IN3170/4170, Spring 2023

Philipp Häfliger
hafliger@ifi.uio.no

Excerpt of Sedra/Smith Chapter 15: Digital gates basics

Content

A digital MOSFET model

Logic gates (book 15.1)

Inverter Gate Delay (book 15.4)

Transistor Sizing (book 15.5)

Content

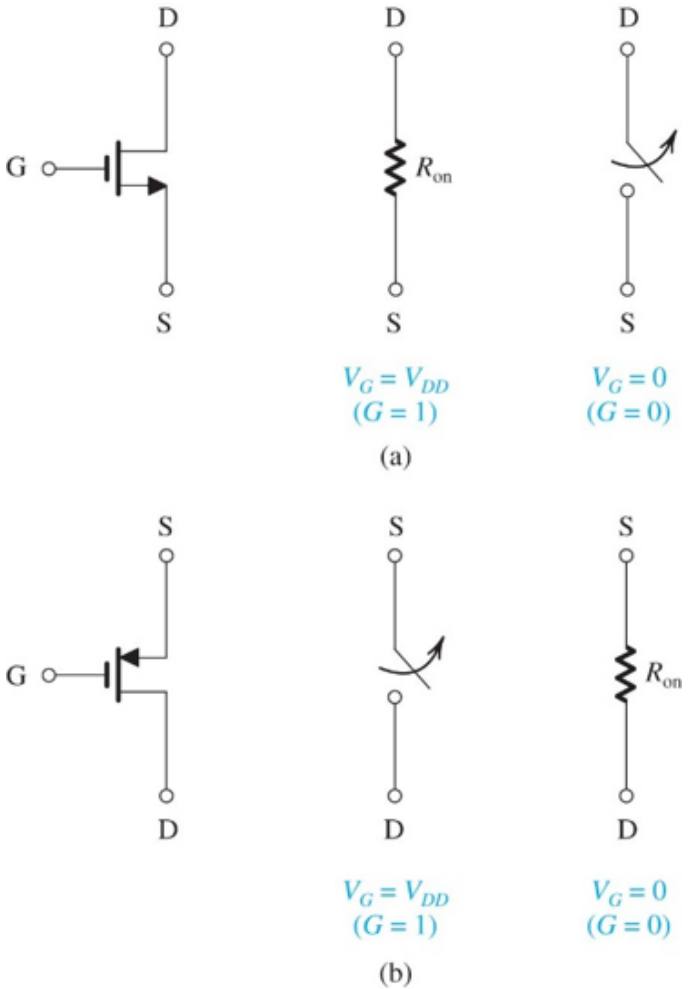
A digital MOSFET model

Logic gates (book 15.1)

Inverter Gate Delay (book 15.4)

Transistor Sizing (book 15.5)

Digital MOSFET Model Variant



Thus in the digital transistor abstraction both pFET and nFET have two conductive states controlled by the input voltage at the gate.

Content

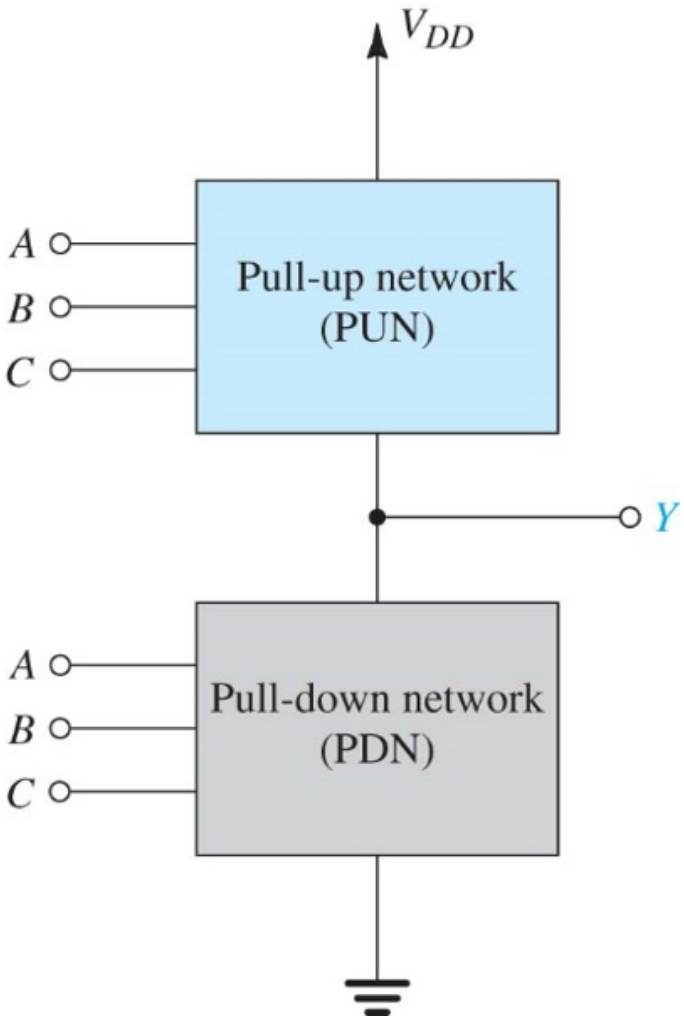
A digital MOSFET model

Logic gates (book 15.1)

Inverter Gate Delay (book 15.4)

