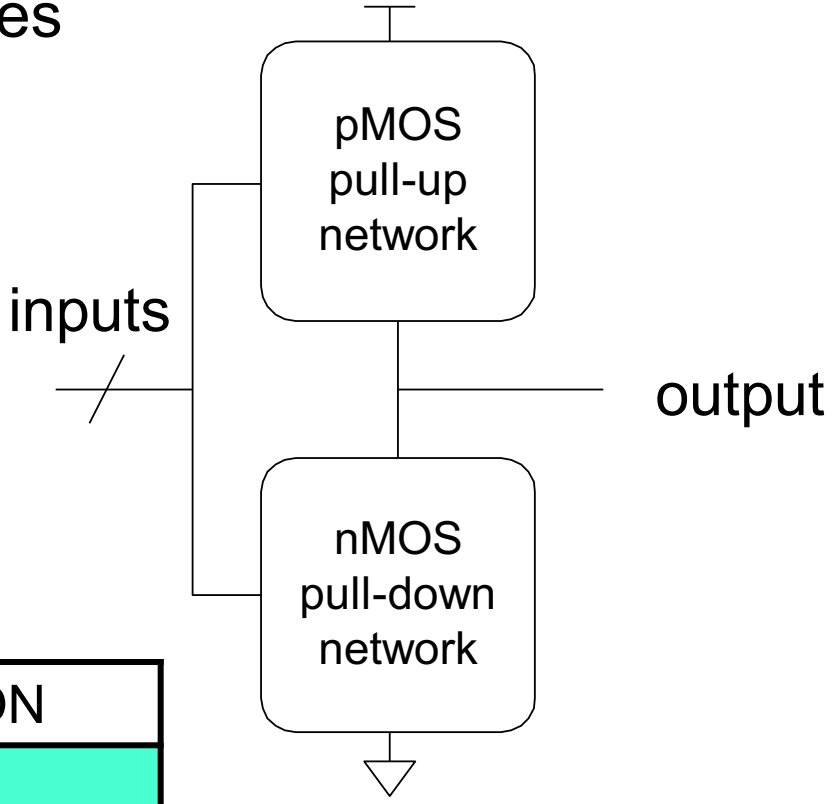
3-input NAND Gate

- Y pulls low if ALL inputs are 1
- Y pulls high if ANY input is 0



Complementary CMOS

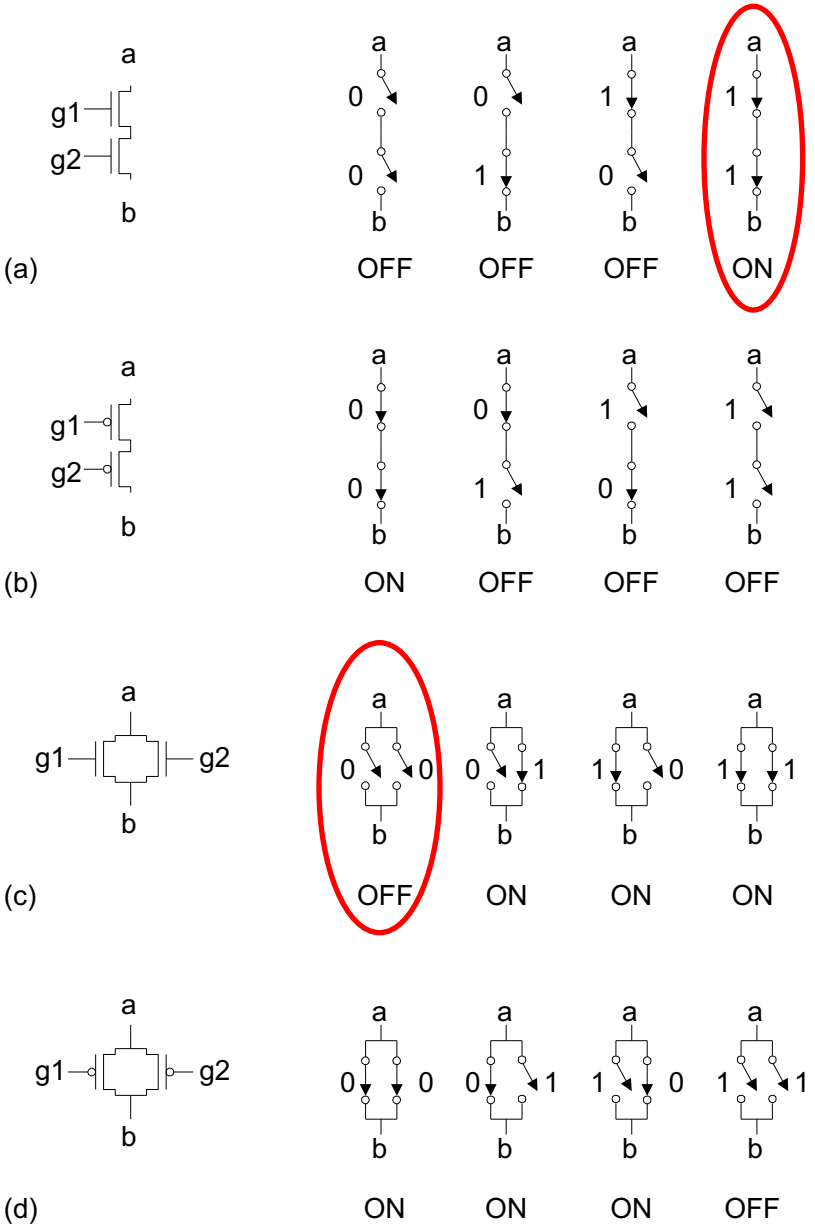
- Complementary CMOS logic gates
  - nMOS *pull-down network*
  - pMOS *pull-up network*
  - a.k.a. static CMOS



	Pull-up OFF	Pull-up ON
Pull-down OFF	Z (floating – in some cases)	1
Pull-down ON	0	X (crowbar - never)

