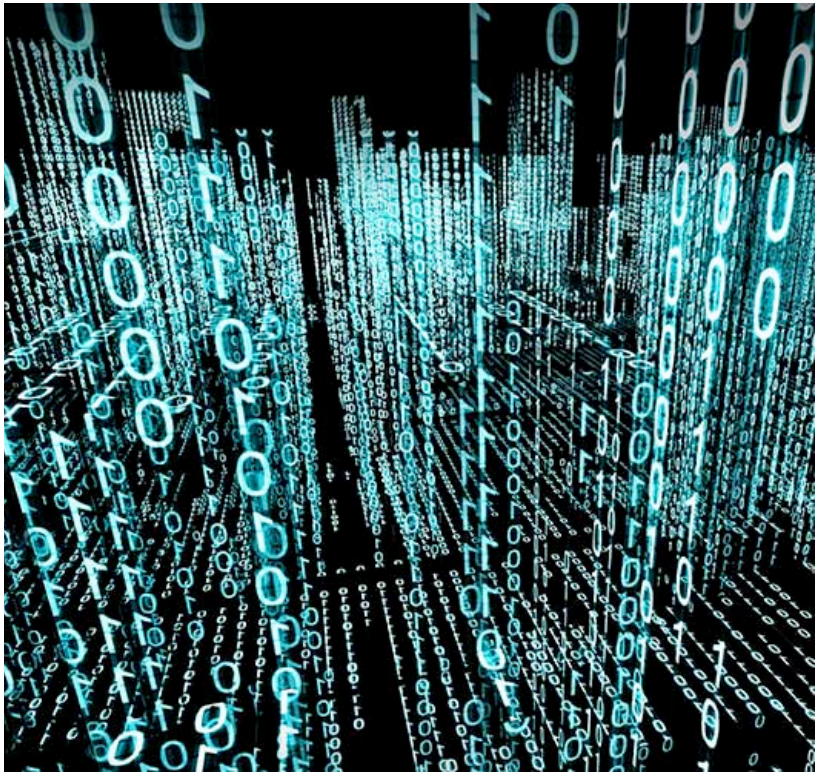


EECS 151/251A

Spring 2024

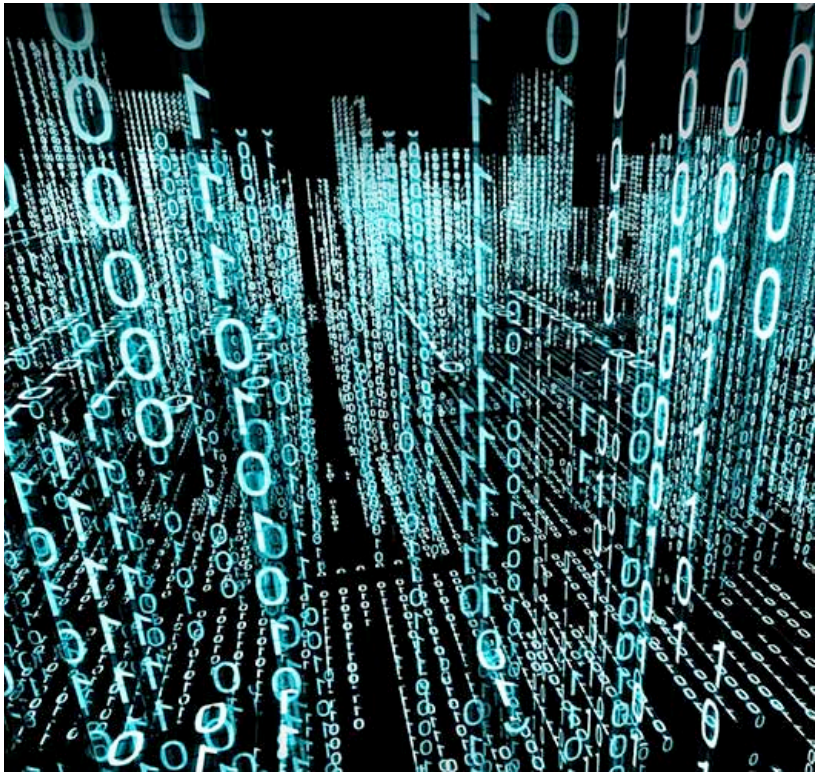
Digital Design and Integrated Circuits

Instructors:
Wawrzynek

Lecture 10: CMOS2

Announcements

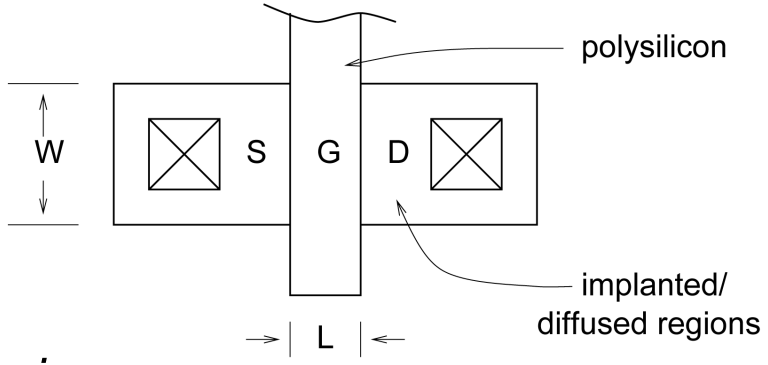
- Monday is an Academic Holiday
- New Midterm Date/Time:
  - **Tue Mar 12 2024 7:00-10:00PM**
  - **MOFF101, VLSB2040**
- Lecture Schedule change:
  - Monday Mar 11, in-class MT review



CMOS Transistors

Transistor Strength and Symmetry

1. Transistor "strength" proportional to W/L. In digital circuits, L is almost always minimal allowed by process.

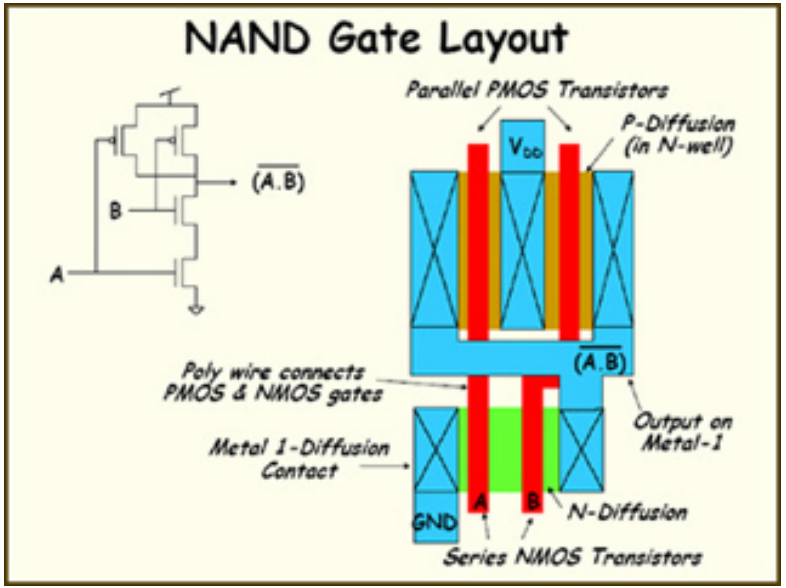


2. MOS transistors are symmetrical devices (Source and drain are interchangeable). But usually designed to be used in one direction.

For nFET, source is the node w/ the lowest voltage. For pFET source is node with highest voltage.