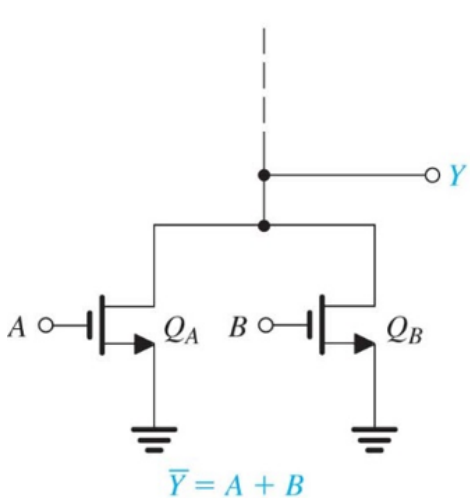
Transistor Sizing (book 15.5)

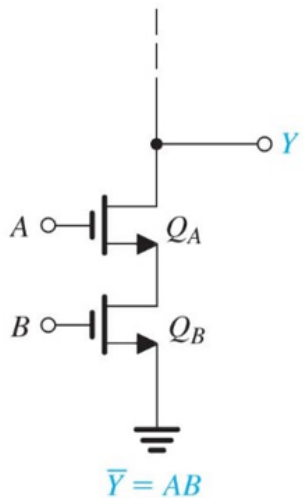
Pull-Up and Pull-Down

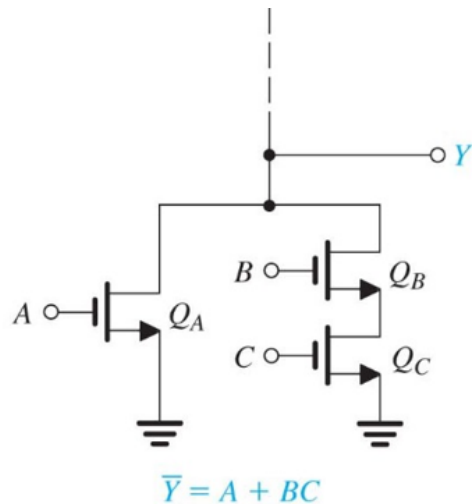


Logic functions can be implemented by complementary pull-up and pull down networks ... or you can just add a resistor (or single transistor with constant gate voltage) replacing the PUN or PDN... Would that be smart? Note: The PUN is composed of pFETs and the PDN of nFETs. Why?

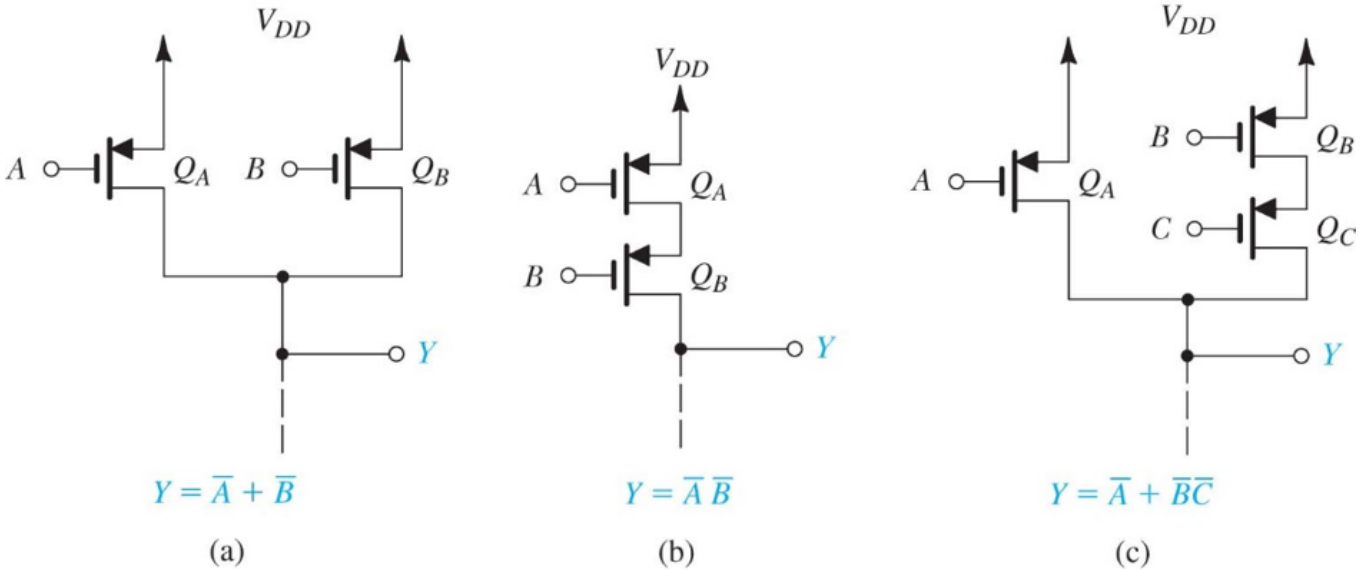
PDN examples







PUN examples



Complementary PUN and PDN (1/2)

Usually, logic gates are composed of a PUN and PDN that implement eachothers *inverse* function. Such that always either the PUN or the PDN are active.
E.g. the PDN expression:

$$\bar{Y} = (A + B) * C$$

You can directly implement PDN with active high input variables, since a high value opens a nFET...
The corresponding PUN expression:

$$Y = \overline{(A + B) * C} = ...$$

You can directly implement PUNs with active low variables, since a low value opens a pFET, so we need to modify that expression ...