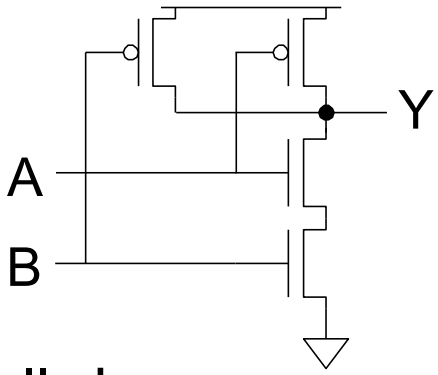
Series and Parallel

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



Conduction Complement

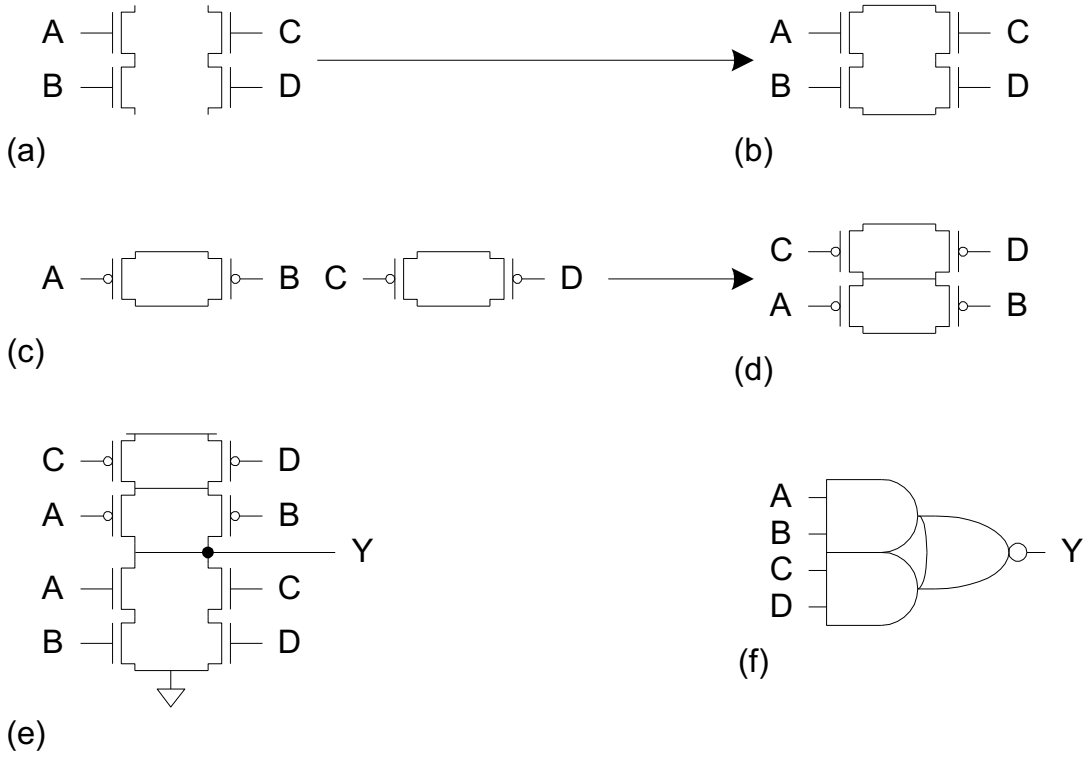
- Complementary CMOS gates always produce 0 or 1
- Ex: NAND gate
  - Series nMOS: Y=0 when both inputs are 1
  - Thus Y=1 when either input is 0
  - Requires parallel pMOS



- Rule of *Conduction Complements*
  - Pull-up network is complement of pull-down
  - Parallel -> series, series -> parallel

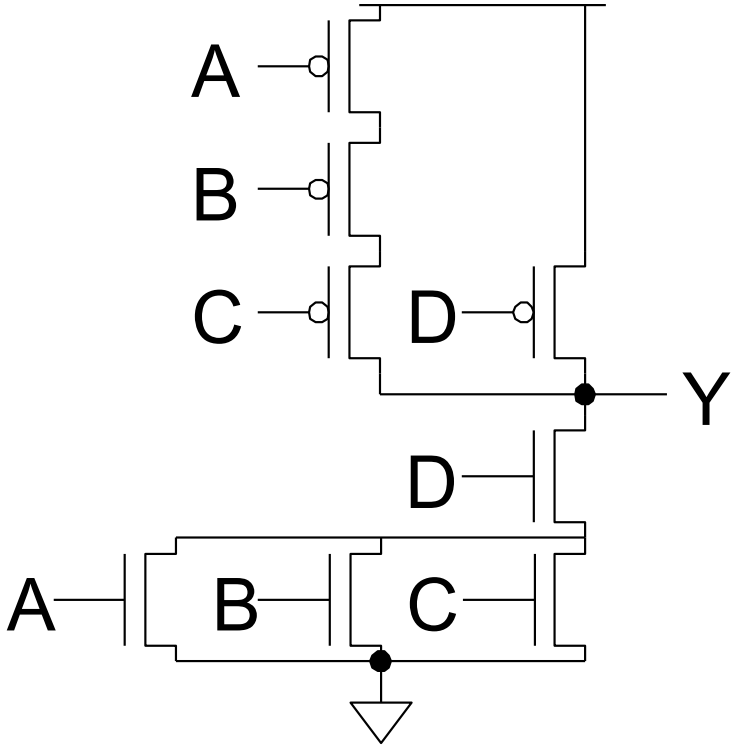
Compound Gates

- *Compound gates* can do any inverting function
- Ex: $Y = A \bullet B + C \bullet D$ (AND-AND-OR-INVERT, AOI22)



Example: O3AI

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

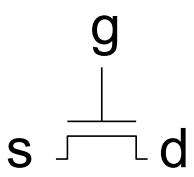


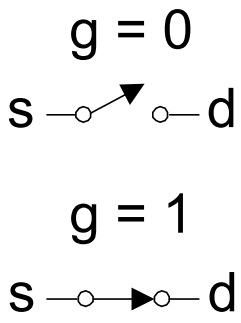
Signal Strength

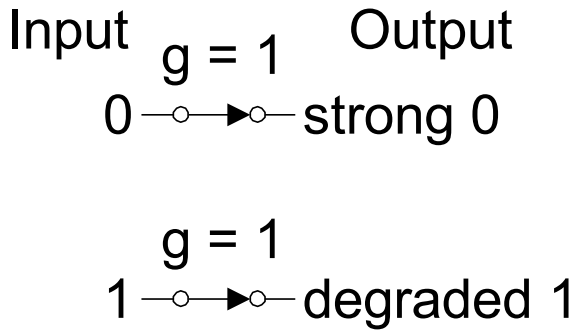
- *Strength* of signal
  - How close it approximates ideal voltage source
- V_{DD} and GND rails are strongest 1 and 0
- nMOS pass strong 0
  - But degraded or weak 1
- pMOS pass strong 1
  - But degraded or weak 0
- Thus:
  - **nMOS** are best for **pull-down** network
  - **pMOS** are best for **pull-up**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

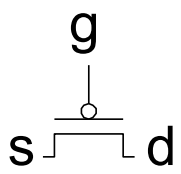
Pass Transistors

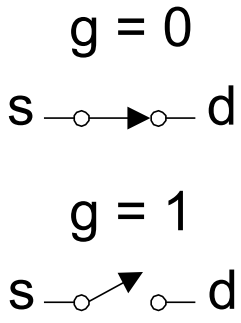
- Transistors can be used as switches

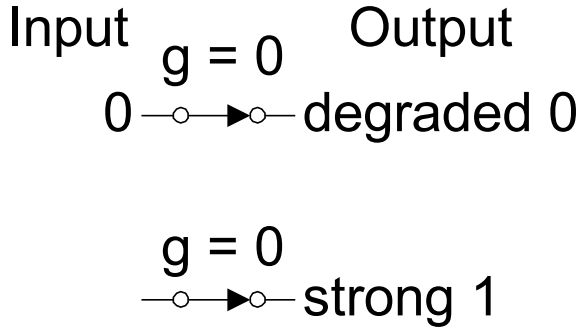






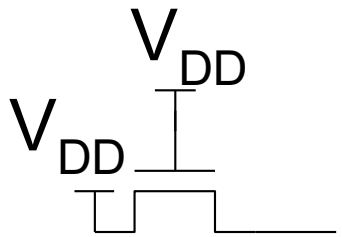




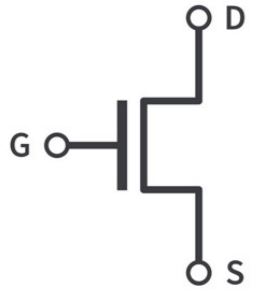


Pass Transistors: Degraded Signals

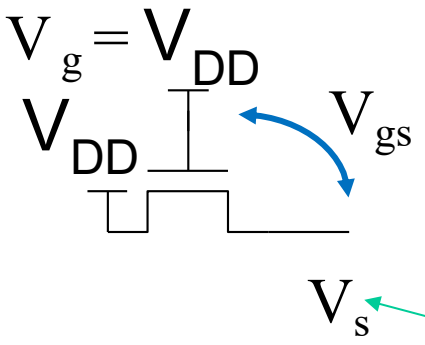
- We have assumed source is grounded
- What if source > 0?
  - e.g. pass transistor passing V_{DD}