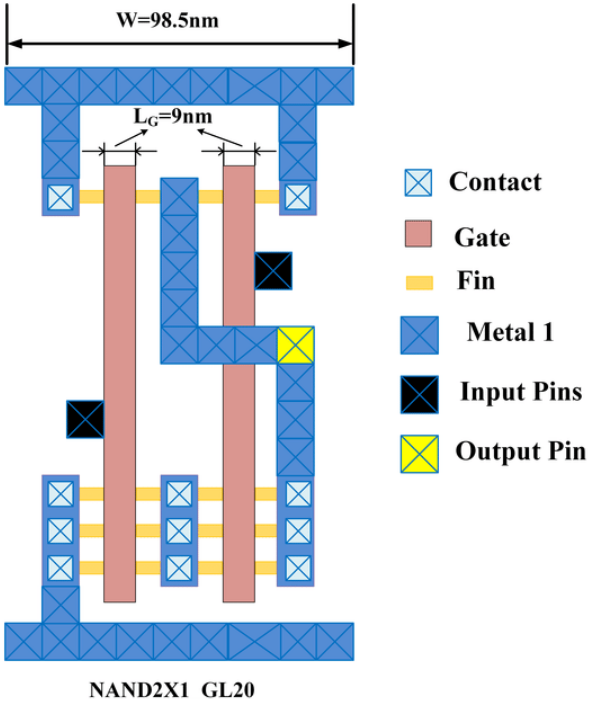
Circuit Layout Examples

- 2-input NAND



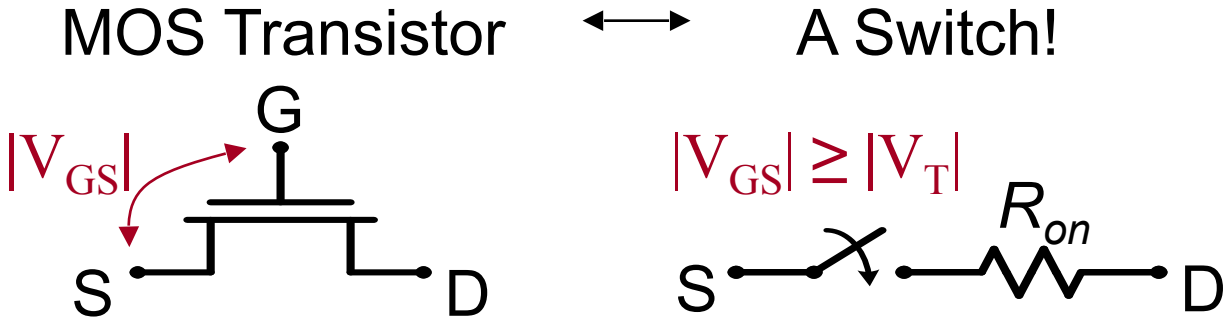
NAND gate layout from Lecture 3: CMOS Technology and Logic Gates. (Image by Professors Arvind and Asanovic.)

Finfet layout



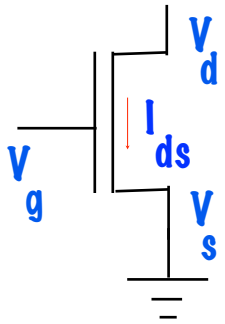
from Ji Li

MOS Transistor as a Resistive Switch

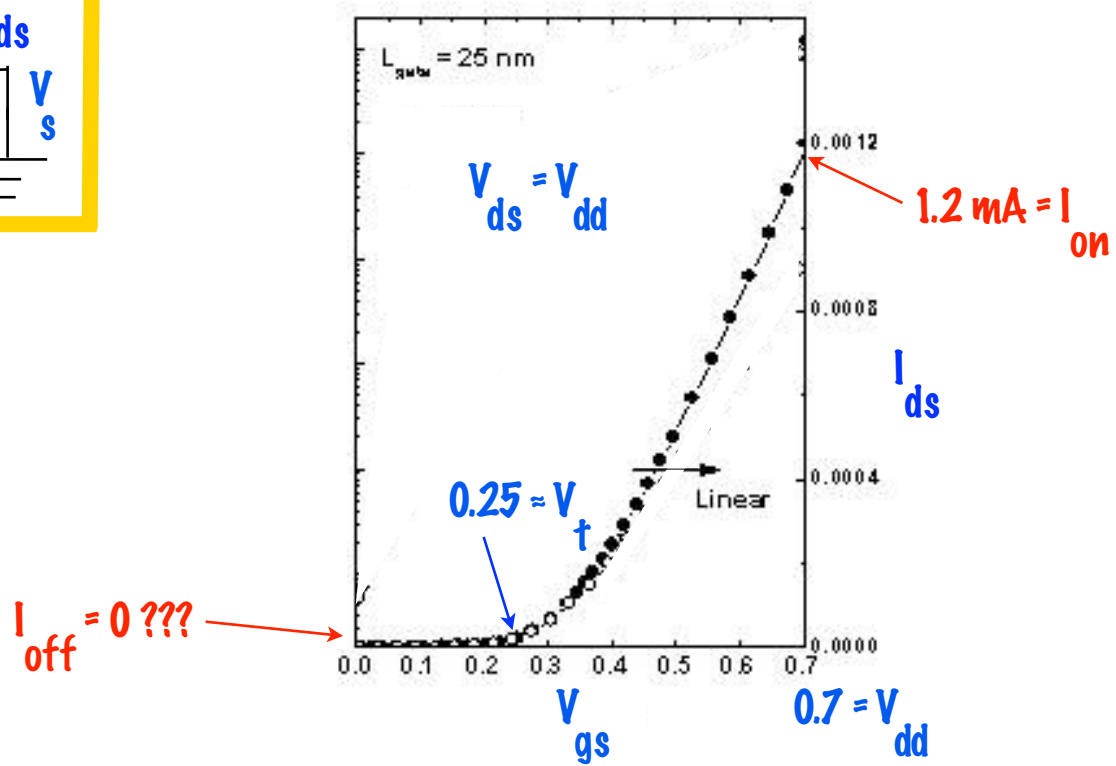


Let's look beneath the abstraction:
V_T *and* ***R_{on}***

MOSFET Threshold Voltage

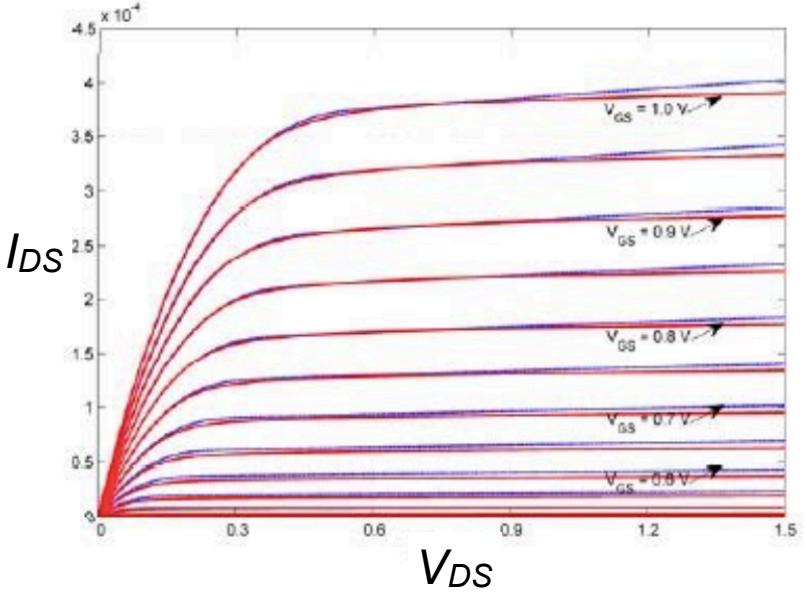


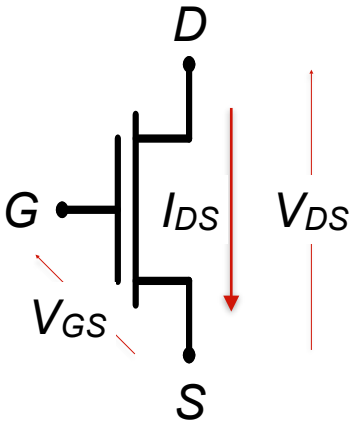
Transistor "turns on" when V_{gs} is > V_t.



Transistor "resistance"

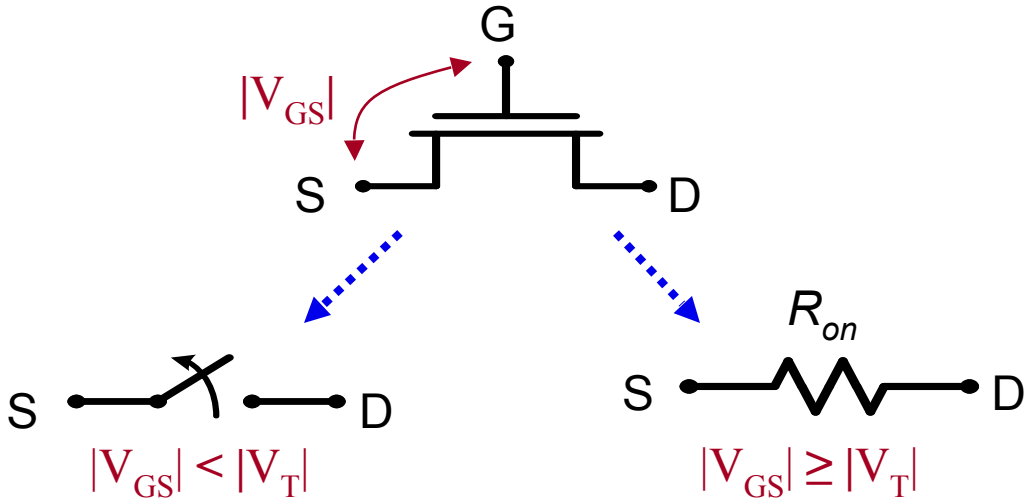
- Nonlinear I/V characteristic:



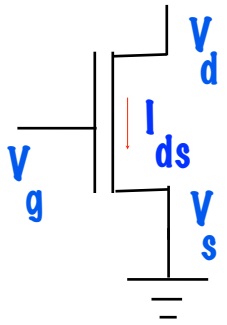


- But, linearizing makes all delay and power calculations simple (usually just 1st order ODEs):

ON/OFF Switch Model of MOS Transistor



Plot on a "Log" Scale to See "Off" Current

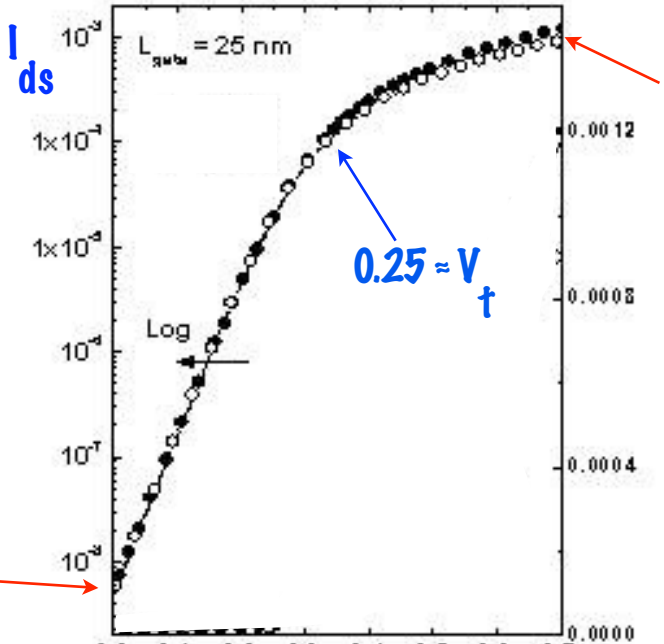


$V_{ds} = V_{dd}$

Process engineers can:

increase I_{on} by lowering V_t - but that raises I_{off}

decrease I_{off} by raising V_t - but that lowers I_{on}.



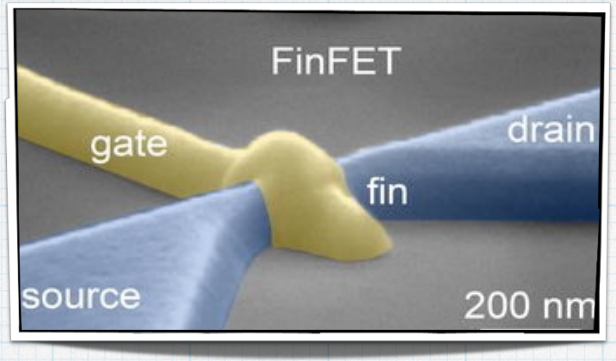
1.2 mA = I_{on}

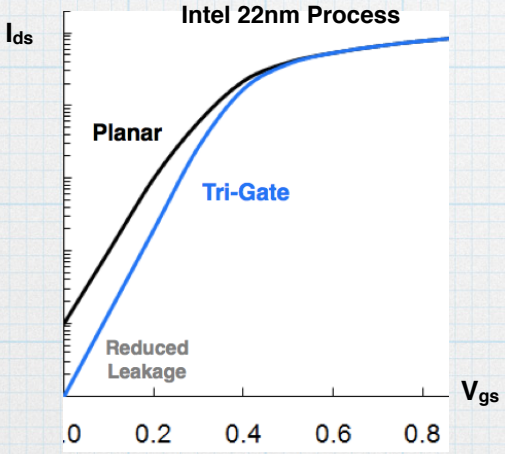
0.25 ≈ V_t

$I_{off} \approx 10$ nA

0.7 = V_{dd}

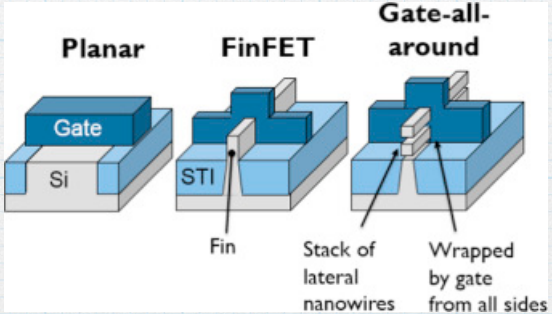
Latest Modern Process



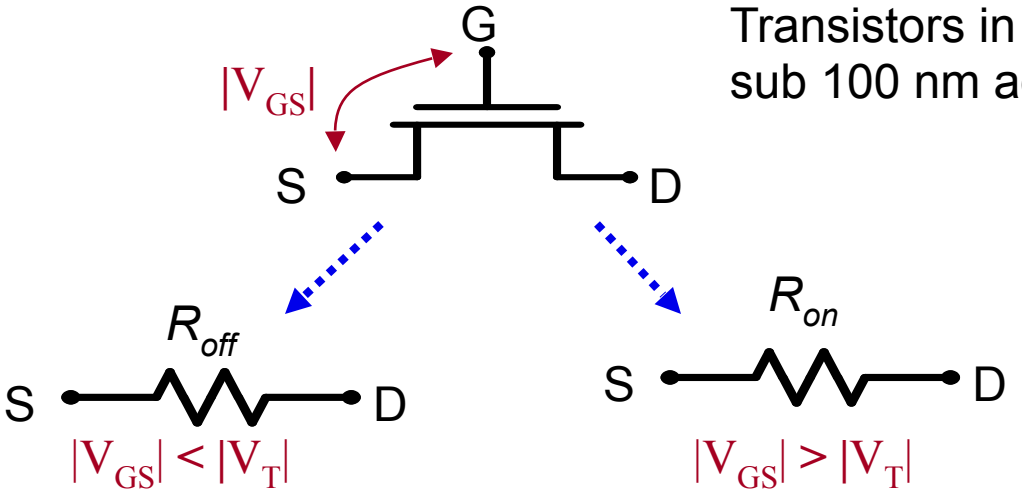


Transistor channel is a raised fin.

Gate controls channel from sides and top.

