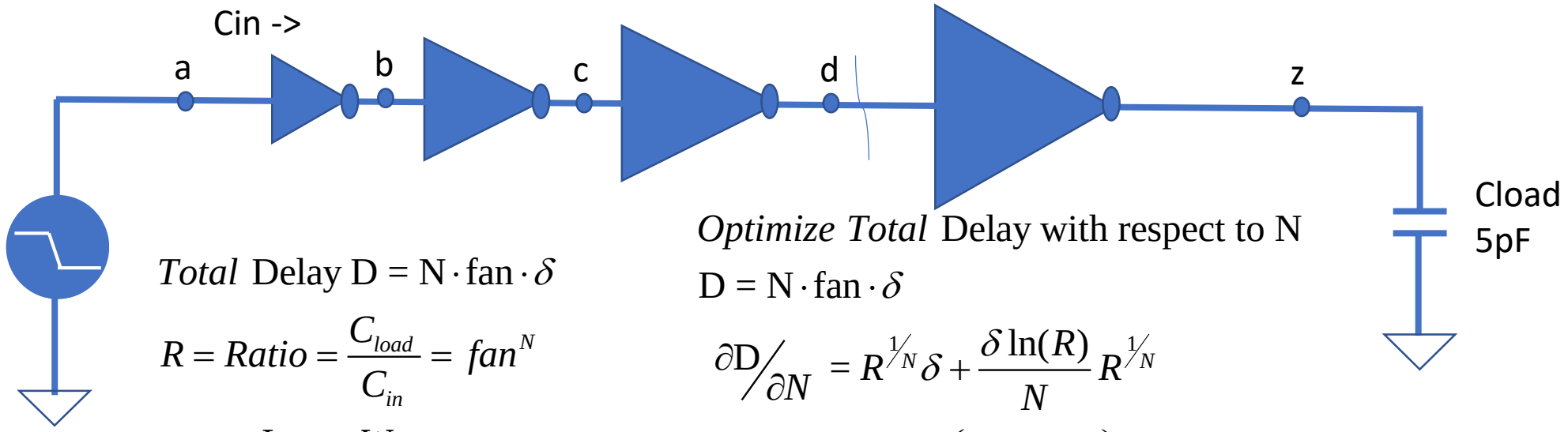
Project
Drive a 3rd party tool using Python to optimize a result

olhartin@asu.edu

Inverter chain problem

- Consider a chain of Inverters (assume an odd number N in the chain)



$$Total\ \text{Delay}\ D = N \cdot fan \cdot \delta$$

$$R = Ratio = \frac{C_{load}}{C_{in}} = fan^N$$

$$fan = \frac{I_{out}}{I_{in}} = \frac{W_i}{W_{i-1}}$$

$$D = N \cdot fan \cdot \delta$$

$$Optimize\ Total\ \text{Delay with respect to N}$$

$$D = N \cdot fan \cdot \delta$$

$$\partial D / \partial N = R^{1/N}\delta + \frac{\delta \ln(R)}{N} R^{1/N}$$

$$\partial D / \partial N = R^{1/N}\delta\left(1 - \frac{\ln(R)}{N}\right) \rightarrow 0$$

$$\ln(R) = N, R = e^N$$

$$fan = R^{1/N} = (e^N)^{1/N} = e,\ \text{ideal answer}$$

The problem

- Assume the width of the first inverter is the minimum value, that the fan defines the factor with which that width must increase for each inverter in the chain in order to drive the next inverter.
- Your objective is to find the configuration with the minimum delay between the transient source and the load capacitor and will be measured by the time difference between when the source goes above 50 Vdd and when the load goes below 50% Vdd.
- Variables are the number of stages N, and the width increase in each stage which we named fan.
- The calculation we did shows that ideally we will need ~2.7 or 3 stages, but this is an ideal answer and your actual answer will be different.