Complementary PUN and PDN (2/2)

The PDN expression:

$$\bar{Y} = (A + B) * C$$

The corresponding PUN expression (rendered implementation friendly):

$$Y = \overline{(A + B) * C} = \overline{(A + B)} + \bar{C} = (\bar{A} * \bar{B}) + \bar{C}$$

Deriving complementary PUN/PDN

from PDN to PUN:

1. Derive Boolean function from circuit with nFETs in series meaning '$*$' and nFETs in parallel meaning '$+$'. (e.g.: $\bar{Y} = f(A, B)$)
2. Invert the function (e.g. $Y = \overline{f(A, B)}$)
3. Find an expression only based on active low variables (e.g. $Y = f(\bar{A}, \bar{B})$)
4. Derive circuit from expression as for PDN but with pFETs

Boolean Logic Rules

$a \wedge b \vee c = (a \wedge b) \vee c$	$a \vee b \wedge c = a \vee (b \wedge c)$	(priority)
$a \wedge b = b \wedge a$	$a \vee b = b \vee a$	(commutativity)
$(a \wedge b) \wedge c = a \wedge (b \wedge c)$	$(a \vee b) \vee c = a \vee (b \vee c)$	(associativity)
$\bar{\bar{a}} = a$		(involution)
$a \wedge \bar{a} = 0$	$a \vee \bar{a} = 1$	(completness)
$a \wedge a = a$	$a \vee a = a$	(idempotency)
$a \wedge 1 = a$	$a \vee 0 = a$	(boundedness)
$a \wedge 0 = 0$	$a \vee 1 = 1$	(boundedness)
$a \wedge (a \vee b) = a$	$a \vee (a \wedge b) = a$	(absorbtion)
$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$	$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$	(distributivity)
$\overline{a \vee b} = \bar{a} \wedge \bar{b}$	$\overline{a \wedge b} = \bar{a} \vee \bar{b}$	(deMorgan)

Most popular gates as building blocks

Two input NAND and NOR, and one input NOT/inverter (drawn live)

Content

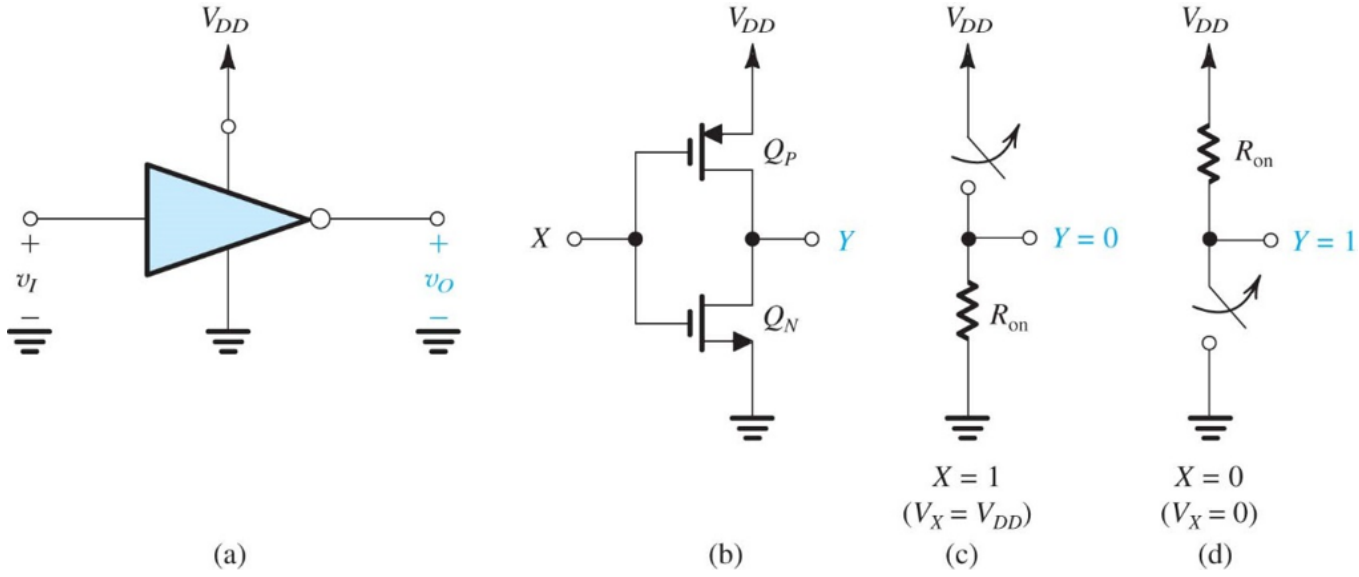
A digital MOSFET model

Logic gates (book 15.1)

Inverter Gate Delay (book 15.4)

Transistor Sizing (book 15.5)

