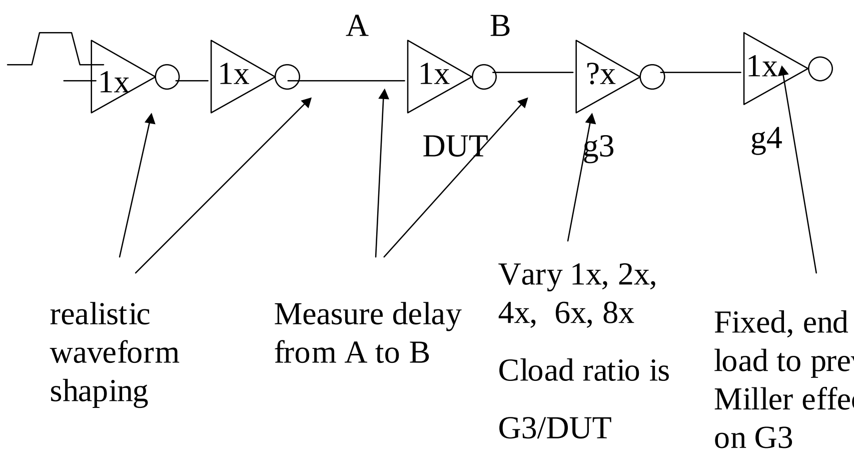
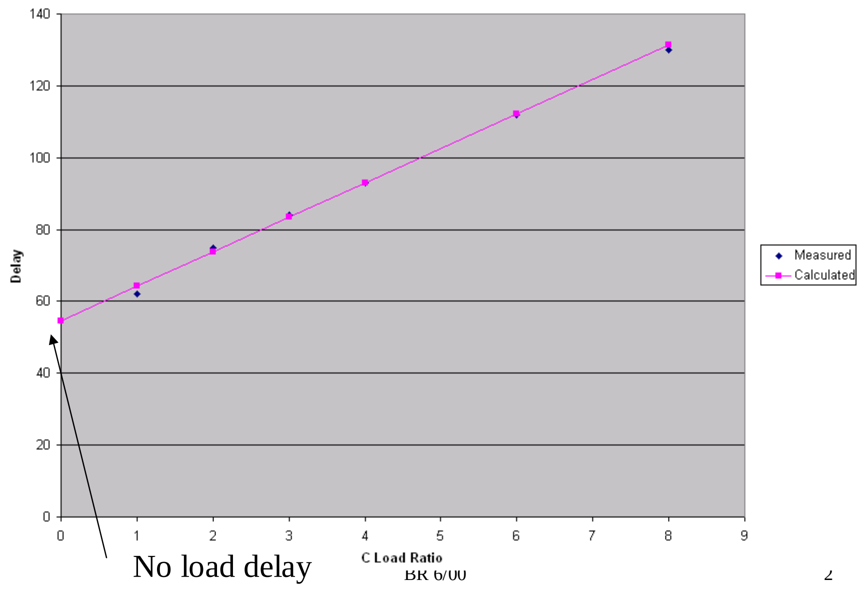
Logic Effort Revisited

A better way to measure P, Tau


A
B
1x
1x
1x
?x
1x
DUT
g3
g4
realistic waveform shaping
Measure delay from A to B
Vary 1x, 2x, 4x, 6x, 8x
Cload ratio is
G3/DUT
Fixed, end load to prevent Miller effect on G3

Plot Delay, Fit to Straight line (delay = mX + b)



Tau, Pinv

By definition, ginv = 1.0

From fitted line of mx + b, Tau can be calculated at any point as:

delay = tau (g*h + Pinv)
= tau * g *h + tau * Pinv

When X=0, delay = tau*Pinv = b (y-intercept).

So:

Tau = (delay_measured – b)/Cload

When Tau is known, can compute Pinv

Old vs New

Using Vdd = 2.5, Leda 0.25u

	Tau	Pinv	Pnand2	Pnand4
old	8.4	6.6	11	23
new	9.6	5.7	12	30

Differences mainly due to realistic waveshaping of inputs.

