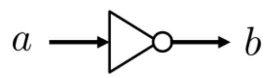
Beyond classical circuit design
lecture 8

CMOS Synthesis

Further Reading

Jan M. Rabaey, Anantha Chandrakasan, Borivoje Nikolic: *Digital Integrated Circuits. A Design Perspective. 2nd edition.* Prentice Hall, 2003.

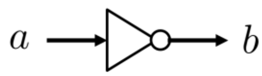
Back to synthesis...



$$a \rightarrow b\downarrow$$
$$\neg a \rightarrow b\uparrow$$

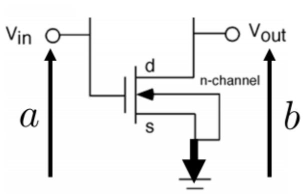
CMOS inverter + ideal characteristics

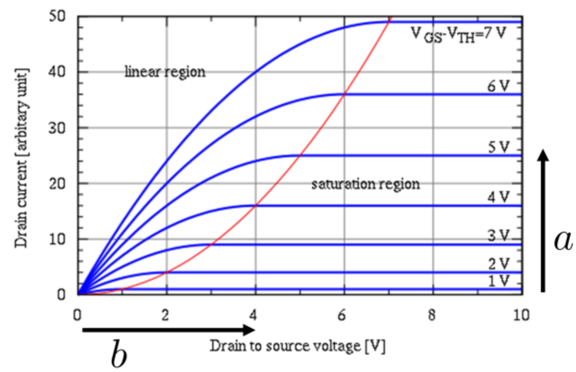
Back to synthesis…



$a \to b\downarrow$

$\neg a \to b\uparrow$

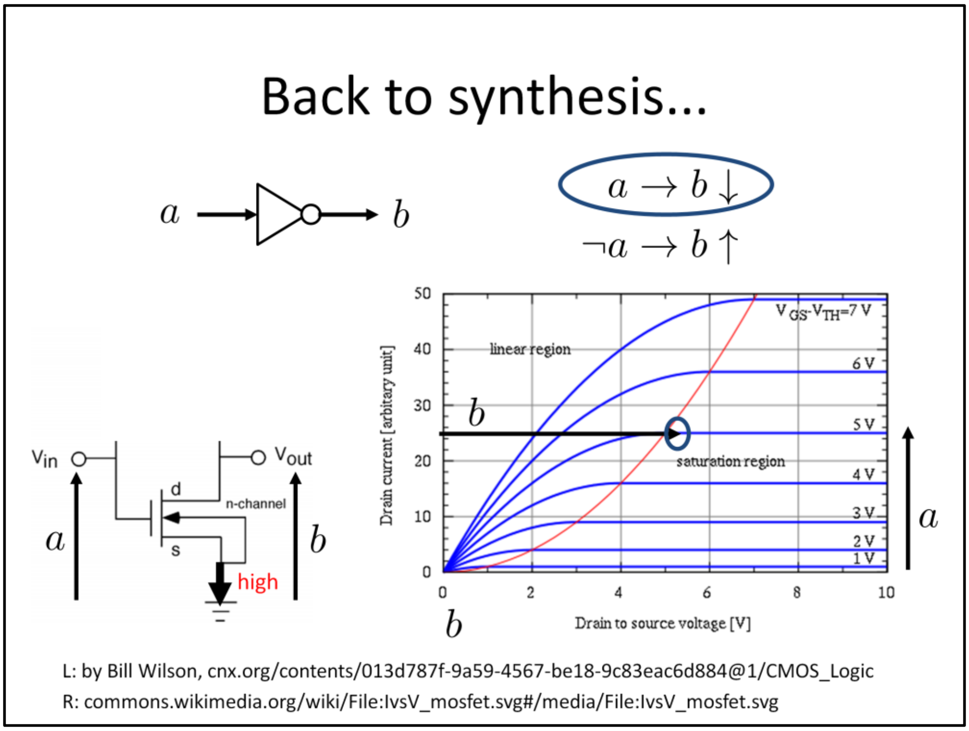




L: by Bill Wilson, cnx.org/contents/013d787f-9a59-4567-be18-9c83eac6d884@1/CMOS_Logic

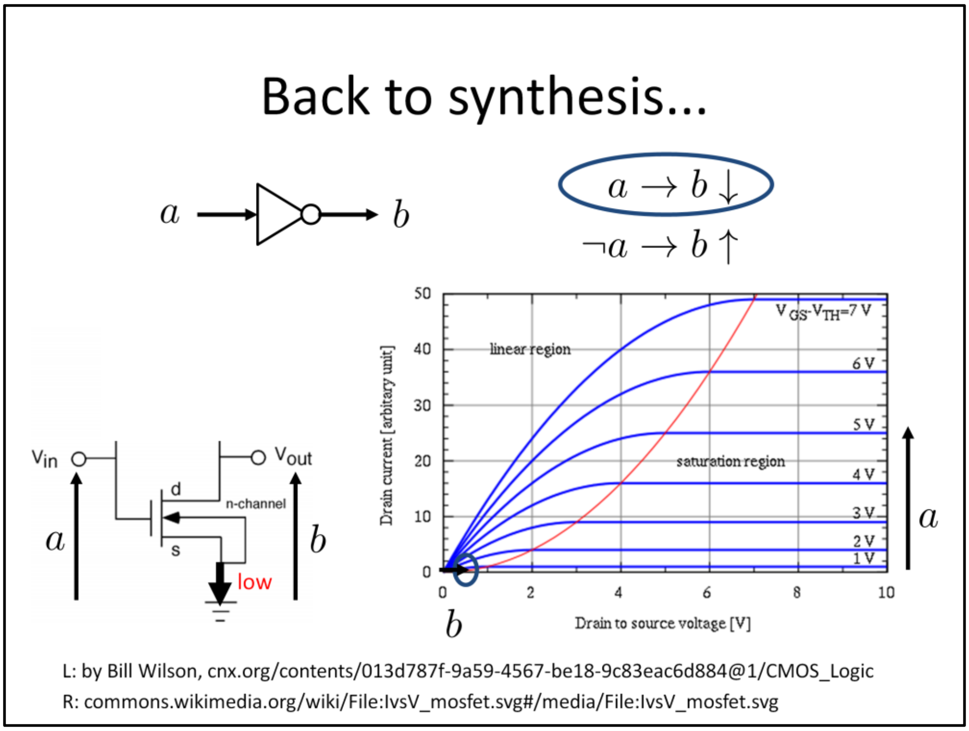
R: commons.wikimedia.org/wiki/File:IvsV_mosfet.svg#/media/File:IvsV_mosfet.svg

