Characterizing Differential Amplifiers for Communications Circuits

Measuring Performance with Simulation for First Pass Success

Jonathan David

Cadence Design Systems, Inc.

Topics

- Background, Motivation & Tools
- Key Specifications for Design & Optimization
- Using the Test bench
- PVT (Process,Voltage,Temperature) Validation
- Advanced Measurements
- Using OCEAN Scripts
- Summary

First Pass Success and Design Reuse are needed to Meet Present Challenges

- Large Mixed Signal Chips
- Smaller Market Windows
- Short Development Cycles

- Lower Supply Voltages
- Smaller Processes
- Higher Performance
- Less Power

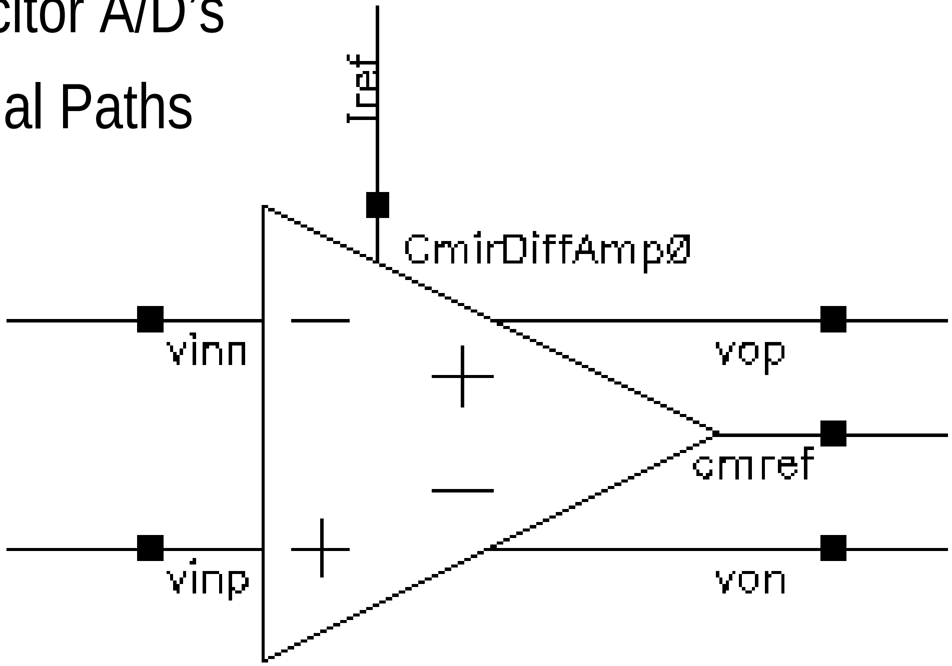
Performance Verification is our Strategy

- Spectre Simulator
- Analog Artist UI
- OCEAN scripts
- Verilog-A behavioral Language
- Diva LPE

- We need to Simulate the same circuits we Fab.

Differential Amp is Basic to Comms

- Switched Capacitor A/D's
- Differential Signal Paths



Common-Mode Feedback
is required

Amp Specifications are no secret

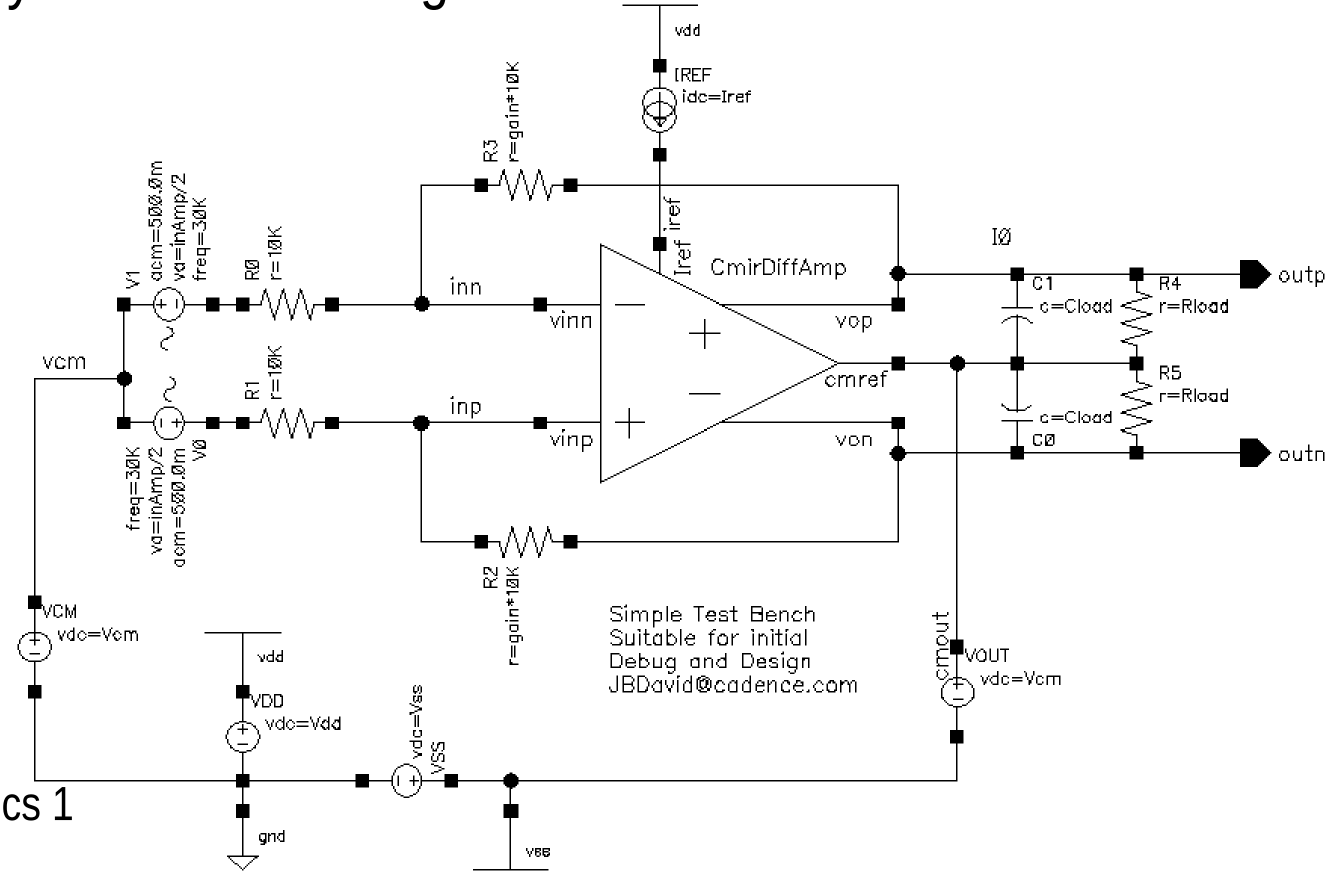
A_{OL} ϕ_{OL} vs. Freq. A_d A_c	GB	Gain & Phase Margin
V_{OS} vs Temp V_{IO} Drift	V_{OS} vs Proc/Mmatch	I_B I_{OS} I_{IO} Drift
Zin Zout	Slew Rate F_{MAX}	Overshoot Settling Time
CMR Common Mode Range	V_{OMax} I_{SC} Output Compliance	I_{Supply} vs PVT Min V_{supply}
ENV f_{BV}	ENI f_{BI}	CMRR & PSRR
THD	**SFDR**	**IP3**

Key to Design & Optimization

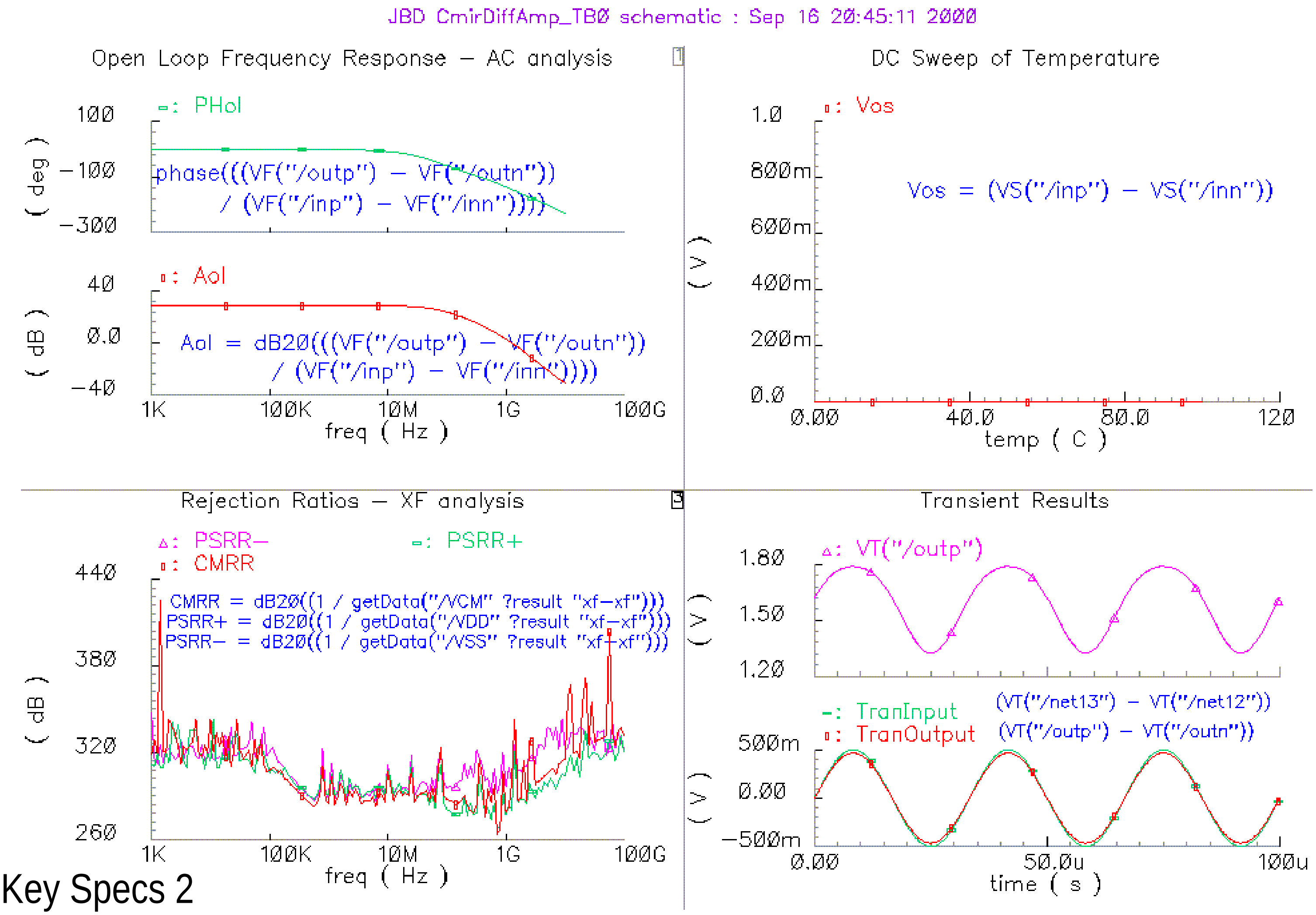
A_{OL} ϕ_{OL} vs. Freq. A_d A_c	GB	Gain & Phase Margin
V_{OS} vs Temp V_{IO} Drift	V_{OS} vs Proc/Mmatch	~~I_B I_{OS}~~ ~~I_{IO} Drift~~
Zin Zout	Slew Rate F_{MAX}	Overshoot Settling Time
CMR Common Mode Range	V_{OMax} I_{SC} Output Compliance	I_{Supply} vs PVT Min V_{supply}
ENV f_{BV}	ENI f_{BI}	CMRR & PSRR
THD	**SFDR**	**IP3**

Simplest Test Bench is Unity Gain Circuit

- Key to Understanding the Full Test Bench.