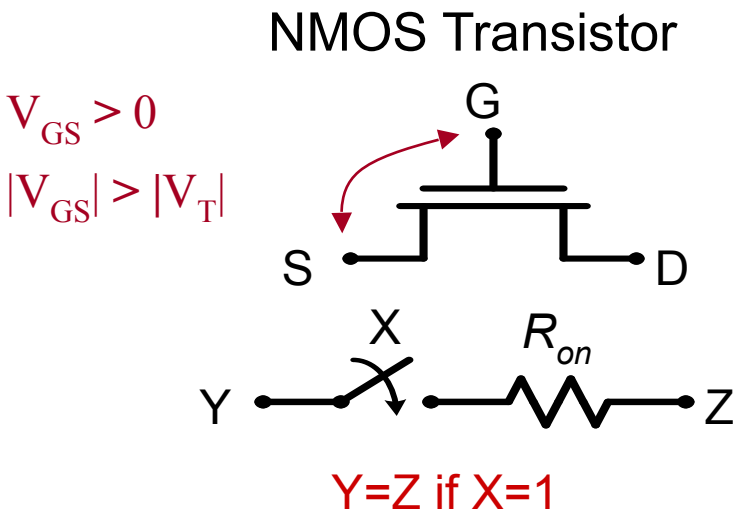


A More Realistic Switch

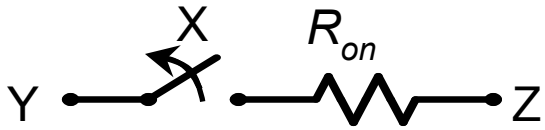


Transistors in the sub 100 nm age

A Logic Perspective



A Complementary Switch

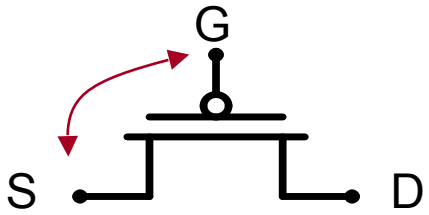


Y=Z if X=0

PMOS Transistor

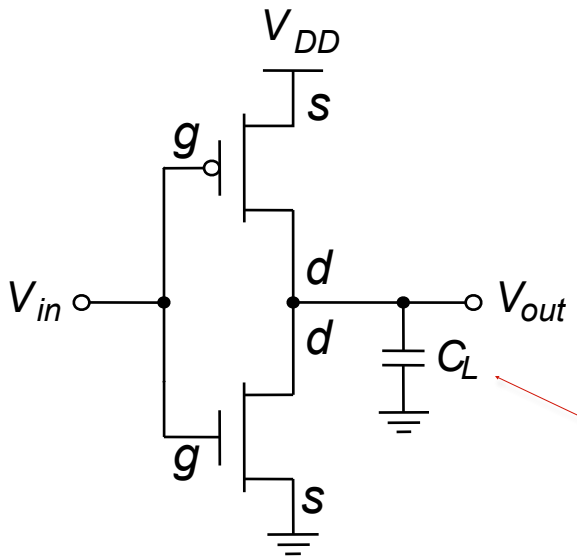
$V_{GS} < 0$

$|V_{GS}| > |V_T|$



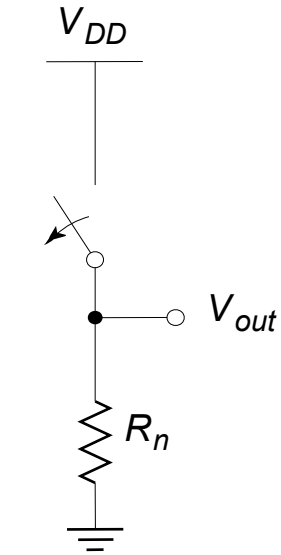
Remember, source is the node w/ the highest voltage.

The CMOS Inverter: A First Glance

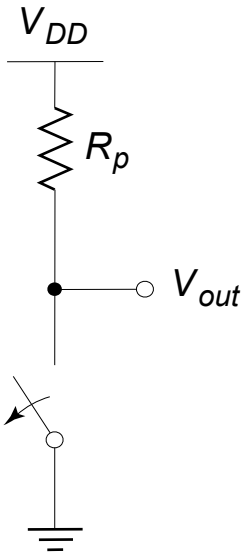


Represents the sum of all the capacitance at the output of the inverter and everything to which it connects: (drains, interconnections gate capacitance of next gate(s))

The Switch Inverter
First-Order DC Analysis*



$V_{in} = V_{DD}$

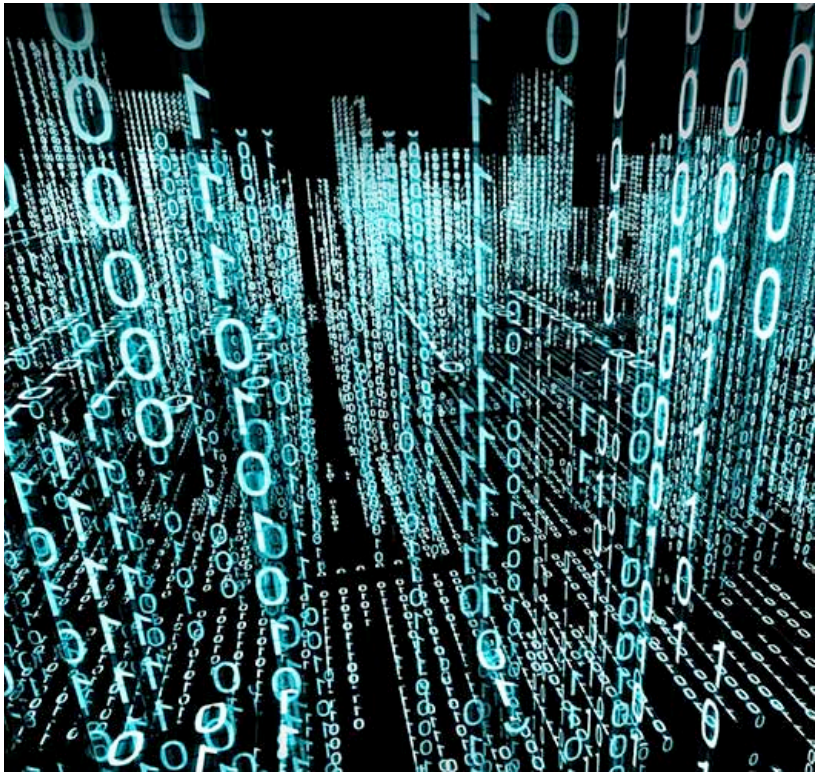


$V_{in} = 0$

$V_{OL} = 0$

$V_{OH} = V_{DD}$

*First-order means we will ignore Capacitance.



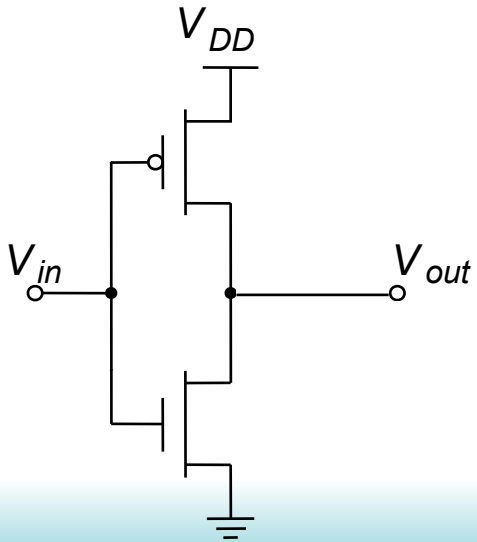
Switch logic

Static Logic Gates (most common and straight forward type of gate)

- At every point in time (except during the switching transients) each gate output is connected to either V_{DD} or V_{GND} via a low resistive path.

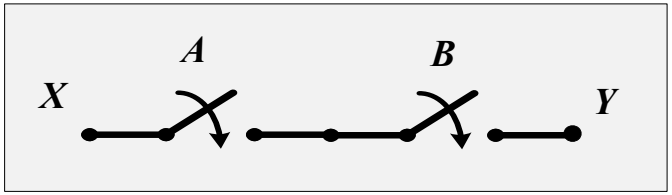
- The output of the gate assumes at all times the value of the Boolean function implemented by the circuit (ignoring, once again, the transient effects during switching periods).

Example: CMOS Inverter



Building logic from switches (NMOS)

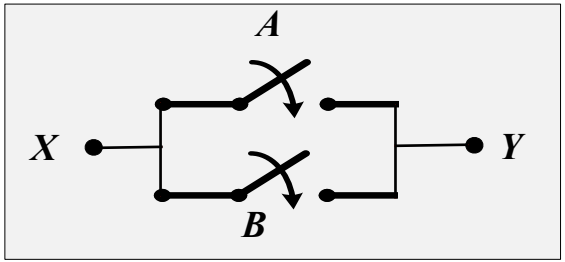
Series



AND

$Y = X$ if A AND B

Parallel



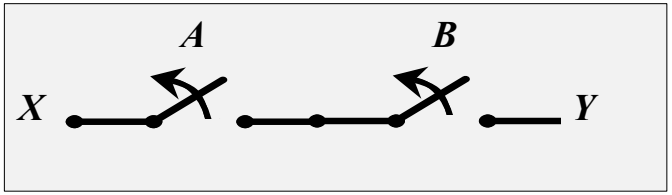
OR

$Y = X$ if A OR B

(output undefined if condition not true)

Logic using inverting switches (PMOS)

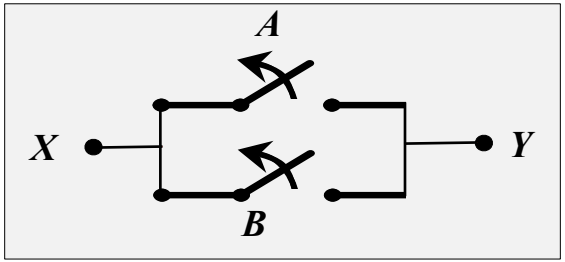
Series



NOR

$$Y = X \text{ if } \overline{A} \text{ AND } \overline{B}$$
$$= \overline{A + B}$$

Parallel



NAND

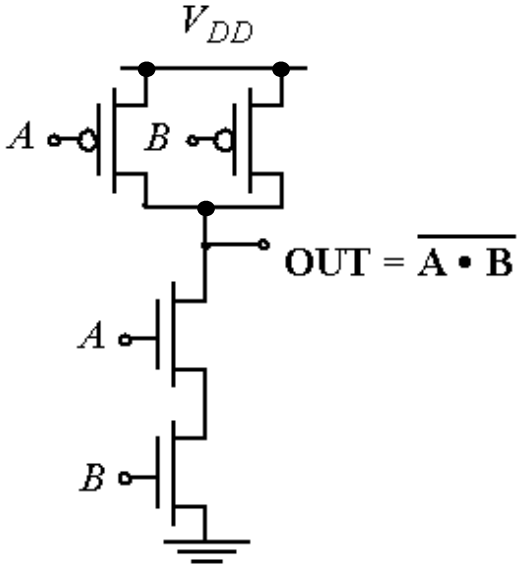
$$Y = X \text{ if } \overline{A} \text{ OR } \overline{B}$$
$$= \overline{AB}$$