Sizing values in Decoder value not changed much.

Measuring Actual Logical Effort

When replace all gates in test circuit with Nand2, and plot:

$\text{delay} = M_{nand2} * X + B$

versus

$\text{delay} = M_{inv} * X + B$

the ratio of M_{nand2}/M_{inv} is the logical effort of the Nand2 since the Cload ratios are the same, and Tau is the same.

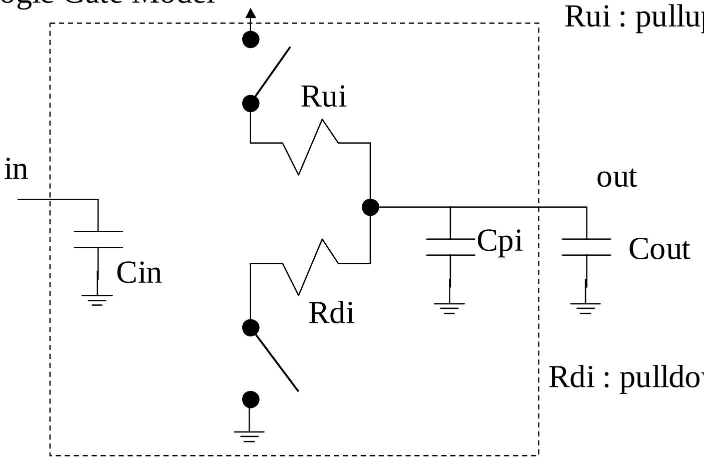
Logical Effort of Nand2, Nand4

	Nand2 g	Nand4 g
Book	1.33	2
Measured	1.6	2.2

Derivation of Logical Effort Equations

Logic Gate Model

Rui : pullup resistance



Rdi : pulldown resistance

Cpi: parasistic cap of gate

Tau

Tau (τ) is the absolute delay of a 1x inverter driving 1x inverter *with no parasitics.* We assume equal pullup/pulldown Rinv, and Cin = Cinv, so:

Tau = κ * Rinv * Cinv

where κ is a constant charateristic of the fabrication process that relates RC time constants to delay.

Note: Tau is NOT the no-load delay of an inverter. Also, it is not the delay of a 1x inverter driving a 1x inverter since this includes the parasitic delay!

Template Circuit

A template circuit is chosen as the basis upon which other gates are scaled. The scaling factor is α .

Ct is the input cap of the template.
Rt is the pullup or pulldown resistance of the template.
Cpt is the parasitic capacitance of the template.

$$Cin = \alpha * Ct$$

$$Ri = Rui = Rdi = Rt / \alpha$$

$$Cpi = \alpha * Cpt$$

RC Delay

$$Dabs = \kappa\ Ri\ (Cout + Cpi)$$

$$= \kappa\,(Rt/\alpha)\,Cin\,(Cout/Cin) + \kappa\,(Rt/\alpha)\,(\alpha\,Cpt)$$

$$= (\kappa\,Rt\,Ct)\,(Cout/Cin) + \kappa\,Rt\,Cpt$$

Written in this form, can see relation to logical effort model:

$$Dabs = \tau\,(gh + p)$$

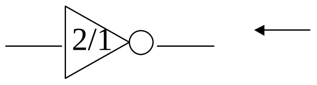
$\tau = \kappa\,Rinv\,Cinv$ (previous definition)

$g = (Rt\,Ct)/(Rinv\,Cinv)$ Note: if template = 1X inverter, then g = 1 !!!!

$h = Cout/ Cin$

$p = (Rt\,Cpt)/ (Rinv\,Cinv)$ Note: book value of Pinv = 1 only true if Cpt = Cinv!!

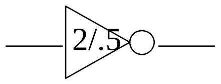
Logical Effort of Skewed Gates (unequal rise/fall delays)



Template gate, g = 1, equal rise/fall delays

The Leffort definition of a skewed gate is a gate that will produce the same output current for the critical transistion as the template gate, and produce less current for the non-critical transition.

