Path delay optizimization
Review logical effort
Prelab 2

Extending inverter delay optimization to other types of gates

Week 3

- Monday lab1
  - CMOS inverter static and dynamic
- Tuesday
  - Lecture Delay with gates, logical effort
  - Postlab review lab 1
- Thursday
  - Prelab lab 2, Lecture on optimal path delay
  - Tutorial POTW (Victor)
- Friday Deadline prelab 2
  - Schematic entry of carry ckt of full adder

From MUD cards

- Too many efforts!
- Inputs to the gate & logical effort
  - How to calculate g, and, p, for different inputs.
  - Not very clear on the input capacitance C_{IN}. What if we have more than one input, which one should we choose?
- Resistance and scaling
  - Worst-case R for parallel and series connection.
  - How to decide nMOS and pMOS widths during calculation of logical effort.
  - For bigger schematics how to find worst-case scenario.
- The constant part
  - How to calculate C_{par} for compound network..

Too many efforts

$$\underset{\text{Stage effort or Effort delay}}{f} = \underset{\text{Logical effort}}{g} \times \underset{\text{Electrical effort}}{h}$$

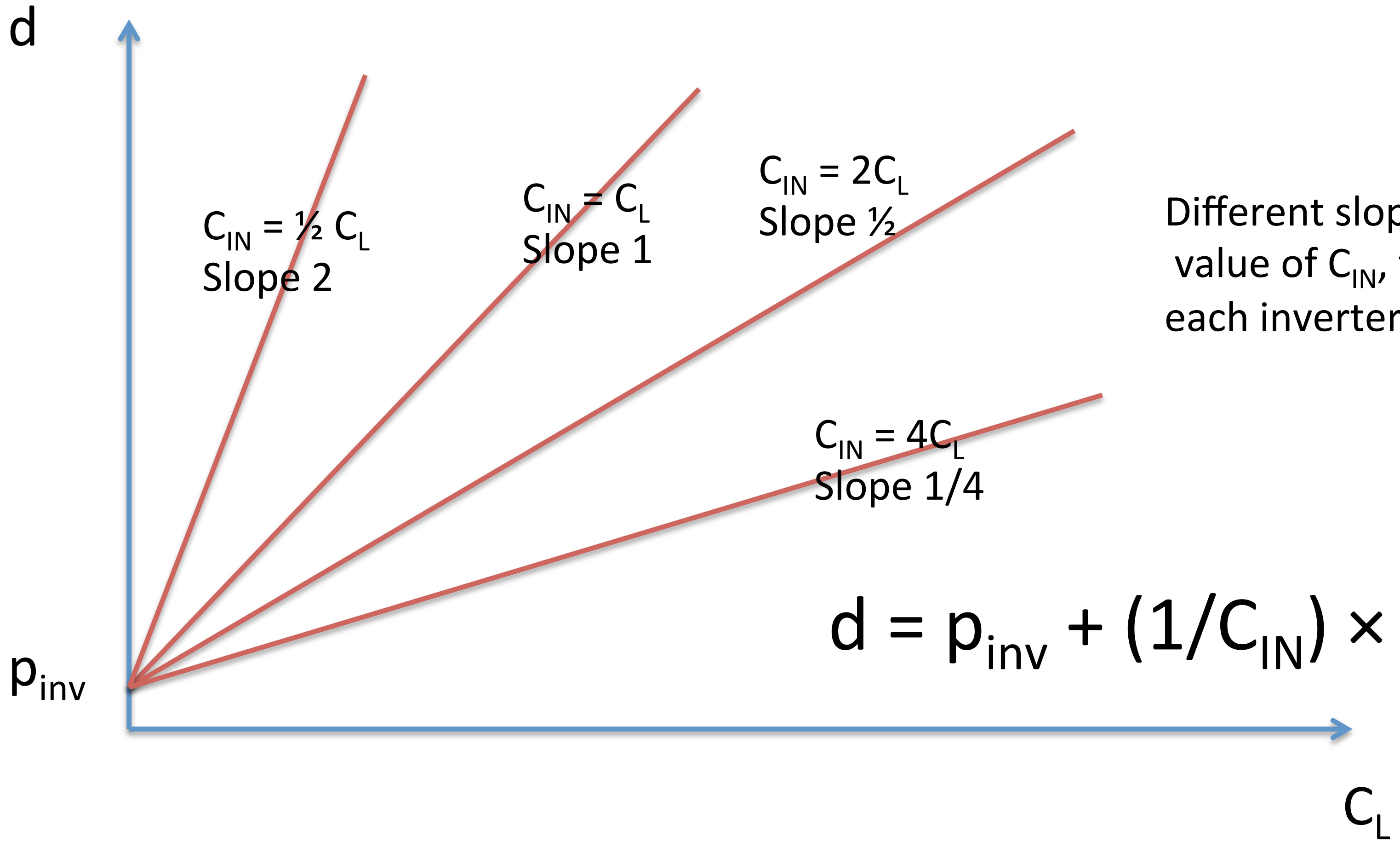
Only three efforts - all in the delay that depends on the external load C_L!
Two of them combined make up the third one.
"h" is our variable, our "x", as we will see later in this lecture.

And then there is a constant part, p, the parasitic delay

$$d = g \times h + p$$

Compare to standard linear equation: $y = k \times x + m$

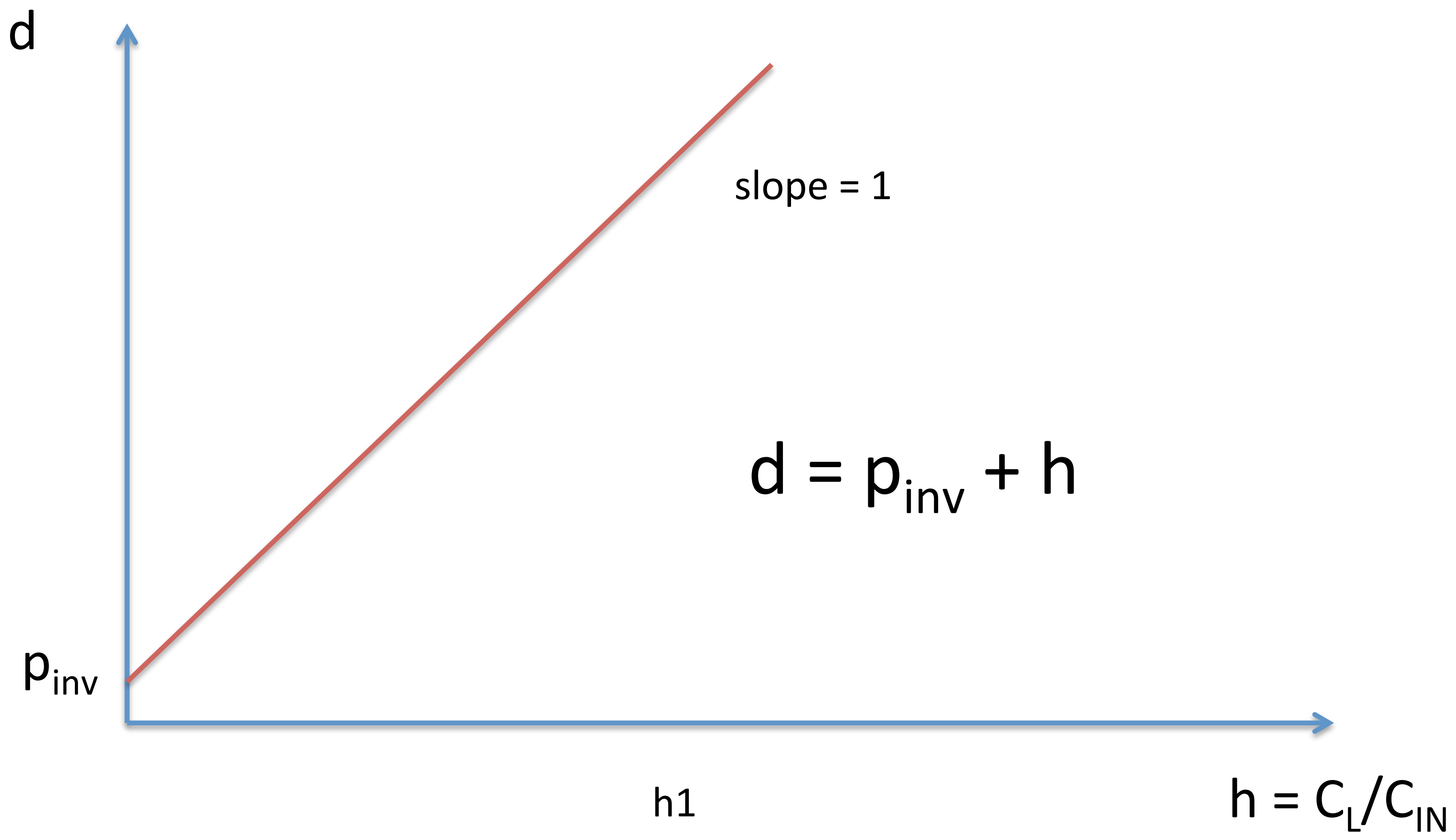
Graph of normalized inverter delay as a function of C_L