(output undefined if condition not true)

Example Gate: NAND

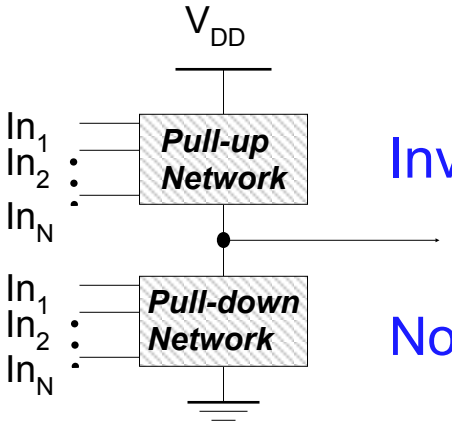
A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

Truth Table of a 2 input NAND gate



- pull-down network: G = AB ⇒ Conduction to GND
- pull-up network: F = $\overline{A}$ + $\overline{B}$ = $\overline{AB}$ ⇒ Conduction to V_{DD}

Static Complementary CMOS



Inverting switches (pmos)

$F(In_1, In_2, \ldots In_N)$

Non-Inverting switches (nmos)

PUN and PDN are dual logic networks:

series connections in the PUN are parallel connections in the PDN
parallel connections in the PUN are series connection sin the PDN

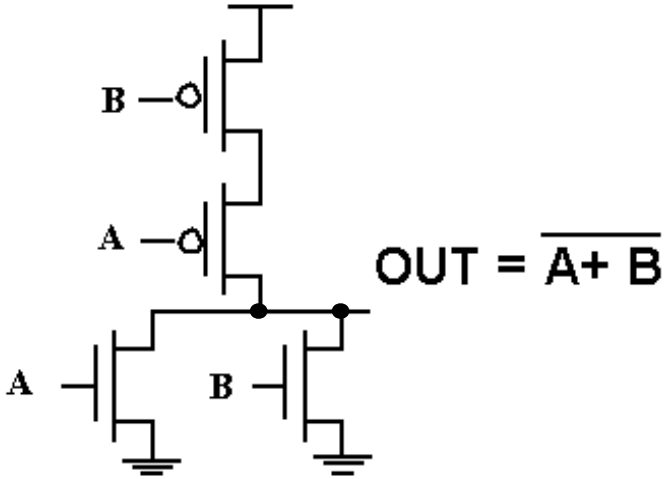
PUN and PDN functions are complements:

guarantees they are mutually exclusive, under all input values,
one or the other is conductive, but never both!

Example Gate: NOR

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

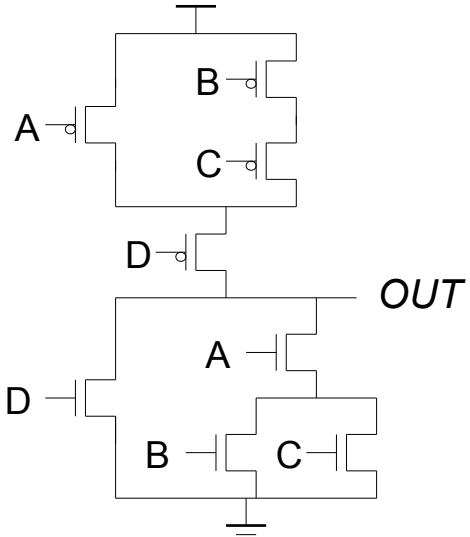
Truth Table of a 2 input NOR gate



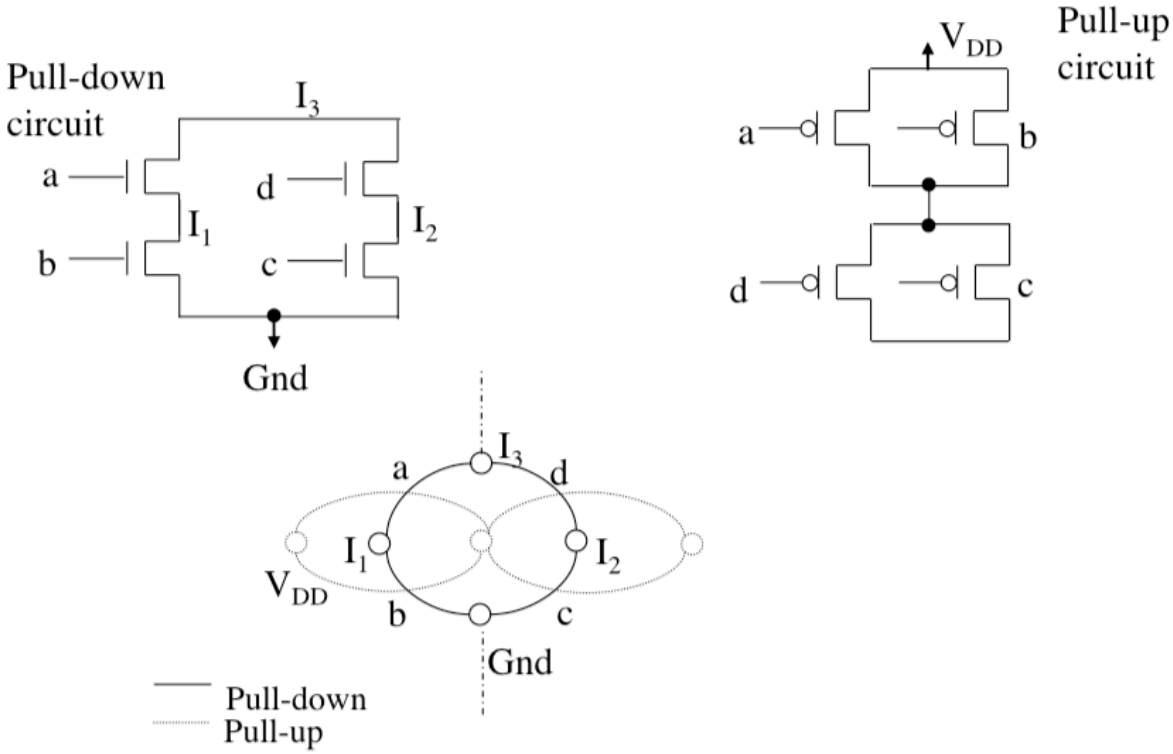
Complex CMOS Gate

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

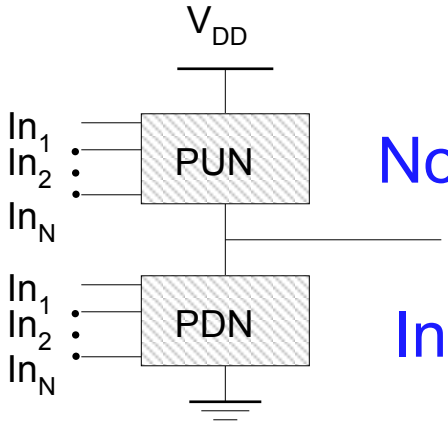
$$OUT = \overline{D \cdot A + B \cdot C}$$



Graph Models for Duals



Non-inverting logic



$F(In_1, In_2, \ldots In_N)$

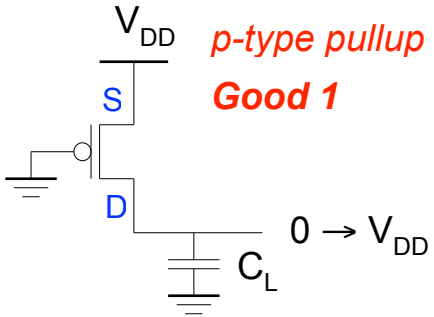
Non-Inverting switches (nmos)

Inverting switches (pmos)

Why is this a bad idea?

