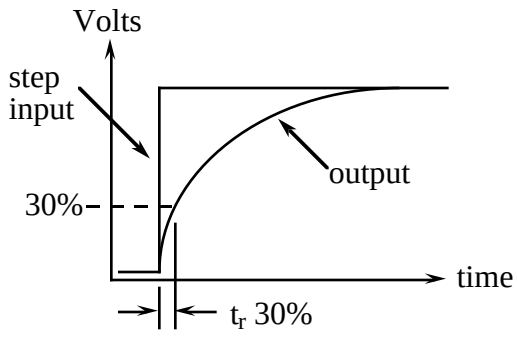
Modeling Delay

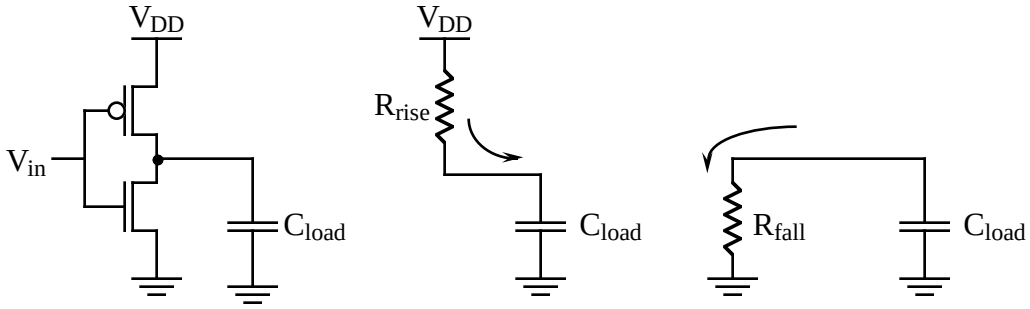
For a step input, then propagation delay simplifies to just rise/fall time of the output to a particular point (50%, 30%/70%, etc.).



Delay can be modeled in terms of an RC delay:

$$\tau_{rise} = R_{rise}\, C_{load} \qquad \text{and} \qquad \tau_{fall} = R_{fall}\, C_{load}\ ,$$

for a **particular** V_{DD}.



Effective Resistance is inversely proportional to ß of transistor.

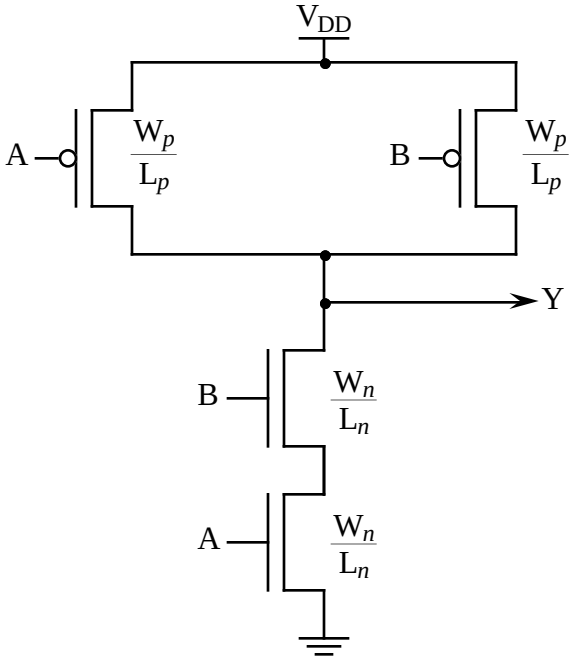
$$R_{rise} = k_{rise} \times \frac{1}{ß_p}$$

$$R_{fall} = k_{fall} \times \frac{1}{ß_n}$$

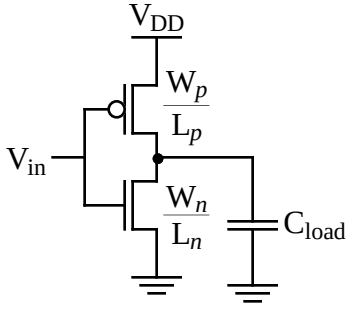
How do I determine k_{rise}, k_{fall}? Do SPICE simulation for a particular C_{load}, measure delay, solve for k_{rise}, k_{fall}.