Pass Transistors: Degraded Signals

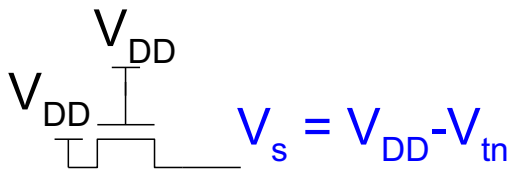


- We have assumed source is grounded
- What if source > 0?
  - e.g. pass transistor passing V_{DD}
- $V_g = V_{DD}$
  - If $V_s > V_{DD}-V_t$, $V_{gs} < V_t$
  - Hence transistor would turn itself off
- nMOS pass transistors pull no higher than $V_{DD}-V_{tn}$
  - Called a degraded "1"
  - Approach degraded value slowly (low I_{ds})
- pMOS pass transistors pull no lower than V_{tp}

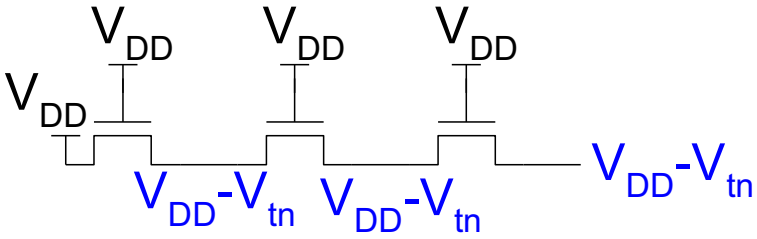


NMOS: S is at the lowest voltage

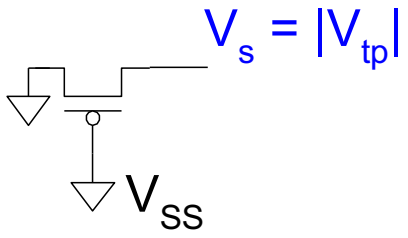
Pass Transistor Ckts



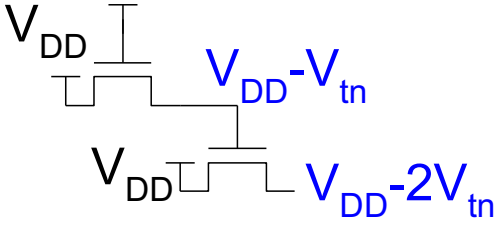
✘ Degraded "1"



✘ Very slow



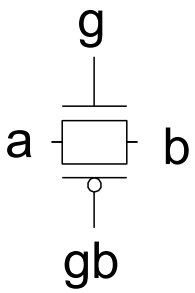
✘ Degraded "0"



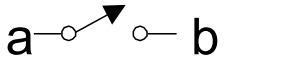
✘ Very slow

Transmission Gates

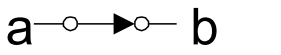
- Pass transistors produce degraded outputs
- *Transmission gates* pass both 0 and 1 well



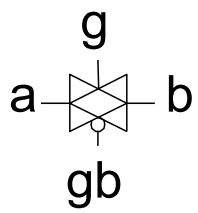
g = 0, gb = 1

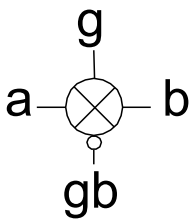


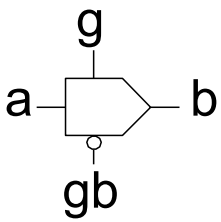
g = 1, gb = 0



Input	Output
g = 1, gb = 0 0	strong 0
g = 1, gb = 0 1	strong 1



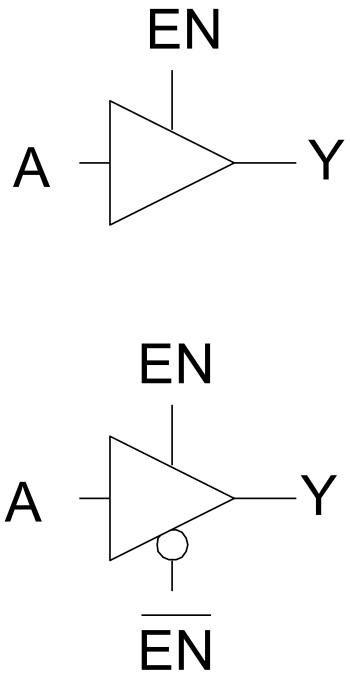




Tristates

- *Tristate buffer* produces Z when not enabled

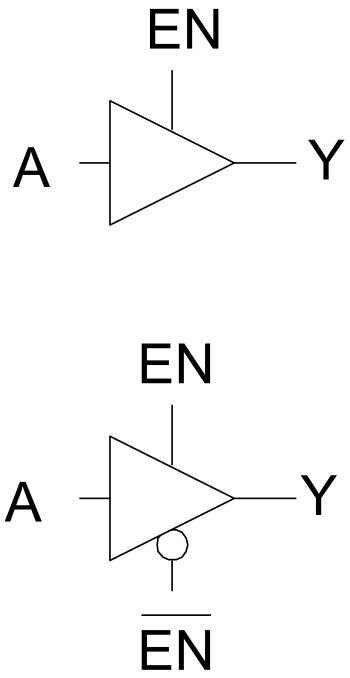
EN	A	Y
0	0	
0	1	
1	0	
1	1	



Tristates

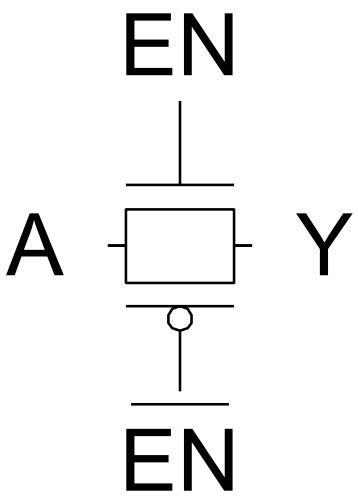
- *Tristate buffer* produces Z when not enabled