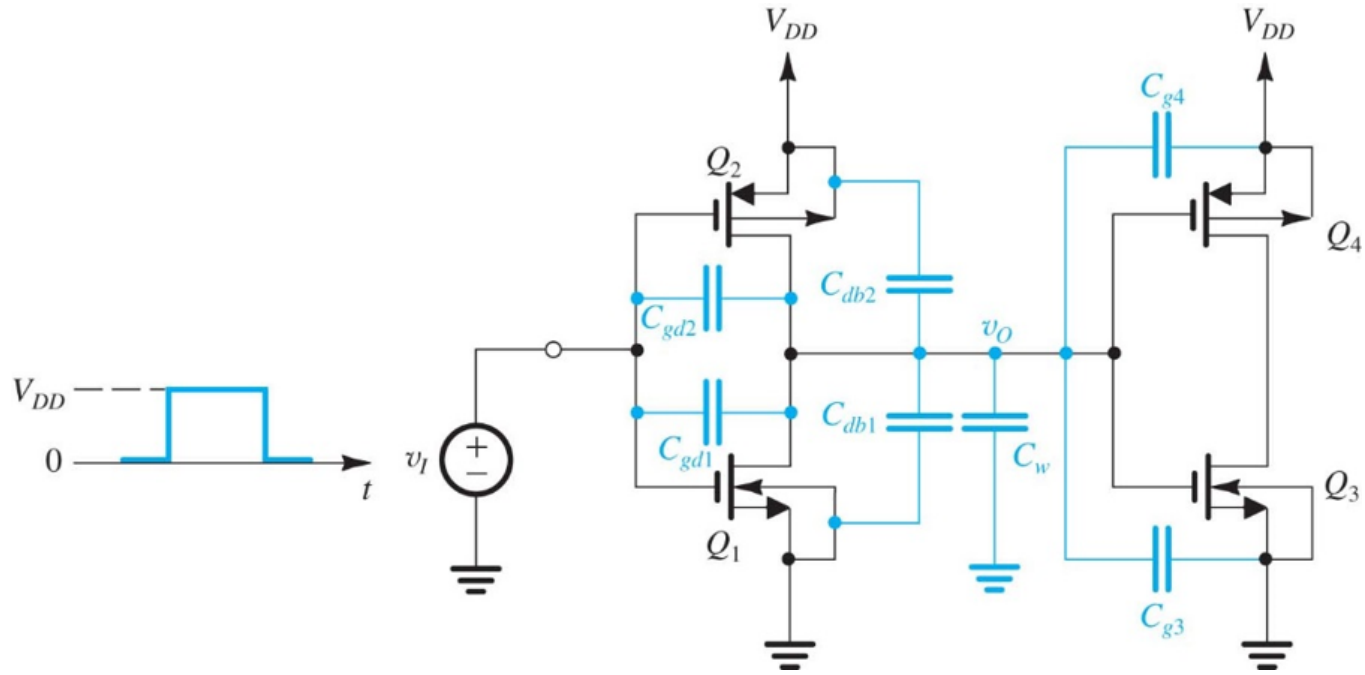
CMOS Inverter Model



Intermezzo: Charge Transfer via Capacitor and Miller Effect

(drawn live)