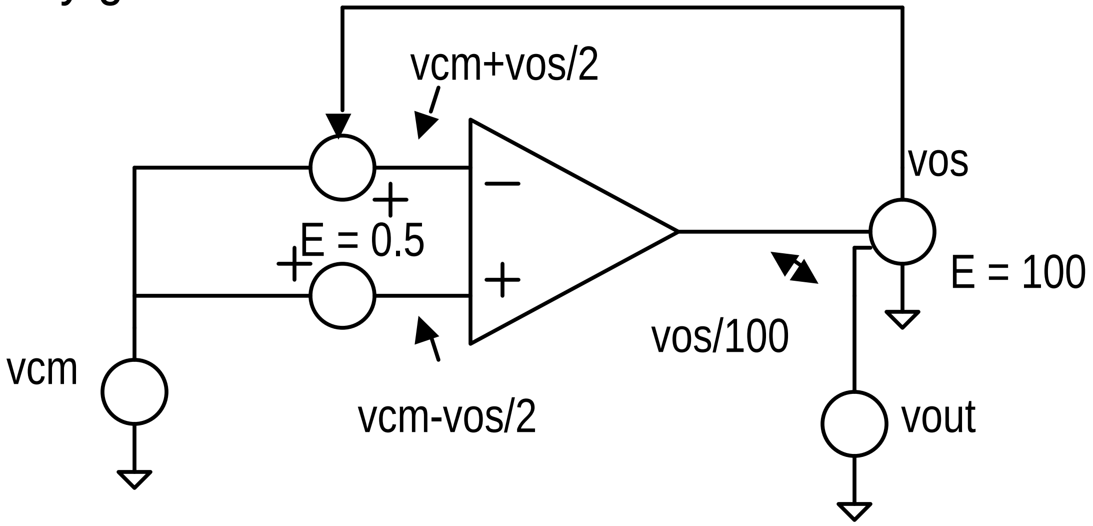
Interactive Results for Debug & Design



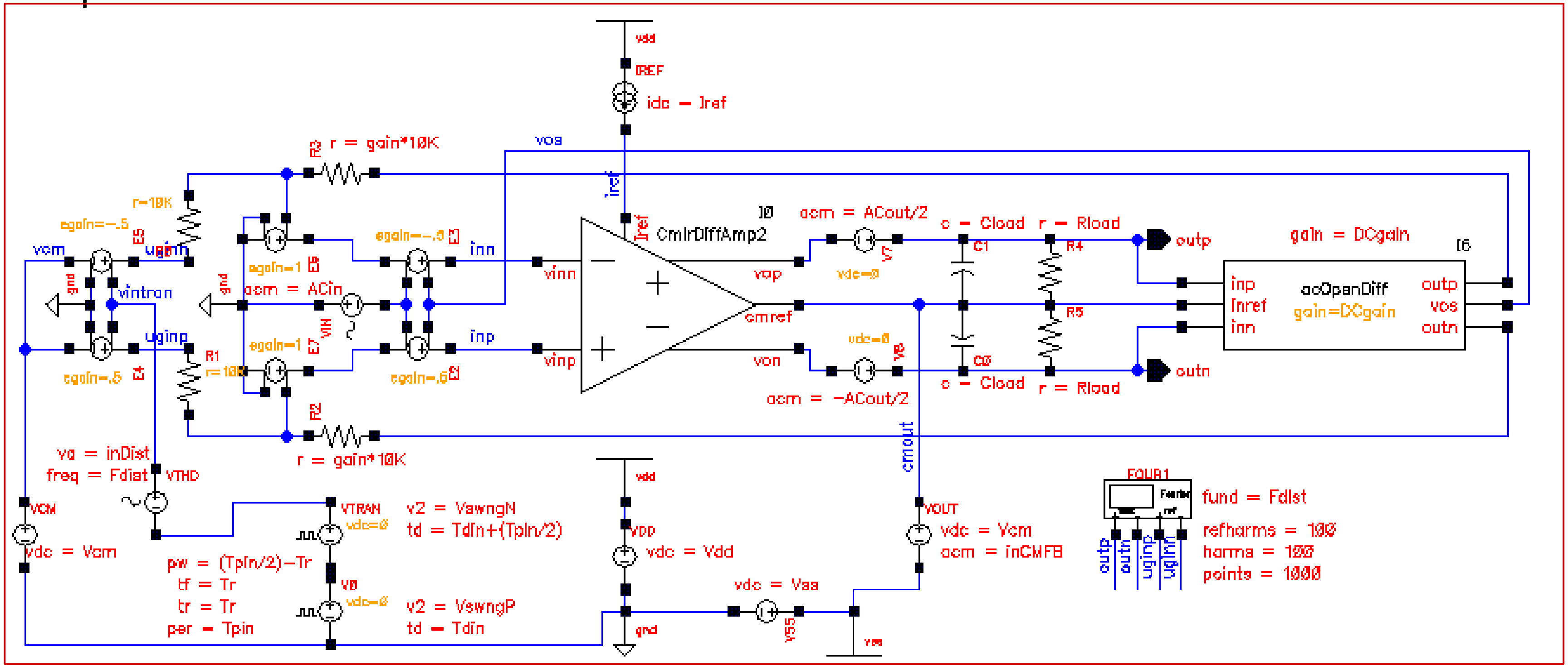
Key Specs 2

Offset Amplification Lowers Output Error

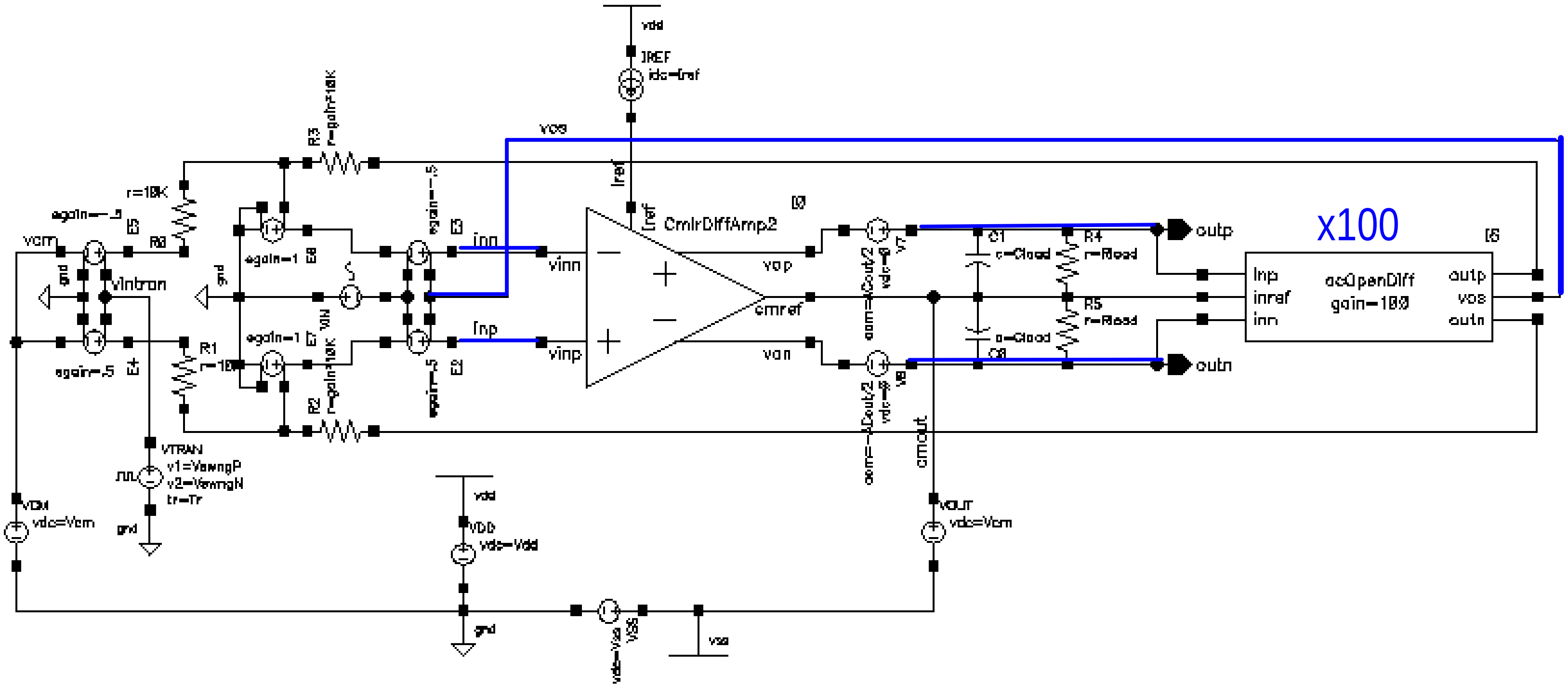
- acOpenDiff written in Verilog-A provides:
  - Output provides Gain for Offset voltage during DC-OP.
  - Output is “DC” for AC simulation, and Noise. – thus opening the loop.
  - Unity gain for Transient.



Variables are key to TB Flexibility



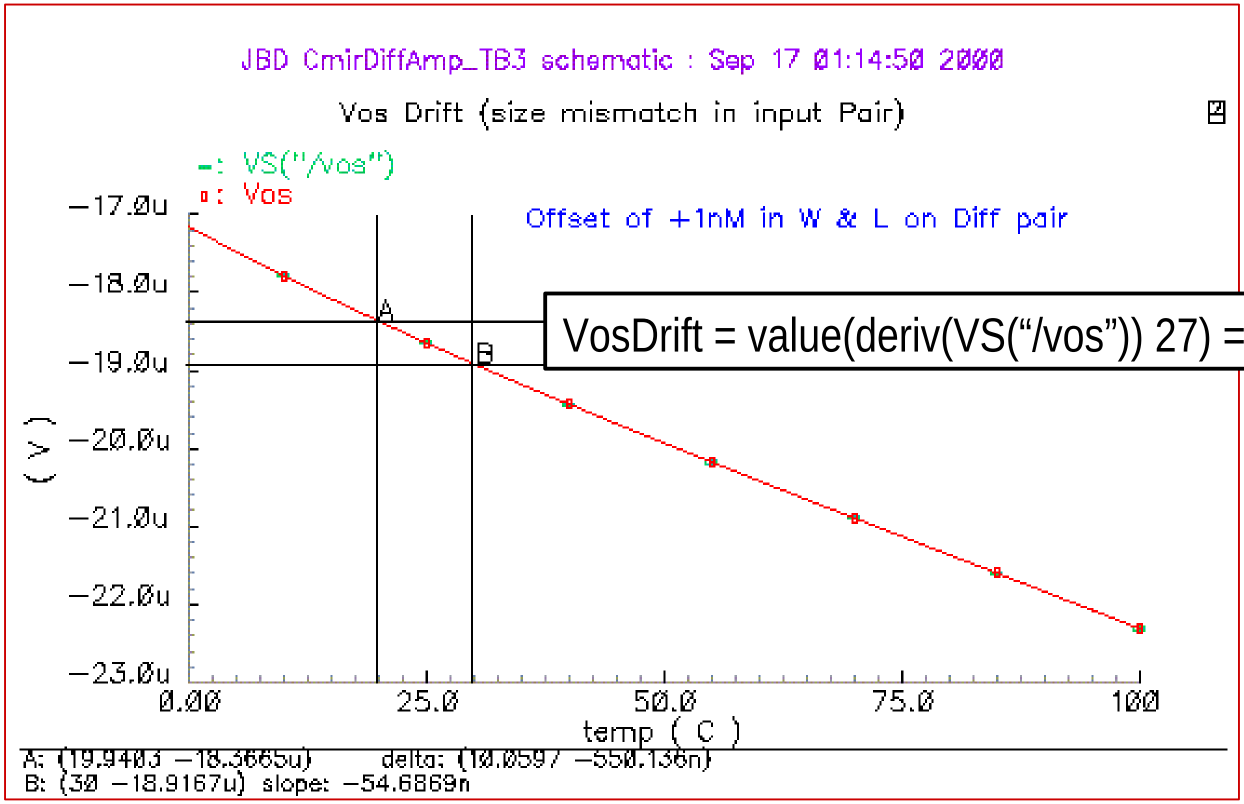
DC Operation Point Signal Loop



TB Use

Vos Drift

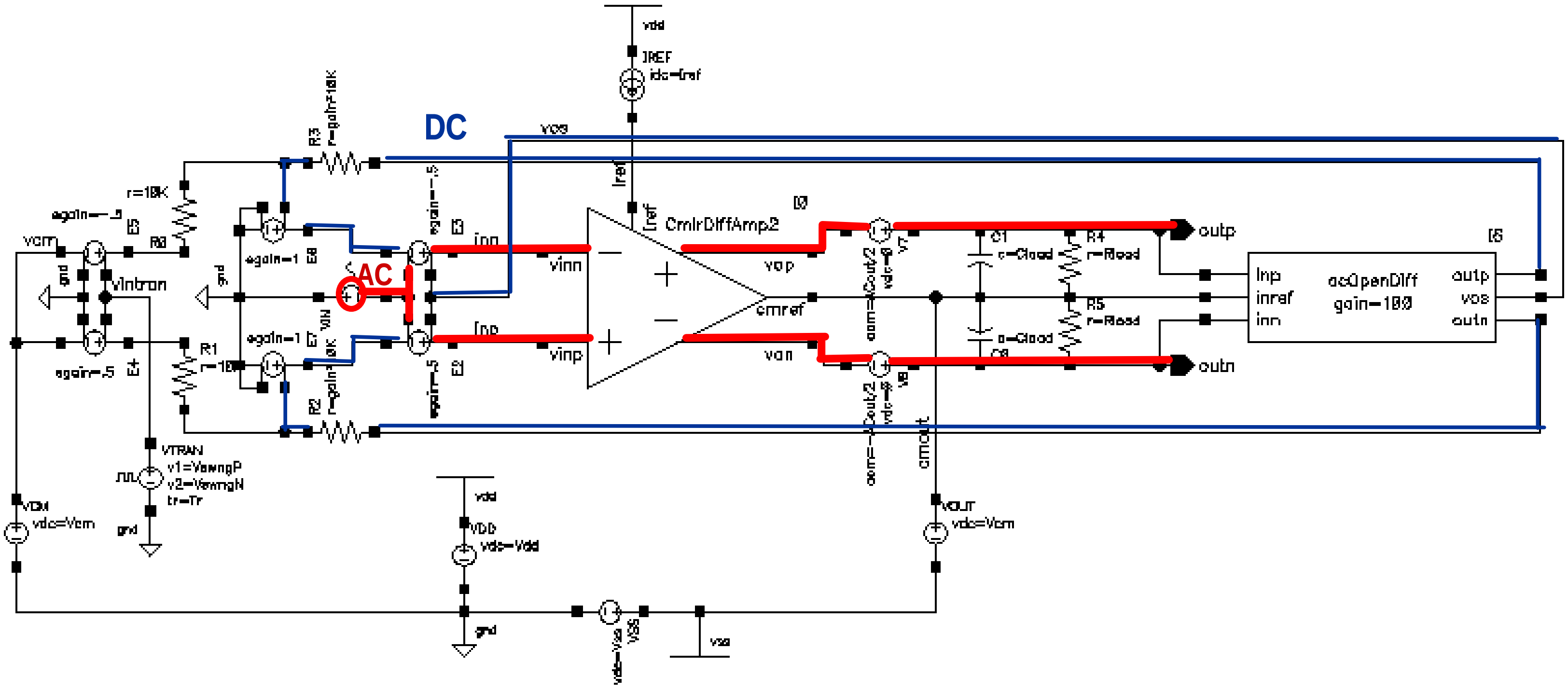
- Built-in Offset & Biasing Basic for others
- Differential Amps Ideally have no Offset.. Introduce some.



```
A: (19.9403 -18.3665u)        delta: (10.0597 -550.136n)
B: (30 -18.9167u) slope: -54.6869n
```

VosDrift = value(deriv(VS("/vos")) 27) = -54n V/°C

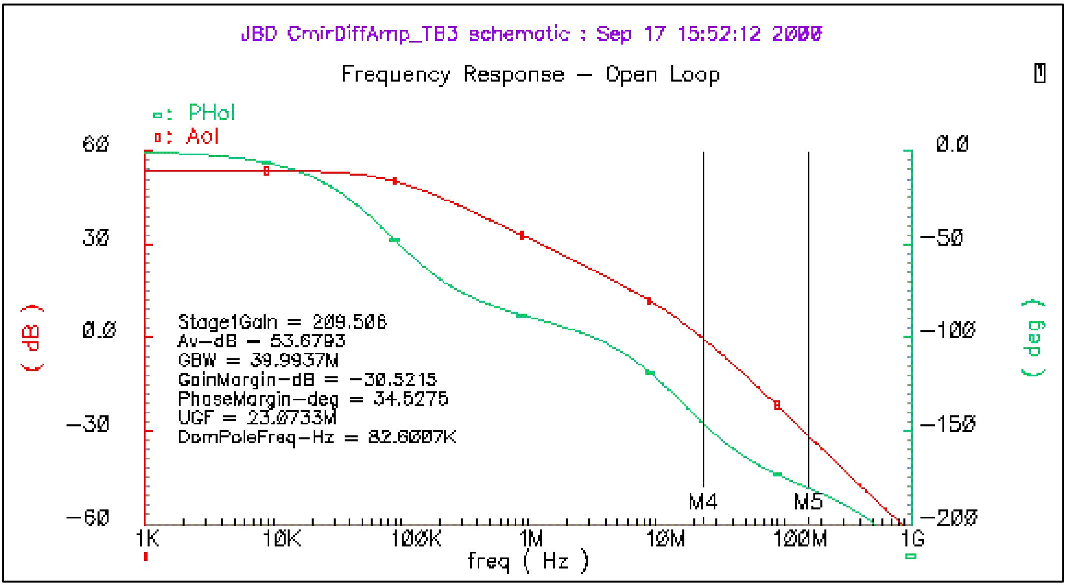
AC Open Loop Signal Path



TB Use

Open Loop Gain, Phase + Margins

- Calculate Av UGB GM, PhaseM, GBW
- Plot Aol Φol



```
Stage1Gain = 209.506
Av–dB = 53.6793
GBW = 39.9937M
GainMargin–dB = –30.5215
PhaseMargin–deg = 34.5275
UGF = 23.0733M
DomPoleFreq–Hz = 82.6007K
```

```
Av = value(db20(VF(“/outp”) –VF(“/outn”)) 0)
GBW = gainBwProd(VF(“/outp”) –VF(“/outn”))
UGB = cross(db20(VF(“/outp”) –VF(“/outn”)) 0 1 “either”)
GM = gainMargin(VF(“/outp”) –VF(“/outn”))
ΦM = phaseMargin(VF(“/outp”) –VF(“/outn”))
DomPole = bandwidth((VF(“/outp”) –VF(“/outn”)) 3 “low”)
```

TB Use

These require V(inp,inn) = 1 over entire Range

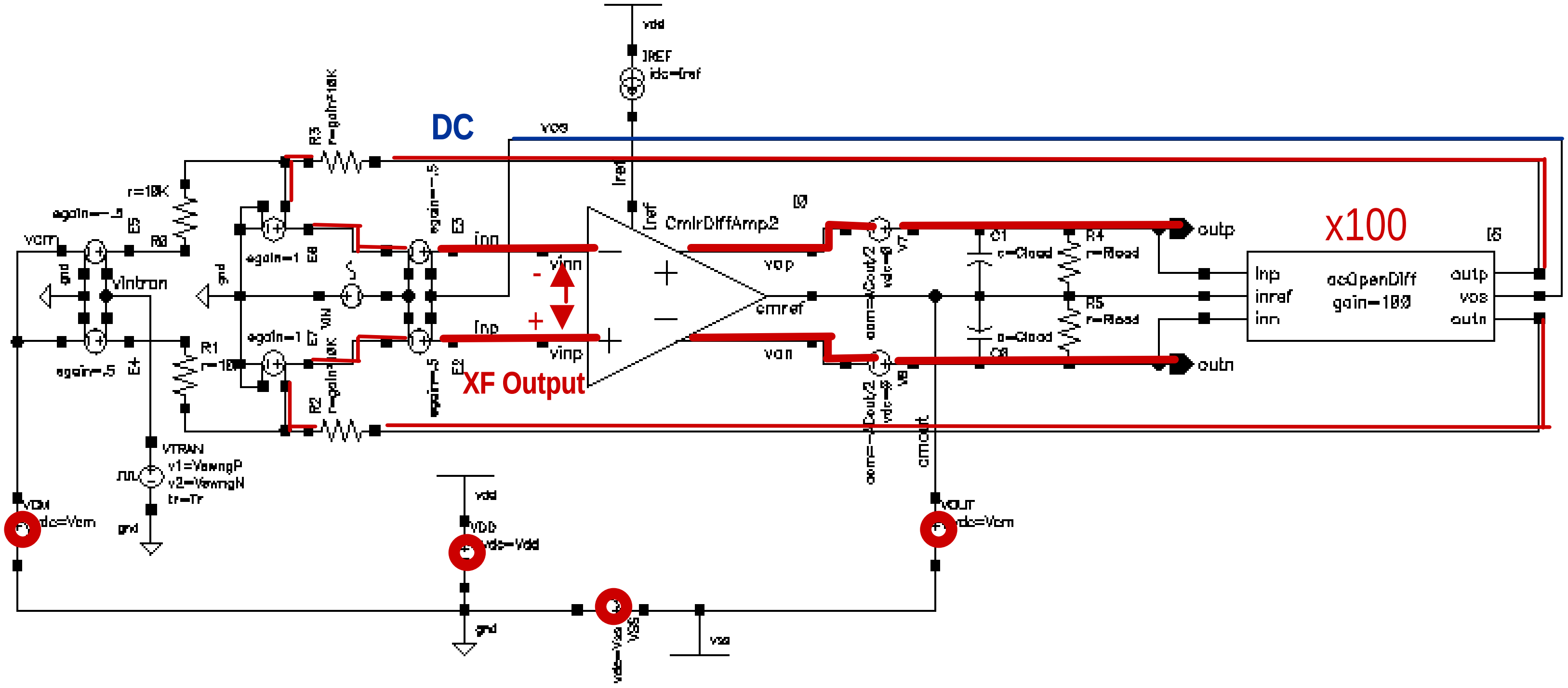
Or use: (VF(“/outp”) – VF(“/outn”))

/ (VF(“/inp”) – VF(“/inn”))

CMRR & PSRR

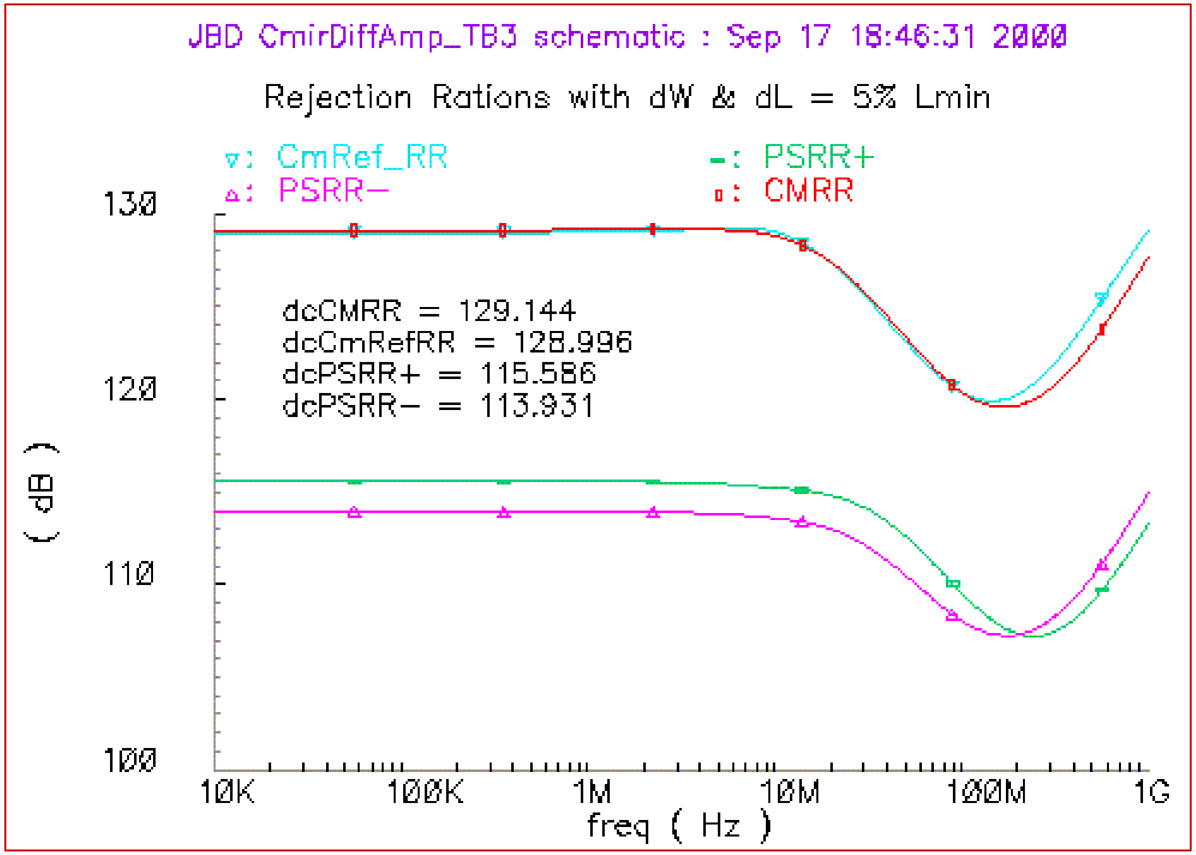
- CMRR = Ad/Ac = $\dfrac{\partial Vout\ /\ \partial Vin}{\partial Vout\ /\ \partial Vcm} = \dfrac{\partial Vcm}{\partial Vin}$
  - XF analysis gives ∂(output node)/ ∂Vsource
  - Selection inp **<—>** inn for output gives ∂Vin/ ∂Vcm
  - XF("/VCM") = 1/CMRR
- PSRR = Ad/Asource = ∂Vsupply/ ∂Vin
  - PSRR+ = dB20(1/ getData("/VDD" ?results "xf-xf"))
  - OCEAN expression works in Calculator & Interactive Sessions

XF Simulation Signal Path



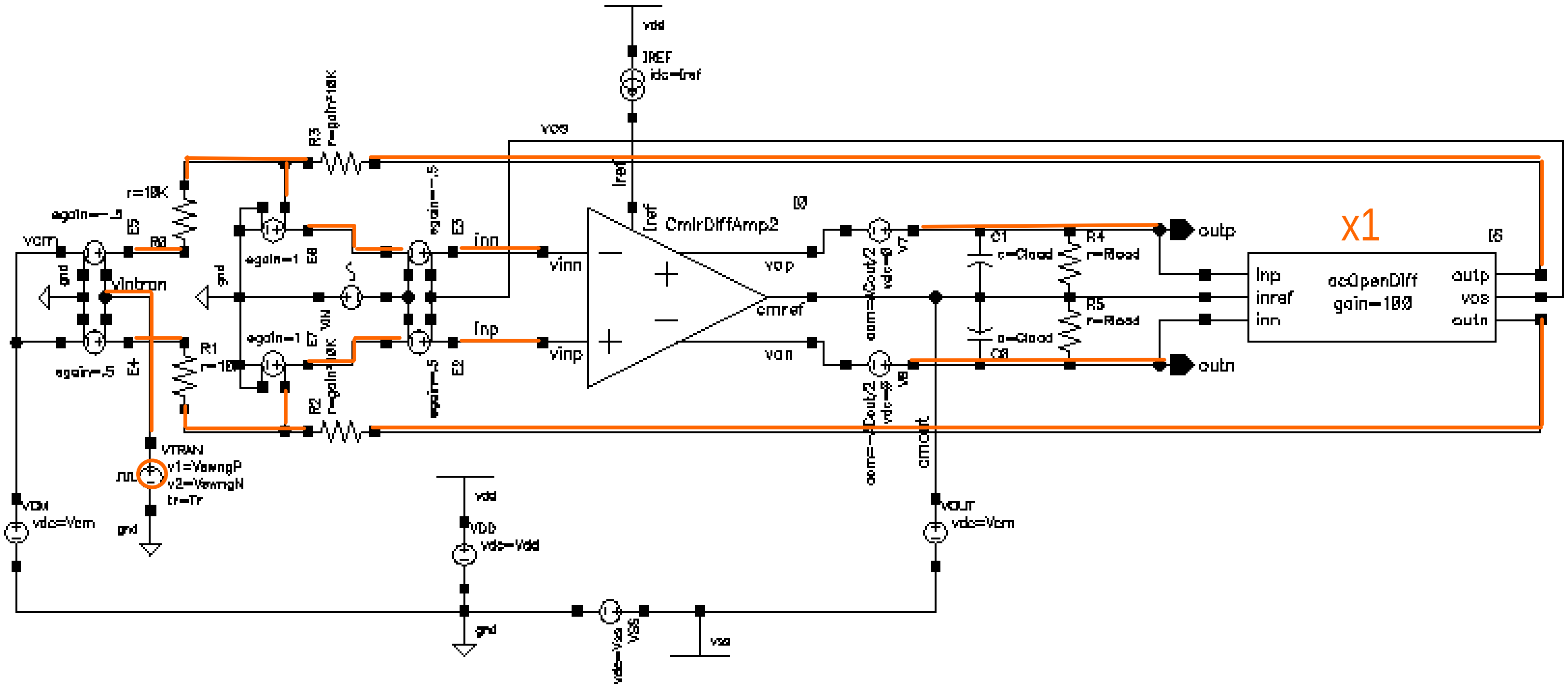
Rejection Ratios

- Use value(<expression> 0) to get “DC” value.



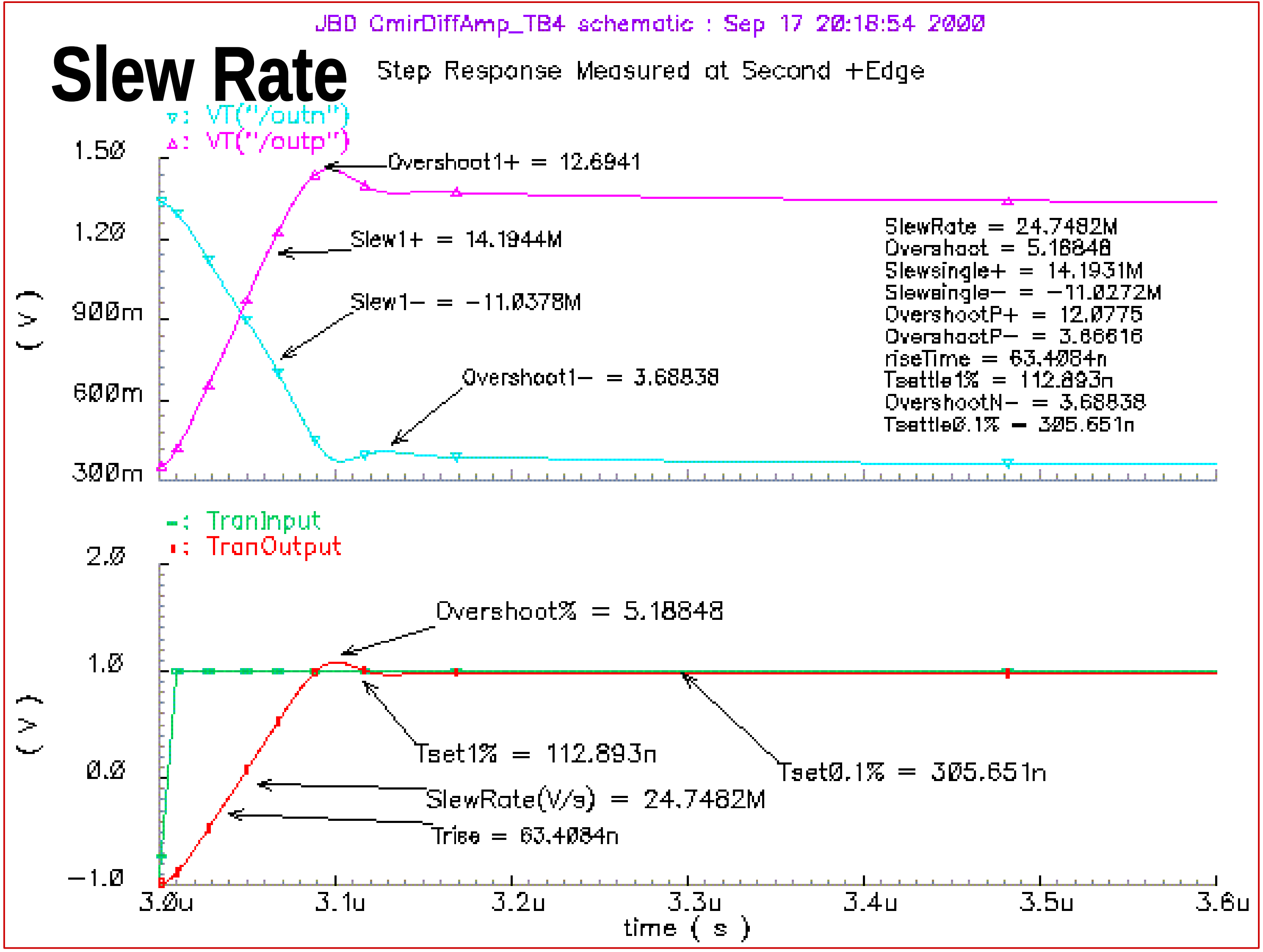
dcCMRR = 129.144
dcCmRefRR = 128.996
dcPSRR+ = 115.586
dcPSRR− = 113.931

Transient Signal Path



TB Use

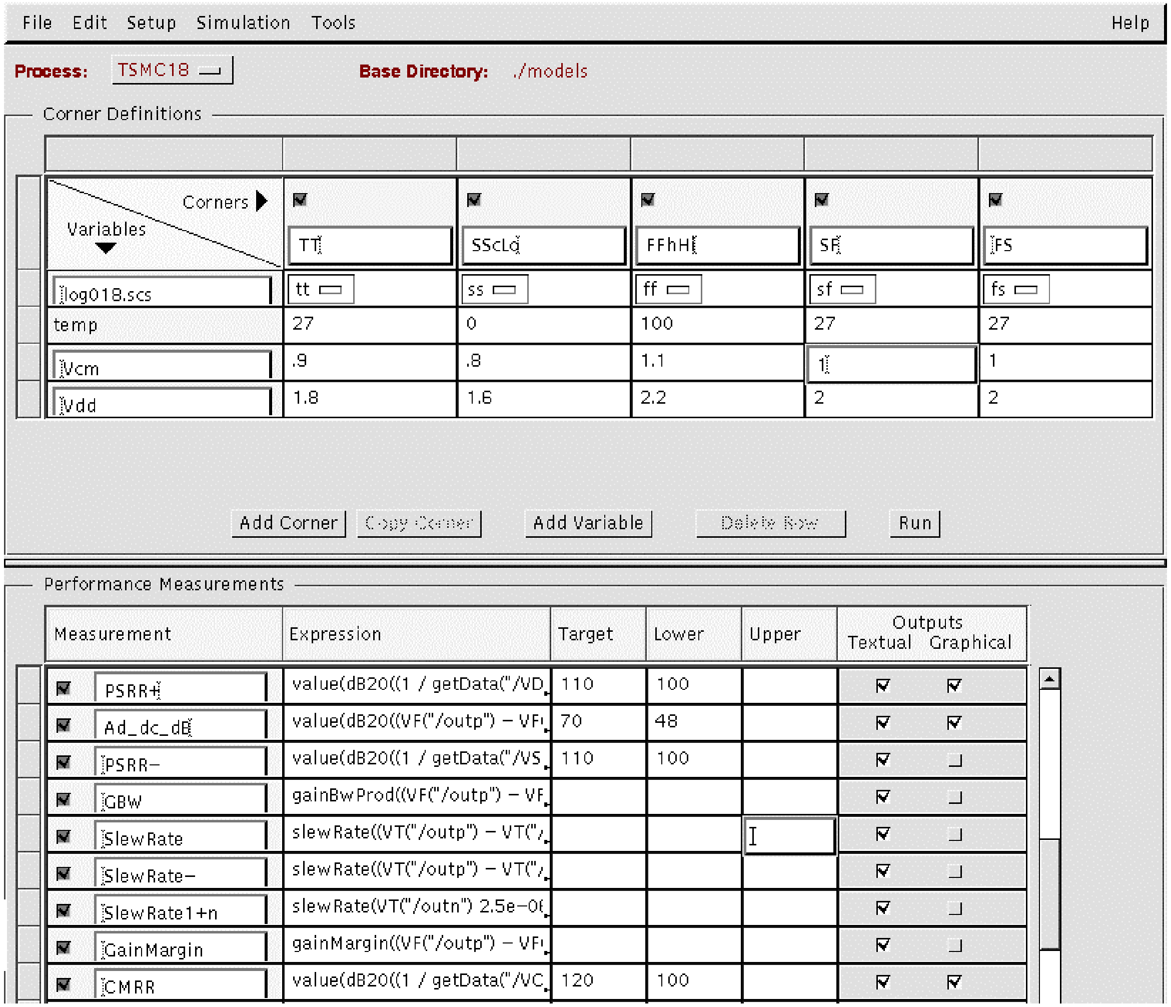
Slew Rate


JBD CmirDiffAmp_TB4 schematic : Sep 17 20:18:54 2000
Step Response Measured at Second +Edge
▿: VT("/outn")
▵: VT("/outp")
1.50
1.20
900m
600m
300m
(V)
Overshoot1+ = 12.6941
Slew1+ = 14.1944M
Slew1− = −11.0378M
Overshoot1− = 3.68838
SlewRate = 24.7482M
Overshoot = 5.18848
Slewsingle+ = 14.1931M
Slewsingle− = −11.0272M
OvershootP+ = 12.0775
OvershootP− = 3.66616
riseTime = 63.4084n
Tsettle1% = 112.893n
OvershootN− = 3.68838
Tsettle0.1% = 305.651n
-: TranInput
▪: TranOutput
2.0
1.0
0.0
−1.0
(V)
Overshoot% = 5.18848
Tset1% = 112.893n
Tset0.1% = 305.651n
SlewRate(V/s) = 24.7482M
Trise = 63.4084n
3.0u
3.1u
3.2u
3.3u
3.4u
3.5u
3.6u
time (s)

Interactive “single state” measurements

A_{OL} ϕ_{OL} vs. Freq. A_d A_c	GB	Gain & Phase Margin
V_{OS} vs Temp V_{IO} Drift	V_{OS} vs Proc/Mmatch	~~I_B I_{OS}~~ ~~I_{IO} Drift~~
Zin Zout	Slew Rate F_{MAX}	Overshoot Settling Time
CMR Common Mode Range	V_{OMax} I_{SC} Output Compliance	I_{Supply} vs PVT Min V_{supply}
ENV f_{BV}	ENI f_{BI}	CMRR & PSRR
THD	**SFDR**	**IP3**

Corners Analysis Gets PVT Results



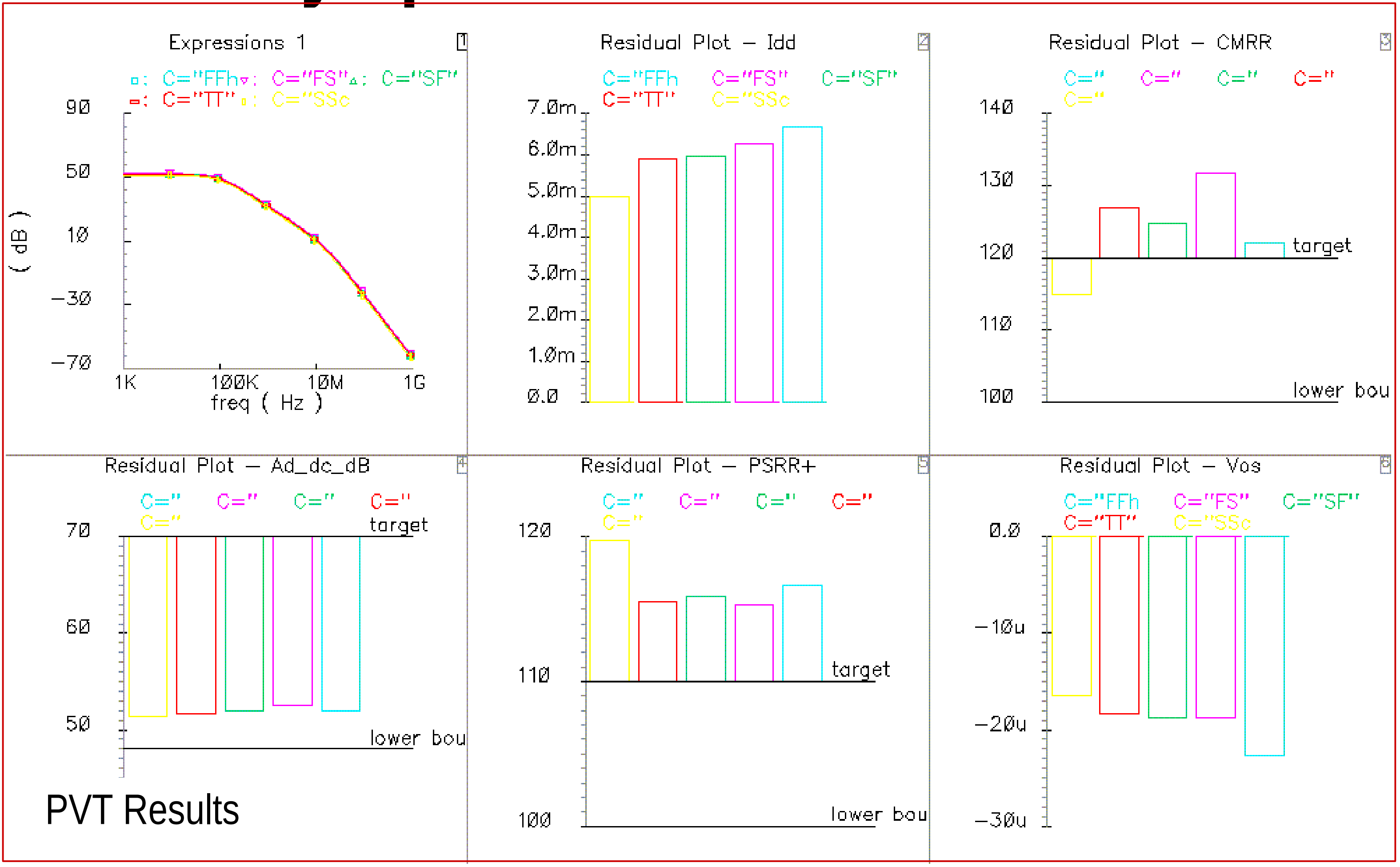
Process: TSMC18

Base Directory: ./models

Variables / Corners	☑ TT	☑ SScLq	☑ FFhH	☑ SF	☑ FS
log018.scs	tt	ss	ff	sf	fs
temp	27	0	100	27	27
Vcm	.9	.8	1.1	1	1
Vdd	1.8	1.6	2.2	2	2

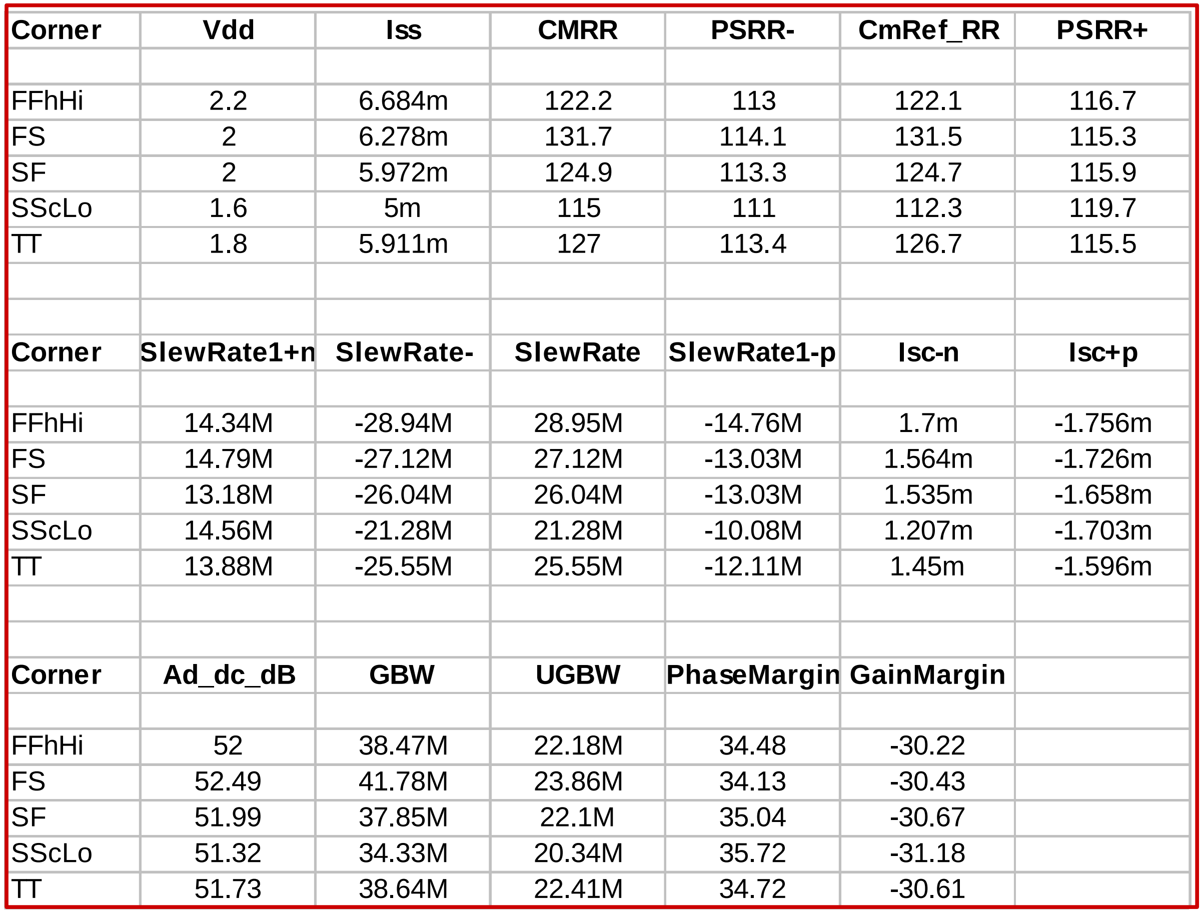
	Measurement	Expression	Target	Lower	Upper	Outputs Textual	Outputs Graphical
☑	PSRR+	value(dB20((1 / getData("/VD	110	100		☑	☑
☑	Ad_dc_dB	value(dB20((VF("/outp") – VF	70	48		☑	☑
☑	PSRR–	value(dB20((1 / getData("/VS	110	100		☑	☐
☑	GBW	gainBwProd((VF("/outp") – VF				☑	☐
☑	SlewRate	slewRate((VT("/outp") – VT("/				☑	☐
☑	SlewRate–	slewRate((VT("/outp") – VT("/				☑	☐
☑	SlewRate1+n	slewRate(VT("/outn") 2.5e–0				☑	☐
☑	GainMargin	gainMargin((VF("/outp") – VF				☑	☐
☑	CMRR	value(dB20((1 / getData("/VC	120	100		☑	☑

Plot Key Specifications



PVT Results

Results can be Tabulated for Reference



Corner	Vdd	Iss	CMRR	PSRR-	CmRef_RR	PSRR+
FFhHi	2.2	6.684m	122.2	113	122.1	116.7
FS	2	6.278m	131.7	114.1	131.5	115.3
SF	2	5.972m	124.9	113.3	124.7	115.9
SScLo	1.6	5m	115	111	112.3	119.7
TT	1.8	5.911m	127	113.4	126.7	115.5
Corner	**SlewRate1+n**	**SlewRate-**	**SlewRate**	**SlewRate1-p**	**Isc-n**	**Isc+p**
FFhHi	14.34M	-28.94M	28.95M	-14.76M	1.7m	-1.756m
FS	14.79M	-27.12M	27.12M	-13.03M	1.564m	-1.726m
SF	13.18M	-26.04M	26.04M	-13.03M	1.535m	-1.658m
SScLo	14.56M	-21.28M	21.28M	-10.08M	1.207m	-1.703m
TT	13.88M	-25.55M	25.55M	-12.11M	1.45m	-1.596m
Corner	**Ad_dc_dB**	**GBW**	**UGBW**	**PhaseMargin**	**GainMargin**	
FFhHi	52	38.47M	22.18M	34.48	-30.22	
FS	52.49	41.78M	23.86M	34.13	-30.43	
SF	51.99	37.85M	22.1M	35.04	-30.67	
SScLo	51.32	34.33M	20.34M	35.72	-31.18	
TT	51.73	38.64M	22.41M	34.72	-30.61	

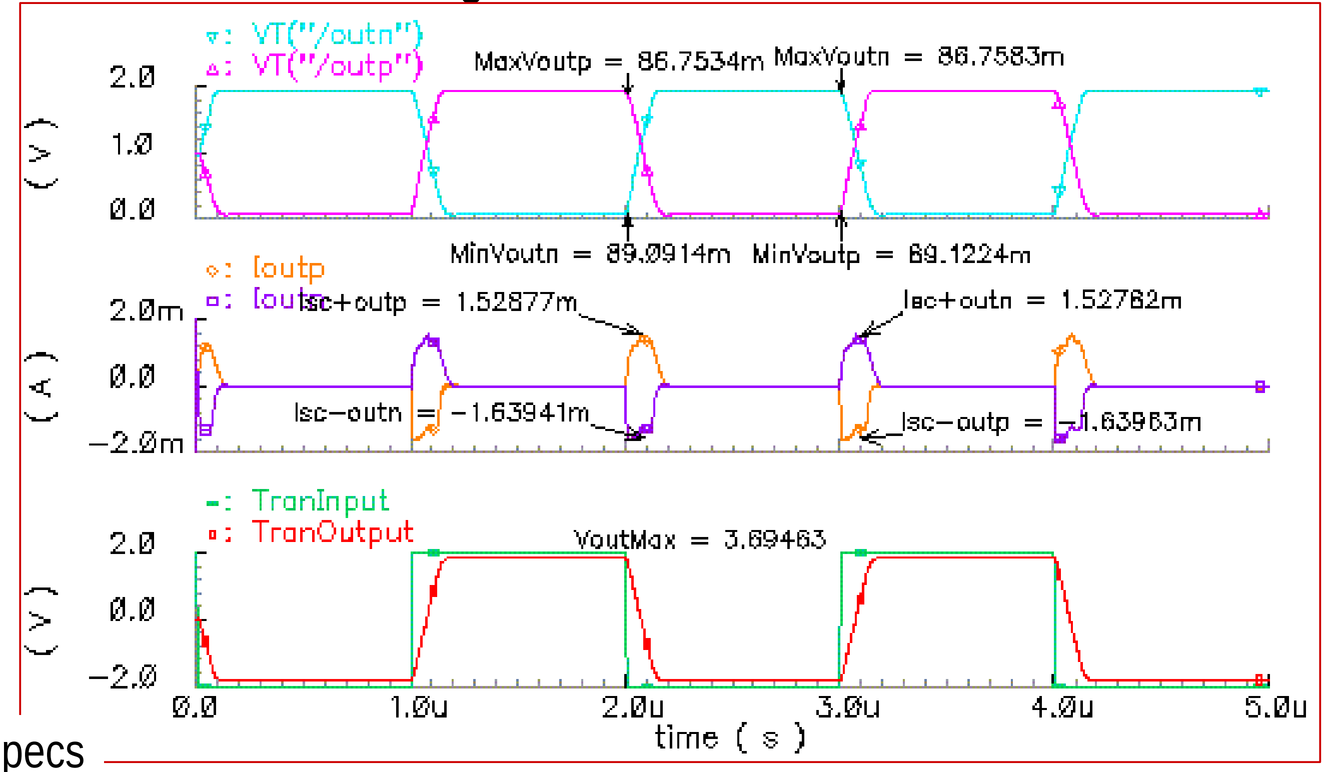
Text file imported to Excel

Other Single Simulation measurements

A_{OL} ϕ_{OL} vs. Freq. A_d A_c	GB	Gain & Phase Margin
V_{OS} vs Temp V_{IO} Drift	V_{OS} vs Proc/Mmatch	~~I_B I_{OS}~~ ~~I_{IO} Drift~~
Zin Zout	Slew Rate F_{MAX}	Overshoot Settling Time
CMR Common Mode Range	V_{OMax} I_{SC} Output Compliance	I_{Supply} vs PVT Min V_{supply}
ENV f_{BV}	ENI f_{BI}	CMRR & PSRR
THD	**SFDR**	**IP3**

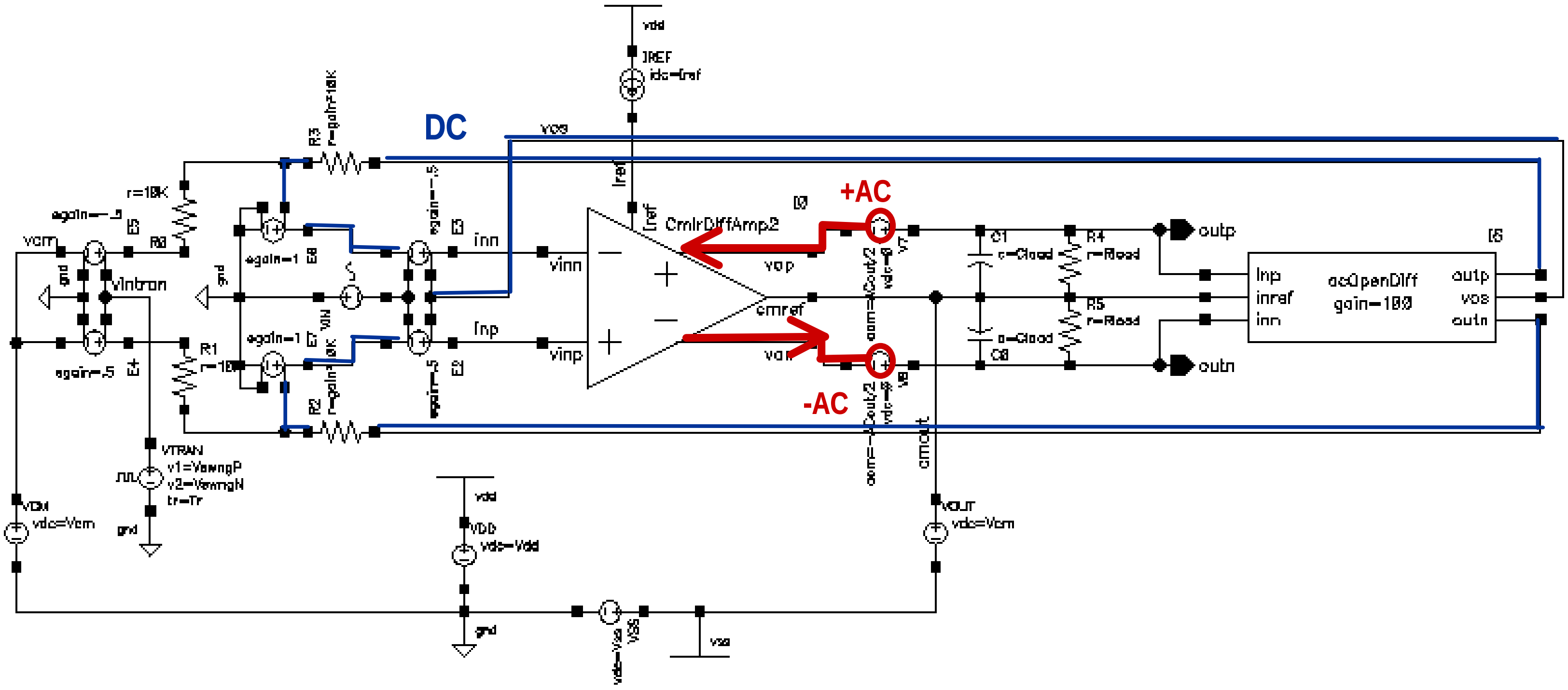
Output Compliance uses overdriven TRAN

- Isc Can’t be measured Directly
  - Use I Cload during transition.

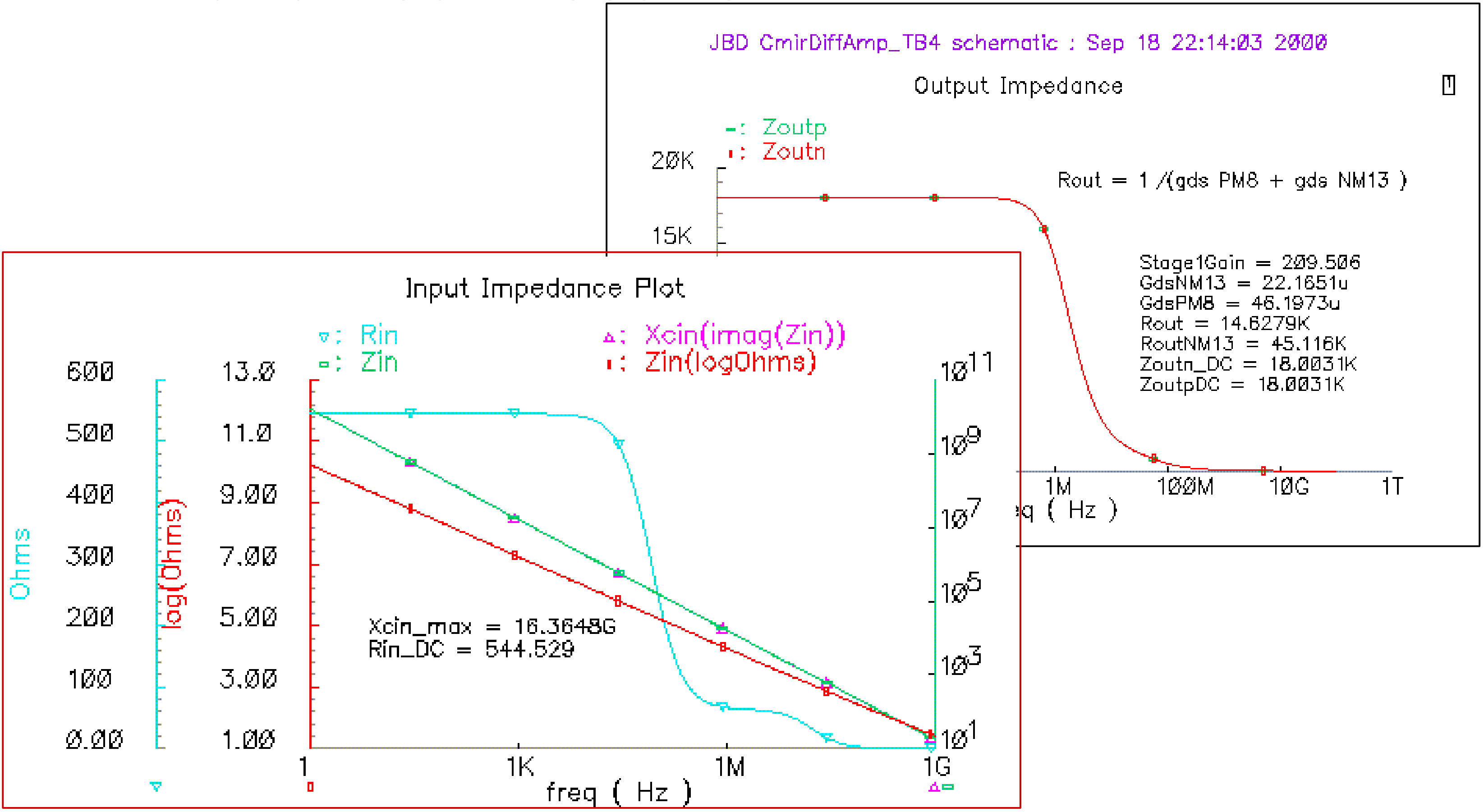


Zout setup

Zin Measured During AC Open Loop



Zin and Zout Results



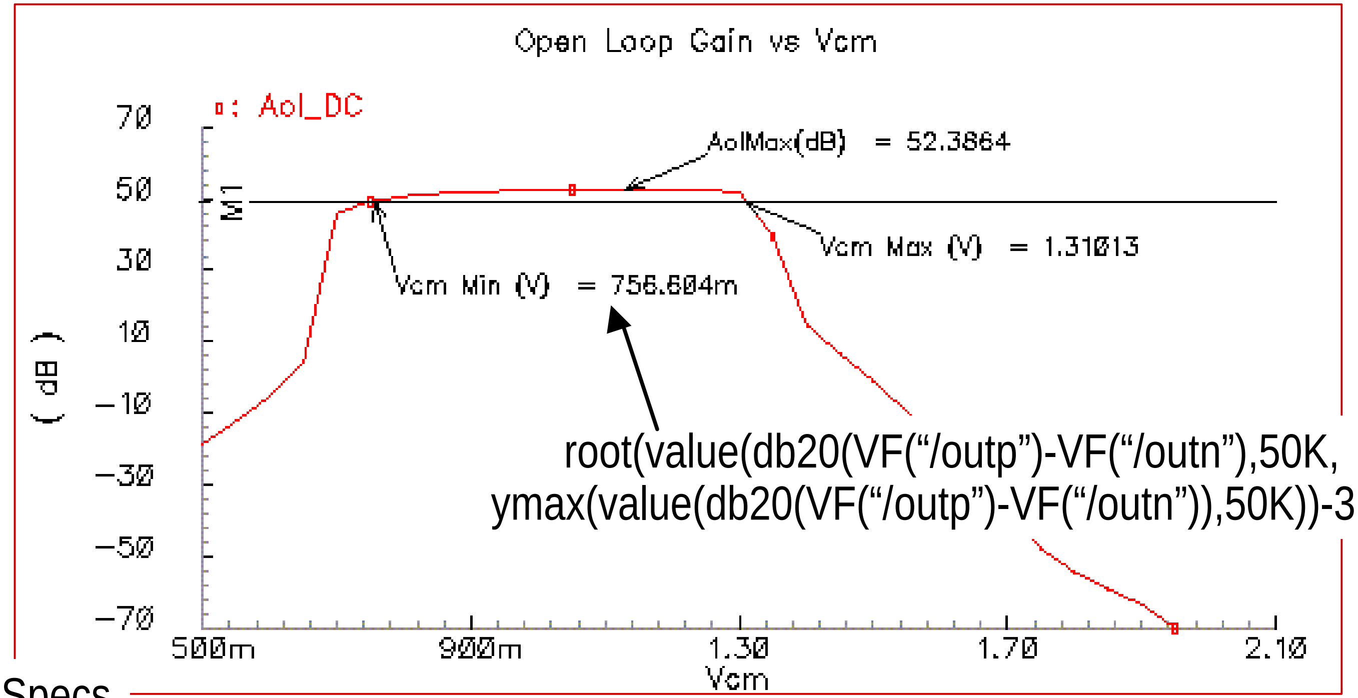
```
Rout = 1 /(gds PM8 + gds NM13 )
```

```
Stage1Gain = 209.506
GdsNM13 = 22.1651u
GdsPM8 = 46.1973u
Rout = 14.6279K
RoutNM13 = 45.116K
Zoutn_DC = 18.0031K
ZoutpDC = 18.0031K
```

```
Xcin_max = 16.3648G
Rin_DC = 544.529
```

Common Mode Range

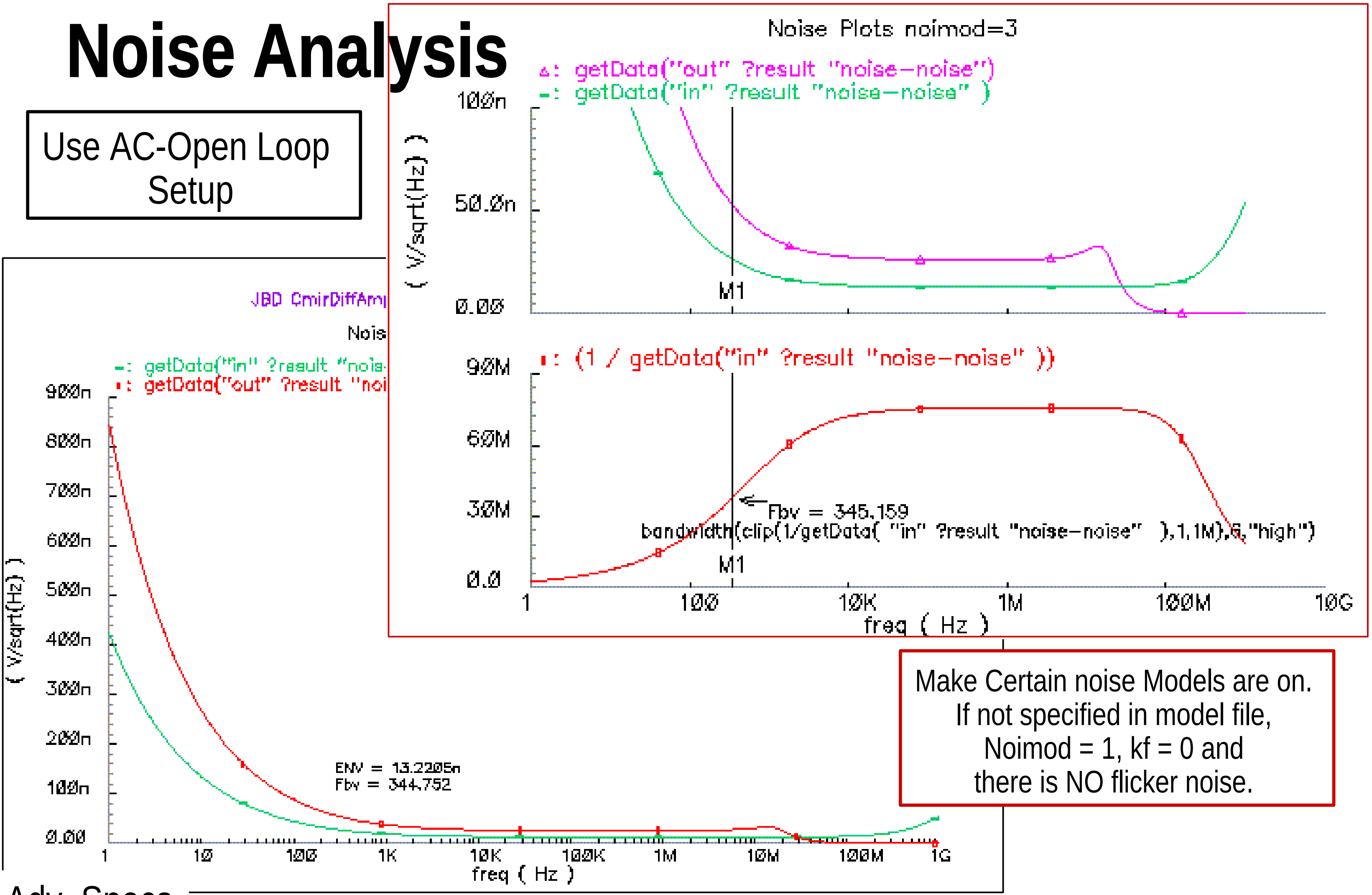
- Parametric DC + AC analysis, vary Vcm from 0 to 2
- Find Vcm min & max where gain is ½ of Max (-3db)



root(value(db20(VF("/outp")-VF("/outn"),50K,
ymax(value(db20(VF("/outp")-VF("/outn")),50K))-3 , 1)

Noise Analysis

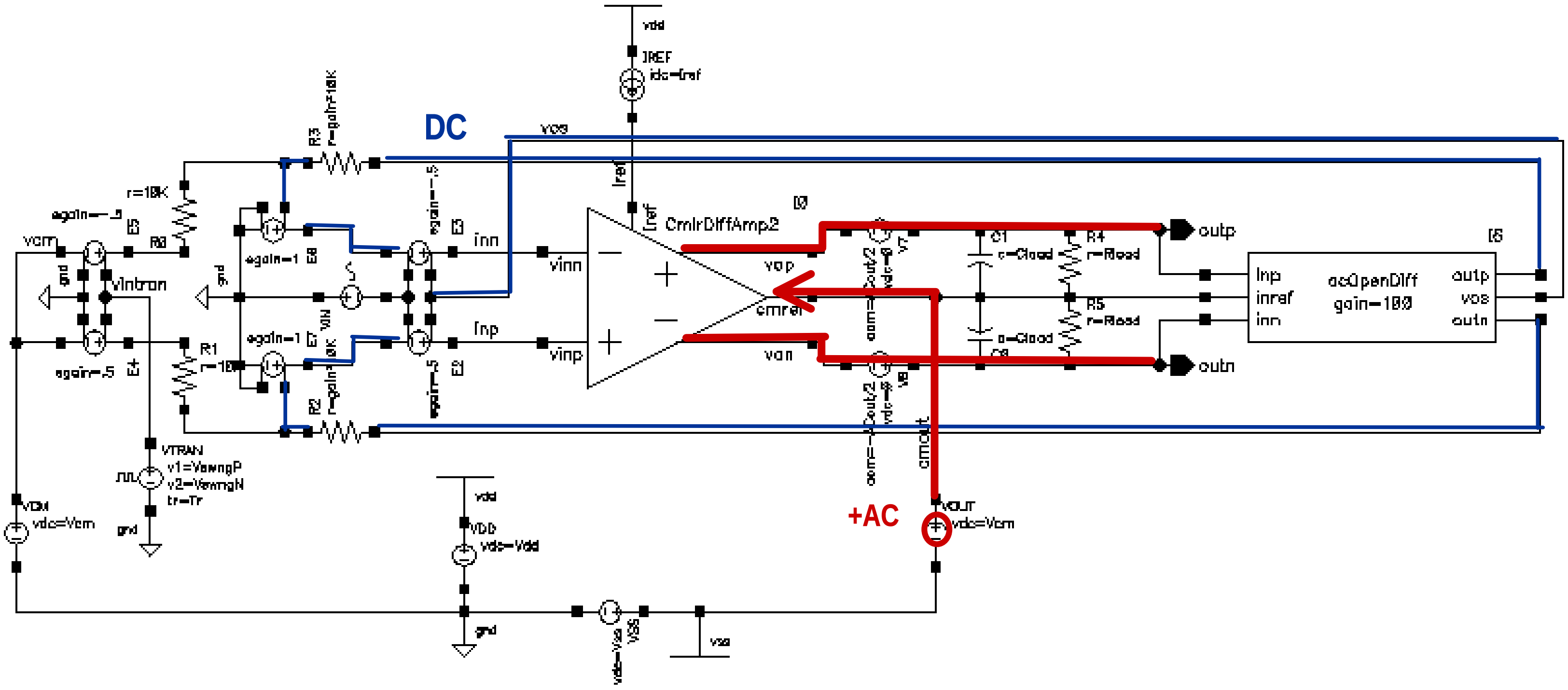
Use AC-Open Loop Setup



Make Certain noise Models are on.
If not specified in model file,
Noimod = 1, kf = 0 and
there is NO flicker noise.

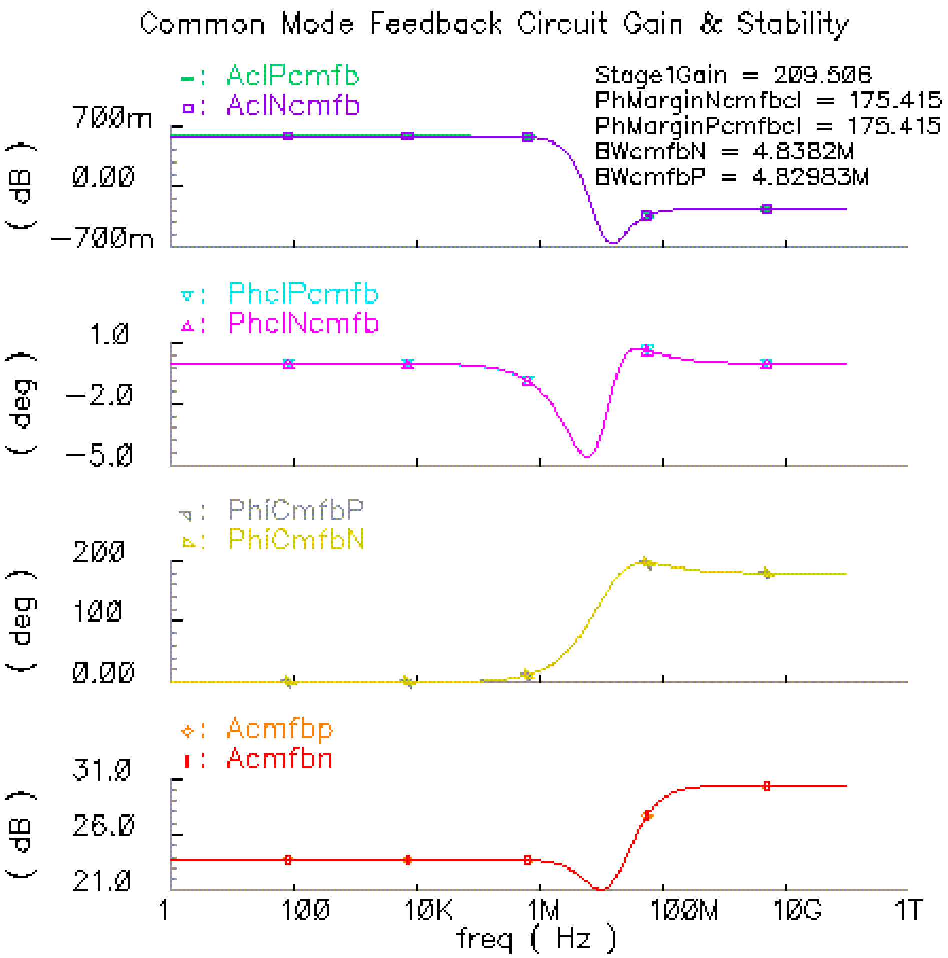
CMFB Loop setup

Treat as 2 Unity Gain Connected Amps



CMFB Stability Results

- These results point to an area of possible improvement.
- BandWidth is for 0.5 dB reduction in signal.



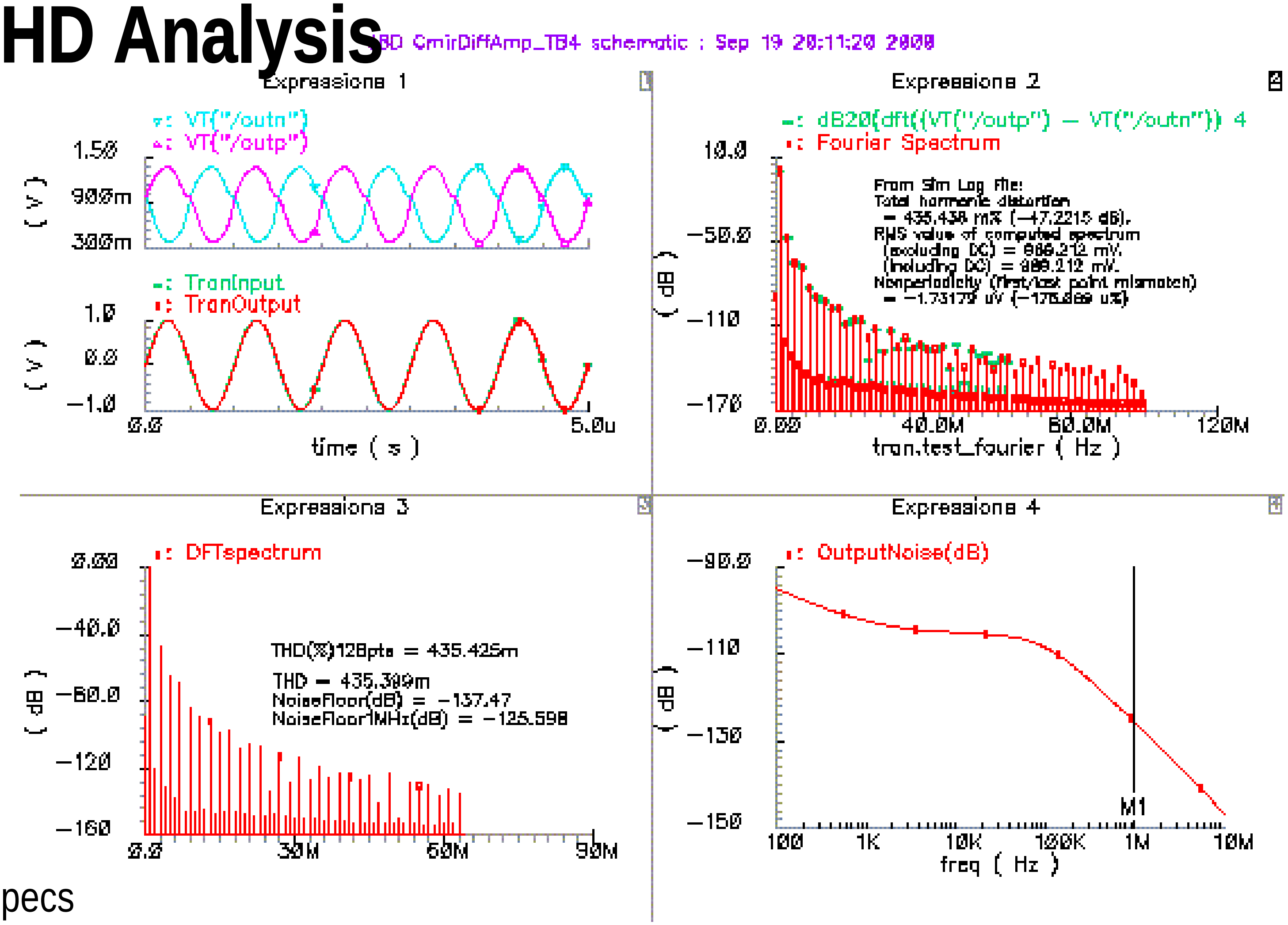
Common Mode Feedback Circuit Gain & Stability

Stage1Gain = 209.506
PhMarginNcmfbcl = 175.415
PhMarginPcmfbcl = 175.415
BWcmfbN = 4.8382M
BWcmfbP = 4.82983M

2 ways to get Spectrum & THD

- Input a sinusoid, do Fourier Transform of the Output.
- Calculator: use DFT & THD functions.
  - Uses 2^N data-points, interpolated for even spacing.
  - Set simulation options to ensure enough timesteps
- Spectre: use internal Integral Based Fourier analysis.
  - Place" fourier2ch" element in the Testbench.
  - Spectrum in results DB, THD listed in Log.
  - No interpolation used
  - Timesteps are controlled for you.

THD Analysis

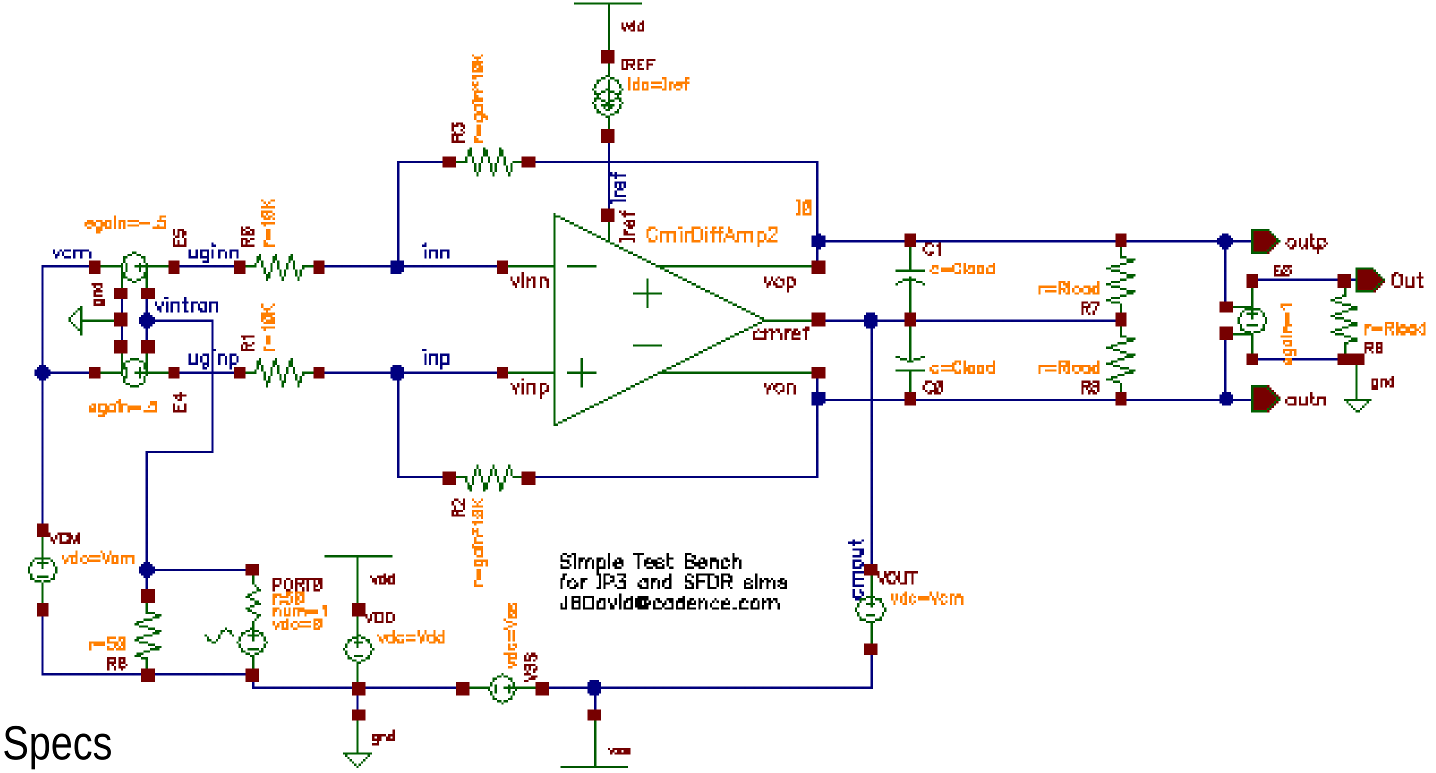


```
From Sim Log File:
Total harmonic distortion
 = 435.438 m% (-47.2215 dB).
RMS value of computed spectrum
 (excluding DC) = 888.212 mV.
 (including DC) = 888.212 mV.
Nonperiodicity (first/last point mismatch)
 = -1.73179 uV (-176.888 u%)
```

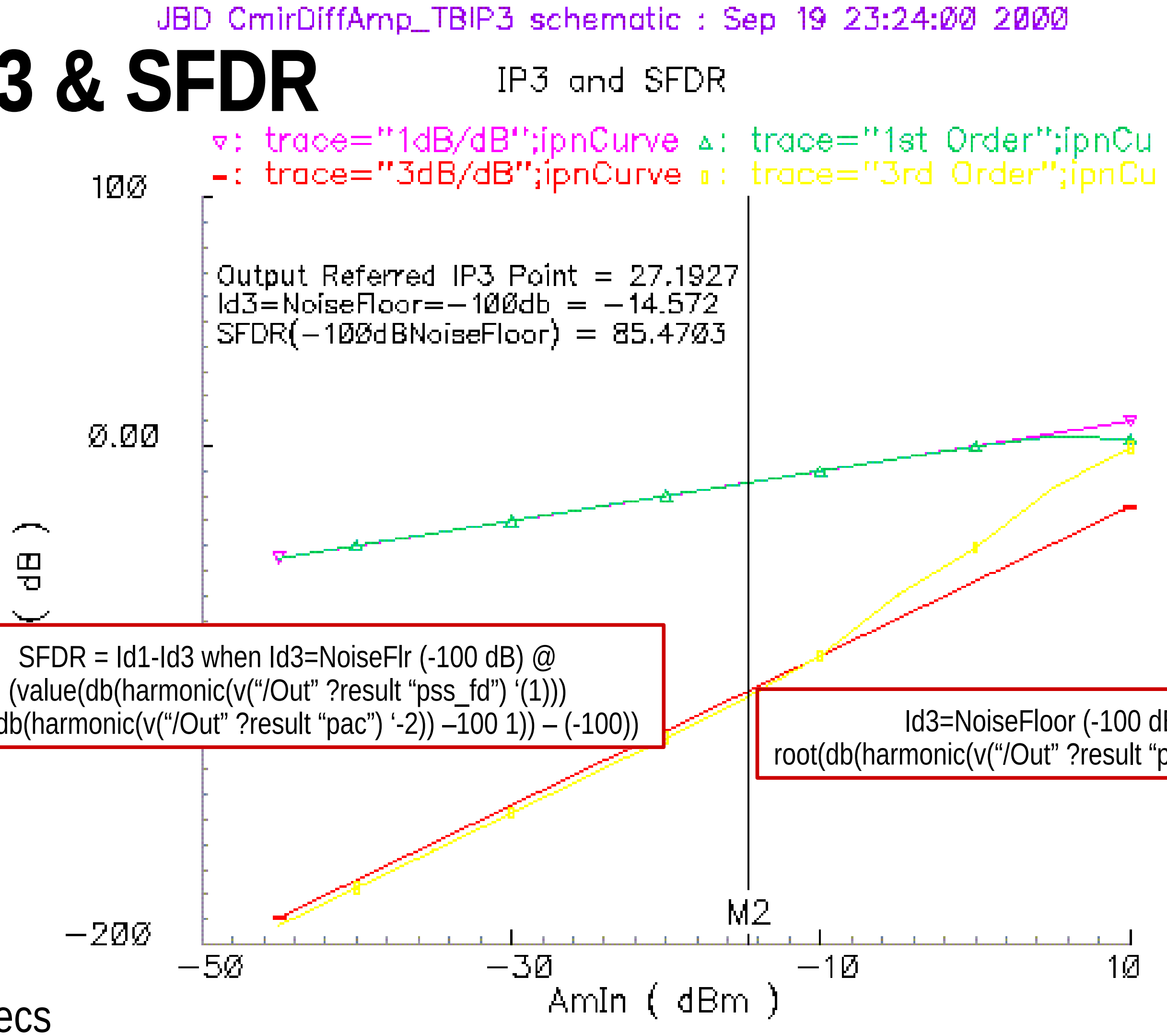
```
THD(%)128pts = 435.425m

THD = 435.309m
NoiseFloor(dB) = -137.47
NoiseFloor1MHz(dB) = -125.598
```

IP3 & SFDR testbench

- spectreRF is the easiest way to get the results.
- Uses variation on the simple testbench



IP3 & SFDR

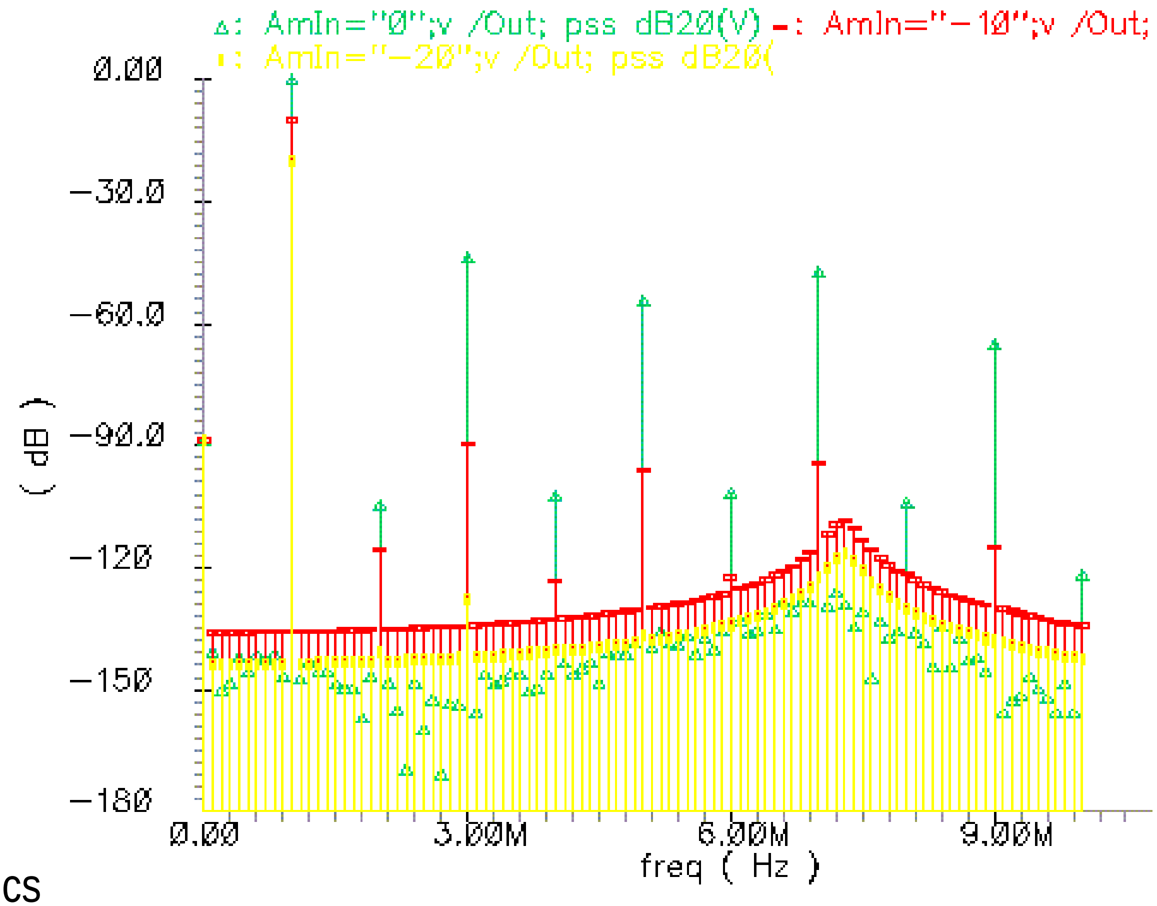


```
Output Referred IP3 Point = 27.1927
Id3=NoiseFloor=-100db = -14.572
SFDR(-100dBNoiseFloor) = 85.4703
```

SFDR = Id1-Id3 when Id3=NoiseFlr (-100 dB) @
(value(db(harmonic(v(“/Out” ?result “pss_fd”) ‘(1)))
root(db(harmonic(v(“/Out” ?result “pac”) ‘-2)) –100 1)) – (-100))

Id3=NoiseFloor (-100 dB) @
root(db(harmonic(v(“/Out” ?result “pac”) ‘-2)) –100 1)

Distortion is Causing Excess 3rd Product



Offset Voltage

- the Vos measurement is simple.
- Assuring that it represents something useful is Not.
- Vos for Balanced Diff Amps ideally is 0.
- Monte Carlo analysis in Spectre®
    - does both Process and Mismatch parameter randomization
    - No tie to Layout, or identification of “pairs”
    - Requires fancy Model work to make “useful”
    - Matched Pair Pcell & inline Circuits would help
- Corners Analysis with special models could be easier.
- Parametric Analysis can work with variables for vt and w & L shifts.

Investment in TB pays off with Scripts

- Almost Any Interactive Setup has Script Equivalent
- Each Tool can create a script from its current state.
  - duplicates setup, simulations and measurements
  - Artist, Parameter Analyzer, CornersTool, Monte Carlo, Optimizer
  - Plots, and Printouts can be created in a file.
- For 1 Cell this aids in re-verifying performance after ECO and with Layout Parasitics.
- Also Enables soft IP re-use. Cell can be re-optimized to different specs, or process.
- Script can Generate Output file in HTML with links to PDF or GIF versions of plots.

Self Verifying Scripts.

