Postlab lab 4
Recap power
Adder recap

October 11, 2018

Week 6

- Monday lab 4
  - Wires, Clock tree simulation
- Tuesday
  - Adder exercise part 1
  - Lecture Power
- Thursday
  - Postlab review lab 4, recap power, adder consolidation
  - Tutorial POTW Power

Hand-in problem set 2

- Due on Monday October 15 23.55
- Grading works the same way as for prelabs
  - (but we will not be able to start grading immediately)
- 5 problems
  - 3 on power
    - Power/delay/area tradeoff in tapered buffer from lab 4
    - Activity factors
    - Designing the power switch for reducing leakage
  - 2 on adders
    - Redesigning you cell from lab 2 and 3 with P and G signals
    - The dot operator / PG logic

Learning outcomes   In context

- design static CMOS logic gates (pull-up and pull-down networks) and implement these as standard cell
- from simple MOS transistor models, estimate static and dynamic properties of CMOS inverters and use these properties to model more complex gates.
- derive logical-effort normalized-delay parameters from circuit diagrams or layout, and use these parameters to estimate and trade off performance measures such as critical-path delays and power dissipation in present and future CMOS technologies.
- find critical paths in more complex combinatorial circuits, such as adders, and determine and minimise their delays.
- analyse wire-delay-dominated cases such as clock distribution and global interconnect, and suggest suitable buffering schemes to minimize delay or delay spread.
- design simple sequential systems that meet set-up and hold time constraints for timing circuits, including the effect of metastability in synchronisation.,
- use industrial-type design automation tools to design, implement and verify basic CMOS circuit elements following the design flow supported by such tools.

What does “in context” mean?

- All information may not be readily available. May have to deduce it from data, graph or assume something reasonable.
- The problem often has to be “extracted” from its context to make it “clean”.
- You have to decide which method to use.
- There may not be one “right answer”. You may have to argue for why you decide to design one way or the other.

From MUD cards

- Does gate leakage dominate over subthreshold leakage in CMOS process below 45 nm?
- How to compute activity factors for entire systems/chips?
- What happens if we lower V_{DD} too much?

Subthreshold leakage

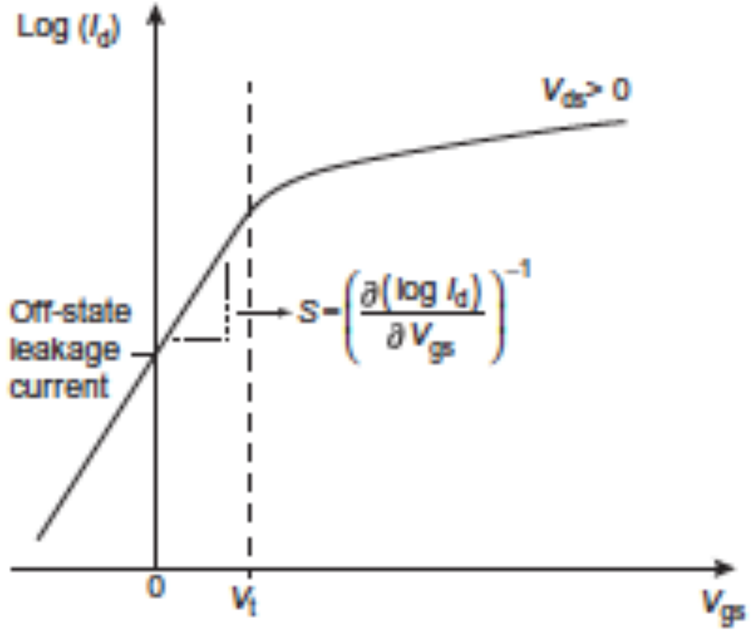


FIGURE 5.1

The transistor's drain current (on a logarithmic scale) as a function of its gate voltage. The off-state leakage current, threshold voltage, and subthreshold swing are marked.

Gate leakage

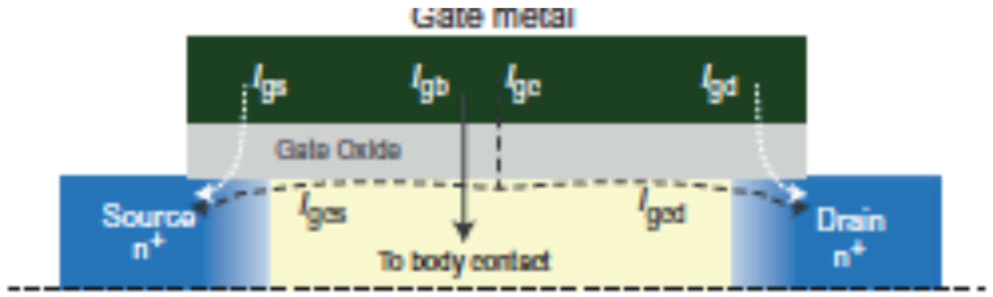


FIGURE 5.5

Half cross-section of the fin of the FinFET shown in Figure 5.2. The components of the tunneling current are shown. I_{gs} and I_{gd} are tunneling currents in the gate-to-source/drain overlap regions; I_{gb} flows between the gate and the body; I_{gc} is the gate-to-channel tunneling current and it is partitioned into I_{gcs} and I_{gcd}, which flow out of the source and drain, respectively.