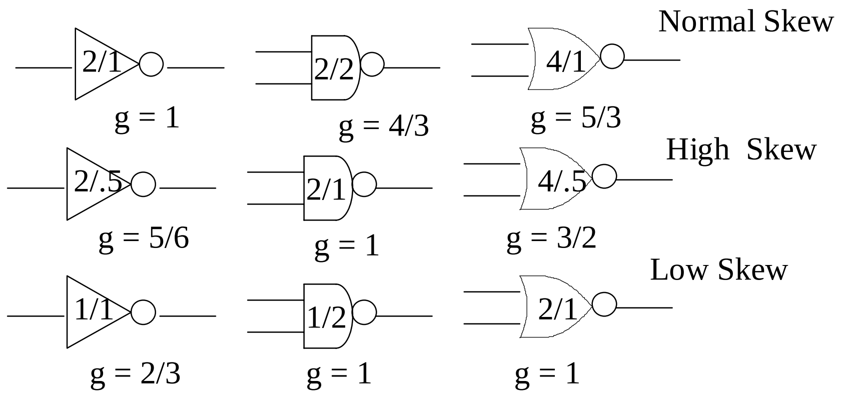
Assume that for a high skew gate, we reduce the non-critical current by ½ (.5). Then:



High skew gate (favors rising transition)
g (rising) = Cin / Cin(template)
= (2 + .5) / (2+1) = 5/6

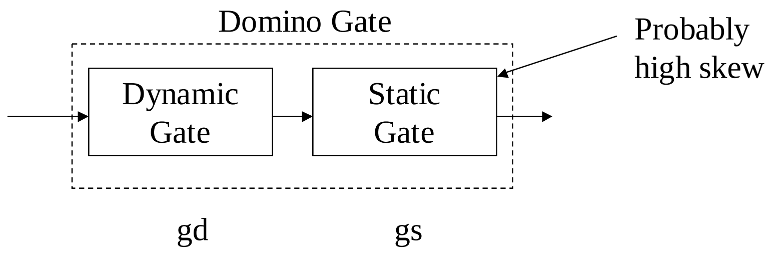
g (falling) will be ½ of critical, so 5/3

Skewed Gates



Logical efforts shown for critical transitions. Note that skewed gates always have lower logical efforts for the critical transition than the corresponding non-skewed version.

Logical Effort of Domino Gates

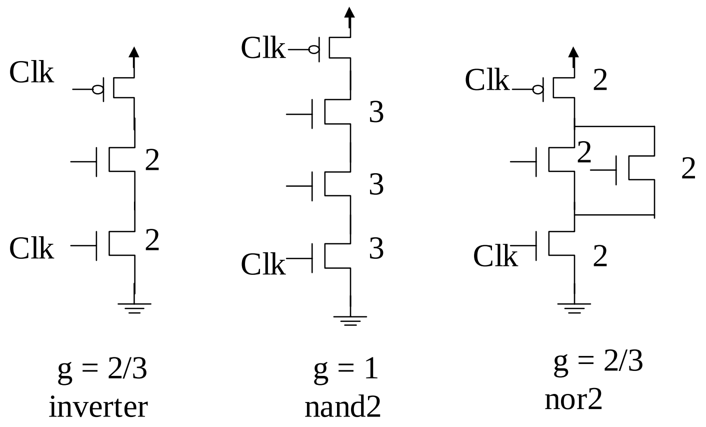


g (domino) = g(dynamic) * g(static)

Both dynamic and static portions enter into the stage count for sizing purposes (so a minimum 2-stage circuit)

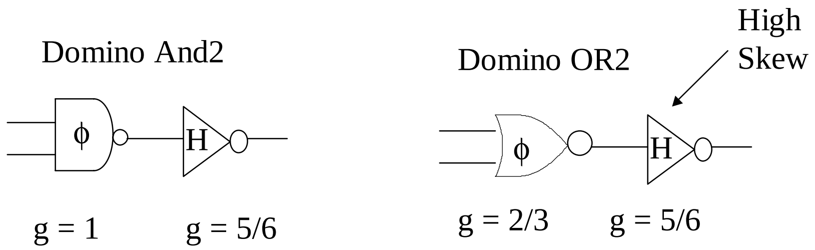
Logical Effort Calculation for Dynamic Gates