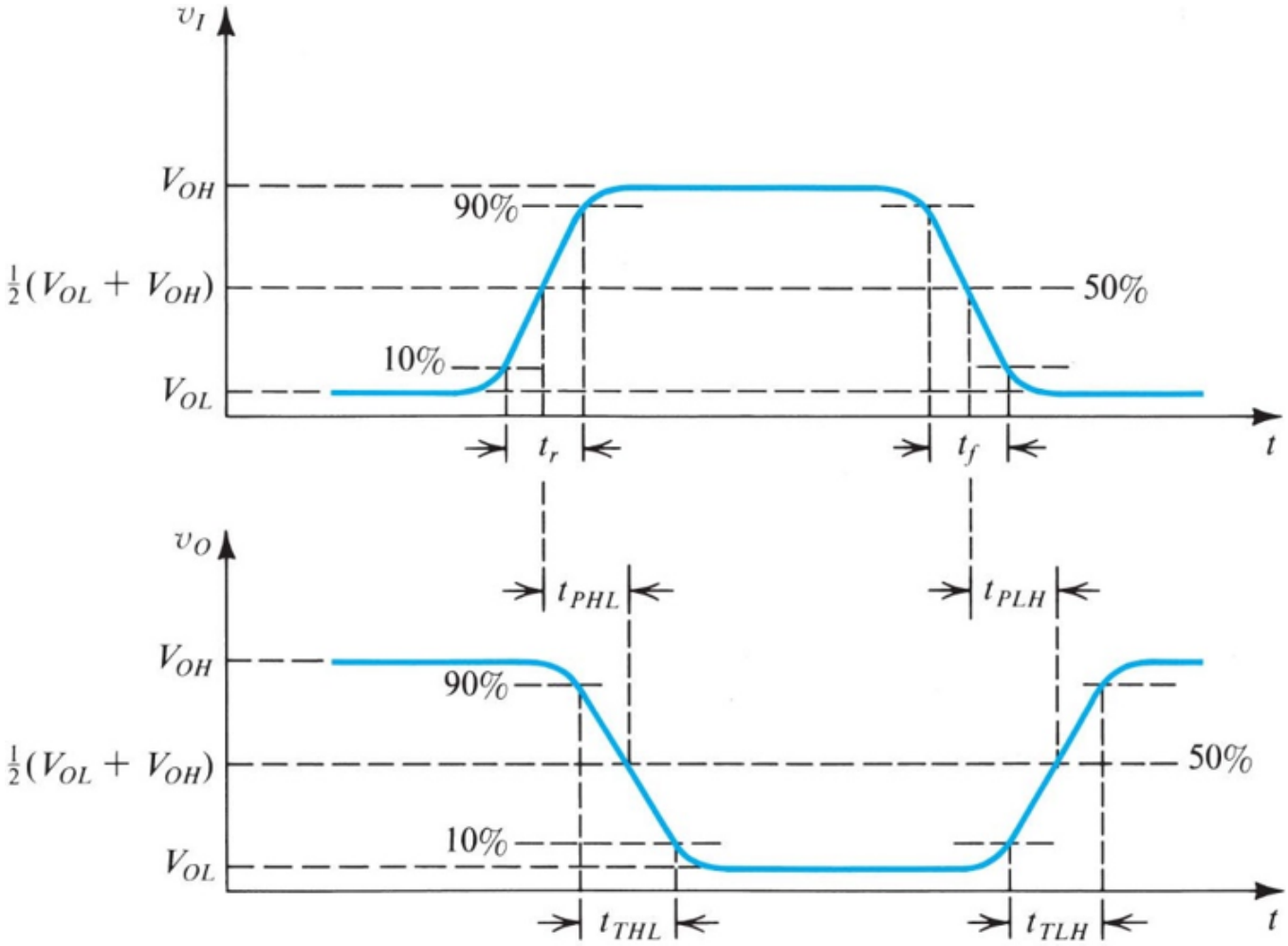
CMOS Inverter HF Model



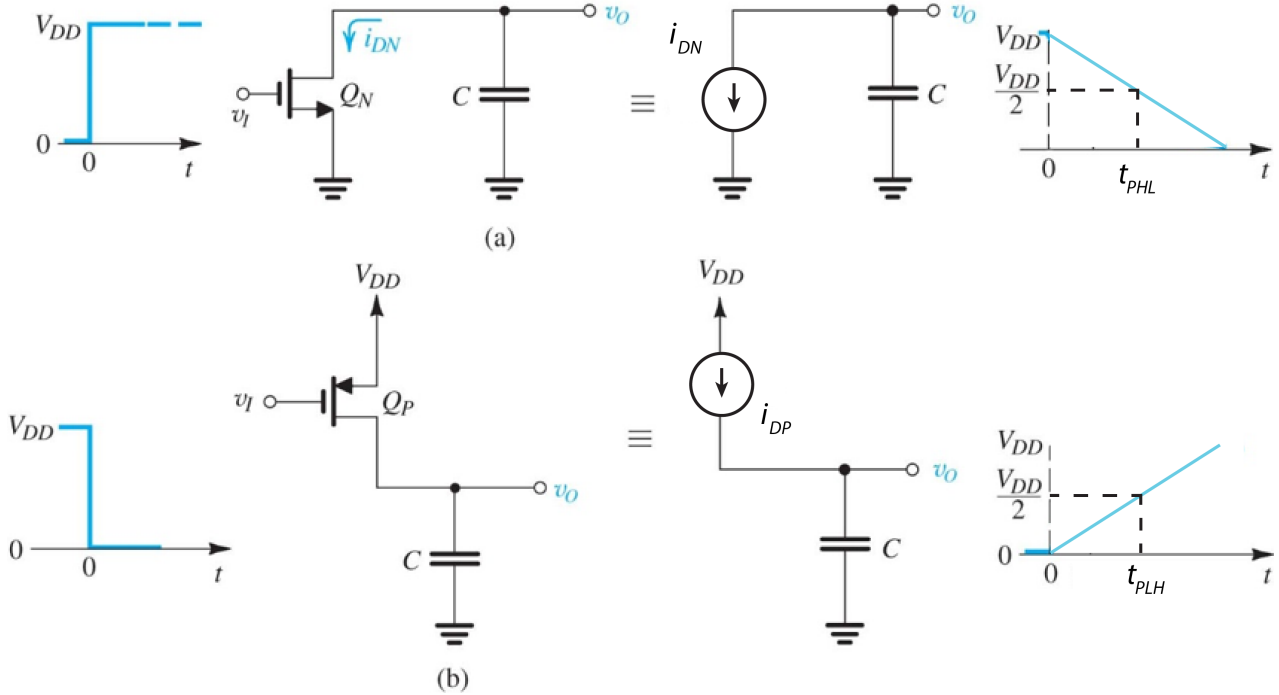
$$C = 2 * (C_{gd1} + C_{gd2}) + C_{db1} + C_{db2} + C_{g3} + C_{g4} + C_W \quad (15.59)$$

The term $2 * (C_{gd1} + C_{gd2})$ comes from 'kick forward' charge injection as the output moves in the opposite direction of the input. So as the input moves from the switching threshold to the rail the output needs to replace that charge in order to move from the opposite rail to the switching threshold.