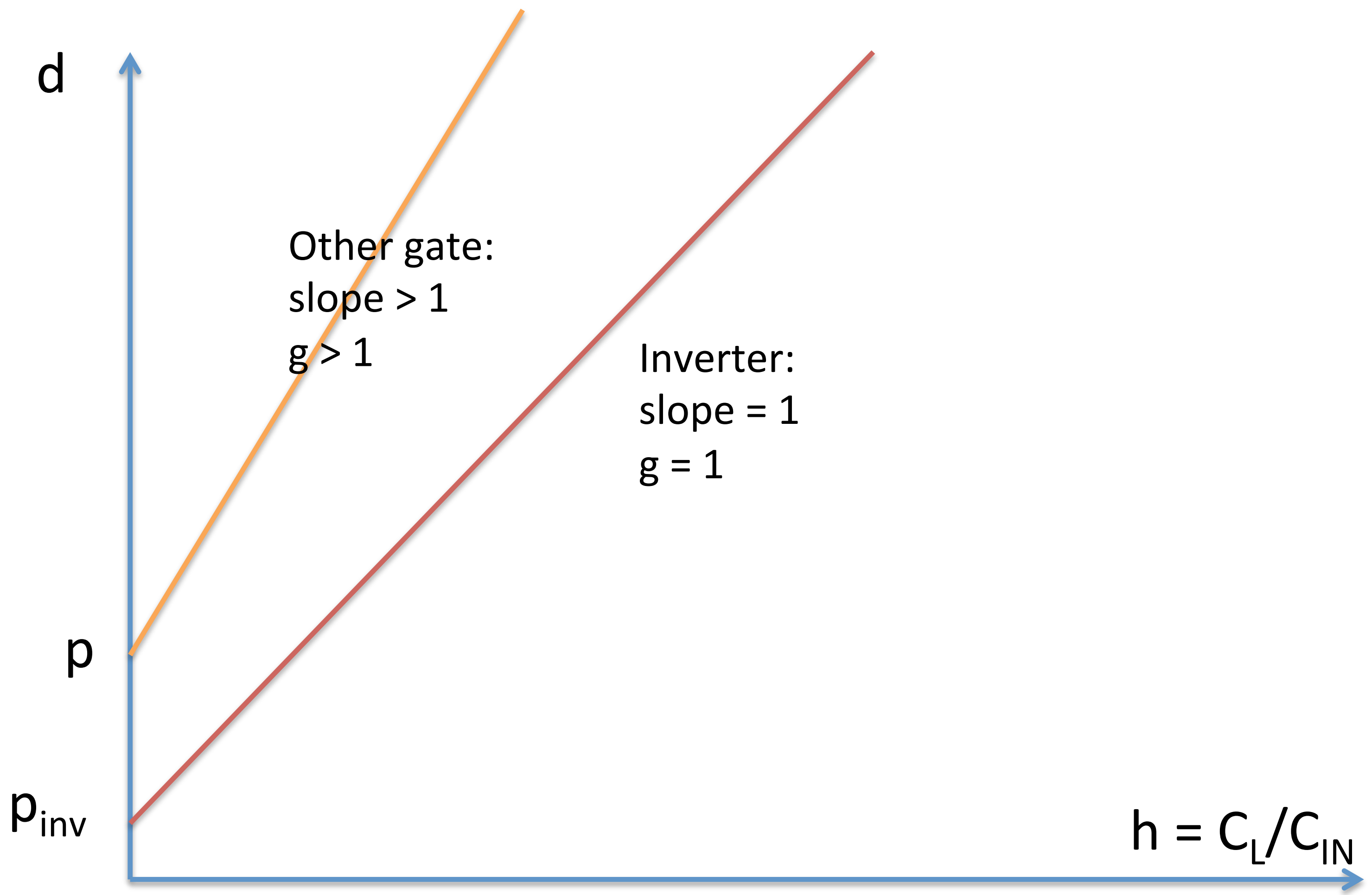
Different slope for each value of C_{IN}, that is each inverter size

$$d = p_{inv} + (1/C_{IN}) \times C_L$$

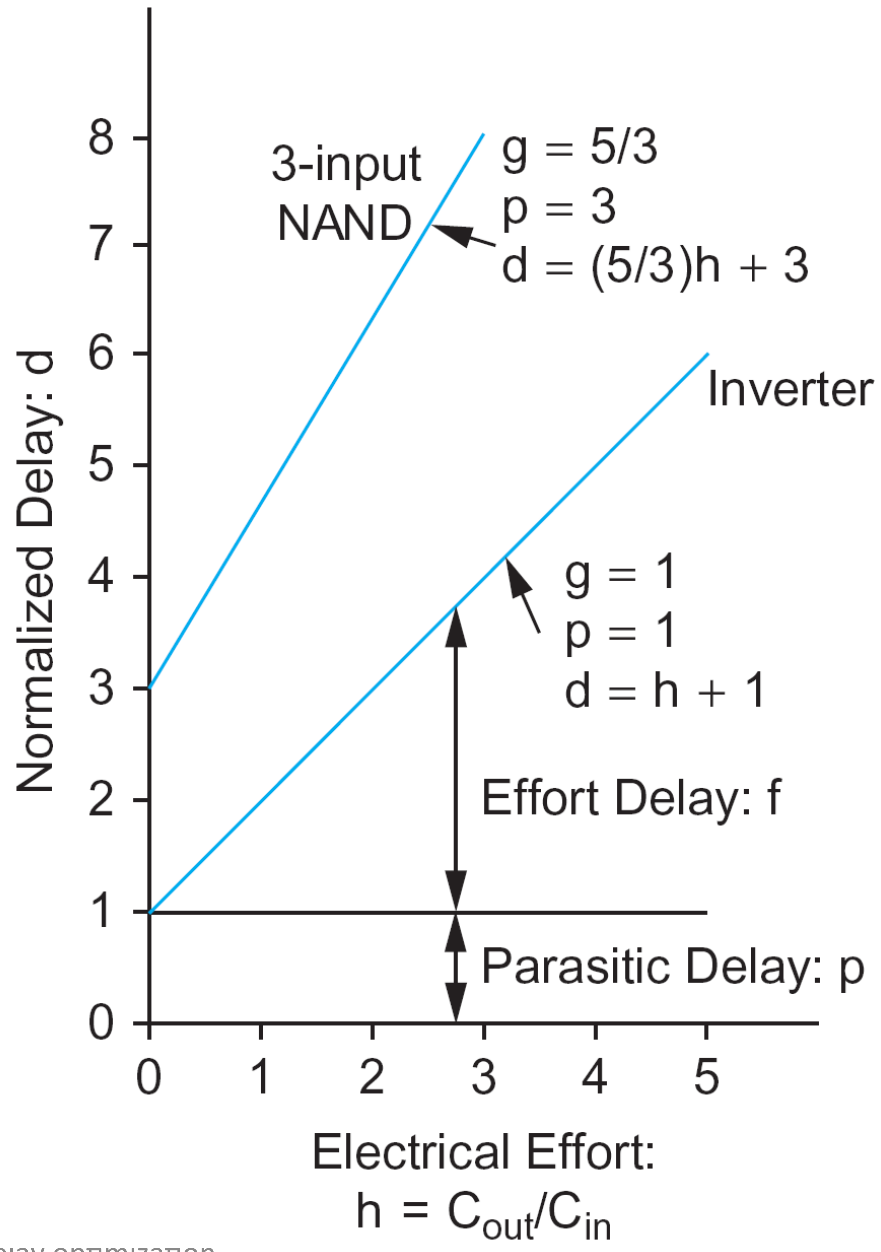
Graph of normalized delay for inverter as a function of h



Graph of normalized delay other gates



Example: NAND3



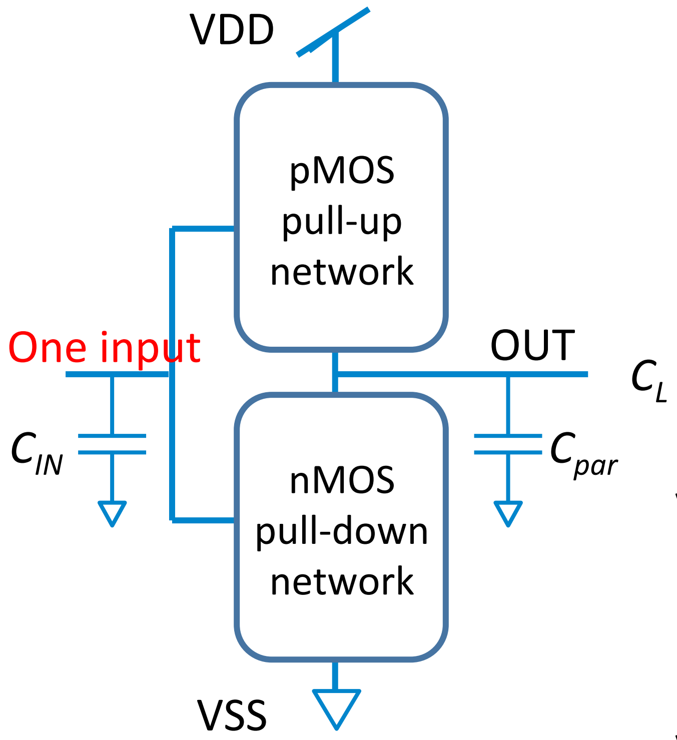
Unclear things

Propagation delay, t_{pd}, from *one input signal* to the *output* is modeled with the normalized delay, d,
so C_{IN} is the input capacitance for ONE of the gate inputs (the one where we assume our input signal is entering).

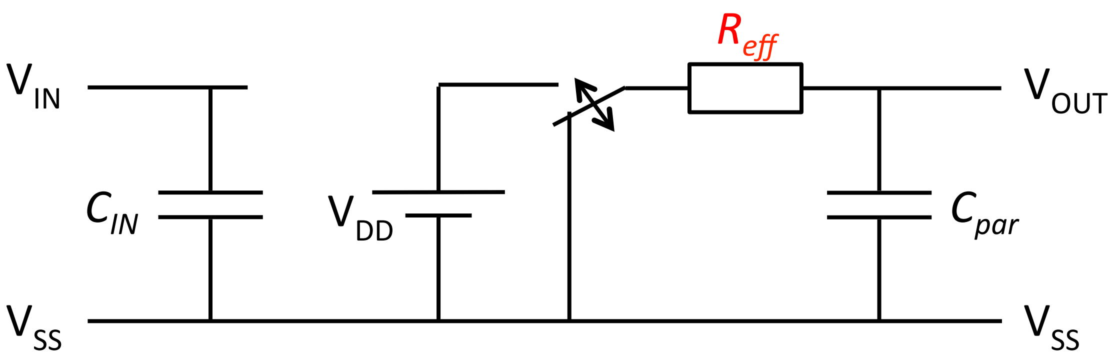
Often all gate inputs have same C_{IN}, but not always.

Logic gate propagation delay model

Now we want to apply the same model to any CMOS logic gate



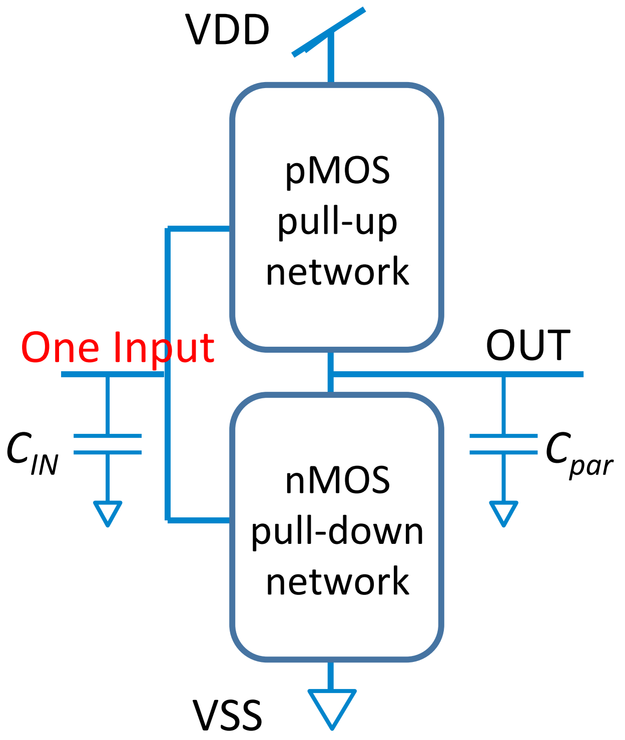
Note that already here R_{eff} is placed to the right of the switch, which means we assume the the n-net path and p-net path worst-case resistances are the same.