A chapter on leakage currents in finFETs is now available in Documents -> Extra readings for the interested
The description of the different types of leakage is very good there.

Does gate leakage dominate subthreshold leakage below 45 nm?

- Short answer:
- Not yet, due to:
  - high-K dialectrics employed from 45 nm and below.
  - FinFETS
- However, it is just a temporary solution!

Activity factor definition

α= Probability that output switches from 0 to 1

P_i= probability that node *i* is 1

$\overline{P_i} = 1 - P_i$= probability that node *i* is 0

If the probabilities are uncorrelated:

$$\alpha = \overline{P_i} P_i = (1 - P_i) P_i$$

For random data: $P_i = 0.5$ so $\alpha = 0.25$

Some two-input functions

A	B	AND	NAND	OR	NOR	XOR
0	0	0	1	0	1	0
0	1	0	1	1	0	1
1	0	0	1	1	0	1
1	1	1	0	1	0	0

Step 1. Write down the expression for the probability of the output of each of the function being one.

Step 2. Calculate activity factors for the five gates with PA=PB=0.5. Report in Socrative.

Result: All gates that have a 3-to-1 division of ones and zeros has the activity Factor 3/16, but the XOR gates has 1/4.

Activity factors in large systems

- For certain signals it is easy:
  - Clocks: activity factor = 1
  - Logic signals that flip every clock cycle = ½
    - An example is a ring oscillator
- Otherwise you have to simulate and/or estimate

What happens when you decrease V_{DD} "too much"?