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



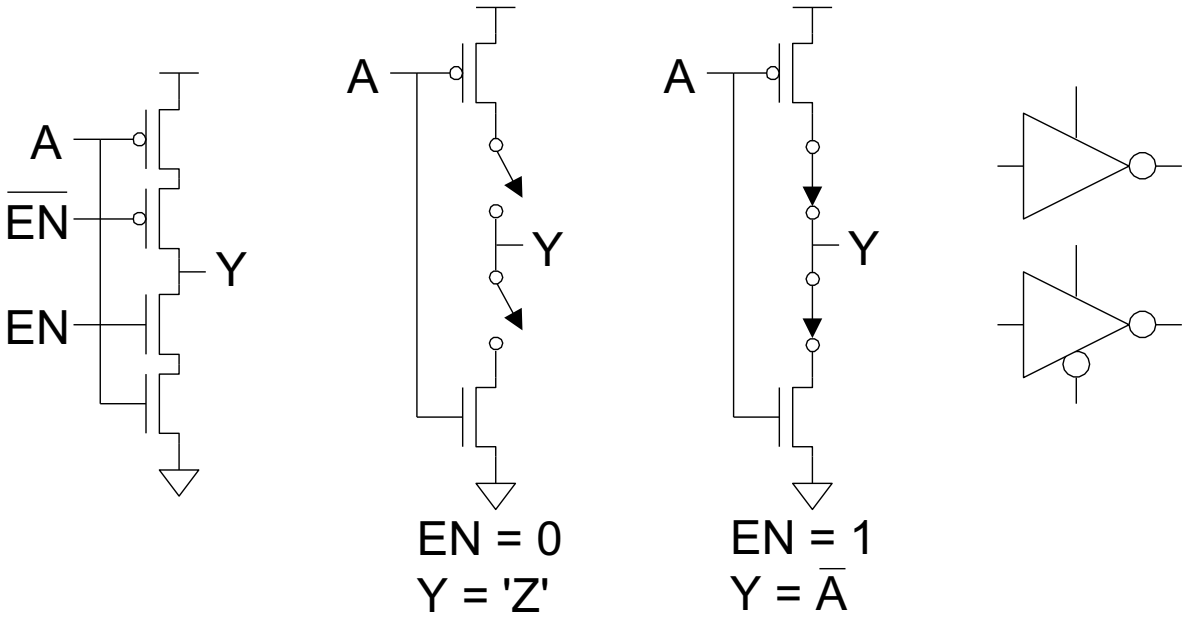
Nonrestoring Tristate

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



Restoring Tristate Inverter

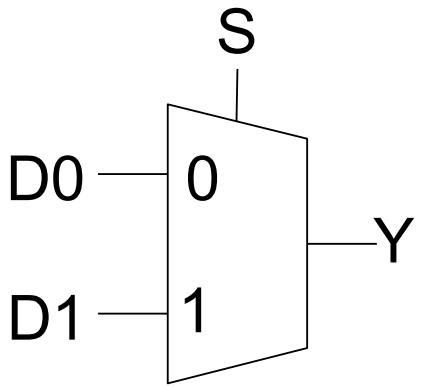
- Tristate inverter produces restored output
  - Violates conduction complement rule
  - Because we want a Z output



Multiplexers

- 2:1 *multiplexer* chooses between two inputs

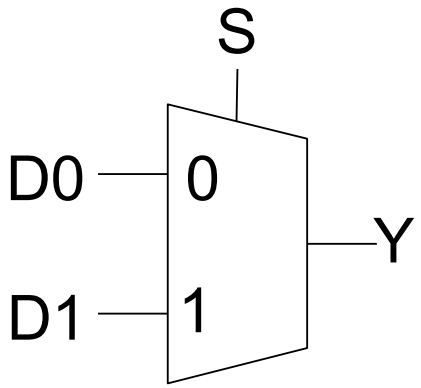
S	D1	D0	Y
0	X	0	
0	X	1	
1	0	X	
1	1	X	



Multiplexers

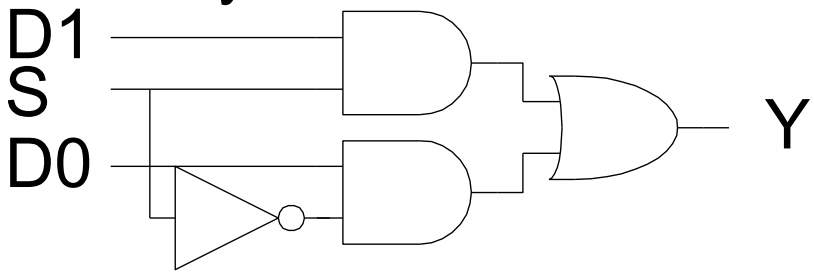
- 2:1 multiplexer chooses between two inputs

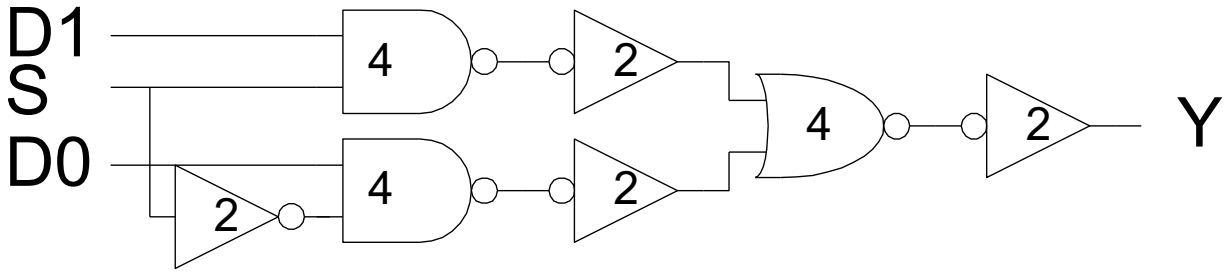
S	D1	D0	Y
0	X	0	0
0	X	1	1
1	0	X	0
1	1	X	1



Gate-Level Mux Design

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

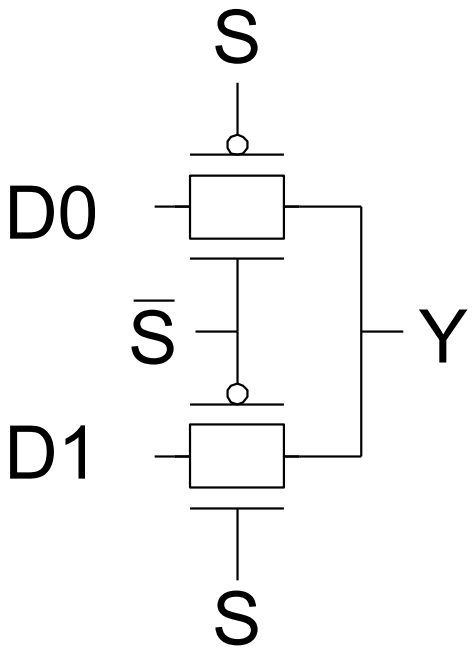




- Can be further optimized to 14 transistors (the last three stages can be replaced by a NAND gate)