C_{IN} is the input capacitance for one of the inputs to the logic gate
C_{par} is the total parasitic capacitance at the gate output

Logic gate propagation delay model

Now we want to apply the same model to any CMOS logic gate

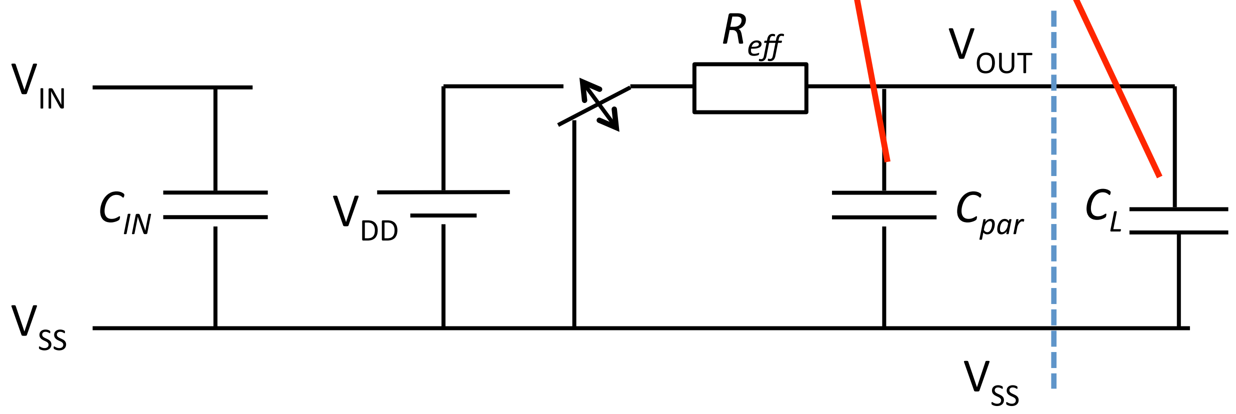


Assume all pull-up/down paths have same resistance R_{eff}. Obviously, the logic gate will have larger RC product than the inverter! What is the delay with load C_L from one of the inputs to the logic gate?

As before, normalize to process time constant tau!

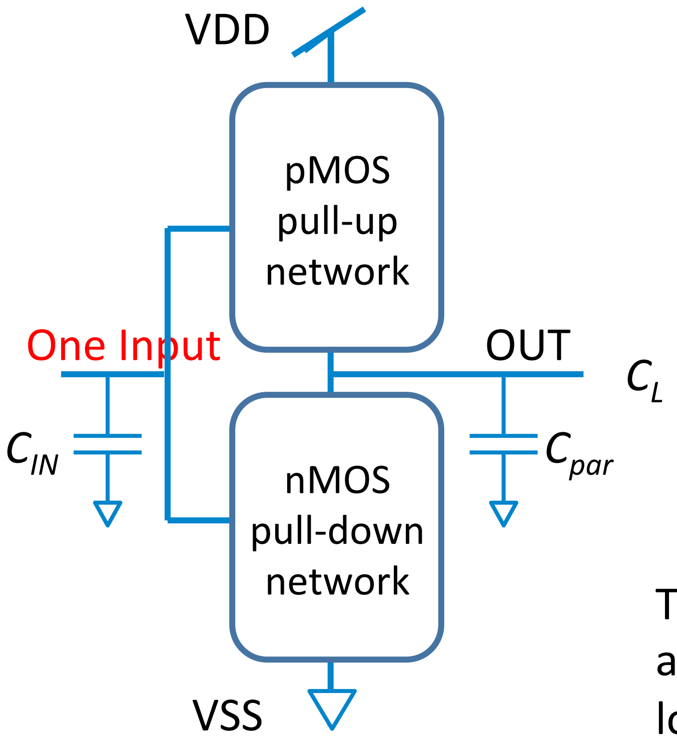
C_L

$$t_{pd} = 0.7R_{eff}\left(C_{par} + C_L\right) = \tau\left(\frac{R_{eff}C_{par}}{\left(RC_G\right)_{inv}} + \frac{R_{eff}C_L}{\left(RC_G\right)_{inv}}\right)$$



Logic gate propagation delay model

To simplify, we suggest sizing MOSFETs for equal effective resistances, i.e. $R_{eff}=R$



Consider one delay term at a time!

$$\text{parasitic delay: } p = \frac{RC_{par}}{(RC_G)_{inv}} = \frac{C_{par}}{C_D}\underbrace{\frac{C_D}{C_G}}_{p_{inv}}$$

Same for all inputs

$$\text{stage effort}: f = \frac{RC_L}{(RC_G)_{inv}} = \underbrace{\frac{C_{IN}}{C_G}}_{g}\underbrace{\frac{C_L}{C_{IN}}}_{h} = gh$$

One input

This method conveniently separates the driving strength of a logic gate in terms of its **logical effort**, ***g***, and its external load in terms of its **electrical effort, *h***. And all with respect to the properties of the inverter.

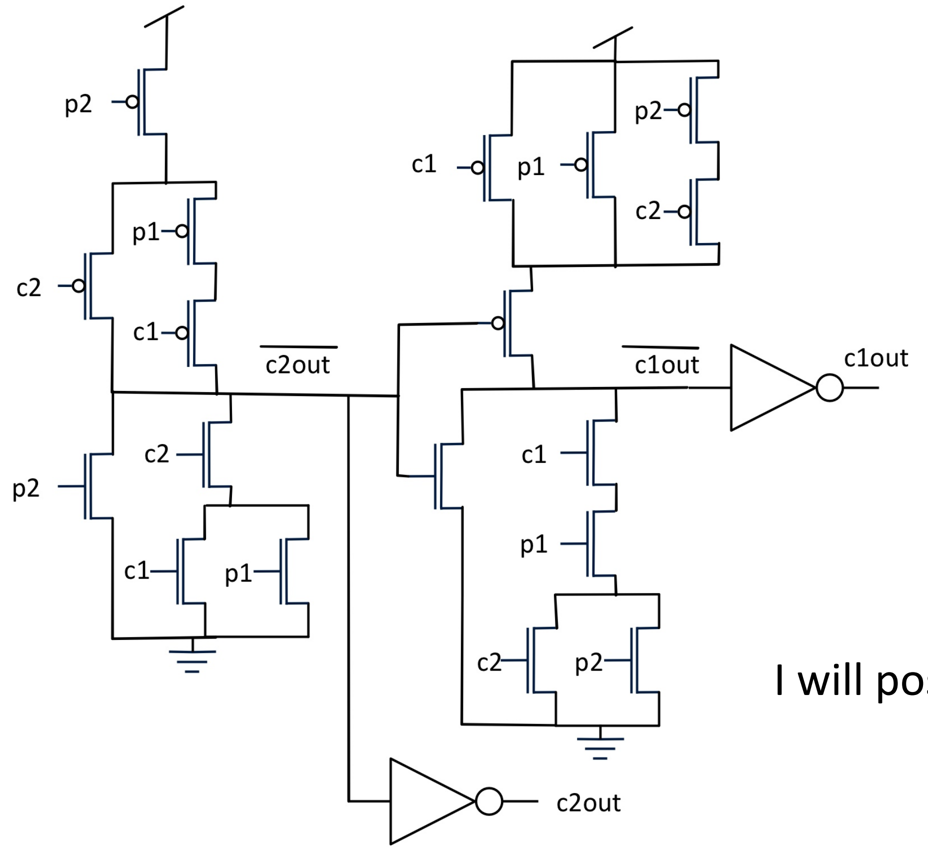
Deciding on resistances/scaling

- Make all paths have the same resistance:
  - Parallel paths:
    - Assume that only one of any parallel path is conducting at a time
  - Serial paths
    - All transistors groups in series have to conduct for a path to be formed, but only one of any transistors in parallel within groups.
    - As long as possible assign all transistors in series the same resistance (if there is no particular reason to do otherwise).

Parasitic capacitances

- All FETs have parasitic capacitances at drain and source
- Our simplified model is that only parasitic capacitances at the output node contribute to parasitic delay – not entirely true.
- Drain capacitances are proportional to **transistor width** & proportional to **gate capacitance**.
- Approach to find parasitic delay p:
  - Sum widths of all FETs connected to output node of gate.
  - Divide by width of FETS connected to output node in inverter with same R.
  - Multiply by p_{inv} factor (C_D/C_G)

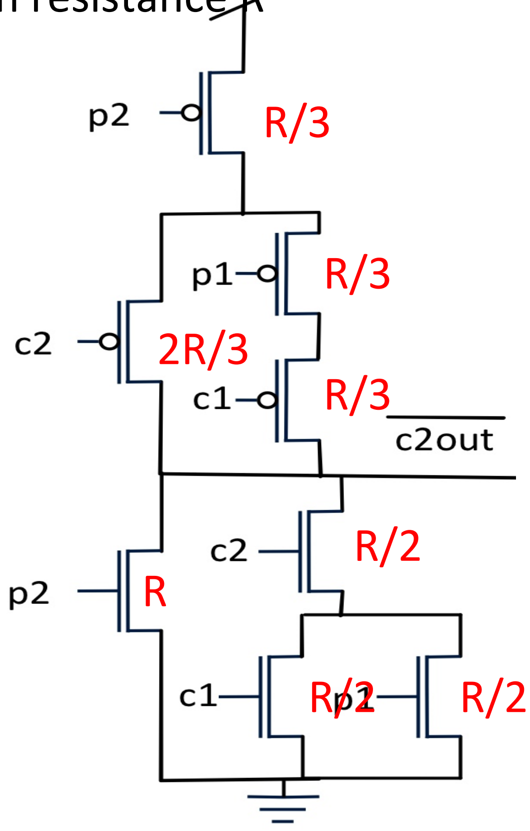
Two complex gates for you to practice on (but not right now)



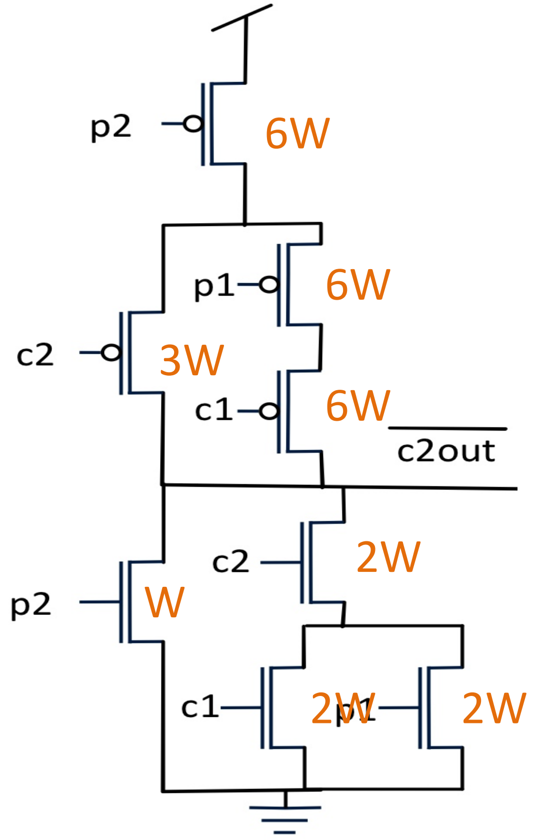
I will post the solution(s).

