HSPICE® User Guide: RF Analysis

Version E-2010.12, December 2010

SYNOPSYS®

Copyright Notice and Proprietary Information

Right to Copy Documentation

The license agreement with Synopsys permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any. Licensee must assign sequential numbers to all copies. These copies shall contain the following legend on the cover page:

> "This document is duplicated with the permission of Synopsys, Inc., for the exclusive use of _____ and its employees. This is copy number _____."

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Registered Trademarks (®)

Synopsys, AMPS, Astro, Behavior Extracting Synthesis Technology, Cadabra, CATS, Certify, CHIPit, CoMET, Design Compiler, DesignWare, Formality, Galaxy Custom Designer, HAPS, HapsTrak, HDL Analyst, HSIM, HSPICE, Identify, Leda, MAST, METeor, ModelTools, NanoSim, OpenVera, PathMill, Physical Compiler, PrimeTime, SCOPE, Simply Better Results, SiVL, SNUG, SolvNet, Syndicated, Synplicity, the Synplicity logo, Synplify, Synplify Pro, Synthesis Constraints Optimization Environment, TetraMAX, UMRBus, VCS, Vera, and YIELDirector are registered trademarks of Synopsys, Inc.

Trademarks (™)

AFGen, Apollo, Astro-Rail, Astro-Xtalk, Aurora, AvanWaves, BEST, Columbia, Columbia-CE, Confirma, Cosmos, CosmosLE, CosmosScope, CRITIC, CustomExplorer, CustomSim, DC Expert, DC Professional, DC Ultra, Design Analyzer, Design Vision, DesignerHDL, DesignPower, DFTMAX, Direct Silicon Access, Discovery, Eclypse, Encore, EPIC, Galaxy, HANEX, HDL Compiler, Hercules, Hierarchical Optimization Technology, High-performance ASIC Prototyping System, HSIM^{plus}, i-Virtual Stepper, IICE, in-Sync, iN-Tandem, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, Liberty, Libra-Passport, Library Compiler, Magellan, Mars, Mars-Rail, Mars-Xtalk, Milkyway, ModelSource, Module Compiler, MultiPoint, Physical Analyst, Planet, Planet-PL, Polaris, Power Compiler, Raphael, Saturn, Scirocco, Scirocco-i, Star-RCXT, Star-SimXT, StarRC, System Compiler, System Designer, Taurus, TotalRecall, TSUPREM-4, VCS Express, VCSi, VHDL Compiler, VirSim, and VMC are trademarks of Synopsys, Inc.

Service Marks (^{sm})

MAP-in, SVP Café, and TAP-in are service marks of Synopsys, Inc.

SystemC is a trademark of the Open SystemC Initiative and is used under license.
ARM and AMBA are registered trademarks of ARM Limited.
Saber is a registered trademark of SabreMark Limited Partnership and is used under license.
All other product or company names may be trademarks of their respective owners.

Contents

Inside This Manual xi
The HSPICE Documentation Set xiii
Searching Across the HSPICE Documentation Set xiv
Conventions xv
Customer Support xv

1. HSPICE RF Features and Functionality 1
HSPICE RF Overview 2
HSPICE RF Features 3
HSPICE and HSPICE RF Differences 7
HSPICE/HSPICE RF Single Binary Integration 10
Use of Example Syntax 11

2. Getting Started 13
Running HSPICE RF Simulations 13
Netlist Overview 14
Parametric Analysis Extensions 14
Generating Output Files 14
HSPICE RF Output File Types 15
Using Custom WaveView™ 16

3. HSPICE RF Tutorial 19
Example 1: Using .LIN Analysis for a NMOS Low Noise Amplifier 20
Example 2: Using HB Analysis for a Power Amplifier 25
Example 3: Using HB Analysis for an Amplifier 29
Device Model Cards 33
Example 4: Using HBOSC Analysis for a Colpitts Oscillator 37
Example 5: Using HBOSC Analysis for a CMOS GPS VCO 41

Example 6: Using Multi-Tone HB and HBAC Analyses for a Mixer 50
Two-tone HB Approach 51
HBAC Approach 52
Comparing Results 53
Example 7: Using Shooting Newton Analysis on a Driven Phase Frequency Circuit and a Ring Oscillator 55
Introduction 55
Shooting Newton Analysis Setup 55
Driven Phase Frequency Detector Example 56
Ring Oscillator Example 62
Other Shooting Newton Analyses 66
RF Demonstration Input Files 67

4. Testbench Elements 69
Values for Elements 70
Ideal Transformer Format in HSPICE RF 70
DC Block and Choke Elements 71
Multi-Terminal Linear Elements 72
Scattering Parameter Data Element 72
Port Element 74
Port Element Syntax 75
Using the Port Element for Mixed-Mode Measurement 78
Steady-State Voltage and Current Sources 79
I and V Element Syntax 79
Steady-State HB Sources 83
Phase Differences Between HB and SIN Sources 85
Behavioral Noise Sources 86
Using Noise Analysis Results as Input Noise Sources 87
Power Supply Current and Voltage Noise Sources 88
Function Approximations for Distributed Devices 90
Foster Pole-Residue Form for Transconductance or Gain 90
Advantages of Foster Form Modeling 90
G and E-element Syntax 91
Complex Signal Sources and Stimuli 92
Vector-Modulated RF (VMRF) Source 92

Voltage and Current Source Elements 94
SWEEPBLOCK in Sweep Analyses 100
Input Syntax 101
Using SWEEPBLOCK in a DC Parameter Sweep 101
Using in Parameter Sweeps in TRAN, AC, and HB Analyses 102
Limitations 102
Clock Source with Random Jitter 102
Syntax of SIN, COS, and Pulse Sources 103
References 106

5. Steady-State Harmonic Balance Analysis 107
Harmonic Balance Analysis 108
Harmonic Balance Equations 109
Features Supported 110
Input Syntax 110
HB Analysis Spectrum 111
HB Analysis Options 113
Harmonic Balance Output Measurements 115
Output Syntax 116
Calculating Power Measurements After HB Analyses 119
Calculations for Time-Domain Output 122
Output Examples 124
Using .MEASURE with .HB Analyses 124
HB Output Data Files 126
Errors and Warnings 128
References 130

6. Steady-State Shooting Newton Analysis 131
SN Steady-State Time Domain Analysis 131
SN Analysis Syntax 132
SN Analysis Output 134
Shooting Newton with Fourier Transform (.SNFT) 137
.SNFT Input Syntax 138

7. Oscillator and Phase Noise Analysis 143

Harmonic Balance or Shooting Newton for Oscillator Analysis 144
Oscillator Classification 144

Harmonic Balance Oscillator Analysis (.HBOSC) 144
Input Syntax for Harmonic Balance Oscillator Analysis 145
HB Simulation of Ring Oscillators 150

HBOSC Analysis Using Transient Initialization 151
Additional .HBOSC Analysis Options 152
.HBOSC Output Syntax 153
Troubleshooting Convergence Problems 153

Oscillator Analysis Using Shooting Newton (.SNOSC) 158
.SNOSC Output Syntax 160

Phase Noise Analysis (.PHASENOISE) 161
Phase Noise Analysis Overview 161
Identifying Phase Noise Spurious Signals 163
PHASENOISE Input Syntax 164
Phase Noise Algorithms 167
PHASENOISE Output Syntax 168
Phase Noise Analysis Options 171
Measuring Phase Noise with .MEASURE PHASENOISE 172
Amplitude Modulation/Phase Modulation Separation 173

Accumulated Jitter Measurement for Closed Loop PLL Analysis 177
Jitter Measurements from Phase Noise 178

Small-Signal Phase-Domain Noise Analysis (.ACPHASENOISE) 184
AC Phase Noise Analysis Syntax 185
ACPHASENOISE Analysis .PRINT/.PROBE Syntax 185
Errors and Warnings 186

References 187

8. Large Signal Periodic AC, Transfer Function, and Noise Analyses 189

Multitone Harmonic Balance AC Analysis (.HBAC) 189
Prerequisites and Limitations 190
Input Syntax 191
Output Syntax 191
HBAC Output Data Files 193

Errors and Warnings 194
Shooting Newton AC Analysis (.SNAC) 195
Prerequisites and Limitations 196
Input Syntax 196
Output Syntax 197
SNAC Output Data Files 198
Errors and Warnings 199
SNAC Example 200
Phase Noise and Buffer Chains 200
Multitone Harmonic Balance Noise (.HBNOISE) 201
Supported Features 202
Input Syntax 203
Output Syntax 205
Output Data Files 206
Measuring HBNOISE Analyses with .MEASURE 206
Errors and Warnings 209
HBNOISE Example 209
Shooting Newton Noise Analysis (.SNNOISE) 209
Supported Features 210
Input Syntax 211
Output Syntax 214
Output Data Files 215
Measuring SNNOISE Analyses with .MEASURE 215
SNNOISE Analysis Example 217
Periodic Time-Dependent Noise Analysis (.PTDNOISE) 218
PTDNOISE Input Syntax 220
PTDNOISE Output Syntax and File Format 222
Error Handling and Warnings 224
Usage Example 224
Multitone Harmonic Balance Transfer Function Analysis (.HBXF) 226
Supported Features 227
Input Syntax 227
Output Syntax 228
Output Data Files 229
Example 229
HBXF Test Listing 230
Shooting Newton Transfer Function Analysis (.SNXF) 230

Input Syntax 231
Output Syntax 231
Output Data Files 232
Example 232
SNXF Test Listing 233
References 234

9. Transient Noise Analysis 235
Overview of HSPICE/HSPICE RF Transient Noise Analysis 236
Techniques Available in HSPICE 237
Modeling Frequency-Dependent Noise Sources 238
Monte Carlo Noise Analysis 239
Stochastic Differential Equation (SDE) Analysis 241
Setting up a .TRANNOISE Analysis 243
Input Syntax 243
Monte Carlo Output Data 245
SDE Output Data 246
Monte Carlo Examples 247
SDE Examples 247
Jitter Measurements from .TRANNOISE Analysis Results 247
Correlating Noise Results: .TRANNOISE (Monte Carlo) and .NOISE 251
Error Handling, Error Recovery, Status Reporting 252
References 253

10. S-parameter Analyses 255
Frequency Translation S-Parameter (HBLIN) Extraction 256
HB Analysis 258
Port Element 258
HBLIN Analysis 259
Output Syntax 262
Output Data Files 263
Large-Signal S-parameter (HBLSP) Analysis 263
Limitations 264
Input Syntax 265
Output Syntax 267

Output Data Files 268

11. Envelope Analysis 269

Envelope Simulation 269
Envelope Analysis Commands. 270
Output Syntax 273
Envelope Output Data File Format. 274

12. Post-Layout Analysis 277

Post-Layout Back-Annotation. 277
Standard Post-Layout Flow 279
Selective Post-Layout Flow 283
Additional Post-Layout Options 286
Selective Extraction Flow 287
Overview of DSPF Files 288
Overview of SPEF Files 295

Linear Acceleration 307
PACT Algorithm 308
PI Algorithm 309
Linear Acceleration Control Options Summary 309

13. Using HSPICE with HSPICE RF 313

RF Numerical Integration Algorithm Control 313

RF Transient Analysis Accuracy Control 314
.OPTION SIM_ACCURACY. 314
Algorithm Control 315

RF Transient Analysis Output File Formats 317
Tabulated Data Output 317
WDB Output Format. 318
TR Output Format 318
NW Output Format. 318
VCD Output Format 319
turboWave Output Format 319
Undertow Output Format 319
CSDF Output Format 320

Compressing Analog Files. 320

Eliminating Voltage Datapoints 320
Eliminating Current Datapoints 321

14. Advanced Features 323

Creating a Configuration File 323
Inserting Comments in a .hspice File 326

Using Wildcards in HSPICE RF 327

Limiting Output Data Size 327
SIM_POSTTOP Option 328
SIM_POSTSKIP Option 328
SIM_POSTAT Option 328
SIM_POSTDOWN Option 329
SIM_POSTSCOPE Option 329

Probing Subcircuit Currents 329

Generating Measurement Output Files 331

Optimization 332
Optimizing AC, DC. and TRAN Analyses 333
Optimizing HB Analysis 334
Optimizing HBOSC Analysis 335

Using CHECK Statements 336
Setting Global Hi/Lo Levels 336
Slew, Rise, and Fall Conditions 337
Edge Timing Verification 338
Setup and Hold Verification 338
IR Drop Detection 340

POWER DC Analysis 340
Power DC Analysis Output Format 341

POWER Analysis 342
Setting Default Start and Stop Times 342
Controlling Power Analysis Waveform Dumps 343

Detecting and Reporting Surge Currents 343

Index 345

About This Guide

This manual contains detailed reference information, application examples, and design flow descriptions that show how HSPICE RF features can be used for RF circuit characterization. The manual supplements the HSPICE user documentation by describing the additional features, built on top of the standard HSPICE feature set, that support the design of RF and high-speed circuits. Where necessary, the manual describes differences that might exist between HSPICE RF and HSPICE.

Note: This manual discusses only HSPICE RF features. For information on other HSPICE applications, see the other HSPICE manuals, listed in The HSPICE Documentation Set on page xiii.

Inside This Manual

This manual contains the chapters described below. For access to the other manuals in the HSPICE documentation set, see the next section, Searching Across the HSPICE Documentation Set on page xiv.

Chapter	Description
Chapter 1, HSPICE RF Features and Functionality	Introduces HSPICE RF features and functionality.
Chapter 2, Getting Started	Describes how to set up your environment, invoke HSPICE RF, customize your simulation, and redirect input and output.
Chapter 3, HSPICE RF Tutorial	Provides a quick-start tutorial for users new to HSPICE RF.
Chapter 4, Testbench Elements	Discusses the syntax for the specialized elements supported by HSPICE RF for high-frequency analysis and characterization.