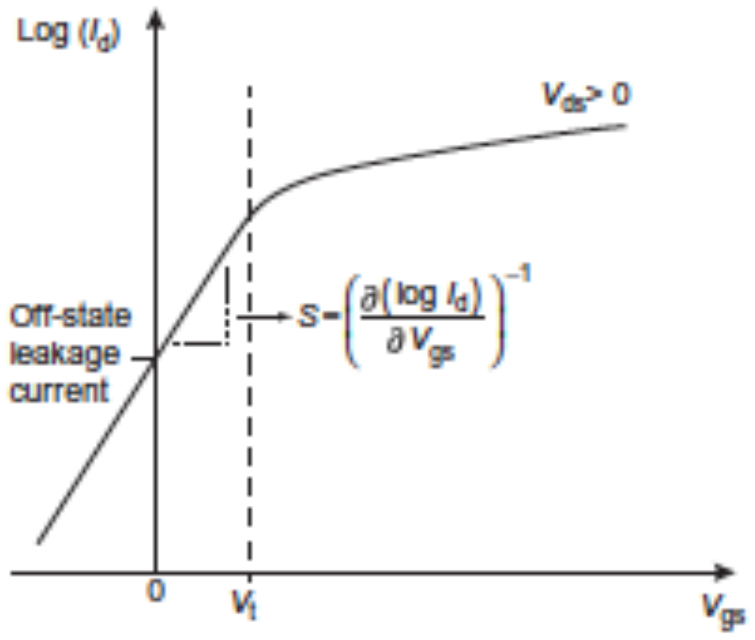


FIGURE 5.1

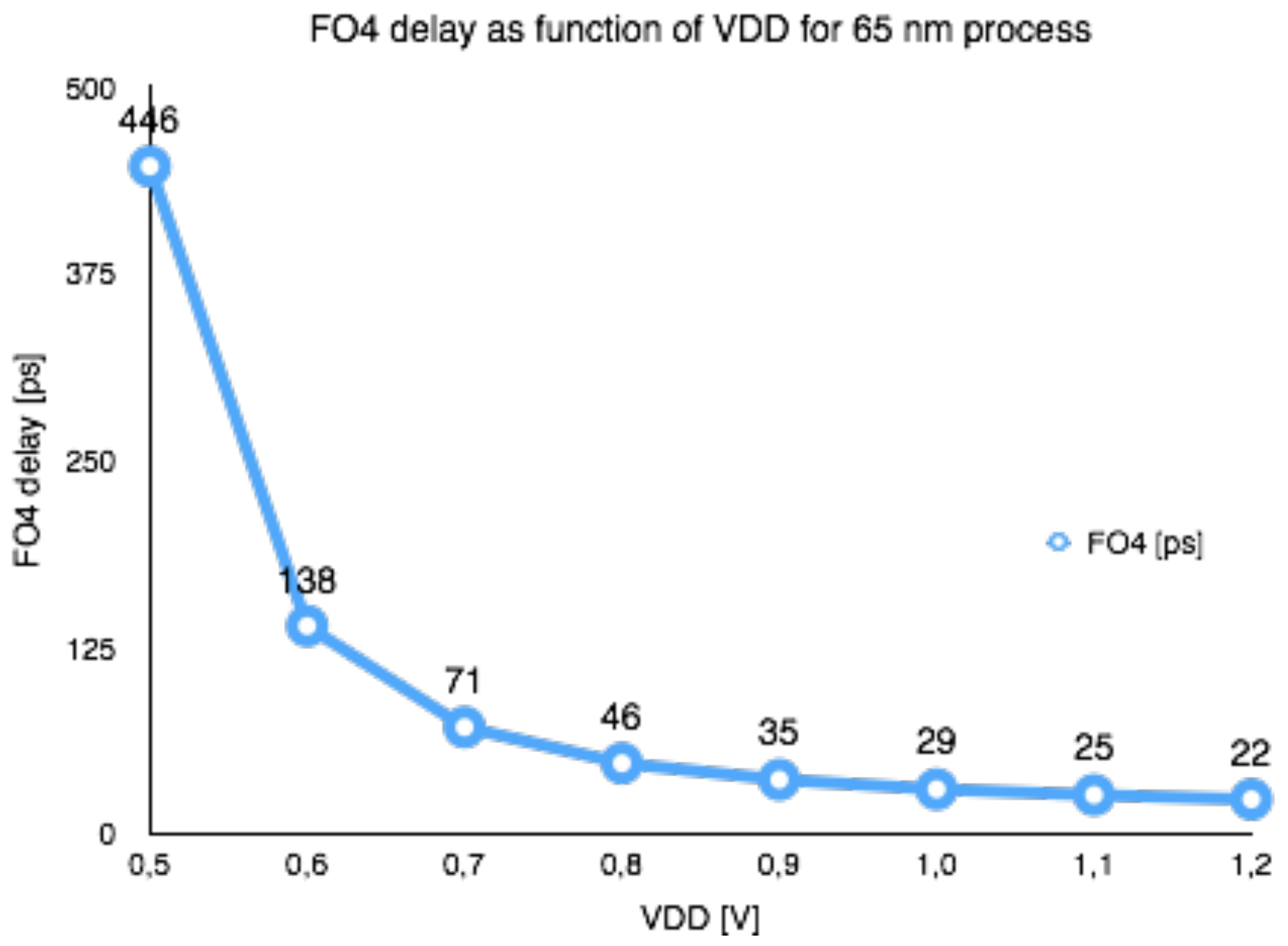
The transistor's drain current (on a logarithmic scale) as a function of its gate voltage. The off-state leakage current, threshold voltage, and subthreshold swing are marked.

Below the threshold voltage there is also current.

But is it very small compared to the capacitances!

Consequence:
Really SLOW circuits that use very little power.

FO4 experiment



Prelab 4 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 $\ln H \dfrac{\ln h - (p_{inv} + h)/h}{(\ln h)^2} = 0, \text{ i.e. } \ln h = \dfrac{p_{inv} + h}{h}$

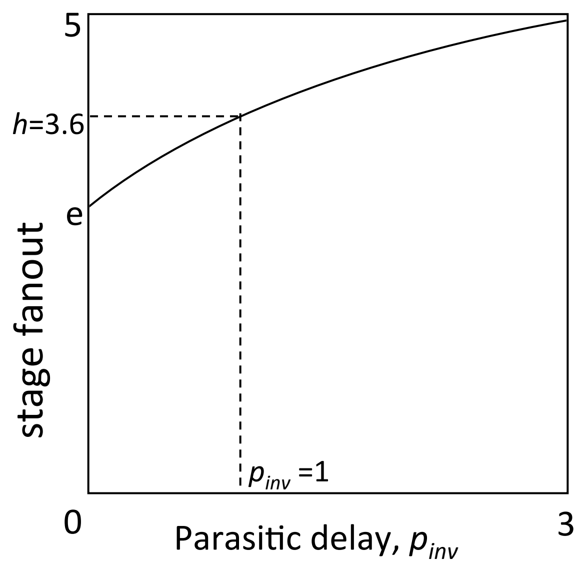
Analytical solution is possible only for p_{inv}=0:

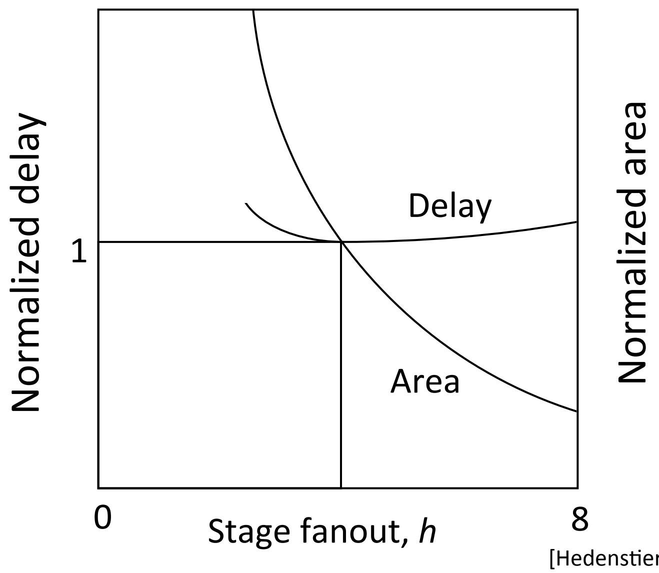
h=e=2.72 which gives N = ln H

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





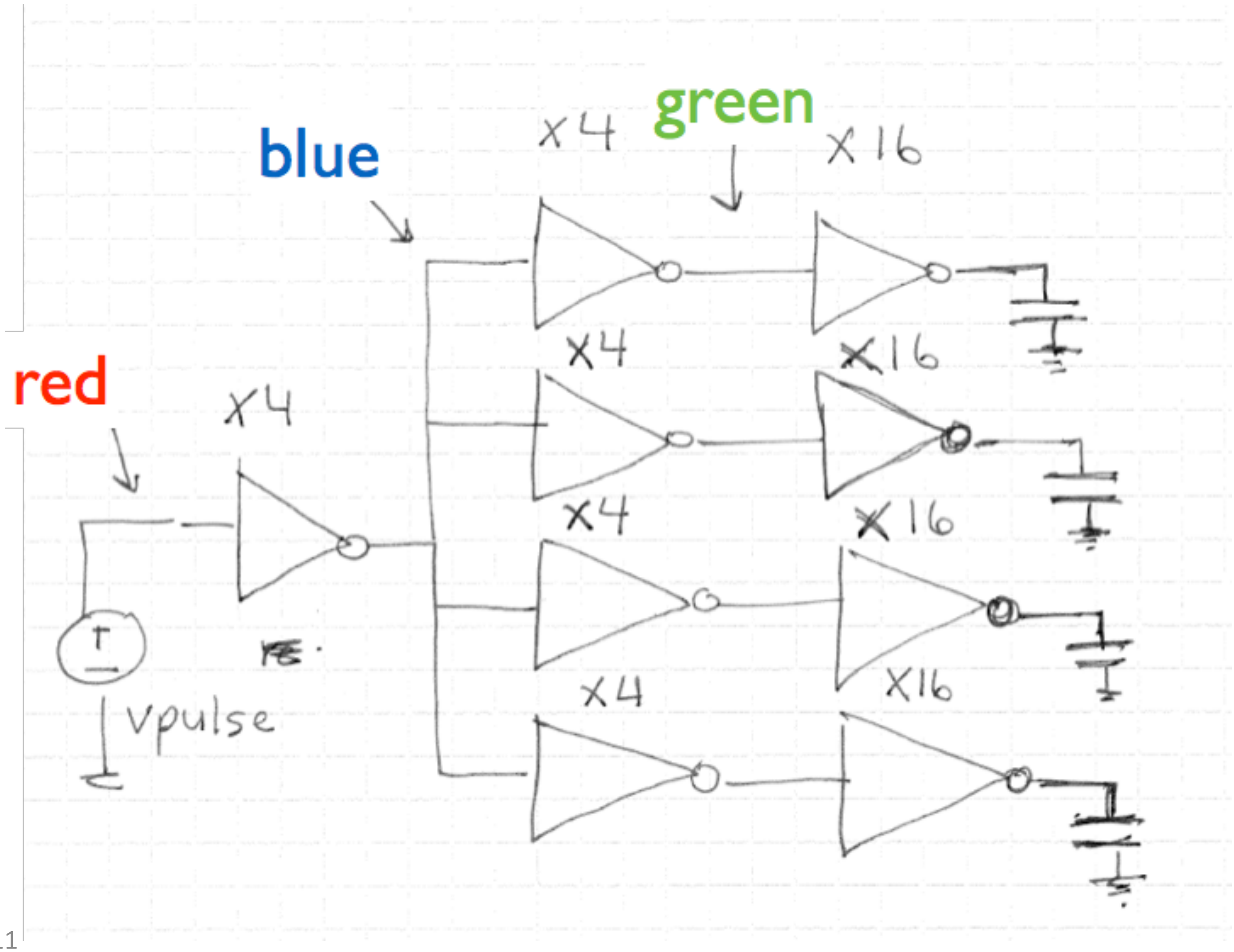
[Hedenstierna & Jeppson 1987]