Solution 1 complex gate 1

Resistances for equal worst-case path resistance R



Corresponding widths

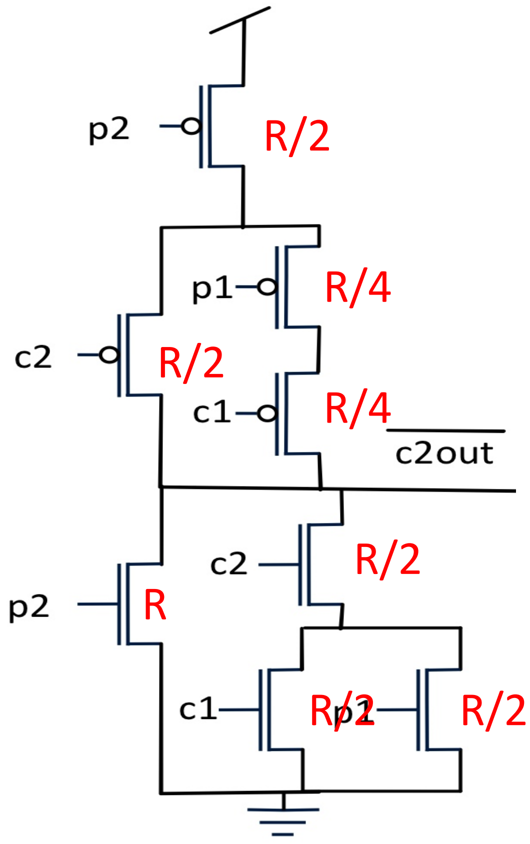


gc1 = (6+2)/3 =8/3
gc2 = (3+2)/3 =5/3
gp1 = (6+2)/3= 8/3
gp2 = (6+1)/3 = 7/3

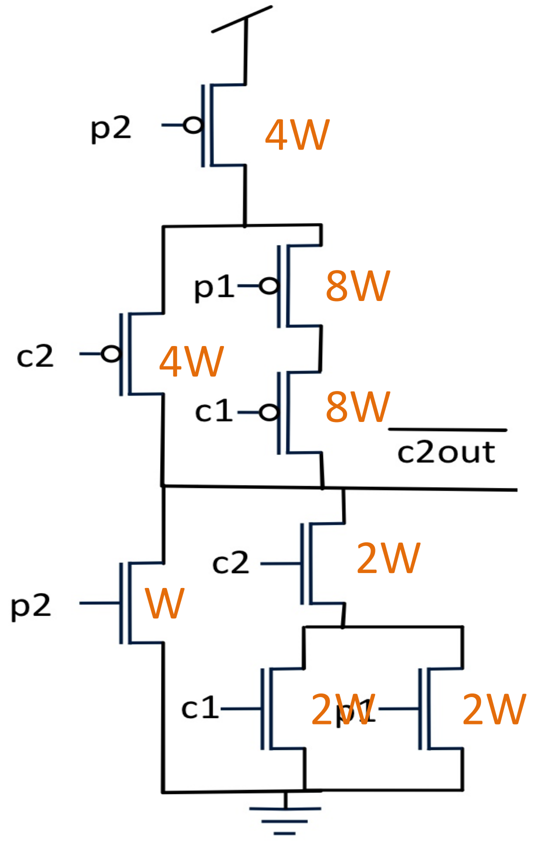
p =(1+2+3+6)/3 pinv= 12/3 pinv = 4 pinv

Solution 2 complex gate 1

Resistances for equal worst-case path resistance R, p-net changed



Corresponding widths



$g_{c1} = (8+2)/3 = 10/3$

$g_{c2} = (4+2)/3 = 6/3$

$g_{p1} = (8+2)/3 = 10/3$

$g_{p2} = (4+1)/3 = 5/3$

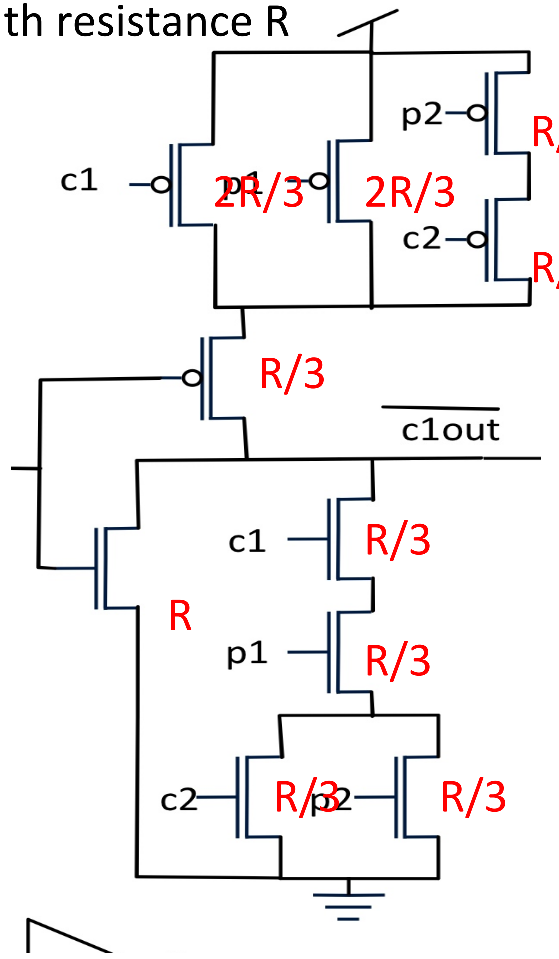
$p = (1+2+4+8)/3\ p_{inv} = 15/3\ p_{inv} = 5\ p_{inv}$

Complex gate 1 which solution is better?

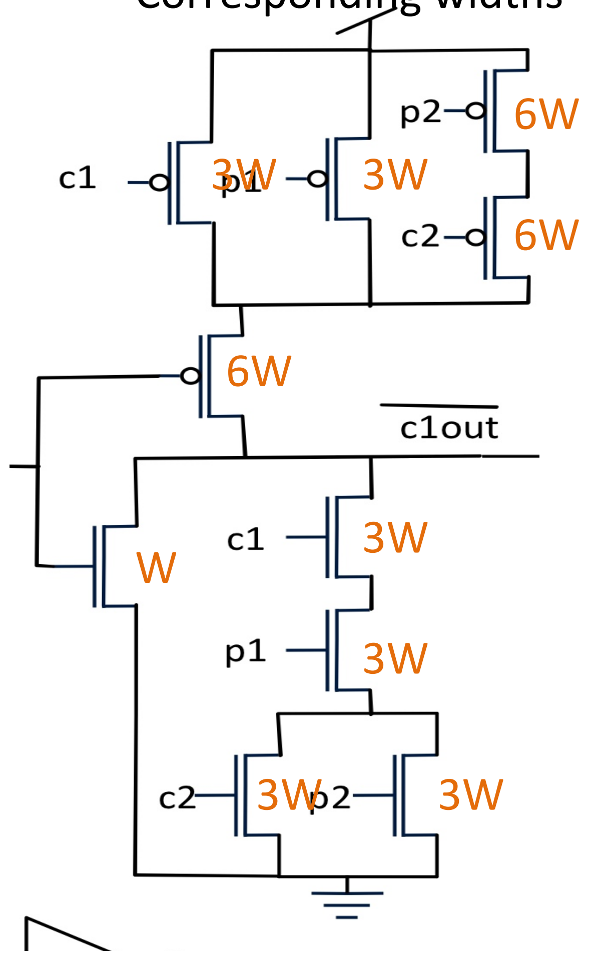
- It depends on which one is the critical input, the one on the critical path, that limits the speed, but unless it is p2, solution 1 is better. The resistances are more similar and also the parasitic delay is smaller.
- Note: There was a constraint due to the physical layout that made us not put the p-net upside down (more about layout next week!).

Solution 1 complex gate 2

Resistances for equal worst-case path resistance R



Corresponding widths

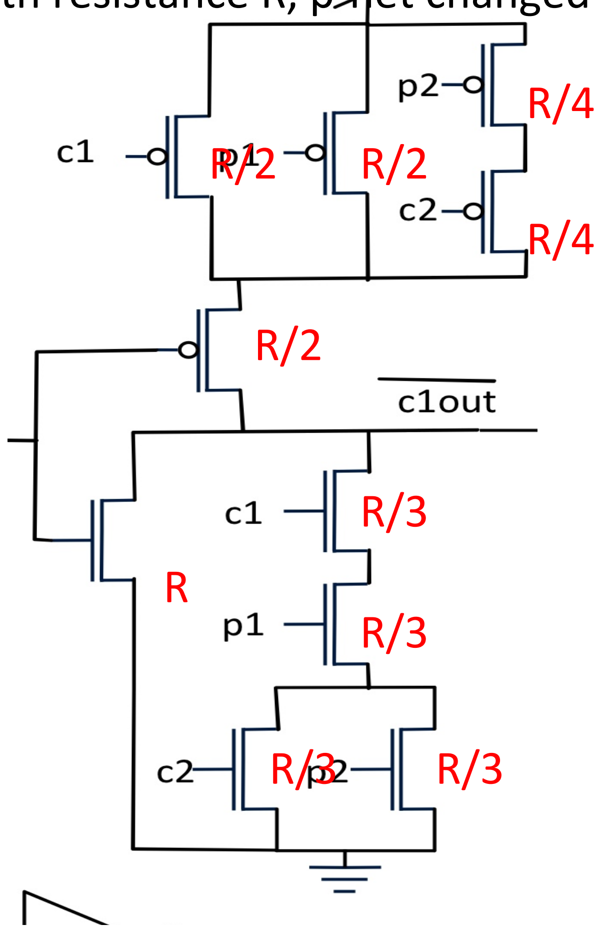


gc1 = (3+3)/3 =2
gc2 = (6+3)/3 =3
gp1 = (3+3)/3= 2
gp2 = (6+3)/3 =3
gCoutin = 7/3

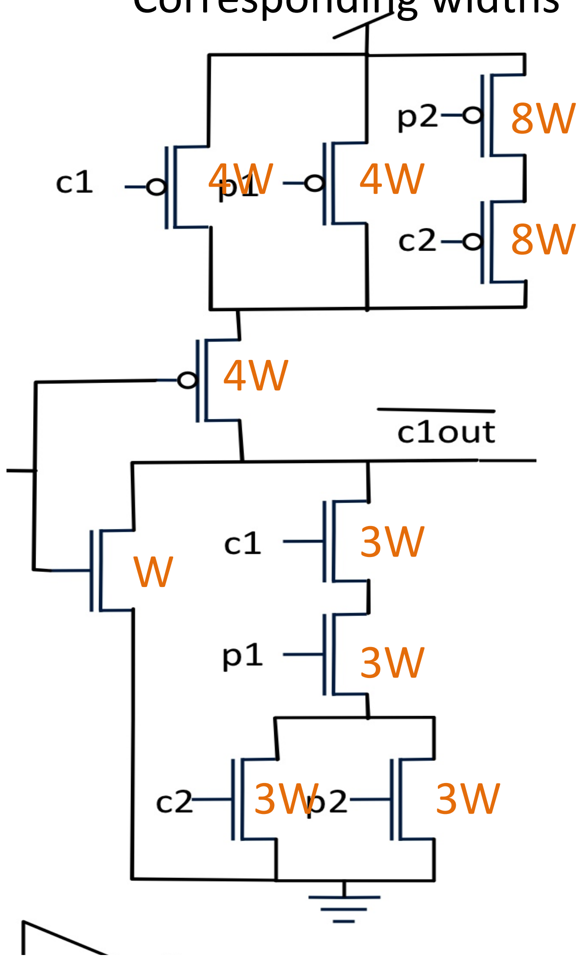
p =(1+3+6)/3 pinv= 10/3 pinv

Solution 2 complex gate 2

Resistances for equal worst-case path resistance R, p-net changed



Corresponding widths



gc1 = (3+4)/3 =7/3
gc2 = (8+3)/3 =11/3
gp1 = (3+4)/3= 7/3
gp2 = (8+3)/3 =11/3
gCoutin = 5/3

p =(1+3+4)/3 pinv= 8/3 pinv

Complex gate 2 which solution is better?