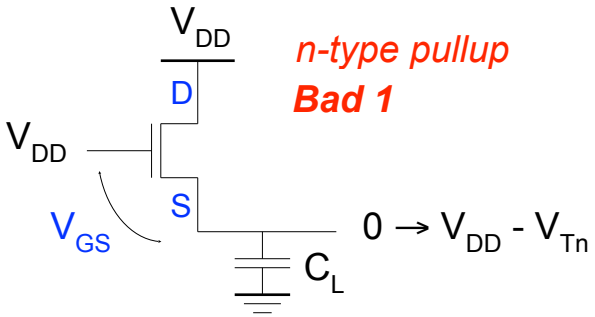
PUN and PDN are dual logic networks
PUN and PDN functions are complementary

Switch Limitations



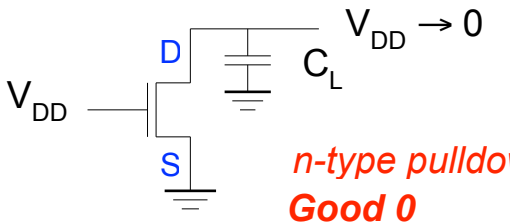
p-type pullup

Good 1



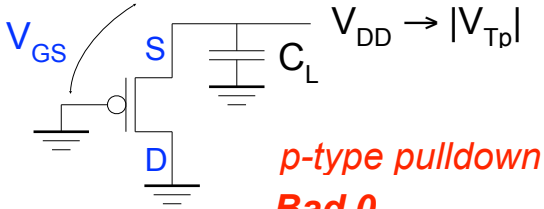
n-type pullup

Bad 1



n-type pulldown

Good 0

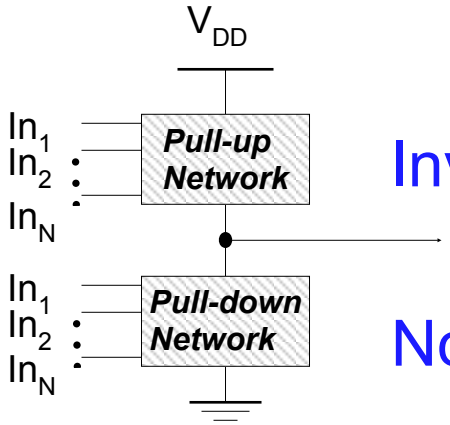


p-type pulldown

Bad 0

Tough luck …

"Static" CMOS gates

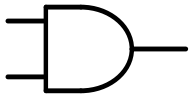


Inverting switches (PMOS transistors)

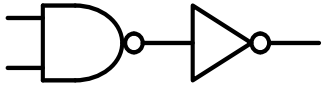
$F(In_1, In_2, \ldots In_N)$

Non-Inverting switches (NMOS transistors)

- Static CMOS gates are always inverting

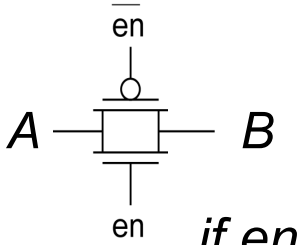


AND = NAND , INV



Transmission Gate

- Transmission gates are the way to build ideal “switches” in CMOS.
- In general, for an ideal switch, both transistor types are needed:
  - nFET to pass zeros.
  - pFET to pass ones.
- The transmission gate is bi-directional (unlike logic gates).

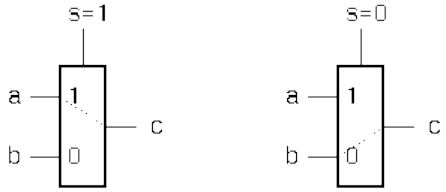


if en==1 then A connects to B

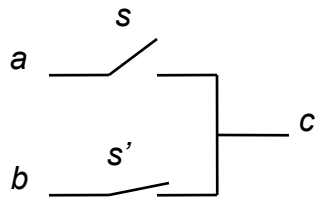
- Does not directly connect to Vdd and GND, but can be combined with logic gates or inverters to simplify many logic functions.

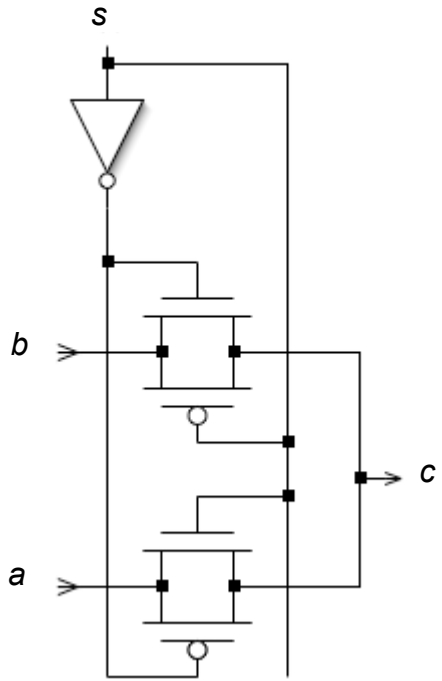
Transmission-gate Multiplexor

2-to-1 multiplexor:
c = sa + s'b



Switches simplify the implementation:



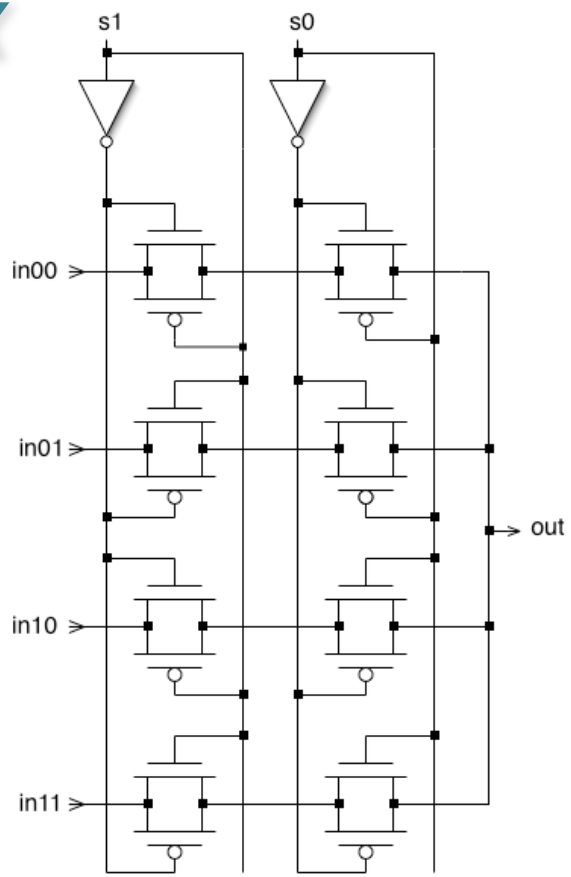


Compare the cost to logic gate implementation.

Care must be taken to not string together many pass-transistor stages.
Occasionally, need to "rebuffer" with static gate.

4-to-1 Transmission-gate Mux

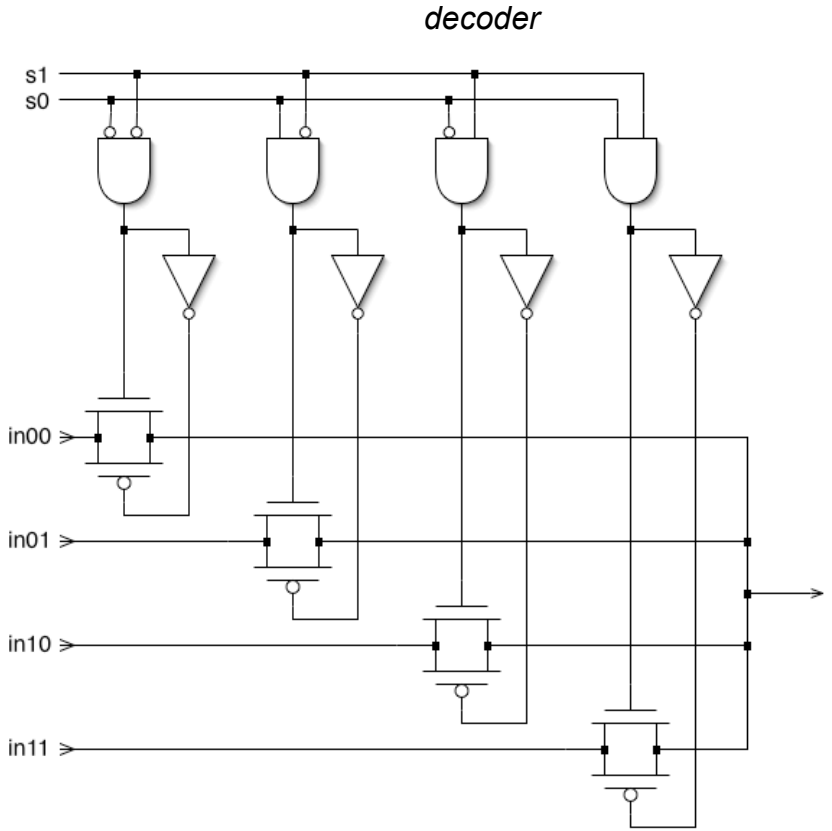
- The series connection of pass-transistors in each branch effectively forms the AND of s1 and s0 (or their complement).
- Compare cost to logic gate implementation



Any alternate solutions?