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

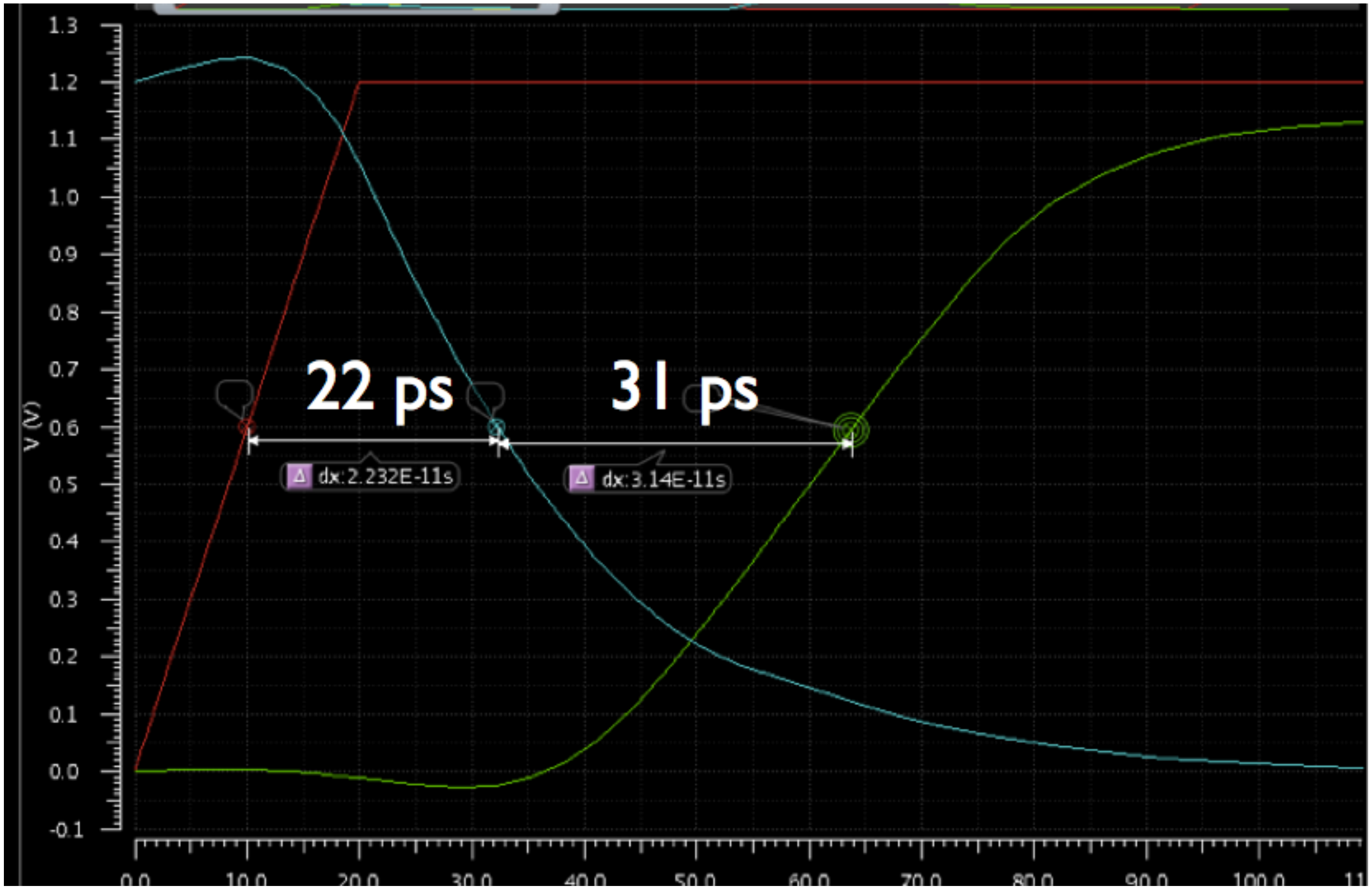
**Note that the propagation delay minimum is rather flat,
while total inverter area on the silicon decreases rapidly when larger stage fanout is used.
Silicon real estate (=cost) can be saved for relatively little loss of speed!**

Postlab 4
Why is the simulated delay in the tapered bufffer longer than calculated?