n-stack
CMOS inverter



n-stack
CMOS inverter

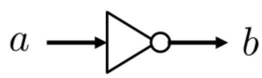
high a & b -> high DS current!



n-stack
CMOS inverter

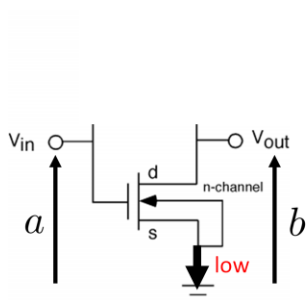
now this is possible

Back to synthesis...



$a \to b\downarrow$

$\neg a \to b\uparrow$



low a ->

nFET in cutoff region

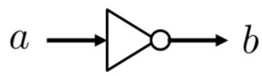
high a ->

nFET in ohmic region

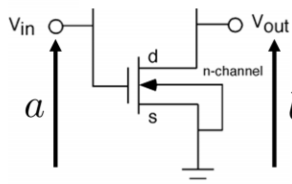
by Bill Wilson, cnx.org/contents/013d787f-9a59-4567-be18-9c83eac6d884@1/CMOS_Logic

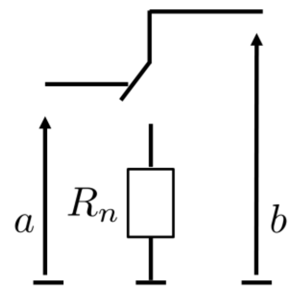
from the characteristics from the slide before we see that the nFET, in presence of a high input a, is in the ohmic region ->
it behaves like a resistor

Back to synthesis...



$$a \to b\downarrow$$
$$\neg a \to b\uparrow$$



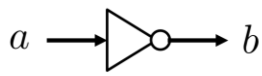


by Bill Wilson, cnx.org/contents/013d787f-9a59-4567-be18-9c83eac6d884@1/CMOS_Logi

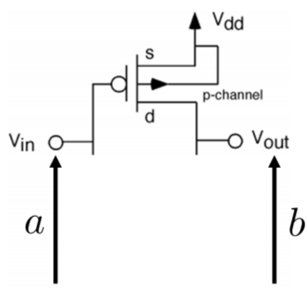
we can approximate nMOSFET as a switch here.

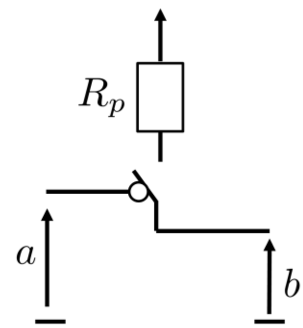
open & ohmic region

Back to synthesis...



$$a \to b\downarrow$$
$$\neg a \to b\uparrow$$

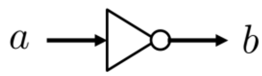




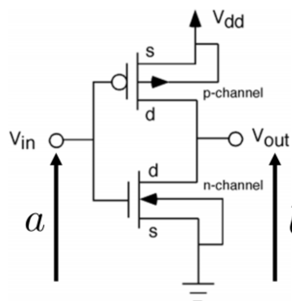
by Bill Wilson, cnx.org/contents/013d787f-9a59-4567-be18-9c83eac6d884@1/CMOS_Logic

