EENG 4250/ENAS 8750
Introduction to VLSI System Design

Rajit Manohar

Computer Systems Lab

http://csl.yale.edu/~rajit

Fall 2025

Yale

Inverting logic

CMOS is said to be "inverting."

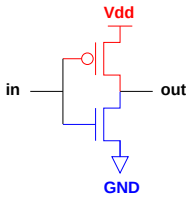
Given a one-stage circuit $\mathcal{C}$ computing

$$y = \mathcal{C}(\mathbf{x})$$

where **x** is a Boolean vector of inputs, we know that if any component of **x** changes from *false* to *true*, the output changes from *true* to *false* or remains unchanged.

Why?

Ratioless logic

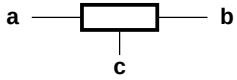


in	out
0	1
1	0

$R_{on} \approx 10^4\,\Omega$

$R_{off} \approx 10^9\,\Omega$

Switches



Switch is controlled by $V(c)$.

$$c \Rightarrow (a = b)$$

Computing with switches

Use the switches to set nodes to specific voltages.

The voltage could come from:

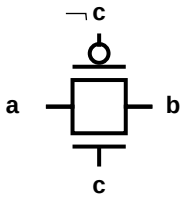
- Power supply
- Another node

Restoring logic switches power supplies.

Pass gate logic switches other nodes.

Yale

Switches in CMOS



A restoring inverter using switches?

NAND? NOR?

Yale

Pass gates

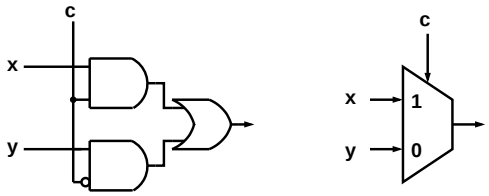
Pros and Cons:

- Can reduce transistor count
- Can reduce logic stages (non-inverting logic)
- Does not restore signals!
- Unintended "sneak" paths

Yale

Multiplexer

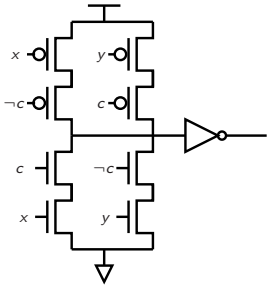
A 2-input MUX:



Terrible!

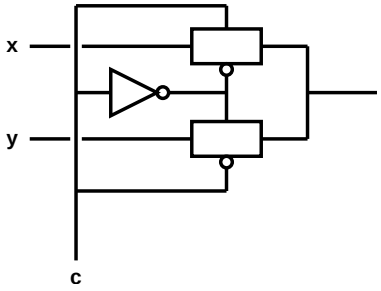
Yale

Restoring multiplexer



$(\neg x \wedge c) \vee (\neg y \wedge \neg c) \mapsto \_z\uparrow$
$(x \wedge c) \vee (y \wedge \neg c) \mapsto \_z\downarrow$

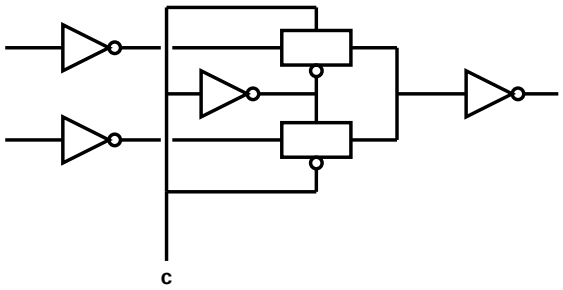
Pass gate MUX



Simple and fast, but it doesn't restore the quality of the logic signals.

Pass gate MUX

To restore the pass gate MUX signals:



Can we simplify this?

Simple functions: static CMOS

If we want to compute the Boolean function $f(\mathbf{x})$:

- Pull-up switching network: $f(\mathbf{x})$
- Pull-down switching network: $\neg f(\mathbf{x})$

The net result: the output is *always* connected to a power supply.

$f(\mathbf{x}) \mapsto z\uparrow$
$\neg f(\mathbf{x}) \mapsto z\downarrow$

Dynamic logic

We can implement arbitrary switching networks for pull-ups and pull-downs.

$B^+ \mapsto z\uparrow$
$B^- \mapsto z\downarrow$
(called *production rules*)

We require: $\neg(B^+ \wedge B^-)$ non-interference

If $\neg(\neg B^+ \wedge \neg B^-)$ combinational or static

If $B^+ \neq \neg B^-$ state-holding or dynamic

Yale

AVLSI