Felix_Chen_Lab4

Tags

Part 1

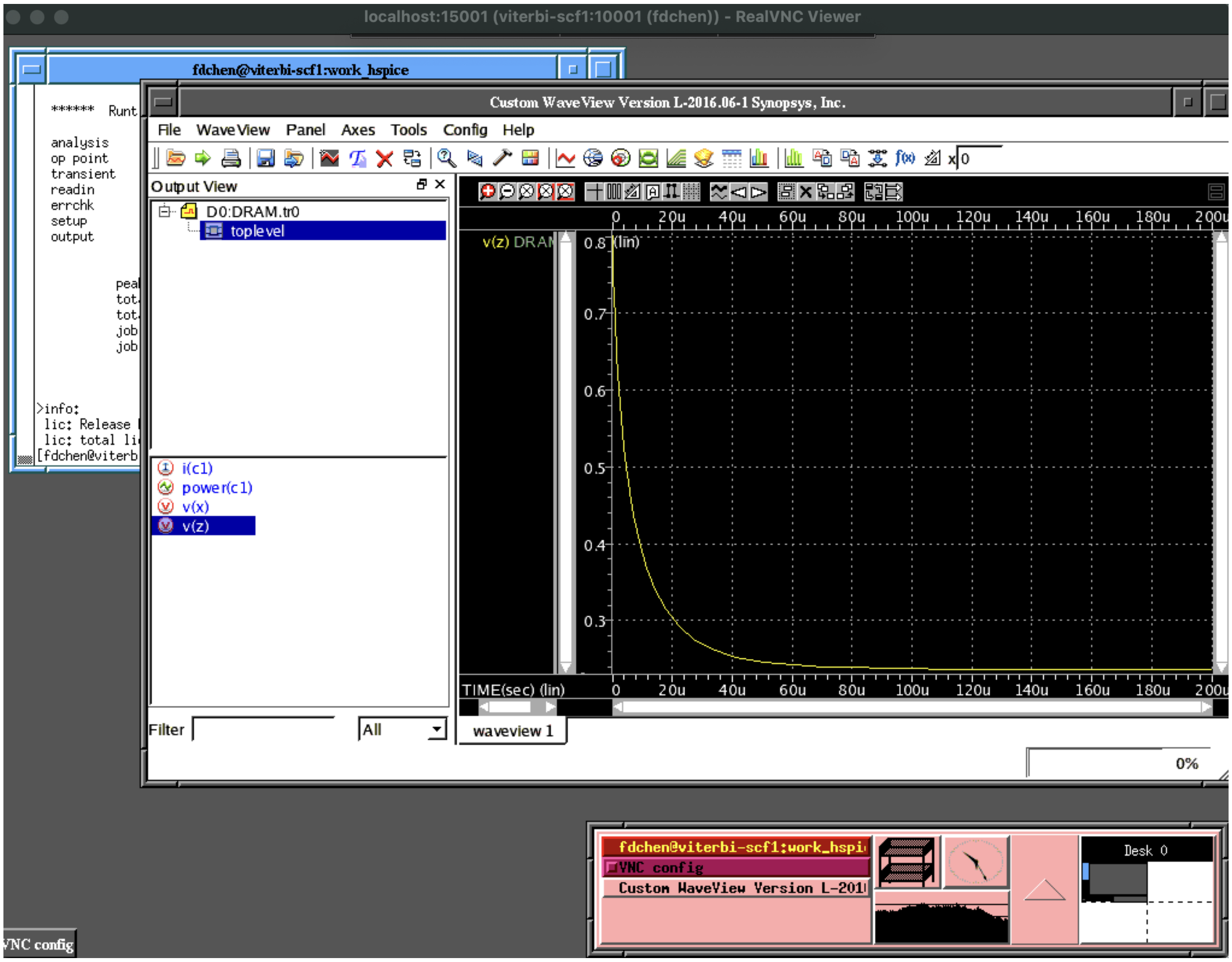
DRAM

The hspice file

Lab4 > DRAM.sp

```
** Simple DRAM using PTM 45 nm Node

.include CMOSP.inc
.include CMOSN.inc

.PARAM VDD = 0.8

** Circuit Netlist

** Supply and Input Sources
VSUP X 0 'VDD/2'
VG Y 0 0

** Transistors
M1 Z Y X VDD CMOSN L=45n W=120n

** Capacitor
C1 Z 0 C=10f

** Initial condition
.IC V(Z) = 'VDD'

** Analysis Setup
.TRAN 0.001u 800u

** Control Information
.OPTION POST BRIEF NOMOD PROBE MEASOUT

** Print and Measurement
.PRINT V(X) V(Z)
.MEASURE TRAN RTL TRIG AT=0 TARG V(Z) VAL=0.6 FALL=1
.MEASURE TRAN AVG_CUR AVG I(C1) FROM 0 to 'RTL'
.MEASURE TRAN AVG_PWR AVG P(C1) FROM 0 to 'RTL'

.END
```

C=0.1fF

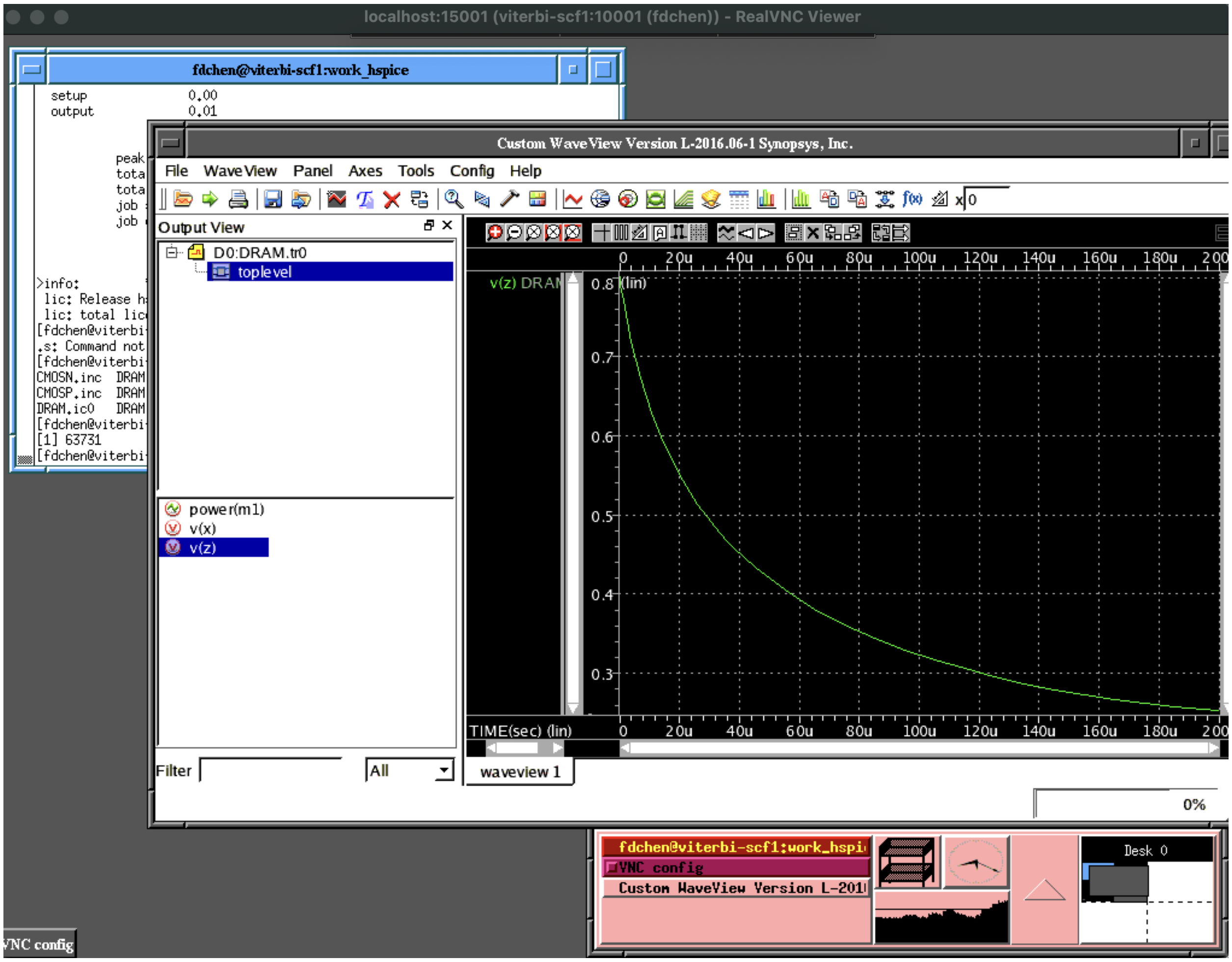


RTL = $2.151e^{-6}$

average power = $-6.546e^{-12}$

average current = $-9.342e^{-12}$

C=1fF

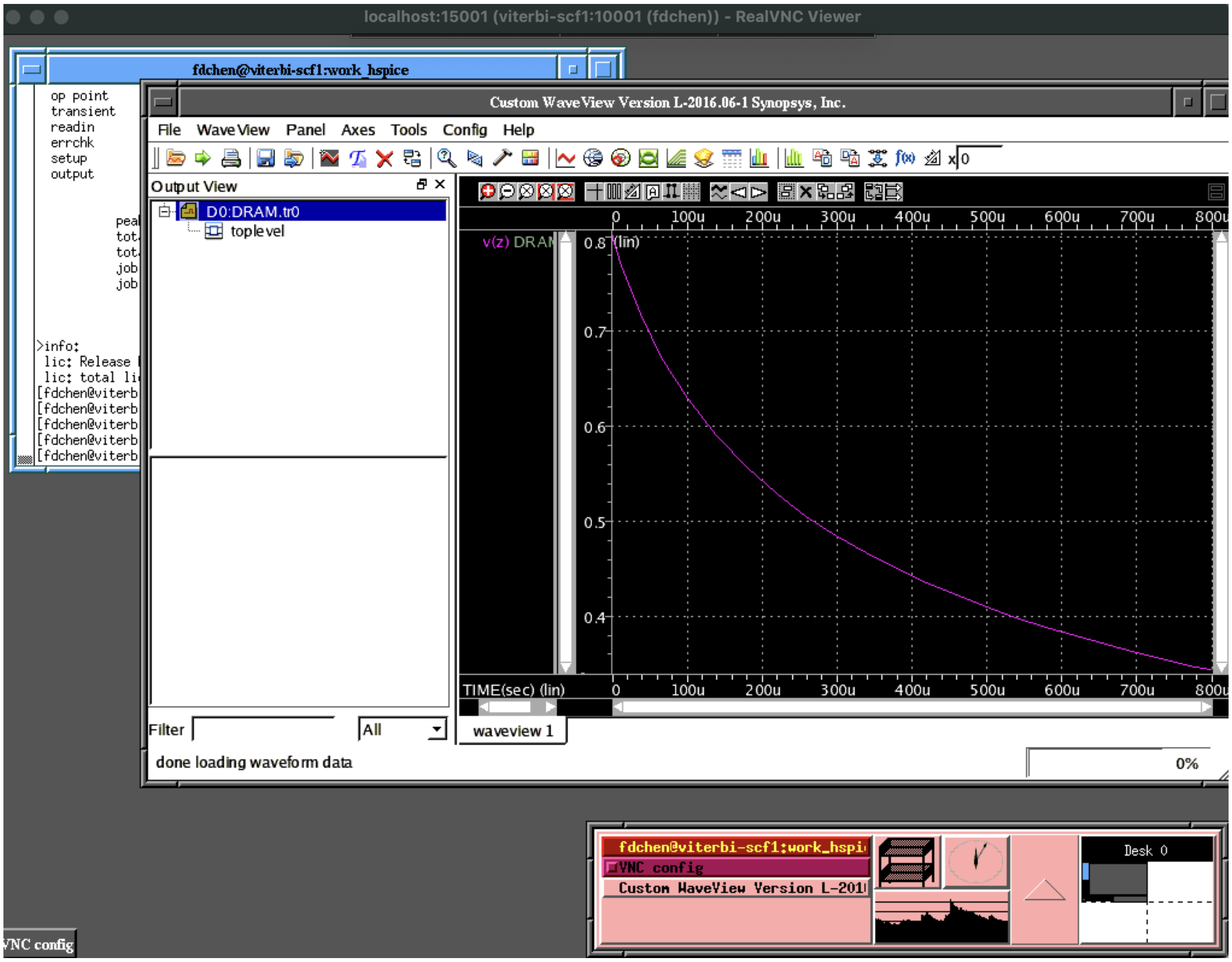


RTL = $1.370e^{-5}$

average power = $-1.024e^{-11}$

average current = $-1.457e^{-11}$

C=10fF



RTL = $1.29e^{-4}$

average power = $-1.093e^{-11}$

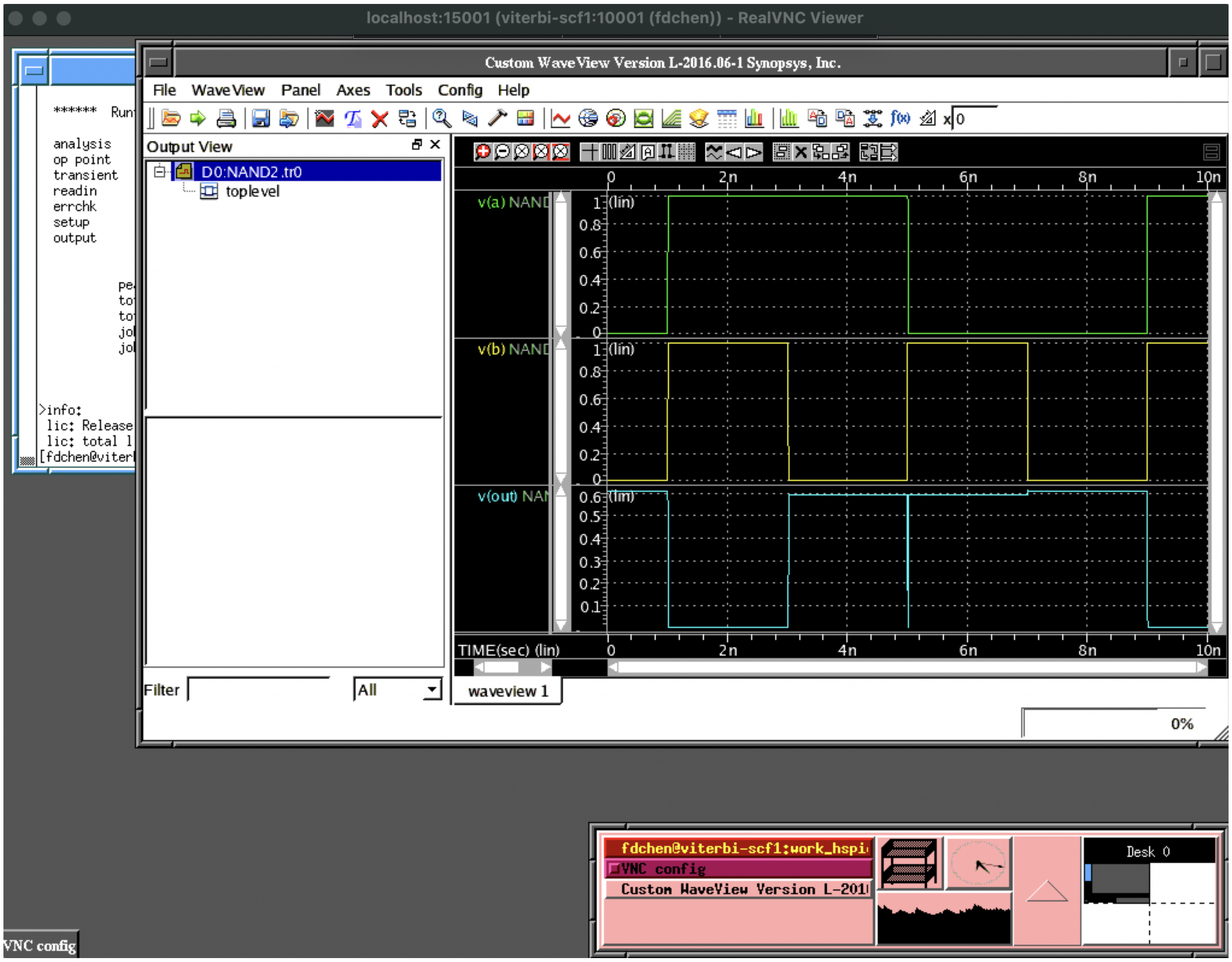
average current = $-1.557e^{-11}$

Observation

The higher the capacitance of the capacitor is, the longer it takes to discharge. This makes sense because capacitors with the most capacitance holds the most charge for the same voltage.

NAND2

Waveform