- Also here it depends on which one is the critical input (the one that is on the critical path and thus limits the speed), but from the schematic it looked like it would be c_{outin}. In that case solution 2 is better. It also has a lower parasitic delay.

Outline

- A linear model for non-inverting gates
  - Still d = gh + p, find p and g
  - An example
- Prelab 2 task
  - ILAs
  - The carry circuit
- Review tapered buffer – only inverters
  - Optimal h,optimal N
  - Review the example
- Path delay with other gates
  - Optimal f
  - An example

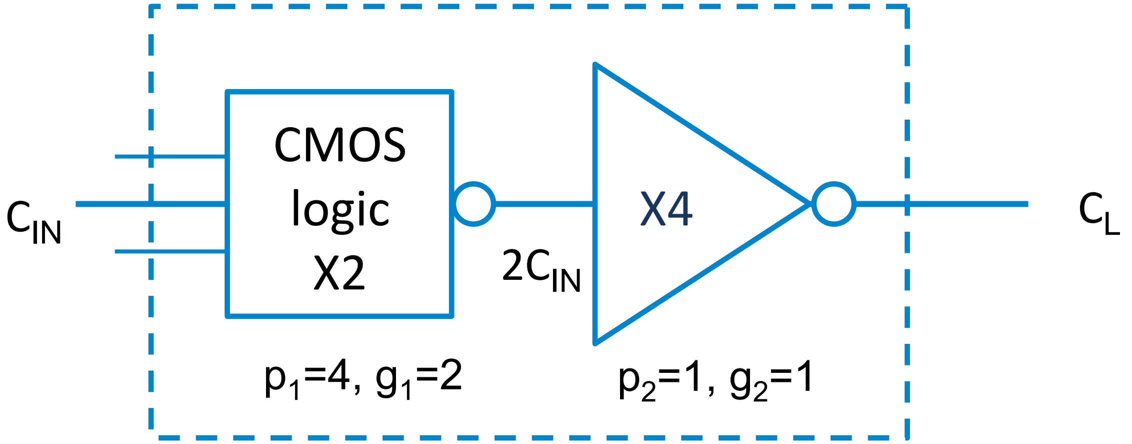
Model for a non-inverting gate

Linear model d = f + p = gh + p should hold also for non-inverting gate

f is the part that **depends on C_L**, f = gh where h is defined as C_L/C_{IN}

Thus $f_{gate} = C_{IN}/C_{INinv} \times C_L/C_{IN} = g_{gate} \times h_{gate}$

p is the static part of the delay: the part that **does not depend on C_L**

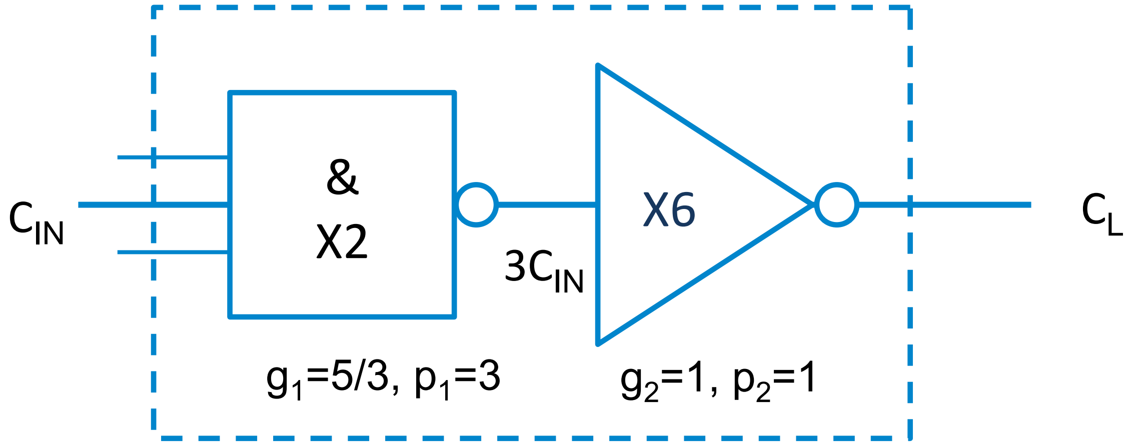


$$d = \underbrace{4 + 2\frac{X4}{X2}}_{\text{1st stage}} + \underbrace{1 + \frac{C_L}{2C_{IN}}}_{\text{inverter}} = \underbrace{9}_{p} + \underbrace{\frac{C_{IN}}{2C_{IN}}}_{g}\underbrace{\frac{C_L}{C_{IN}}}_{h} = \underbrace{9}_{p} + \underbrace{\frac{1}{2}}_{g}\underbrace{\frac{C_L}{C_{IN}}}_{h}$$

The parasitic delay *p*=9
Logical effort *g*=0.5

Note that g < 1 is possible because here it depends on scaling between gates!

Quiz about 3-input AND gate = 3-input NAND + inverter



What is p_{AND} and g_{AND} (the same for all three inputs)?

Work in small groups.
Enter your answers in Socrative: Room: MCC0922018.

Prelab 2

Iterative Logic

Prelab 2

- Design task: design with MOSFETs the carry bit-cell for an 8-bit ripple-carry adder!
- The 8-bit carry logic is to be implemented by an iterative logic array consisting of eight instances of the bit-cell that you have designed.
- The carry bit-cell has three inputs (two bits a, b and a carry-in memory bit), and one output, the carry-out memory bit to the next more significant bit.

Designing an adder word slice

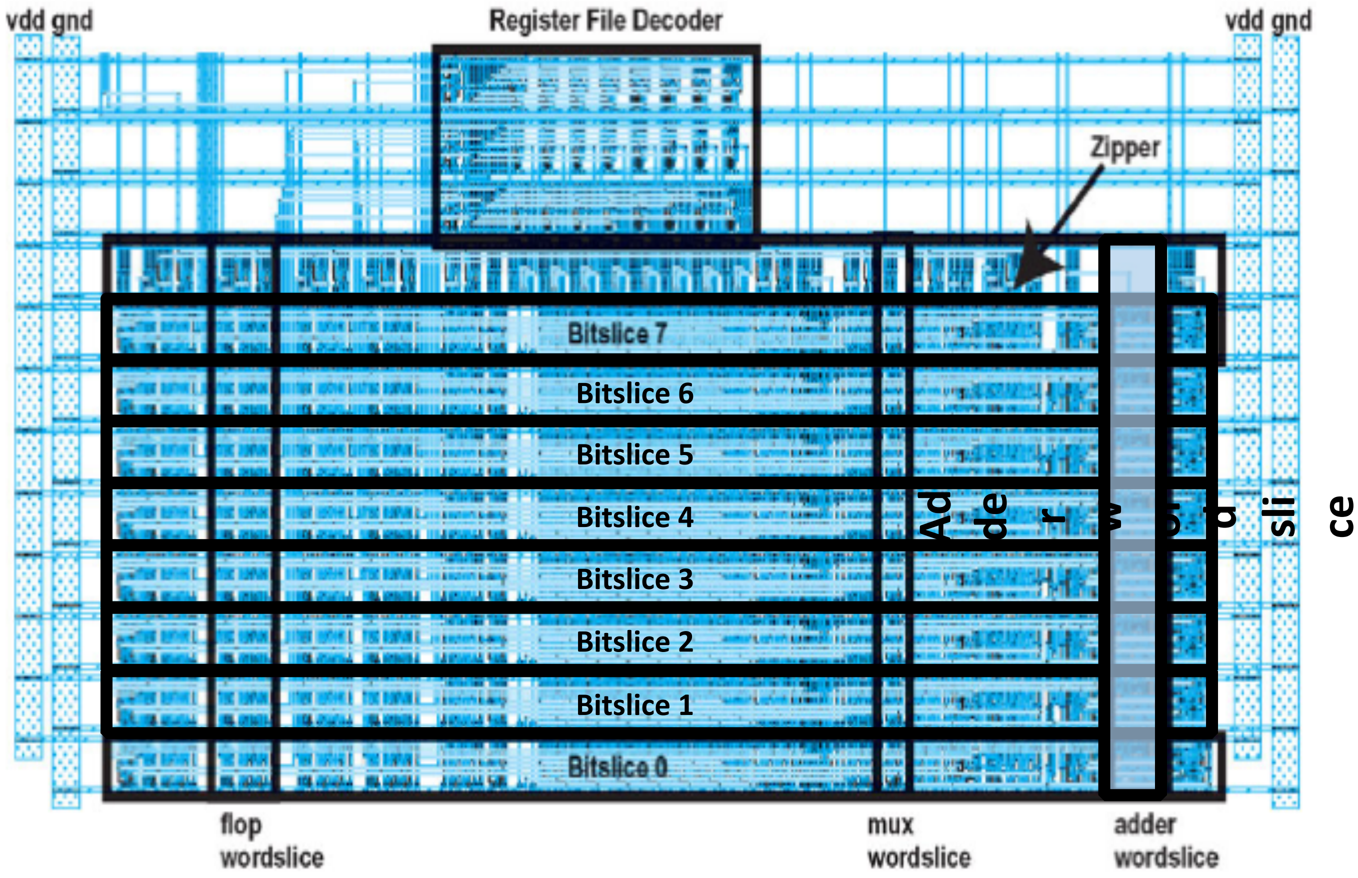
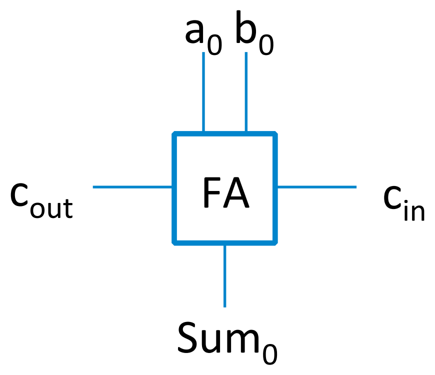
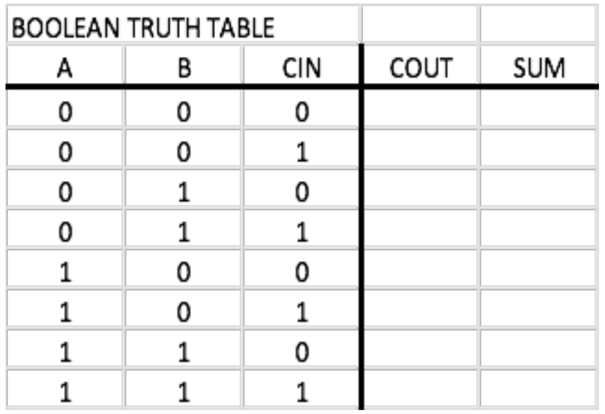