Propagation delay and transition time



IC model for delay

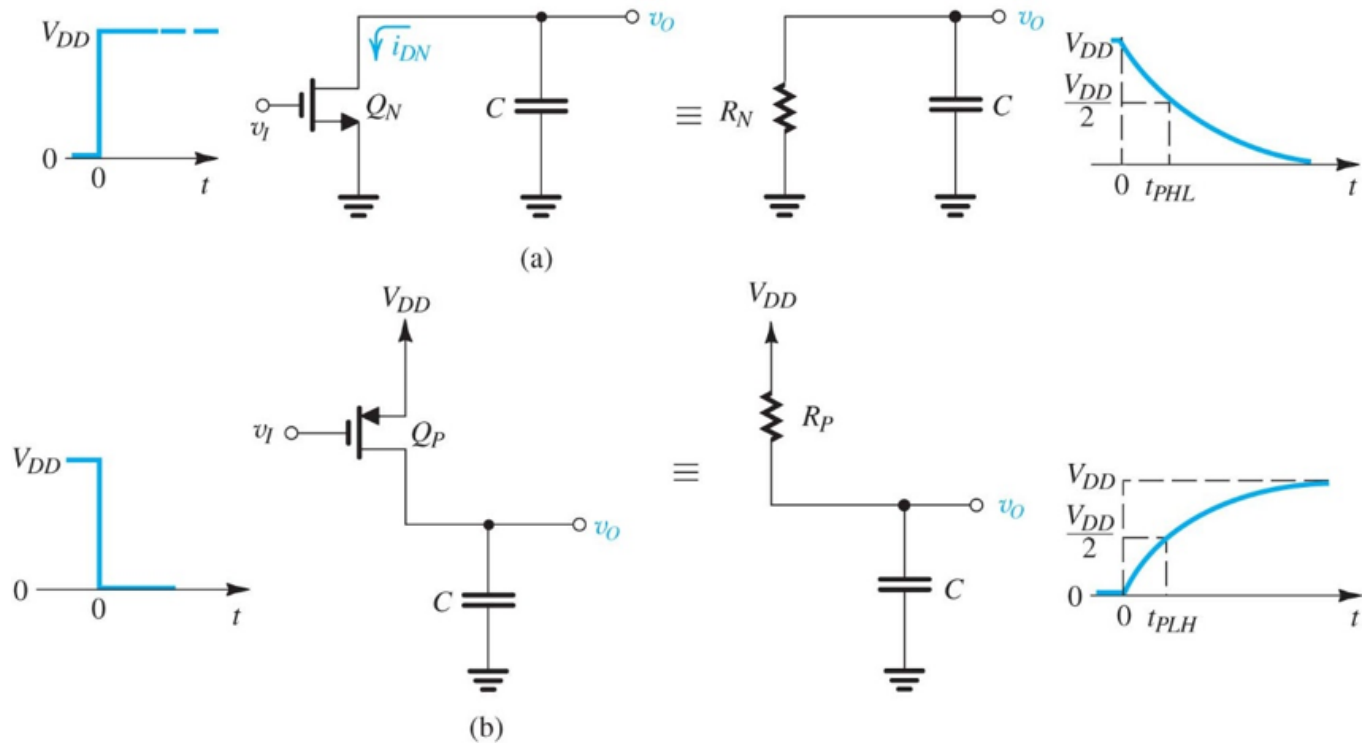


$$\hat{i}_{DN} = \frac{i_{DN}(\mathrm{Vdd}) + i_{DN}(\frac{\mathrm{Vdd}}{2})}{2} \quad (15.47)$$

$$t_{PHL(LH)} = \frac{C\mathrm{Vdd}}{2\hat{i}_{DN(DP)}} = \frac{\alpha_{n(p)}C}{k_{n(p)}\mathrm{Vdd}} \quad (15.46, 15.50)$$

$$\alpha_{n(p)}(Vdd, V_{tn(tp)}) = \frac{2}{\left[\frac{7}{4} - \frac{3V_{tn(tp)}}{Vdd} + \left(\frac{V_{tn(tp)}}{Vdd}\right)^2\right]} \quad (15.51, 15.53)$$

RC model for delay



$$R_{N(P)} = \frac{12.5(30)}{(W/L)_{n(p)}} \text{k}\Omega \quad (15.56)$$

$$t_{PHL(LH)} = 0.69 R_{n(p)} C \quad (15.54)$$

Content

A digital MOSFET model

Logic gates (book 15.1)

Inverter Gate Delay (book 15.4)

Transistor Sizing (book 15.5)

Observations

- Carrier mobility μ_n and thus k_n for nFETs is usually 2-3 times bigger than for pFETs.
- To balance t_{PHL} and t_{PLH} often pFETs are chosen to be wider than nFETs. (Because of noise margin considerations and for minimizing worst case propagation delay.)
- However, making pFETs wider sacrifices layout space, so often the open transistor conductance is not 100% compensated and the switching threshold is not 100% symmetric.
- In contrast to analog circuits, here maximizing speed (f_T) is prefered before maximizing gain (A_0), i.e. digital transistors are most often minimum length. However, small gain still has a negative impact on noise margin.
- These models are really crude and only give an idea of which parameters to tweak in which direction for an expected result. For reasonable absolute estimates one needs to consult transistor level simulation at least, or even post-layout simulation.