p-stack
CMOS inverter

Putting it together

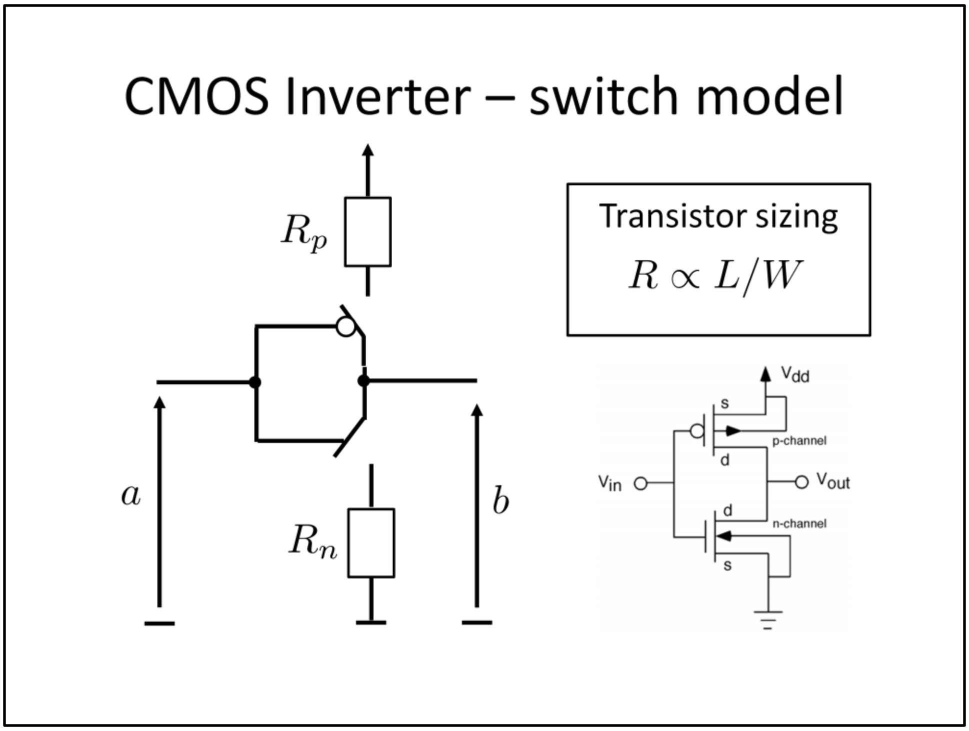


$$a \rightarrow b\downarrow$$
$$\neg a \rightarrow b\uparrow$$



by Bill Wilson, cnx.org/contents/013d787f-9a59-4567-be18-9c83eac6d884@1/CMOS_Logic

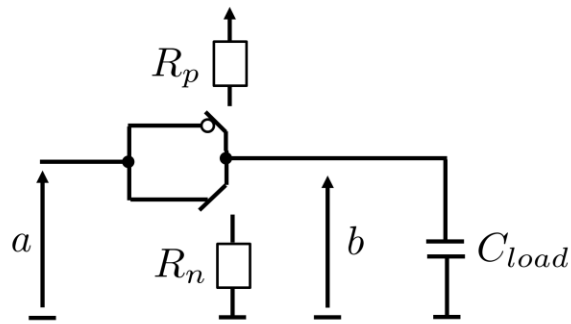
p-stack
CMOS inverter

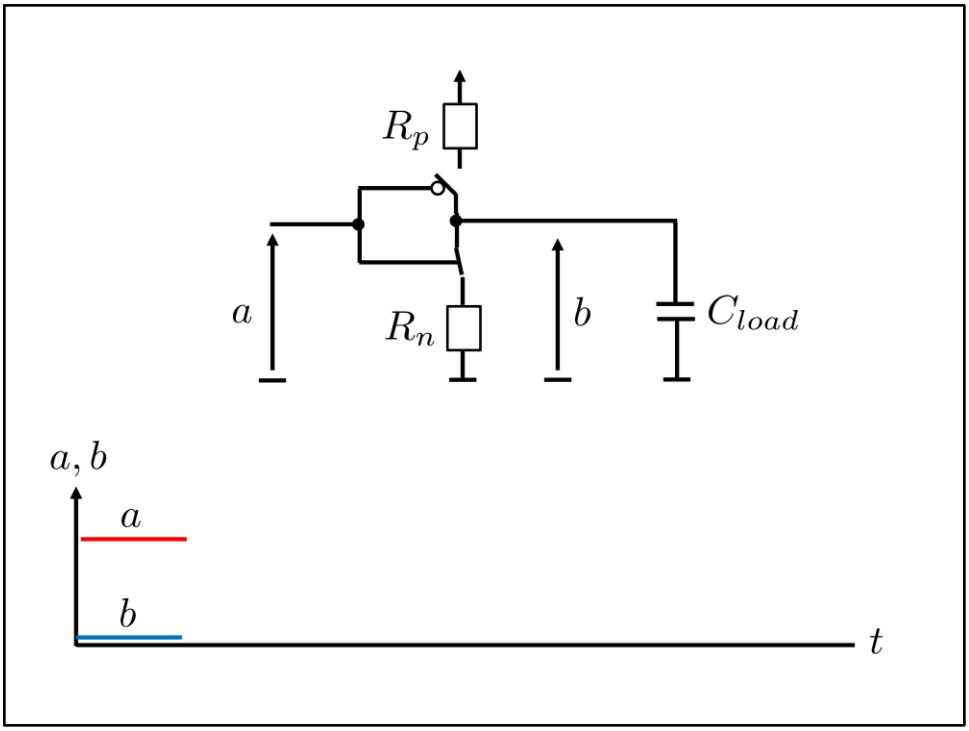


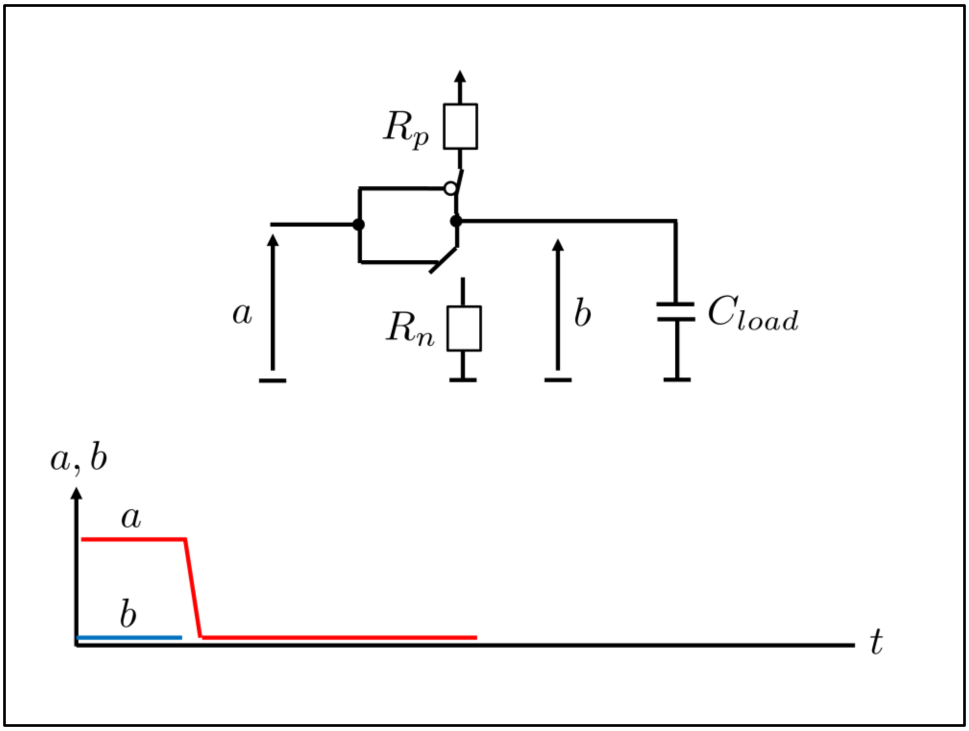
steady state (static) model: CMOS as switches
in this model: exactly one open and one closed at a time.
-> low static power consumption