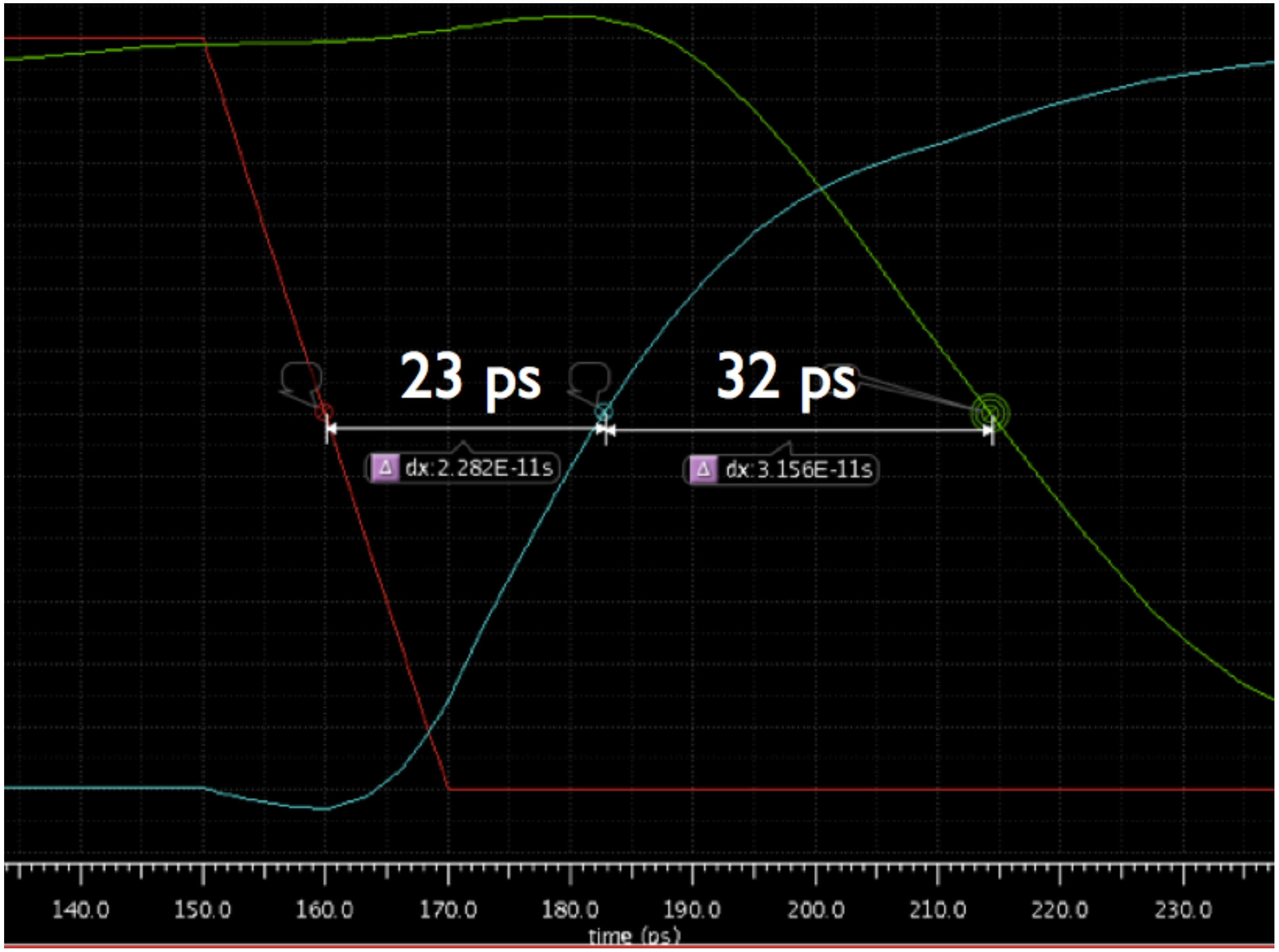
FO4 experiment revisited



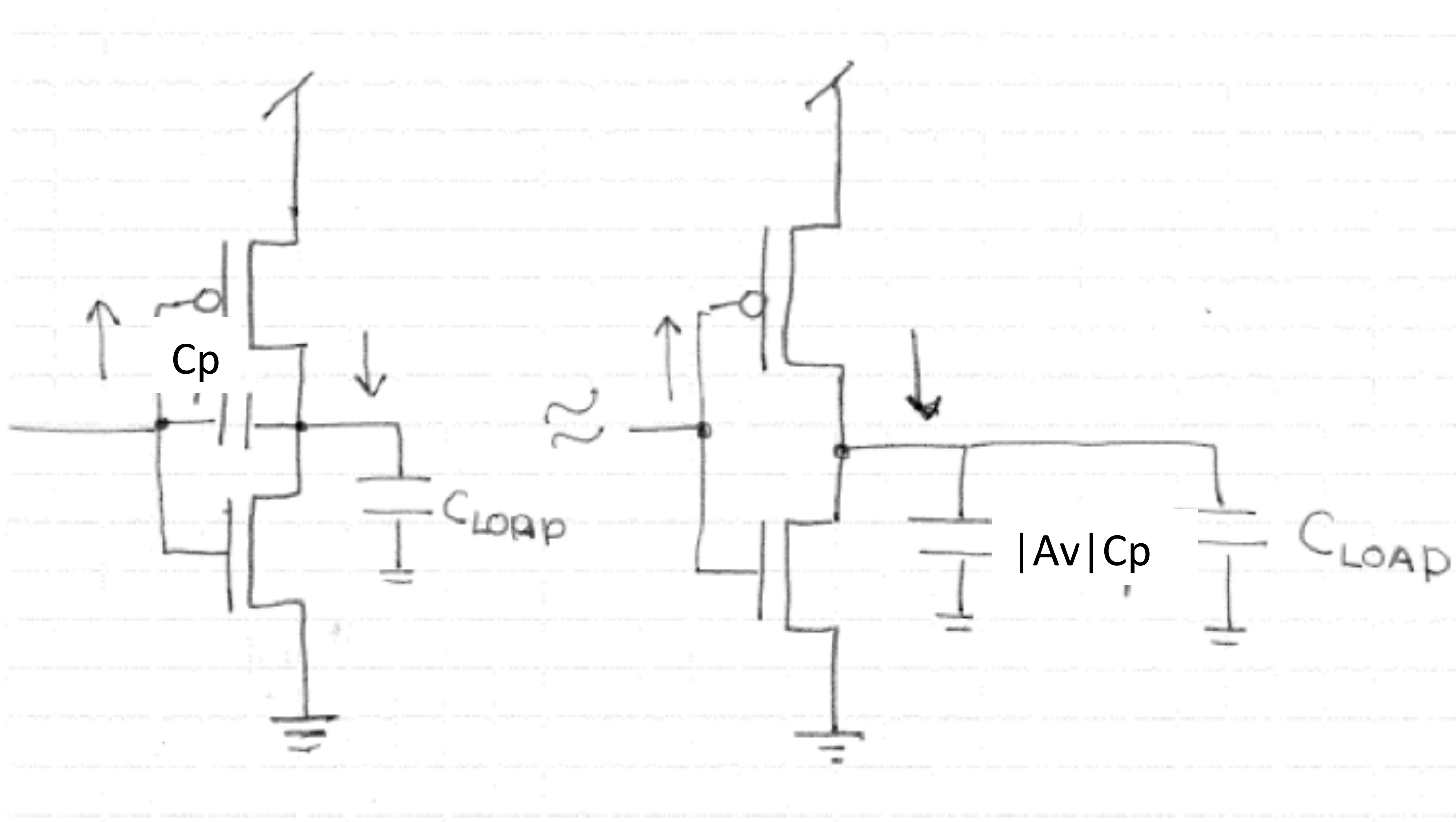
FO4 revisited results



FO4 delay revisited