Only compute logical effort based on N-tree, size N-tree relative to Template gate, which is 2/1 inverter



Clock capacitance load does not count in logical effort

Domino Logic Effort



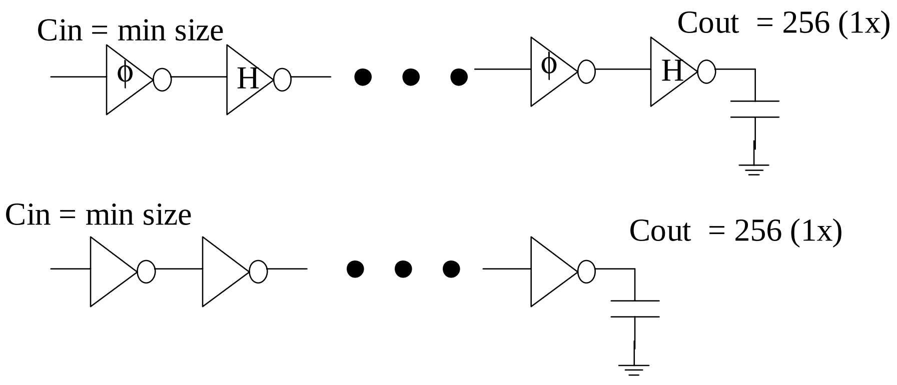
g(dom_and2) = 5/6

g(dom_or2) = 5/9

Note that logical efforts of dynamic gates less than static counterparts. This is because of less load on inputs. Also dynamic gates start switching at Vin= Vt since no P-tree.

A Sizing Problem

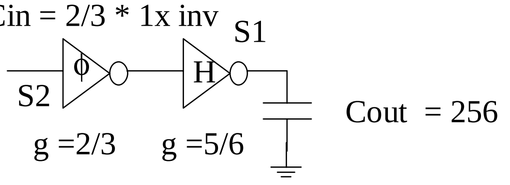
Determine Number of Stages to drive a load



Will not worry about inversion/non-inversion of signal

Dynamic Logic Sizing

Cin = 2/3 * 1x inv



G = 2/3 * 5/6 = 0.56, B = 1, H = 256/ (2/3) = 384
F = 0.56 * 1 * 384 = 215

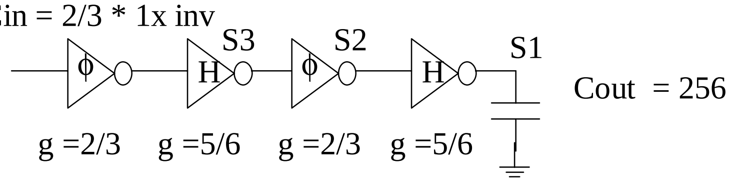
Fmin = $(215)^{1/2}$ = 14.6

S1 = g* 256/Fmin = (5/6) * 256/14.6 = 14.6
S2 = g * 14.6/Fmin = (2/3) *(14.6)/(14.6) = 2/3 (consistent)

Delay = 2 (14.6) + P_invH + P_dynamic
= 29.2 + P_invH + P_dynamic

Dynamic Logic Sizing (cont).

Cin = 2/3 * 1x inv



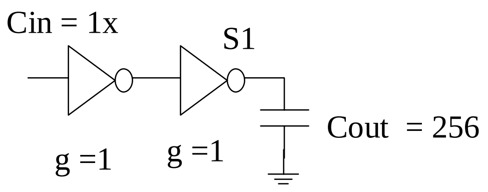
H = 256/ (2/3) = 384

G = $(2/3 * 5/6)^2$ = $(0.56)^2$ = .31

F = 0.31 * 1 * 384 = 119

Fmin = $(119)^{1/4}$ = 3.3
S1 = g * 256/ 3.3 = (5/6) * 256 / 3.3 = 64.6
S2 = g * 64.6/3.3 = (2/3) * 64.6/3.3 = 13
S3 = g * 13/3.3 = (5/6) * 13/3.3 = 3.3
S4 = g * 3.3/3.3 = (2/3) * 1 = 2/3 (consistent)
Delay = 4 * (3.3) + 2*P_invH + 2*P_dynamic
= 13.3 + 2 *P_invH + 2*P_dynamic