Chapter	Description
Chapter 5, Parameters and Functions	Describes how to use parameters within HSPICE RF netlists.
Chapter 5, Steady-State Harmonic Balance Analysis	Describes how to use harmonic balance analysis for frequency-driven, steady-state analysis.
Chapter 6, Steady-State Shooting Newton Analysis	Describes HSPICE RF steady-state time domain analysis based on Shooting-Newton.
Chapter 7, Oscillator and Phase Noise Analysis	Describes how to use HSPICE RF to perform oscillator and phase noise analysis on autonomous (oscillator) circuits.
Chapter 8, Large Signal Periodic AC, Transfer Function, and Noise Analyses	Describes how to use harmonic balance-based and Shooting Newton AC analysis as well as nonlinear, steady-state noise analysis and XF analysis.
Chapter 9, Transient Noise Analysis	Describes the HSPICE and HSPICE RF solutions to perform transient noise analysis and compute noise statistics and their variation over time for circuits driven with non-periodic waveforms.
Chapter 10, S-parameter Analyses	Describes how to use periodically driven nonlinear circuit analyses as well as noise parameter calculation.
Chapter 11, Envelope Analysis	Describes how to use envelope simulation.
Chapter 12, Post-Layout Analysis	Describes the post-layout flow, including post-layout back-annotation, DSPF and SPEF files, linear acceleration, check statements, and power analysis.
Chapter 13, Using HSPICE with HSPICE RF	Describes how various analysis features differ in HSPICE RF as compared to standard HSPICE.
Chapter 14, Advanced Features	Describes how to invoke HSPICE RF and how to perform advanced tasks, including redirecting input and output.

The HSPICE Documentation Set

This manual is a part of the HSPICE documentation set, which includes the following manuals:

Manual	Description
HSPICE User Guide: Simulation and Analysis	Describes how to use HSPICE to simulate and analyze your circuit designs, and includes simulation applications. This is the main HSPICE user guide.
HSPICE User Guide: Signal Integrity	Describes how to use HSPICE to maintain signal integrity in your chip design.
HSPICE User Guide: RF Analysis	Describes how to use special set of analysis and design capabilities added to HSPICE to support RF and high-speed circuit design.
HSPICE Reference Manual: Commands and Control Options	Provides reference information for HSPICE and HSPICE RF commands and options.
HSPICE Reference Manual: Elements and Device Models	Describes standard models you can use when simulating your circuit designs in HSPICE, including passive devices, diodes, JFET and MESFET devices, and BJT devices.
HSPICE Reference Manual: MOSFET Models	Describes available MOSFET models you can use when simulating your circuit designs in HSPICE.
HSPICE Integration to Cadence® Virtuoso® Analog Design Environment User Guide	Describes use of the HSPICE simulator integration to the Cadence tool.
AMS Discovery Simulation Interface Guide for HSPICE	Describes use of the Simulation Interface with other EDA tools for HSPICE.
AvanWaves User Guide	Describes the AvanWaves tool, which you can use to display waveforms generated during HSPICE circuit design simulation.

Searching Across the HSPICE Documentation Set

You can access the PDF format documentation from your install directory for the current release by entering `-docs` on the terminal command line when the HSPICE tool is open.

Synopsys includes an index with your HSPICE documentation that lets you search the entire HSPICE documentation set for a particular topic or keyword. In a single operation, you can instantly generate a list of hits that are hyper-linked to the occurrences of your search term. For information on how to perform searches across multiple PDF documents, see the HSPICE release notes.

Note: To use this feature, the HSPICE documentation files, the Index directory, and the index.pdx file must reside in the same directory. (This is the default installation for Synopsys documentation.) Also, Adobe Acrobat must be invoked as a standalone application rather than as a plug-in to your web browser.

You can also invoke HSPICE and RF documentation in a browser-based help system by entering `-help` on your terminal command line when the HSPICE tool is open. This provides access to all the HSPICE manuals with the exception of the *AvanWaves User Guide* which is available in PDF format only.

Known Limitations and Resolved STARs

You can find information about known problems and limitations and resolved Synopsys Technical Action Requests (STARs) in the *HSPICE Release Notes* shipped with this release. For updates, go to SolvNet.

To access the *HSPICE Release Notes*:

1. Go to https://solvnet.synopsys.com/ReleaseNotes. (If prompted, enter your user name and password. If you do not have a Synopsys user name and password, follow the instructions to register with SolvNet.)
2. Select Download Center> HSPICE> version number> Release Notes.

Conventions

The following typographical conventions are used in Synopsys HSPICE documentation.

Convention	Description
`Courier`	Indicates command syntax.
`Italic`	Indicates a user-defined value, such as *`object_name`*.
`Bold`	Indicates user input—text you type verbatim—in syntax and examples. **Bold** indicates a GUI element.
[ ]	Denotes optional parameters, such as: `write_file [-f filename]`
`...`	Indicates that parameters can be repeated as many times as necessary: *`pin1 pin2 ... pinN`*
|	Indicates a choice among alternatives, such as `low | medium | high`
+	Indicates a continuation of a command line.
/	Indicates levels of directory structure.
Edit > Copy	Indicates a path to a menu command, such as opening the Edit menu and choosing Copy.
Control-c	Indicates a keyboard combination, such as holding down the Control key and pressing c.

Customer Support

Customer support is available through SolvNet online customer support and through contacting the Synopsys Technical Support Center.

Accessing SolvNet

SolvNet includes an electronic knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. SolvNet also gives you access to a wide range of Synopsys online services, which include downloading software, viewing Documentation on the Web, and entering a call to the Support Center.

To access SolvNet:

1. Go to the SolvNet Web page at http://solvnet.synopsys.com.
2. If prompted, enter your user name and password. (If you do not have a Synopsys user name and password, follow the instructions to register with SolvNet.)

If you need help using SolvNet, click Help on the SolvNet menu bar.

The link to any recorded training is
https://solvnet.synopsys.com/trainingcenter/view.faces

Access recent release update training by going to
https://solvnet.synopsys.com/search/advanced_search.faces

Contacting the Synopsys Technical Support Center

If you have problems, questions, or suggestions, you can contact the Synopsys Technical Support Center in the following ways:

- Open a call to your local support center from the Web by going to http://solvnet.synopsys.com/EnterACall (Synopsys user name and password required).
- Send an e-mail message to your local support center.
  - E-mail support_center@synopsys.com from within North America.
  - Find other local support center e-mail addresses at http://www.synopsys.com/support/support_ctr.
- Telephone your local support center.
  - Call (800) 245-8005 from within the continental United States.
  - Call (650) 584-4200 from Canada.
  - Find other local support center telephone numbers at http://www.synopsys.com/support/support_ctr.

1

HSPICE RF Features and Functionality

Introduces HSPICE RF features and functionality.

HSPICE RF is a special set of analysis and design capabilities that support the design of RF and high-speed circuits. This functionality, built on top of the standard HSPICE feature set, is also useful for analog and signal integrity applications. Although the HSPICE and HSPICE RF simulators share a common set of device models and simulation capabilities, HSPICE RF includes several modeling, simulation, and measurement additions that augment the ultimate-accuracy analog circuit simulation capabilities of HSPICE.

Note: This manual describes the additional features and capabilities of HSPICE RF. Where necessary, the manual describes differences between HSPICE RF and HSPICE. For information about standard HSPICE device models, syntax, and simulation control, you can refer to one of the other HSPICE manuals in the HSPICE documentation set, listed in The HSPICE Documentation Set on page xiii.

- HSPICE RF Overview
- HSPICE and HSPICE RF Differences
- HSPICE/HSPICE RF Single Binary Integration
- Use of Example Syntax

HSPICE RF Overview

HSPICE RF consists of:

- The hspicerf simulation engine
- The WaveView Analyzer tool

The hspicerf simulation engine contains extensions to HSPICE for RF design. These extensions are in the form of new analysis commands and new elements. The hspicerf simulation engine processes command and element syntax for new RF simulation features but also accepts standard HSPICE netlist files as input.

The WaveView Analyzer tool has been enhanced with special features for reading and analyzing data created by the HSPICE RF simulation engine. For a basic overview on how to use the WaveView Analyzer to view HSPICE RF output, Using Custom WaveView™ on page 16.

HSPICE RF can be launched with either the integrated executable (`hspice`) or in standalone mode (`hspicerf`). Users can run RF features with one HSPICE license and one HSPICE RF license or with two HSPICE licenses. The following major RF analyses can be run by entering hspice on the command-line:

- `.ENV`
- `.HB`
- `.SN`
- `.HBAC / .SNAC`
- `.HBNOISE / .SNNOISE`
- `.HBOSC /.SNOSC`
- `.HBXF / .SNXF`
- `.HBLIN`
- `.PHASENOISE`
- `.TRANNOISE`
- `.STATEYE`

HSPICE RF Features

This section briefly introduces the features of both the simulation engine and the waveform display tool.

HSPICE RF supports most HSPICE capabilities, and also includes:

- Steady-state frequency-domain analyses for linear and nonlinear circuits.
- High-performance transient analysis for faster simulation of high-speed digital and analog circuits.
- Port-wise automated `.AC` analyses for S (scattering) parameters. The `.LIN` command invokes extraction of noise and linear transfer parameters of a multi-port linear network. Extracts the S parameter and generates the N-port model.

  This command is used in conjunction with the `.AC` command to measure multiport S, Y, and Z parameters, noise parameters, stability and gain factors, and matching coefficients. Additionally, it is used with the Port element, which identifies the network ports and their impedances. You can also use mixed mode with `.LIN`.
- The Port (P) element identifies ports used in LIN analysis (multiport S, Y, or Z parameter and noise parameter extraction). A port element behaves as a noiseless impedance or a voltage source in series with an impedance, depending on the simulation being performed. Different impedances can be specified for DC, transient, AC, HB, and HBAC analyses.
- The S element describes a linear network using multi-port S, Y, or Z parameters in the form of a frequency table. These parameters can come from a `.LIN` simulation or from physical measurement. The standard Touchstone (1.0/2.0) and CITIfile formats are supported in addition to a proprietary HSPICE format.
- The syntax of voltage and current sources as well as Port elements supports the syntax for specifying power sources. In this case, the source value is interpreted as a power value in Watts or dBm units, and the Port element is implemented as a voltage source with a series impedance. The `.HBLSP` command invokes periodically driven nonlinear circuit analyses for power-dependent S parameters.
- Harmonic Balance (`.HB`) analysis using Direct and Krylov solvers. The `.HB` command invokes the single and multitone Harmonic Balance algorithm for periodic steady state analysis.

- TRANFORHB element parameter to recognize V/I sources that include SIN and PULSE transient descriptions as well as PWL and VMRF sources.
- Harmonic balance-based periodic AC analysis. The .HBAC command invokes periodic AC analysis for analyzing small-signal perturbations on circuits operating in a large-signal periodic steady state.
- Harmonic Balance-based Periodic Noise analysis (.HBNOISE) for noise analysis of periodically modulated circuits, includes stationary, cyclostationary, and frequency-dependent noise effects.
- Autonomous Harmonic Balance analysis. The .HBOSC command invokes the multitone, oscillator-capable Harmonic Balance algorithm for periodic steady state analysis.
- Perturbation analysis for Oscillator Phase Noise. The .HBAC command invokes phase periodic AC noise for oscillators circuits operating in a large-signal steady-state.
- Oscillator phase noise analysis, including both a nonlinear perturbation method and a PAC method, and includes stationary, cyclostationary, frequency-dependent, and correlated noise effects.
- Frequency translation S-parameter and noise figure extraction with the .HBLIN command.
- Envelope analysis. The .ENV command: invokes standard envelope simulation. The .ENVOSC command invokes envelope startup simulation. The .ENVFFT command invokes envelope Fast Fourier Transform simulation.
- .OPTION HBTRANINIT, HBTRANPTS, and HBTRANSTEP for transient analysis of ring oscillators.
- Convolution for transient analysis of S-parameter data models (S-element).
- Calculation of the transfer function from an arbitrary source and harmonic in the circuit to a designated output with the .HBXF command.
- Reading encrypted netlists.
- .OPTION SIM_ACCURACY provides simplified accuracy control for all simulations while .OPTION SIM_ORDER and SIM_TRAP improve transient analysis simulation controls.
- DSPF Flow for fast analysis using parasitic data from layout.
- .OPTION SIM_LA provides linear acceleration for RC network reduction for faster simulation.

- Saving `.PRINT` simulation output to a separate file.
- `HERTZ` variable for frequency-dependent equations.
- `IC=OFF` in element statements, IC parameter (initial conditions).
- Shooting Newton steady-state time domain analysis; the Shooting Newton algorithm provides functionality to support the following commands: `.SN`, `.SNAC` `.SNFT`, `.SNNOISE`, `.SNOSC`, and `.SNXF`.
- Periodic Time-Dependent Noise Analysis (`.PTDNOISE`) calculates the noise spectrum and the total noise at a point in time. Jitter in a digital threshold circuit can then be determined from the total noise and the digital signal slew rate.
- PSF output using the PSF and ARTIST options to include HSPICE RF analyses such as Harmonic Balance, Shooting Newton, and their associated small-signal analyses within the Cadence™ design environment.
- With the 2008.09 release, the SX-WaveView waveform viewing tool is capable of displaying the following HSPICE RF output file formats: *.tr#, .sc#, .hb#*, *.hr#, .ev#*, *.fe#*, *.jt#*, *.pn#*, *.ls#*, *.p2d#*, *.sw#*, *.ac#*, *.msn#*, *.sn#*, and Touchstone formatted files.
- HSPICE RF supports ISUB syntax consistent with HSPICE with the exception of wildcard support with the "?" sign. For example, `isub(x1.a?)` is not supported by HSPICE RF.
- HSPICE RF is consistent with HSPICE when handling `.PRINT` and `.PROBE` statements containing wildcards.
- When HSPICE RF is run in integrated executable (`hspice`) mode it is supported by HSPICE Precision Parallel (-hpp) for multithread simulations (but not in standalone mode).
- When HSPICE RF is run in integrated executable mode (`hspice`), the case sensitivity feature applies. (When run in standalone mode, HSPICE RF does not support case sensitivity.)