W... width of channel
L... length of channel

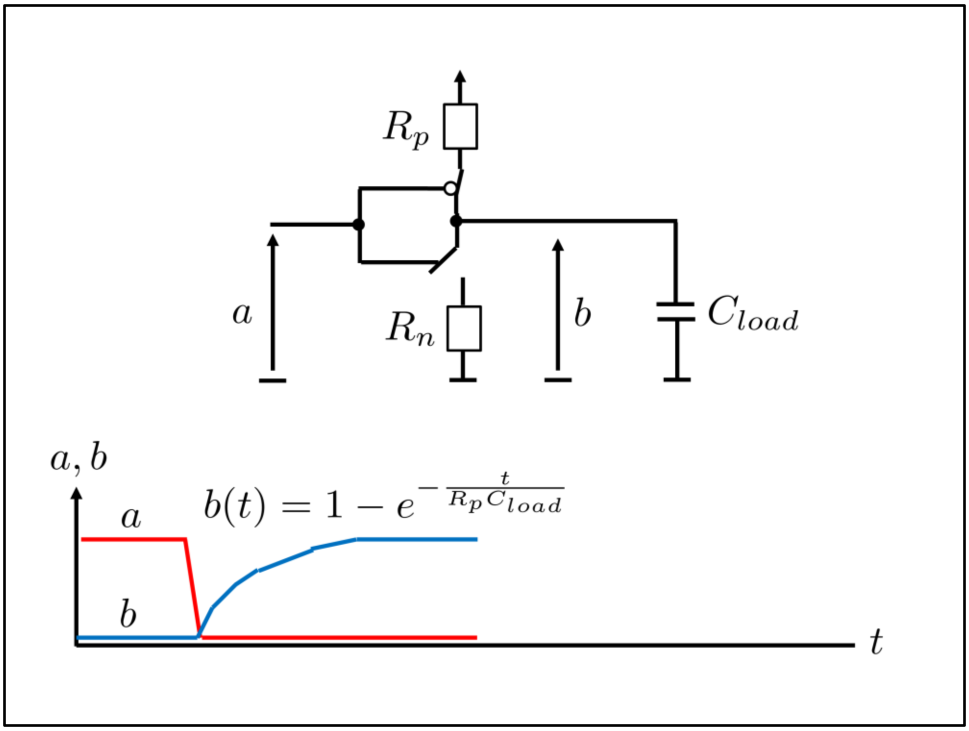
CMOS Inverter – switch transient



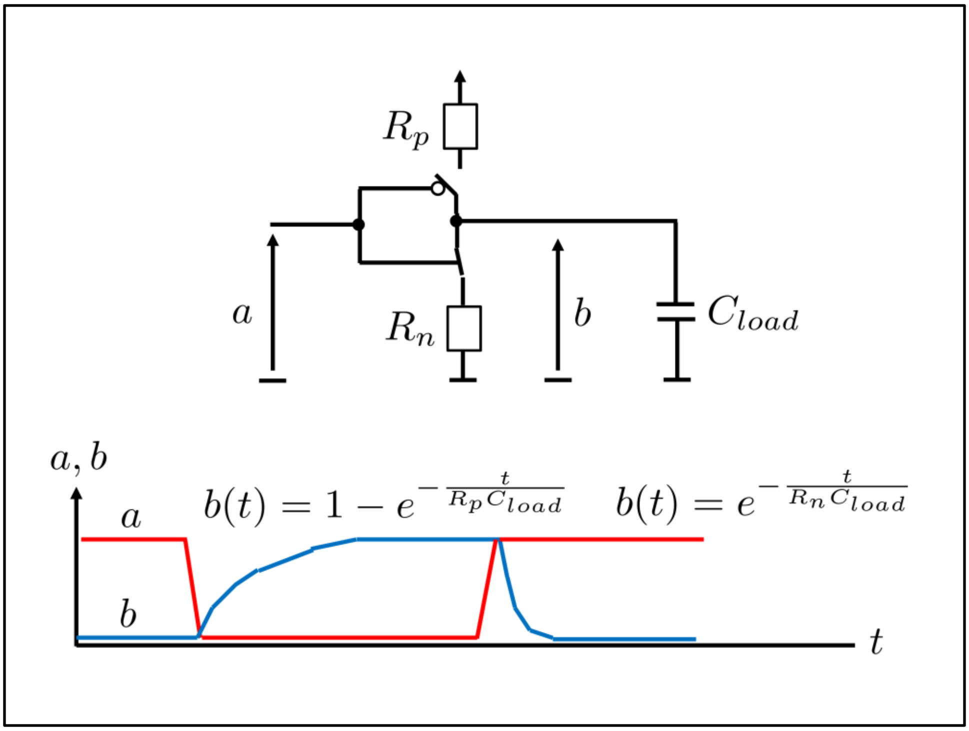

R_p
a
R_n
b
C_{load}
a, b
a
b
t



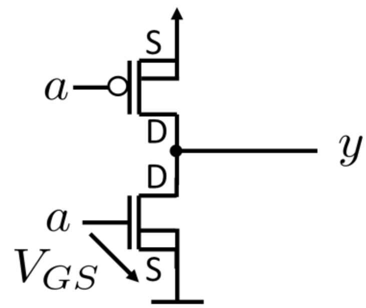
R_p
a
R_n
b
C_{load}
a, b
a
b
t



R_p
R_n
a
b
C_{load}
a, b
a
$b(t) = 1 - e^{-\frac{t}{R_p C_{load}}}$
b
t


R_p
a
R_n
b
C_{load}
a, b
a
$b(t) = 1 - e^{-\frac{t}{R_p C_{load}}}$
$b(t) = e^{-\frac{t}{R_n C_{load}}}$
b
t

Why this way?