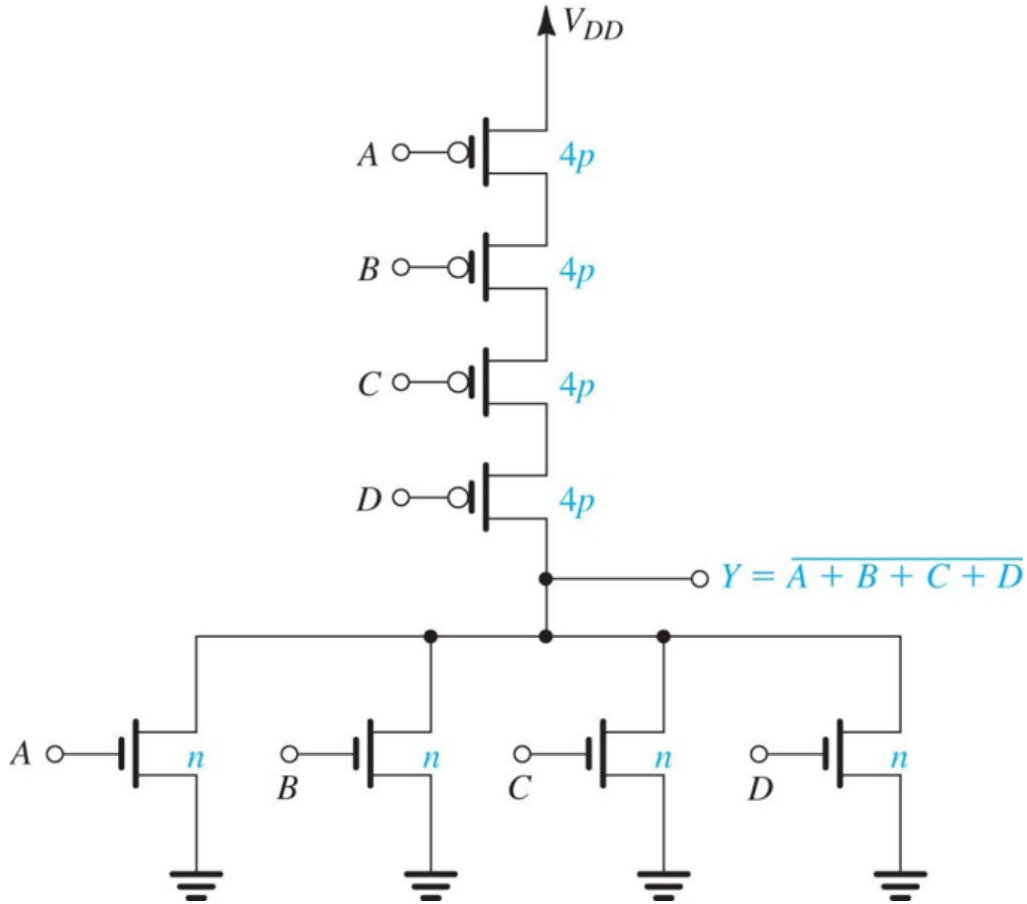
Wider transistors for speed?

In short: widening transistors by a factor S (trading in layout space) increases both the gate output drive ($R_{eq} = \frac{1}{2}(R_N + R_P)$ (15.62)) but also the gate load C. C is composed of (compare Fig. 15.32) contributions by a) the gate itself (in 15.5 refered to as C_{int}, b) the next gate's input capacitance, and c) the interconnect parasitic capacitances C_w or C_{ext}. a) and b) increase also by factor S, thus you 'only' reduce the effect of C_w. So this is only effective up to a certain point at which C_w becomes insignificant.

$$t_P \approx 0.69 \left(R_{eq} C_{int0} + \frac{1}{S} R_{eq0} C_{ext} \right) \quad (15.65)$$

Matching worst case output drive

The idea is to guarantee driving currents for multiple input gates at least equal to that of the inverter. That's to say for the input combination with the least output current. Since the on-conductance of a transistor is proportional to its W/L ratio, if a PUN or PDN has x transisors in series, those transistors need to have a W/L that is at least x times bigger than that of the basic inverter.