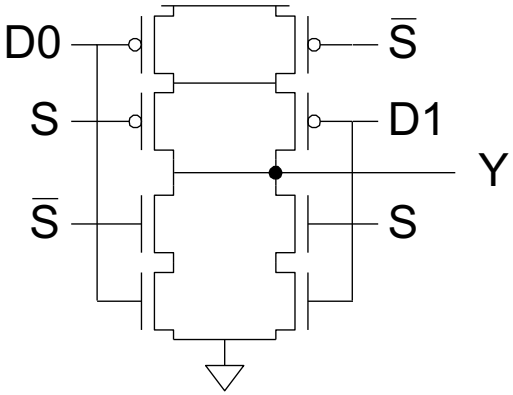
Transmission Gate Mux

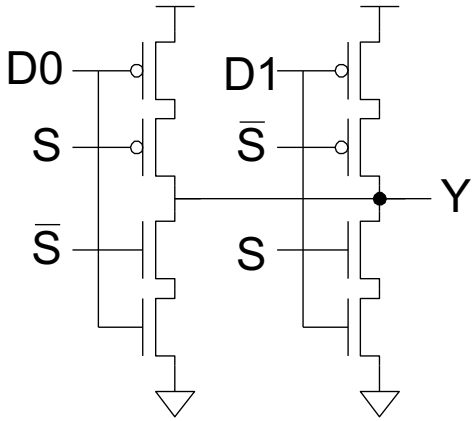
- Nonrestoring mux uses two transmission gates
  - Only 4 transistors

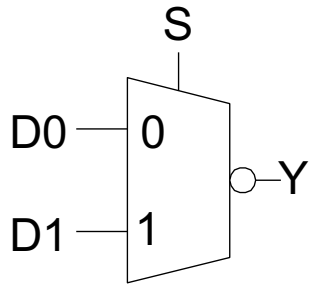


Restoring Inverting Mux

- Inverting multiplexer
  - Use compound gate (by inverting $Y = SD_1 + \overline{S}D_0$)
  - Or a pair of tristate inverters
  - Essentially the same thing
- Noninverting multiplexer adds an inverter

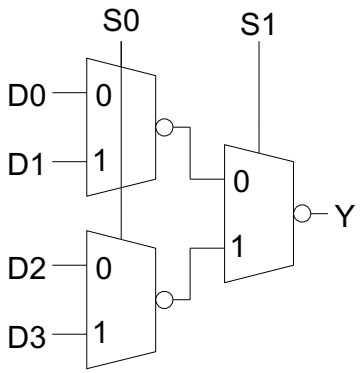


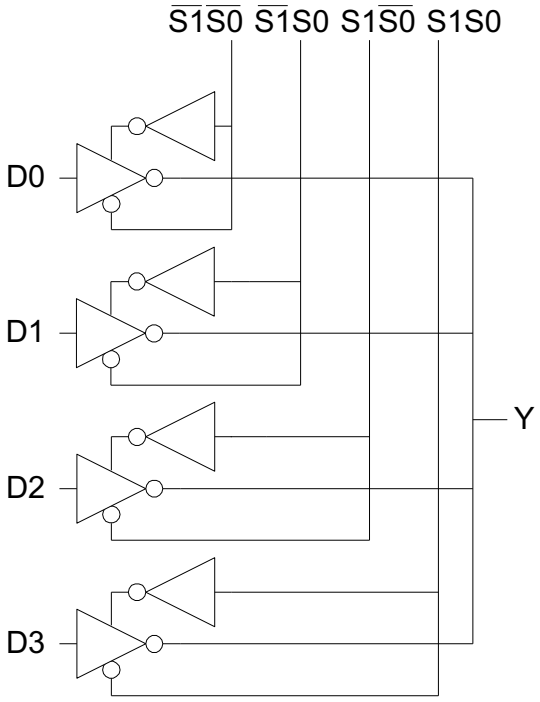




4:1 Multiplexer

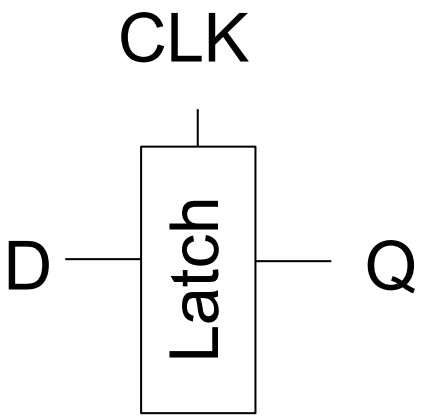
- 4:1 mux chooses one of 4 inputs using two selects
  - Two levels of 2:1 muxes
  - Or four tristates

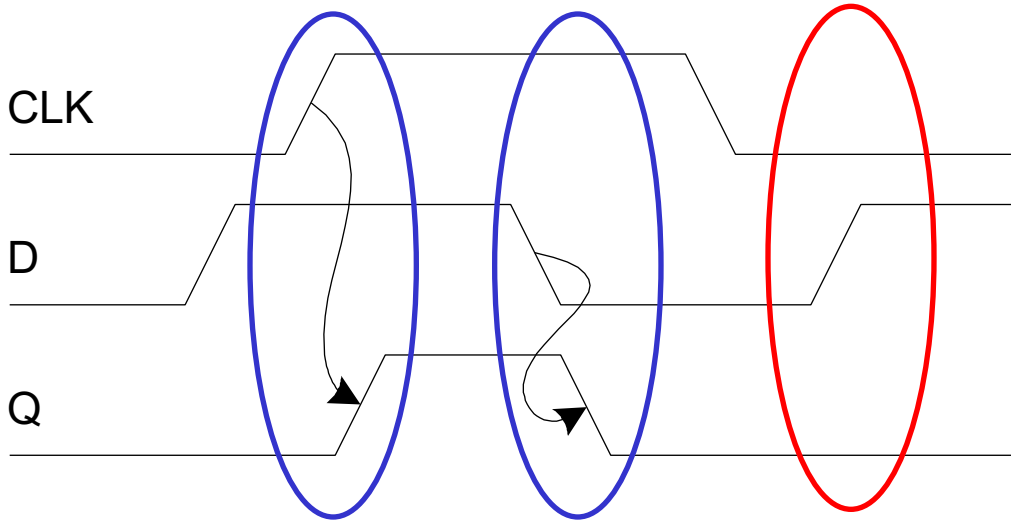




D Latch

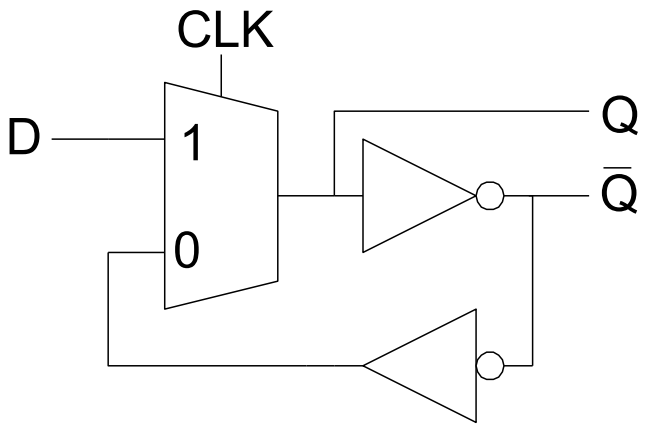
- When CLK = 1, latch is *transparent*
  - D flows through to Q like a buffer
- When CLK = 0, the latch is *opaque*
  - Q holds its old value independent of D
- a.k.a. *transparent latch* or *level-sensitive latch*