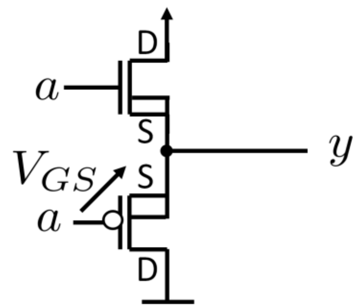
V_{GS} independent of y

-> high a -> nMOS always connects

-> pulls down y to 0V (= strong 0)

pMOS pulling down

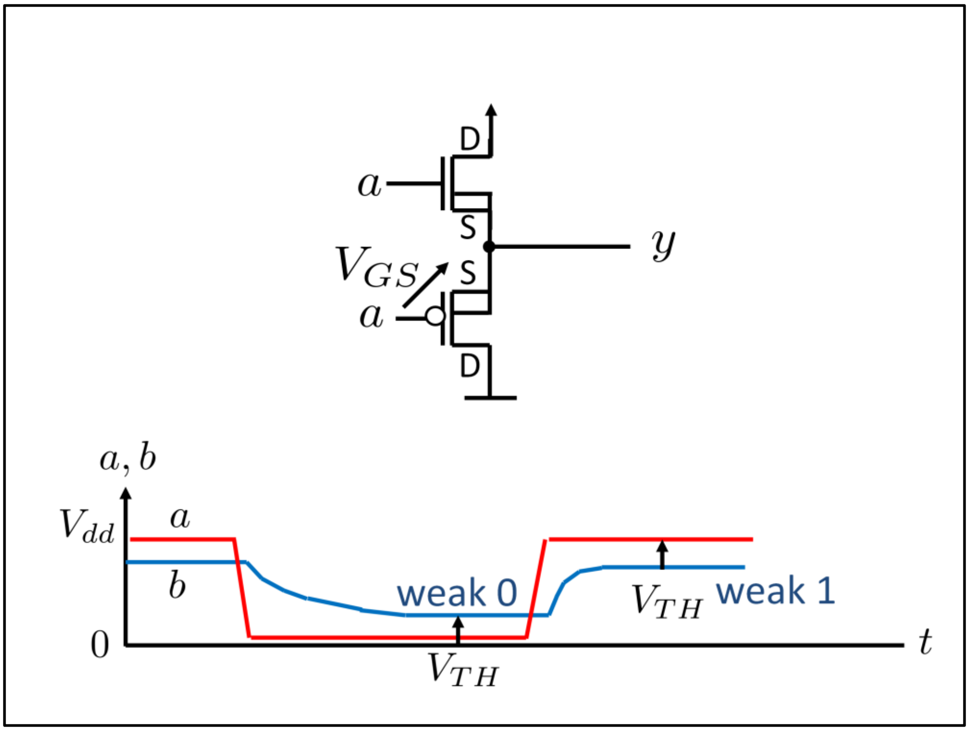
... not inverter,
but also:



V_{GS} depends on y

-> when $y \leq V_{TH}$: pMOS cutoff

-> pulls down y to V_{TH} (= weak 0)



reduced swing

CMOS

nMOS:

drives strong 0, weak 1

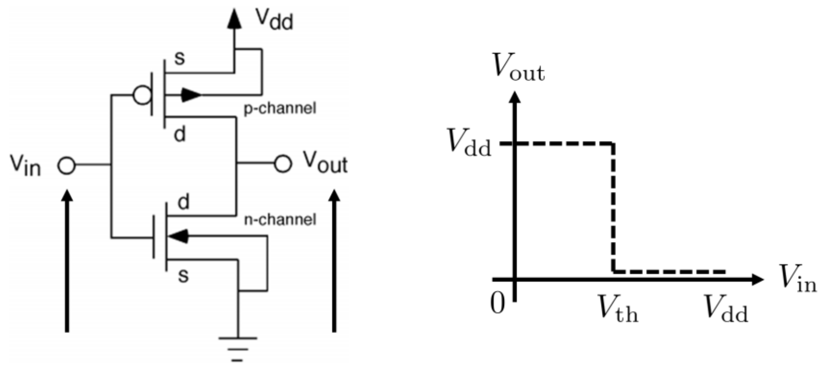
-> used for pull-down

pMOS:

drives strong 1, weak 0

-> used for pull-up

CMOS Inverter – switch model

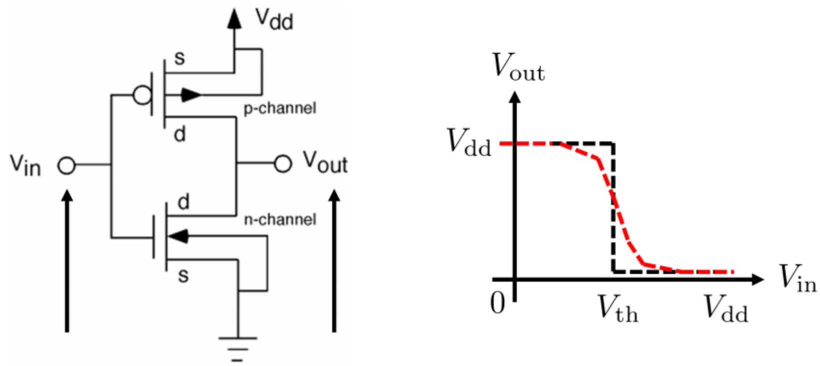


by Bill Wilson, cnx.org/contents/013d787f-9a59-4567-be18-9c83eac6d884@1/CMOS_Logic