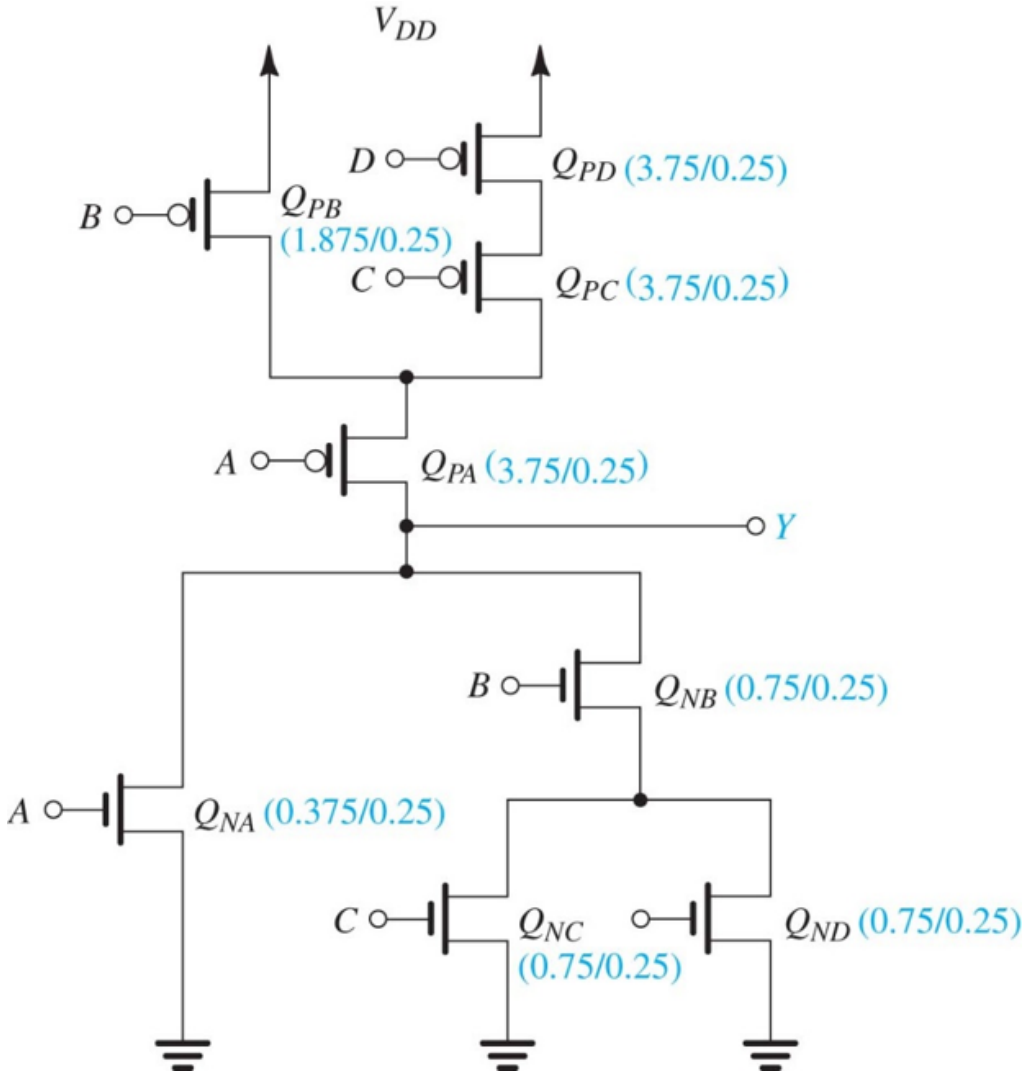


NOR gates, like the one above, get the worst out of this deal, having to increase the size of the PFETs that are already bigger. Thus, NAND gates are often prefered to implement combinational logic.

P.S: Large Gates vs Multiple Gates

NOTE: Exceedingly large PUN/PDN with the worst conducting case path involving a long series of transistors between Vdd/Gnd is dangerous. Rather compose logical function of multiple smaller gates in series.

A more complicated example

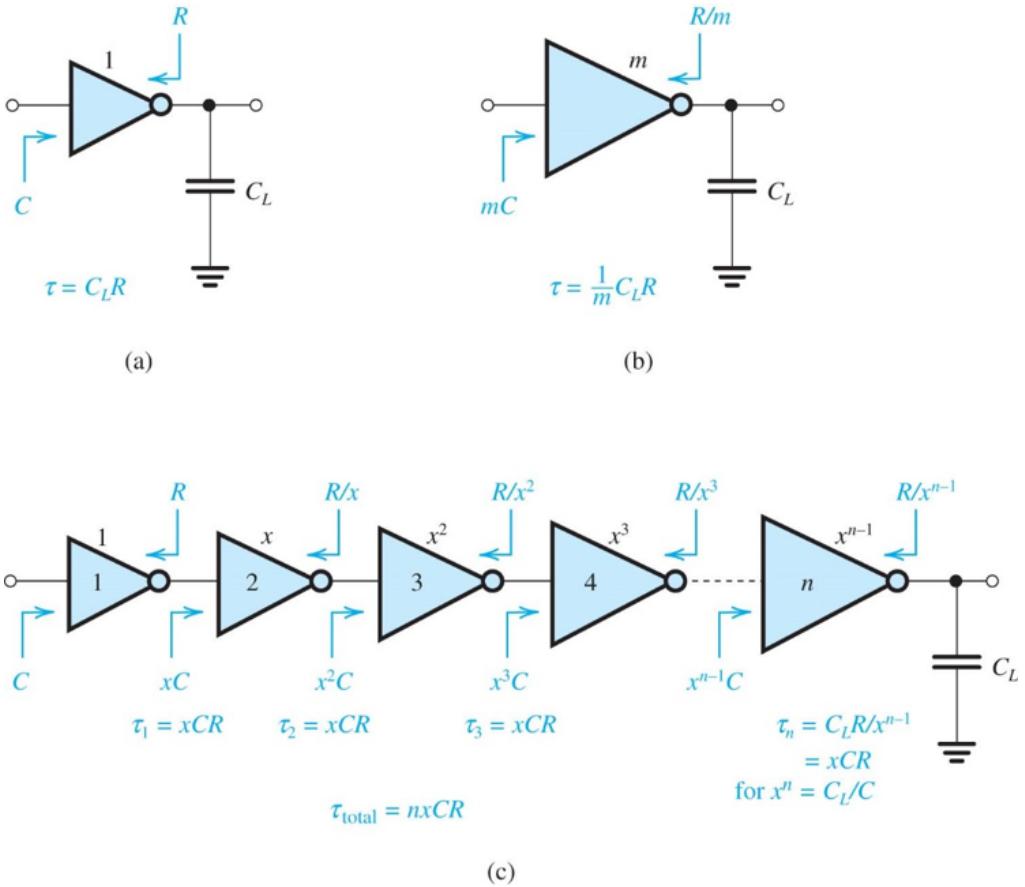


Optimizing Propagation Delay

Maintaining output drive is not enough to maintain the same propagation delay as in an inverter chain: A bigger fan-in increases the capacitive load at both input and output. Thus, a fan-in bigger than 4 is impractical (also for other reasons, such as noise margin). For bigger fan-in, rather use a tree-structure with multiple stages of gates, thereby achieving better propagation delay as well as layout space. (See problem 15.54 in the paper exercises).
Fan-out (the number of gates the output of a gate connects to) has also a even more detrimental effect on propagation delay, by linearly increasing the load.

Driving big loads

This may happen at the output terminals from an ASIC or also for internal bus-lines or simply long connections.



This is optimized by extending geometric increase by factor x all the way to C_L ($\Leftrightarrow x^n = \frac{C_L}{C}$ (15.70)) and by choosing $x = e = 2.718$ (15.72).

Often this is not followed precisely. Layout space might not allow for a too big n and C_L might also be unknown or variable, but a pragmatic choice of x and n will still yield an improvement. Often n is simply 2 or 4 and x quite a bit bigger than e.