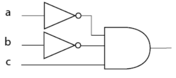
Chapter-3
Logic gates

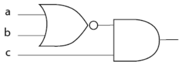
Gate design

Why designing gates for logic functions is non-trivial:

- may not have logic gates in the libray for all logic expressions;
- a logic expression may map into gates that consume a lot of area, delay, or power.



a'b'c



(a+b)'c

Two logic networks are equivalent but with different structures. Which is best depends on the requirements—the relative importance of area and delay—and the characteristics of the technology.

Gate design…

- We must work with both logic expressions and gate networks to find the best implementation of a function, *keeping in mind*:
- combinational logic expressions are the specification;
- logic gate networks are the implementation;
- area, delay, and power are the costs.

Completeness (*universality*)

- A set of functions *f1*, *f2*, ... is complete (universal) if every Boolean function can be generated by a combination of the functions.
- Complete (universal): NAND, NOR
- AND, OR: Not complete.

If your set of logic gates is not complete, you can't design arbitrary logic.

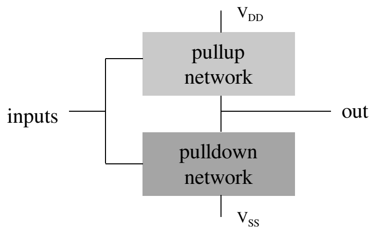
Static complementary gates

One family of logic gates circuits: the *static complementary gate*

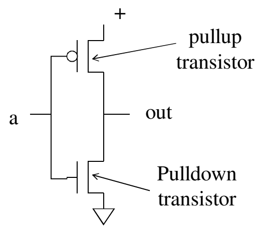
- *Complementary*: have complementary pullup (p-type) and pulldown (n-type) networks.
- *Static*: do not rely on stored charge.
- Simple, effective, reliable; hence ubiquitous.

Static complementary gate structure

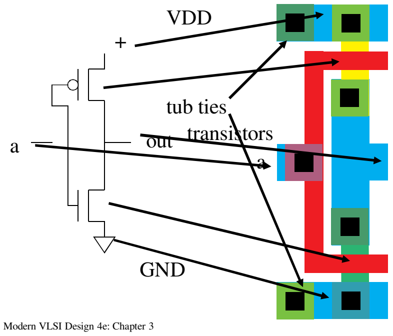
Pullup and pulldown networks:



Inverter

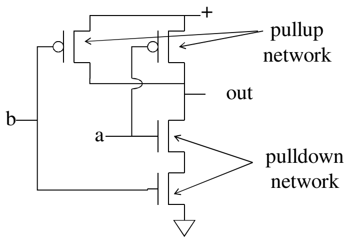


Inverter layout

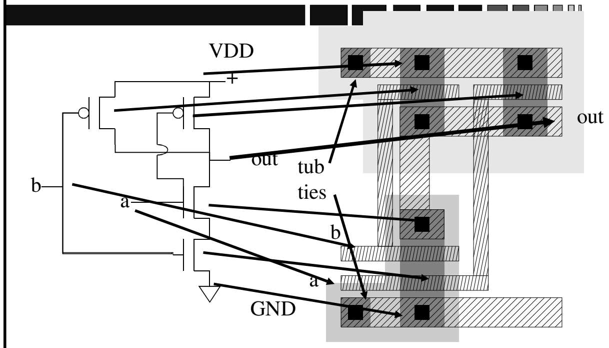


(tubs not shown)

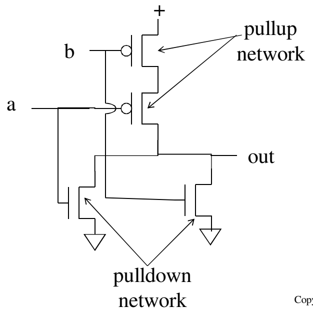
NAND gate



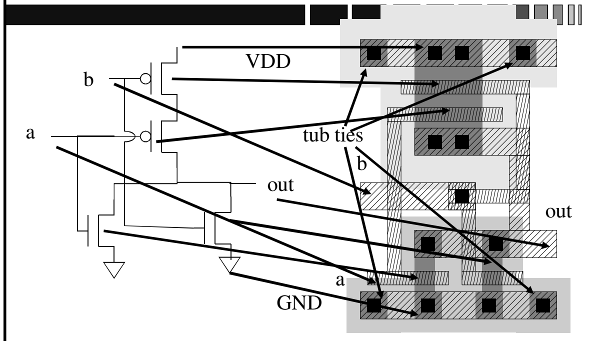
NAND layout



NOR gate



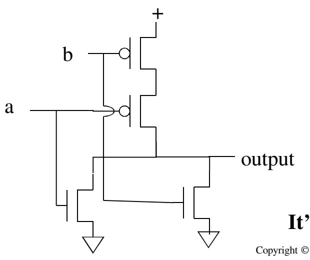
NOR layout



CMOS gate practice

- Design a gate that performs the following function

$(\overline{a+b})=(a+b)'$



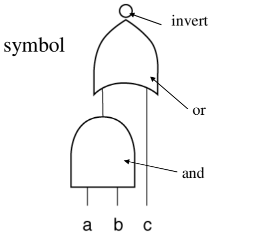
It's a NOR

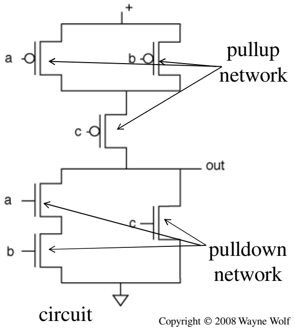
AOI/OAI gates

- AOI = and/or/invert; OAI = or/and/invert.
- Implement larger functions.
- Pullup and pulldown networks are compact: smaller area, higher speed than NAND/NOR network equivalents.
- AOI312: *AND* 3 inputs, *AND* 1 input (dummy), *AND* 2 inputs; *OR* together these terms; then invert.

AOI example

out = [ab+c]': AOI21

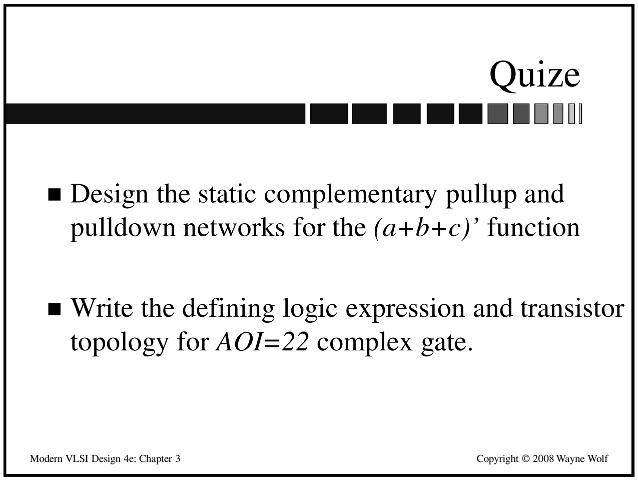




circuit

Pullup/pulldown network design

- Pullup and pulldown networks are duals.
- To design one gate, first design one network, then compute dual to get other network.
- Example: design network which pulls down when output should be 0, then find dual to get pullup network.



Quize

- Design the static complementary pullup and pulldown networks for the *(a+b+c)’* function

- Write the defining logic expression and transistor topology for *AOI=22* complex gate.