HSPICE RF also adds the following measurement capabilities to HSPICE:

- Small-signal scattering parameters.
- Small-signal two-port noise parameters.
- 1dB compression point.
- Intercept points (for example, IP2, IP3).

- Mixer conversion gain and noise figure.
- VCO output spectrum.
- Oscillator phase noise.
- Options simplify specifying levels of accuracy. As a result, HSPICE RF provides effective simulation solutions for RF, high-speed, and PCB signal integrity circuit challenges.
- Verilog-A is supported for all HSPICE RF analyses, including:
  - HB
  - HBOSC
  - HBAC
  - HBNOISE
  - HBXF
  - PHASENOISE
  - SN
  - SNOSC
  - ENV
- Standard restrictions for Verilog-A in periodic steady-state analysis are the same as other RF simulators that use Verilog-A. For example:
  - Verilog-A modules that are time dependent cannot be used for HB or SN unless the time dependence is periodic with a period that matches the HB or SN setup.
  - Verilog-A modules with “internal states” are not guaranteed to work correctly in HB or SN because the internal state cannot be tracked by the engine, so HB or SN may think it is converged to a periodic steady-state even though the internal state may not be in periodic steady state.
  - Some event-driven constructs in Verilog-A may not be compatible with HB.
- For RF netlist input guidelines see Chapter 4, Input Netlist and Data Entry in the *HSPICE User Guide: Simulation and Analysis*.

- For information on RF use of Parameters and Functions see Chapter 10, Parameters and Functions in the *HSPICE User Guide: Simulation and Analysis*.
- For information on use of Monte Carlo sweeps, see Chapter 20, Monte Carlo - Traditional Flow and Statistical Analysis in the *HSPICE User Guide: Simulation and Analysis*.

HSPICE and HSPICE RF Differences

The following tables give an overview of which features (Table 1) and device models (Table 2 on page 10) in HSPICE are not supported in HSPICE RF.

Table 1 HSPICE Features Not in HSPICE RF

Feature	See
Read hspice.ini file.	*HSPICE User Guide: Simulation and Analysis*
Short names for internal sub-circuits, such as 10:M1.	*HSPICE User Guide: Simulation and Analysis*
.MODEL types: AMP and PLOT for graphs	*HSPICE Reference Manual: Commands and Control Options*
Parameter definition (.PARAM) for Monte Carlo statistical functions	*HSPICE Reference Manual: Commands and Control Options*
.PLOT simulation output (obsolete for HSPICE too which uses .PRINT/.PROBE)	*HSPICE Reference Manual: Commands and Control Options*
.GRAPH simulation output (uses PLOT model type) .GRAPH and .PLOT obsolete in HSPICE.	*HSPICE User Guide: Simulation and Analysis* *HSPICE Reference Manual: Commands and Control Options*
.WIDTH, and many .OPTION COMMANDS	*HSPICE User Guide: Simulation and Analysis* *HSPICE Reference Manual: Commands and Control Options*
.OPTION ACCT	*HSPICE User Guide: Simulation and Analysis* *HSPICE Reference Manual: Commands and Control Options*

Table 1 HSPICE Features Not in HSPICE RF (Continued)

Feature	See
Element template output	*HSPICE User Guide: Simulation and Analysis*
Condition-controls: IF-ELSEIF-ELSE-ENDIF structure is not currently supported in HSPICE RF; however, you may be able to use the "a? b: c" construct in expressions, which performs if/then/else for expression evaluation	*HSPICE User Guide: Simulation and Analysis* *HSPICE Reference Manual: Commands and Control Options*
Group time delay parameters in AC analysis output	*HSPICE User Guide: Simulation and Analysis*
.DISTO distortion analysis and associated output commands	*HSPICE User Guide: Simulation and Analysis*
.SAVE and .LOAD	*HSPICE Reference Manual: Commands and Control Options*

Table 1 HSPICE Features Not in HSPICE RF (Continued)

Feature	See
Options that activate unsupported features in HSPICE RF: FAST GSHDC GSHUNT LIMPTS OFF RESMIN TIMERES All version options	*HSPICE Reference Manual: Commands and Control Options*
Options ignored by HSPICE RF, because they are not needed since they are replaced by automated algorithms: ABSH ABSV ABSVAR BELV BKPSIZ CHGTOL CONVERGE CSHDC CVTOL DCFOR DCHOLD DCON DCSTEP DI DV DVDT FAST FS FT GMAX GRAMP GSHDC GSHUNT ICSWEEP IMAX IMIN ITL3 ITL5 ITLPZ LIMPTS LVLTIM MAXAMP MBYPASS NEWTOL RELH RELI RELQ RELV RELVAR TRTOL All matrix options	
All error options Some Transient/AC input/output (I/O) options. HSPICE RF *does* support POST and PROBE options.	*HSPICE Reference Manual: Commands and Control Options*
Sub-circuit cross-listing in a .pa file	*HSPICE User Guide: Simulation and Analysis*, Chapter 3
-r command-line argument for a remote host	*HSPICE User Guide: Simulation and Analysis*
.OP supports node voltage for any time, but supports element values *only* for t=0.	*HSPICE Reference Manual: Commands and Control Options*

Table 1 HSPICE Features Not in HSPICE RF (Continued)

Feature	See
Sensitivity analysis (.SENS)	*HSPICE User Guide: Simulation and Analysis* *HSPICE Reference Manual: Commands and Control Options*
DC mismatch analysis (.DCMATCH)	*HSPICE User Guide: Simulation and Analysis* *HSPICE Reference Manual: Commands and Control Options*

Table 2 Device Models Not in HSPICE RF

Model	See
B-element: IBIS buffer— Bname n1 n2 [...] parameters	*HSPICE User Guide: Signal Integrity*
data-driven I element (current source)	*HSPICE Reference Manual: Elements and Device Models*
data-driven V element (voltage source)	*HSPICE Reference Manual: Elements and Device Models*
BJT LEVEL=10 (MODELLA)	*HSPICE Reference Manual: Elements and Device Models*, Chapter 5
MOSFET Levels 4-8.	*HSPICE Reference Manual: MOSFET Models*

HSPICE/HSPICE RF Single Binary Integration

With the D-2010.03 release, users can run HSPICE and HSPICE RF features from the unified HSPICE binary on all primary platforms. In addition, Synopsys has unified HSPICE and HSPICE RF licensing. Users can now run RF features with 1 HSPICE license and 1 HSPICE RF license or with 2 HSPICE licenses.

The command line entry is same as that of HSPICE:

```
hspice -i inputfile_name -o
```

All major RF analyses listed here are supported in the HSPICE and HSPICE RF Integration feature.

- `.ENV`
- `.HB`
- `.HBAC / .SNAC`
- `.HBXF / SNXF`
- `.HBLIN`
- `.HBNOISE / .SNNOISE`
- `.PHASENOISE`
- `.TRANNOISE`
- `.STATEYE`
- and more...

In addition, the other triggers for calling the RF engine include:

- vmrf sources
- cos sources
- power sources
- spur G and E control sources
- sweepblock
- source with “noisefile”
- L-element with transformer_nt
- G-element node1 node2 node3 node4 noise
- vmrf sources

Use of Example Syntax

To copy and paste proven syntax use the demonstration files shipped with your installation of HSPICE (see Listing of Demonstration Input Files). Attempting to copy and paste from the book or help documentation may present unexpected results, as text used in formatting may include hidden characters, white space, etc. for visual clarity only.

2

Getting Started

Describes how to set up your environment, invoke HSPICE RF, customize your simulation, redirect input and output, and use the Custom WaveView™ tool.

Before you run HSPICE RF, you need to set up several environment variables. You can also create a configuration file to customize your simulation run.

HSPICE RF accepts a netlist file from standard input and delivers the ASCII text simulation results to HTML or to standard output. Error and warning messages are forwarded to standard error output.

These topics are covered in the following sections:

- Running HSPICE RF Simulations
- Netlist Overview
- Parametric Analysis Extensions
- Generating Output Files
- Using Custom WaveView™

Running HSPICE RF Simulations

Use the following syntax to invoke HSPICE RF:

```
hspicerf [-a] inputfile [outputfile] [-n] [-h] [-v]
```

For a description of the `hspicerf` command syntax and arguments, see section hspicerf in the *HSPICE Reference Manual: Commands and Control Options*.

Netlist Overview

The circuit description syntax for HSPICE RF is compatible with the SPICE and HSPICE input netlist format. For a description of an input netlist file and methods of entering data, see Chapter 4, Input Netlist and Data Entry in the *HSPICE User Guide: Simulation and Analysis*.

Parametric Analysis Extensions

All major HSPICE RF analyses (`.TRAN`, `.AC`, `.DC`, and `.HB`) support the following parameter sweeps with the same syntax as standard HSPICE:

- LIN
- DEC
- OCT
- DATA
- POI

You can also use the `MONTE` keyword for a Monte Carlo analysis or the `OPTIMIZE` keyword for optimization.

Generating Output Files

HSPICE RF generates a table of simulation outputs.

- If the output is text (the default), the text is put into a *.lis* file.
- If you specify `.OPTION POST`, then HSPICE RF generates simulation output in a format suitable for a waveform display tool.
- The default output format for transient analysis in HSPICE RF is the same as in HSPICE: the .tr0 file format. For additional information, see Standard Output Files in the *HSPICE User Guide: Simulation and Analysis*.

The Synopsys interactive waveform display tool, Custom WaveView, can display both the text simulation results and binary output within the X-window environment.

All output functions (.PRINT, .PROBE, .MEASURE, and so on) can use power output variables in the form p(*devicename*), just as in HSPICE. You can also use the “power” keyword.

Larger output files from multi-million transistor simulations might not be readable by some waveform viewers. Options are available that enable you to limit the output file size. See Limiting Output Data Size on page 327 for more information.

HSPICE RF Output File Types

Table 3 shows the output file extensions that HSPICE RF analyses produce. The base file name of each output file is the same as the input netlist file’s base name. The # at the end of each file extension represents the .ALTER run from which the file came.

In general, text output from .PRINT commands is intended to be read by users, while binary output from .PROBE or .OPTION POST is intended to be read by the Custom WaveView tool.

Table 3 HSPICE RF Output File Types

Command	Text Output	Output for CosmosScope
AC analysis (.AC)	.printac#	.ac#
AC noise analysis (.NOISE)	.printac#	.ac#
DC sweep (.DC)	.printsw#	.sw#
Envelope analysis (.ENV)	.printev#	.ev#
Envelope FFT (.ENVFFT)	(none)	.fe#
Harmonic Balance (.HB)	.printhb#	.hb#
Harmonic Balance AC (.HBAC)	.printhb#	.hb#
.HBLIN analysis	.PRINT output: .printhl# S-param output: .SnP	.PROBE output: .hl# S-paramr output: .SnP
.HBLSP large-signal	.PRINT output: .printls# S-param output: .p2d#	.PROBE output: .ls# S-param output: .p2d#

Table 3 HSPICE RF Output File Types

Command	Text Output	Output for CosmosScope
.HBLSP small-signal	.PRINT output: .printss# S/noise output: .S2P#	.PROBE output: .ss# S/noise output: .S2P#
HBAC noise (.HBNOISE)	.printsnpn#	.pn#
Harmonic Balance OSC (.HBOSC)	.printhb#	.hb#
Harmonic Balance TRAN (.HBTRAN)	.printhr#	.hr#
Transfer Functions (.HBXF)	.printxf#	.xf#
Oscillator startup (.ENVOSC)	.printev#	.ev#
.LIN analysis	.PRINT output: .printac#; S/noise output: .sc#, .SnP, .citi#	.PROBE output: .ac#; S/noise output: .sc#, .SnP, .citi#
Phase Noise (.PHASENOISE)	.printsnpn#	.pn#
.SN analysis	.printsn#	.sn#
Transient analysis (.TRAN)	.printtr#	.tr#

Using Custom WaveView™

The Custom WaveView has been enhanced to support viewing and processing of HSPICE RF output files. This section presents a basic overview of how to use the WaveView to view HSPICE RF output.

- To start the Custom WaveView tool, type **wv** on the UNIX/Linux command line.
- Choose File > Import Waveform File (or press CTRL-O) to open the Open Waveform Files dialog box. Use the File Filters to limit the file names to Waveform Files. A list of the HSPICE RF file types can be found in Table 3

on page 15. When you open a file, its contents appear in the file browser. The file browser lists all open plot files. If you click on the '+' near a waveform file name, the hierarchy of the waveform file is shown and clicking on top level or any hierarchy level will display the contents of the waveform file to be shown in the signal browser. To plot one of the signals listed here in the waveview, you can either double-click the signal label or select, drag, and drop the signal label to the waveview.