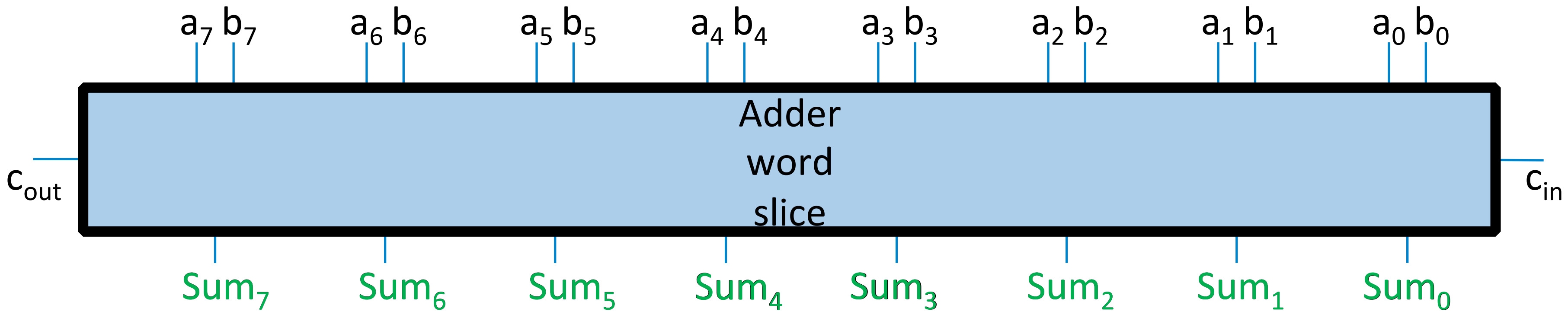
FIGURE 1.67 MIPS datapath layout

Iterative logic arrays: FULL ADDER

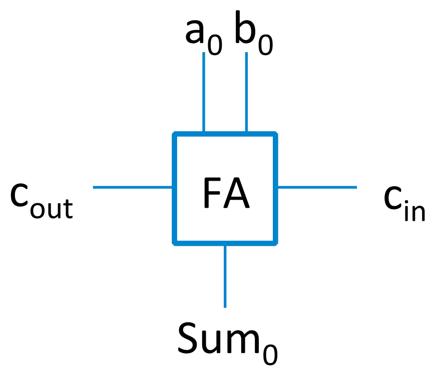




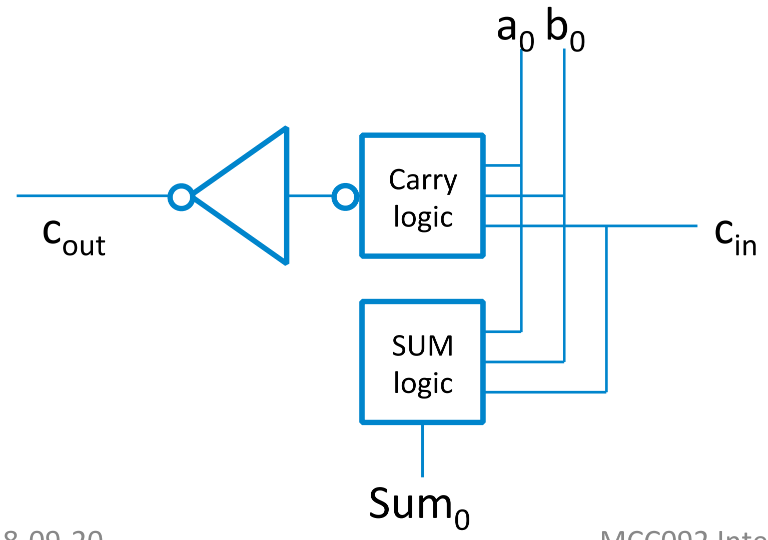
BOOLEAN TRUTH TABLE				
A	B	CIN	COUT	SUM
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		



Iterative logic arrays: FULL ADDER



BOOLEAN TRUTH TABLE				
A	B	CIN	COUT	SUM
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		



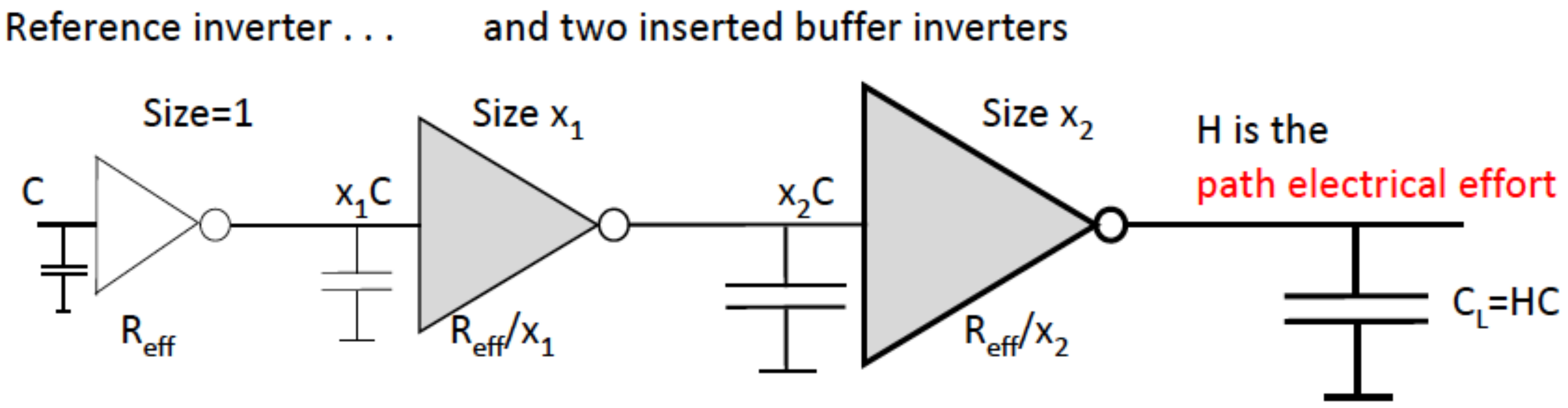
Prelab design tasks (see prelab 2 instructions):

- Design **carry logic** as a SUM of products!
- Draw MOSFET schematic!
- Calculate logical effort and parasitic delay of the inverting carry-logic cell that you have designed!
- Calculated delay through carry chain

Prelab 2

What is still unclear?

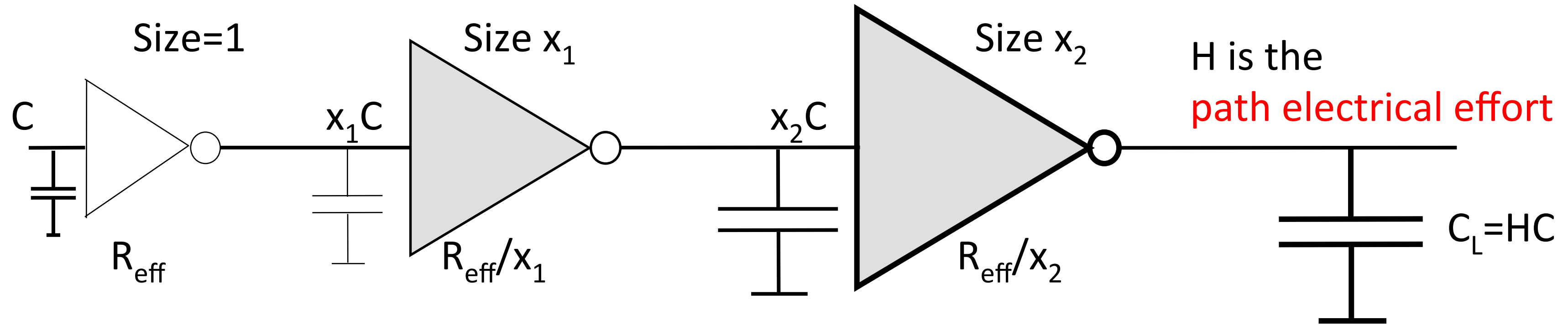
Review from lecture 4: Tapered buffer



The tapered buffer

Please note that H is the path electrical effort while *h* is the stage electrical effort.

Reference inverter . . . and two inserted buffer inverters

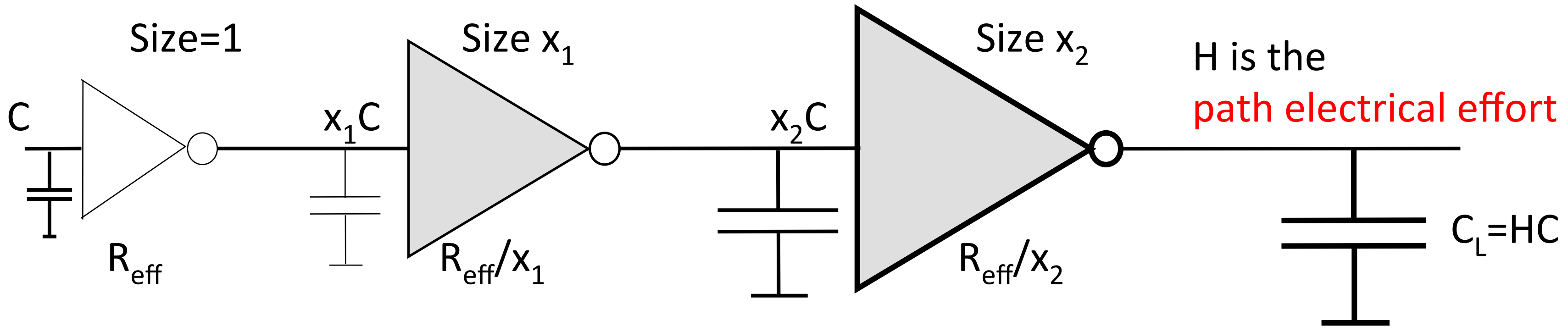


With two intermediate buffer inverters we obtain a normalized delay relative to tau:
$D=(p_{inv}+h_1)+(p_{inv}+h_2)+(p_{inv}+h_3)$
where we have defined the stage electrical efforts, or fanouts, *h*.
Here $h_1=x_1$, $h_2=x_2/x_1$, and $h_3=x_3/x_2$).
We know we must have $h_1h_2h_3 = H$, *So* only h_1 and h_2 are independent variables, the third, $h_{3,}$ becomes $h_3=H/h_1h_2$.
TASK: Show that minimum delay is obtained for $h_1=h_2=h_2=h=\sqrt[3]{H}$ >>> $D=3(p_{inv}+\sqrt[3]{H})$

The tapered buffer

Please note that H is the path electrical effort while *h* is the stage electrical effort.