CMOS inverter + ideal characteristics

V_th ... threshold voltage

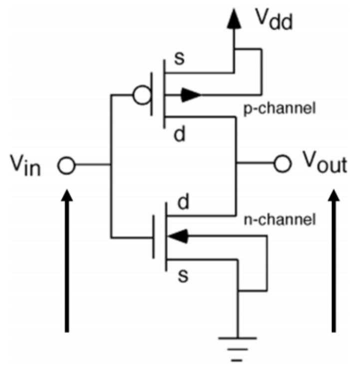
CMOS Inverter – detailed static



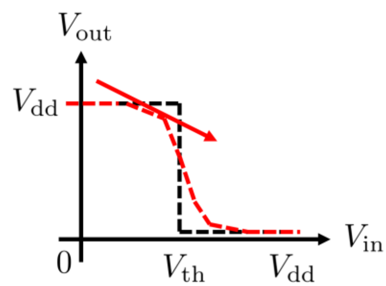
by Bill Wilson, cnx.org/contents/013d787f-9a59-4567-be18-9c83eac6d884@1/CMOS_Logic

red line: realistic inverter characteristics

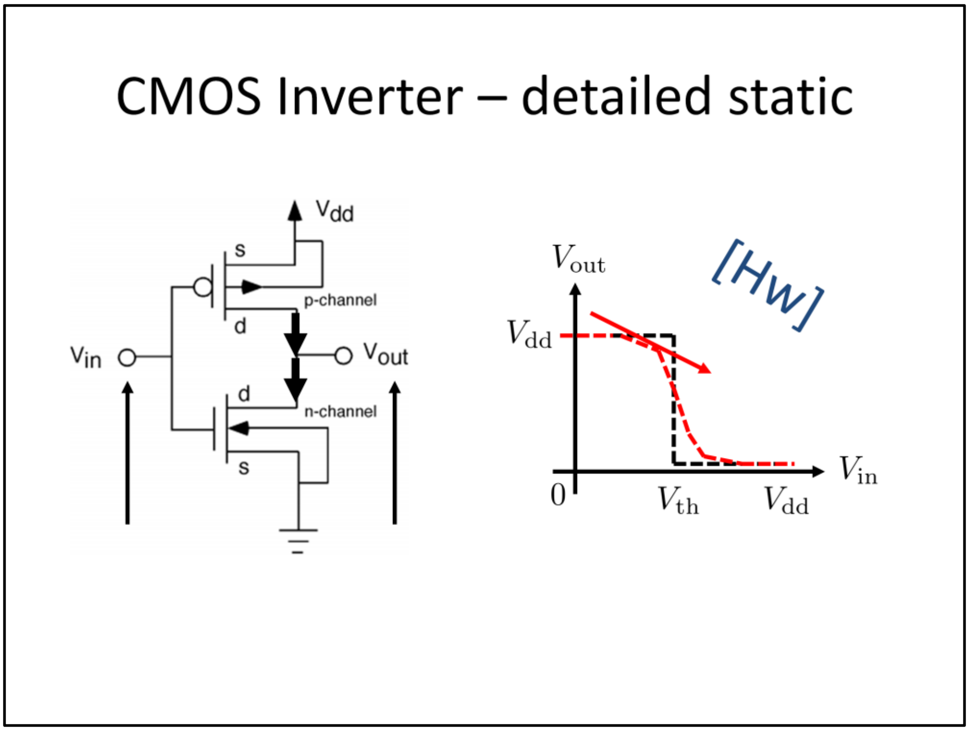
CMOS Inverter – gain



$$g(x) = \left.\frac{dV_{out}}{dV_{in}}\right|_{V_{in}=x}$$

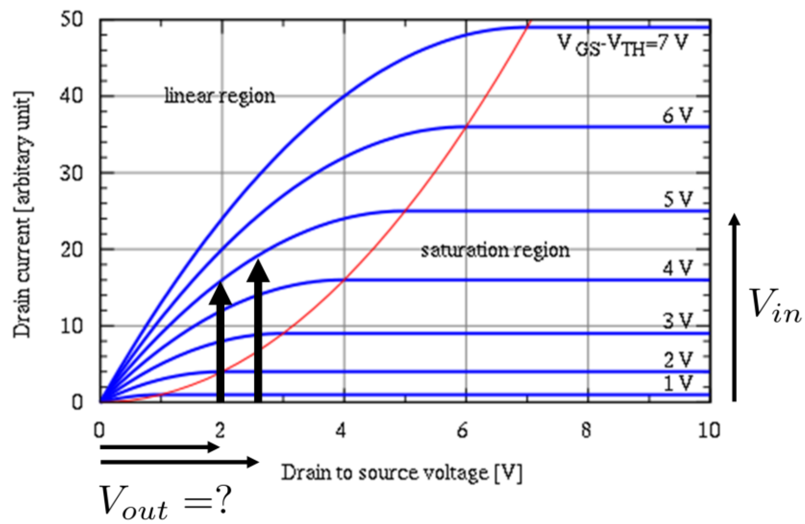


by Bill Wilson, cnx.org/contents/013d787f-9a59-4567-be18-9c83eac6d884@1/CMOS_Logic



assume: no output current in steady state ->
n-channel current = p-channel current