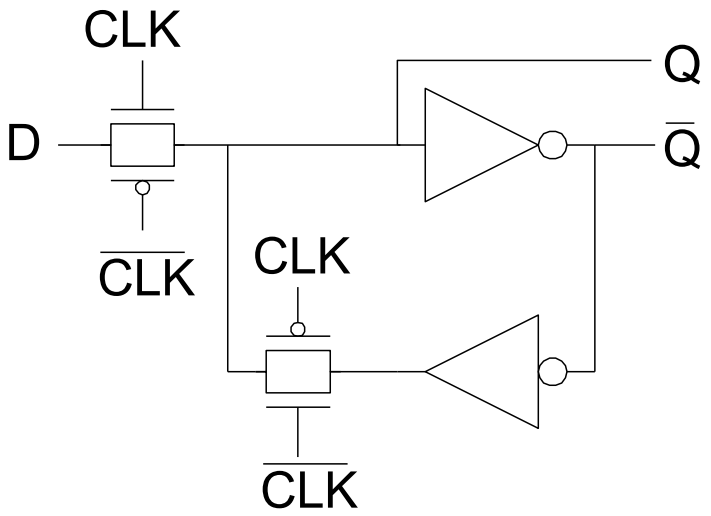




D Latch Design

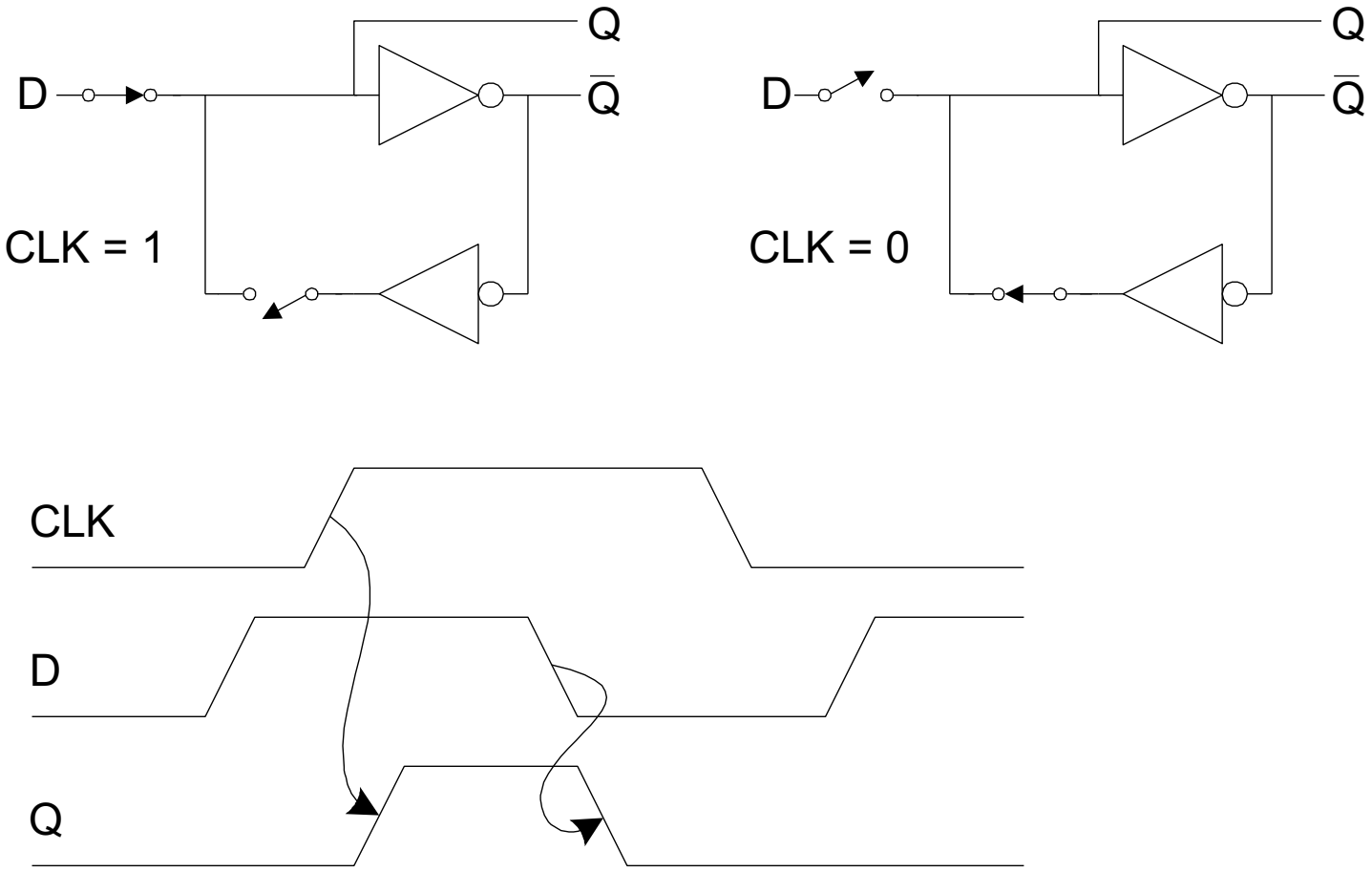
- Multiplexer chooses D or old Q





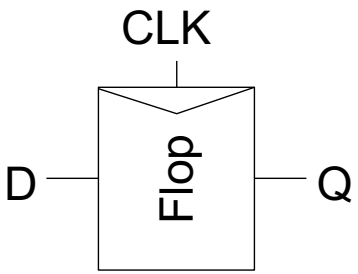
- Design hints – to avoid **common beginner's mistakes:**
  - always buffer latch outputs (e.g., in a shift register)
  - always account for (i.e., simulate) the parasitics (e.g., C and R) of wires coming to/from a latch (or a shift register)