- To create a panel, use the Panel > New menu and select the panel type, X-Y, Smith Chart, or Polar Plot. You can also use the panel icon in the toolbar to create new panels.
- To create a new chart, use the File > New menu. Select either XY Graph, Smith Chart, or Polar Chart. You can also use the first three icons in the toolbar to create new chart windows.
- Use the waveview tool bar to change how signals look, delete signals, group or ungroup signals.
- Right click on a frequency domain signal name and use the To Time Domain command to convert the histogram waveform (for example, from an .hb0 file) to a time domain waveform.
- Configure the axis scaling and grid by right-clicking a horizontal or vertical axis and selecting the desired scale or grid from the context sensitive menu.
- Zoom in and out, using the zoom icons on the waveview tool bar, or use the mouse cursor to select an area directly on the waveview.
- Dynamic meters can be used to see the signal's precise value at different points. From the menu, select Tools > Dynamic Meter or use the Dynamic Meter icon in the tool bar. Select and configure the desired Dynamic Meter. The meter can then be moved to the desired location on the selected signal.
- To use the measurement tools, choose Tools > Measurement. Three RF measurements have been added under the All tab of the Measurement Tool window:
  - 1db compression point (P1dB).
  - 2nd order intercept point (IP2).
  - 3rd order intercept point and spurious free dynamic range (IP3/SFDR).

3

HSPICE RF Tutorial

Provides a quick-start tutorial for users new to HSPICE RF.

This tutorial assumes you are familiar with HSPICE and general HSPICE syntax, but new to RF analysis features. The most basic RF analysis features are presented here, using simple examples. The end of this chapter contains a listing of HSPICE RF demonstration files available for your use when you have access to the HSPICE RF installation directory.

This tutorial covers the following examples:

- Example 1: Using .LIN Analysis for a NMOS Low Noise Amplifier
- Example 2: Using HB Analysis for a Power Amplifier
- Example 3: Using HB Analysis for an Amplifier
- Example 4: Using HBOSC Analysis for a Colpitts Oscillator
- Example 5: Using HBOSC Analysis for a CMOS GPS VCO
- Example 6: Using Multi-Tone HB and HBAC Analyses for a Mixer
- Example 7: Using Shooting Newton Analysis on a Driven Phase Frequency Circuit and a Ring Oscillator

In addition, there is a section listing all RF examples, including those not covered in these tutorial examples:

- RF Demonstration Input Files

Example 1: Using .LIN Analysis for a NMOS Low Noise Amplifier

The .LIN command simplifies the calculation of linear multi-port transfer parameters and noise parameters. In the LIN analysis, Port (P) elements are used to specify port numbers and their characteristic impedances. The analysis automatically computes the frequency-dependent complex transfer coefficients between ports. The result is a convenient means to get scattering parameters, noise parameters, stability parameters, and gain coefficients. The .LIN command renders obsolete the .NET command. By default, the output from the .LIN command is saved in the *.sc0 file format that can, in turn, be referenced as a model file for the S-parameter element.

To set up a linear transfer parameter analysis, the HSPICE input netlist must contain the following:

- Use the .AC command to activate small-signal AC analysis, and to specify a frequency sweep. Also, use the .AC command to specify any other parameter sweeps of interest.
- Use the .LIN command with the .AC command to activate small-signal linear transfer analysis. The .AC command specifies the base frequency sweep for the LIN analysis. The LIN analysis automatically performs multiple AC and NOISE analyses as needed to compute all complex signal transfer parameters.
- The necessary number of port (P) elements, numbered sequentially beginning with one to define the terminals of the multi-port network. For example, a two-port circuit must contain two port elements with one listed as port=1 and the other as port=2. The port elements define the ordering for the output quantities from the .LIN command (for example, the terminals for port=1 are used for S11, Y11, and Z11 measurements).

Much of the LIN analysis is automated so the HSPICE input netlist often does not require the following:

- AC signal sources. The `.LIN` command computes transfer parameters between the ports with no additional AC sources needed.
- DC sources. You can analyze a purely passive circuit without adding sources of any kind.

The following tutorial example shows how to set up a LIN analysis for an NMOS low noise amplifier circuit. This netlist is shipped with the HSPICE RF distribution as *gsmlna.sp* and is available in the directory: $*installdir*/demo/hspicerf/examples.

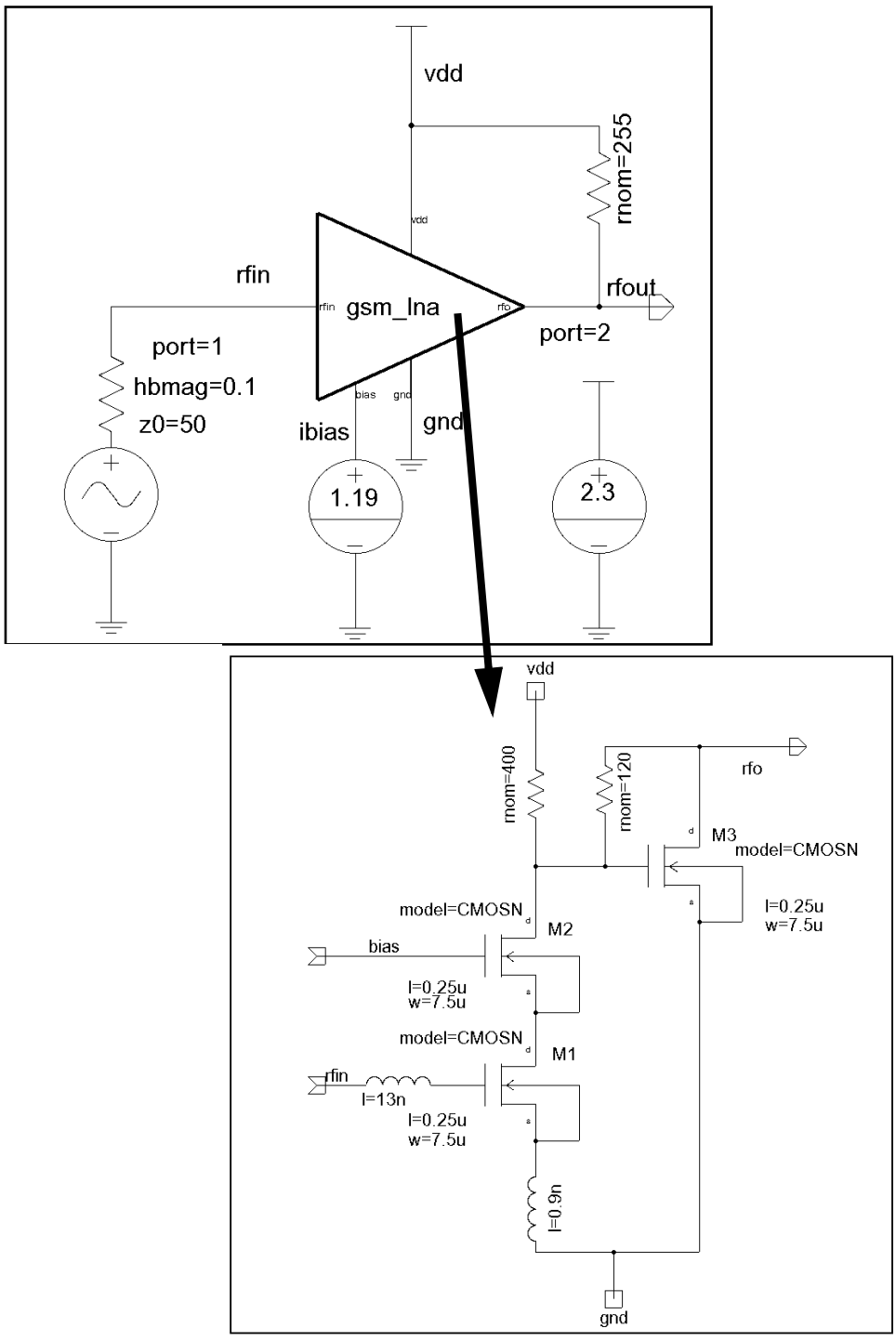


Figure 1 Schematic Showing Instantiation for Low Noise Amplifier

```
** NMOS 0.25um Cascode LNA for GSM applications
** setup for s-parameter and noise parameter measurements
**
.temp 27
.options post
.param   Vdd=2.3
.global gnd
**
** Cascode LNA tuned for operation near 1 GHz
**
M1 _n4 _n3 _n5 _n5 CMOSN l=0.25u w=7.5u
+     as=15p ad=15p ps=19u pd=19u m=80
M2 _n6 _n1 _n4 _n4 CMOSN l=0.25u w=7.5u
+     as=15p ad=15p ps=19u pd=19u m=80
M3  rfo _n6 gnd gnd CMOSN l=0.25u w=7.5u
+     as=15p ad=15p ps=19u pd=19u m=40
r1 _vdd _n6 400
l1 _n5 gnd l=0.9nH
l2 rfin _n3 l=13nH
vvb _n1 gnd   dc=1.19 $ bias for common base device
vvdd _vdd gnd dc=Vdd
rfb rfo _n6 120 $ feedback
**
** 50 Ohm input port (incl. bias), 255 Ohm output port.
**
P1 rfin gnd  port=1 z0=50  dc = 0.595  $ input port includes DC bias
P2 rfo _vdd port=2 z0=255 $ port doubles as pull-up resistor
**
** Measure s-parameters and noise parameters
**
.AC DEC 50 100MEG 5G
.LIN noisecalc=1 sparcalc=1 format=touchstone
.PRINT S11(DB) S21(DB) S12(DB) S22(DB) NFMIN
**
.PROBE NFMIN G_AS K_STABILITY_FACTOR
** Approximate parameters for TSMC 0.25 Process (MOSIS run T17B)
**
.MODEL CMOSN NMOS (                                LEVEL   = 49
3.1             TNOM    = 27              TOX     = 5.8E-9
+XJ      = 1E-7           NCH     = 2.3549E17       VTH0    = 0.3819327
+K1      = 0.477867       K2      = 2.422759E-3     K3      = 1E-3
+K3B     = 2.1606637      W0      = 1E-7            NLX     = 1.57986E-7
+DVT0W   = 0              DVT1W   = 0               DVT2W   = 0
+DVT0    = 0.5334651      DVT1    = 0.7186877       DVT2    = -0.5
+U0      = 289.1720829    UA      = -1.300598E-9    UB      = 2.3082E-18
```

```
+UC       = 2.841618E-11   VSAT     = 1.482651E5     A0       = 1.6856991
+AGS      = 0.2874763      B0       = -1.833193E-8   B1       = -1E-7
+KETA     = -2.395348E-3   A1       = 0              A2       = 0.4177975
+RDSW     = 178.7751373    PRWG     = 0.3774172      PRWB     = -0.2
+WR       = 1              WINT     = 0              LINT     = 1.88839E-8
+XL       = 3E-8           XW       = -4E-8          DWG      = -1.2139E-8
+DWB      = 4.613042E-9    VOFF     = -0.0981658     NFACTOR  = 1.2032376
+CIT      = 0              CDSC     = 2.4E-4         CDSCD    = 0
+CDSCB    = 0              ETA0     = 5.128492E-3    ETAB     = 6.18609E-4
+DSUB     = 0.0463218      PCLM     = 1.91946        PDIBLC1  = 1
+PDIBLC2  = 4.422611E-3    PDIBLCB  = -0.1           DROUT    = 0.9817908
+PSCBE1   = 7.982649E10    PSCBE2   = 5.200359E-10   PVAG     = 9.31443E-3
+DELTA    = 0.01           RSH      = 3.7            MOBMOD   = 1
+PRT      = 0              UTE      = -1.5           KT1      = -0.11
+KT1L     = 0              KT2      = 0.022          UA1      = 4.31E-9
+UB1      = -7.61E-18      UC1      = -5.6E-11       AT       = 3.3E4
+WL       = 0              WLN      = 1              WW       = 0
+WWN      = 1              WWL      = 0              LL       = 0
+LLN      = 1              LW       = 0              LWN      = 1
+LWL      = 0              CAPMOD   = 2              XPART    = 0.5
+CGDO     = 5.62E-10       CGSO     = 5.62E-10       CGBO     = 1E-12
+CJ       = 1.641005E-3    PB       = 0.99           MJ       = 0.4453094
+CJSW     = 4.179682E-10   PBSW     = 0.99           MJSW     = 0.3413857
+CJSWG    = 3.29E-10       PBSWG    = 0.99           MJSWG    = 0.3413857
+CF       = 0              PVTH0    = -8.385037E-3   PRDSW    = -10
+PK2      = 2.650965E-3    WKETA    = 7.293869E-3    LKETA    = -6.070E-3)
*
.END
```

A LIN analysis also includes the following:

- `.LIN` command:

  ```
  .LIN noisecalc=1 sparcalc=1 format=touchstone
  ```

  This invokes a LIN analysis and activates noise calculations and S-parameter output files in TouchStone format.

- Two port elements:

  ```
  P1 rfin gnd port=1 z0=50 dc=0.595
  ```

  Specifies that an input port is assumed between terminals `rfin` and ground, that it is has a 50 ohm termination, and it has a built-in DC bias of 0.595 V. The output (second) port is:

  ```
  P2 rfo _vdd port=2 z0=255
  ```

This syntax specifies that the output port is between terminals `rfo` and `vdd`, and is being used as a pull up resistor with impedance of 255 ohms.

- A `.PRINT` command for plotting the output S parameters in dB and the noise figure minimum.

To run this netlist, type the following command:

```
hspicerf gsmlna.sp
```

This produces three output files, named *gsmlna.ac0*, *gsmlna.s2p*, and *gsmlna.printac0*, containing the AC analysis results, S-parameter and noise parameter results, and the requested PRINT data.