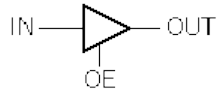
Alternative 4-to-1 Multiplexor

- This version has less delay from in to out.
- In both versions, care must be taken to avoid turning on multiple paths simultaneously (shorting together the inputs).



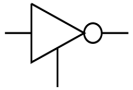
Tri-state Buffers

Tri-state Buffer:

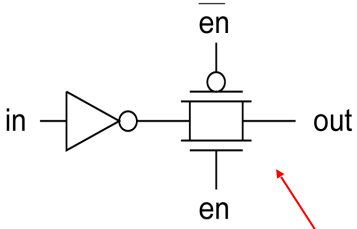


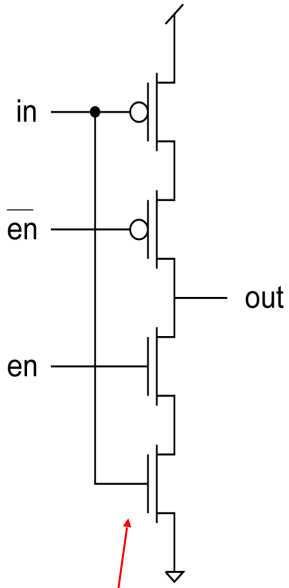
OE	IN	OUT
0	0	Z
0	1	Z
1	0	0
1	1	1

"high impedance" (output disconnected)



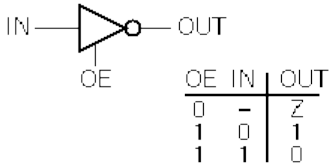
CMOS Implementation





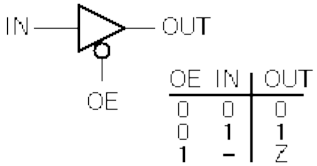
transmission gate provide the isolation: usually designed this way

Variations:



OE	IN	OUT
0	–	Z
1	0	1
1	1	0

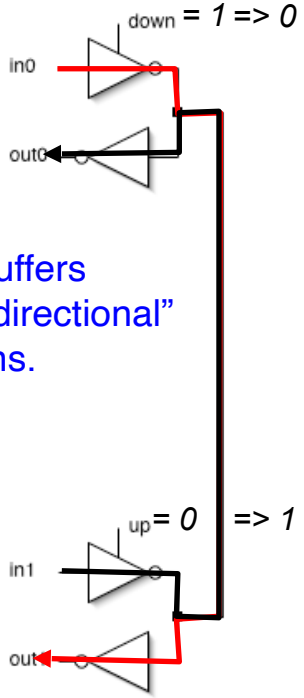
Inverting buffer



OE	IN	OUT
0	0	0
0	1	1
1	–	Z

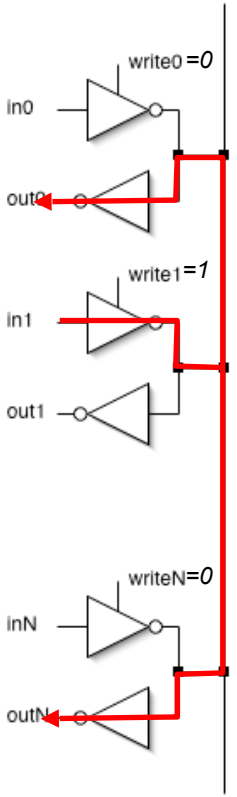
Inverted enable

Tri-state Buffers



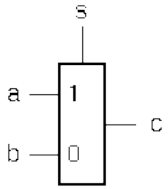
Tri-state buffers enable “bidirectional” connections.

Tri-state buffers are used when multiple circuits all connect to a common node or wire. Only one circuit at a time is allowed to drive the bus. All others “disconnect” their outputs, but can “listen”.

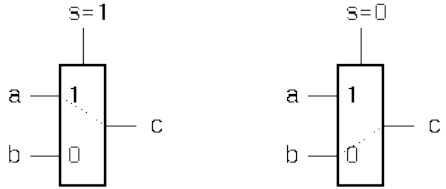


Tri-state Based Multiplexor

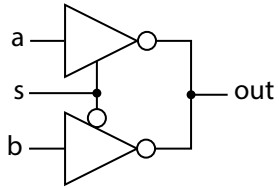
Multiplexor:

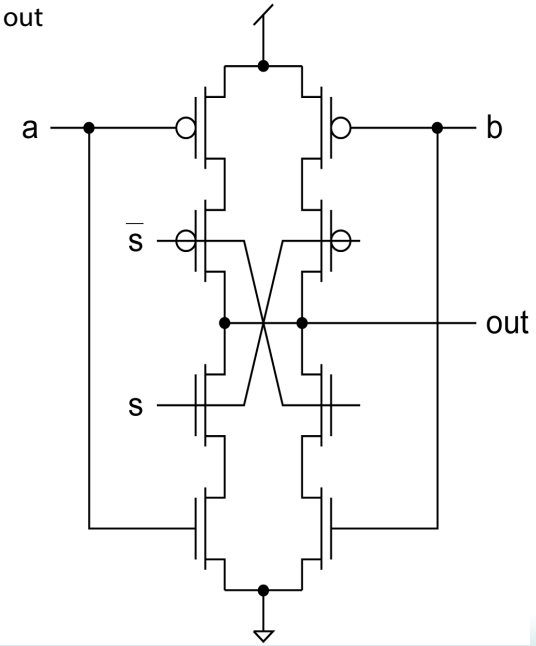


If s=1 then c=a else c=b



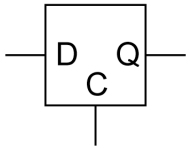
Transistor Circuit for inverting-multiplexor:

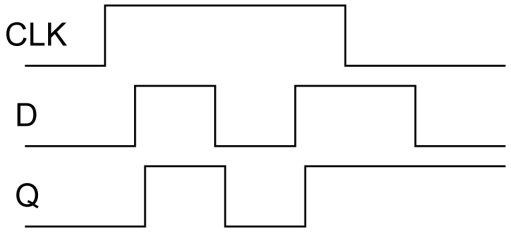




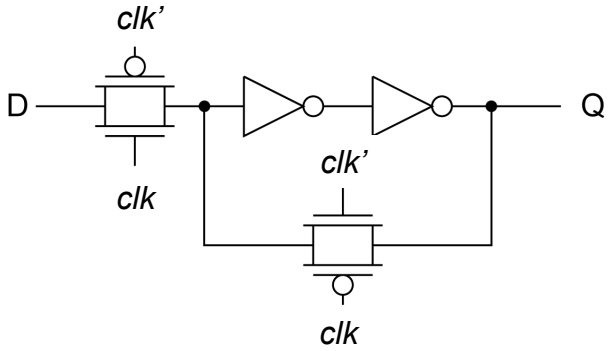
Latches and Flip-flops

Positive Level-sensitive ***latch***:



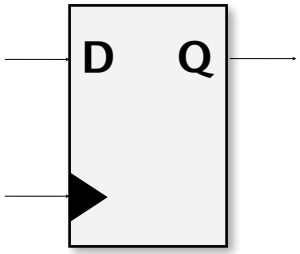


Latch *Implementation:*

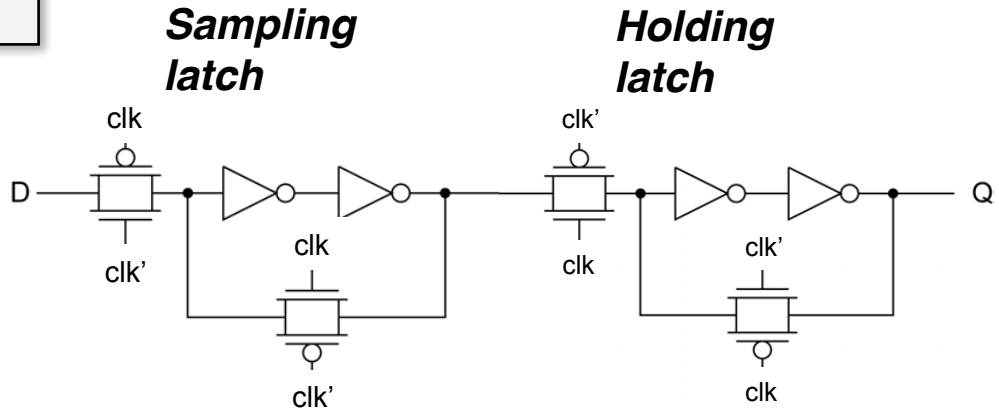


When the clock is high, the latch is "transparent", when clock is low, it holds the last seen value