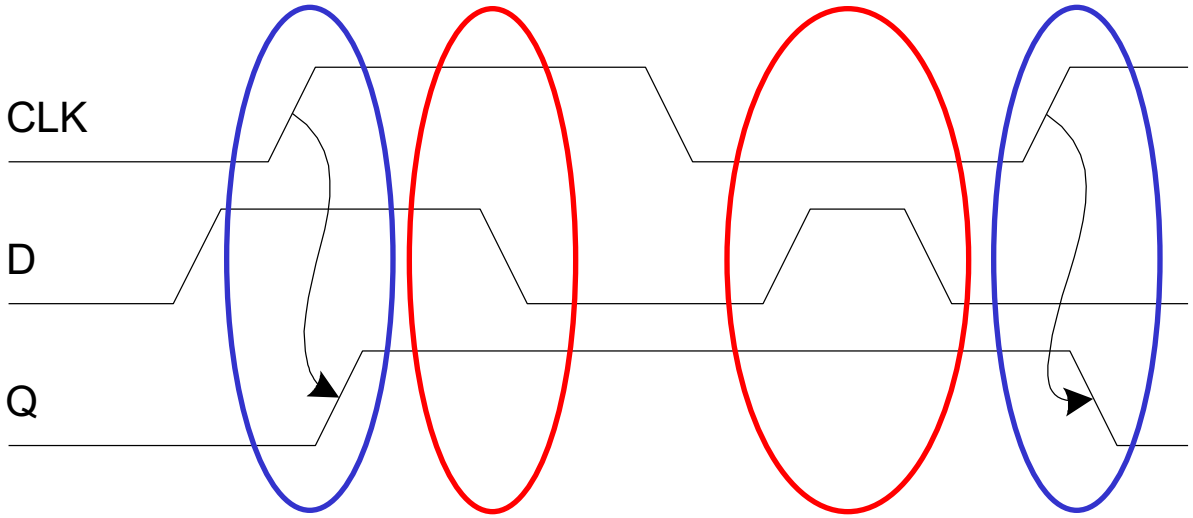
D Latch Operation



D Flip-flop

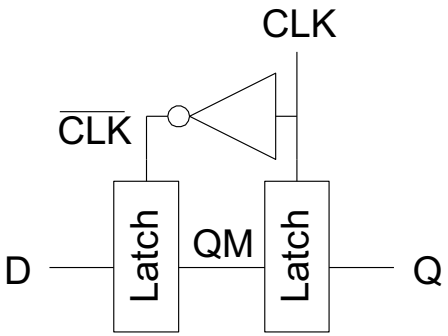
- When CLK rises, D is copied to Q
- At all other times, Q holds its value
- a.k.a. *positive edge-triggered flip-flop*, *master-slave flip-flop*

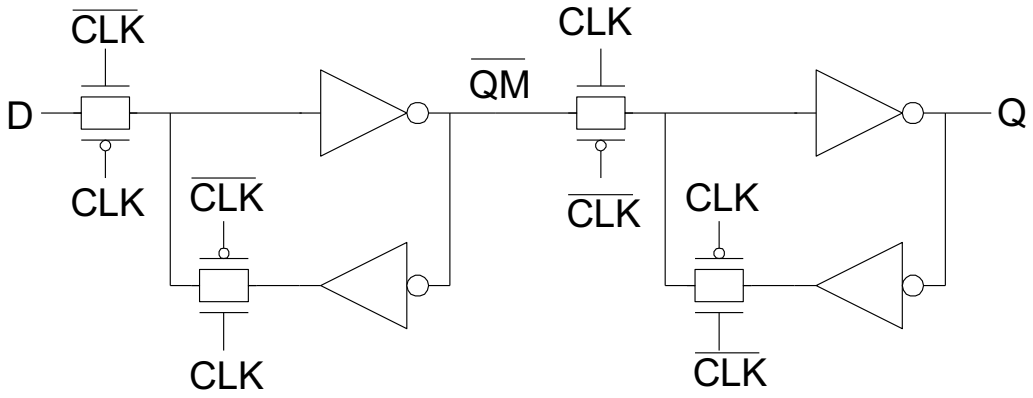




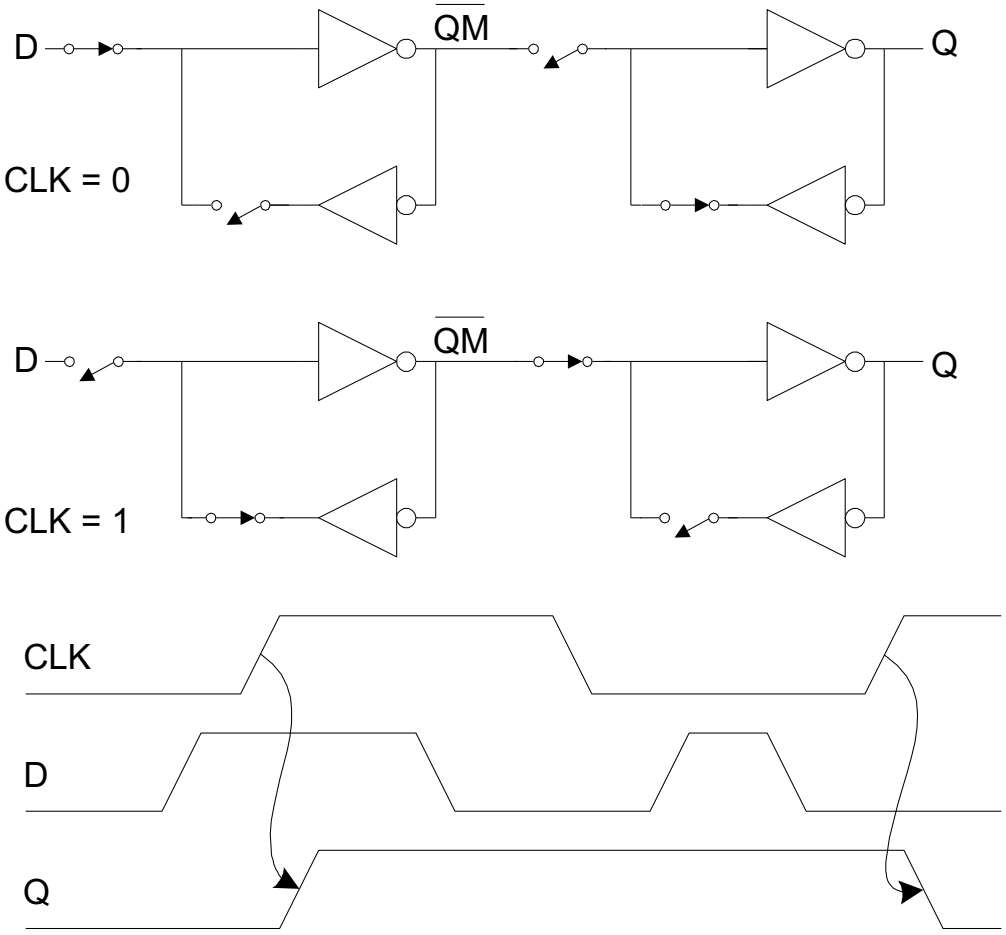
D Flip-flop Design

- Built from D latches (master and slave)



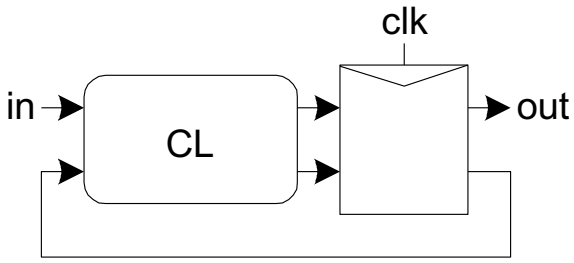


D Flip-flop Operation

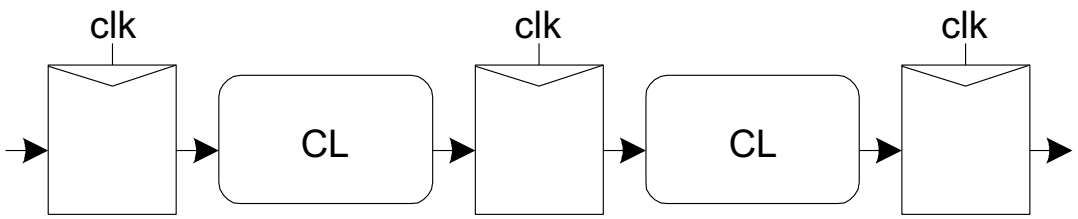


Sequencing

- *Combinational logic*
  - output depends on current inputs
- *Sequential logic*
  - uses flip-flops or latches
  - output depends on current and previous inputs
  - requires separating previous, current, future
  - called *state* or *tokens*
  - Ex: FSM, pipeline