These values of k_{rise}, k_{fall} would be valid for the **particular** V_{DD} you used in the simulations.

By characterizing an inverter this way, then one can predict delay for more complex gates after transforming the complex gates into an "equivalent" inverter. In this procedure, the original characterized inverter is sometimes called the "base" inverter.



In delay terms (for this NAND), for t_{rise}, would expect to have the same worst case t_{rise} as the base inverter because either A or B *p*MOS will pulling.



For t_{fall}, would expect NAND gate to be twice as slow (as the base inverter) because channel lengths add.

More accurate delay model breaks gate delay into two parts: $\tau_{internal}$, τ_{output}

$\tau_{internal}$ = gate delay with zero load (only internal capacitance values affect delay)

$\tau_{external}$ = portion of delay proportional to external load.

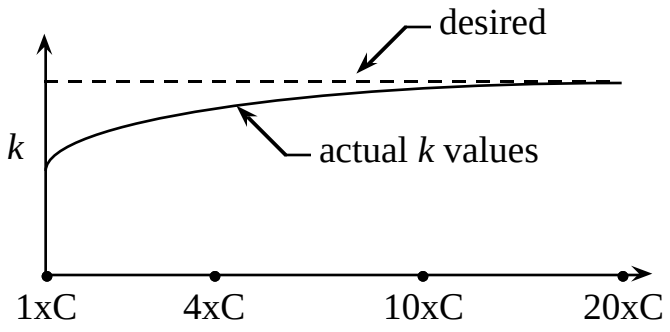
$$\tau_{gate} = \tau_{internal} + k \frac{C_{load}}{C_{unit\ load}}; \qquad C_{unit\ load} = C_{1x\ load}$$

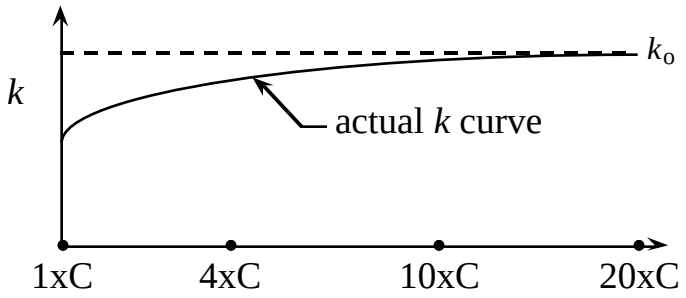
Make SPICE measurement at no load, get $\tau_{internal}$.

Make SPICE measurement at unit load (typical output load), determine *k* value.

Hopefully, *k* is a constant for different output loads, but may not be.

In this case, take SPICE measurements at different output loads and perform a curve fit of *k* against C values.





$$k = k_o - \rho[\exp(-C_{norm} \times \alpha)]$$

$\Rightarrow$ curve fit and get values for k_o, α, ρ

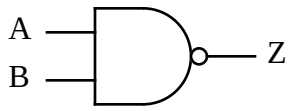
$$C_{norm} = \frac{C_{load}}{C_{1x\ load}}$$

Delay calculation:

a) compute k value based on C_{load}

b) compute delay value based on k

For a gate, need propagation delay factor for each input, both H $\rightarrow$ L and L $\rightarrow$ H



$T_{phl\_a\_to\_z}$, $T_{phl\_b\_to\_z}$

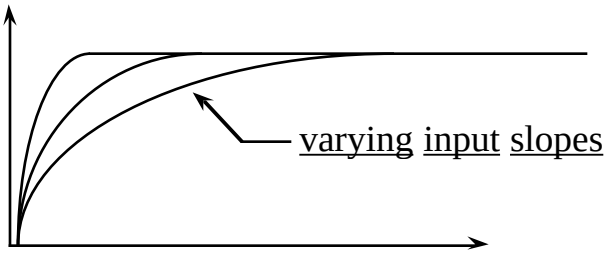
$T_{plh\_a\_to\_z}$, $T_{plh\_b\_to\_z}$

Would need k_o, α, ρ, and $\tau_{internal}$ parameters for <u>each one of these</u>

But wait a minute! . . . All of the previous discussion assumed a step input!