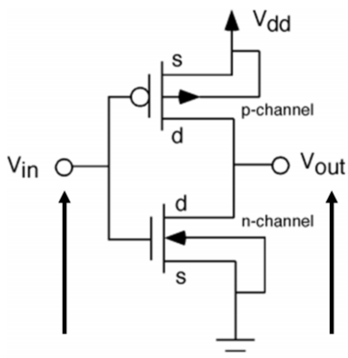
for dependency of gate/drain/source voltages and channel currents see characteristics

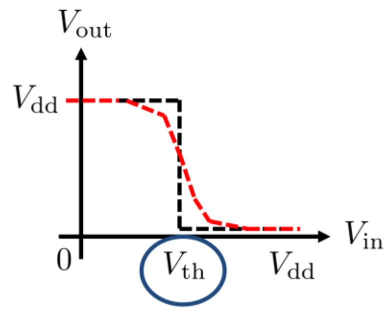


from commons.wikimedia.org/wiki/File:IvsV_mosfet.svg#/media/File:IvsV_mosfet.svg

[hw]: match via current

CMOS Inverter – detailed static





Threshold can be moved by relative sizing

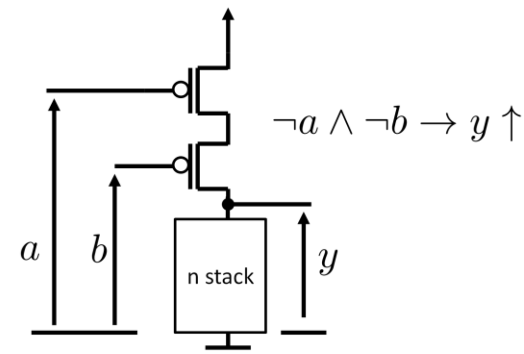
-> low/high-threshold INV

CMOS Design

Combinational Logic:

n-stack: down transition $G \to y \downarrow$

p-stack: up transition $\neg G \to y \uparrow$



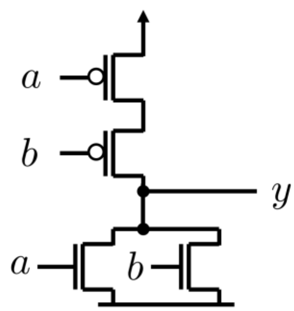
combinational logic design

NOR

Combinational Logic:

n-stack: down transition $a \vee b \to y \downarrow$

p-stack: up transition $\neg(a \vee b) \to y \uparrow$



combinational logic design

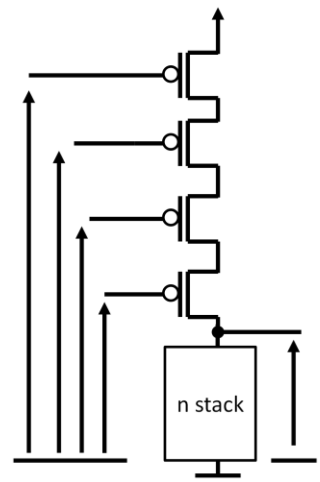
CMOS Design

Deep stacks slow!