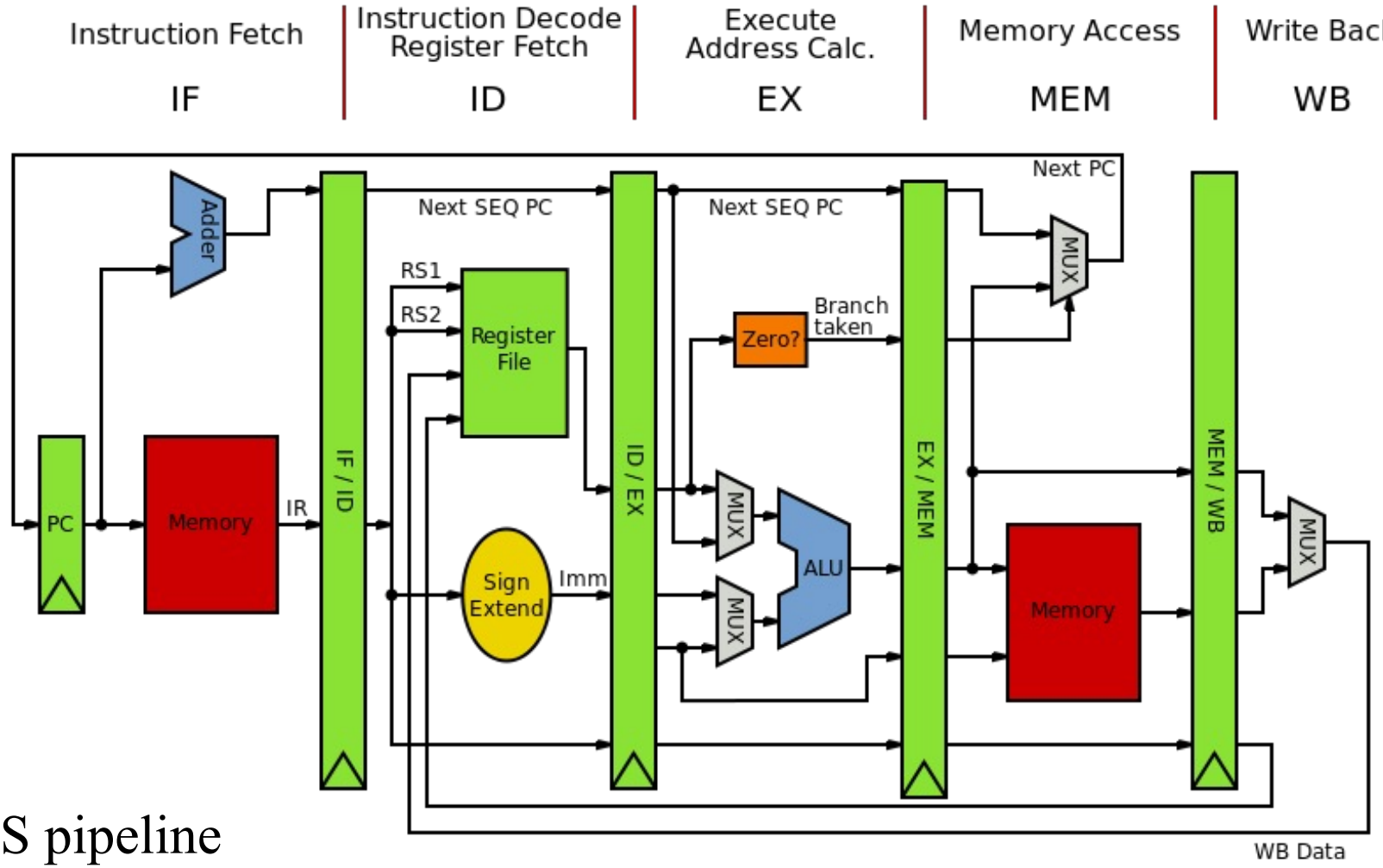


Finite State Machine



Pipeline

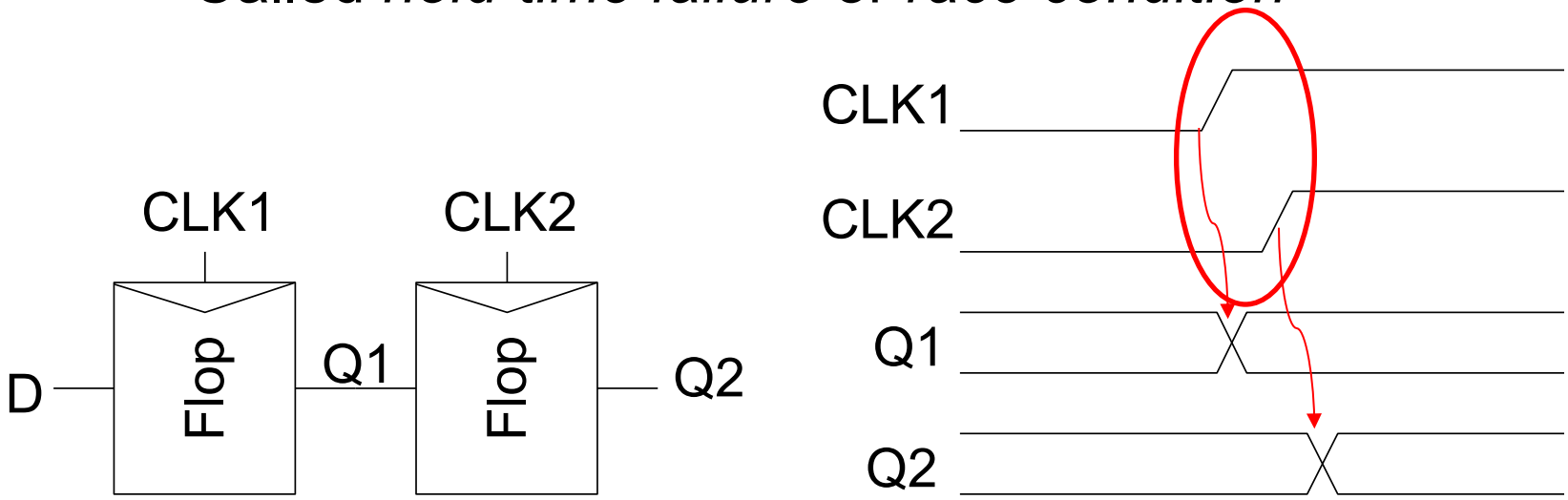
Example: μProcessor Pipeline



MIPS pipeline example (Wikipedia)

Race Condition

- 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*



✘ Token passes through two stages in one clock cycle

Nonoverlapping Clocks

- Nonoverlapping clocks can prevent races
  - As long as nonoverlap exceeds clock skew
  - Allows for safer but slower design