localhost:15001 (viterbi-scf1:10001 (fdchen)) - RealVNC Viewer
Custom WaveView Version L-2016.06-1 Synopsys, Inc.
File WaveView Panel Axes Tools Config Help
Output View
D0:NAND2.tr0
toplevel
analysis
op point
transient
readin
errchk
setup
output
>info:
lic: Release
lic: total l
v(a) NAND
v(b) NAND
v(out) NAN
TIME(sec) (lin)
Filter
All
waveview 1
0%
fdchen@viterbi-scf1:work_hspi
VNC config
Custom WaveView Version L-201
Desk 0
VNC config

File

CMOSN.inc | DRAM.sp | NAND2.sp | inverter.sp | RC.sp

Lab4 > NAND2.sp

```
** Simple DRAM using PTM 45 nm Node

.include CMOSP.inc
.include CMOSN.inc

** Circuit Netlist

** Supply and Input Sources
V_SUP VDD 0 1
VPIN_A A 0 PULSE 0 1.0 1ns 10ps 10ps 4ns 8ns
VPIN_B B 0 PULSE 0 1.0 1ns 10ps 10ps 2ns 4ns

** Transistors
M_PU_A OUT A VDD CMOSP L=45n W=180n
M_PU_B OUT B VDD CMOSP L=45n W=180n

M_PD_A OUT A X CMOSN L=45n W=240n
M_PD_B X B 0 CMOSN L=45n W=240n

** Initial condition
.IC V(X) = 0

** Analysis Setup
.TRAN 0.1ns 10n

** Control Information
.OPTION POST BRIEF NOMOD PROBE MEASOUT

** Print and Measurement
.PRINT V(A) V(B) V(OUT)

.END
```

Part 2

DRAM.sp

```
** Simple DRAM using PTM 45 nm Node

.include CMOSP.inc
.include CMOSN.inc
.include input_data.txt

.PARAM VDD = 0.8

** Circuit Netlist

** Supply and Input Sources
VSUP X 0 'VDD/2'
VG Y 0 0

** Transistors
M1 Z Y X VDD CMOSN L=45n W=120n

** Capacitor
C1 Z 0 C=10f

** Initial condition
.IC V(Z) = 'VDD'

** Analysis Setup
.TRAN 1u 200u sweep data = mydata

** Control Information
.OPTION POST BRIEF NOMOD PROBE MEASOUT

** Print and Measurement
.PRINT V(X) V(Z)
.MEASURE TRAN RTL TRIG AT=0 TARG V(Z) VAL=0.6 FALL=1
.MEASURE TRAN AVG_CUR AVG I(C1) FROM 0 to 'RTL'
.MEASURE TRAN AVG_PWR AVG P(C1) FROM 0 to 'RTL'

.END
```

Explanation

- DRAM.sp uses input data from file input_data.txt to run sweep analysis on DRAM with varying levels of temperature, width, and capacitance and logs it's result.