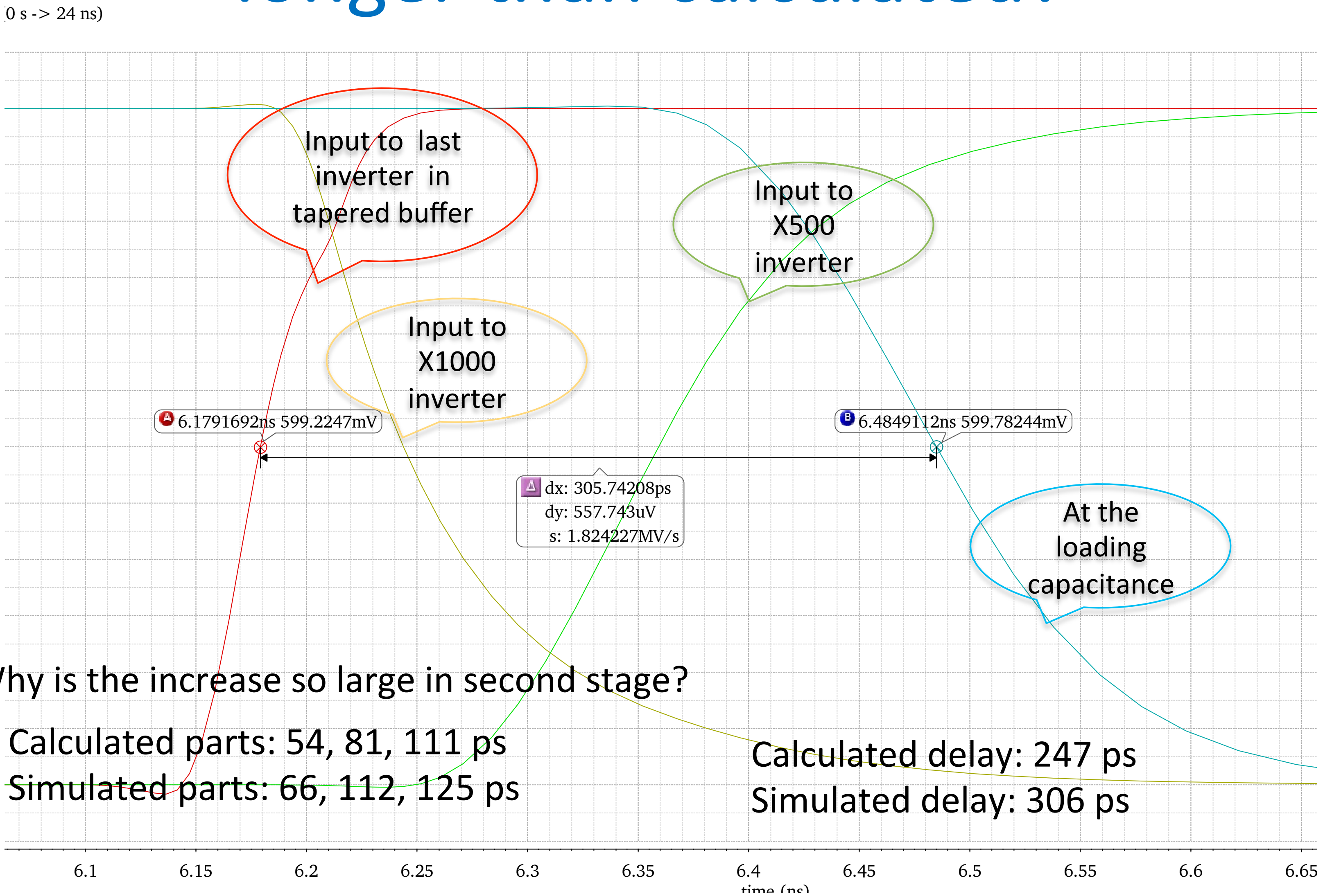


Miller effect or bootstrapping



See Weste & Harris 4.4.6.6

Postlab 4 Why is delay with repeaters longer than calculated?