You are given

- InvChain.sp
  - This is a spice file that has 5 inverters. Each is made with an inverter subcircuit composed of matched nmos and pmos in a ---
  - A transient us supplied that goes from 0 to 3 volts and feeds the first inverter in the chain. The load at the end of the inveter chain is a 5pF capacitor.
  - In this spice file a measurement is defined that measures the time difference between the input of the first inverter going above 1.5 volt and when the voltage across the load capacitor falls to 1.5 volt.
- CADhspice.py
  - Python version 3.4 is loaded on the eecad machines (use 10 for now) and use the command -> python3.4 file.py
  - This is a python file that reads the result of the simulation using InvChain.sp and changes the parameters, and circuit in order to optimize the minimum delay.
  - You want to optimize the fan and number of stages.
  - The solution you will get is dependent of the transistors used, and they are coming form cmoslibrary.lib

InvChain.sp hspice input file

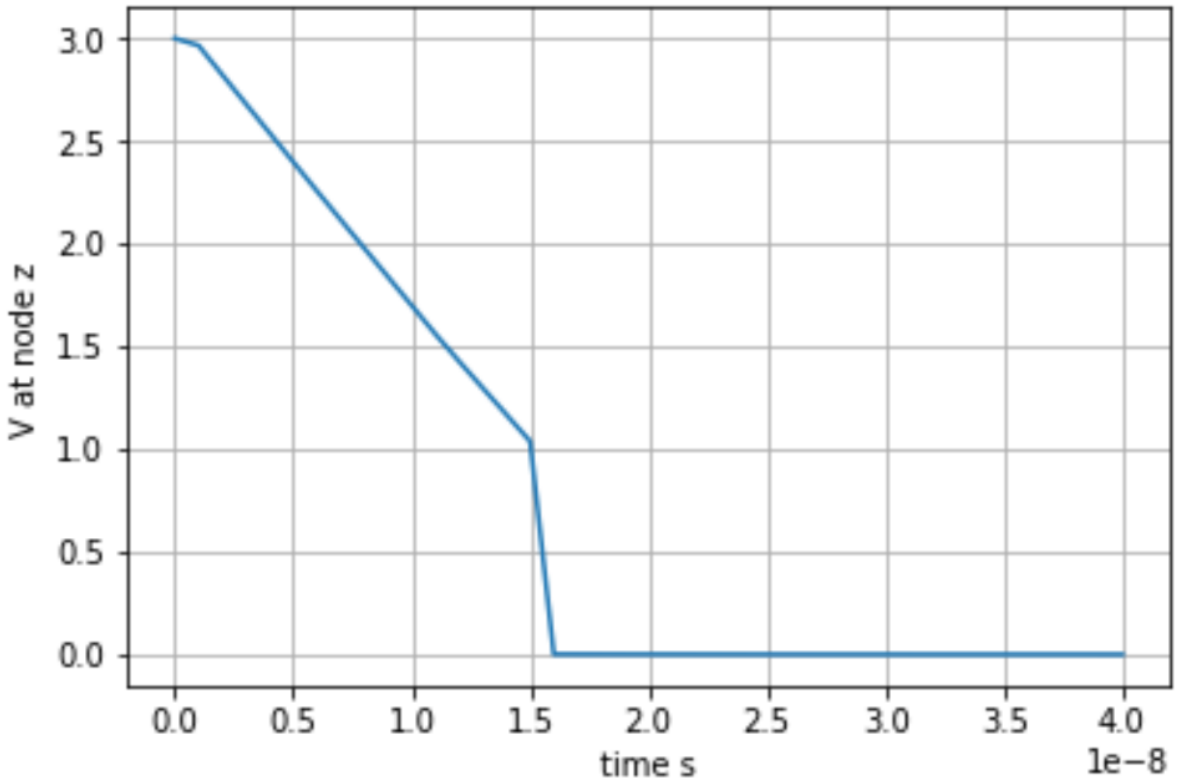
Note the way the nodes are named, a to b
b to c
c to d

and finally always z to 0 across the load capacitor

```
Lab 1 Problem 1A
.lib 'cmoslibrary.lib' nominal                          technology library
.param pvcc = 3                                         VCC for this technology
.param alpha = 1.7                                      pmos is wider by this factor
.param fan = 1                                          fan out parameter which you can change
.global vcc! gnd!
.subckt inv A Z                                         inverter subcircuit
  m1 Z A gnd! gnd! nmos w=1.4u l=0.35u  AD=0.7p                    NMOS
  m2 Z A vcc! vcc! pmos w=(1.4u*alpha) l=0.35u AD= 0.7p*alpha      PMOS
.ends
Xinv1 a b inv M=1                                       Inverter chain with multiplier M 1 and then
Xinv2 b c inv M=fan                                     increasing width by a factor of fan on each
Xinv3 c d inv M=fan*fan                                 stage
Xinv4 d e inv M=fan*fan*fan
Xinv5 e z inv M=fan*fan*fan*fan                         the number of these stages is dependent on N
Cload z gnd! 5pF
Vin a gnd! 0V PWL 0 0NS 1NS 3 20NS 3                    supplies the transient source which goes from 0 to 3 V
.measure TRAN tphl_inv  TRIG v(Xinv1.a) VAL = 1.5 RISE = 1 TARG v(Xinv5.z) VAL=1.5 FALL = 1     Measures result on last inv
Vgnd gnd! 0 DC = 0
Vvcc vcc! 0 DC = 3V
.tran 1NS 40NS                                          Runs the transient problem for 40 ns
.print tran v(a) v(z)
.OPTION MEASFORM=3
.OPTION POST
.TEMP 25
.end
```