Is this realistic? **No**

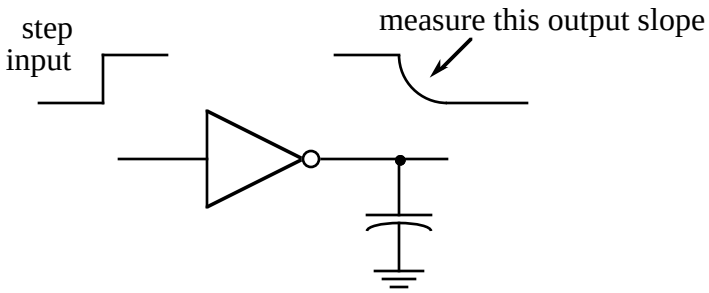
Actual waveforms in circuit look something like this:



How do I determine the range of input slopes I might see in a circuit?

⇒ Need to know fastest slope, slowest slope

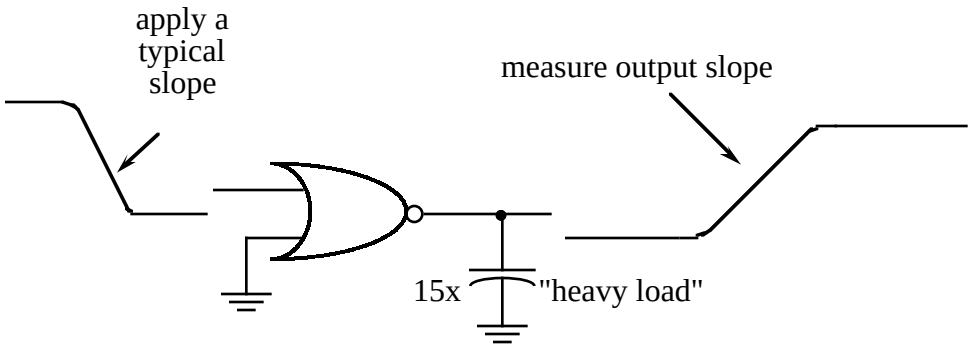
Fastest case would probably be for the inverter driving a 1x load, pulling down.



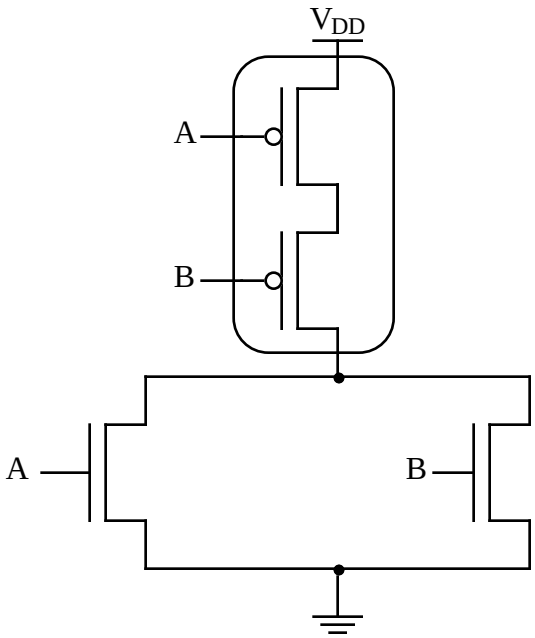
⇒ Measure the output slope. Call it your fastest slope.

⇒ Measure again for 4x load and call that your typical slope.

To get a representative "slow" slope, use a 2-input NOR gate pulling high



Why use a NOR gate?