Why is the increase so large in second stage?

Calculated parts: 54, 81, 111 ps

Simulated parts: 66, 112, 125 ps

Calculated delay: 247 ps

Simulated delay: 306 ps

Adder exercises

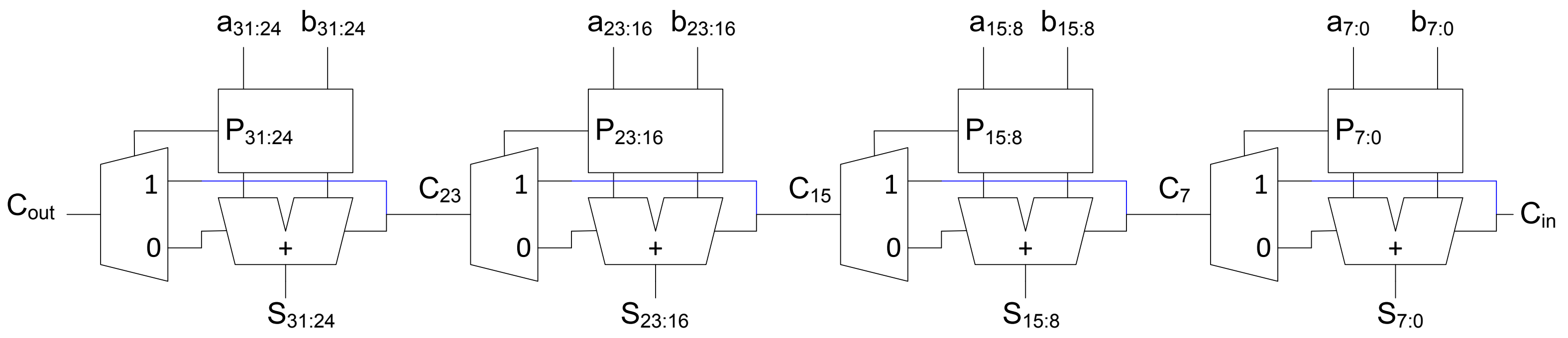
- Exercise 1:
  - Ripple-carry adder/subtractor
  - Ripple-carry adder with PG signals
  - Ripple-carry adder with block P
- Exercise 2
  - Tree carry calculation, prefix adders
    - One example: Sklansky

Carry-Skip Adder

- Carry-ripple is slow through all N stages
- Carry-skip allows carry to skip over groups of n bits
  - Decision based on n-bit propagate signal

N-bit adder

n-bit blocks



k=N/n number of n-bit blocks

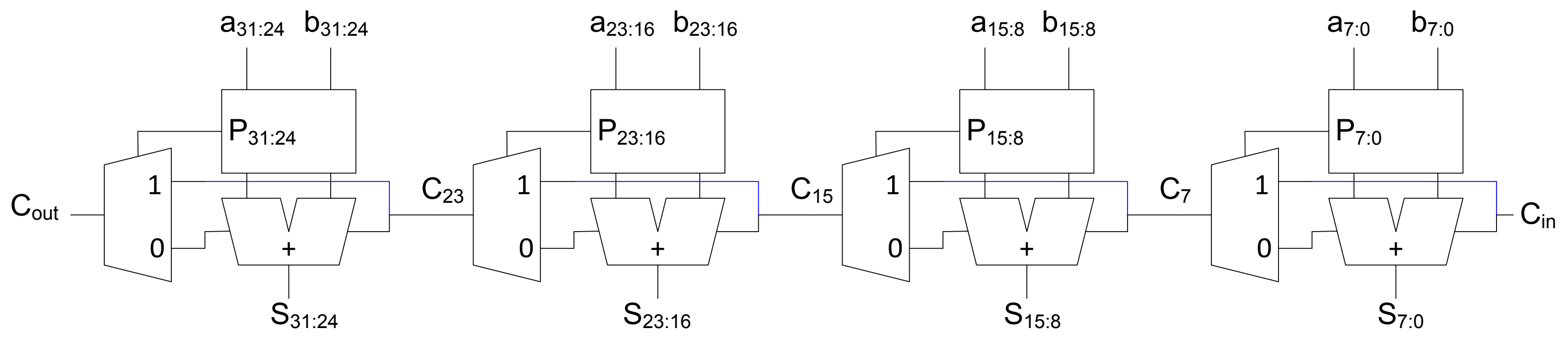
Verify the delay model given in eq. 11.13: $t_{skip} = t_{pg} + 2(n-1)t_{AO} + (k-1)t_{mux} + t_{XOR}$

Carry-Skip Adder

- Carry-ripple is slow through all N stages
- Carry-skip allows carry to skip over groups of n bits
  - Decision based on n-bit propagate signal

N-bit adder

n-bit blocks



k=N/n number of n-bit blocks

For 32-bit adder we had 31×t_{AO}; Now only t_{PG}+14t_{AO} + 3t_{MUX}
Delay is cut in half!