Test run

- hspice InvChain.sp
- read the output file with python, and parse out the result, and plot
- We find that it drops to 1.5 V or half VCC(3V) in just under 11 ns for a fan of 1 and N = 5
- You must get this to run first




```
1.09418e-08 s
```

Guidance

- Change CADhspice.py so that it does this

  - run InvChain.sp
  - run hspice using InvChain.sp as an input deck
  - read the resulting tphl_inv from output
  - change the InvChain.sp deck to look for a better value of tphl_inv

methods for changing the hspice input deck

- Most obvious and slowest method, read the file and use regular expressions to find the lines to change and change them
  - this will work well but file reads, and writes are slower than everything else
- Another way to go is to store key parts of the deck in a list and write the list
  - And create the parts that do change based on the input parameters for that case
  - this may be faster because instead of a read and a write, it is just a write
  - remember you will evaluate many cases so the speed of evaluating a case is important, and
    - in some problems determines whether you can use the method or not

Methods for finding best N and fan

- Method 1, most straight forward
- you are looking for best values of fan, and N
- Only odd integers for N are possible, which means you are looing at 1, 3, 5, 7, 9, 11 maybe a short list
- In the real world you would know what widths are available in your technology. We will assume the fan parameter must be an integer. Large numbers would be a problem so we can guess that the values might be limited to 1, 2, 3, 4, 5, 6, 7 bearing in mind that we calculated fan = e = 2.718. A reference suggest a real answer might be 5.8.
- This is 42 points. We should be able to tell if there is a minimum in this range and re-center.

Method 1

- for each case
  - ➢write the Hspice file with N inverters, and fan parameter set
  - ➢run hspice on this file
  - ➢read the resulting output file parsing out the tphl parameter
- You can read an initial file and then update it making changes or write the entire file making changes to appropriate lines as you go

Less straight forward but more interesting

- Method 2
- write a function
  - def tphl_(N,fan):
  - write the Hspice file with N inverters, and fan parameter set
  - run hspice on this file
  - read the resulting output file parsing out the tphl parameter
  - return(tphl)
- then you can use an optimizer of your own construction, or in scypi.optimize (make sure it is loaded on this server up front, don't assume it is) which calls this function
- You must constrain N to odd integers, and fan to integers, which will be more difficult in this approach

- This is probably a better method and the function is a good idea, no matter how you do it but scipy.optimize has not yet been loaded on the server.

This is what I did

- N is the number of inverters, fan is the width increase of each stage
- I created a function that determines tphl based on N and fan like this

```
def timeinvchain(N,fan):
##          function writes an hspice input file using these parameters n, fan (see slide 5)
        generatehspicedeck(N,fan,'InvChain00.sp')
##          command runs hspice on this input file
        p=subprocess.Popen(["hspice","InvChain00.sp"], stdout=subprocess.PIPE)
##          standard out is rerouted to p.stdout which can be parsed like
##          lines read in a file to find the parameter tphl
        tphl = readstdout(p.stdout)
        return(tphl)
```

Now you can use this function to optimize tphl based on parameters N and fan

Use python 3.6

- system eecad10
- [userid@eecad10 ~]$ scl enable rh-python36 bash
- [userid @eecad10 ~]$ which python
- /opt/rh/rh-python36/root/usr/bin/python
- [userid @eecad10 ~]$ python

- Now you can use optimize.fmin_bfgs(….)
- the only thing you have to deal with is that the optimizer doesn’t knw the restrictions on N and fan

Deliverables

1) your code (.py file)

2) an InvChain.sp file generated by your code (this is a text file)

3) an output generated by Hspice (this is a text file)

4) an image grab of your output showing your optimized values for tphl, N, and fan (as an image this should be in pdf format)

5) these values of tphl, N, and fan, and a description of how your code found them, be detailed. This should be a paragraph with at least 4 sentences, but not more than 6 sentences.