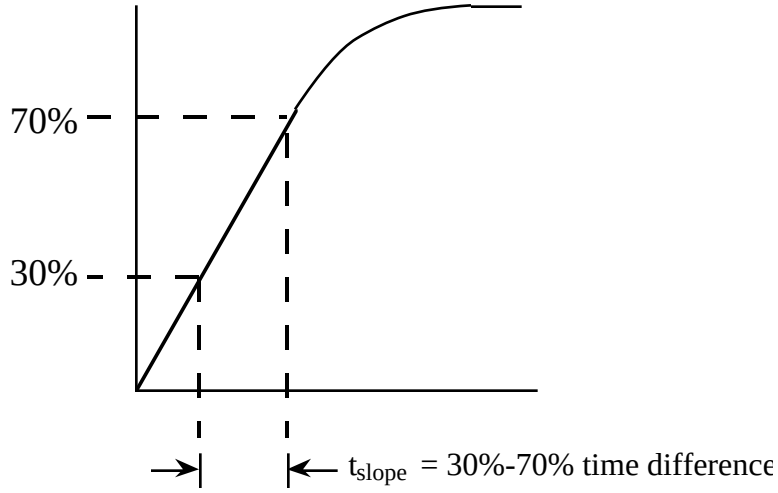


Two *p*MOS's in series will <u>be</u> <u>slow</u>.

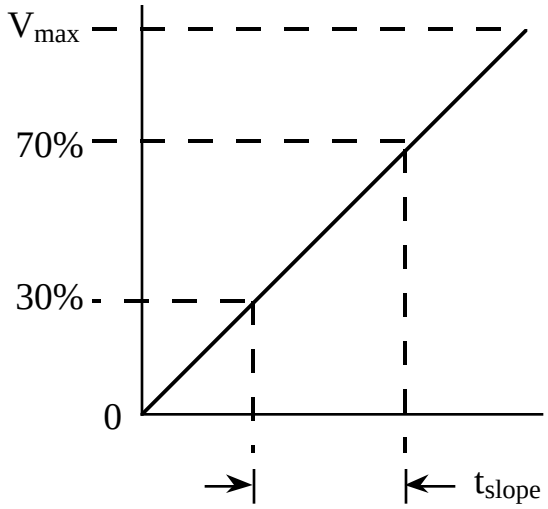
Now that you have a <u>fast</u> slope, and <u>slow</u> slope, pick values in between and generate tables of model parameters (k_o, α, ρ, $\tau_{internal}$)

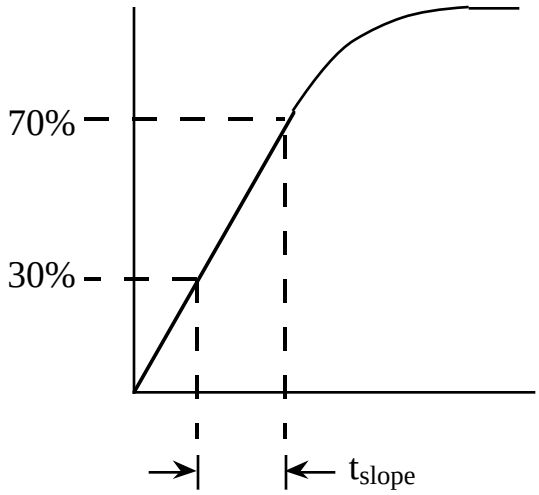
For different slopes values — do table lookup based on input slope value.

How do I define input slope? ⇒ Typically 30%-70% points



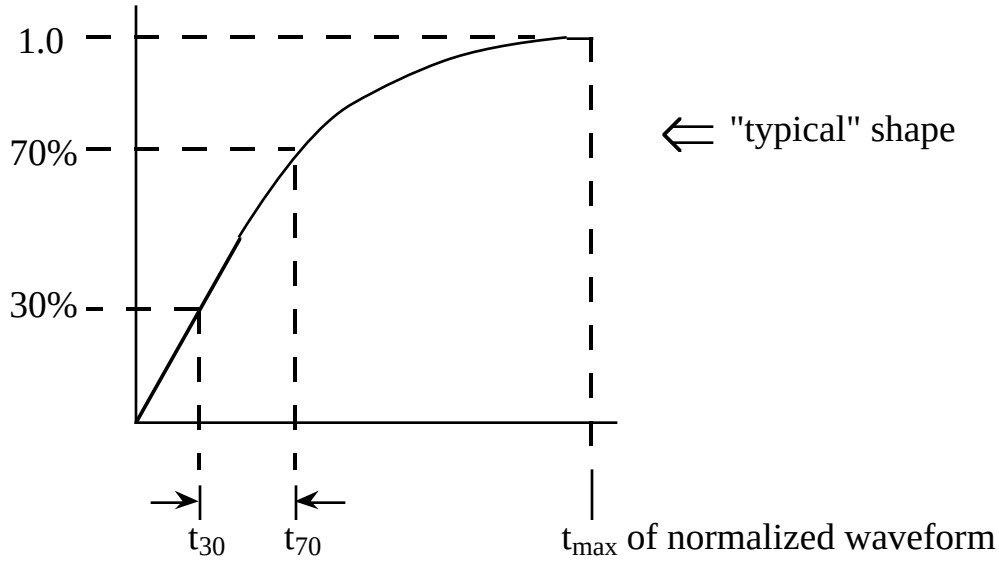
During characterization, I can apply a straight line input (as shown below left)