To view the output:

1. Type **wv** *at the prompt* to invoke Custom WaveView.
2. Use File > Import Waveform File and select the *gmslna.s2p* file from the Open: Waveform Files dialog box.
3. Use Panel > New > Smith Chart or use the tool bar to open a blank Smith chart in the waveform.
4. Select the S(1,1) and S(2,2) signals in the signal browser. Drag and drop the selected signals in the Smith Chart.
5. Use Panel > New > Polar Plot or the use the tool bar to open a blank Polar chart.
6. Select the S(2,1) signal in the signal browser. Drag and drop the signal in the Polar chart to plot the complex gain of the LNA.
7. Use File > Import Waveform File and select the *gmslna.ac0* file from the Open: Waveform Files dialog box.
8. Use Waveview > New to open a new waveform.
9. Select the g_as (associated gain), k_stability_factor (Rollet stability factor) and nfmin (the noise figure minimum) signals in the signal browser. Drag and drop the selected signals in the new waveform.

Example 2: Using HB Analysis for a Power Amplifier

The `.HB` command computes periodic steady-state solutions of circuits. This analysis uses the Harmonic Balance (HB) technique for computing such solutions in the frequency domain. The circuit can be driven by a voltage,

power, or current source, or it may be an autonomous oscillator. The HB algorithm represents the circuit's voltage and current waveforms as a Fourier series, that is, a series of sinusoidal waveforms.

To set up a periodic steady-state analysis, the HSPICE input netlist must contain:

- A `.HB` command to activate the analysis. The `.HB` command specifies the base frequency (or frequencies, also called tones) for the analysis, and the number of harmonics to use for each tone. The `.HB` command can specify base tones so that the circuit solution is represented as a multi-dimensional Fourier series. The number of terms in the series are determined by the number of harmonics; more harmonics result in higher accuracy, but also longer simulation times and higher memory usage.
- One or more signal sources for driving the circuit in HB analysis, if the circuit is driven. In the case of autonomous oscillator analysis, no signal source is required. Signal sources are specified using the HB keyword on the voltage or current source syntax. Power sources are specified by setting the power switch on voltage/current sources to 1; in this case, the source value is treated as a power value in Watts instead of a voltage or current.

Optionally, the netlist can also contain a set of control option for optimizing HB analysis performance.

The following example shows how to set up a Harmonic Balance analysis on an NMOS Class C Power Amplifier. The example compares transient analysis results to Harmonic Balance results.

The following netlist performs both a transient and a Harmonic Balance analysis of the amplifier driven by a sinusoidal input waveform. The `accurate` option is set to ensure sufficient number of time points for comparison with HB. This example is included with the HSPICE RF distribution as *pa.sp* and is available in directory $*installdir*/demo/hspicerf/examples.

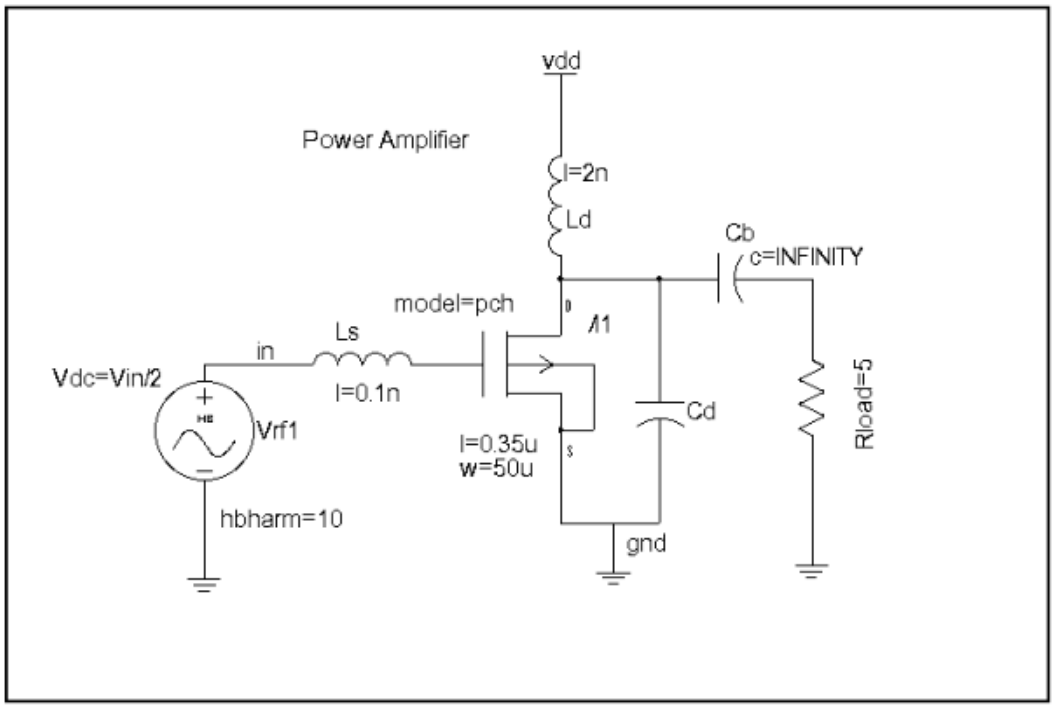


Figure 2 Power Amplifier

```
.options POST accurate
.param f0=950e6 PI=3.1415926 Ld=2e-9 Rload=5 Vin=3.0
.param Lin=0.1n Vdd=2 Cd='1.0/(4*PI*PI*f0*f0*Ld)'
M1 drain gt 0 0 CMOSN L=0.35u W=50u AS=100p AD=100p
PS=104u PD=104u M=80
Ls in gt   Lin $ gate tuning
Ld drain vdd Ld $ drain tuning
Cd drain 0    Cd
Cb drain out   INFINITY $ DC block
Rload out   0   Rload
Vdd vdd 0   DC    Vdd
Vrf1 in   0   DC 'Vin/2.0'
+ SIN ('Vin/2' 'Vin/2' 'f0' 0 0 90)
+ HB 'Vin/2' 0.0 1 1
.hb tones=f0 nharms=10
.tran 10p 10n
.probe hb p(Rload)
.probe tran p(Rload)
.include cmos49_model.inc
.end
```

An HB analysis uses the following:

- An .HB command:

  ```
  .hb tones=f0 nharms=10
  ```

  For a single tone analysis with base frequency 950 mHz and 10 harmonics.

- The HB source in Vrf1:

  ```
  HB 'Vin/2' 0.0 1 1.
  ```

  This creates a sinusoidal waveform matching the transient analysis one. The amplitude is Vin/2=1.5 V, and it applies to the first harmonic of the first tone, 950 MHz.

- A .PROBE command for plotting the output power:

  ```
  .probe hb p(Rload)
  ```

To run this netlist, type the following command:

```
hspicerf pa.sp
```

This produces two output files named *pa.tr0* and *pa.hb0*, containing the transient and HB output, respectively. To view and compare the output:

1. Type **wv** at the prompt to invoke Custom WaveView.
2. Use File > Import Waveform File and select the pa.tr0 and pa.hb0 files from the Open: Waveform Files dialog box.
3. Select the v(out) signal from the *pa.hb0* in the signal browser. Double-click on the signal name or drag and drop the signal in the waveform. The histogram shows lines at 950MHz, and multiples thereof, up to 9.5GHz.
4. In the waveform, right click in the name area of the panel containing the signal v(out), left-click on the waveform label for v(out) from the pa.hb0 file. From the Panel menu, choose Signal 'v(out)' > To Time-Domain.
5. In the Convert to Time Domain window, change the X-End(sec) value to 10n.
6. Click OK to accept the settings.
7. The new waveform shows a new time domain waveform named IFT.0|v(out).
8. Select the v(out) signal from the pa.tr0 in the signal browser. Drag and drop the signal in the waveform containing IFT.0|v(out). This should overlay the IFT.0|v(out) and v(out) signals on the same waveform. Zoom into the transitions to see the slight differences between the waveforms.

Example 3: Using HB Analysis for an Amplifier

This example takes the LNA circuit of Example 1 and performs a simulation using two closely spaced steady-state tones to study the compression and third order distortion properties of the amplifier. The example file *gsmlnaIP3.sp* is located at: /$*installdir*/demo/hspicerf/examples/
See Figure 1 on page 22 for the schematic view.

```
** NMOS 0.25um Cascode LNA for GSM applications
** Test bench setup for two-tone power sweep in dBm
** to extract IP3.
.temp 27
.options post=2
.param   Vdd=2.3
.global  gnd
.param Pin:dBm=-30.0
.param Pin=Pin:dBm
.param Pin:W='1.0e-3*pwr(10.0,Pin/10.0)'   $ Change to Watts for
sources
**
** Cascode LNA tuned for operation near 1 GHz
**
M1 _n4 _n3 _n5 _n5 CMOSN l=0.25u w=7.5u as=15p ad=15p ps=19u
pd=19u m=80
M2 _n6 _n1 _n4 _n4 CMOSN l=0.25u w=7.5u as=15p ad=15p ps=19u
pd=19u m=80
M3  rfo _n6 gnd gnd CMOSN l=0.25u w=7.5u as=15p ad=15p ps=19u
pd=19u m=40
r1 _vdd _n6 400
l1 _n5  gnd  l=0.9nH
l2 rfin _n3  l=13nH  $ 0.65n
vvb _n1 gnd   dc=1.19 $ bias for common base device
vinb rfinb gnd dc=0.595
lchk rfin rfinb INFINITY  $ Choke
cblk rfin rfind INFINITY  $ DC block
vvdd _vdd gnd dc=Vdd
rfb  rfo _n6 120      $ feedback
**
** Two-tone input source (DC blocked at this point)
**
Vin rfind gnd dc=0 power=1 z0=50  $ 50 Ohm src
+ HB Pin:W 0 1 1  $ tone 1
+ HB Pin:W 0 1 2  $ tone 2
Rload rfo _vdd R=255
**
** HB test bench to measure IP3 and IP2
**
.HB tones=900MEG,910MEG nharms=11 11 intmodmax=7
+ SWEEP Pin:dBm -50.0 0.0 2.0
.print HB P(Rload) P(Rload)[1,0] P(Rload)[2,0] P(Rload)[2,-1]
.probe HB P(Rload) P(Rload)[1,0] P(Rload)[2,0] P(Rload)[2,-1]
**
** Approximate parameters for MOSIS 0.25um process (run T17B)
**
.MODEL CMOSN NMOS(              LEVEL    = 49
+VERSION    = 3.1    TNOM    = 27    TOX    = 5.8E-9
+XJ        = 1E-7    NCH       = 2.3549E17    VTH0    = 0.3819327
```

```
+K1    = 0.477867    K2          = 2.422759E-3   K3     = 1E-3
+K3B    = 2.1606637    W0          = 1E-7    NLX     = 1.579864E-7
+DVT0W    = 0    DVT1W      = 0    DVT2W    = 0
+DVT0    = 0.5334651    DVT1       = 0.7186877    DVT2    = -0.5
+U0   = 289.1720829    UA    = -1.300598E-9   UB    = 2.308197E-18
+UC    = 2.841618E-11   VSAT     = 1.482651E5    A0    = 1.6856991
+AGS    = 0.2874763    B0    = -1.833193E-8    B1    = -1E-7
+KETA    = -2.395348E-3   A1    = 0    A2    = 0.4177975
+RDSW    = 178.7751373    PRWG    = 0.3774172    PRWB    = -0.2
+WR    = 1    WINT    = 0    LINT    = 1.888394E-8
+XL    = 3E-8    XW    = -4E-8    DWG    = -1.213938E-8
+DWB    = 4.613042E-9    VOFF    = -0.0981658  NFACTOR = 1.2032376
+CIT    = 0    CDSC    = 2.4E-4    CDSCD    = 0
+CDSCB    = 0    ETA0    = 5.128492E-3    ETAB    = 6.18609E-4
+DSUB    = 0.0463218    PCLM    = 1.91946    PDIBLC1   = 1
+PDIBLC2 = 4.422611E-3   PDIBLCB   = -0.1    DROUT    = 0.9817908
+PSCBE1 = 7.982649E10  PSCBE2 = 5.200359E-10  PVAG = 9.314435E-3
+DELTA    = 0.01    RSH   = 3.7    MOBMOD    = 1
+PRT    = 0    UTE    = -1.5    KT1    = -0.11
+KT1L    = 0    KT2    = 0.022    UA1    = 4.31E-9
+UB1    = -7.61E-18    UC1    = -5.6E-11    AT    = 3.3E4
+WL    = 0    WLN    = 1    WW    = 0
+WWN    = 1    WWL    = 0    LL    = 0
+LLN    = 1    LW    = 0    LWN    = 1
+LWL    = 0    CAPMOD    = 2    XPART    = 0.5
+CGDO    = 5.62E-10    CGSO    = 5.62E-10    CGBO    = 1E-12
+CJ    = 1.641005E-3    PB    = 0.99    MJ    = 0.4453094
+CJSW    = 4.179682E-10   PBSW    = 0.99    MJSW    = 0.3413857
+CJSWG    = 3.29E-10    PBSWG    = 0.99    MJSWG    = 0.3413857
+CF    = 0    PVTH0    = -8.385037E-3   PRDSW    = -10
+PK2 = 2.650965E-3 WKETA= 7.293869E-3   LKETA  = -6.070221E-3   )
*
.END
```

First, notice that we have defined variables that allow power to be swept in dBm units.

```
.param Pin:dBm=-30.0
.param Pin=Pin:dBm
.param Pin:W='1.0e-3*pwr(10.0,Pin/10.0)'
```

References to sources must use SI units in with the previous equation to convert from dBm to Watts. The colon (:) is used as a labeling convenience.

Second, a voltage source element is used as a two-tone power source by setting the power flag and a source impedance of 50 ohms is specified. The HB

keyword is used to identify the amplitude (interpreted in Watts with the power flag set), phase, harmonic index, and tone index for each tone.

```
Vin rfind gnd dc=0 power=1 z0=50 $ 50 Ohm src
+ HB Pin:W 0 1 1   $ tone 1
+ HB Pin:W 0 1 2   $ tone 2
```

Third, the `.HB` command designates the frequencies of the two tones and establishes the power sweep using the dBm power variable. The `intmodmax` parameter has been set to 7 to include intermodulation harmonic content up to 7th order effects.

```
.HB tones=900MEG,910MEG nharms=11 intmodmax=7
+ SWEEP Pin:dBm -50.0 0.0 2.0
```

Last, the HSPICE RF ability to specify specific harmonic terms is used in the `.PRINT` and `.PROBE` statements to pull out the signals of particular interest. Notice the three different formats:

```
.PRINT HB P(Rload)
```

This reference dumps a complete spectrum in RMS Watts for the power across resistor `Rload`.

```
.PRINT HB P(Rload)[1,0]
```

This reference selectively dumps the power in resistor `Rload` at the first harmonic of the 1st tone.

```
.PRINT HB P(Rload)[2,-1]
```

This reference selectively dumps the power in resistor `Rload` at the 3rd intermodulation product frequency (890 MHz).

To run this simulation, type the following at the command line:

```
hspicerf gsmlnaIP3.sp
```

Viewing Results using Custom WaveView

For this analysis, the .PRINT statement will generate a *gsmlnaIP3.printhb0* file. Assume you want to find out the output power through the load resistor at the first tone when the input power is 0.1mW.

To view the file:

1. Type **wv** *at the prompt* to invoke Custom WaveView.
2. Use File > Import Waveform File and select the *gsmlnaIP3.hb0* file.

3. Select the signal Pr(rload) [1,0] in the signal browser. Drag and drop the signal in the waveform. The X-axis will be the input power and the Y-axis will be the output power. The input and output power in dBm will be displayed.
4. To measure the 1dB compression point of the amplifier, select Measurement from the tools menu. In the Measurement Tools window, select the All tab. In the type section of the window, click on the RF radial button. In the measurement section, P1dB will be selected by default. Click OK; this will place a dynamic meter in the waveform.
5. Move the dynamic meter near the signal, the meter will show the 1dB compression point, the linear gain of the amplifier and the input power where the 1dB compression point is measured. For best results, the asymptotic line drawn by the dynamic meter should overlay the linear portion of the amplifier power curve.
6. Use Waveview > New to open a new waveform.
7. Select the signals Pr(rload) [1,0], Pr(rload) [2,-1], and Pr(rload) [2,0] in the signal browser. Drag and drop the signals in the waveform.
8. Click the signal mode icon in the waveform and select the Pr(rload) [1,0] and Pr(rload) [2,-1] signals by clicking on the waveform labels displayed in the waveform.
9. The 3rd order intercept point is also measured by using the measurement tool. In the measurement section, select IP3/SFDR. Click OK; this will place a dynamic meter in the waveform.
10. Move the dynamic meter near the Pr(rload)[1,0] signal, the meter will show the 3rd order intercept point and the input power where the 3rd order intercept point is measured. For best results, the asymptotic lines drawn by the dynamic meter should overlay the linear portion of the Pr(rload) [1,0] and Pr(rload) [2,-1] signals.

Device Model Cards

The following is an NMOS model in cmos49_model.inc file used in the power amplifier example. It is available in directory $<*installdir*>/demo/hspicerf/examples.

```
**
** NMOS IC Quadrature VCO circuit for GPS local oscillator

**
** Twin differential negative resistance VCOs
** using NMOS transistors for varactors, coupled
** to produce quadrature resonances.
** Design based on 0.35um CMOS process.
**
** References:
**  >P. Vancorenland and M.S.J. Steyaert, "A 1.57-GHz fully
**    integrated very low-phase-noise quadrature VCO,"
**    IEEE Trans. Solid-State Circuits, May 2002, pp.653-656.
**  >J. van der Tang, P. van de Ven, D. Kasperkovitz, and A.
Roermund,
**   "Analysis and design of an optimally coupled 5-GHz quadrature
**   LC oscillator,"  IEEE Trans. Solid-State Circuits, May 2002,
**    pp.657-661.
** >F. Behbahani, H. Firouzkouhi, R. Chokkalingam, S. Delshadpour,
**   A. Kheirkhani, M. Nariman, M. Conta, and S. Bhatia,

**   "A fully integrated low-IF CMOS GPS radio with on-chip analog
**  image rejection,"  IEEE Trans. Solid-State Circuits, Dec. 2002,
**   pp. 1721-1727.
**
** Setup for Harmonic Balance Analysis
**
** Oscillation Frequency: ~ 1575 MHz (GPS L1 frequency)
** Amplitude:  ~5 Volts peak-to-peak (zero to 5V)
** Vdd: 2.5 V
**
**
** Simulation Options :
.option POST
**
.param Vtune=2.0  $ Failures: vtune=1
.param Cval=0.2p
*---------------------------------
Vtune vc gnd  DC Vtune
Vdd vdd gnd 2.5
*---------------------------------
* First oscillator section
** Low-Q resonator with Vdd at center tap of inductors
R1a IP ri 100k  $ These R's set the Q
R1b ri IN 100k
L1 IP vdd 16.5nH
L2 vdd IN 16.5nH
Cc1 IP gnd Cval $ I to Q
```

```
Cc2 IN gnd Cval $ -I to Q
** Differential fets
M1 IP IN cs gnd NMOS l=0.35u w=15u
M2 IN IP cs gnd NMOS l=0.35u w=15u
** Bias fet - bias at Vdd -- too high?
Mb cs  vdd gnd gnd NMOS l=0.35u w=15u
** fets used as varactors
Mt1 vc IP vc gnd NMOS l=0.35u w=2u M=50
Mt2 vc IN vc gnd NMOS l=0.35u w=2u M=50
*--------------------------------
** Second oscillator section
** Low-Q resonator with Vdd at center tap of inductors
R1a_b QP ri_b 100k  $ These R's set the Q
R1b_b ri_b QN 100k
L1_b QP vdd 16.5nH
L2_b vdd QN 16.5nH
Cc1_b QP gnd Cval $ -Q to -I
Cc2_b QN gnd Cval $ -Q to I
** Differential fets
M1_b QP QN cs_b gnd NMOS l=0.35u w=15u
M2_b QN QP cs_b gnd NMOS l=0.35u w=15u
** Bias fet - bias at Vdd -- too high? 2nd in parallel
Mb_b cs_b  vdd gnd gnd NMOS l=0.35u w=15u
** fets used as varactors

Mt1_b vc QP vc gnd NMOS l=0.35u w=2u M=50
Mt2_b vc QN vc gnd NMOS l=0.35u w=2u M=50
*
*--------------------------------
* Differentiators Coupling transistors for quadrature
*
.param Cdiff=0.14p difMsize=50u
vidiff  dbias gnd 1.25
viqdiff vdcdif gnd 1.75
Midiff1 dQP dbias gnd gnd NMOS l=0.35u w=difMsize
Midiff2 dQN dbias gnd gnd NMOS l=0.35u w=difMsize
Midiff3 dIN dbias gnd gnd NMOS l=0.35u w=difMsize
Midiff4 dIP dbias gnd gnd NMOS l=0.35u w=difMsize
Cdiff1  dQP QP Cdiff
Cdiff2  dQN QN Cdiff
Cdiff3  dIN IN   Cdiff
Cdiff4  dIP IP   Cdiff
Mc_QP1 IP vdcdif dQP gnd NMOS l=0.35u w=difMsize
Mc_QN2 IN vdcdif dQN gnd NMOS l=0.35u w=difMsize
Mc_QN3 QP vdcdif dIN gnd NMOS l=0.35u w=difMsize
Mc_QP4 QN vdcdif dIP gnd NMOS l=0.35u w=difMsize
*--------------------------------
* Transient Analysis Test Bench
```

```
* Use to show oscillator start up
* 2mA pulse used to start oscillator
*iosc IP IN PULSE ( 0 2m .01n .01n .01n 10n 1u )
*.probe tran v(IP) v(IN)
*.print tran v(IP) v(IN)
*.TRAN .01n 10n
*------------------------------
* Harmonic Balance Test Bench
*
.sweepblock vtune_sweep

+ 0 5 0.2
+ 2 3 0.1
.HBOSC tones=1550e6 nharms=12
+ PROBENODE=IP,QN,4
+ sweep Vtune sweepblock=vtune_sweep
**
.phasenoise dec 10 100 1e7
.print phasenoise phnz
.probe phasenoise phnz
.print hb v(IP,IN) v(IP,IN)[1] v(QP,QN) v(QP,QN)[1]
.probe hb v(IP,IN) v(IP,IN)[1] v(QP,QN) v(QP,QN)[1]
.probe hb hertz[1][1]
*
* NMOS Device from MOSIS 0.35um Process
*
* BSIM3 VERSION 3.1 PARAMETERS
*
* DATE: Mar  8/00
* LOT: n9co                  WAF: 07
* Temperature_parameters=Default
*
.MODEL NMOS NMOS (                                LEVEL   = 49
+VERSION = 3.1            TNOM    = 27          TOX = 7.9E-9
+XJ      = 1.5E-7         NCH     = 1.7E17        VTH0 = 0.5047781
+K1      = 0.5719698      K2      = 0.0197928     K3     = 33.4446099
+K3B     = -3.1667861     W0      = 1E-5        NLX    = 2.455237E-7
+DVT0W   = 0               DVT1W   = 0             DVT2W   = 0
+DVT0    = 2.8937881      DVT1    = 0.6610934     DVT2   = -0.0446083
+U0     = 421.8714618    UA     = -1.18967E-10  UB     = 1.621684E-18
+UC      = 3.422111E-11   VSAT    = 1.145012E5    A0     = 1.119634
+AGS      = 0.1918651      B0      = 1.800933E-6    B1      = 5E-6
+KETA     = 3.313177E-3    A1       = 0              A2      = 1
+RDSW    = 984.149934    PRWG   = -1.133763E-3  PRWB   = -7.19717E-3
+WR      = 1           WINT    = 9.590106E-8   LINT    = 1.719803E-8
+XL      = -5E-8         XW      = 0           DWG    = -2.019736E-9
+DWB      = 6.217095E-9    VOFF    = -0.1076921     NFACTOR = 0
```

```
+CIT      = 0              CDSC     = 2.4E-4          CDSCD    = 0
+CDSCB    = 0           ETA0    = 0.0147171     ETAB    = -7.256296E-3
+DSUB    = 0.3377074     PCLM    = 1.1535622     PDIBLC1 = 2.946624E-4
+PDIBLC2 = 4.171891E-3   PDIBLCB = 0.0497942     DROUT   = 0.0799917
+PSCBE1  = 3.380501E9    PSCBE2  = 1.69587E-9     PVAG    = 0.4105571
+DELTA    = 0.01           MOBMOD   = 1               PRT      = 0
+UTE      = -1.5           KT1      = -0.11           KT1L     = 0
+KT2     = 0.022         UA1     = 4.31E-9       UB1     = -7.61E-18
+UC1      = -5.6E-11       AT       = 3.3E4           WL       = 0
+WLN      = 1             WW       = -1.22182E-15   WWN      = 1.1657
+WWL      = 0              LL       = 0               LLN      = 1
+LW       = 0              LWN      = 1               LWL      = 0
+CAPMOD   = 2             XPART    = 0.4           CGDO     = 3.73E-10
+CGSO     = 3.73E-10      CGBO     = 1E-11         CJ       = 8.988141E-4
+PB      = 0.8616985     MJ      = 0.3906381     CJSW    = 2.463277E-10
+PBSW     = 0.5072799     MJSW     = 0.1331717     PVTH0    = -0.0143809
+PRDSW   = -81.683425    WRDSW   = -107.8071189  PK2     = 1.210197E-3
+WKETA   = -1.00008E-3    LKETA   = -6.1699E-3     PAGS    = 0.24968
+AF       = 1.0            KF       = 1.0E-30       )
*
.END
```

The following is the BJT model file, bjt.inc used in oscillator example. It is available in directory $*installdir*/demo/hspicerf/examples.

```
* RF Wideband NPN Transistor die SPICE MODEL
.MODEL RF_WB_NPN   NPN
+ IS     = 1.32873E-015   BF     = 1.02000E+002
+ NF     = 1.00025E+000   VAF    = 5.19033E+001
+ EG     = 1.11000E+000   XTI    = 3.00000E+000
+ CJE    = 2.03216E-012   VJE    = 6.00000E-001
+ MJE    = 2.90076E-001   TF     = 6.55790E-012
+ XTF    = 3.89752E+001   VTF    = 1.09308E+001
+ ITF    = 5.21078E-001   CJC    = 1.00353E-012
+ VJC    = 3.40808E-001   MJC    = 1.94223E-001
```

Example 4: Using HBOSC Analysis for a Colpitts Oscillator

This section demonstrates HSPICE RF oscillator analysis using a single transistor oscillator circuit. Oscillator analysis is an extension of Harmonic Balance in which the base frequency itself is an unknown to be solved for. In

oscillator analysis, the user supplies a guess at the base frequency, and no voltage or current source stimulus is needed.

To activate oscillator analysis, include a `.HBOSC` command with:

- The `TONE` parameter set to a guess of the oscillation frequency.
- The `PROBENODE` parameter set to identify an oscillating node or pair of nodes. Always specify a pair of nodes; if only one node oscillates, specify ground as the second node. To speed up the simulation, also supply a guess at the magnitude of the oscillating voltage across these nodes.
- The `FSPTS` parameter set to a frequency range and number of search points. When you set `FSPTS`, HSPICE RF precedes the HBOSC analysis with a frequency search in the specified range to obtain an optimal initial guess for the oscillation frequency. This can accelerate the HB oscillator convergence.

In conjunction with oscillator analysis, HSPICE RF can perform phase noise analysis. Phase noise analysis measures the effect of transistor noise on the oscillator frequency. Phase noise analysis is activated using the `.PHASENOISE` command; this command sets a set of frequency points for phase noise analysis. The `.PRINT` and `.PROBE` commands can be used to output phase noise values.

The following netlist, *osc.sp*, simulates an oscillator, and performs phase noise analysis. This example is included with the HSPICE RF distribution as *pa.sp* and is available in directory $*installdir*/demo/hspicerf/examples.

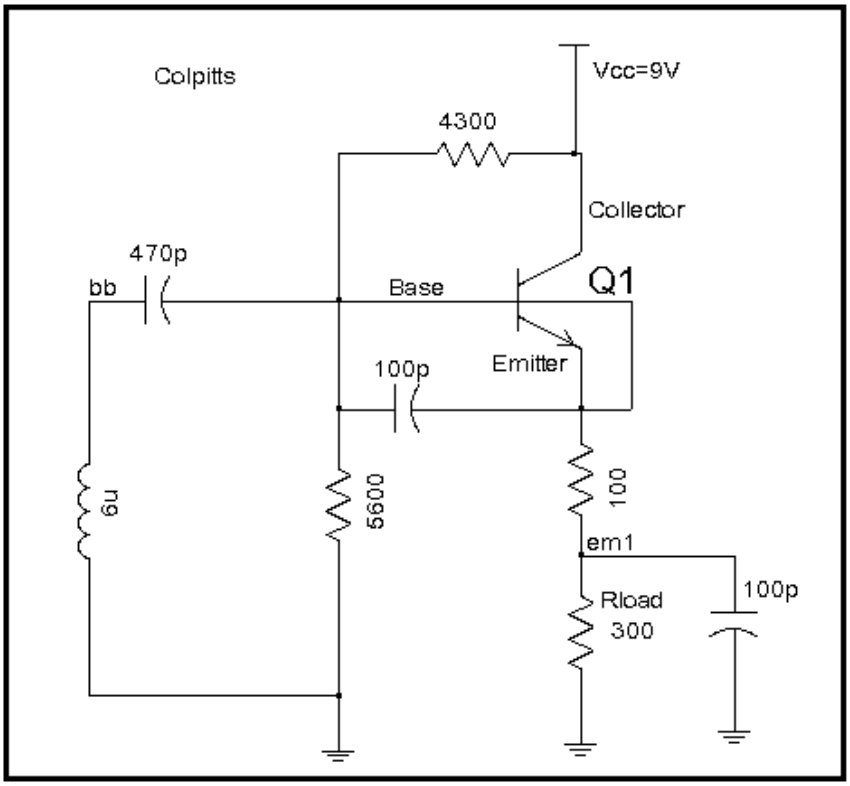


Figure 3 Colpitts Oscillator

Use the `.HBOSC` command with the `PROBENODE` and `FSPTS` parameters set.

```
PROBENODE=emitter,0,4.27
```

Identifies the emitter node as an oscillating node, and provides a guess value of 4.27 volts for the oscillation amplitude at the emitter node.

```
FSPTS=40,9e6,1.1e7
```

Causes an initial frequency search using 40 equally-spaced points between 9 and 11 MHz.

In the `.PHASENOISE`, `.PRINT`, and `.PROBE` commands:

```
.PHASENOISE V(emitter) dec 10 10k 1meg
```

Runs phase noise analysis at the specified offset frequencies, measured from the oscillation carrier frequency. The frequency points specified here are on a logarithmic scale, 10 points per decade, 10 kHz to 1 MHz.

- `.PROBE PHASENOISE PHNOISE` and the similar `.PRINT` command instruct HSPICE RF to output phase noise results to the *osc.pn0* and *osc.printpn0* files.

```
**
** Uses emitter resistor limiting to keep output sinusoidal.
** Output can be taken at the emitter (eml node).
**
*-------------------------------------------------------
* Options for Oscillator Harmonic Balance Analysis...
*
.OPTIONS post sim_accuracy=100 hbsolver=0
*-------------------------------------------------------
* Bias NPN transistor for 5V Vce, 10mA Ic
* Emitter follower Colpitts design
Vcc collector 0      9V
Q1 collector base emitter emitter RF_WB_NPN
Re1   emitter   eml     100
RLoad eml        0      300
Rb1   collector base 4300
Rb2   base       0     5600
*
*-------------------------------------------------------
* Capacitive feedback network
Ce   0    eml      100pF
Cfb base eml      100pF
Cbb base bb       470pF
Lb bb     0      6uH
*-------------------------------------------------------
* Simulation control for automated oscillator analysis
*
.HBOSC tones=1.0e7 nharms=15
+PROBENODE=emitter,0,4.27
+FSPTS=40,9.e6,1.1e7
*
.PHASENOISE V(emitter) DEC 10 10K 1MEG
+METHOD=0 CARRIERINDEX=1
*
.print hbosc vm(eml) vp(eml) vr(emitter) vi(emitter)
.print hbosc vm(emitter) vp(emitter) P(Rload)
.print phasenoise phnoise
.probe phasenoise phnoise
.probe hbosc v(emitter) v(eml)
.include bjt.inc
.END
```

After you run this netlist, examine the *osc.printhb0* file.

- At the top is the oscillator frequency (about 10.14 MHz) and the `.PRINT HBOSC` output.
- The first 2 lines show that the eml node oscillates around 3V with an amplitude of about 2.85V.
- The emitter node oscillates around 4V with an amplitude of about 4.27V.

Also examine the osc.printpn0 file, which contains the phase noise results in text form.

You can view the *osc.hb0* and *osc.pn0* files in Custom WaveView.

1. Type **wv** *at the prompt* to invoke Custom WaveView.
2. Use File > Import Waveform File and select the *osc.hb0* file from the Open: Waveform Files dialog box.
3. Select the v(emitter) signal in the signal browser. Double-click on the signal name or drag and drop the signal in the waveform.
4. In the waveform, right-click in the name area of the panel containing the signal v(emitter), left-click on the waveform label for v(emitter) in the waveform. From the Panel menu, choose Signal 'v(emitter)' > To Time-Domain.
5. To accept the defaults for range and interval, click OK in the Convert to Time domain window.
6. In a new waveform, you should now see a time domain waveform named IFT.0|v(emitter).

To run a transient simulation for comparison:

1. Use the `.TRAN 1n 10u` command.
2. Add `ic=10n` to the `Lb` inductor.

   The resulting waveforms should be the same as those from HB oscillator analysis.

Example 5: Using HBOSC Analysis for a CMOS GPS VCO

This second oscillator analysis example involves two negative resistance oscillators coupled at 90 degrees. MOS capacitors are used as varactors. This

VCO topology is common for GPS applications and produces quadrature LO outputs near 1550 MHz. The purpose of this example is to generate the VCO tuning curve (output level and frequency as a function of tuning voltage), as well as its phase noise characteristics as a function of tuning voltage.

The oscillator analysis is activated using the `.HBOSC` command:

- The `TONE` parameter sets an oscillation frequency (near 1550 MHz).
- The `NHARMS` parameter sets the harmonic content to 11th order.
- The `PROBENODE` parameters identify the drain pins across the first oscillator section as the pair of oscillating nodes. This is a differential oscillator, and the approximate value for this differential amplitude is 6.1 V.
- The `FSPTS` parameters set the search frequency range between 1500 and 1600 MHz.
- The `SWEEP` parameters set a tuning voltage sweep from 2.0 to 3.2 V.

The following example is based on demonstration netlist *gpsvco.sp*, which is available in directory $*installdir*/demo/hspicerf/examples. This netlist simulates the oscillator schematic Figure 4 and performs phase noise analysis.