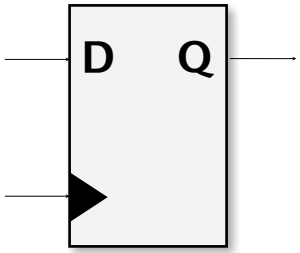
Positive edge-triggered flip-flop



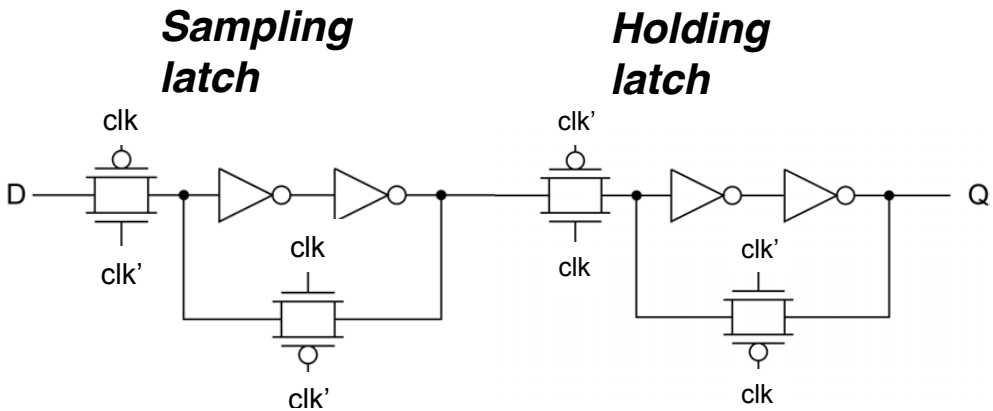
A flip-flop "samples" on the positive clock-edge, and then "holds" value.



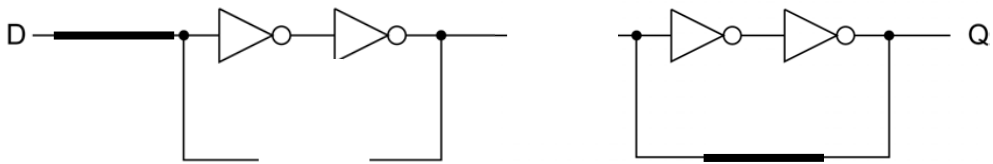
Sensing: When clock is low



A flip-flop "samples" on the positive clock-edge, and then "holds" value.



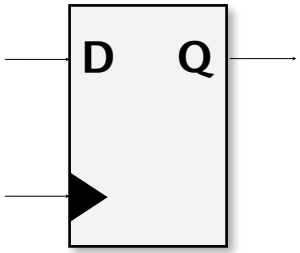
clk = 0
clk' = 1



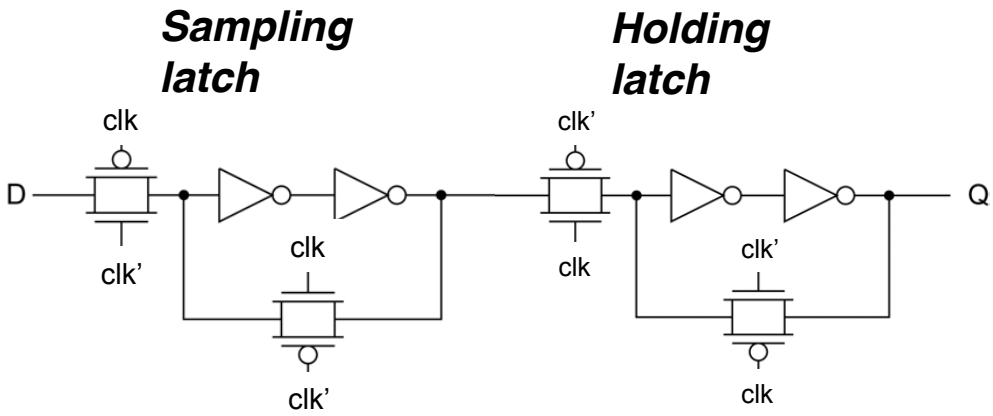
Ready to capture new value on clock edge

Is outputing last previous captured value.

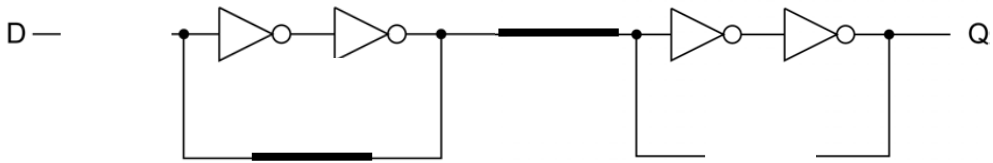
Capture: When clock goes high



A flip-flop "samples" on the positive clock-edge, and then "holds" value.



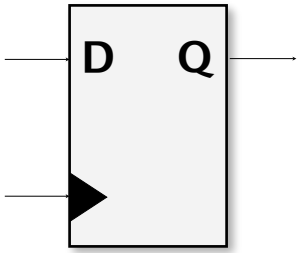
clk = 1
clk' = 0



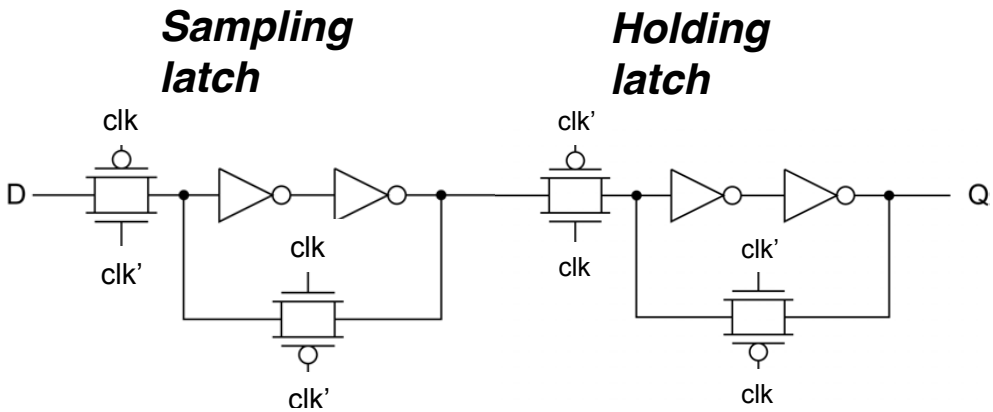
Remembers value just captured.

Outputs value just captured.

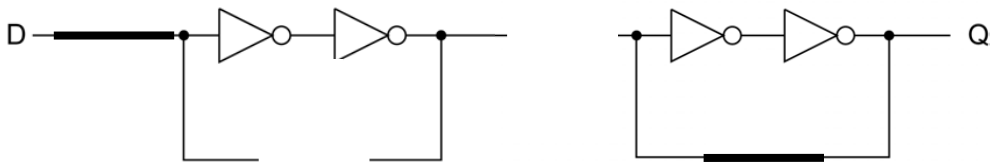
Return to sensing: When clock is low



A flip-flop "samples" just before the edge, and then "holds" value.



clk = 0
clk' = 1

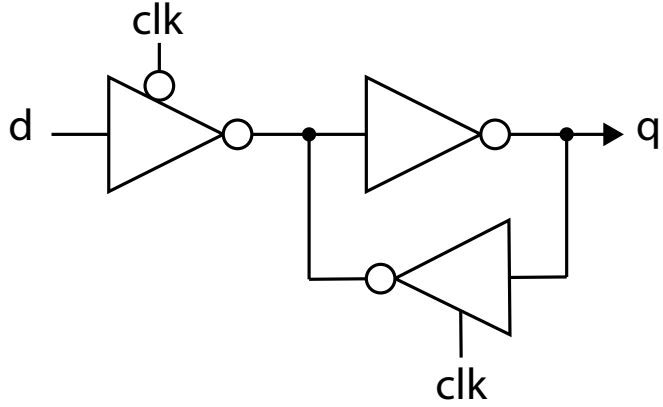


Ready to capture new value on clock edge

Continue to output captured value.

Tri-state-Inverter Latch implementation

- Commonly used in standard cell flip-flops.
- More transistors than pass-transistor version, but more robust.
- Lays out well with modern layout rules.



Negative Level-sensitive latch: