Monte Carlo simulation

…… *for better yield and performance*

--A tutorial

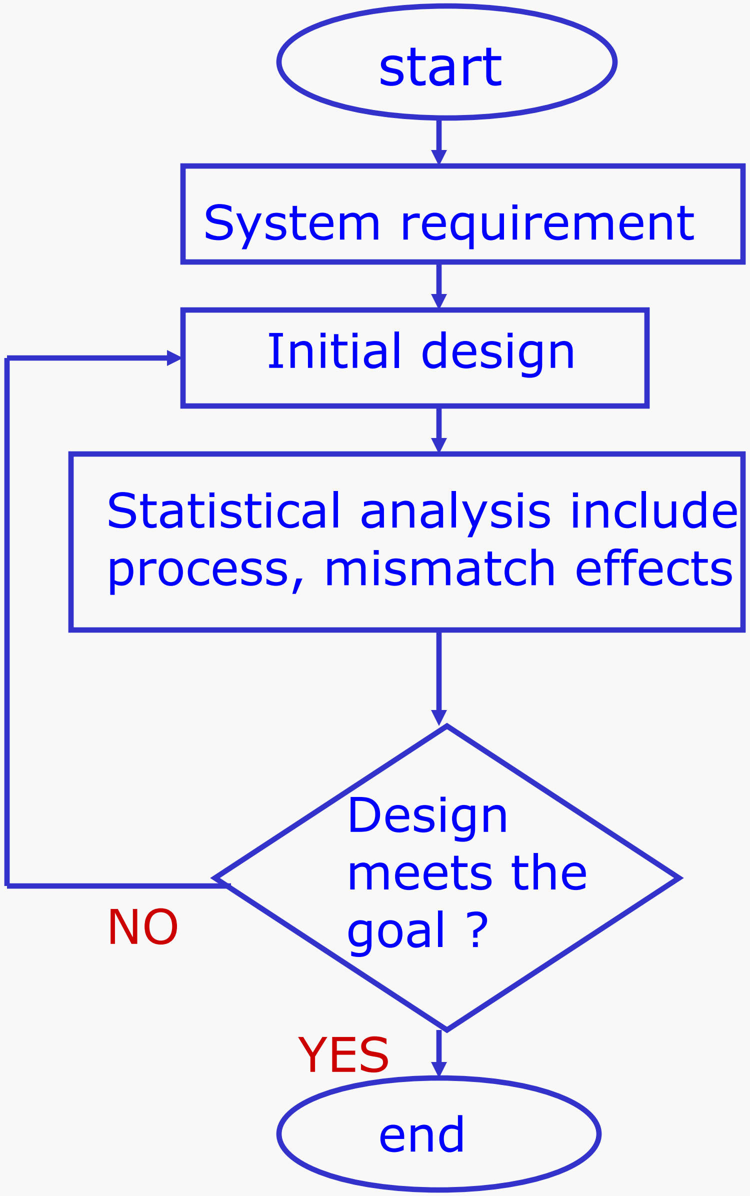
Monte Carlo simulation

……for better yield and performance

If fabrication process parameter and device mismatch effect on same die are not taken in to account then→

- Some design may degrade in performance
- Overall design yield could be unexpectedly low

Hence statistical analysis must find a high place in design cycle



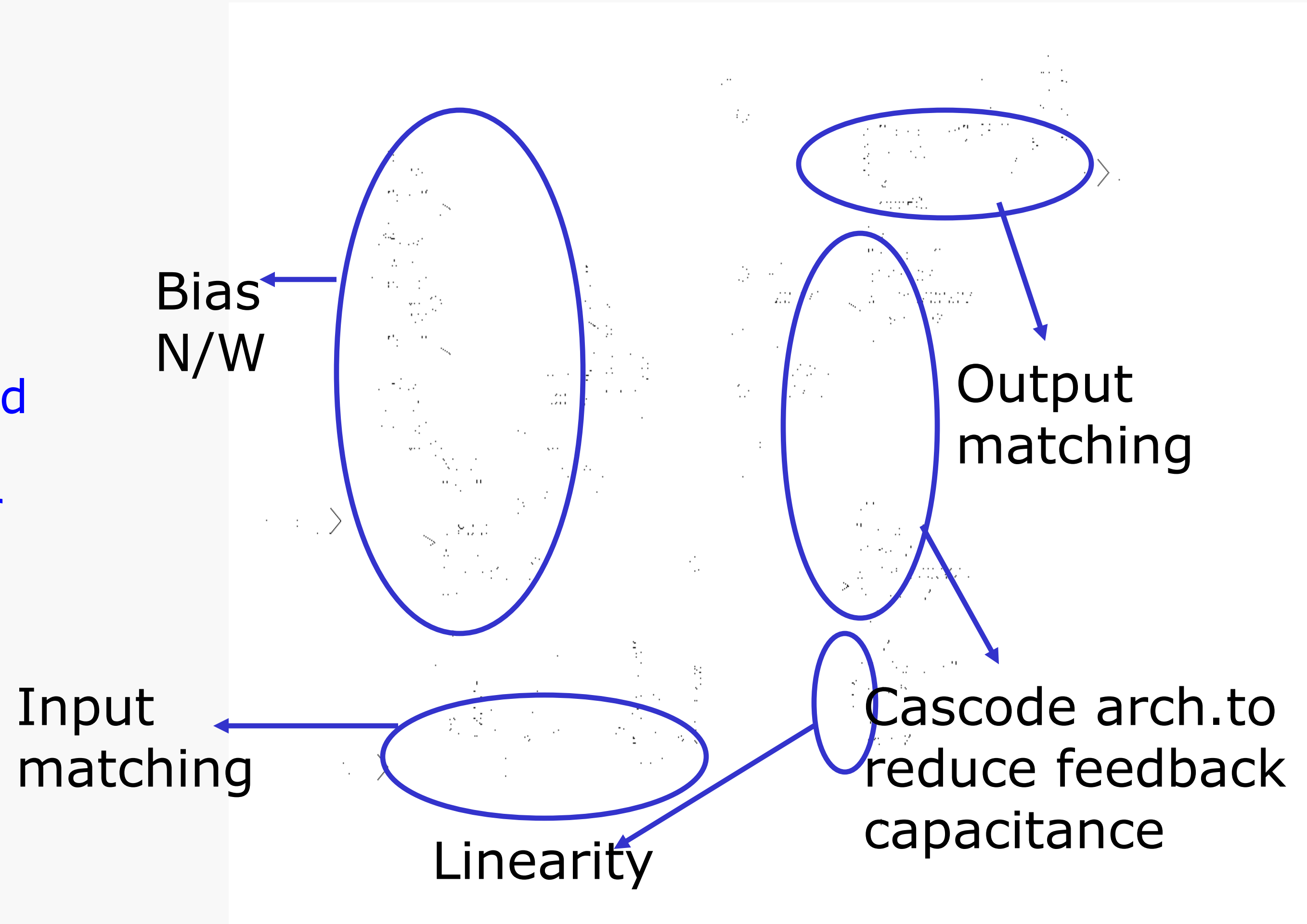
Monte Carlo simulation

- We will perform Monte Carlo analysis on an RF-front end LNA and compare the result if no statistical analysis is done.
- We will also see how to analyze yield and scalar data in Monte Carlo with the help of Low pass filter example.

Monte Carlo simulation(example)

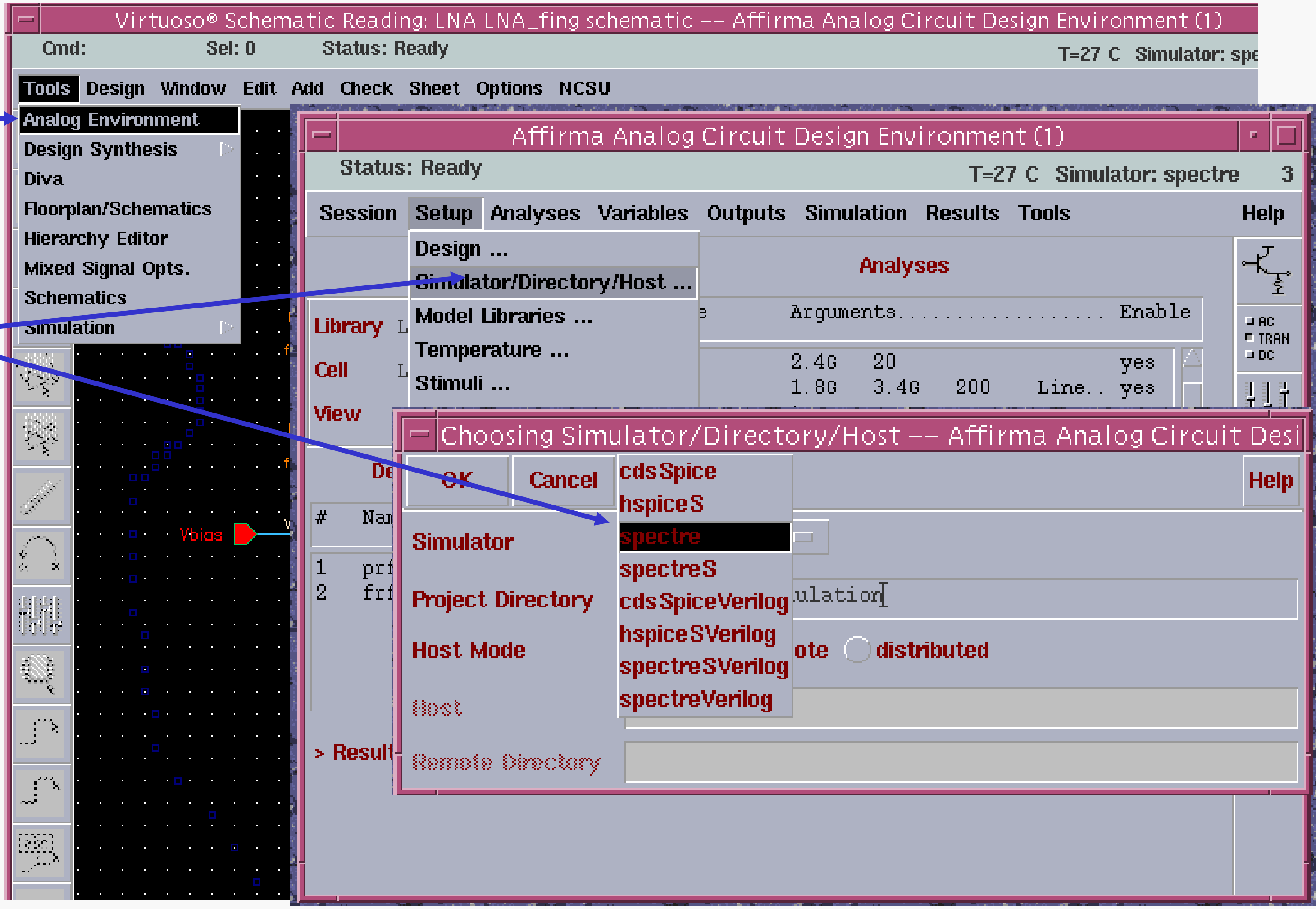
RF-front end (LNA)

- Knowing System requirement
- Initial design based on requirement like noise,gain,narrow or wide band.



Monte Carlo simulation

1. Choosing affirma analog artist
2. Choosing Spectre simulator

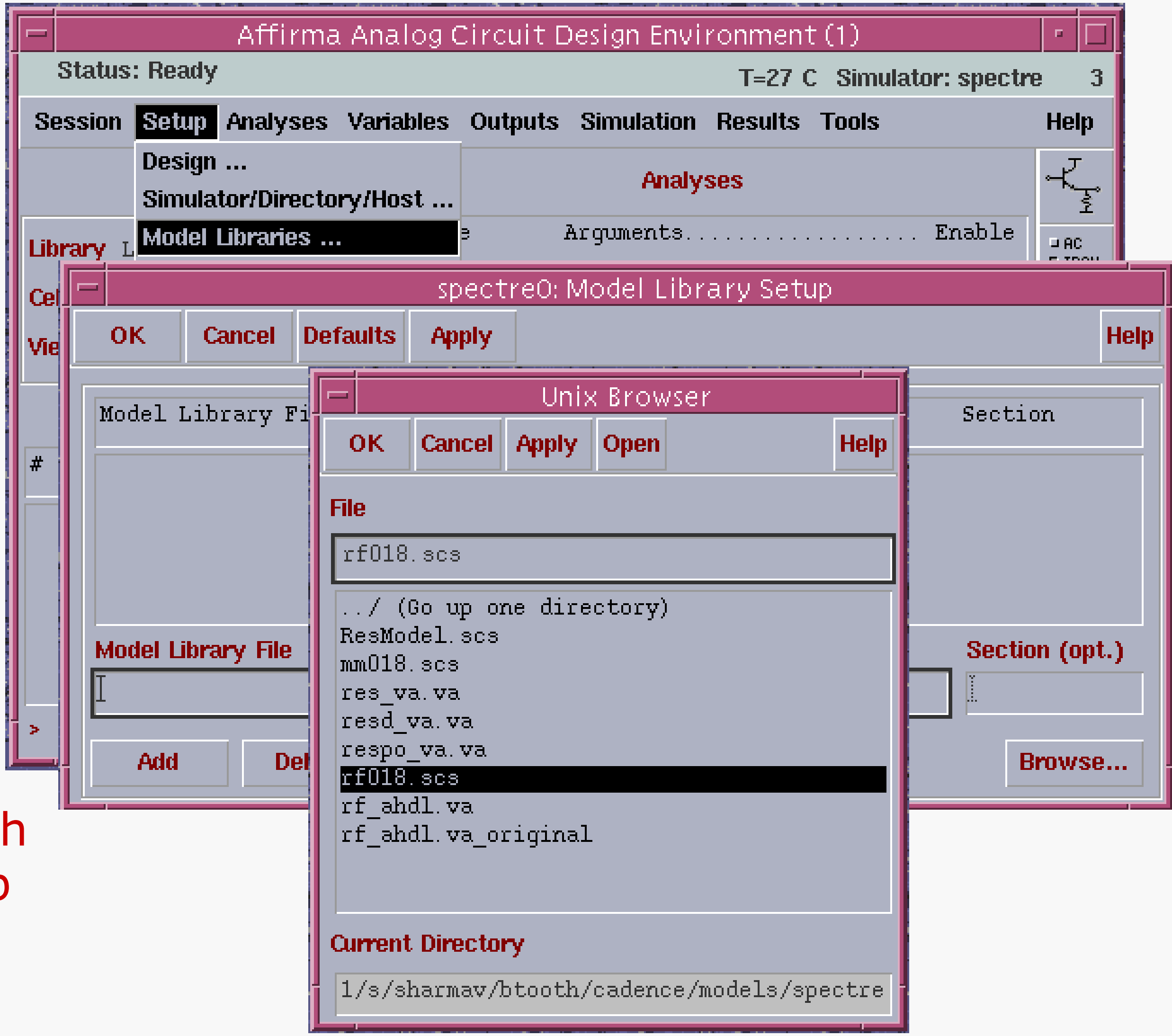


Cadence simulation setup (Normal)

Monte Carlo simulation

1.Choose
setup→ model libraries

2.Browse and choose
model file in the directory



Choosing model file,which contains all MOS,reg.,cap model parameters.

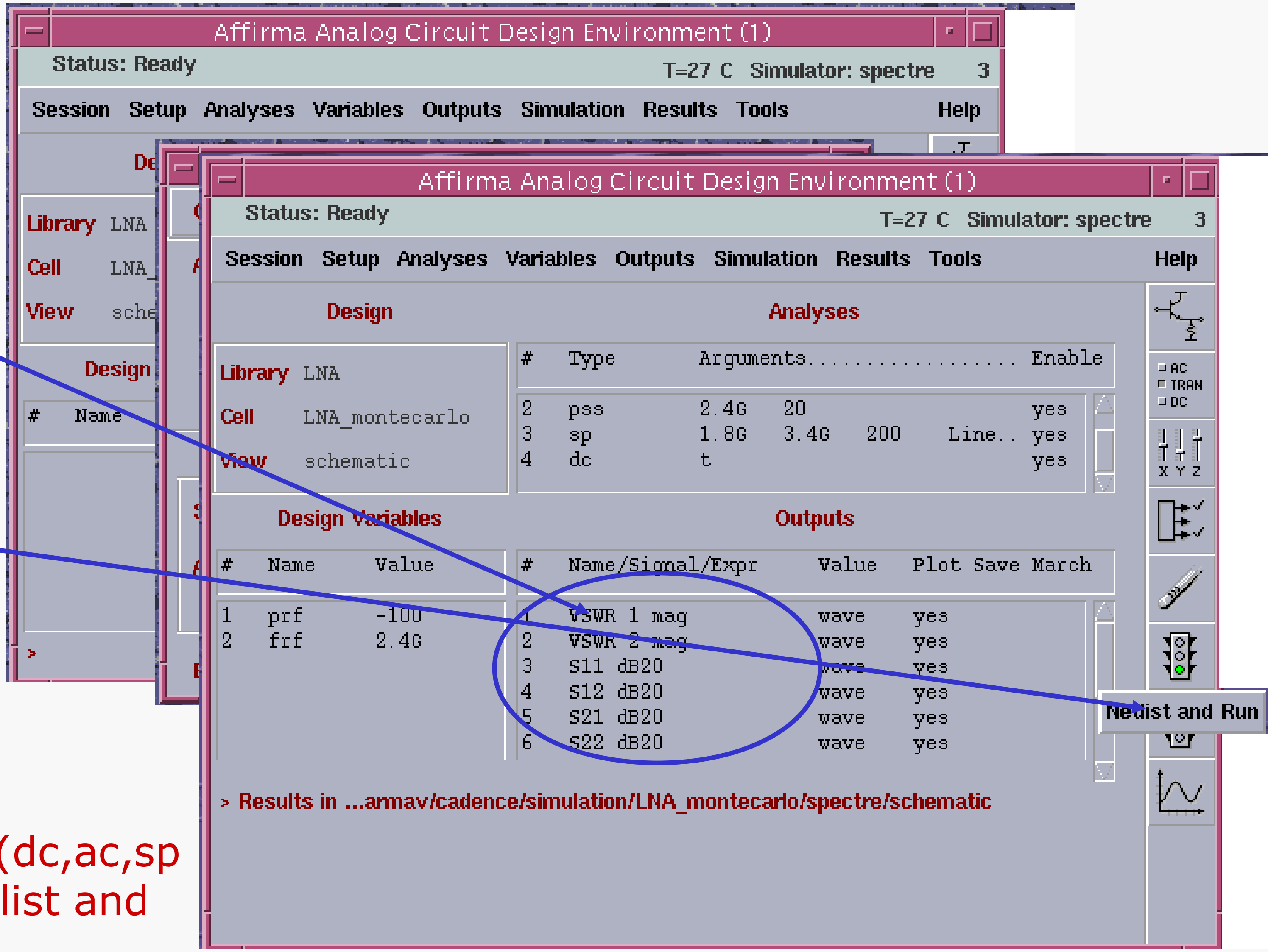
Cadence simulation setup (Normal)

Monte Carlo simulation

1.Choose analysis to run

2.Choose output to plot

3.Create netlist and run



Set up analysis(dc,ac,sp etc.),create netlist and run simulator

Cadence simulation setup (Normal)

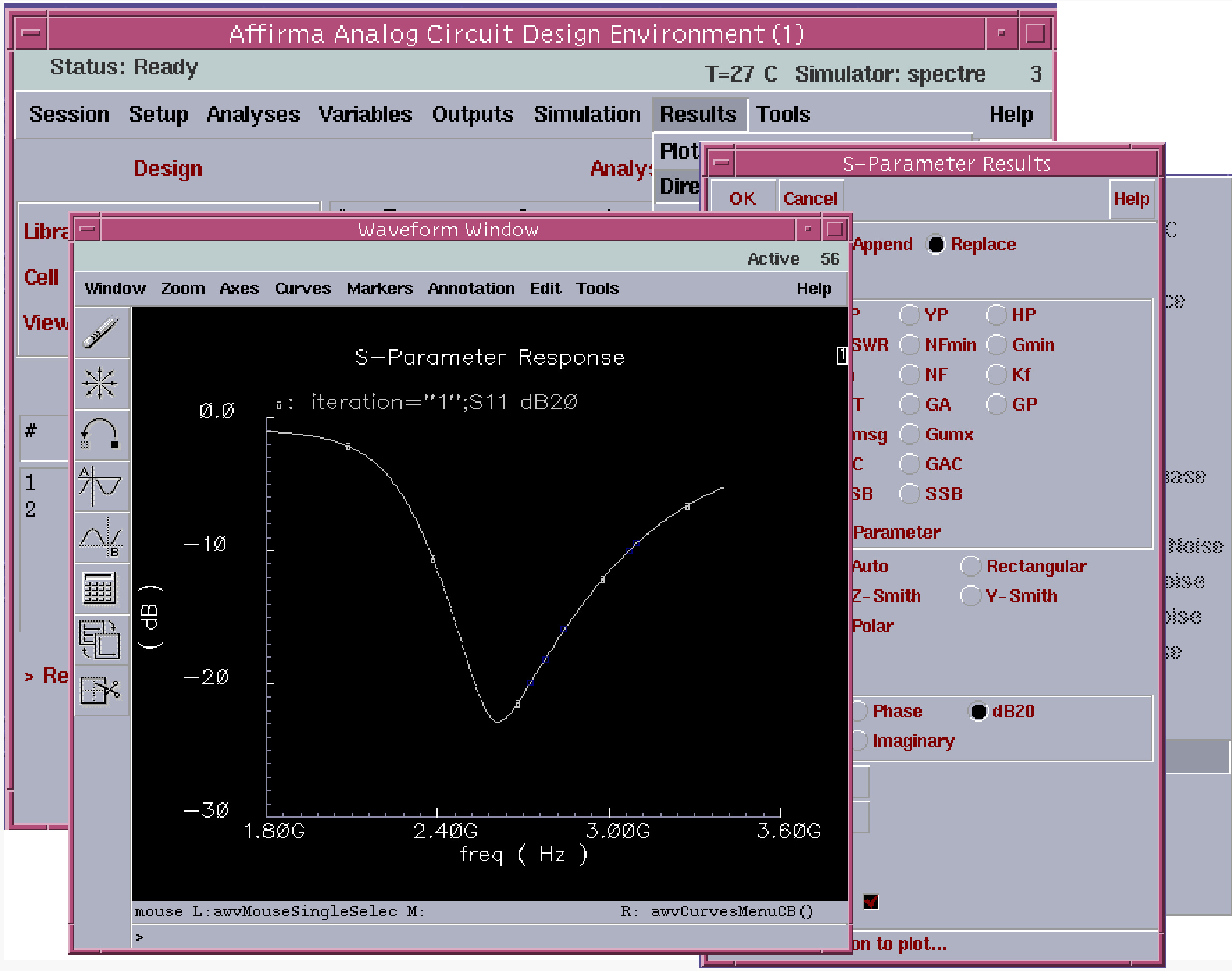
Monte Carlo simulation

Plotting results

1.Choose direct plot for analysis

2.Click to view the desired result

3.Analyze waveform



Cadence simulation setup (Normal)

Monte Carlo simulation

Monte Carlo modeling in Cadence spectre simulator

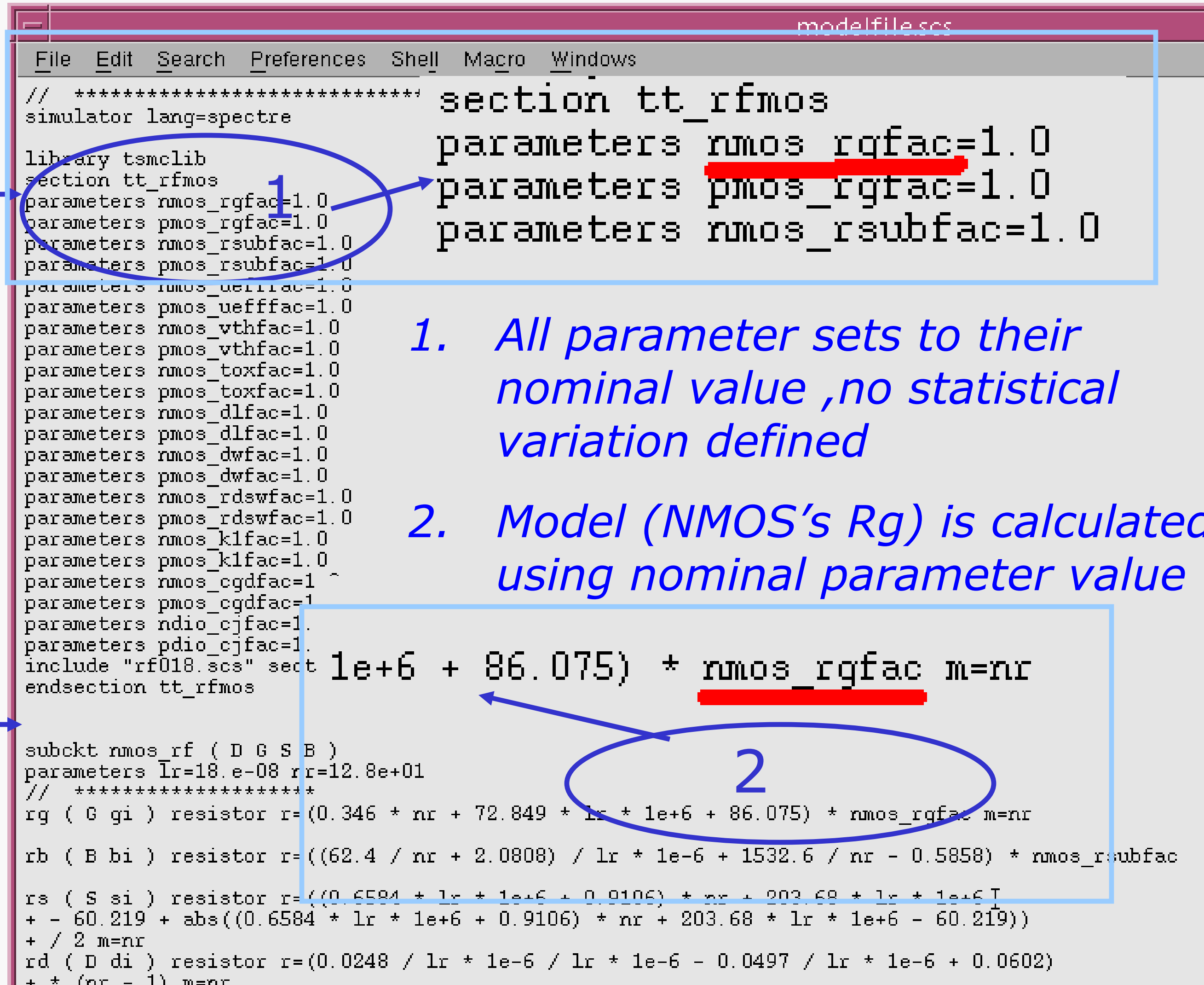
- **Process Section** - describes manufacturing parameter,their statistical variation and a model for device that calculates its(width,length,cap,res. Etc.)according to process parameter.

- **Design- Specific Section** – designer according to his need can specify Monte Carlo analysis.For example in a current mirror circuit,matched transistors are used and designer can give some correlation factor between these matched transistor.

Monte Carlo simulation

Typical Model File

Process Section



1. *All parameter sets to their nominal value ,no statistical variation defined*
2. *Model (NMOS's Rg) is calculated using nominal parameter value*

Cadence simulation setup (Monte Carlo)

Monte Carlo simulation

Defining process,mismatch parameter as statistically assigned value

Process Section

Assesses the device mismatch on different die, which could have gone through some different process parameters during fabrication.

Assesses the device mismatch on same die,which could have gone through some different process parameter.

modelfile.scs

File Edit Search Preferences Shell Macro Windows

```
parameters pmos_rdswfac=1.0
parameters nmos_k1fac=1.0
parameters pmos_k1fac=1.0
parameters nmos_cgdfac=1.0
parameters pmos_cgdfac=1.0
parameters ndio_cjfac=1.0
parameters pdio_cjfac=1.0
include "rf018.scs" section=rf_macro
endsection tt_rfmos


//***********Include one STATISTICAL Section in the model file ***************

section stats

simulator lang=spectre

statistics {
      process {
                      vary     nmos_rgfac              dist=gauss std= 15e-2

                      vary     nmos_rsubfac            dist=gauss std= 18e-2

                      vary     nmos_vthfac             dist=gauss std= 12e-2

                      vary     nmos_toxfac             dist=gauss std= 8e-2


                    }

      mismatch {
                      vary     nmos_rgfac              dist=gauss std= 12e-3

                      vary     nmos_rsubfac            dist=gauss std= 15e-3

                      vary     nmos_vthfac             dist=gauss std= 16e-3

                      vary     nmos_toxfac             dist=gauss std= 11e-4


      }
}

endsection stats
```

Variation defined as a distributed function

Cadence simulation setup (Monte Carlo)

Monte Carlo simulation

Design Specific Section

This includes the circuit connectivity(two resistors, and corresponding current sources that feed them)

Defining correlation between two devices(R1,R2) †

†Note :Alternatively this information can also be inserted through Artist Monte Carlo Tool.

```
                        vary     nmos_rdswfac           dist=gauss std=2e-3
                        vary     nmos_k1fac             dist=lnorm std=2e-3
                        vary     nmos_cgdfac            dist=gauss std=0.10

        }
}

//**********DESIGN SPECIFIC SECTION ****************
//Two resistors ,4K nominal,different geometries

R1(1 0) RPLR Rnom=4kOhm WB=5
R2(2 0) RPLR Rnom=4kOhm WB=10

//Current source biasing

J1(0 1) isource dc=1mA //force 1 mA through R1
J2(0 2) isource dc=1mA //force 1 mA through R2


//Monte Carlo analysis specification

m1 montecarlo saveprocessparams=yes processscalarfile="../process_simple.dat"
+       numruns=3 variations=mismatch seed=10

{
        dcop dc

        export v1 =oceanEval("v(\"1\")")
        export v2 =oceanEval("v(\"2\")")
        export v1_2 =oceanEval "v(\"1\")- "v(\"2\")
}

// Match pairs,specify correlation Co-efficients

statistics {

correlate dev= [R1 R2]  cc=0.75 //correlate the resistors

endsection stats


//****************************************************************************
```

Cadence simulation setup (Monte Carlo)

Monte Carlo simulation

Model file used for LNA example

Note→This is not based on foundry data but modeled for illustrative purposes.

modelfile.scs

File Edit Search Preferences Shell Macro Windows

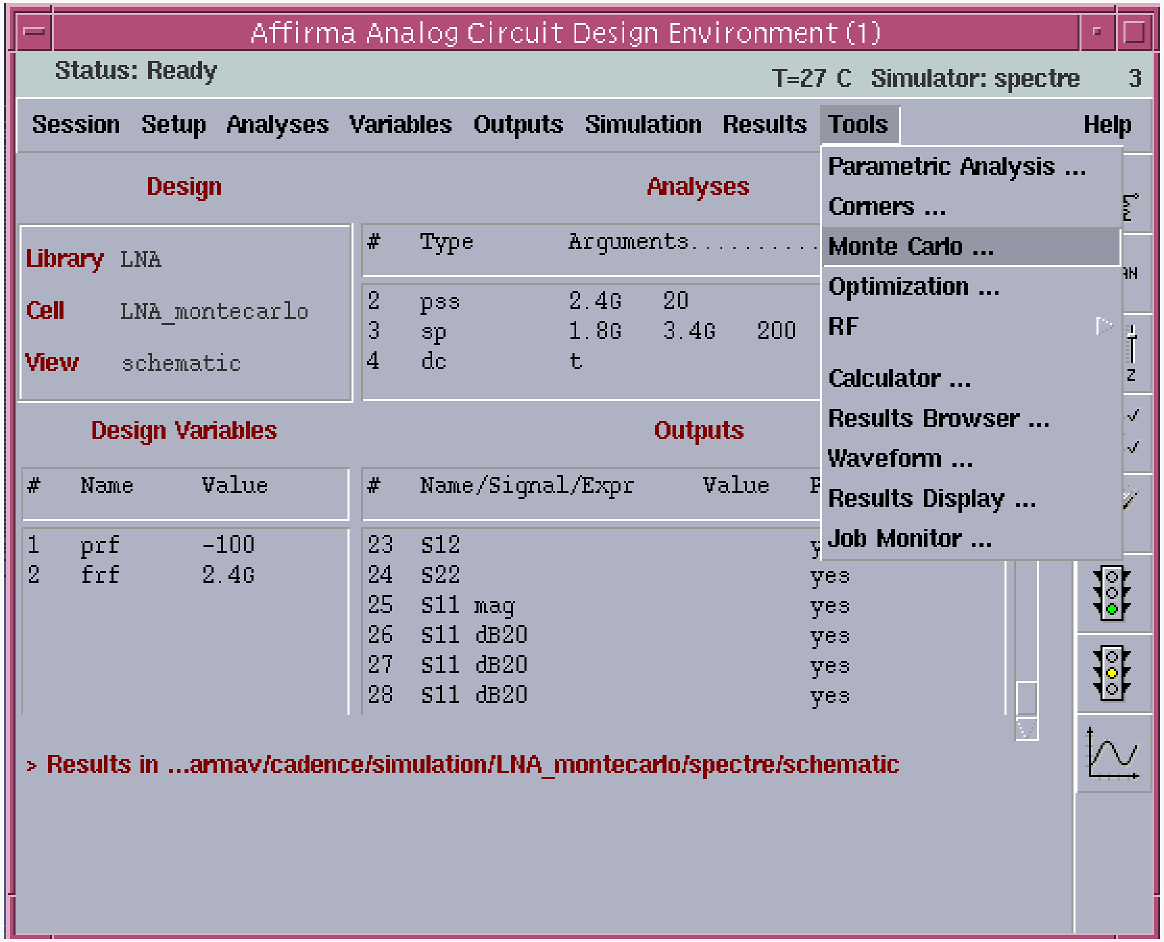
```
simulator lang=spectre
statistics {
        process {
                        vary     nmos_rgfac          dist=gauss std=0.10
                        vary     pmos_rgfac          dist=gauss std=0.10
                        vary     nmos_rsubfac        dist=gauss std=0.12
                        vary     pmos_rsubfac        dist=gauss std=0.170
                        vary     nmos_uefffac        dist=lnorm std=0.9
                        vary     pmos_uefffac        dist=lnorm std=1.1
                        vary     nmos_vthfac         dist=gauss std=12e-2
                        vary     pmos_vthfac         dist=gauss std=12e-2
                        vary     nmos_toxfac         dist=gauss std=0.17
                        vary     pmos_toxfac         dist=gauss std=0.9
                        vary     nmos_dlfac          dist=gauss std=0.11
                        vary     pmos_dlfac          dist=gauss std=0.8
                        vary     nmos_dwfac          dist=gauss std=0.15
                        vary     pmos_dwfac          dist=gauss std=0.14
                        vary     nmos_rdswfac        dist=gauss std=0.11
                        vary     pmos_rdswfac        dist=gauss std=12e-2
                        vary     nmos_k1fac          dist=lnorm std=20e-2
                        vary     pmos_k1fac          dist=lnorm std=19e-2
                        vary     nmos_cgdfac         dist=gauss std=0.12
                        vary     pmos_cgdfac         dist=gauss std=0.13
                        vary     ndio_cjfac          dist=gauss std=0.15
                        vary     pdio_cjfac          dist=gauss std=0.16
                       }
        mismatch {
                        vary     nmos_rgfac          dist=gauss std=0.01
                        vary     pmos_rgfac          dist=gauss std=0.01
                        vary     nmos_rsubfac        dist=gauss std=2e-3
                        vary     pmos_rsubfac        dist=gauss std=2e-3
                        vary     nmos_uefffac        dist=lnorm std=0.10
                        vary     pmos_uefffac        dist=lnorm std=0.10
                        vary     nmos_vthfac         dist=gauss std=2e-3
                        vary     pmos_vthfac         dist=gauss std=2e-3
                        vary     nmos_toxfac         dist=gauss std=0.10
                        vary     pmos_toxfac         dist=gauss std=0.10
                        vary     nmos_dlfac          dist=gauss std=0.10
                        vary     pmos_dlfac          dist=gauss std=2e-3
                        vary     nmos_dwfac          dist=gauss std=2e-3
                        vary     pmos_dwfac          dist=gauss std=0.10
                        vary     nmos_rdswfac        dist=gauss std=0.10
                        vary     pmos_rdswfac        dist=gauss std=2e-3
                        vary     nmos_k1fac          dist=lnorm std=2e-3
                        vary     pmos_k1fac          dist=lnorm std=2e-3
                        vary     nmos_cgdfac         dist=gauss std=0.10
                        vary     pmos_cgdfac         dist=gauss std=0.10
                        vary     ndio_cjfac          dist=gauss std=0.10
                        vary     pdio_cjfac          dist=gauss std=0.10
        }
}
```

Cadence simulation setup (Monte Carlo)

Monte Carlo simulation

After Initial design that meets the system requirement,statistical analysis must have to be carried out.

1. Make sure the addition of process and mismatch parameter section in model file.
2. Make certain to include the particular section (for exa.Stats in spectre) in simulation model library
3. Go to tool→Monte Carlo in affirma analog artist



Cadence simulation setup (Monte Carlo)

Monte Carlo simulation

Choose no of iteration(default=100)

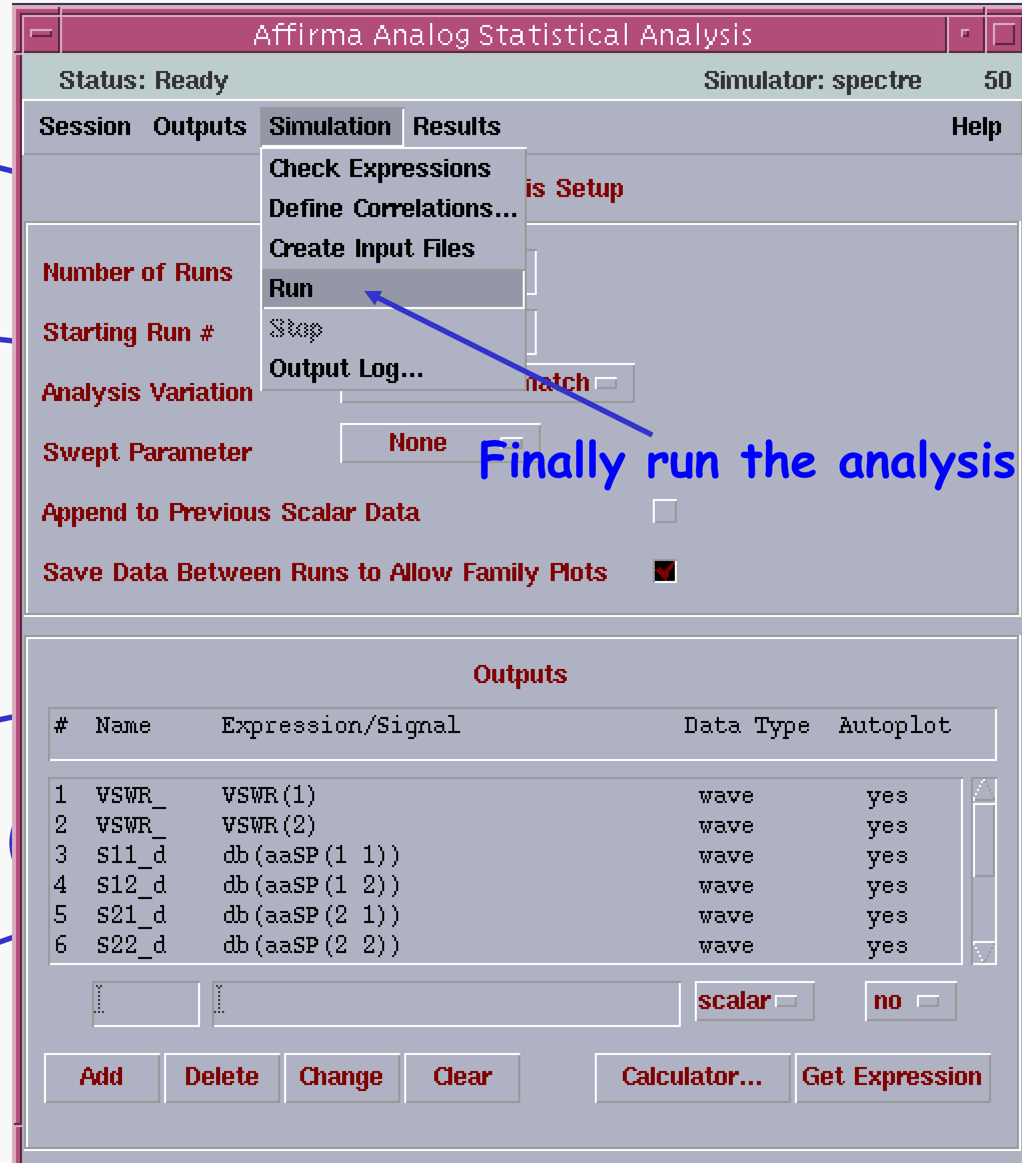
1.Choose which variation to include

Process→device mismatch effect on two diff.die

Mismatch→device mismatch effect on same die

2.Click if you want to see the family of curve i.e. curve from each iteration

3.Define the expressions / signals on which Monte Carlo analysis will be performed.



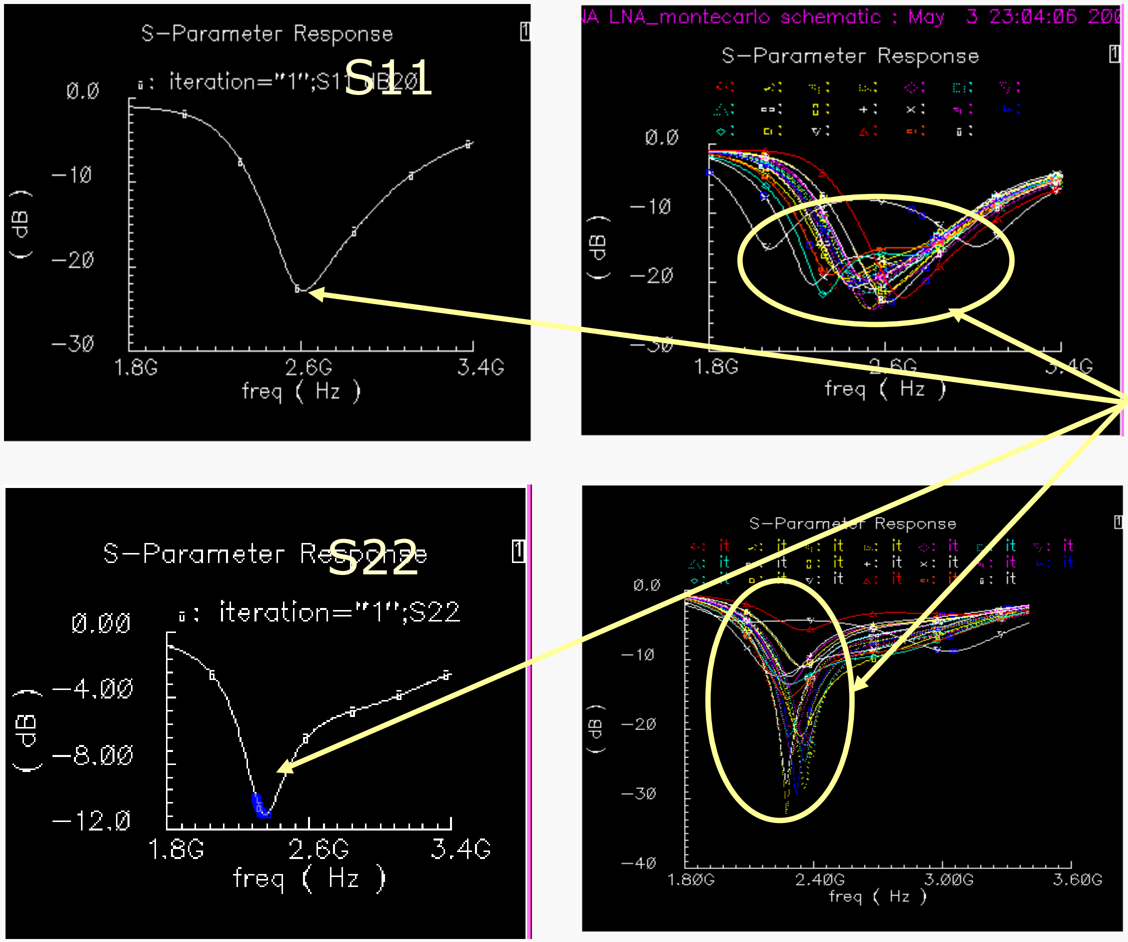
Finally run the analysis

Note: calculator can also be used to get these expression

Cadence simulation setup (Monte Carlo)

Monte Carlo simulation (Analyzing waveform)

Matching



Process parameter and mismatch effect

DEGRADES

Input & Output matching N/W

DEGRADES

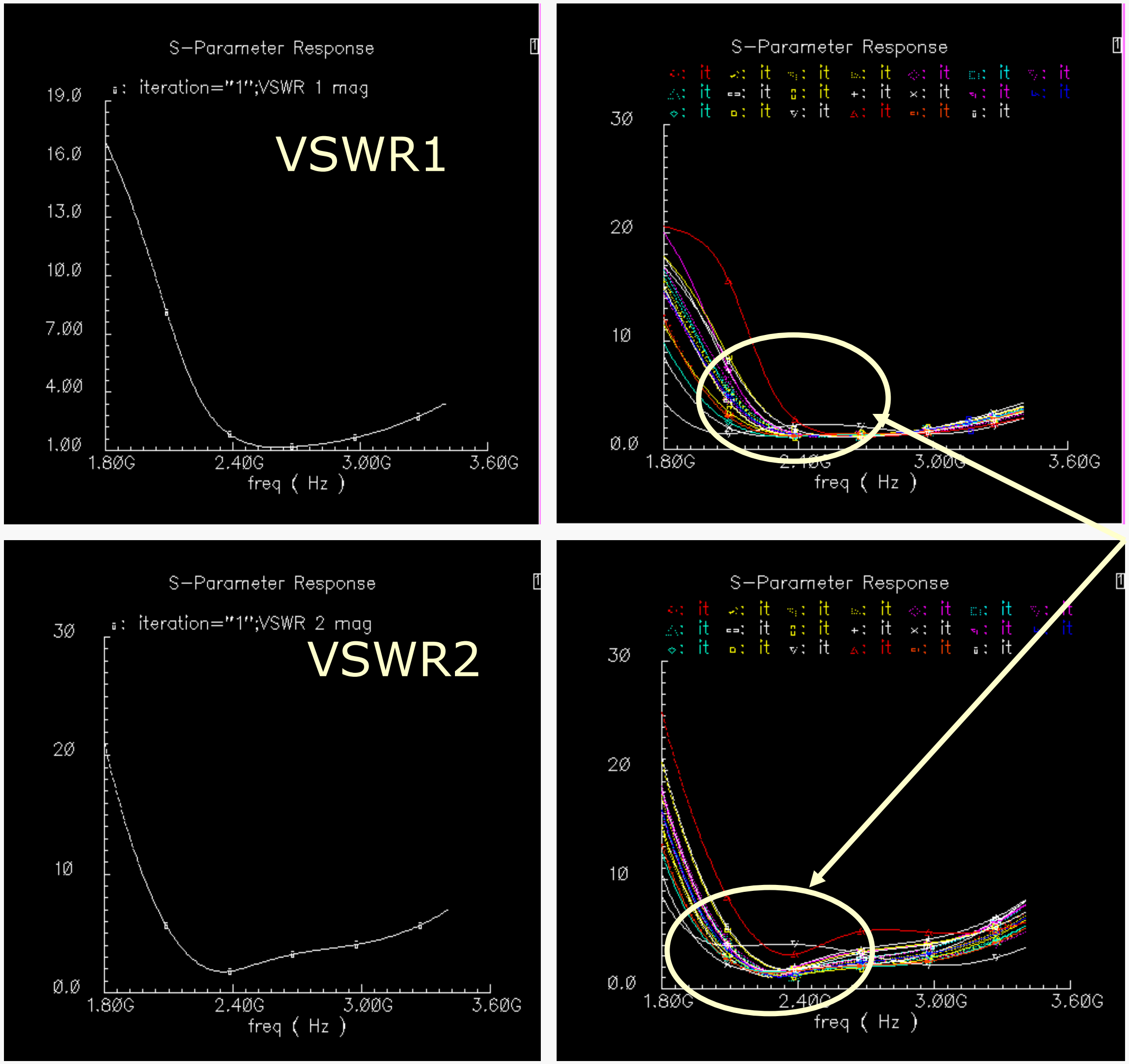
Overall design performance (noise,gain etc.)

Normal simulation
(without statistical variation)

Monte Carlo Simulation
(with statistical variation)

Monte Carlo simulation (Analyzing waveform)

Matching(VSWR):It tells how well input and output N/W are matched.



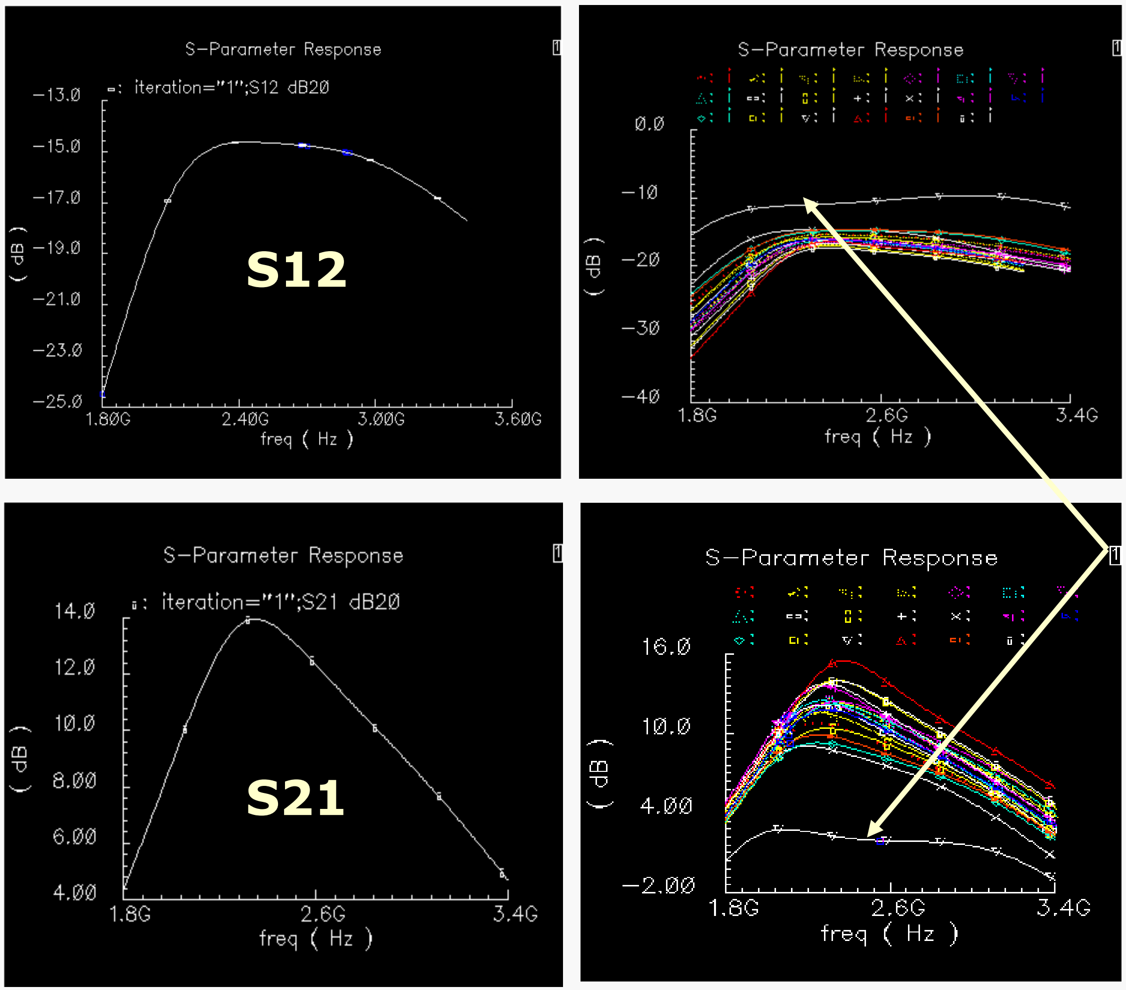
Variations in VSWR

Normal simulation

Monte Carlo simulation

Monte Carlo simulation (Analyzing waveform)

Matching(forward and reverse transmission gain)



Normal simulation

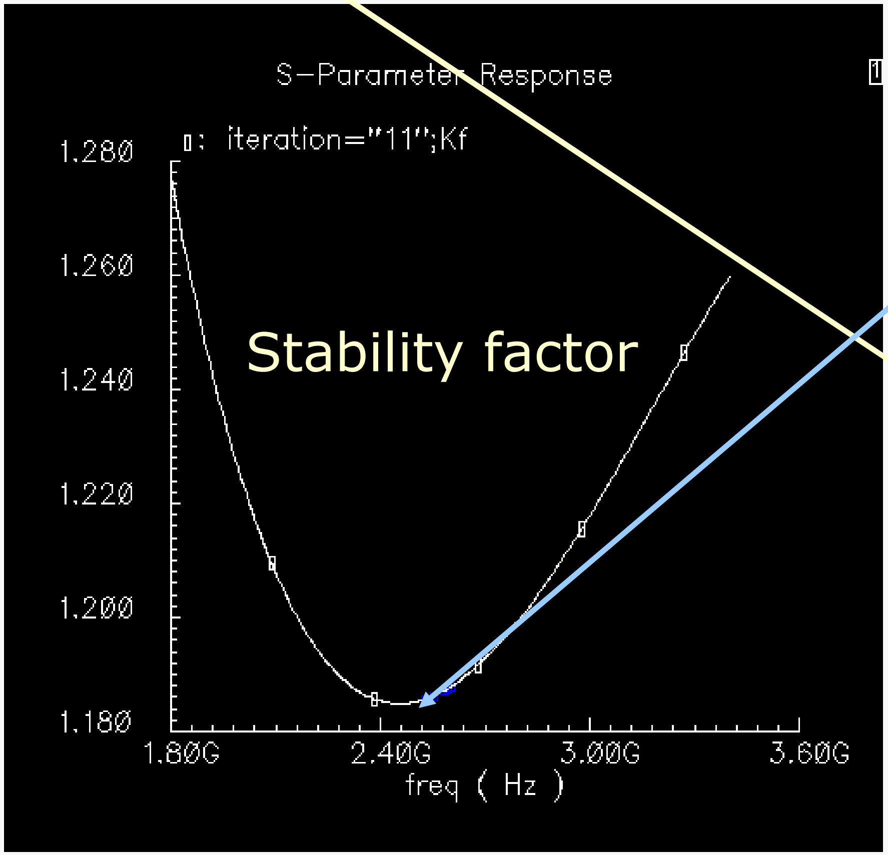
Monte Carlo simulation

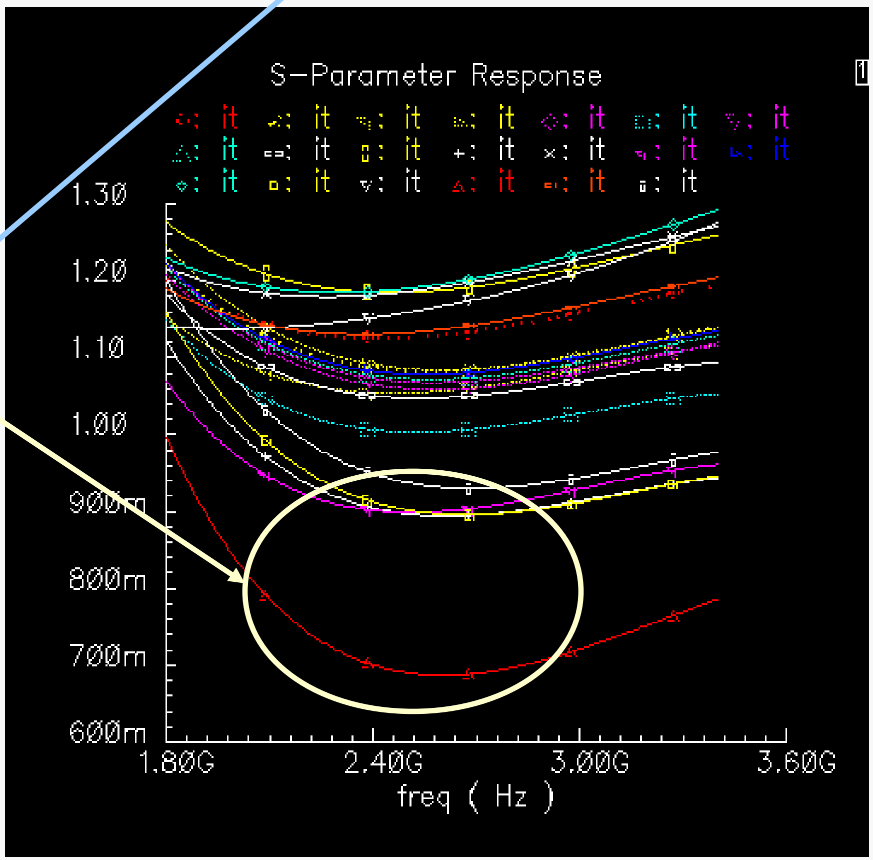
It has deteriorated the performance significantly, as a minimum S12 and maximum S21 value is desirable.

<u>Monte Carlo simulation</u> (Analyzing waveform)

<u>Stability</u>:A Kf value >1,is desired for an stable amplifier

Kf value has become <1,and consequently creating a potential unstability,hence a large margin is required at initial design phase.



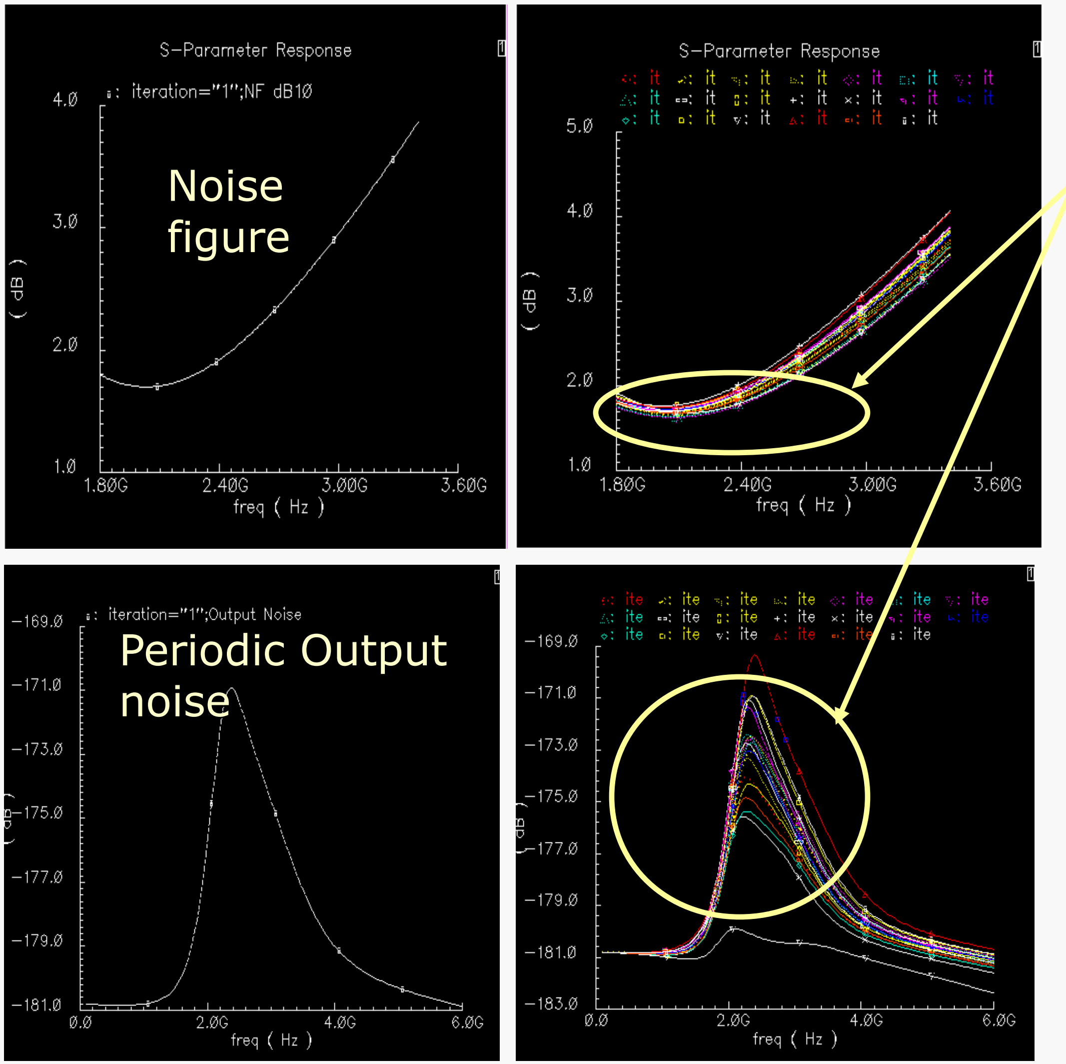


Normal simulation

Monte Carlo simulation

Monte Carlo simulation (Analyzing waveform)

Noise Performance



Normal simulation

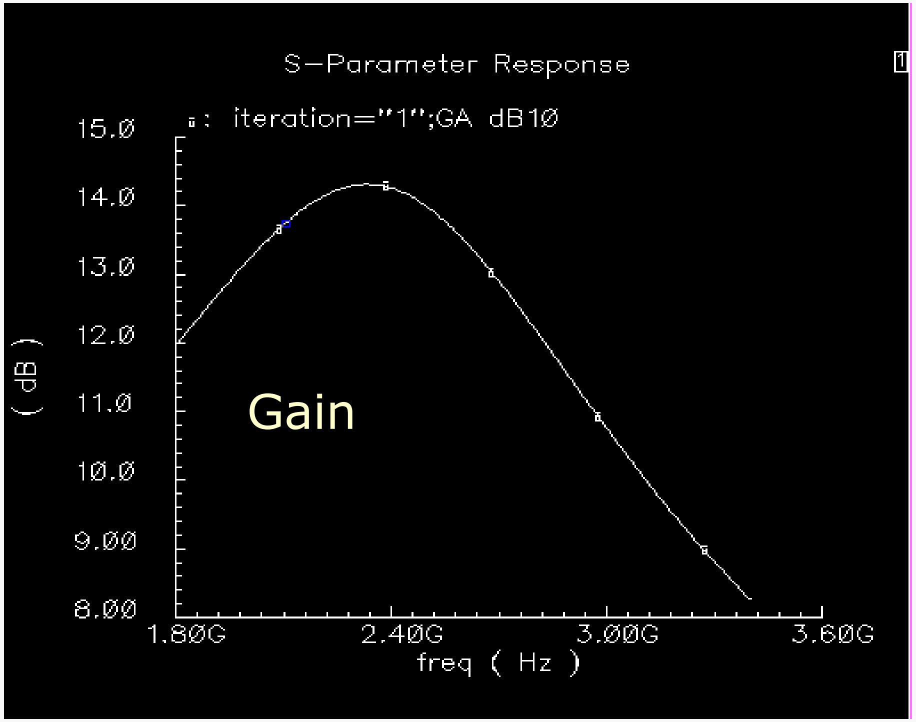
Monte Carlo simulation

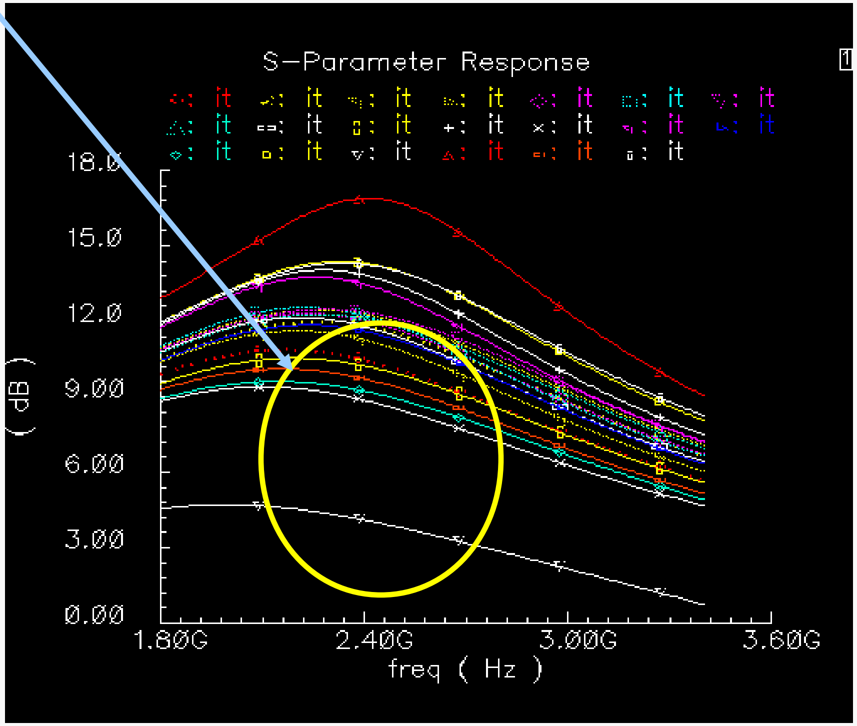
As visible, design has a robust noise performance at desired band(2.4-2.5 GHz) **BUT….→**

Monte Carlo simulation (Analyzing waveform)

But...→ LNA as an RF-front end has to provide enough gain with maximum noise suppression to maintain an allowable SNR at demodulator's input.

It fails to meet the gain requirement





Monte Carlo simulation

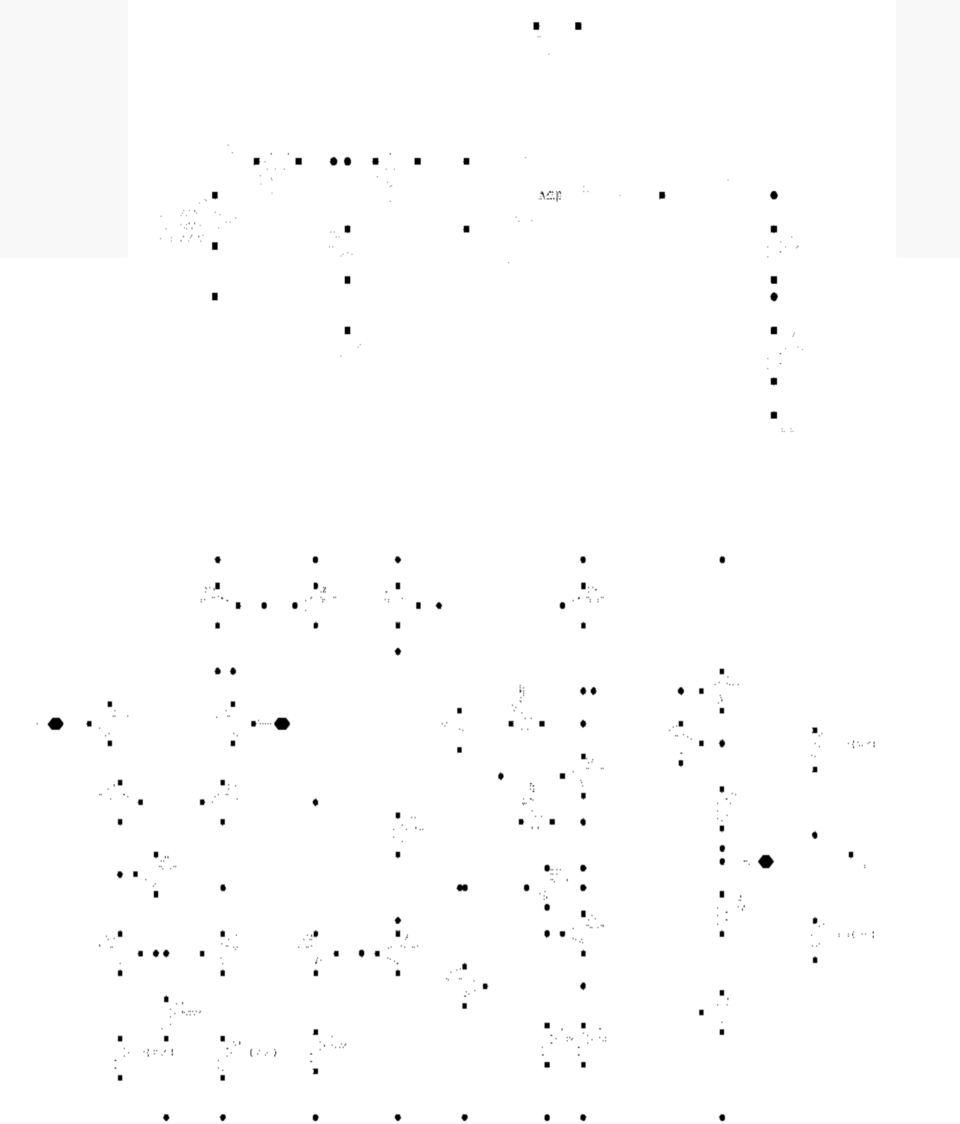
We will quickly go over another example of low pass filter and see how to analyze scalar data and yield through Monte Carlo simulation

Monte Carlo simulation

Low-Pass Filter

Initial Design:

Circuit designing according to system requirement

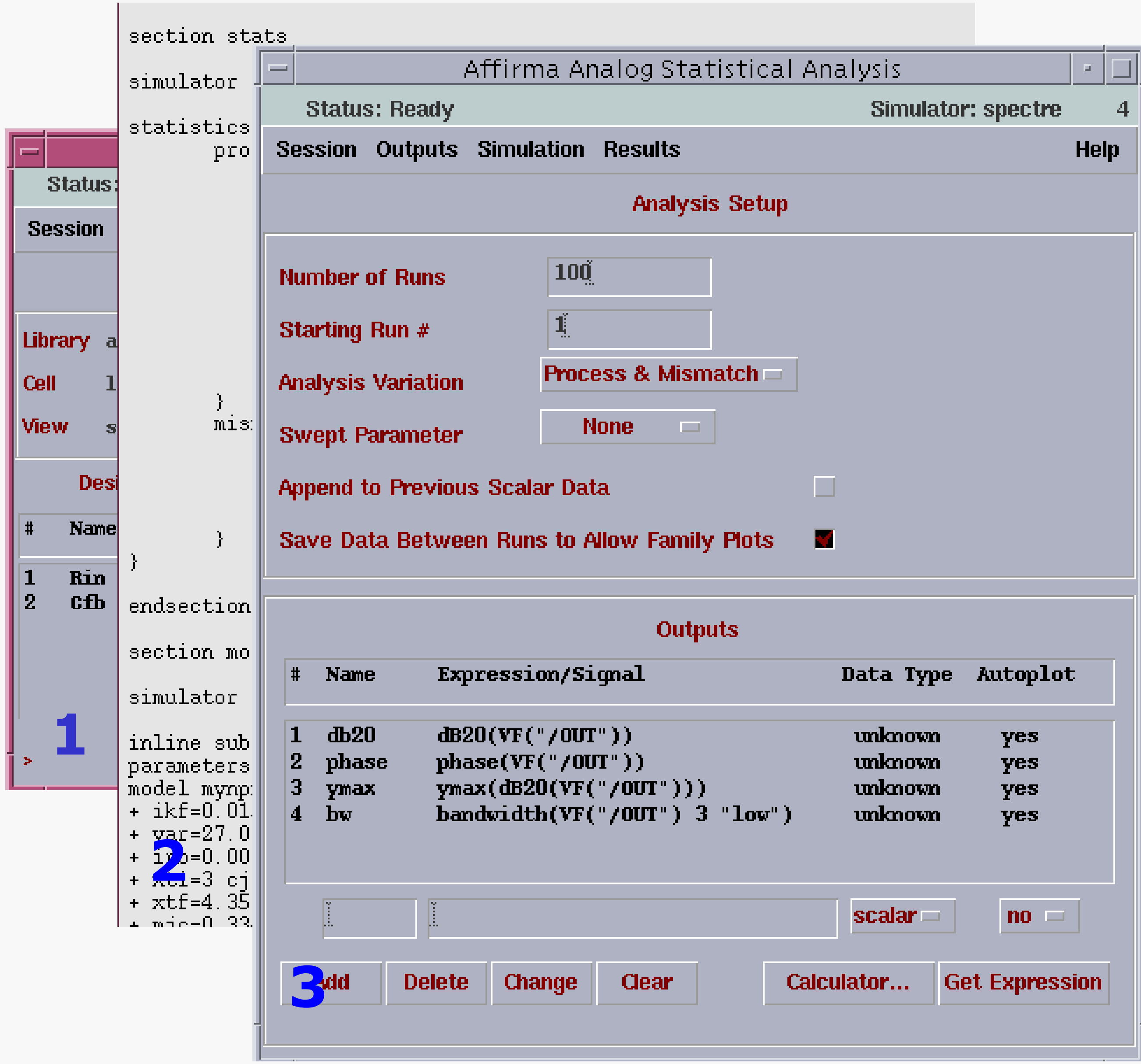


Monte Carlo simulation

1. Running normal analysis

2. Specifying statistical variation in model file

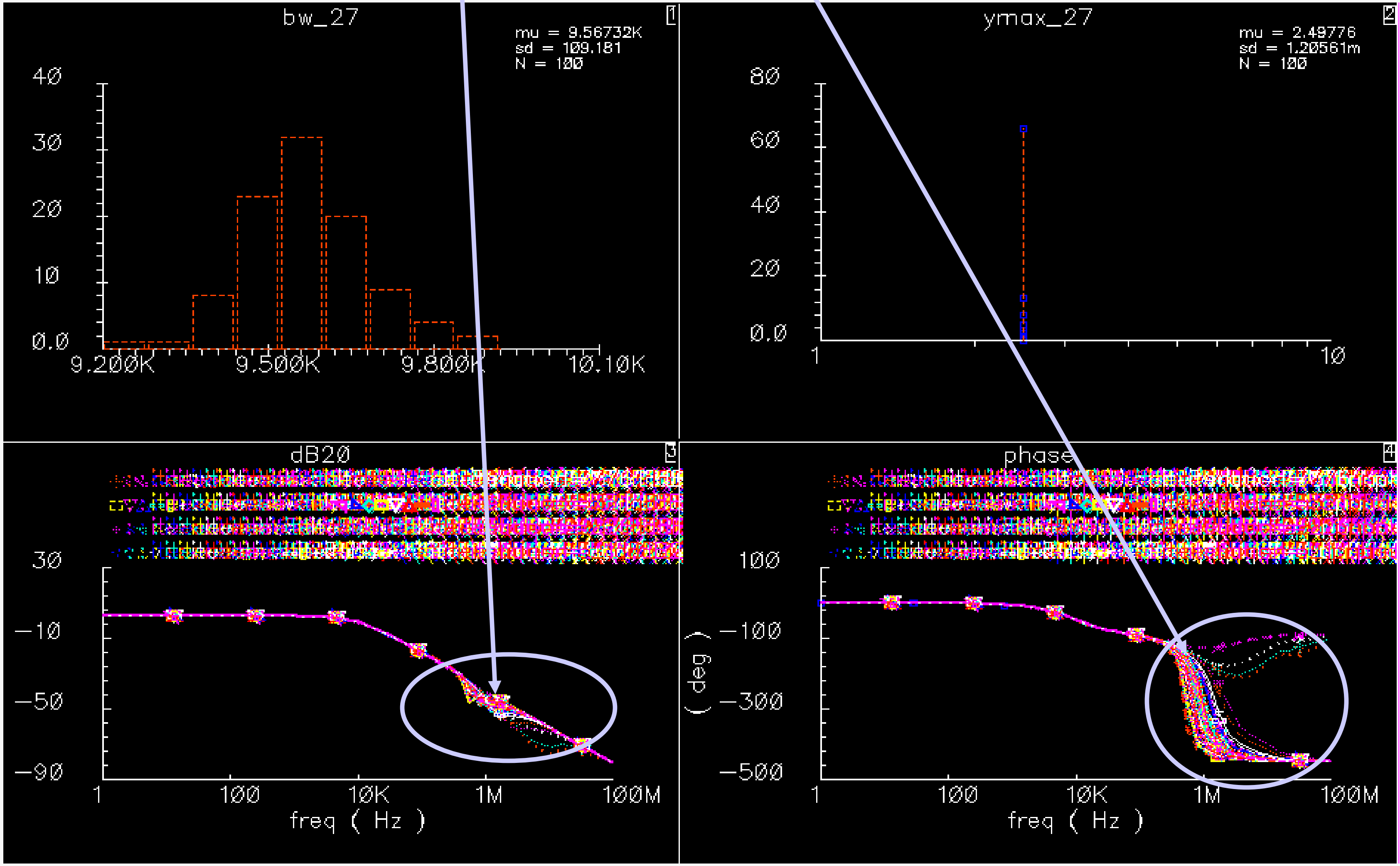
3. Running Monte Carlo analysis



Cadence simulation setup (Monte Carlo)

Monte Carlo simulation (Analyzing results)

Simulation shows *db20* and *phase* values are greatly affected by statistical variations introduced in transistor. Hence the need for ***redesigning*** the circuit

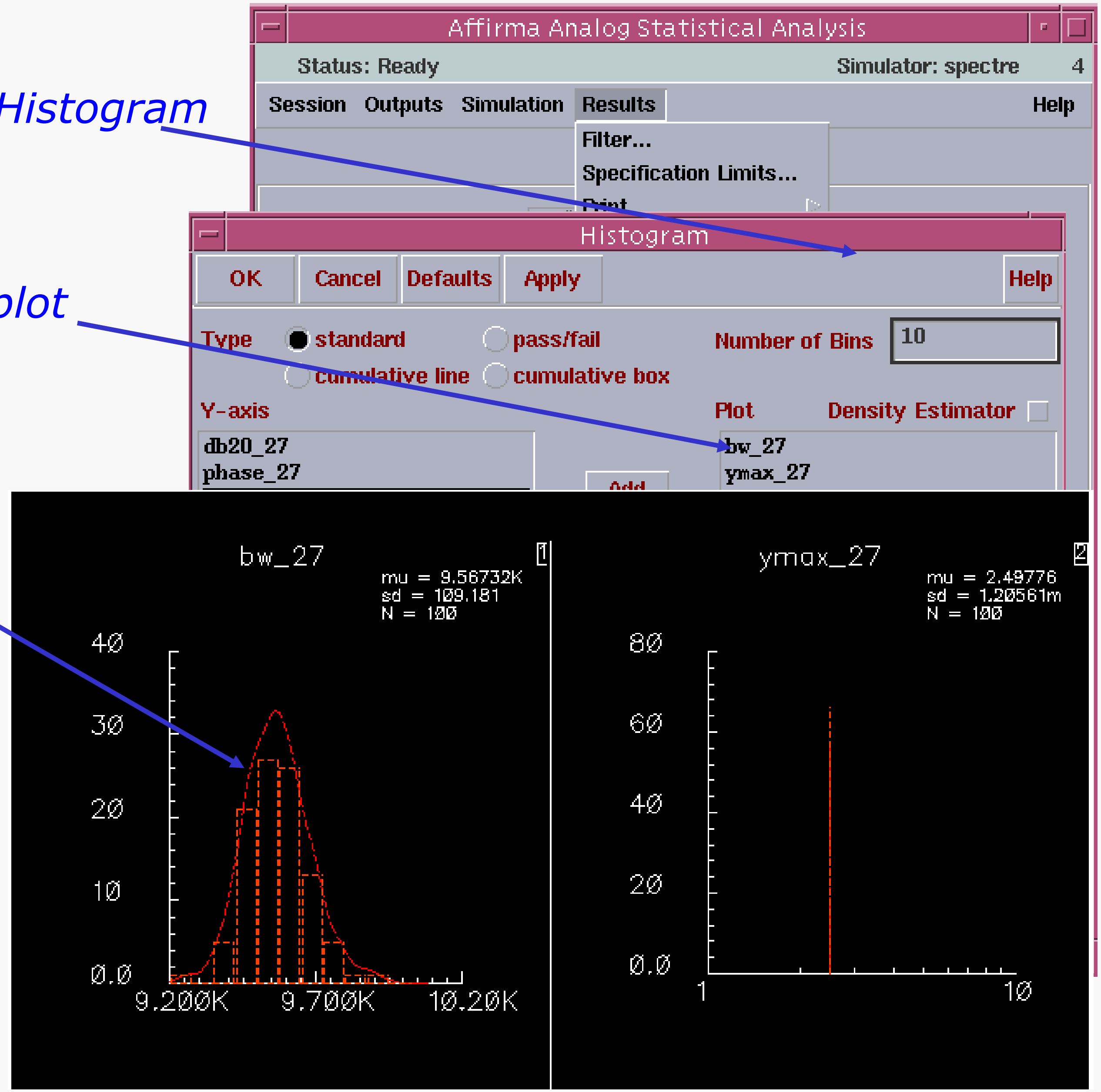


Monte Carlo simulation (Analyzing Scalar data)

1. Choose *results→plot→ Histogram*

2. Choose *parameters to plot*

3. Analyze the histogram appeared in waveform window



Monte Carlo simulation

(Analyzing Yield)

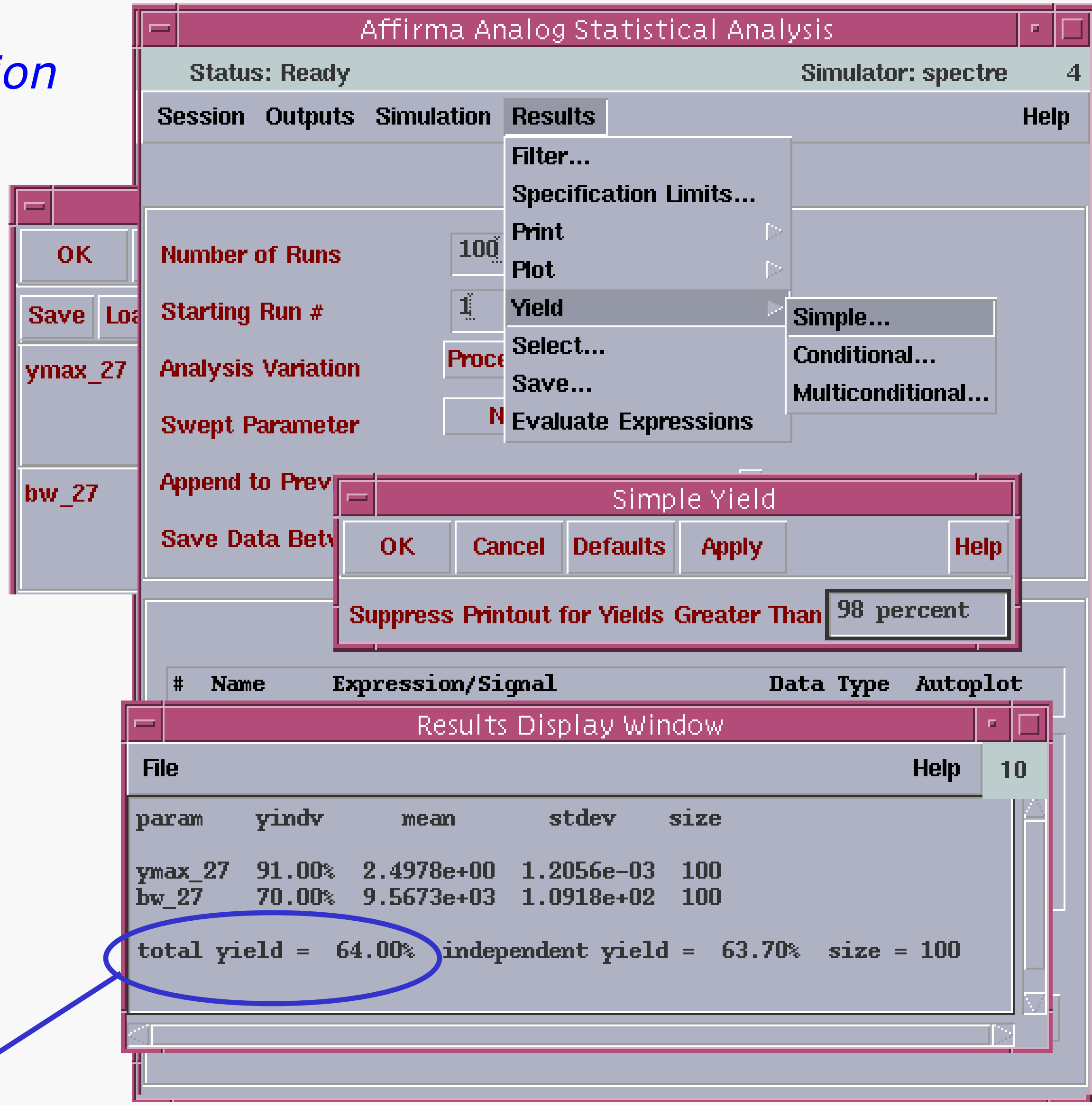
1. Choose *results*→ *specification limits*

2. Set *bounds* and *limits*

3. Choose Results→ *yield* → *simple* in analysis window

4. Set suppression value for yield

5. Analyze yield



Only 64% iterations passes the specified limits for bandwidth and ymax

Monte Carlo simulation (PLL Components)

Overview

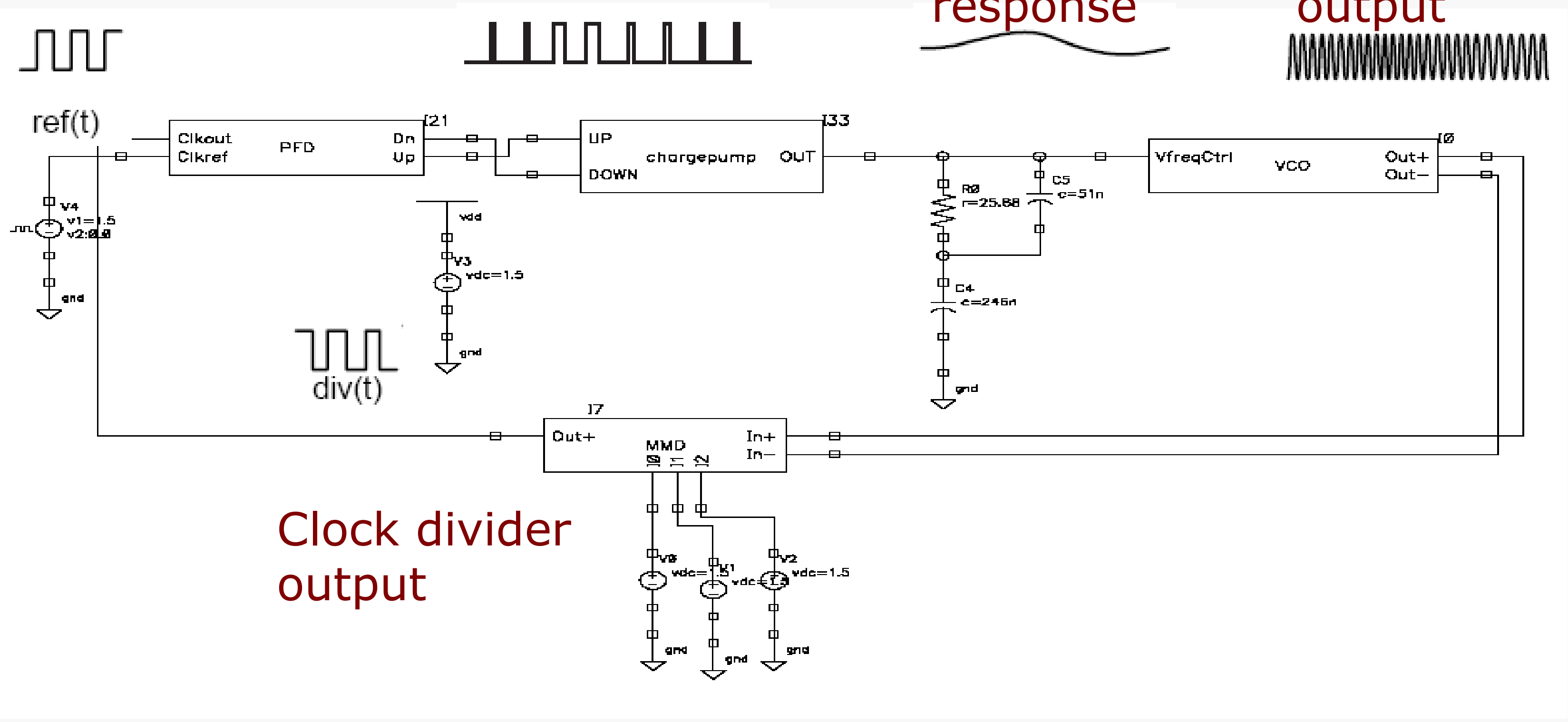
- Phase/frequency detector determines the difference between the phase or frequency of two signals
- The loop filter removes the high-frequencies from the voltage-controlled oscillator (VCO) controlling voltage
- The VCO produces and output frequency controlled by a voltage

Reference clock

Detector output

Loop filter response

Oscillator output



Clock divider output

Monte Carlo simulation (PLL Components)

Noise Sources

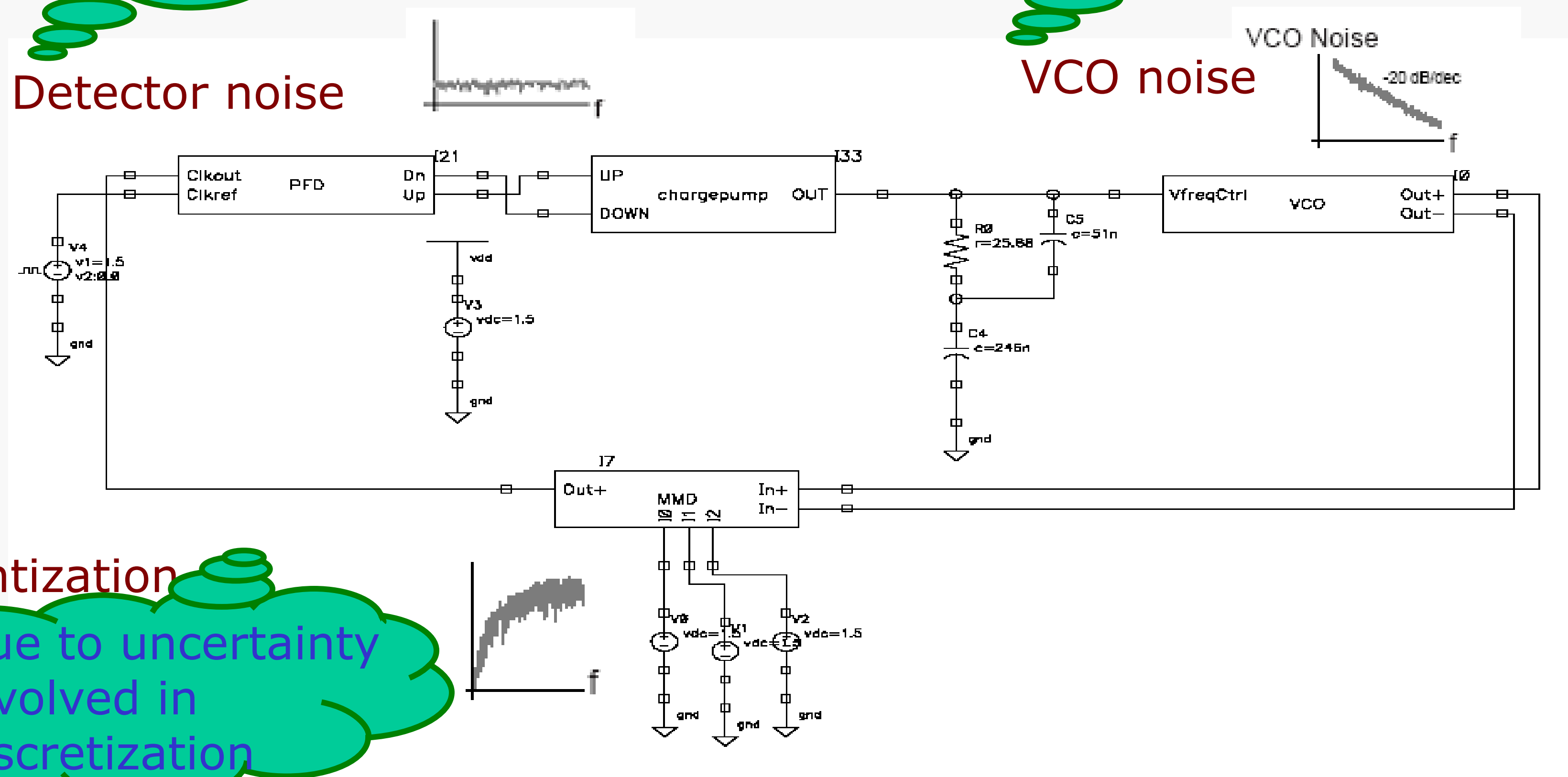
In PLL design it is highly desirable to be able to see the impact of all noise sources,which in turn affects the overall PLL performance.

Due to reference jitter

Due to variation in control voltage

Detector noise

VCO noise



Σ - Δ Quantization n

Due to uncertainty involved in discretization

<u>Monte Carlo simulation</u> (VCO)

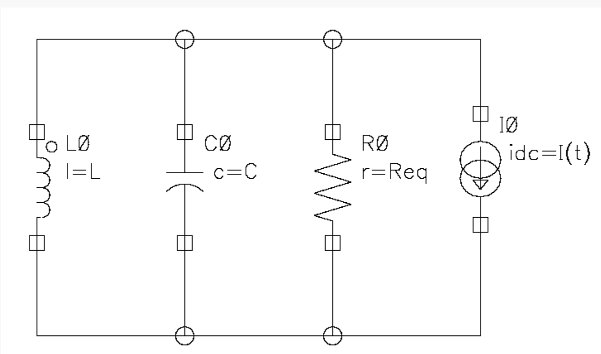
An oscillator is a circuit capable of maintaining electric oscillations.

Frequency of oscillation $=1/(LC)^{1/2}$

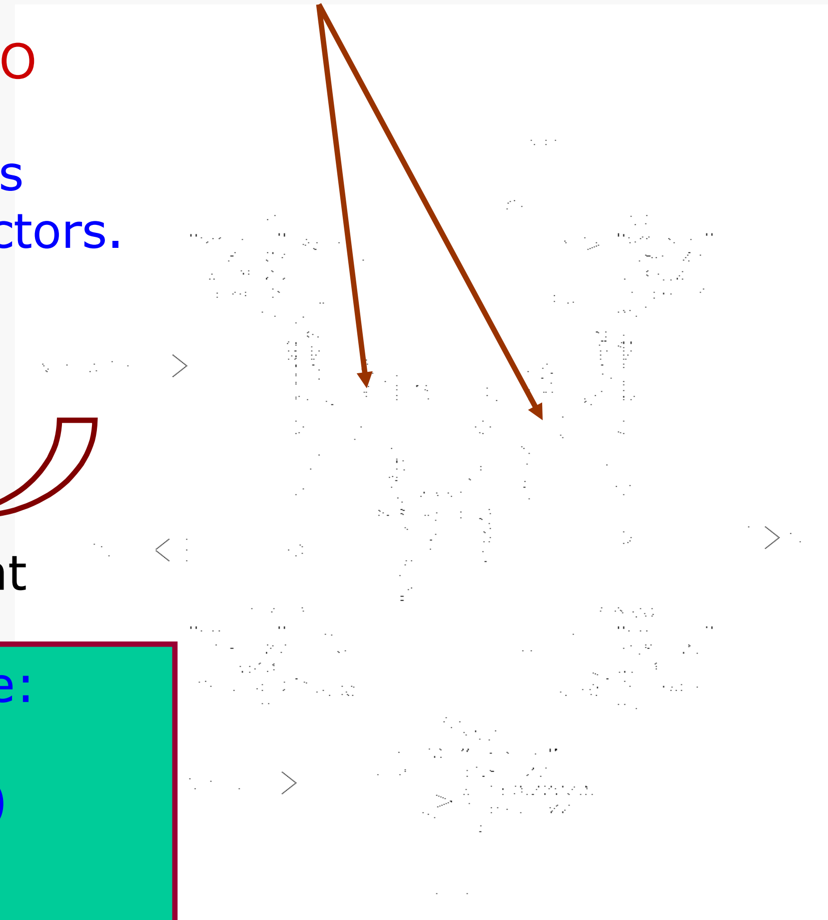
Controlled by voltage dependent capacitance (varactor)

Complimentary Cross-Coupled LC VCO

- Power efficient since bias current is shared between the two transconductors.



equivalent



For operation in current -limited regime:

$VO = (4/\pi).I_{bias}.R_{eq}$ (Ideal switching)

$VO(apx) = I_{bias}.R_{eq}$ (High frequency)

Monte Carlo simulation (VCO – Phase Noise)

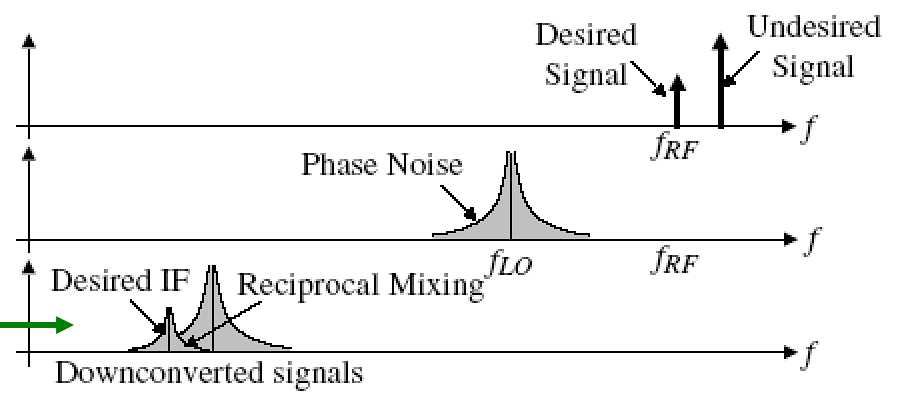
Causes of spectral purity degradation (phase noise):

1.) Random noise in the reference input, the PFD, loop filter and VCO (also dividers if the PLL is a frequency synthesizer)

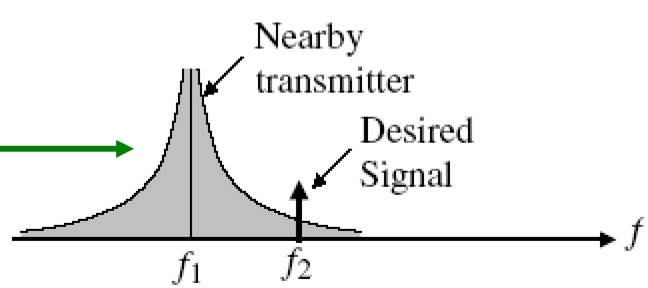
2.) Spurious sidebands – high energy sidebands with no harmonic relationship to the generated output signal. It is systematic in origin.

Why is spectral purity important?

- Phase noise can degrade the sensitivity of a receiver due to reciprocal mixing



- Phase noise produces adjacent channel interference



Monte Carlo simulation (VCO – Phase Noise)

How do the process and mismatch variation affect phase noise?

-- **we will perform monte carlo analysis to assess this**.

- Step1 –Varying the process parameter only
- Step2 – investigating the device mismatch(in diff VCO one side mismatched to the other) in presence of process variation

Cadence Spectre modeling:

•The statistics block contains the distributions for parameters:

Distributions specified in the process block are sampled once per Monte Carlo run, are applied at global scope, and are used typically to represent batch-to-batch (process) variations.

• Distributions specified in the mismatch block are applied on a per-subcircuit instance basis, are sampled once per subcircuit instance, and are used typically to represent device-to-device (on chip) mismatch for devices on the same chip.

Monte Carlo simulation (VCO – Phase Noise) model file

Process section

Define statistical blocks in the model file (ideally it should be provided from the foundry)

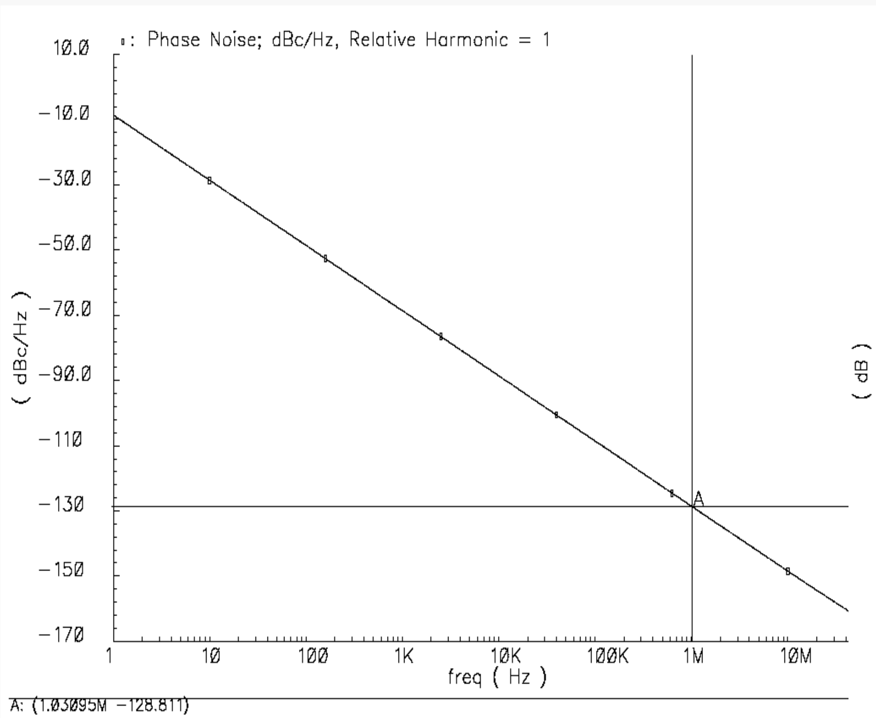
Mismatch section

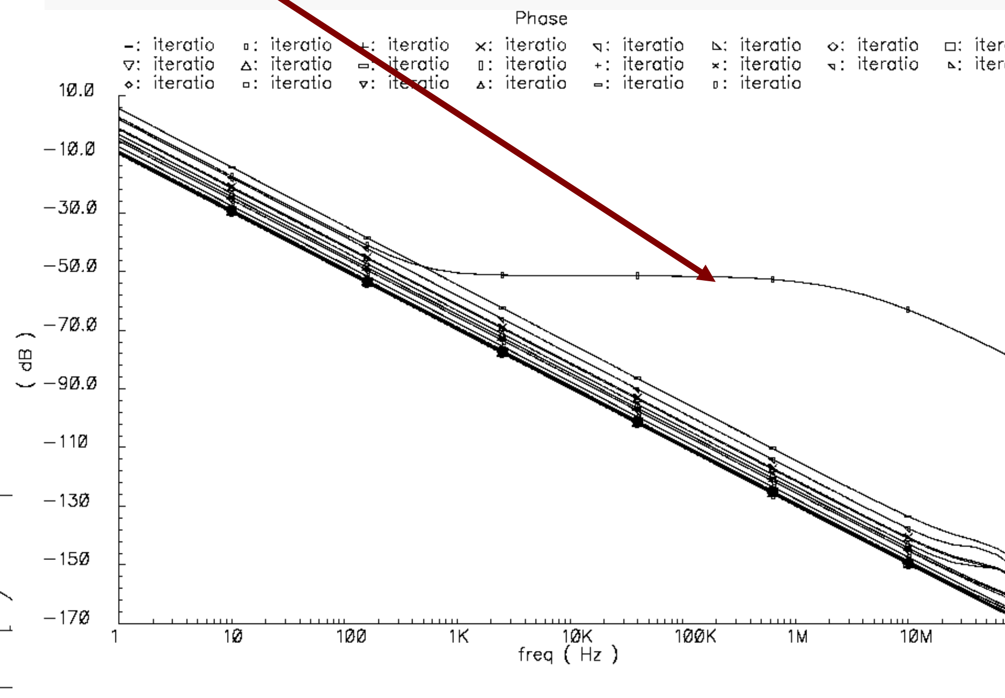
```
simulator lang=spectre

statistics {
        process {
                    vary    nmos_rgfac          dist=gauss std=0.09
                    vary    pmos_rgfac          dist=gauss std=0.09
                    vary    nmos_rsubfac        dist=gauss std=0.10
                    vary    pmos_rsubfac        dist=gauss std=0.10
                    vary    nmos_uefffac        dist=gauss std=0.13
                    vary    pmos_uefffac        dist=gauss std=0.13
                    vary    nmos_vthfac         dist=gauss std=0.12
                    vary    pmos_vthfac         dist=gauss std=0.12
                    vary    nmos_toxfac         dist=gauss std=0.05
                    vary    pmos_toxfac         dist=gauss std=0.05
                    vary    nmos_dlfac          dist=gauss std=0.08
                    vary    pmos_dlfac          dist=gauss std=0.08
                    vary    nmos_dwfac          dist=gauss std=0.10
                    vary    pmos_dwfac          dist=gauss std=0.10
                    vary    nmos_rdswfac        dist=gauss std=0.10
                    vary    pmos_rdswfac        dist=gauss std=0.10
                    vary    nmos_k1fac          dist=gauss std=0.10
                    vary    pmos_k1fac          dist=gauss std=0.10
                    vary    nmos_cgdfac         dist=gauss std=0.12
                    vary    pmos_cgdfac         dist=gauss std=0.12
                    vary    ndio_cjfac          dist=gauss std=0.11
                    vary    pdio_cjfac          dist=gauss std=0.11

                   }
        mismatch {
                    vary    nmos_rgfac          dist=gauss std=0.02
                    vary    pmos_rgfac          dist=gauss std=0.02
                    vary    nmos_rsubfac        dist=gauss std=0.01
                    vary    pmos_rsubfac        dist=gauss std=0.01
                    vary    nmos_uefffac        dist=gauss std=0.02
                    vary    pmos_uefffac        dist=gauss std=0.02
                    vary    nmos_vthfac         dist=gauss std=0.01
                    vary    pmos_vthfac         dist=gauss std=0.01
                    vary    nmos_toxfac         dist=gauss std=0.01
                    vary    pmos_toxfac         dist=gauss std=0.01
                    vary    nmos_dlfac          dist=gauss std=0.03
                    vary    pmos_dlfac          dist=gauss std=0.03
                    vary    nmos_dwfac          dist=gauss std=0.01
                    vary    pmos_dwfac          dist=gauss std=0.01
                    vary    nmos_rdswfac        dist=gauss std=0.02
                    vary    pmos_rdswfac        dist=gauss std=0.02
```

Monte Carlo simulation (VCO – Phase Noise)→STEP-1

Running Monte Carlo for process variation only

With applied statistical variation(in model file) an increase in noise can be observed, and at this run resulted noise is worst and unacceptable.





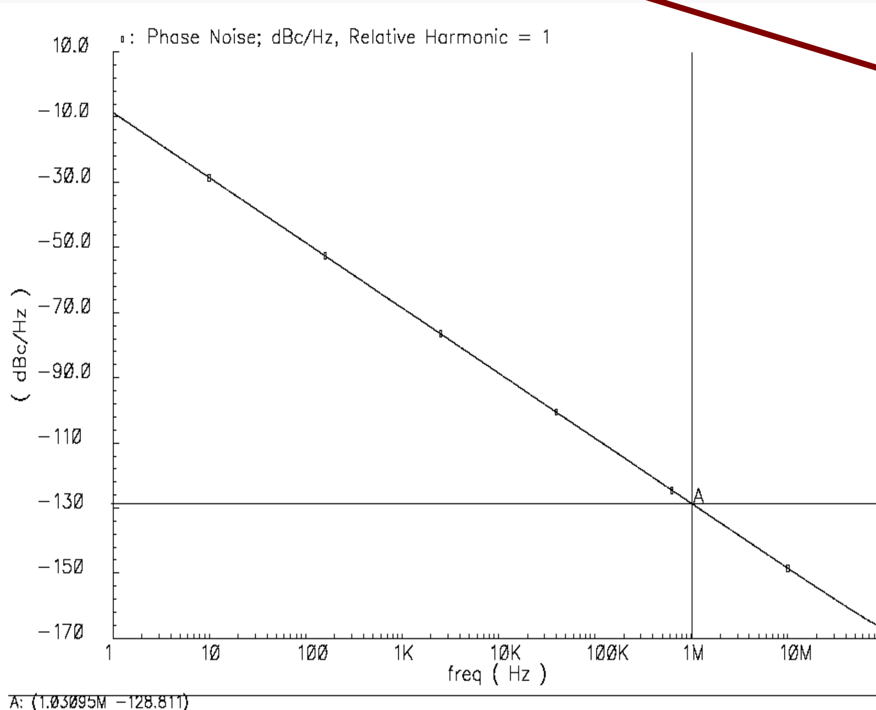
Normal simulation

Monte Carlo simulation

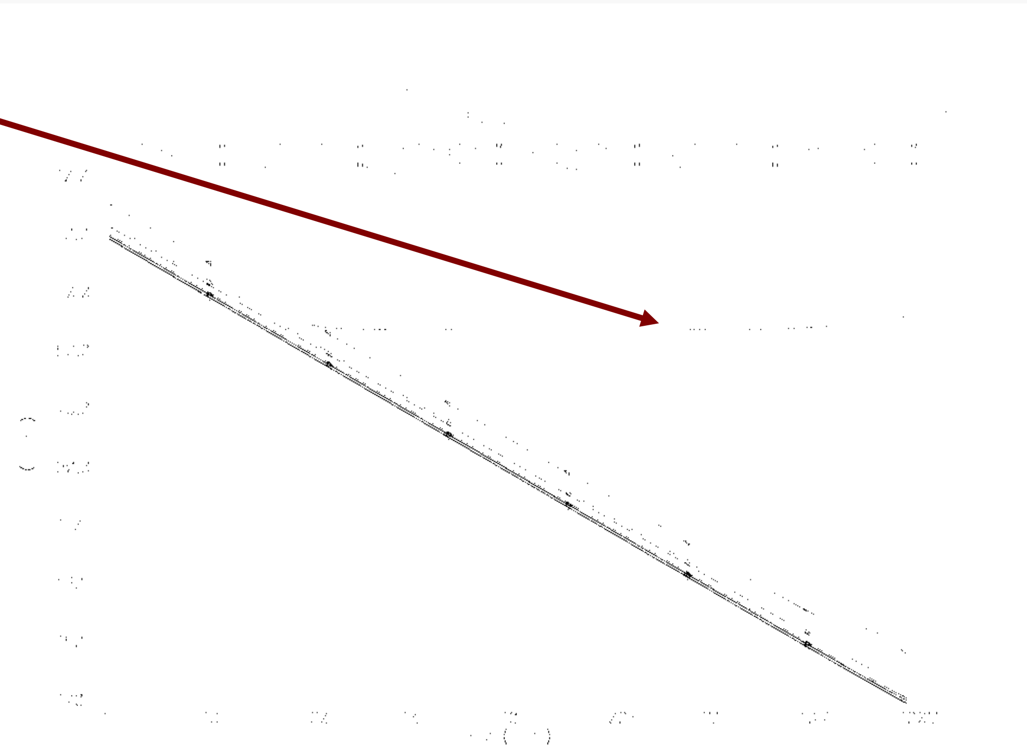
Monte Carlo simulation (VCO – Phase Noise) →STEP-2

Running Monte Carlo for mismatch in 2 sides of Diff. VCO

Again similar looking but not the same results appears and noise at this run is unacceptable .



Normal simulation



Monte Carlo simulation

Note: *When the same parameter is subject to both process and mismatch variations, the sampled process value becomes the mean for the mismatch random number generator for that particular parameter.*

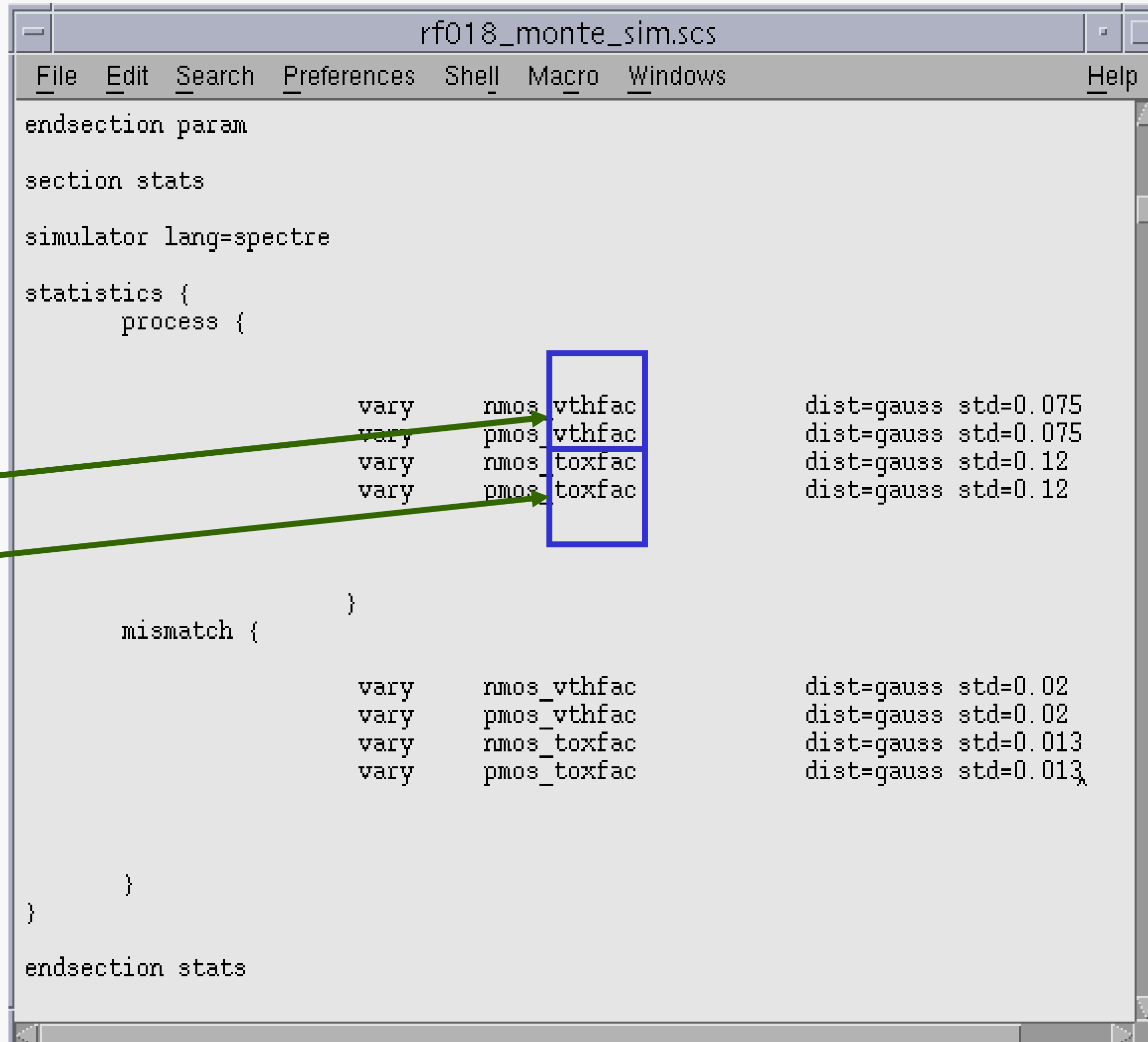
Monte Carlo simulation (VCO – Phase Noise)-- more insight

To get more insight we will vary only few parameter and check how values are assigned for different run as well as the simulation result

Defining variation for only two parameters in the model file -→

-Vth

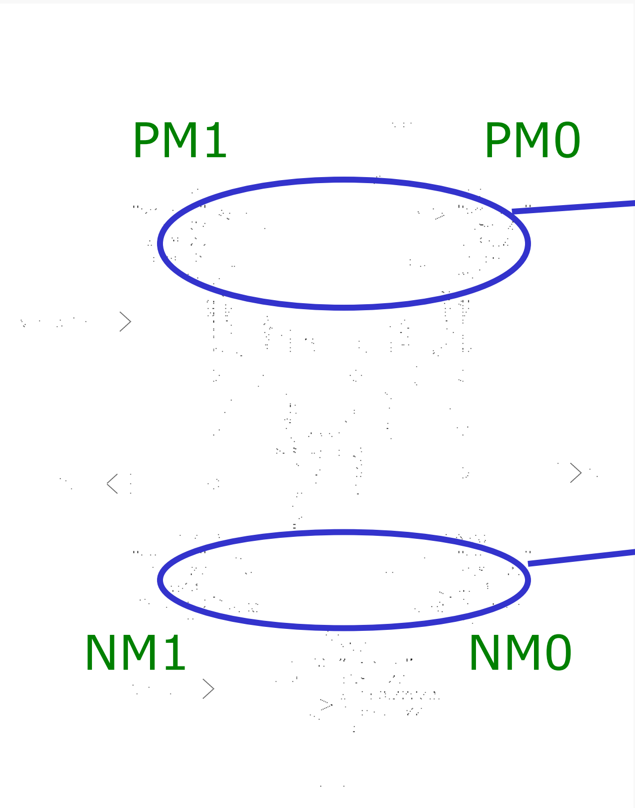
-tox




```
endsection param

section stats

simulator lang=spectre

statistics {
        process {

                        vary    nmos_vthfac        dist=gauss std=0.075
                        vary    pmos_vthfac        dist=gauss std=0.075
                        vary    nmos_toxfac        dist=gauss std=0.12
                        vary    pmos_toxfac        dist=gauss std=0.12

                       }
        mismatch {

                        vary    nmos_vthfac        dist=gauss std=0.02
                        vary    pmos_vthfac        dist=gauss std=0.02
                        vary    nmos_toxfac        dist=gauss std=0.013
                        vary    pmos_toxfac        dist=gauss std=0.013

        }
}

endsection stats
```

Monte Carlo simulation (VCO – Phase Noise)-- more insight

Process variation only



Window Expressions Info Help 9

iteration	MP("/I10/PM1"	MP("/I10/PM1"	MP("/I10/PM1"
string	vtho	tox	cj
1	-435.5m	4.825n	1.121u
2	-457.5m	4.25n	1.121u
3	-431.4m	4.127n	1.121u
4	-407m	3.508n	1.121u

iteration	MP("I10.PM0" "	MP("I10.PM0" "	MP("I10.PM0" "
string	vtho	tox	cj
1	-435.5m	4.825n	1.121u
2	-457.5m	4.25n	1.121u
3	-431.4m	4.127n	1.121u
4	-407m	3.508n	1.121u

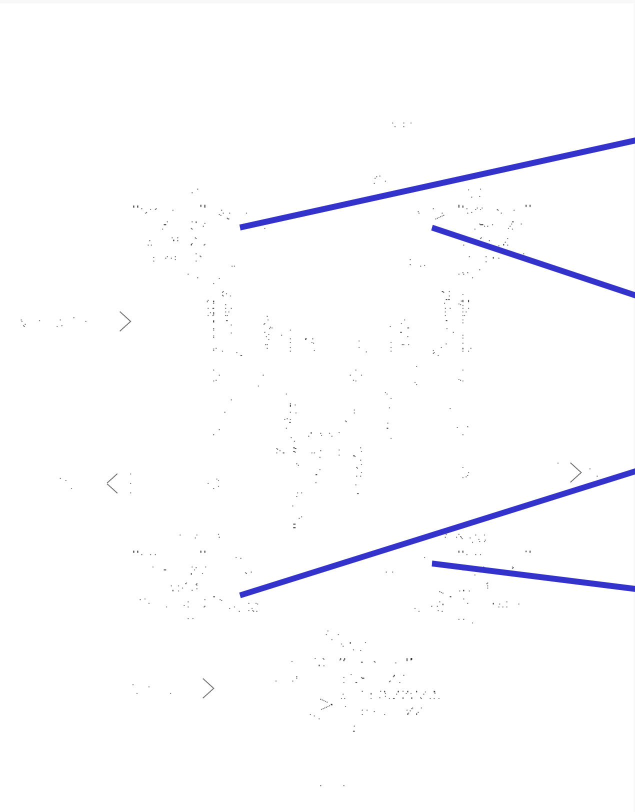
iteration	MP("/I10/NM1"	MP("/I10/NM1"	MP("/I10/NM1"
string	vtho	tox	cj
1	360.2m	4.516n	1u
2	563m	4.168n	1u
3	465.6m	4.53n	1u
4	475.7m	3.547n	1u

iteration	MP("I10.NM0" "	MP("I10.NM0" "	MP("I10.NM0" "
string	vtho	tox	cj
1	360.2m	4.516n	1u
2	563m	4.168n	1u
3	465.6m	4.53n	1u
4	475.7m	3.547n	1u

Here both nmos (pmos) transistors have been assigned same process variation.In each run they take on different parameter according to distribution defined

Monte Carlo simulation (VCO – Phase Noise)-- more insight

Process and Mismatch both variation together,with correlation of 0.2 between the two nmos(pmos) transistor



Window Expressions Info Help

iteration	MP("/I10/PM1"	MP("/I10/PM1"	MP("/I10/PM1"
string	vtho	tox	cj
1	-439.6m	4.751n	1.121u
2	-455.1m	4.233n	1.121u
3	-436.8m	4.122n	1.121u
4	-419.7m	3.565n	1.121u

iteration	MP("/I10/NM1"	MP("/I10/NM1"	MP("/I10/NM1"
string	vtho	tox	cj
1	394.7m	4.472n	1u
2	536.6m	4.159n	1u
3	468.4m	4.485n	1u
4	475.5m	3.601n	1u

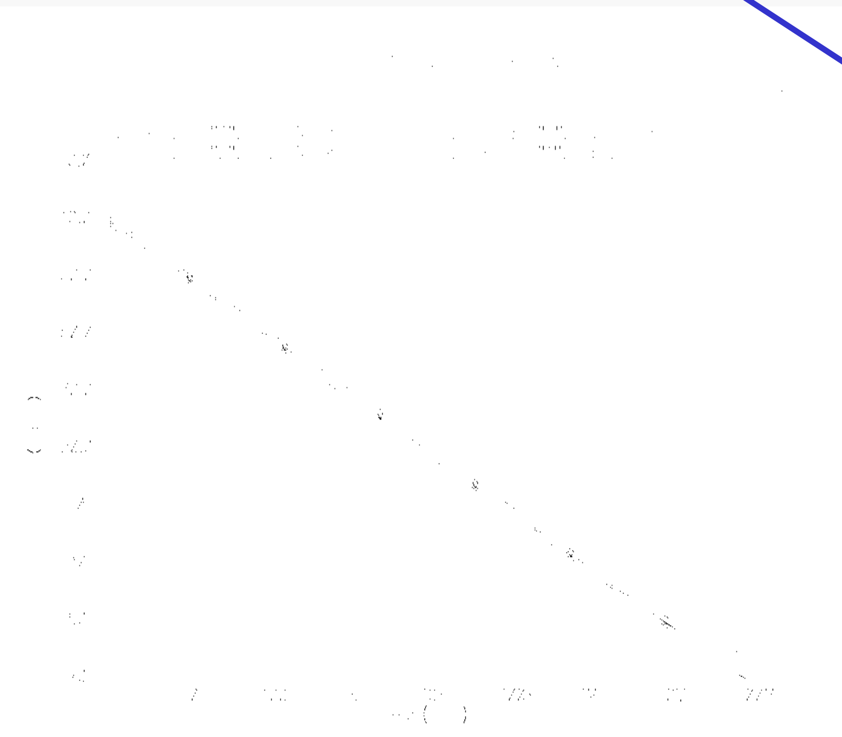
iteration	MP("I10.PM0" "	MP("I10.PM0" "	MP("I10.PM0" "
string	vtho	tox	cj
1	-440.3m	4.713n	1.121u
2	-454.7m	4.224n	1.121u
3	-437.7m	4.12n	1.121u
4	-421.9m	3.594n	1.121u

iteration	MP("I10.NM0" "	MP("I10.NM0" "	MP("I10.NM0" "
string	vtho	tox	cj
1	400.4m	4.45n	1u
2	532.2m	4.155n	1u
3	468.9m	4.462n	1u
4	475.5m	3.627n	1u

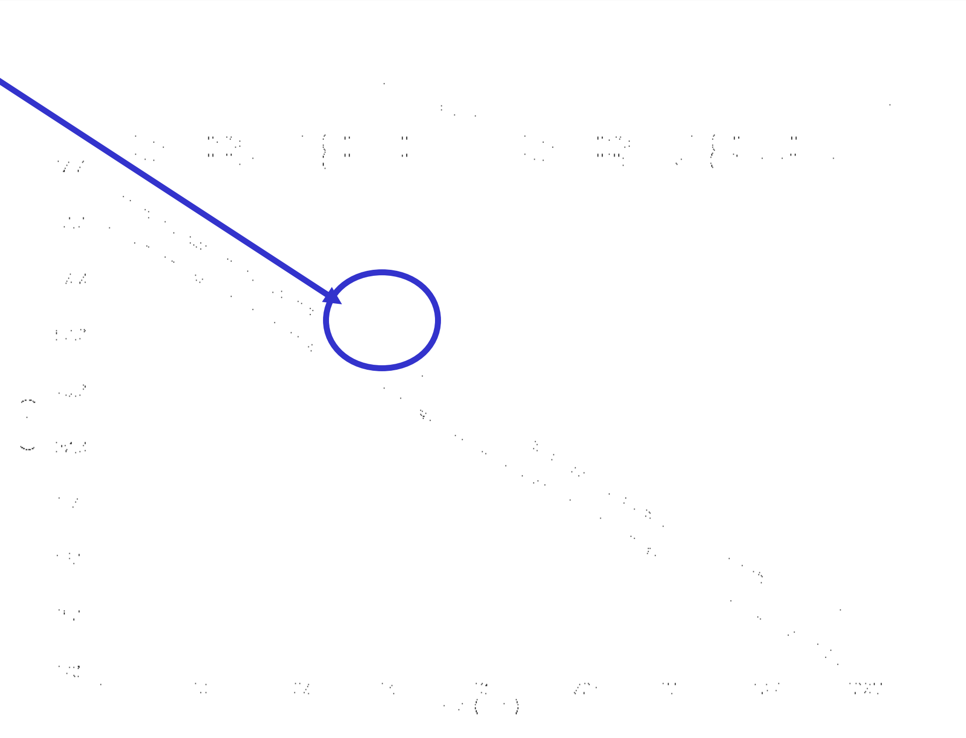
As conspicuous each nmos(pmos) transistor is getting different parameter value in each run.

Monte Carlo simulation (VCO – Phase Noise)-- more insight

As visible in the case of process variation with device mismatch noise has been increased.



Process variation only

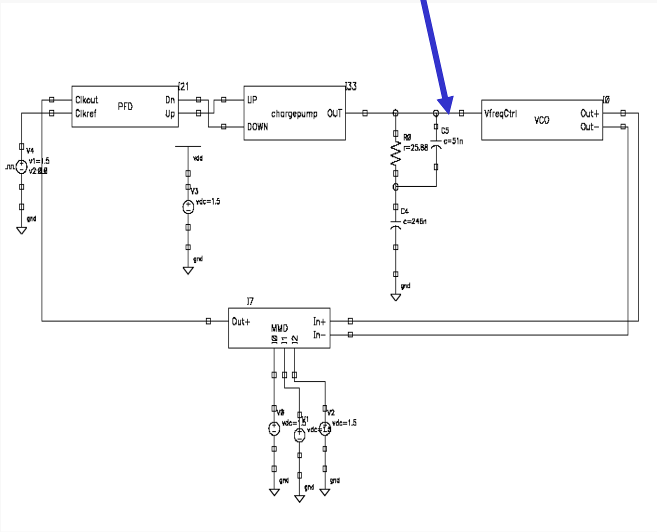


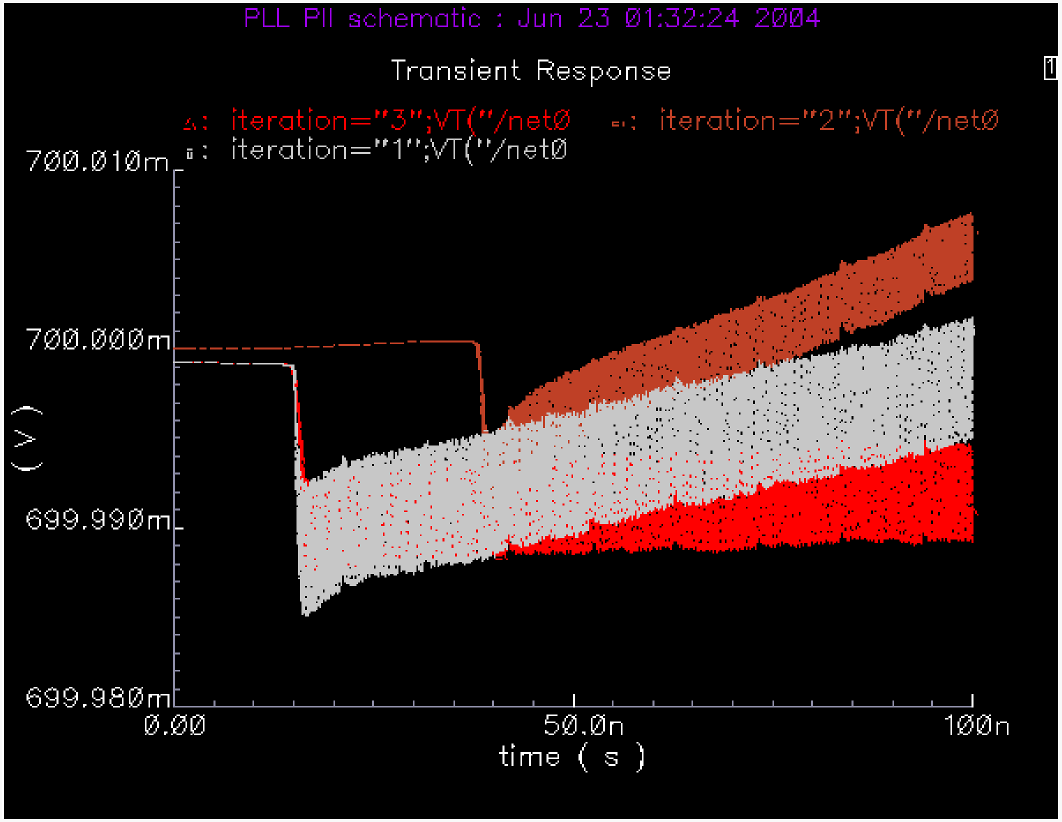
Process & mismatch variation

Monte Carlo simulation (PLL at a glance)

In a PLL all these process variation can degrade it's overall performance significantly. To see the impact of process variation we probe the output after the loop filter.

As clear in one case control voltage (i.e. loop filter output) is ramping rapidly compare to other and thus will result in different performance.





Monte Carlo simulation

Monte Carlo simulation

- In our design PLL has a settling time of 65 us.To simply run the analysis(transistor level) for this much period may take 2-3 days on a single machine.
- To do monte carlo simulation even for 10 run will make the situation worse.

➢To speed up Monte Carlo analyses—to make them run in minutes as opposed to days—

--We need to reduce the run time and can utilize Parallel simulation.

--Such as variance reduction technique can be employed.

Monte Carlo simulation (Seed no & parallel simulation)

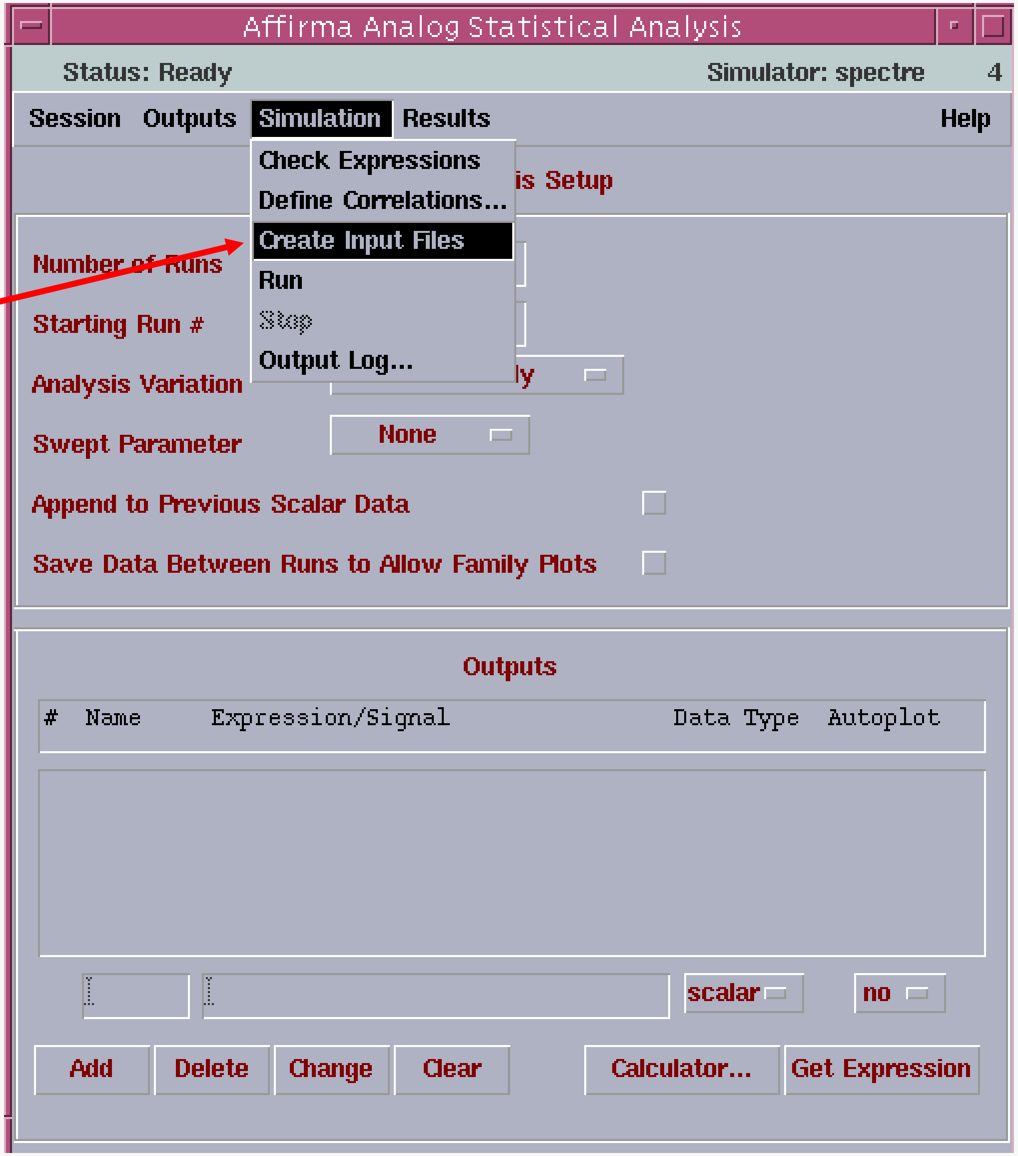
Seed

If Monte Carlo simulation for different seed is required then…..

Step 1.Create netlist(input file)

a)Either from analog artist *or*

b)Tools→ monte carlo → simulation→create_input_files



Note:

(1) Input file should have '.scs' extension (for exa.input.scs)

(2)In spectre one can not specify different seed from GUI(by default it always takes seed=1).

Monte Carlo simulation (Seed no & parallel simulation)

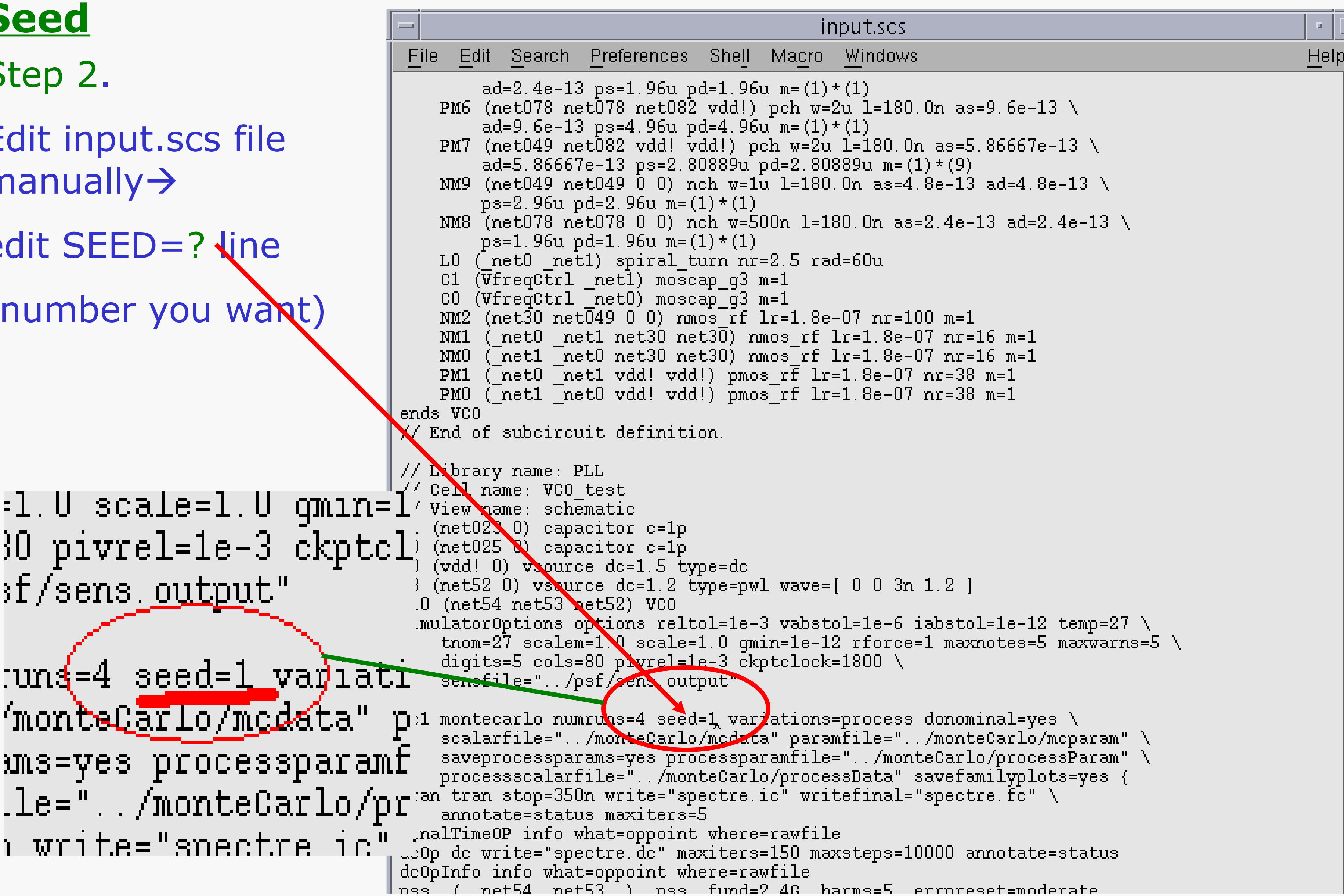
Seed

Step 2.

Edit input.scs file manually→

edit SEED=? line

(number you want)




```
        ad=2.4e-13 ps=1.96u pd=1.96u m=(1)*(1)
    PM6 (net078 net078 net082 vdd!) pch w=2u l=180.0n as=9.6e-13 \
        ad=9.6e-13 ps=4.96u pd=4.96u m=(1)*(1)
    PM7 (net049 net082 vdd! vdd!) pch w=2u l=180.0n as=5.86667e-13 \
        ad=5.86667e-13 ps=2.80889u pd=2.80889u m=(1)*(9)
    NM9 (net049 net049 0 0) nch w=1u l=180.0n as=4.8e-13 ad=4.8e-13 \
        ps=2.96u pd=2.96u m=(1)*(1)
    NM8 (net078 net078 0 0) nch w=500n l=180.0n as=2.4e-13 ad=2.4e-13 \
        ps=1.96u pd=1.96u m=(1)*(1)
    L0 (_net0 _net1) spiral_turn nr=2.5 rad=60u
    C1 (VfreqCtrl _net1) moscap_g3 m=1
    C0 (VfreqCtrl _net0) moscap_g3 m=1
    NM2 (net30 net049 0 0) nmos_rf lr=1.8e-07 nr=100 m=1
    NM1 (_net0 _net1 net30 net30) nmos_rf lr=1.8e-07 nr=16 m=1
    NM0 (_net1 _net0 net30 net30) nmos_rf lr=1.8e-07 nr=16 m=1
    PM1 (_net0 _net1 vdd! vdd!) pmos_rf lr=1.8e-07 nr=38 m=1
    PM0 (_net1 _net0 vdd! vdd!) pmos_rf lr=1.8e-07 nr=38 m=1
ends VCO
// End of subcircuit definition.

// Library name: PLL
// Cell name: VCO_test
// View name: schematic
  (net023 0) capacitor c=1p
  (net025 0) capacitor c=1p
  (vdd! 0) vsource dc=1.5 type=dc
  (net52 0) vsource dc=1.2 type=pwl wave=[ 0 0 3n 1.2 ]
 .0 (net54 net53 net52) VCO
 .mulatorOptions options reltol=1e-3 vabstol=1e-6 iabstol=1e-12 temp=27 \
    tnom=27 scalem=1.0 scale=1.0 gmin=1e-12 rforce=1 maxnotes=5 maxwarns=5 \
    digits=5 cols=80 pivrel=1e-3 ckptclock=1800 \
    sensfile="../psf/sens.output"

 1 montecarlo numruns=4 seed=1 variations=process donominal=yes \
    scalarfile="../monteCarlo/mcdata" paramfile="../monteCarlo/mcparam" \
    saveprocessparams=yes processparamfile="../monteCarlo/processParam" \
    processscalarfile="../monteCarlo/processData" savefamilyplots=yes {
 an tran stop=350n write="spectre.ic" writefinal="spectre.fc" \
    annotate=status maxiters=5
 .nalTimeOP info what=oppoint where=rawfile
 .cOp dc write="spectre.dc" maxiters=150 maxsteps=10000 annotate=status
dcOpInfo info what=oppoint where=rawfile
pss ( net54 net53 ) pss fund=2.4G harms=5 errpreset=moderate
```

```
=1.0 scale=1.0 gmin=1
30 pivrel=1e-3 ckptcl
sf/sens.output"

runs=4 seed=1 variati
/monteCarlo/mcdata" p
ams=yes processparamf
le="../monteCarlo/pr
 write="spectre.ic"
```

Monte Carlo simulation (Seed no & parallel simulation)

Seed

Step 3.Run spectre from command line with option for example…..

spectre -env artist4.4.6 +log ../psf/spectre.out -format psfbin -raw ../psf   input.scs

Here one should execute spectre command(or executable file) from the netlist directory.

For example one wants to simulate "PLL" design from command line

Then go to your simulation directory

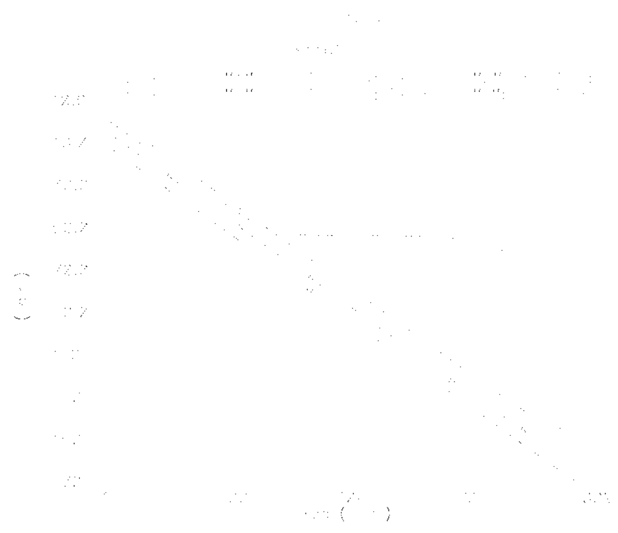
cd …/simulation/pll/spectre/schematic/netlist

and here execute spectre command

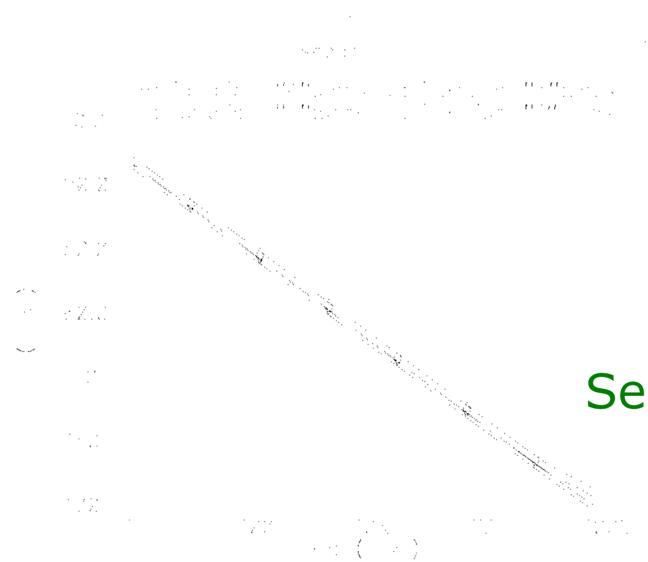
Monte Carlo simulation (Seed no & parallel simulation)

Seed

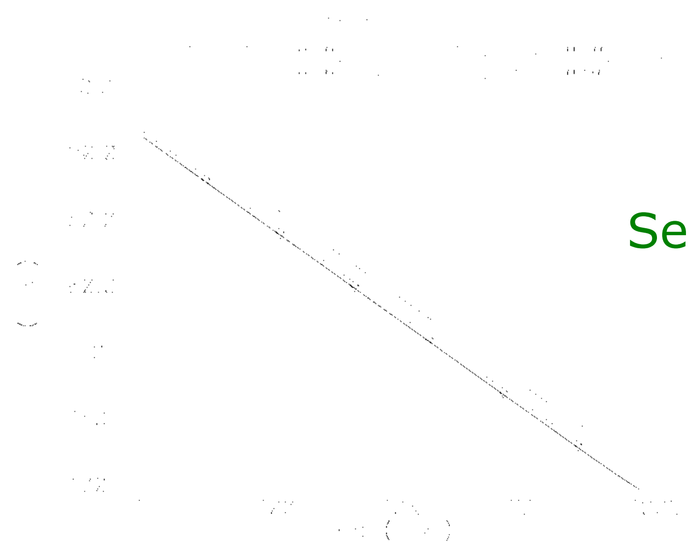
Step 4.Results can be plotted with either from calculator or from Monte Carlo tool…..



Seed=1



Seed=3



Seed=11

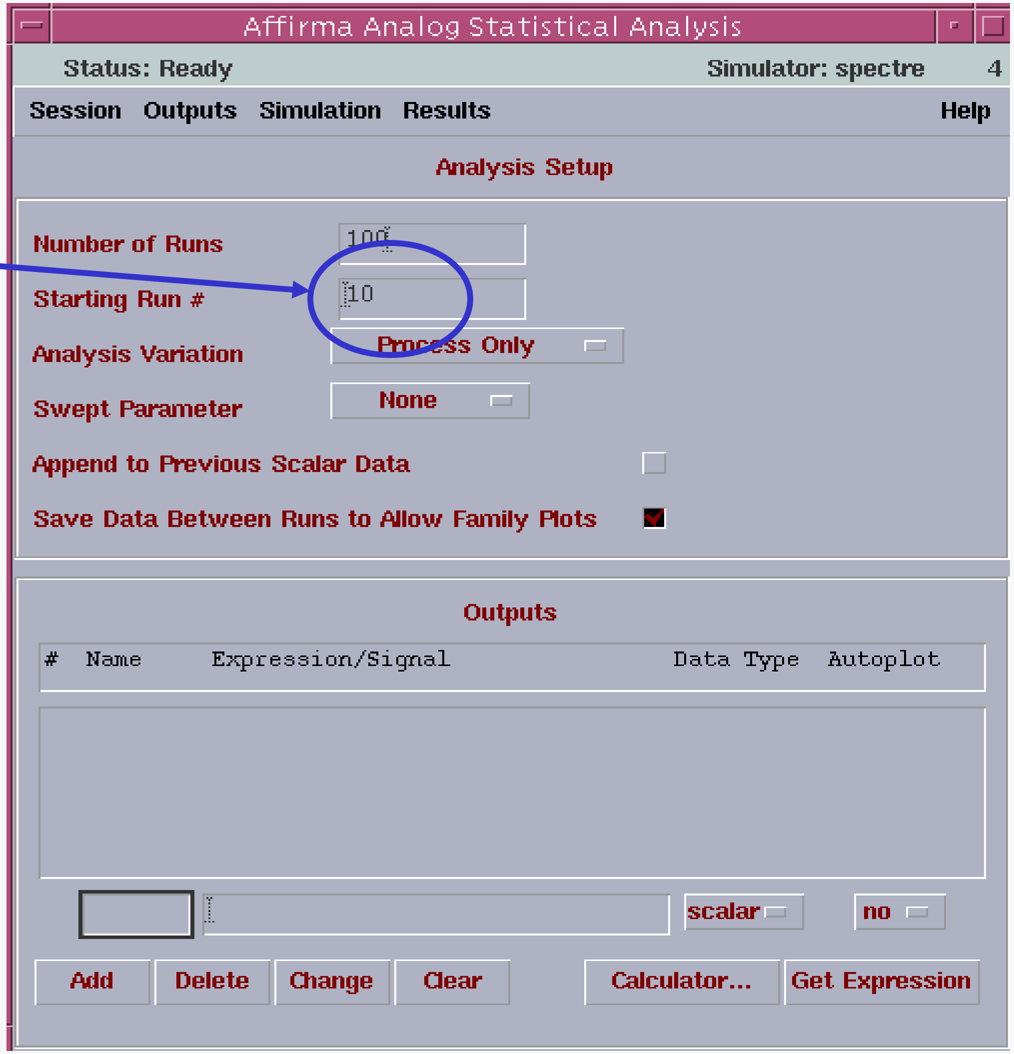
Fig:Plots for different seed value simulation

Monte Carlo simulation (Seed no & parallel simulation)

A way around from GUI

Another way of doing similar thing(giving different seed value) from GUI would be to start simulation from different run,or say to skip some initial run as shown in the fig.

- Here it will skip first 10 runs and simulate from 11'th to 110'th run for 100 iteration
- This is quite similar to assigning different seed value.
- *But beware skipping these runs could take much longer time for a complex design*



Monte Carlo simulation (Seed no & parallel simulation)

Running multiple analysis from one file

This can be done by defining multiple monte carlo analysis statement in the input file as shown below

Analysis 1

Analysis 2

```
mc1 montecarlo numruns=4 seed=1 variations=process donominal=yes \
    scalarfile="../monteCarlo/mcdata" paramfile="../monteCarlo/mcparam" \
    saveprocessparams=yes processparamfile="../monteCarlo/processParam" \
    processscalarfile="../monteCarlo/processData" savefamilyplots=yes {
tran1 tran stop=350n write="spectre.ic" writefinal="spectre.fc" \
    annotate=status maxiters=5
finalTimeOP1 info what=oppoint where=rawfile
dcOp1 dc write="spectre.dc" maxiters=150 maxsteps=10000 annotate=status
dcOpInfo1 info what=oppoint where=rawfile
pss1  (  net54  net53  )  pss  fund=2.4G  harms=5  errpreset=moderate
+    tstab=100.5n  saveinit=yes  annotate=status
pnoise1  (  net54  net53  )  pnoise  sweeptype=relative
+       relharmnum=1  start=1  stop=100M  dec=5  maxsideband=0
+       noisetype=sources  annotate=status
modelParameter1 info what=models where=rawfile
element1 info what=inst where=rawfile
outputParameter1 info what=output where=rawfile
export noExprs1=oceanEval("0")
}
saveOptions1 options save=allpub


mc2 montecarlo numruns=4 seed=5 variations=process donominal=yes \
    scalarfile="../monteCarlo/mcdata" paramfile="../monteCarlo/mcparam" \
    saveprocessparams=yes processparamfile="../monteCarlo/processParam" \
    processscalarfile="../monteCarlo/processData" savefamilyplots=yes {
tran2 tran stop=350n write="spectre.ic" writefinal="spectre.fc" \
    annotate=status maxiters=5
finalTimeOP2 info what=oppoint where=rawfile
dcOp2 dc write="spectre.dc" maxiters=150 maxsteps=10000 annotate=status
dcOpInfo2 info what=oppoint where=rawfile
pss2  (  net54  net53  )  pss  fund=2.4G  harms=5  errpreset=moderate
+    tstab=100.5n  saveinit=yes  annotate=status
pnoise2  (  net54  net53  )  pnoise  sweeptype=relative
+       relharmnum=1  start=1  stop=100M  dec=5  maxsideband=0
+       noisetype=sources  annotate=status
modelParameter2 info what=models where=rawfile
element2 info what=inst where=rawfile
outputParameter2 info what=output where=rawfile
export noExprs2=oceanEval("0")
}
saveOptions2 options save=allpub
```

Note: For each analysis a different name to child analysis(for example ac,dc,tran) and to output file has to be assigned.

Monte Carlo simulation (Seed no & parallel simulation)

Running script for executing multiple files (sequentially)

This can be done by making an executable file as shown and running it from command window

runSimulation

File Edit Search Preferences Shell Macro Windows Help

```
spectre  -env artist4.4.6 +log ../psf/spectre.out -format psfbin -raw ../psf   input_0.scs
spectre  -env artist4.4.6 +log ../psf/spectre.out -format psfbin -raw ../psf   input_1.scs
spectre  -env artist4.4.6 +log ../psf/spectre.out -format psfbin -raw ../psf   input_2.scs
spectre  -env artist4.4.6 +log ../psf/spectre.out -format psfbin -raw ../psf   input_3.scs
spectre  -env artist4.4.6 +log ../psf/spectre.out -format psfbin -raw ../psf   input_4.scs
spectre  -env artist4.4.6 +log ../psf/spectre.out -format psfbin -raw ../psf   input_5.scs
```

These file can be used to simulate different design as well as same design (with different seed value in it)

NOTE:In all cases spectre command(or executable file) must be excited from the netlist directory.

Monte Carlo simulation (Seed no & parallel simulation)

Parallel Simulation

One can easily set up queues, where a particular queue is set up using the built in Cadence"LBS" system.

1.Create a configuration file:

queueName numberOfMachines
machine1 numberOfJobs

machine2 numberOfJobs

queue2 numberOfMachines ...

e.g. parallelQueue 1 linuxMachine 4

2. Pick a machine as your queue manager, and then run:

cdsqmgr /path/to/the/queue_config

3. Before running DFII, do:

setenv LBS_CLUSTER_MASTER queueMachineName

where ***queueMachineName*** was the machine you ran cdsqmgr on.

4. Then one can submit Artist jobs as "distributed" as shown in the …….***next slide***

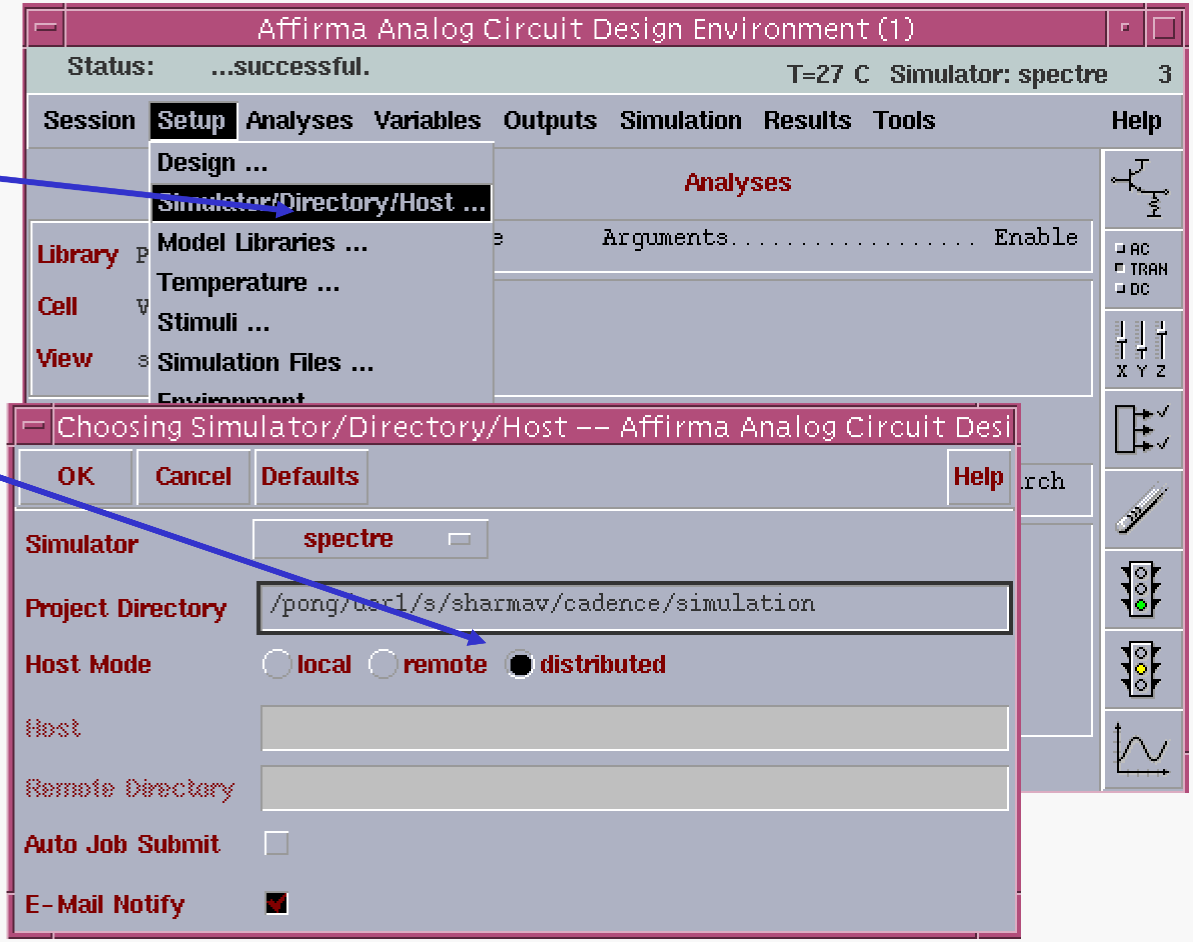
Monte Carlo simulation (Seed no & parallel simulation)

Parallel Simulation

Setting for distributive processing

1.from analog artist go to

2.set distributive and assign jobs to all machine



<u>Monte Carlo simulation</u> (Seed no & parallel simulation)

<u>References:</u>

1. Lecture notes of ***Michael Perrott*** Massachusetts Institute of Technology.

2. Lecture notes of ***Phillip Allen*** Georgia institute of technology.

3. Cadence Spectre user guide.

4. Inputs from Andrew Beckett, cadence Inc.