Not very realistic. Probably want to apply a more realistic waveform (above right).

Can have a <u>normalized</u> waveform:



Want to scale this to new waveform based on <u>new</u> $\mathbf{t_{slope}}$

Y_{values} scaled by V_{DD}.

$$\Rightarrow \text{new } Y_{value} = \text{old } Y_{value} \times V_{DD}$$

X_{value} (time) scaling???

$$t_{max} = (t_{70} - t_{30}) \times 2.5$$

$$t_{max\_new} = (t_{70\_new} - t_{30\_new}) \times 2.5$$

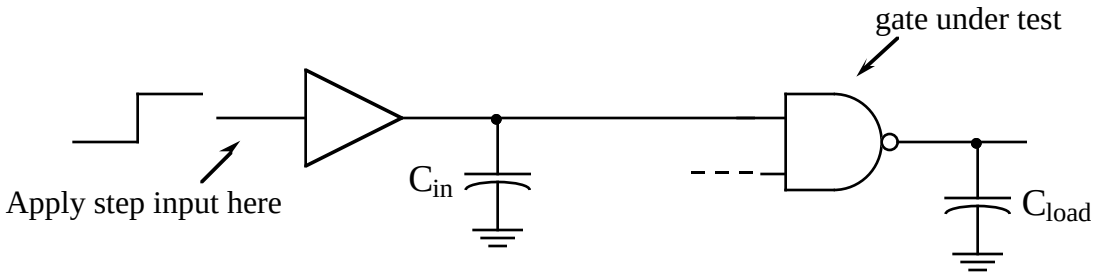
This last equation (above) implies that given t_{70_new} and t_{30_new}, *then* t_{max_new} can be computed.

Then

$$X_{new} = \frac{X_{old}}{t_{max\_old}} \times t_{max\_new}$$

Nice thing about previous method using a "scaled" waveform is that you can get a desired slope very easily.

Most realistic waveform is achieved however by having another gate drive the input:

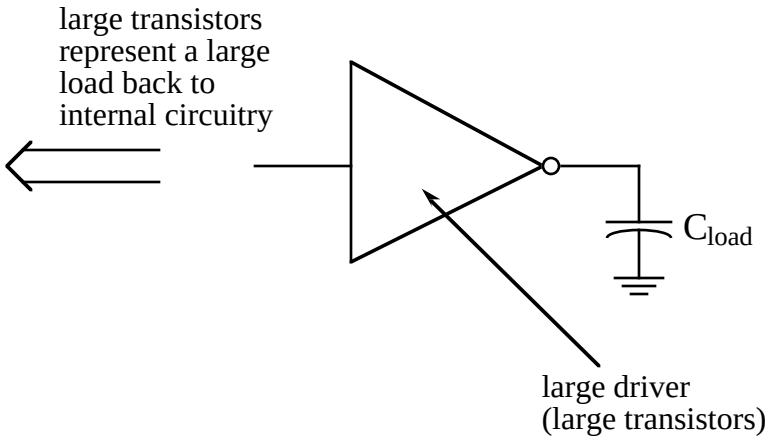


Vary C_{in} to control input slope. Only problem is that precise control of input slope is difficult, must be able to accurately predict slope of output gate based upon value of "C_{in}".