Reference inverter . . . and two inserted buffer inverters



With two intermediate buffer inverters we obtain a normalized path delay:

$D=(p_{inv}+h_1)+(p_{inv}+h_2)+(p_{inv}+H/h_1h_2)$.

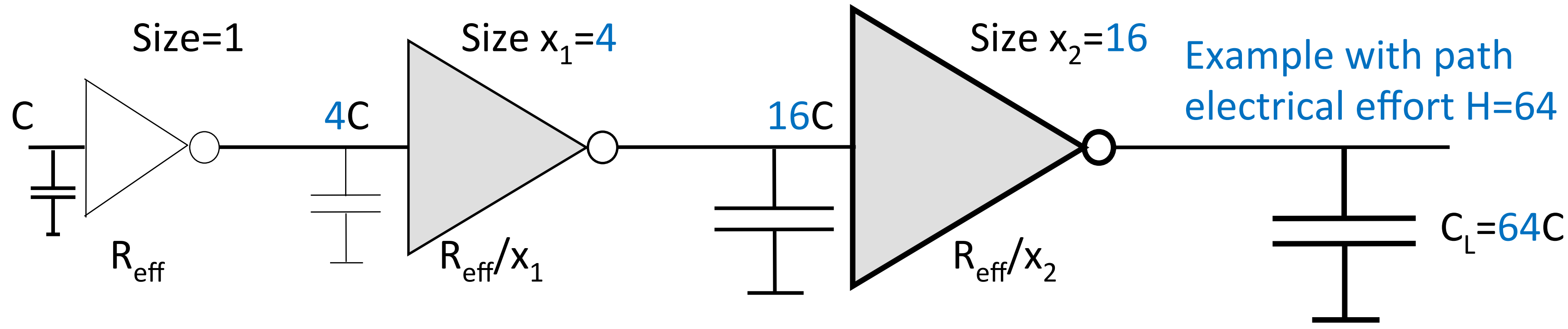
Taking partial derivatives wrt h_1 and h_2 we obtain

$$\frac{d}{dh_1}D(h_1,h_2)=1-\frac{H}{h_1^2h_2}=0,\ \text{and}\ \frac{d}{dh_2}D(h_1,h_2)=1-\frac{H}{h_1h_2^2}=0$$

This yields $h=h_1=h_2$ and $H=h^3 \rightarrow\ h=\sqrt[3]{H}$

The tapered buffer

Reference inverter . . . and two inserted buffer inverters



Example with path electrical effort H=64

Sharing the load equally between the inverters yields equal stage fanouts $h=\sqrt[3]{64}=4$

The total delay is then equal to 3 FO4 delays, i.e. 15tau=75 ps (assuming p_{inv}=1).

The tapered buffer

Solving this problem we start by having derived that minimum delay occurs when stage electrical efforts, *h*, are equal.

Hence path propagation delay is given by $D = N(p_{inv} + h)$

Furthermore, $h = \sqrt[N]{H}$, i.e. $H = h^N$.

Taking natural logarithms we obtain number of inverters $N = \dfrac{\ln H}{\ln h}$

We rewrite path delay equation as $D = \dfrac{p_{inv} + h}{\ln h} \ln H$

Looking for minimum path delay by taking derivatives of *D* wrt *h*

we obtain $\ln H \dfrac{\ln h - (p_{inv} + h)/h}{(\ln h)^2} = 0, \text{ i.e. } \ln h = \dfrac{p_{inv} + h}{h}$

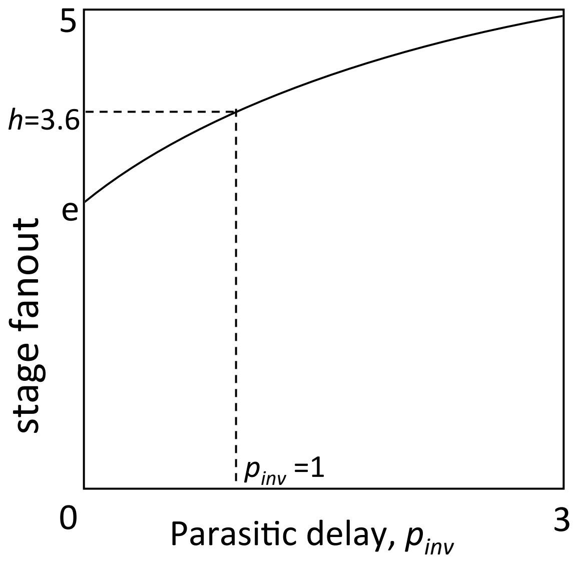
Analytical solution is possible only for $p_{inv}=0$:

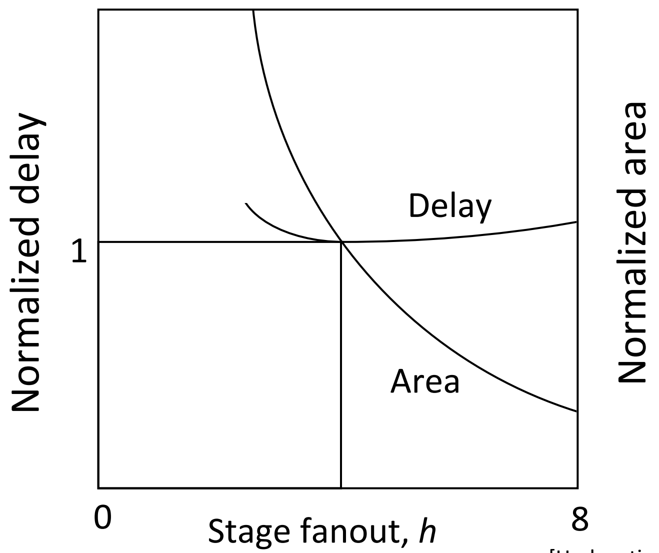
$h=e=2.72$ which gives N = ln H

Note: Derivation inserted for completeness.
You don't have to learn to do this derivation!

The tapered buffer

- For $p_{inv} \neq 0$ the equation has to be solved numerically





[Hedenstierna & Jeppson 1987]

For typical values of p_{inv} the optimum tapering factor is between 3.6 and 5. Typically a FO4!

Note that the propagation delay minimum is rather flat,
while total inverter area on the silicon decreases rapidly when larger stage fanout is used.
Silicon real estate (=cost) can be saved for relatively little loss of speed!

Review from lecture 4: Tapered CMOS inverter stages

$H = C_L/C_{IN}$: path electrical effort

Equal stage electrical effort, h, gives shortest delay

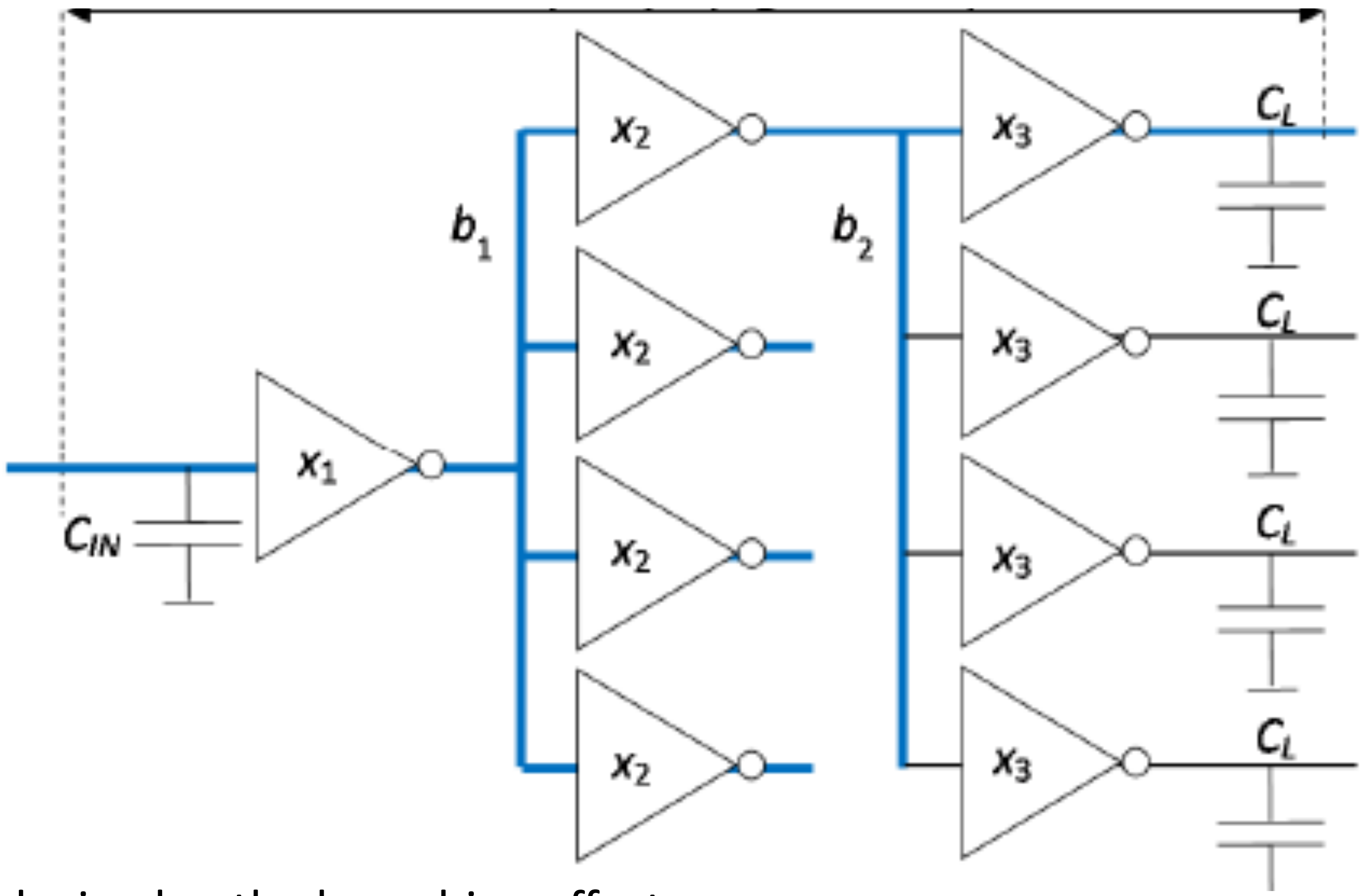
- $h_{opt}^N = H$, where N is number of stages
- That is $h_{opt} = \sqrt[N]{H}$

Normalized path delay for path is called D_{opt}:

- $D_{opt} = N \times h_{opt} + P$, with P sum of all parasitic delay
- $\mathbf{D_{opt} = N \times h_{opt} + N \times p_{inv}}$

If N not given, select N such that h is close to 4; to save area make h a bit higher than 4.