esp. mind deep p-stacks

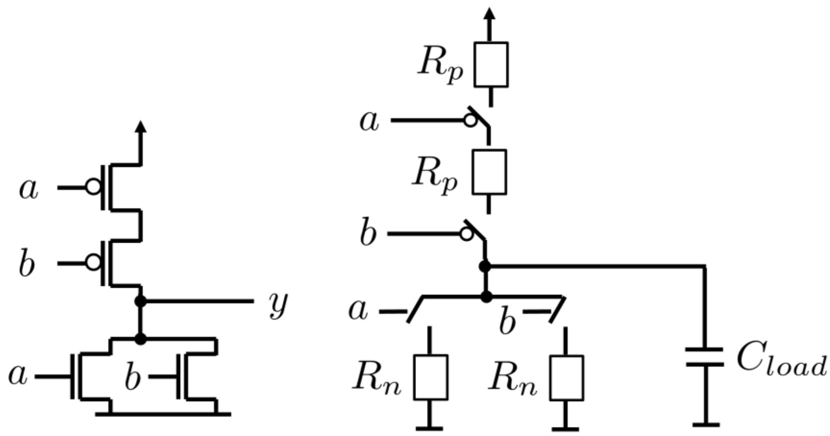
Reason: remember

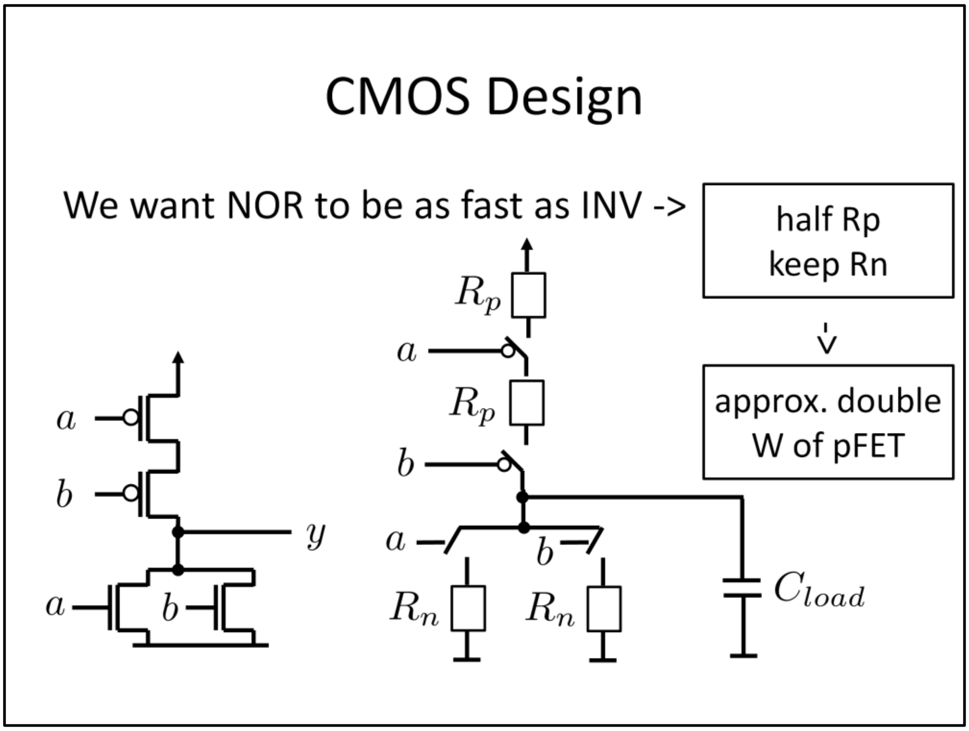
resistor/switch model



CMOS Design

We want NOR to be as fast as INV.





We have to double the width of the pFET but we can keep the width of the nFET here.

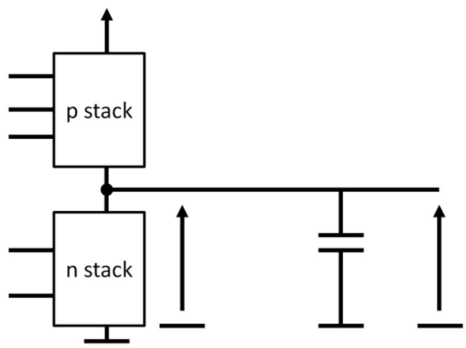
State holding logic?

$$G_{\mathrm{down}} \to y \downarrow$$
$$G_{\mathrm{up}} \to y \uparrow$$

-> statesizer!

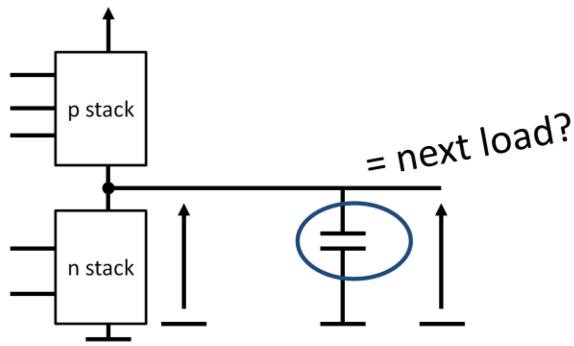
V1 - State holding logic

$$G_{\mathrm{down}} \to y\downarrow$$
$$G_{\mathrm{up}} \to y\uparrow$$



V1’ - State holding logic

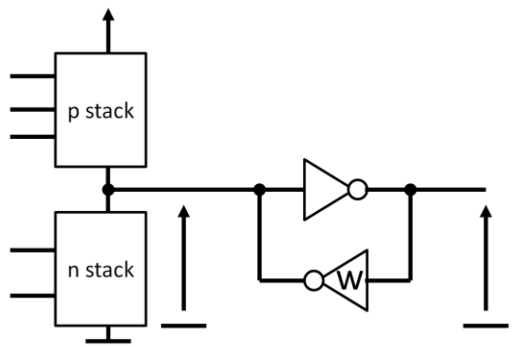
$$G_{\text{down}} \rightarrow y\downarrow$$
$$G_{\text{up}} \rightarrow y\uparrow$$



load C sometimes enough if refreshrate is high enough -> in this case no need for extra logic.

V2 - State holding logic

$$G_{\text{down}} \to y\downarrow$$
$$G_{\text{up}} \to y\uparrow$$

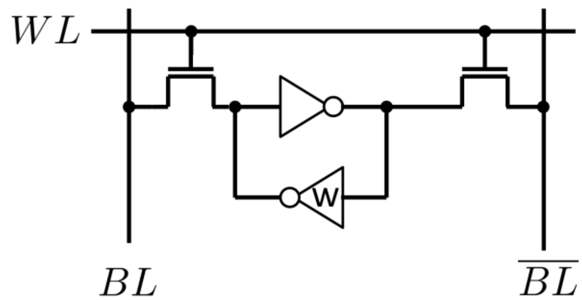


w: weak inverter that is overdriven by the p/n stack output.

State holding logic

Both used in memory design...

6T-SRAM cell

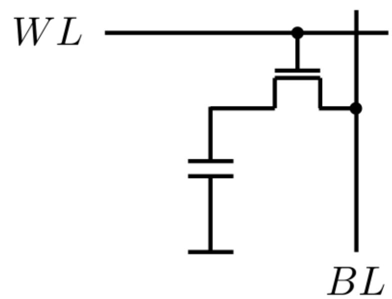


WL: word line, to select the cell
BL: bit line, read and write data port

State holding logic

Both used in memory design...

DRAM cell (refresh ~ 20Hz)



WL: word line, to select the cell
BL: bit line, read and write data port

V3 - State holding logic

[Hw]

$$\begin{aligned} G_{\text{down}} &\to y\downarrow \\ G_{\text{up}} &\to y\uparrow \end{aligned} \quad \text{->} \quad \begin{aligned} G_{\text{down}} \lor (\neg y \land \neg G_{\text{up}}) &\to y\downarrow \\ G_{\text{up}} \lor (y \land \neg G_{\text{down}}) &\to y\uparrow \end{aligned}$$

$$\text{->} \quad \begin{aligned} G_{\text{down}} \lor (\neg y \land \neg G_{\text{up}}) &\to y'\downarrow \\ G_{\text{up}} \lor (y \land \neg G_{\text{down}}) &\to y'\uparrow \\ \neg y' &\to y\downarrow \\ y' &\to y\uparrow \end{aligned}$$

first step: Boolean rewriting
second step: resolve the problem that it is forbidden for y to occur in the premise (see lecture on PRs)