



Advanced Simulations

Prof. Dr. P. Fischer

Lehrstuhl für Schaltungstechnik und Simulation
Technische Informatik der Uni Heidelberg



What we know so far

- **DC Simulation**
 - Determine g_m , r_{ds} , ...
 - Find operation points, dynamic range, ...
- **Transient Simulation**
 - Functionality
 - Risetimes
 - Large Signal behaviour
- **AC Simulation**
 - Good to assess bandwidth / speed as function of parameters (results can be better extracted than in transient)
 - Faster simulation
 - Only for small subcircuits
- **Mixed Mode**
 - Digital blocks are described easier / sim runs faster



Further Simulation Tools

- Parameters, Simulation Settings
- Waveform Calculator

- AC Noise
- Transient Noise

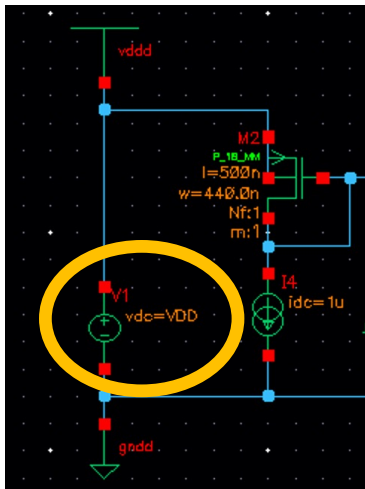
- Corner Simulation
- Monte Carlo Simulation

- Extracted Simulation

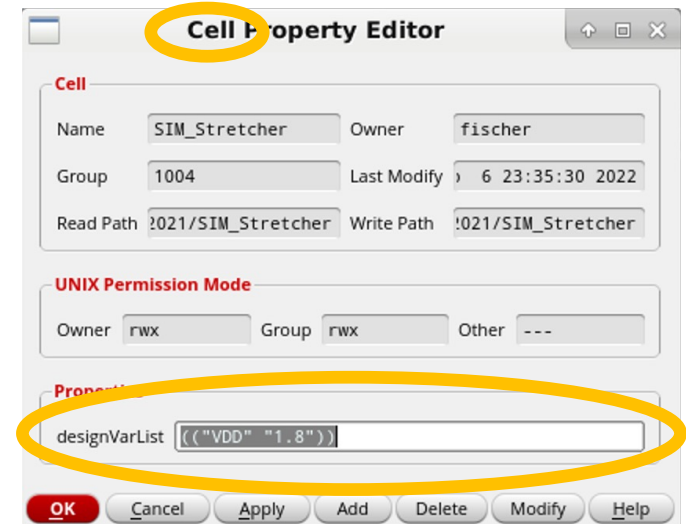
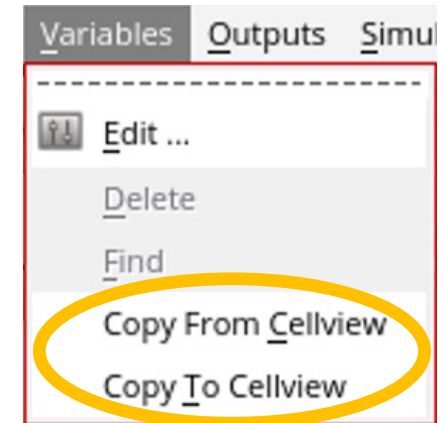
- Scripted Simulations with Ocean



Parameters



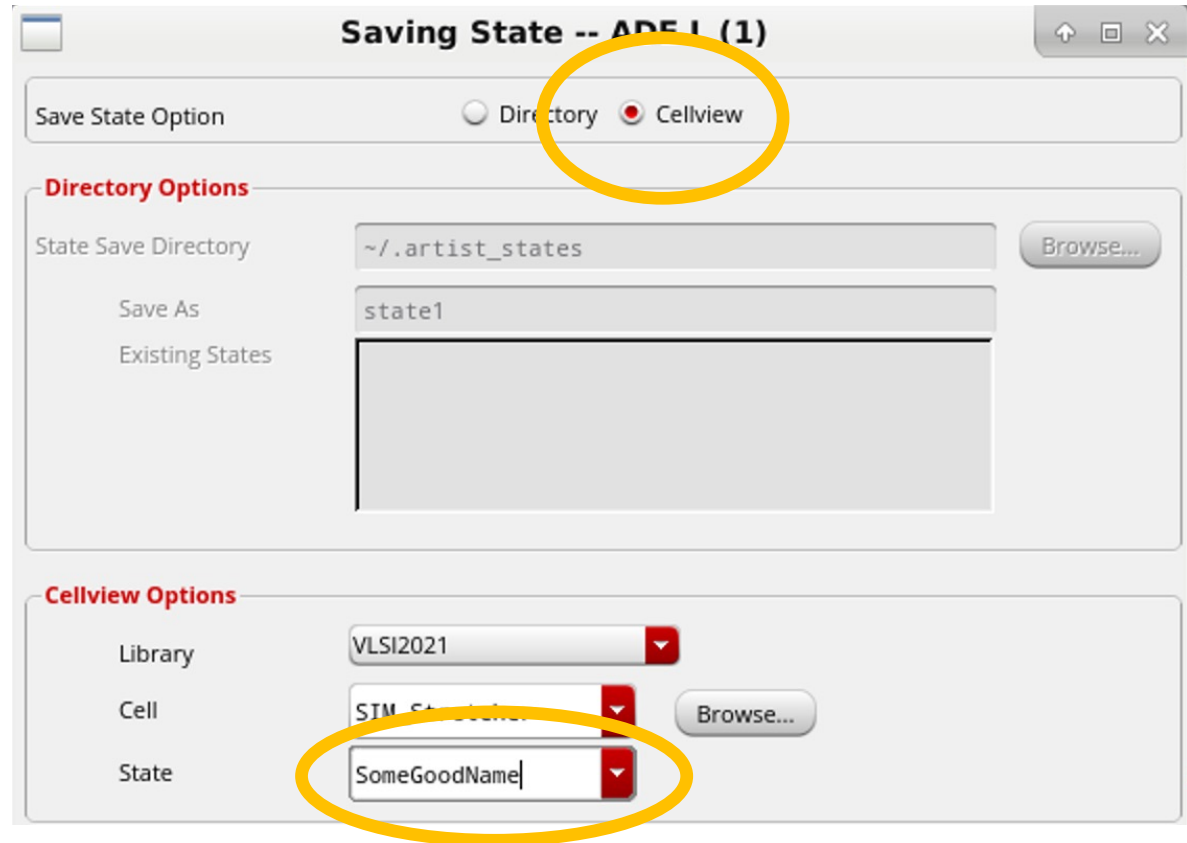
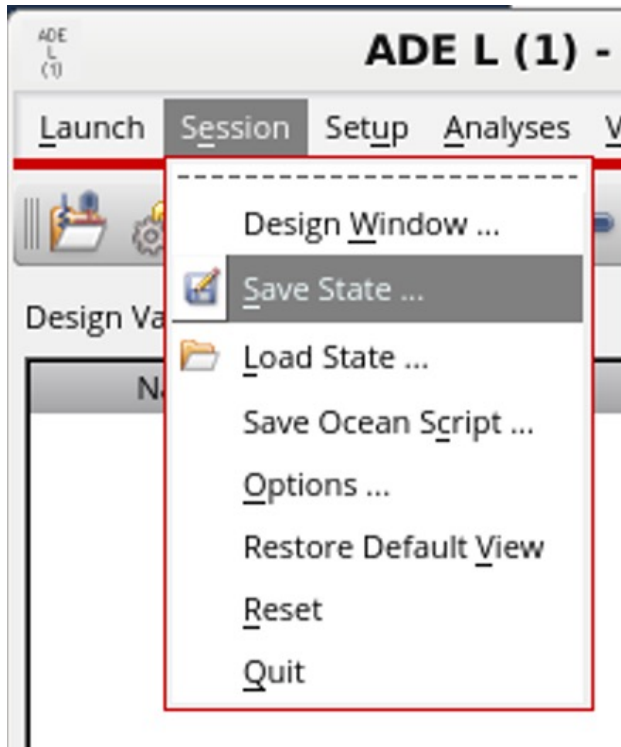
- If parameters are used in a schematic, they can be pulled to the simulator with Variables->Copy From Cellview
- Write them back to cell with Variables->Copy To Cellview
- If you do not use a parameter in the schematic any more, it is still stored and shows up when you get the parameters.
 - Delete it in the simulation window and write back to schematic
- Parameters are stored in the *cell*, not in a view (i.e. they are shared between views).
 - They can be seen in the library manager as properties of the cell (right click cell -> Properties)





Simulation Settings

- Can be saved with Session->Save State
- Best save to *Cellview* with a clear name

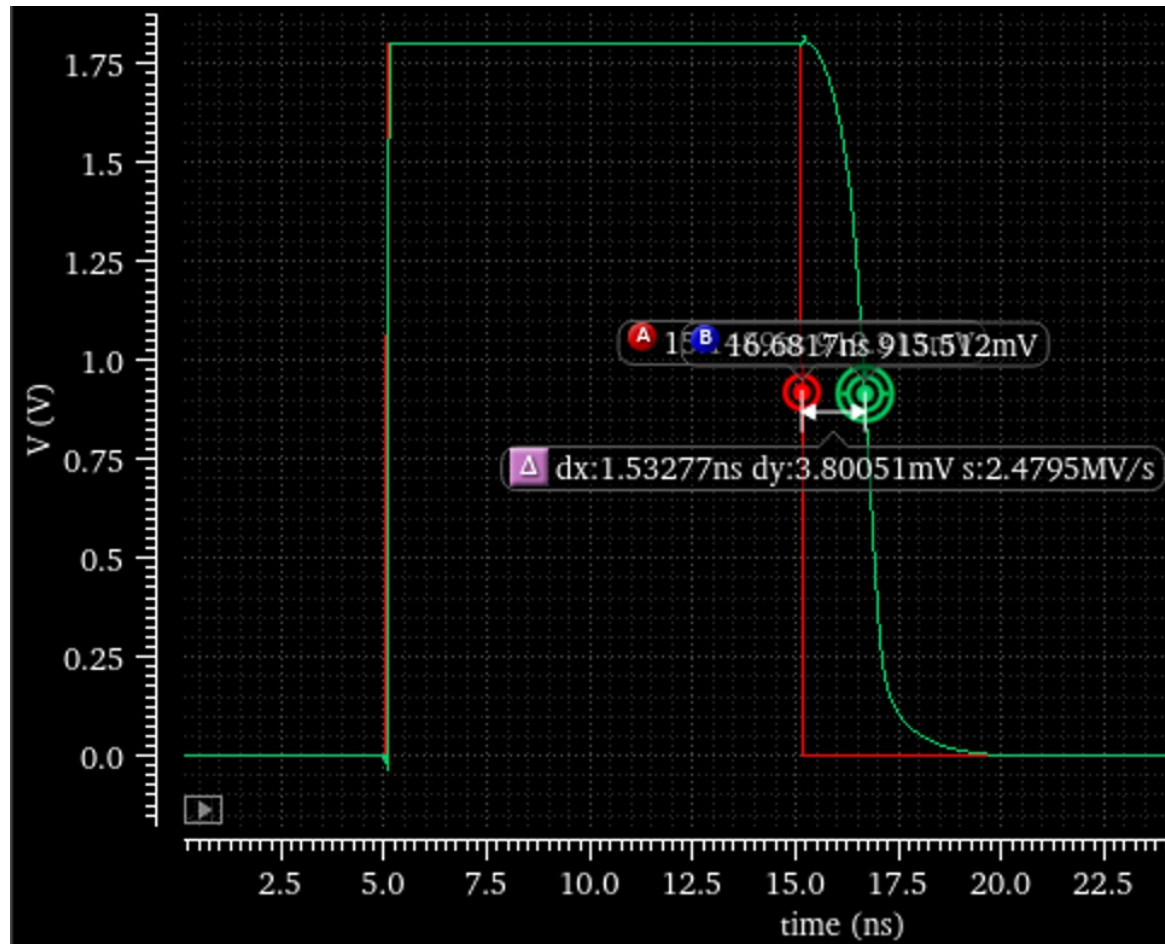




Waveform Viewer

Cursor

- “a” -> create cursor at position
- “b” -> differences are shown

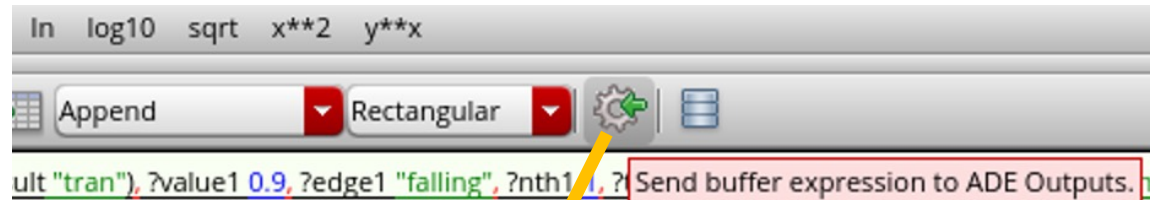




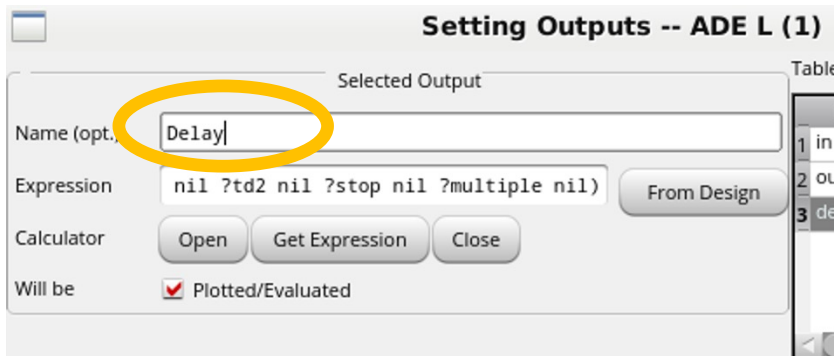
Waveform Calculator

- Can be used to analyse waveforms

- Click on wave
- open calculator



- Fix expression (e.g. derivative)
- Best send to ADE (button), so that it is conserved
- Assign a useful name in ADE



Name/Signal/Expr	Value	Plot	Save	Save Options
1 in		<input checked="" type="checkbox"/>	<input type="checkbox"/>	allv
2 out		<input checked="" type="checkbox"/>	<input type="checkbox"/>	allv
3 Delay	1.54n	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

- NB: simulation variables can be accessed with
 - `pv("/name" "value" ?result "variables")`
- Calculated expressions can be referred to just by the name

Maybe also:
VAR("/lbias")
VDC("/netname")



FURTHER SIMULATION TYPES



AC Noise

- In this simulation type, the components generate noise voltages and currents (as function of frequency)
- The 'output signal' then also has noise, coming from various sources in the circuit
 - Total noise is calculated by integration over all frequencies.
- Details:
 - The (spectral) noise density depends on the components
 - It takes 'text book' values e.g. for resistors, these can be overridden
 - Components can be made 'noiseless' so that the effect of individual components can be studied
 - MOS devices need a noise model, which contains in particular the frequency dependent $1/f$ noise
 - Models are often unreliable....
 - Dependencies on component parameters (L) often wrong.



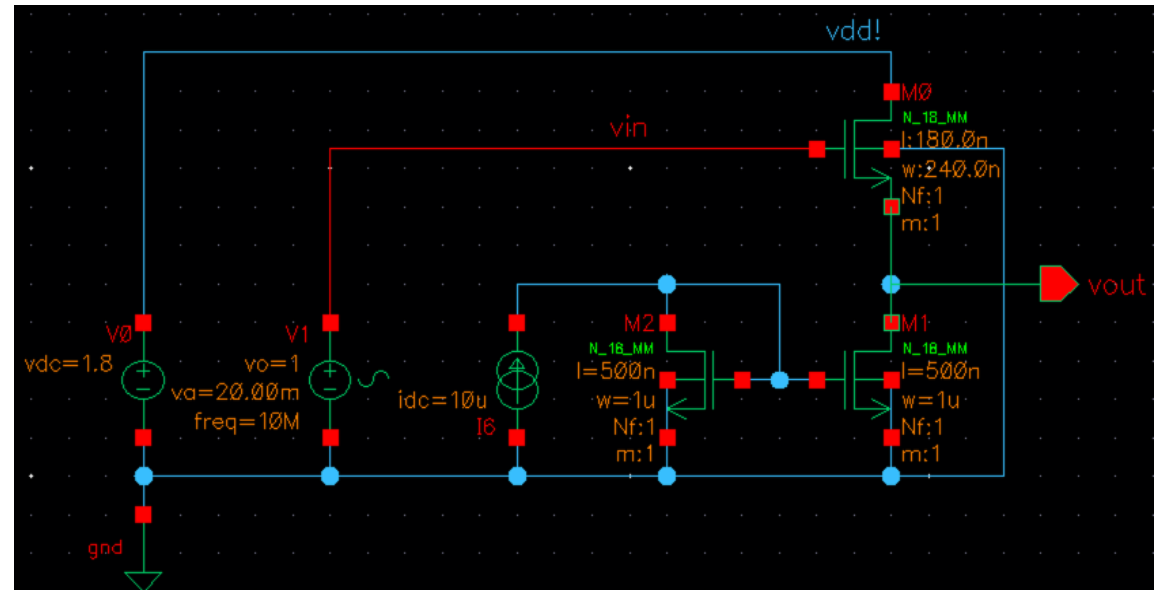
Transient Noise

- AC Noise simulation has drawbacks:
 - It is 'abstract' – needs clear understanding of what 'AC' is..
 - Needs careful analysis (integration)
 - It's **small signal** simulation. Large signal circuits (e.g. comparators) are hard to simulate
- Noise can also be generated in the time domain -> transient Noise analysis
 - 'Easy to use' (but noise sources must again be set up correctly!)
 - 'direct result'
 - Very slow, because time steps must be very small to take into account high frequency noise.
 - Simulators can do no 'tricks' to increase time steps because all signals change all the time (from their noise...)

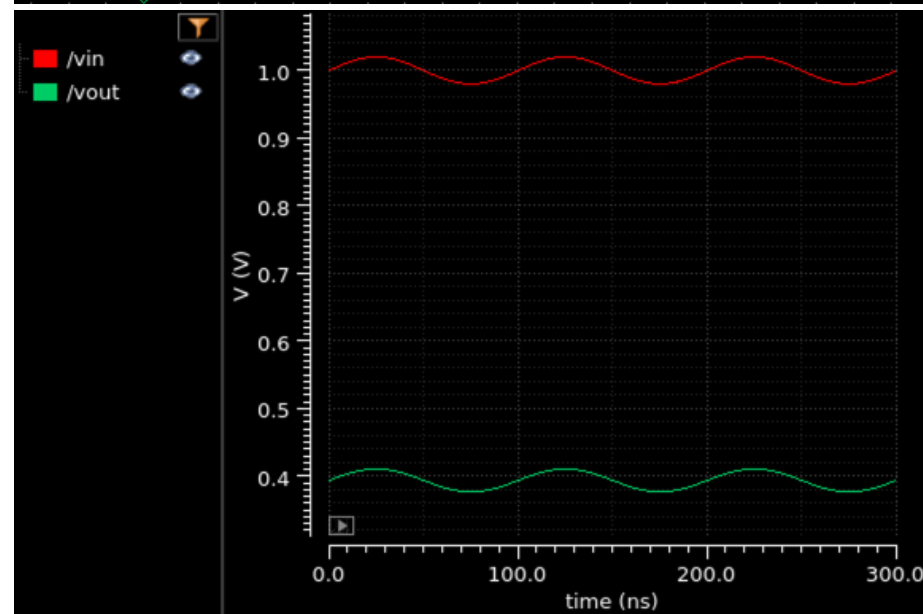


Transient Noise: Example

- Consider a source Follower:



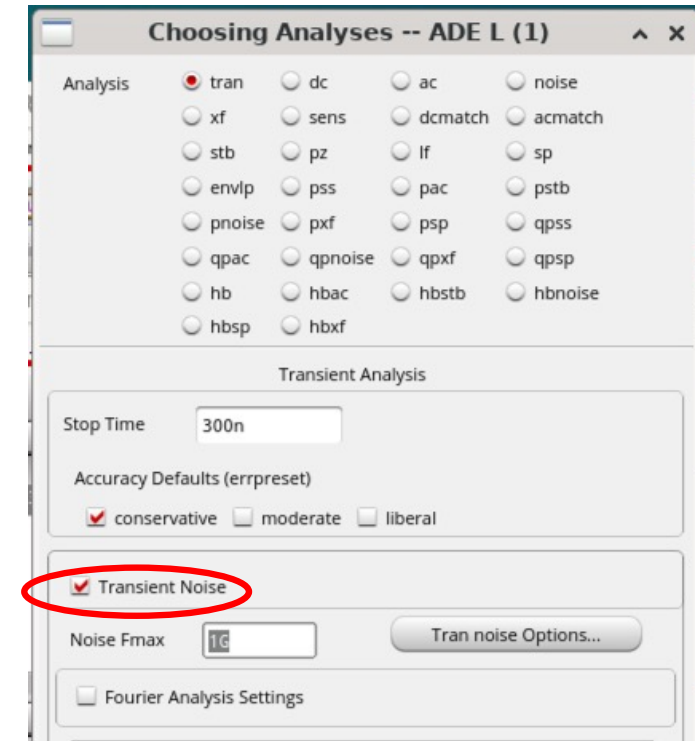
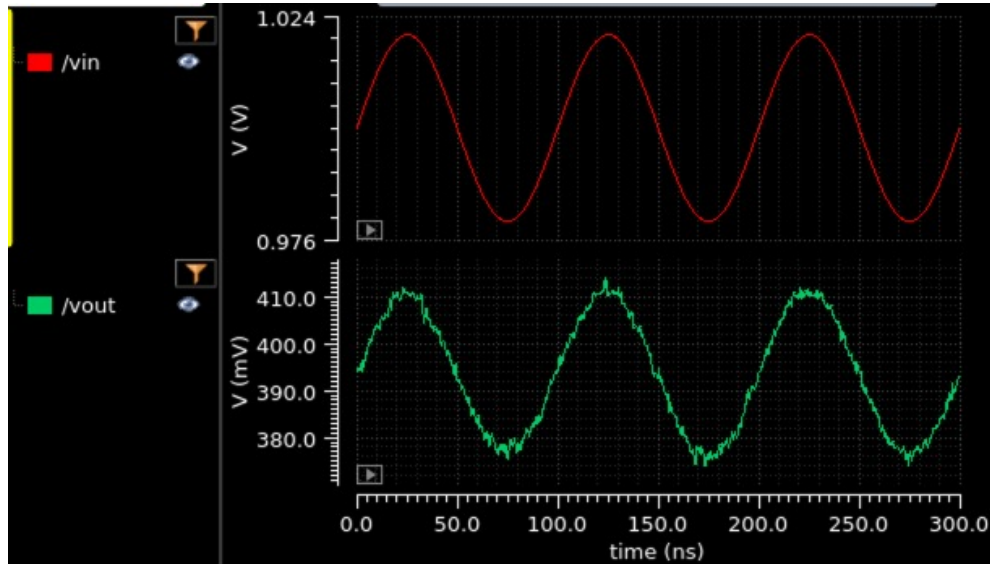
- Transient (noiseless):



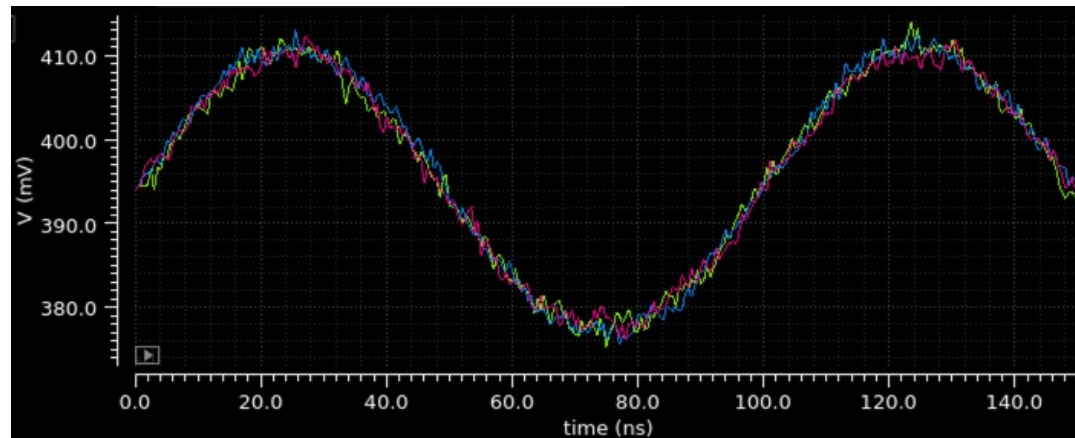


Transient Noise: Example

- Transient simulation with noise:
- Result:



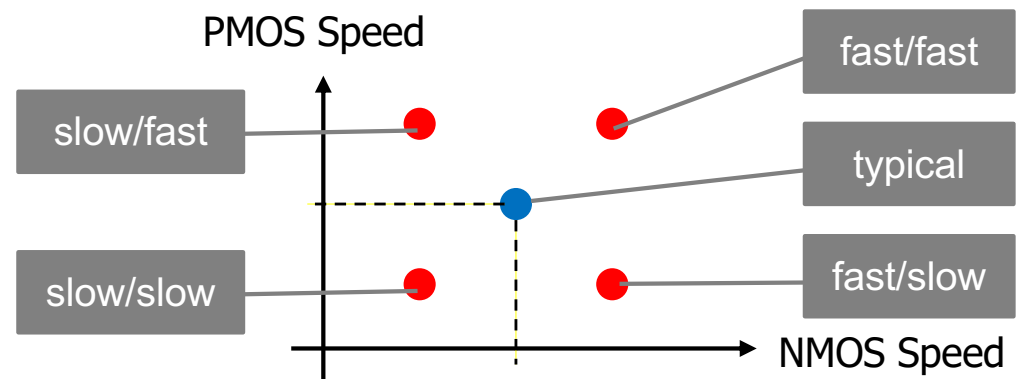
- Every Run is different ! :





Corner Simulation

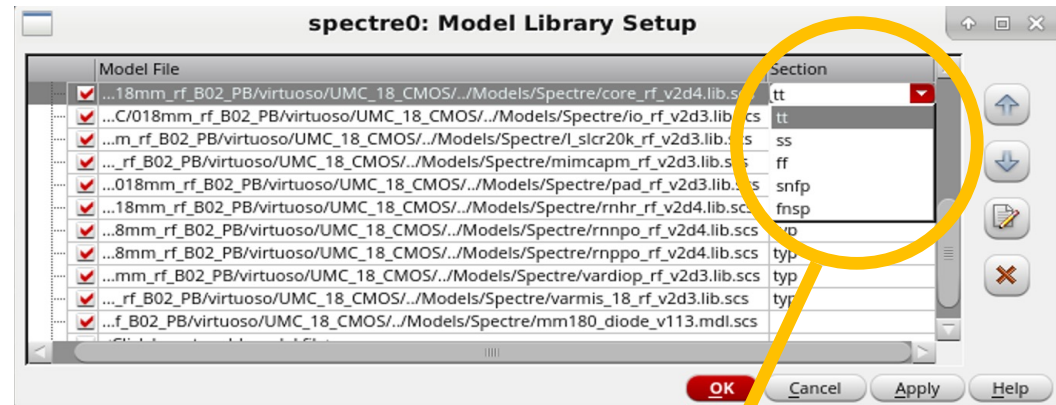
- The provided component models from the fab are for an *average* production, called a *'typical'* run.
- A production run can *systematically* deviate from 'typical' and devices behave differently. Such a run can still be 'in specs' because the vendor does not promise exact values for all parameters but ranges
 - e.g.: V_{Th} is typically 0.5 V, but can range from 0.4 ... 0.6 V
- Most vendors provide additional sets of model files for such 'corner' runs (i.e. runs which are 'just ok')
 - Most critical components are NMOS and PMOS
 - Models are often
 - tt – typical
 - ss – slow/slow
 - sf – slow/fast
 - ...





Corner Simulation

- You should simulate all 5 cases!
- ff gives ‘*highest speed*’, but *maximal power consumption*!
 - (fast: devices with lower threshold, higher K-factor, higher supply,..)
- ss gives *critical speed of design*
- sf and fs have large asymmetries between NMOS and PMOS and can be *dangerous*!
 - e.g.: Can a SRAM cell still be written if NMOS is weak and PMOS is strong?



- Technically
 - Sometimes, must change model files
 - Sometimes, ONE model file has several ‘*sections*’
- It can be impossible or tricky to ‘loop’ over all 5 models to see all results in one parametric plot....



Monte Carlo Simulation

- Different (identically sized) devices on a chip are not ***exactly*** the same in reality.
- Small variations between ‘identical’ devices may be a problem, e.g.
 - Input offset voltage of amplifiers
 - Nonlinearity in segmented DACs
- In a ‘Monte Carlo’ Simulation, the parameters of ***each device in the circuit*** are varied a bit (randomly)
- By running many simulations (with new variations), the sensitivity of the circuit can be quantified.
- This needs information about the ‘real’ variation of all parameters, for instance average value and standard dev.
 - Quality of these parameters sometimes not clear...

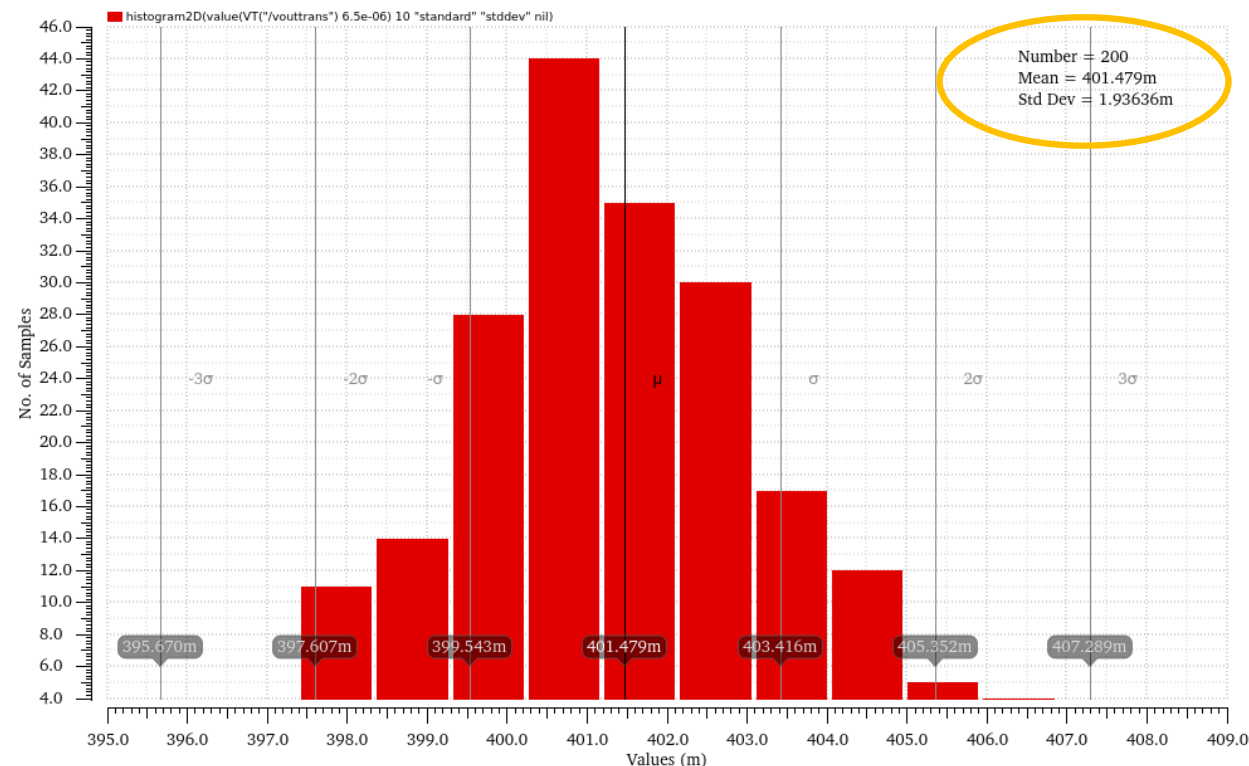


Monte Carlo Simulation

- Result of a Monte Carlo Run is a ‘probability’ distribution of a certain parameter, for instance
 - DC input level of an amplifier
 - Voltage output of a voltage reference
 - Threshold of a comparator

histogram2D(value(VT("/vouttrans") 6.5e-06) 10 "standard" "stddev" nil)