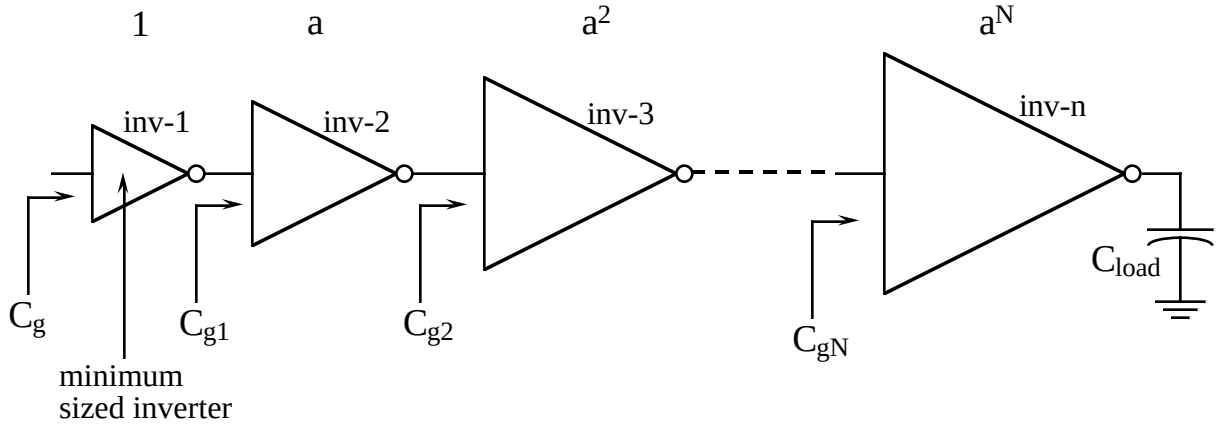
This will simulate realistic driving conditions.

Stage Ratio - Delay Optimization

To drive a large load, do not just want to make one large driver



Want to drive the load with a series of progressively larger drivers



n stages (number of inverters = n)

Each driver (inverter) larger than preceding driver by stage ratio "a".

Let C_g be gate load of first driver which is minimum size. Then, C_{gN} will be $C_g \times a^N$ and want

$$C_g\, a^n \le C_L\,, \qquad \text{[Note: } n = N + 1\text{]}$$

to guarantee that none of capacitances internal to the chain of inverters exceed C_{load}. For example, if $C_{gN} \ge C_{load}$, why we would need the n-th inverter at all!

So when the condition $C_g a^n \le C_L$ is set equal we have

$$a^n = \frac{C_L}{C_g} .$$

Question: What value of "a" will lead to minimum delay? What value of "n"? If we find one, we can compute the other.

Delay through each stage is approximately $a \times t_d$ where t_d is the delay through a minimum-sized inverter driving another minimum-sized inverter.

$$\text{Total Delay} = n \times a \times t_d$$

We know

$$a^n = \left(\frac{C_L}{C_g}\right),$$

so

$$a = \left(\frac{C_L}{C_g}\right)^{1/n}$$

Substituting,

$$\text{Total Delay} = n \left(\frac{C_L}{C_g}\right)^{1/n} t_d$$

To find optimum value for n, differentiate and set equal to zero.

If we do, then we find

$$n_{opt} = \ln\left(\frac{C_L}{C_g}\right)$$

Once we know n_{opt}, find a_{opt}

$$a^n = \left(\frac{C_L}{C_g}\right)$$

$$a^{\ln(C_L/C_g)} = \frac{C_L}{C_g}$$

Take the natural log (i.e., ln) of both sides:

$$\ln\left(\frac{C_L}{C_g}\right) \times \ln(a) = \ln\left(\frac{C_L}{C_g}\right)$$

$$\ln(a) = 1$$

$$\Rightarrow \qquad a = e^1 \approx 2.7 \quad \Leftarrow$$

A more detailed analysis shows that the intrinsic output capacitance of the inverter will affect this ratio.

$$a_{opt} = \exp\left(\frac{k + a_{opt}}{a_{opt}}\right)$$

where

$$k = \frac{C_{drain}}{C_{gate}} .$$

Page 190 of text computed

$$C_{drain} = 0.0043\text{pF}, \; C_{gate} = 0.02\text{pF for } 1\mu\text{m process}$$

$$k = \frac{C_{drain}}{C_{gate}} = 0.215$$

$$a_{opt} = 2.93$$