CMOS inverter stages with branching



Introducing b = the branching effort

$b = (c_{onpath} + c_{offpath})/c_{onpath}$

b is 1 or larger

Path delay with branching

Equal stage electrical effort gives shortest delay

Path effort: $F = H \times B$

- B: path branching effort
- $B = b_1 \times b_2 \times b_3 \times \ldots b_{N-1}$ (assuming N stages)
- $b = (c_{onpath} + c_{offpath})/c_{onpath}$ branching effort

Determine $f_{opt} = \sqrt[N]{F}$

Normalized path delay D_{opt}

- $D_{opt} = N \times f_{opt} + P$ where P is sum of parasitic delay
- $\mathbf{D_{opt} = N \times f_{opt} + N \times p_{inv}}$

Path delay – when (some) gates are not inverters

- Equal stage effort still gives shortest delay
  - Now includes also logical effort: g
- Path effort: $F = G \times H \times B$
  - G: path logical effort
  - $G = g_1 \times g_2 \times g_3 \times \dots g_N$ (assuming N stages)
  - B defined as before
- Determine f_{opt} as $\sqrt[N]{F}$.
- Normalized path delay D_{opt}

  $\mathbf{D_{opt} = N \times f_{opt} + P}$**,** P is sum of parasitic delay

Approach for delay optimization= path sizing

Read W&H section 4.5.4
For more practical considerations

- Given: **N** stages**, g** and **p** for all gates + any branching efforts **b**
- Calculate path effort **F** from F = G×H×B
- Calculate stage effort $\mathbf{f_{opt}}$ as $\sqrt[N]{F}$
- (Calculate path delay D = N×f_{opt} + P)
- Find gate sizes $\mathbf{X_2}$ to $\mathbf{X_N}$ starting from start or end of path.
  - Note that X_1 (input capacitance of first stage) does not change!
  - Don't forget to include branching efforts!
- Check also that f_1 /f_N (for the remaining stage) is also f_{opt} so you did not make a mistake!

Clock tree task from latest exam

Solution:

f_{opt} = 32/3

D_{opt} = 38

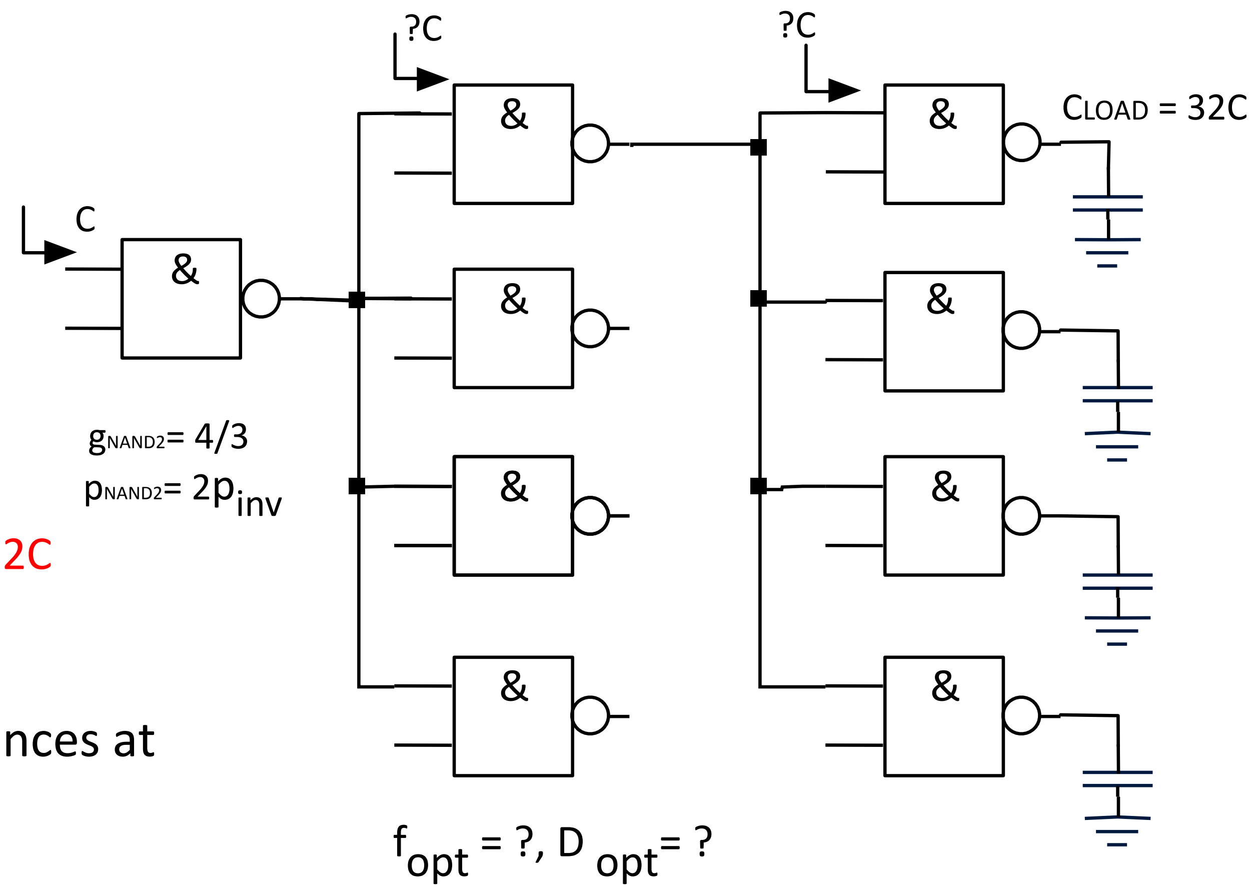
(assuming p_{inv} = 1)

Stage 3: C_{IN} =

(4/3)× 32C/(32/3) = 4C

Stage 2: C_{IN} =

(4/3) × (4Cx4)/(32/3) = 2C



Find f_{opt}, D_{opt} and the input capacitances at stages 2 and 3

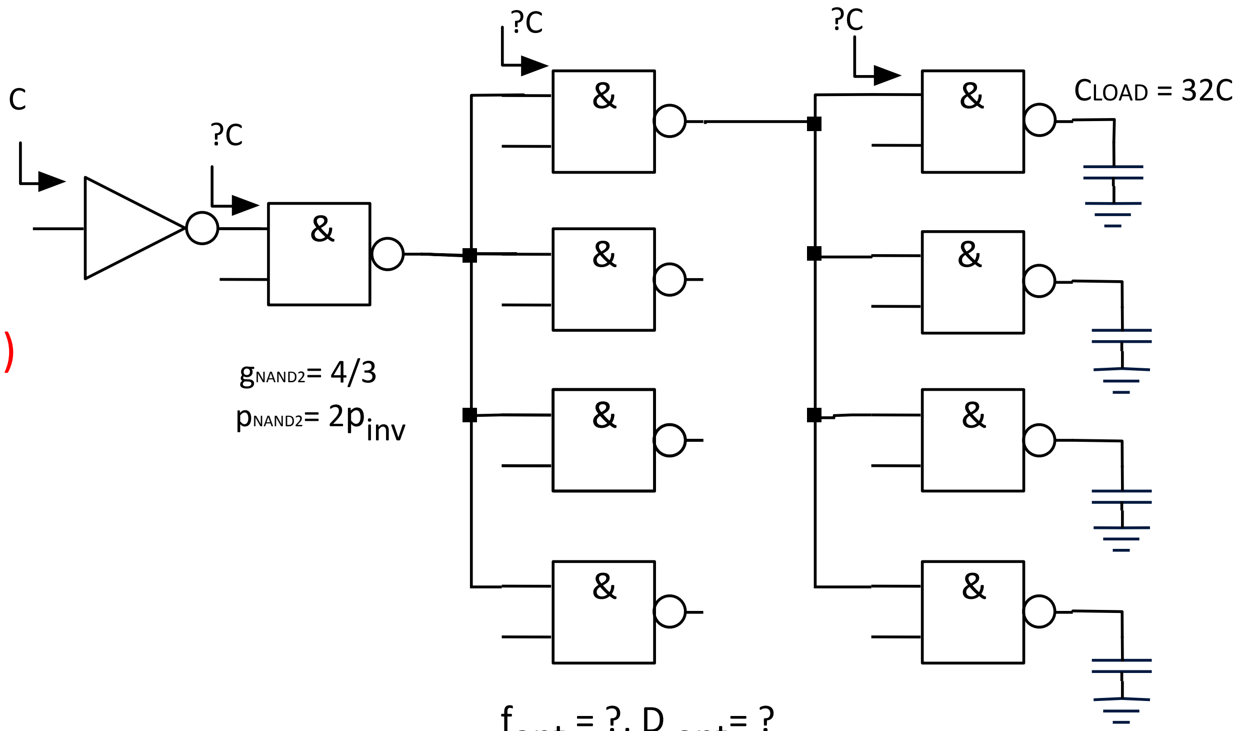
Work in small groups

Replies in socrative.com room: MCC0922018 as usual

Improve delay by adding one inverter stage – how much is path delay improved?

Solution:
$F = (32/3)^3$
$f_{opt} = 4\sqrt{F} \approx 5.9$
$D_{opt} = 4 \times 5.9 + 7p_{inv} = 30.6$ (assuming $p_{inv} = 1$)

$C_{IN2} = 5.9C$
$C_{IN3} = 6.5C$
$C_{IN4} = 7.2C$



$f_{opt} = ?$, $D_{opt} = ?$

Find new
f_{opt}, D_{opt}
with added stage

Later you can calculate the sizes too
I will post those numbers

Summary path delay optimization

Goal: minimize normalized path delay (that is, critical path delay)

Path effort $F = G \times H \times B$ (general expression for all cases)

path electrical effort: $H = C_L/C_{IN}$ (for entire path)

path branch effort: $B = b_1 \times \ldots\ldots \times b_N$ (for entire path)

path logical effort: $G = g_1 \times \ldots\ldots \times g_N$ (for entire path)

Optimal stage effort is $f_{opt} = \sqrt[N]{F}$

Optimal path delay D_{opt} is then: $D_{opt} = \mathrm{N} \times f_{opt} + P$

where P is path parasitic delay = sum of all p

Read W&H section 4.5 Logical Effort of Paths

Conclusion

- Review of logical effort muddy issues
- Linear model for non-inverting gate
  - d + gh, g due to scaling between gate and inverter, p the internal delay
- Prelab 2
  - Design a carry cell
- Review tapered buffer
  - h_{opt}, N_{opt}
  - $h_{opt} = \sqrt[N]{H}$
  - N_{opt} corresponds to FO4 stage delay but shallow minimum
- Path delay optimization
  - Path effort: F = GHB
  - $f_{opt} = \sqrt[N]{F}$
  - $D_{opt} = N \times f_{opt} + P$
  - Finding the sizes from start or end of path.