1





Parasitic (Extracted) Simulation

- The interconnections in schematics are idealised.
 - MOS caps and Source/Drain diode caps are taken into account for by models!
 - (Best immediately think at ‘other cells’ that will be connected to a signal later!..)
- In reality, traces have R and C
 - This will affect (slow down) the circuit and produce crosstalk
- ‘Parasitic’ Rs and Cs are only known when layout is done.
- They can be extracted with suited tools -> ‘extracted netlist’
- This netlist can be used in simulation instead of the ideal netlist to get ‘real’ behaviour.
- Advice: Try to anticipate long traces and put them as extra caps in the simulation schematic right away!



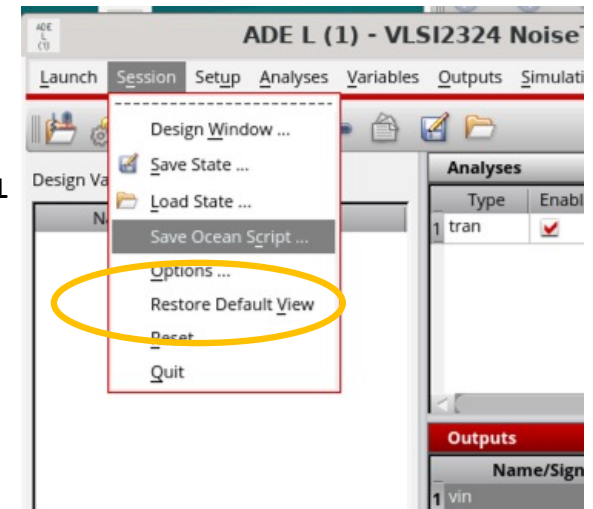
Scripted Simulations with Ocean - Cool & Easy!

- Repetitive simulations can be scripted using 'oceanscript'
- These are basically SKILL commands controlling spectre
- Start with an ADE simulation. Then export that simulation to an ocean script (.oce) from ADE->Session->Save Ocean...

```

simulator( 'spectre )
design("/tmp/ADE-Sim-
fischer/NoiseTran/spectre/schematic/netlist/netlist")
resultsDir("/tmp/ADE-Sim-fischer/NoiseTran/spectre/schematic" )
modelFile(
    ("somefile.scs" "tt") ; "tt" is the corner section typical/typical
    ; include here some more files
)
analysis('tran ?stop "300n" ?tranNoise "Transient Noise" ... more
parameters ...)
envOption('autoDisplay nil 'analysisOrder list("tran") )
saveOption( ?outputParamInfo t ) ; some options...
temp( 27 )

run()
selectResult( 'tran )
plot(getData("/vout") )
    
```



- Edit the skill file as you need.
- Run its from the CIW with (load "file.ocn")





Ocean Example

- Vary dc offset of source follower (p.11) input and measure output amplitude
- Minimalistic file, writing some data to 'R results.txt'

```

simulator( 'spectre )
design("/tmp/ADE-Sim-fischer/NoiseTran/spectre/schematic/netlist/netlist") ; This netlist must exist!
resultsDir("/tmp/ADE-Sim-fischer/NoiseTran/spectre/schematic") ; (create it by running ADE)
modelFile(
'("/opt/eda/UMC/018mm_rf_B02_PB/virtuoso/Models/Spectre/mm180_reg18_v124.lib.scs" "tt")
)
analysis('tran ?stop "300n" )
envOption('autoDisplay nil 'analysisOrder list("tran") )
temp( 27 )

; Self-made SKILL code starts here
; open a file for writing
p = outfile("Result.txt" "w")
(foreach os (list 0.1 0.2 0.5 0.5 0.6 0.8 1 1.2 1.6)
desVar("VOFFSET" os)
run()
selectResult( 'tran )
(fprintf p "Offset = %5.3f, Amplitude = %5.3f\n"
(float os) peakToPeak(v("/vout" ?result "tran-tran")))
)
)
(close p)

```

-> File Result.txt:

```

File Edit View Terminal Tabs Help
Offset = 0.100, Amplitude = 0.000
Offset = 0.200, Amplitude = 0.000
Offset = 0.500, Amplitude = 0.015
Offset = 0.500, Amplitude = 0.015
Offset = 0.600, Amplitude = 0.022
Offset = 0.800, Amplitude = 0.032
Offset = 1.000, Amplitude = 0.034
Offset = 1.200, Amplitude = 0.034
Offset = 1.600, Amplitude = 0.034

```

NMOS Source Follower does not work for low input amplitudes

Gain < 1
(Input amplitude is 40 mV_{pp})