- Widely used in Digital Top Down Design Flows
- Measurements can be made in Ocean
- Transient Measurements can be made in
  - Verilog-A - even in Analog only flow.
  - Verilog - when using Mixed Signal
  - VHDL code can be reused from top level in AMS Designer
  - Verilog-AMS code can also be used to make measurements more efficient.
  - OCEAN calculator expressions AFTER the simulation is complete.

Script Coding for Regression Testing

- Include specification limit checks in Test Harness or OCEAN code.
- On every Spec Failure $strobe / print() a message to the logfile
  - Include a consistent keyword ie “SPECFAIL:”
  - Also list the Spec Limit and Measured result.
- This is in addition to generating Plots / Tables.
- Tables of limits data can be “included” in Verilog-AMS modules
- OCEAN can Load a separate file to set parameter limits variables.

Simulation Wrappers

- Create a Perl script to
  - Spawn the simulations via Ocean.
  - scan Logfiles for keywords including Simulator Warnings and Errors,
  - Count each type and create a report.
  - Build data sheet if Passed & delete sim data.
  - Notify via email if failures and SAVE sim data.

If Pass; post process Results

- Data Sheet info
- Text -> HTML?
- image conversion.
- Final Doc Assembly (Word ? HTML or PDF?)
- Write Results log to Results area

Automating Regression Testing:

- Build database of Cells - Results files and sim. wrapper scripts.
- Periodically check date of design data vs results file.
  - Re-run script if results out of date.
- Run All scripts periodically - at project Gates
- Build dependancy tables
  - more that one simulation may be affected.
- Use of Design Management (DesignSync) may add capability to schedule re-simulation after checking.

Review of the simulation Methodology:

- Plan your simulations around block specification
- Build Test schematic
  - Use Variables for Ocean scripting
  - Use Verilog-A where needed
  - Add Verilog & Verilog-AMS when needed for Mixed Signal testing.
- Use Interactive Sims to create basic Ocean script,
  - Customize plots for your specs.
  - Add Limits checking code.
- Scan Logs for Failure reports with perl.

Amp & Analog Comments

- Opamps were created to model Dynamic systems.

analogue n. Also analog. 1. Something that bears an analogy to something else.

American Heritage Dictionary of the English Language (1980) Houghton Mifflin Co, Boston

“I feel that the engineer who feels he or she has mastered an understanding of all significant variables in a particular design has, almost certainly, not considered all of the significant variables.”

Derek F. Bowers “Reality Driven Analog Integrated Circuit Design” in

Jim Williams(Ed), “Analog Circuit Design: Art, Science and Personalities”

Bibliography

Don Lewis “Testing Operational Amplifiers” Electronics Test (Benwill Publishing) January 1979

Don Lewis “Compensation of Linear IC Test Loops” Electronics Test (Benwill Publishing) May 1979

David Johns, Ken Martin,”Analog Integrated Circuit Design”, Wiley & Sons, New York 1997, esp chapter 6.

Jacob Millman, “MicroElectronics: Digital and Analog Circuits and Systems” McGraw-Hill 1979

Paul Gray and Robert Meyer “Analysis and Design of Analog Integrated Circuits: 2nd Ed” Wiley & Sons 1984

C.F. Wojslaw & E.A. Moustakas “Operational Amplifiers” Wiley & Sons, NY, 1986

Dan Fitzpatrickj, Ira Miller, “Analog Behavioral Modeling with the Verilog-A Language” Kluwer, Boston, 1998

Samir Palnitkar, “ Verilog HDL” SunSoft, Mountain View, CA 1996

Ken Kundert, “The Designer’s Guide to Spice & Spectre”, Kluwer, Boston, 1995

J.E. Solomon. "The monolithic op amp: a tutorial study." IEEE Journal of Solid-State Circuits(1974) SC-9.6 (Dec. 1974 (Special Issue on Analog Circuits)): 314-332. (Also published as Application Note AN-A from National Semiconductor)

G.Ferri and W. Sansen “A Rail-toRail Constant-g_m Low-Voltage CMOS Operational Transconductance Amplifier.” IEEE Journal of Solid-State Circuits(1997), vol 32, October 1997 1563-1567.

M. Yamatke, “A Simplified Test-Set for Op Amp Characterization” National Semiconductor Application Note 24, April 1986

Bibliography (Cont’d)

K. Bult& G.J.G.M. Geelen, “A Fast-Settling CMOS Op Amp for SC Circuits with 90-dB DC gain. IEEE Journal of Solid-State Circuits(1990), Vol. 25, No. 6, Dec. 1990, 1379-1384

T.C. Choi, R.T. Kaneshiro, R.W. Brodersen, P.R. Gray W.B. Jett & M. Wilcox, “High-Frequency CMOS Switched Capacitor Filters for Communications Application” IEEE Journal of Solid-State Circuits(1983), SC-18 (December 1983) 652-663

M. Banu, J.M. Khoury & Y. Tsividis “Fully Differential Operational Amplifiers with Accurate Output Balancing.” IEEE Journal of Solid-State Circuits(1988), Vol. 23, No. 6, Dec. 1988 1410-1414

P.E. Allen and D.R. Holberg, “CMOS Analog Circuit Design” Oxford Univ. Press, New York 1987

K.R. Laker & W.M.C. Sansen, “Design of Analog Integrated Circuits and Systems” McGraw-Hill, New York 1994

Appendix

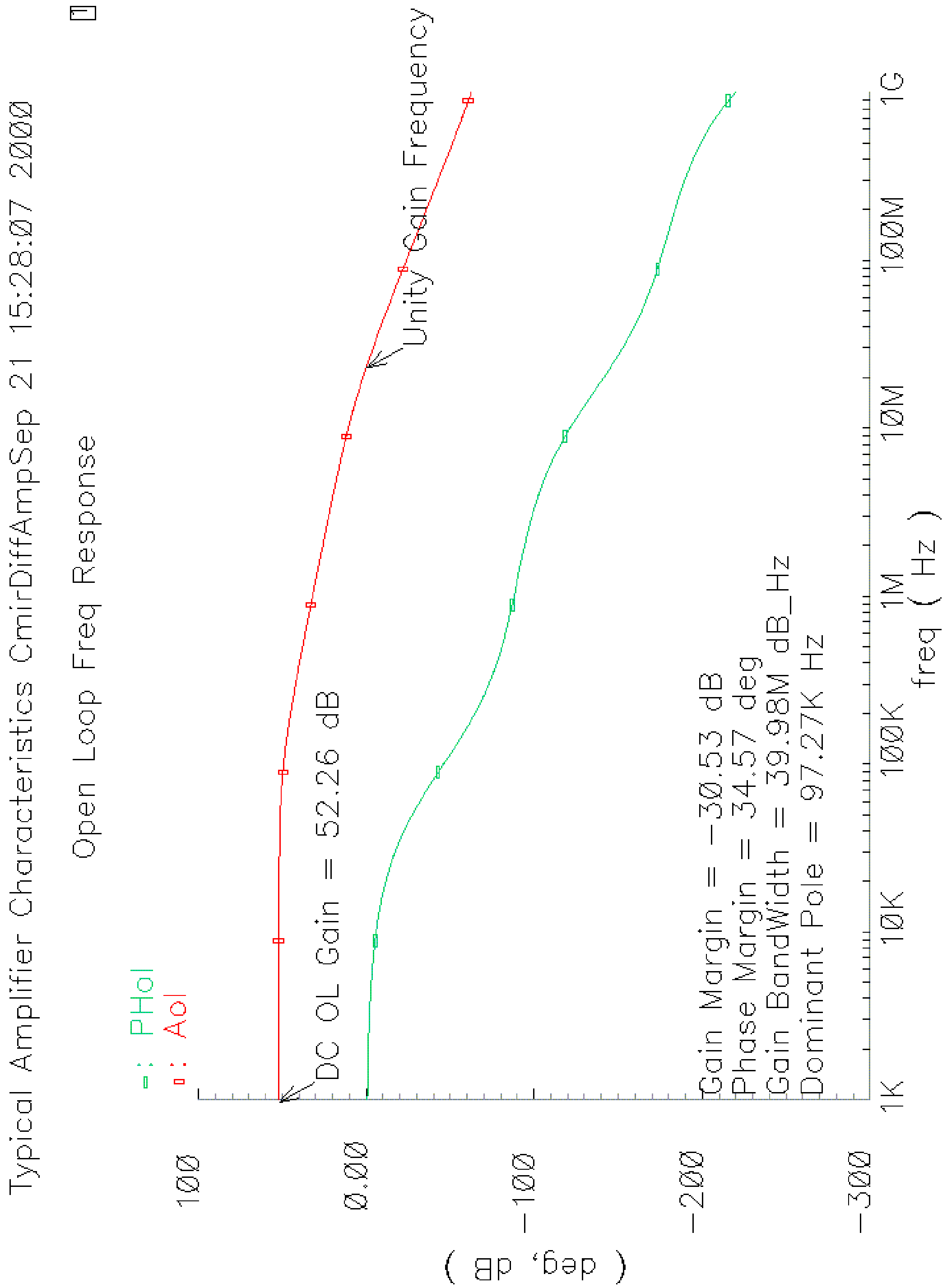
The following pages document the Verilog-A module used to close the loop around the amplifier, and the OCEAN script used to regenerate the Key Data, and a sample of the resulting plots generated using the simulation.

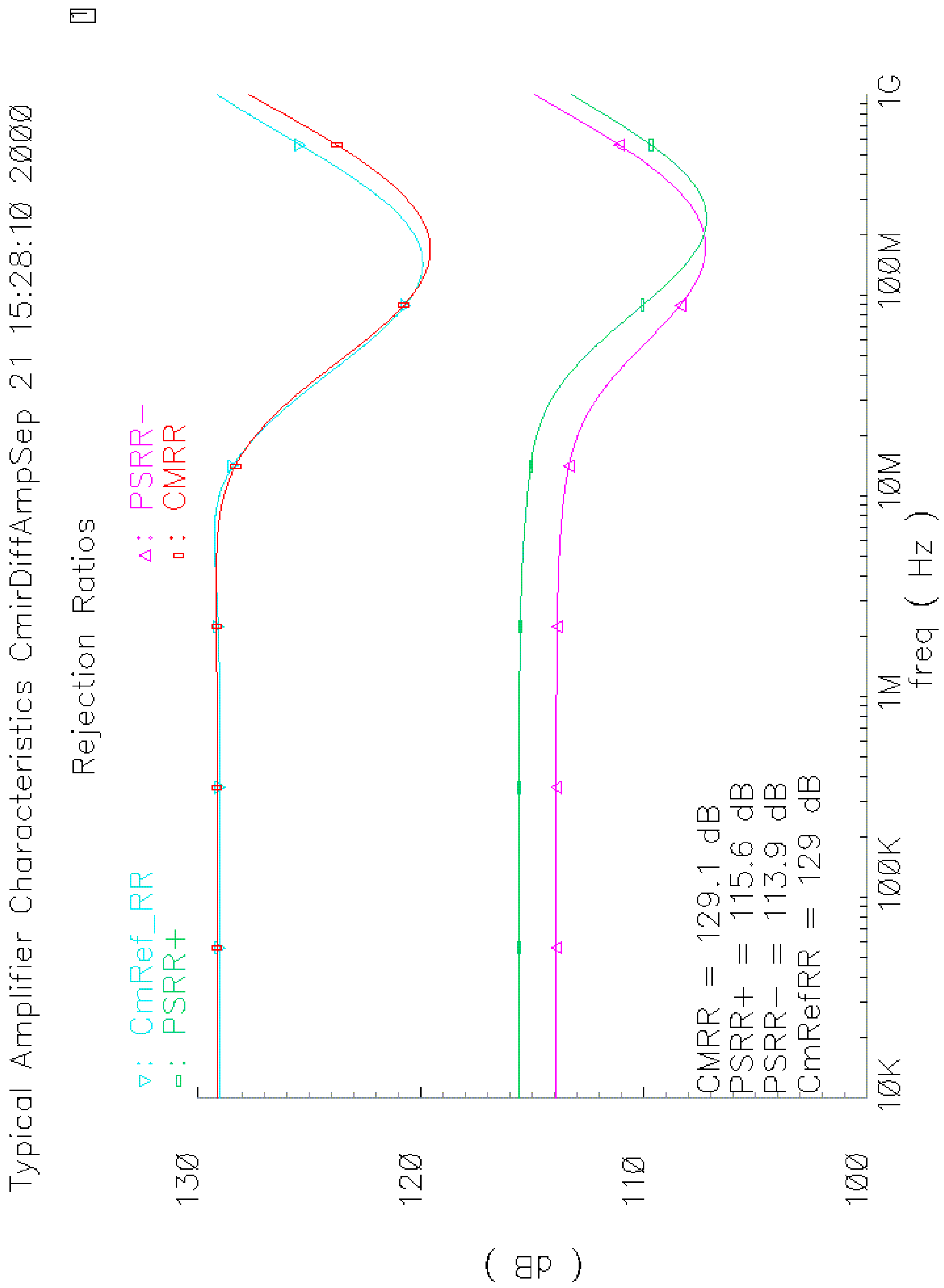
The methods used to control the window size, annotate the data and generate the hardcopy data to postscript file are not often shown in example OCEAN scripts supplied with the software.

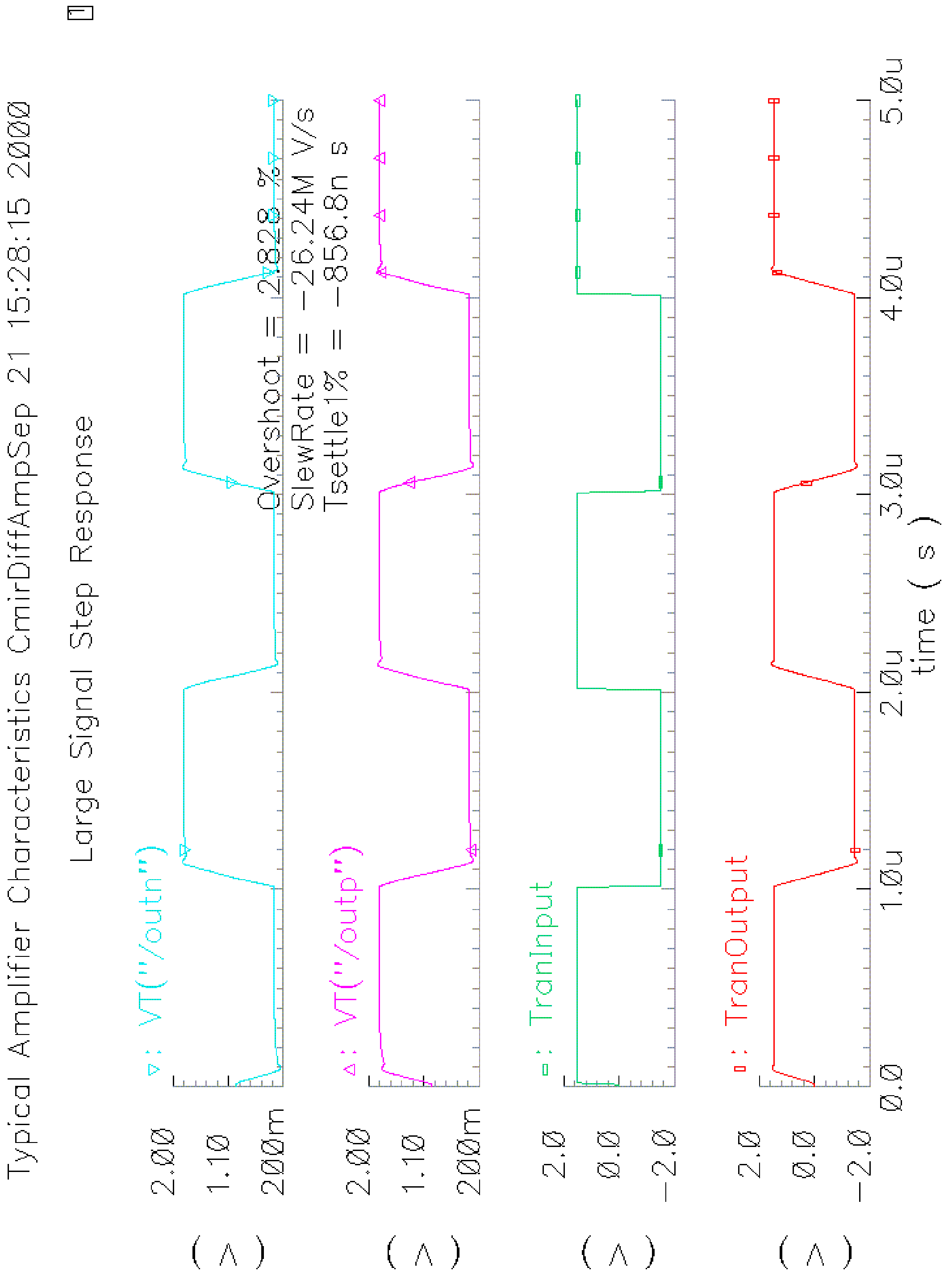
```
// Jonathan David jbdavid@cadence.com for IEEE presentation
//
// This module drives the output based on the input DCOP..
// for small signal analyses.. ac
// Buffers the output for others
// designed to work with differential amplifier in unity gain config..
// but feedback resistors don't have to match input resistors
// average output voltage will go to the feedback resistors
// difference output * gain will go out on vos..
// except during transient when inputs go to outputs


`include "constants.h"
`include "discipline.h"

module acOpenDiff( inp, inn, inref, outp, outn, vos);
   input inp, inn, inref;
   output outp, outn, vos;
   electrical inp, inn, inref, outp, outn, vos;
   parameter real gain = 1; // sets dc gain for closed loop tests.
   real vin, vbias;
   analog begin
      @(initial_step("static")) begin
         vin = gain*V(inp,inn);
       vbias = (V(inp,inref)+V(inn,inref))/2;
      end
      if (analysis("ac","noise")) begin
         V(vos) <+ vin;
         V(outp,inref) <+ vbias;
         V(outn,inref) <+ vbias;
      end
      else if (analysis("static","xf")) begin
         V(vos) <+ gain*V(inp,inn); // the difference goes here
         V(outp,inref) <+ (V(inp,inref)+V(inn,inref))/2; // average
         V(outn,inref) <+ (V(inp,inref)+V(inn,inref))/2; // average
      end
      else begin // transient and large signal analyses
         V(outp,inref) <+ V(inp,inref);
         V(outn,inref) <+ V(inn,inref);
       V(vos) <+ 0;
      end
   end
endmodule
```