Static Logic Sizing

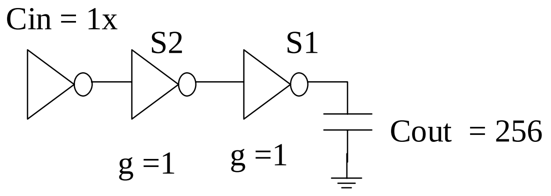


G = 1, H = 256, B = 1
Fmin = $(256)^{1/2}$ = 16

S1 = 256/16 = 16

S2 = 16/16 = 1 (consistent)

Static Logic Sizing (cont)



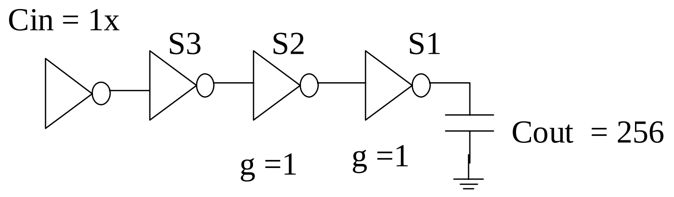
G = 1, H = 256, B = 1
Fmin = $(256)^{1/3}$ = 6.3

S1 = 256/6.3 = 40.6

S2 = 40.6/6.3 = 6.3

S3 = 6.3/6.3 = 1 (consistent)

Static Logic Sizing (cont)



G = 1, H = 256, B = 1

Fmin = $(256)^{1/4}$ = 4

S1 = 256/4 = 64

S2 = 64/4= 16

S3 = 16/4 = 4

S4 = 4/4 = 1 (consistent)

Results

Dynamic	Meas (Tplh)	Predicted
Two Stage	518.0	392.8
Four Stage	488.0	351.7

Static	Meas (Tavg)	Predicted
two stage	634.0	416.6
Three Stage	547.0	345.6
Four Stage	562.0	372.5

Leffort correctly predicted relative magnitudes. Leffort does not capture all of the speedup that will occur in a dynamic implementation. Dynamic implementation benefits from more stages than static.

Optimium Number of Stages?

- What is the optimum number of stages? (Nopt)
- Depends on the relative magnitude of the parasitic gate delays to the delay due to the stage effort
  - Why? Because as you add more stages, you add a fixed parasitic delay, and the delay via each stage effort gets smaller. Larger parasitic delays means Nopt is smaller!
- Authors of logical effort model derive equations that relate parasitic delay to N optimum
  - We will not cover this, since when talking about *absolute delay*, also need to account for slope dependence of delay (which Leffort does not do).
- Practical approach is simply to try a few different N values (will usually not be that many choices), and see what is best. Use Leffort to size gates for a particular N

Optimium Number of Stages? (cont).

- Rule of Thumb from Leffort authors which probably is true:

  The optimum stage effort found for an string of inverters driving a load will be close to the optimum stage effort for general logic driving a load. (assuming no significant off-path load).

- For our static inverter design, we found that a stage effort of about 6 is best. So, for a general logic problem, pick number of stages such that stage effort is about 6 and this will probably be the best number of stages.
- For static logic, the book recommends 4 as the optimum stage effort, but they assume Pinv is between 0.7 and 2.5. Our Pinv is about 6, so higher parasitics means higher stage effort, which will result in less stages.

Optimum Number of Stages? (cont.)

- For our example, the optimum stage effort for static logic was about 6, for dynamic logic was about 3.5.
- This means that to drive the same load, dynamic logic will benefit from more stages than static logic
- If the number of stages chosen is non-optimum by +/- 1, will not significantly affect the delay.
- Other considerations such as Power, Area will definitely affect the choice of the number of stages.