Python scripting files

generate_input.py

- Run this file to generate the input data to do the sweep analysis with

- This file will create a file called “input_data.txt”, which DRAM.sp will use to do sweep analysis

```python
with open('input_data.txt', 'w') as file:
    file.write(".DATA mydata\n")
    file.write("+ TEMP WIDTH CAP\n")
    temp_list = [25, 85]
    width_list = list(range(120, 301, 60)) # stops at 301 to include 300
    cap_list = [0.01] + [8 ** (0.1 * i) for i in range(1, 11, 1)]
    for temp in temp_list:
        for width in width_list:
            for cap in cap_list:
                file.write(str(temp) + " " + str(width) + "n " + str(cap) + "f\n")
    file.write(".ENDDATA")
```

Explanation

This file just generates different combinations of input condition by using for loops, and outputs it in a .txt file.

organize_output.py

- Run this file after you’ve ran the sweep analysis in DRAM.sp and saved the output in a file called “output.lis”
- This file reads “output.lis”, and “input_data.txt”, and generates a new file “output_data.txt” which visualizes the result of the sweep analysis in a nice and readable way

DRAM.sp | NAND2.sp | generate_input.py | organize_output.py M | README.md U | output_data.tx

Lab4 > organize_output.py

```python
from collections import deque

# The final file, line by line
final_result = []

with open('output.lis', 'r') as file:
    with open('input_data.txt', 'r') as input_data:
        # Read in the input data in order
        for line in input_data:
            if len(line) != 0 and line[0] != '.' and line[0] != '+' and line[0] != "\n":
                final_result.append(line[:-1])

    # Read in data
    counter = 0
    # Have a buffer, because rtl, average power and current are usually within 5 lines of "job finished"
    buffer = deque()
    for _ in range(5):
        buffer.append("")
    for line in file:
        # If key line "job concluded" is encountered, look into buffer to get the average currents and such
        if "job concluded" in line:
            while len(buffer) != 0:
                temp = buffer.popleft()
                temp_list = temp.strip().split(" ")
                hot_words = ["rtl=", "avg_cur=", "avg_pwr="]
                if len(temp_list) > 1 and temp_list[0] in hot_words:
                    final_result[counter] += (" " + temp_list[1])
            counter += 1
            # Repopulate the buffer
            for _ in range(5):
                buffer.append("")
        buffer.append(line)
        buffer.popleft()

# Update final_result to add in start and finish
final_result = [".DATA mydata", "+ TEMP WIDTH CAP RTL AVG_CUR AVG_POWER"] + final_result + [".ENDDATA"]

# Write the data out
with open("output_data.txt", 'w') as file:
    for line in final_result:
        file.write(line + "\n")
```

Explanation

This file reads “output.lis” and “input_data.txt” to generate “output_data.txt” which a readable output file with all the results.

- I used a buffer method because I realized the RTL, average power, and average current are always 5 lines before a line that contains “job concluded”. So I used that fact to read in those data from “output.lis”

How to use

This program basically runs a sweep analysis using HSPICE on a DRAm cell using differing combinations of temperature, width, and capacitance to find out what their resulting RTL, average power, and average current is.

In order to run this program do:

1. python3 generate_input.py
2. hspice -i DRAM.sp > output.lis
3. python3 organize_output.py

Then you can find the output in the file: “output_data.txt”

Explanation:

(1) generates the combinations of the vary temperature, width, and capacitance conditions. (2) Runs the hspice program on those combinations and logs it in output.lis file. (3) Reads the output files and organizes the output in a readable manner in “output_data.txt”