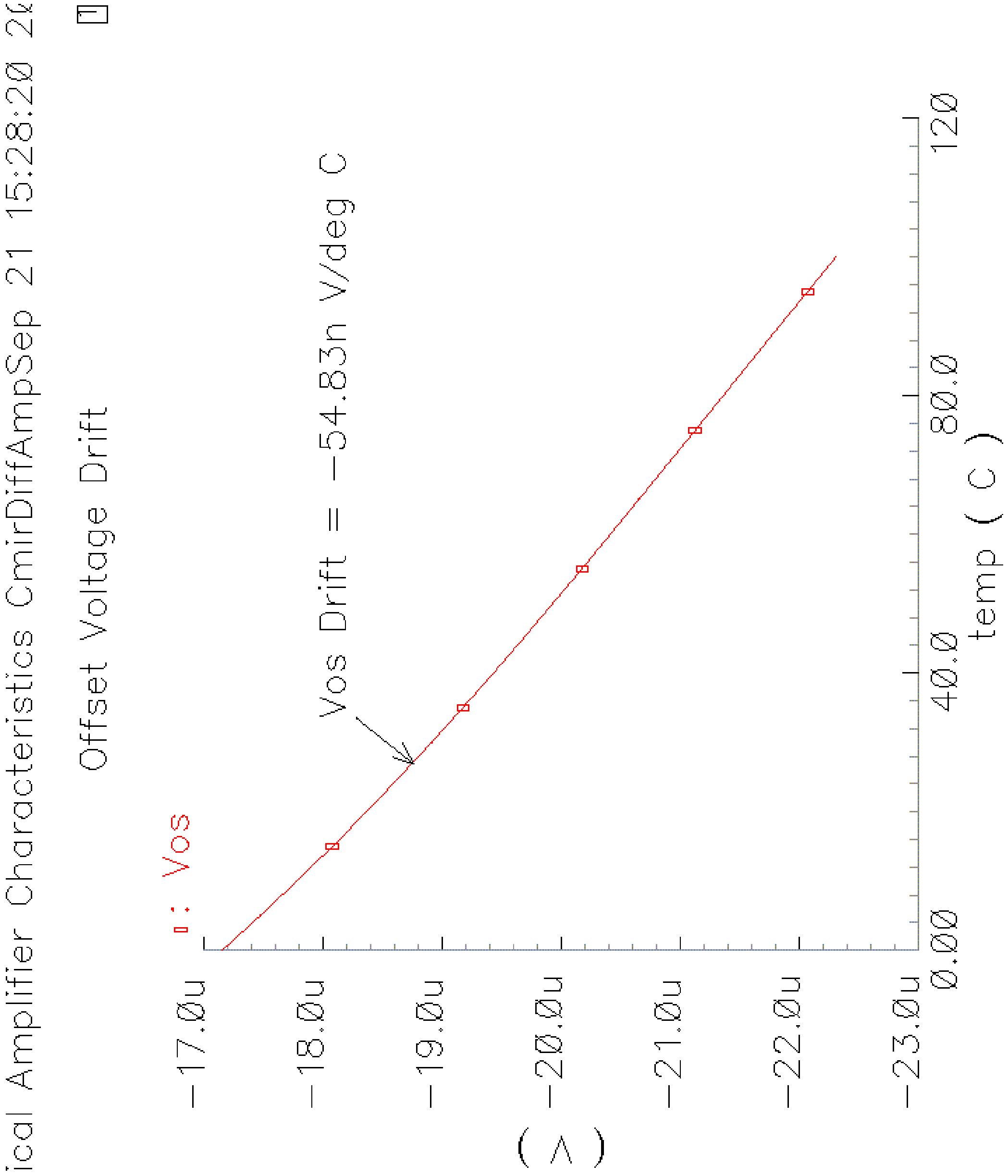
```
simulator( 'spectre )
design(
"/hm/jbdavid/proj445/ieeescv/simulation/CmirDiffAmp_TB4/spectre/schemat
ic/netlist/netlist")
resultsDir(

"/hm/jbdavid/proj445/ieeescv/simulation/CmirDiffAmp_TB4/spectre/keyPerf
typ" )
path( "./models" )
modelFile(
    '("log018.scs" "bip")
    '("log018.scs" "tt_3vna")
    '("log018.scs" "tt_na")
    '("log018.scs" "tt_3v")
    '("ResModel.scs" "res_t")
    '("log018.scs" "res")
    '("log018.scs" "tt")
)
analysis('xf ?start "10K"  ?stop "1G"  ?dec "20"
            ?p "/inp"  ?n "/inn"  )
analysis('dc ?saveOppoint t  ?param "temp"  ?start "0"
            ?stop "100"  ?step "5"  )
analysis('ac ?start "1K"  ?stop "1G"  ?dec "20"  )
analysis('tran ?stop "5u"  )
desVar(       "inCMFB" 0      )
desVar(       "DCgain" 100    )
desVar(       "inDist" 0      )
desVar(       "Fdist" 1M      )
desVar(       "ACin" 1  )
desVar(       "VswngP" 1.5    )
desVar(       "Tpin" 2u )
desVar(       "Tdin" 10n      )
desVar(       "Tr" 10n  )
desVar(       "lndiffa" 1.441u      )
desVar(       "wndiffa" 960.03u     )
desVar(       "Cload" 100p    )
desVar(       "Rload" 100K    )
desVar(       "K" 5     )
desVar(       "wnout" 32u     )
desVar(       "wpcmir" 480u   )
desVar(       "wndiff" 960u   )
desVar(       "lnout" 2.88u   )
desVar(       "lndiff" 1.44u  )
desVar(       "lpcmir" 1.44u  )
desVar(       "gain" 1  )
desVar(       "inAmp" .5      )
desVar(       "wp" 120u )
desVar(       "wn" 120u )
desVar(       "wmir" 3u )
desVar(       "Vss" 0   )
desVar(       "Vdd" 2   )
desVar(       "Vcm" 1.0 )
desVar(       "lpmirr" .36u   )
desVar(       "Iref" 300u     )
desVar(       "ACout" 0 )
desVar(       "VswngN" "-VswngP"    )
option(     'reltol  "1e-5"
)
temp( 27 )
run()
;********** insert Plot Win setup commands here
; OL Freq Response First
;************************
hardCopyOptions( ?hcPlotterName "ps2" )
hardCopyOptions( ?hcOutputFile "RESULTS/CmirDiffAmp/OLFreqResp.ps" )
hardCopyOptions( “hcHeader” nil )
hardCopyOptions( “hcMailLogNames” nil )
OLFrRsp = newWindow()
Pheight = 600
Pwidth = 700
Px = 100
Py = 200
hiResizeWindow( OLFrRsp list(Px:Py Px+Pwidth:Py+Pheight))
addTitle(strcat("Typical Amplifier Characteristics CmirDiffAmp"
          getCurrentTime()))
addSubwindowTitle("Open Loop Freq Response")
;
;***************************************
; Reorder Outputs to get just the AC results
;*****************
;
displayMode( "composite")
Aolf = dB20((VF("/outp") - VF("/outn")))
plot( Aolf ?expr '( "Aol" ) )
PHol = phase(((VF("/outp") - VF("/outn")) / (VF("/inp") - VF("/inn"))))
plot( PHol ?expr '( "PHol" ) )
Aol = value(dB20((VF("/outp") - VF("/outn"))) 0)
GBW = gainBwProd((VF("/outp") - VF("/outn")))
Gmargin = gainMargin((VF("/outp") - VF("/outn")))
Pmargin = phaseMargin((VF("/outp") - VF("/outn")))
UGF = cross(dB20((VF("/outp") - VF("/outn"))) 0 1 "either")
DomPoleFreq = bandwidth((VF("/outp") - VF("/outn")) 3 "low")
;
;//**** Add Scalar Outputs to Plot **
;
addWaveLabel( 1 list( 1000 Aol) ; // point on the wave for the label
  sprintf(nil "DC OL Gain = %2.2f dB" Aol )
  ?textOffset 10:-30
  ?justify "lowerLeft"
  )
addWaveLabel( 1 list( UGF 0) ; // point on the wave for the label
  strcat("Unity Gain Frequency = " aelSuffixWithUnits(UGF "Hz" 4 ) )
  ?textOffset 10:-30
  ?justify "lowerLeft"
  )
addWindowLabel( list(0.15 0.2 )
  strcat("Gain Margin = " aelSuffixWithUnits(Gmargin "dB")
       "\nPhase Margin = " aelSuffixWithUnits(Pmargin "deg")
       "\nGain BandWidth = " aelSuffixWithUnits(GBW "dB_Hz")
       "\nDominant Pole = " aelSuffixWithUnits(DomPoleFreq "Hz")
  )
 )
;
;//***** Plot this window and setup a new one
```

```
;
hardCopy()
hardCopyOptions( ?hcOutputFile "RESULTS/CmirDiffAmp/RejRatios.ps" )
hardCopyOptions( “hcHeader” nil )
hardCopyOptions( “hcMailLogNames” nil )
RRplot = newWindow()
Pheight = 600
Pwidth = 700
Px = 100
Py = 200
hiResizeWindow( RRplot list(Px:Py Px+Pwidth:Py+Pheight))
addTitle(strcat("Typical Amplifier Characteristics CmirDiffAmp"
          getCurrentTime()))
addSubwindowTitle("Rejection Ratios")
;*************************************
CMRR = dB20((1 / getData("/VCM" ?result "xf-xf")))
plot( CMRR ?expr '( "CMRR" ) )
PSRRdd = dB20((1 / getData("/VDD" ?result "xf-xf")))
plot( PSRRdd ?expr '( "PSRR+" ) )
PSRRss = dB20((1 / getData("/VSS" ?result "xf-xf")))
plot( PSRRss ?expr '( "PSRR-" ) )
CmRefRR = dB20((1 / getData("/VOUT" ?result "xf-xf")))
plot( CmRefRR ?expr '( "CmRef_RR" ) )

dcCMRR = value(dB20((1 / getData("/VCM" ?result "xf-xf"))) 0)
dcCmRefRR = value(dB20((1 / getData("/VOUT" ?result "xf-xf"))) 0)
dcPSRRdd = value(dB20((1 / getData("/VDD" ?result "xf-xf"))) 0)
dcPSRRss = value(dB20((1 / getData("/VSS" ?result "xf-xf"))) 0)
addWindowLabel( list(0.15 0.2 ) ;// the relative location for the Text
  strcat("CMRR = " aelSuffixWithUnits(dcCMRR "dB")
       "\nPSRR+ = " aelSuffixWithUnits(dcPSRRdd "dB")
       "\nPSRR- = " aelSuffixWithUnits(dcPSRRss "dB")
       "\nCmRefRR = " aelSuffixWithUnits(dcCmRefRR "dB")
  )
 )
;
;//***** Plot this window and setup a new one
;
hardCopy()
hardCopyOptions( ?hcOutputFile "RESULTS/CmirDiffAmp/StepResponse.ps" )
hardCopyOptions( “hcHeader” nil )
hardCopyOptions( “hcMailLogNames” nil )
StepResp = newWindow()
Pheight = 600
Pwidth = 700
Px = 100
Py = 200
hiResizeWindow( StepResp list(Px:Py Px+Pwidth:Py+Pheight))
addTitle(strcat("Typical Amplifier Characteristics CmirDiffAmp"
          getCurrentTime()))
addSubwindowTitle("Large Signal Step Response")
;*************************************
displayMode( "strip")
TranOutput = (VT("/outp") - VT("/outn"))
plot( TranOutput ?expr '( "TranOutput" ) )
TranInput = (VT("/uginp") - VT("/uginn"))
plot( TranInput ?expr '( "TranInput" ) )
plot( VT("/outp") )
plot( VT("/outn") )
SlewRate = slewRate((VT("/outp") - VT("/outn")) 2.5e-06 t 3.5e-06 t 10
90)
Overshoot = overshoot((VT("/outp") - VT("/outn")) 2.5e-06 t 3.5e-06 t)
Slewsinglep = slewRate(VT("/outp") 2.5e-06 t 3.5e-06 t 10 90)

Slewsinglem = slewRate(VT("/outp") 3.5e-06 t 4.5e-06 t 10 90)

OvershootPp = overshoot(VT("/outp") 2.5e-06 t 3.5e-06 t)

OvershootPm = overshoot(VT("/outp") 3.5e-06 t 4.125e-06 t)

RiseTime = riseTime((VT("/outp") - VT("/outn")) 2.5e-06 t 3.5e-06 t 10
90)
Tsettle1 = (settlingTime((VT("/outp") - VT("/outn")) 3e-06 t 4e-06 t 1)
- cross((VT("/uginp") - VT("/uginn")) 0 2 "rising"))

OvershootN = overshoot(VT("/outn") 3e-06 t 3.125e-06 t)

Tsettle0r1 = (settlingTime((VT("/outp") - VT("/outn")) 3e-06 t 4e-06 t
0.1) - cross((VT("/uginp") - VT("/uginn")) 0 2 "rising"))

addWindowLabel( list(0.6 0.7 ) ;// the relative location for the Text
  strcat("Overshoot = " aelSuffixWithUnits(Overshoot "%")
       "\nSlewRate = " aelSuffixWithUnits(SlewRate "V/s")
       "\nTsettle1% = " aelSuffixWithUnits(Tsettle1 "s")
  )
 )
;
;//***** Plot this window and setup a new one
;
hardCopy()
hardCopyOptions( ?hcOutputFile "RESULTS/CmirDiffAmp/VosDrift.ps" )
hardCopyOptions( “hcHeader” nil )
hardCopyOptions( “hcMailLogNames” nil )
VosDrift = newWindow()
Pheight = 600
Pwidth = 700
Px = 100
Py = 200
hiResizeWindow( StepResp list(Px:Py Px+Pwidth:Py+Pheight))
addTitle(strcat("Typical Amplifier Characteristics CmirDiffAmp"
          getCurrentTime()))
addSubwindowTitle("Offset Voltage Drift")
;*************************************
displayMode( "composite")
plot( VS("/vos") ?expr '( "Vos" ))
Vos = VDC("/vos")
VosDrift = value(deriv(VS("/vos")) 27)
addWaveLabel( 1 list( 27 Vos) ; // point on the wave for the label
  strcat("Vos Drift = " aelSuffixWithUnits(VosDrift "V/deg C" 4 ) )
  ?textOffset 20:30
  ?justify "lowerLeft"
  )
;//***** Plot this window and setup New Analyses and run them..
hardCopy()
```


Typical Amplifier Characteristics CmirDiffAmpSep 21 15:28:07 2000
Open Loop Freq Response
▯ : PHol
▯ : Aol
100
0.00
−100
−200
−300
(deg, dB)
DC OL Gain = 52.26 dB
Unity Gain Frequency
Gain Margin = −30.53 dB
Phase Margin = 34.57 deg
Gain BandWidth = 39.98M dB_Hz
Dominant Pole = 97.27K Hz
1K
10K
100K
1M
10M
100M
1G
freq (Hz)


Typical Amplifier Characteristics CmirDiffAmpSep 21 15:28:10 2000
Rejection Ratios
▽ : CmRef_RR
▫ : PSRR+
△ : PSRR−
▫ : CMRR
130
120
110
100
(dB)
10K
100K
1M
10M
100M
1G
freq (Hz)
CMRR = 129.1 dB
PSRR+ = 115.6 dB
PSRR− = 113.9 dB
CmRefRR = 129 dB


Typical Amplifier Characteristics CmirDiffAmpSep 21 15:28:15 2000
Large Signal Step Response
▽: VT("/outn")
△: VT("/outp")
□: TranInput
□: TranOutput
Overshoot = 2.828 %
SlewRate = -26.24M V/s
Tsettle1% = -856.8n s
2.00
1.10
200m
2.00
1.10
200m
2.0
0.0
-2.0
2.0
0.0
-2.0
(V)
(V)
(V)
(V)
0.0
1.0u
2.0u
3.0u
4.0u
5.0u
time (s)


ical Amplifier Characteristics CmirDiffAmpSep 21 15:28:20 20
1
Offset Voltage Drift
Vos Drift = −54.83n V/deg C
: Vos
−17.0u
−18.0u
−19.0u
−20.0u
−21.0u
−22.0u
−23.0u
(V)
0.00
40.0
80.0
120
temp (C)