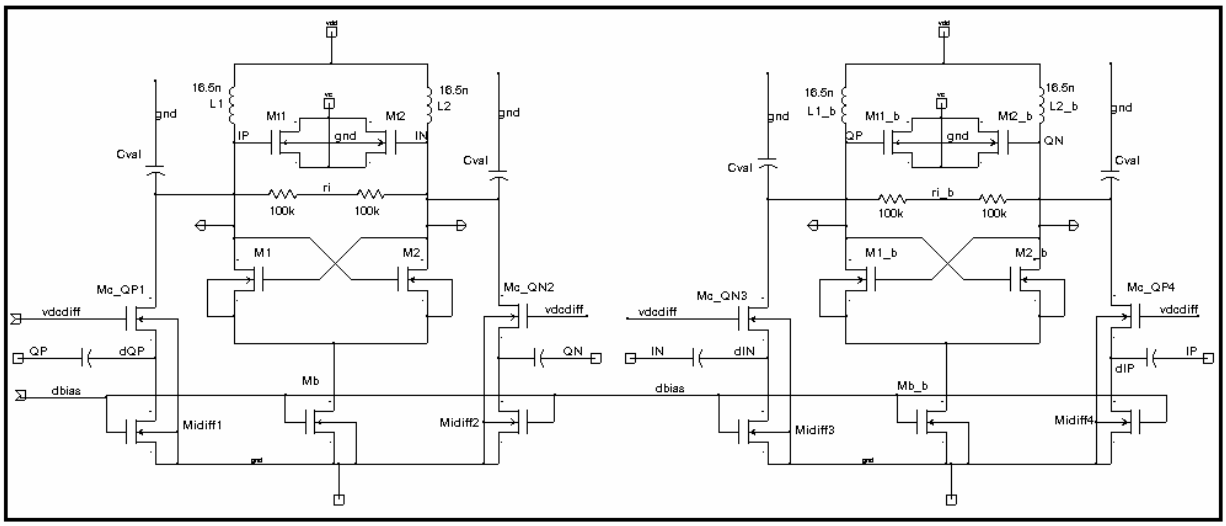


Figure 4 VCO Schematic

```
**
** NMOS IC Quadrature VCO circuit for GPS local oscillator
**
** Twin differential negative resistance VCOs using NMOS
** transistors for varactors, coupled to produce quadrature
** resonances.
** Design based on 0.35um CMOS process.
**
** References:
**  >P. Vancorenland and M.S.J. Steyaert, "A 1.57-GHz fully
**    integrated very low-phase-noise quadrature VCO,"
**    IEEE Trans. Solid-State Circuits, May 2002, pp.653-656.
**  >J. van der Tang, P. van de Ven, D. Kasperkovitz, and A.
Roermund,
**   "Analysis and design of an optimally coupled 5-GHz quadrature
**    LC oscillator,"  IEEE Trans. Solid-State Circuits, May 2002,
**     pp.657-661.
** >F. Behbahani, H. Firouzkouhi, R. Chokkalingam, S. Delshadpour,
**    A. Kheirkhani, M. Nariman, M. Conta, and S. Bhatia,
**   "A fully integrated low-IF CMOS GPS radio with on-chip analog
**  image rejection," IEEE Trans. Solid-State Circuits, Dec. 2002,
**    pp. 1721-1727.
**
** Setup for Harmonic Balance Analysis
**
** Oscillation Frequency: ~ 1575 MHz (GPS L1 frequency)
** Amplitude:  ~5 Volts peak-to-peak (zero to 5V)
** Vdd: 2.5 V
**
** HSPICE Simulation Options:
*.option delmax=1n ACCURATE LIST NODE
**
** HSPICE RF Simulation Options :
.option sim_accuracy=10
**
*.option savehb='a.hbs' loadhb='a.hbs'
.option POST
.param Vtune=2.0  $ Failures: vtune=1
.param Cval=0.2p
*---------------------------------
Vtune vc gnd  DC Vtune
Vdd vdd gnd 2.5
*---------------------------------
* First oscillator section
** Low-Q resonator with Vdd at center tap of inductors
R1a IP ri 100k  $ These R's set the Q
R1b ri IN 100k
L1 IP vdd 16.5nH
```

```
L2 vdd IN 16.5nH
Cc1 IP gnd Cval $ I to Q
Cc2 IN gnd Cval $ -I to Q
** Differential fets
M1 IP IN cs gnd NMOS l=0.35u w=15u
M2 IN IP cs gnd NMOS l=0.35u w=15u
** Bias fet - bias at Vdd -- too high?
Mb cs  vdd gnd gnd NMOS l=0.35u w=15u
** fets used as varactors
Mt1 vc IP vc gnd NMOS l=0.35u w=2u M=50
Mt2 vc IN vc gnd NMOS l=0.35u w=2u M=50
*--------------------------------
** Second oscillator section
** Low-Q resonator with Vdd at center tap of inductors
R1a_b QP ri_b 100k  $ These R's set the Q
R1b_b ri_b QN 100k
L1_b QP vdd 16.5nH
L2_b vdd QN 16.5nH
Cc1_b QP gnd Cval $ -Q to -I
Cc2_b QN gnd Cval $ -Q to I
** Differential fets
M1_b QP QN cs_b gnd NMOS l=0.35u w=15u
M2_b QN QP cs_b gnd NMOS l=0.35u w=15u
** Bias fet - bias at Vdd -- too high? 2nd in parallel
Mb_b cs_b  vdd gnd gnd NMOS l=0.35u w=15u
** fets used as varactors
Mt1_b vc QP vc gnd NMOS l=0.35u w=2u M=50
Mt2_b vc QN vc gnd NMOS l=0.35u w=2u M=50
*
*--------------------------------
* Differentiators Coupling transistors for quadrature
*
.param Cdiff=0.14p difMsize=50u
vidiff  dbias gnd 1.25
viqdiff vdcdif gnd 1.75
Midiff1 dQP dbias gnd gnd NMOS l=0.35u w=difMsize
Midiff2 dQN dbias gnd gnd NMOS l=0.35u w=difMsize
Midiff3 dIN dbias gnd gnd NMOS l=0.35u w=difMsize
Midiff4 dIP dbias gnd gnd NMOS l=0.35u w=difMsize
Cdiff1  dQP QP Cdiff
Cdiff2  dQN QN Cdiff
Cdiff3  dIN IN   Cdiff
Cdiff4  dIP IP   Cdiff
Mc_QP1 IP vdcdif dQP gnd NMOS l=0.35u w=difMsize
Mc_QN2 IN vdcdif dQN gnd NMOS l=0.35u w=difMsize
Mc_QN3 QP vdcdif dIN gnd NMOS l=0.35u w=difMsize
Mc_QP4 QN vdcdif dIP gnd NMOS l=0.35u w=difMsize
*--------------------------------
```

```
* Transient Analysis Test Bench
*
* stimulate oscillation with 2mA pulse
*iosc IP IN PULSE ( 0 2m .01n .01n .01n 10n 1u )
*.probe tran v(IP) v(IN)
*.print tran v(IP) v(IN)
*.TRAN .01n 10n
*------------------------------
* Harmonic Balance Test Bench
*
.sweepblock vtune_sweep
+ 0 5 0.2
+ 2 3 0.1
.HBOSC tones=1550e6 nharms=12
+ PROBENODE=IP,QN,4
+ sweep Vtune sweepblock=vtune_sweep
**
.phasenoise dec 10 100 1e7
.print phasenoise phnz
.probe phasenoise phnz
.print hb v(IP,IN) v(IP,IN)[1] v(QP,QN) v(QP,QN)[1]
.probe hb v(IP,IN) v(IP,IN)[1] v(QP,QN) v(QP,QN)[1]
.probe hb hertz[1][1]
*
* NMOS Device from MOSIS 0.35um Process
*
* BSIM3 VERSION 3.1 PARAMETERS
*
* DATE: Mar  8/00
* LOT: n9co                    WAF: 07
* Temperature_parameters=Default
*
.MODEL NMOS NMOS (                                  LEVEL   = 49
+VERSION = 3.1           TNOM    = 27            TOX     = 7.9E-9
+XJ     = 1.5E-7        NCH     = 1.7E17        VTH0    = 0.5047781
+K1     = 0.5719698     K2      = 0.0197928     K3      = 33.4446099
+K3B    = -3.1667861    W0      = 1E-5          NLX     = 2.455237E-7
+DVT0W   = 0               DVT1W   = 0             DVT2W   = 0
+DVT0   = 2.8937881     DVT1    = 0.6610934     DVT2    = -0.0446083
+U0    = 421.8714618   UA     = -1.18967E-10  UB     = 1.621684E-18
+UC     = 3.422111E-11   VSAT    = 1.145012E5    A0      = 1.119634
+AGS     = 0.1918651      B0      = 1.800933E-6    B1      = 5E-6
+KETA    = 3.313177E-3    A1       = 0              A2      = 1
+RDSW   = 984.149934    PRWG   = -1.133763E-3  PRWB   = -7.19717E-3
+WR     = 1           WINT    = 9.590106E-8   LINT    = 1.719803E-8
+XL     = -5E-8         XW      = 0           DWG     = -2.019736E-9
+DWB     = 6.217095E-9    VOFF    = -0.1076921     NFACTOR = 0
+CIT     = 0              CDSC    = 2.4E-4         CDSCD   = 0
```

```
+CDSCB   = 0           ETA0    = 0.0147171     ETAB    = -7.256296E-3
+DSUB    = 0.3377074     PCLM    = 1.1535622     PDIBLC1 = 2.946624E-4
+PDIBLC2 = 4.171891E-3   PDIBLCB = 0.0497942     DROUT   = 0.0799917
+PSCBE1  = 3.380501E9    PSCBE2  = 1.69587E-9     PVAG    = 0.4105571
+DELTA    = 0.01           MOBMOD   = 1               PRT      = 0
+UTE      = -1.5           KT1      = -0.11           KT1L     = 0
+KT2     = 0.022         UA1     = 4.31E-9       UB1     = -7.61E-18
+UC1      = -5.6E-11       AT       = 3.3E4           WL       = 0
+WLN      = 1             WW      = -1.22182E-15   WWN     = 1.1657
+WWL      = 0              LL       = 0               LLN      = 1
+LW       = 0              LWN      = 1               LWL      = 0
+CAPMOD   = 2             XPART   = 0.4           CGDO    = 3.73E-10
+CGSO    = 3.73E-10      CGBO    = 1E-11         CJ      = 8.988141E-4
+PB     = 0.8616985     MJ     = 0.3906381     CJSW    = 2.463277E-10
+PBSW    = 0.5072799     MJSW    = 0.1331717     PVTH0   = -0.0143809
+PRDSW   = -81.683425    WRDSW   = -107.8071189  PK2    = 1.210197E-3
+WKETA   = -1.00008E-3   LKETA   = -6.1699E-3    PAGS    = 0.24968
+AF       = 1.0            KF       = 1.0E-30      )
*
.END
```

The results of the analysis are displayed in Figure 5 on page 47, Table 6 on page 48 Figure 7 on page 49, and Figure 8 on page 50 using Custom WaveView for VCO waveforms, tuning curves, and phase noise response.

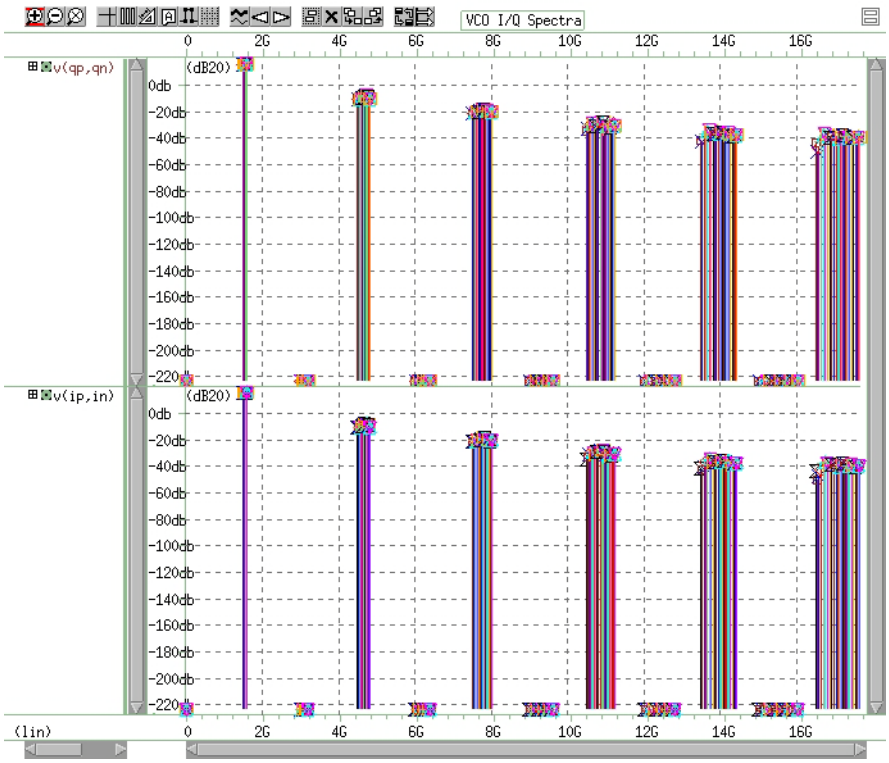


Figure 5 VCO Spectra Output

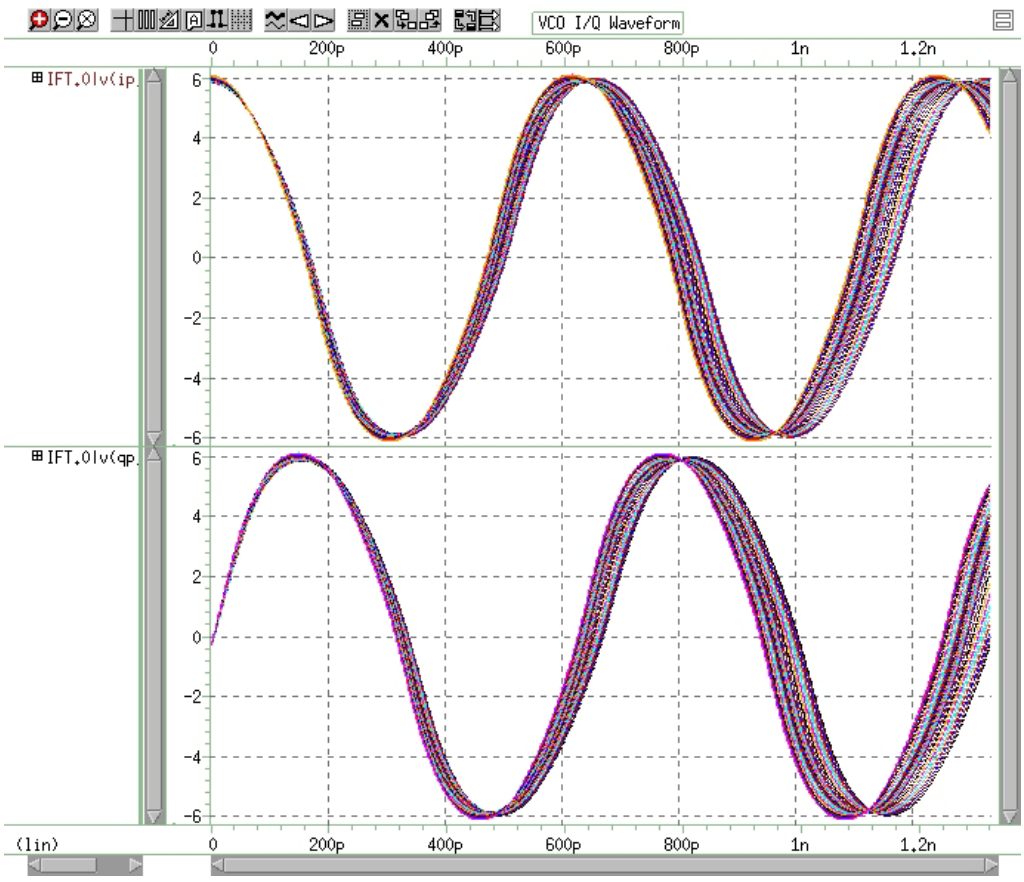


Figure 6 VCO Waveform Output

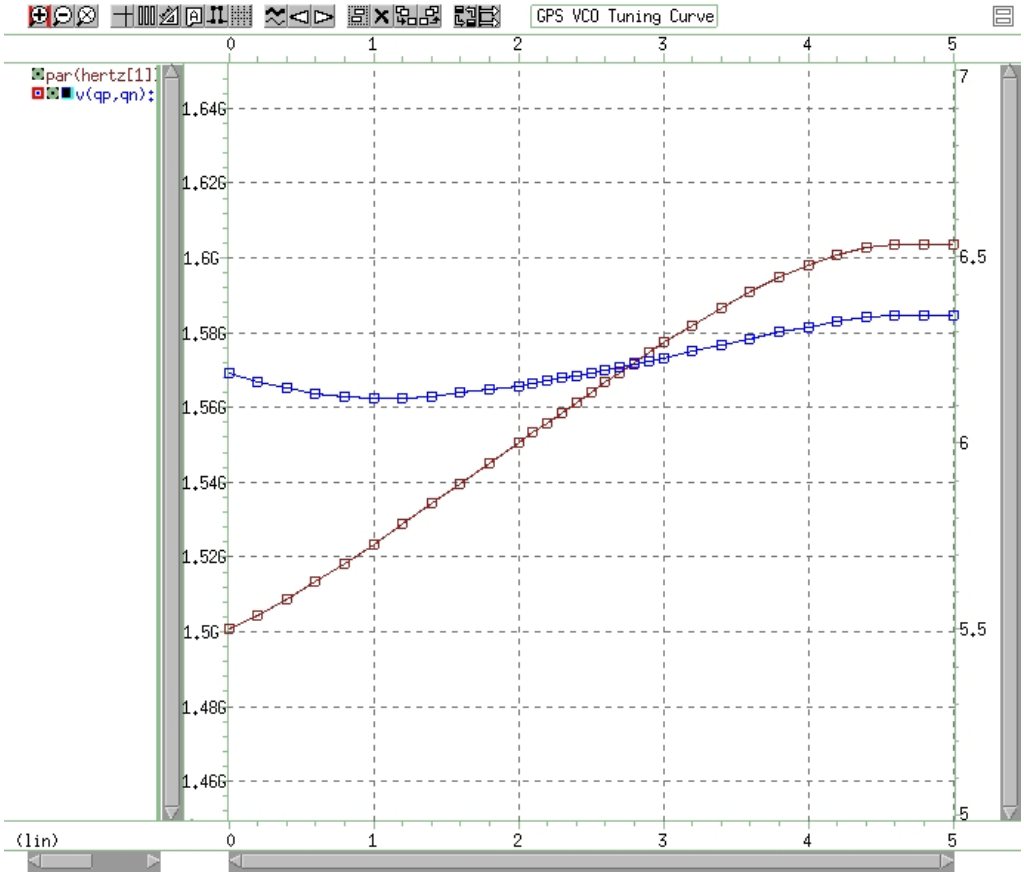


Figure 7 VCO Tuning Curves Output

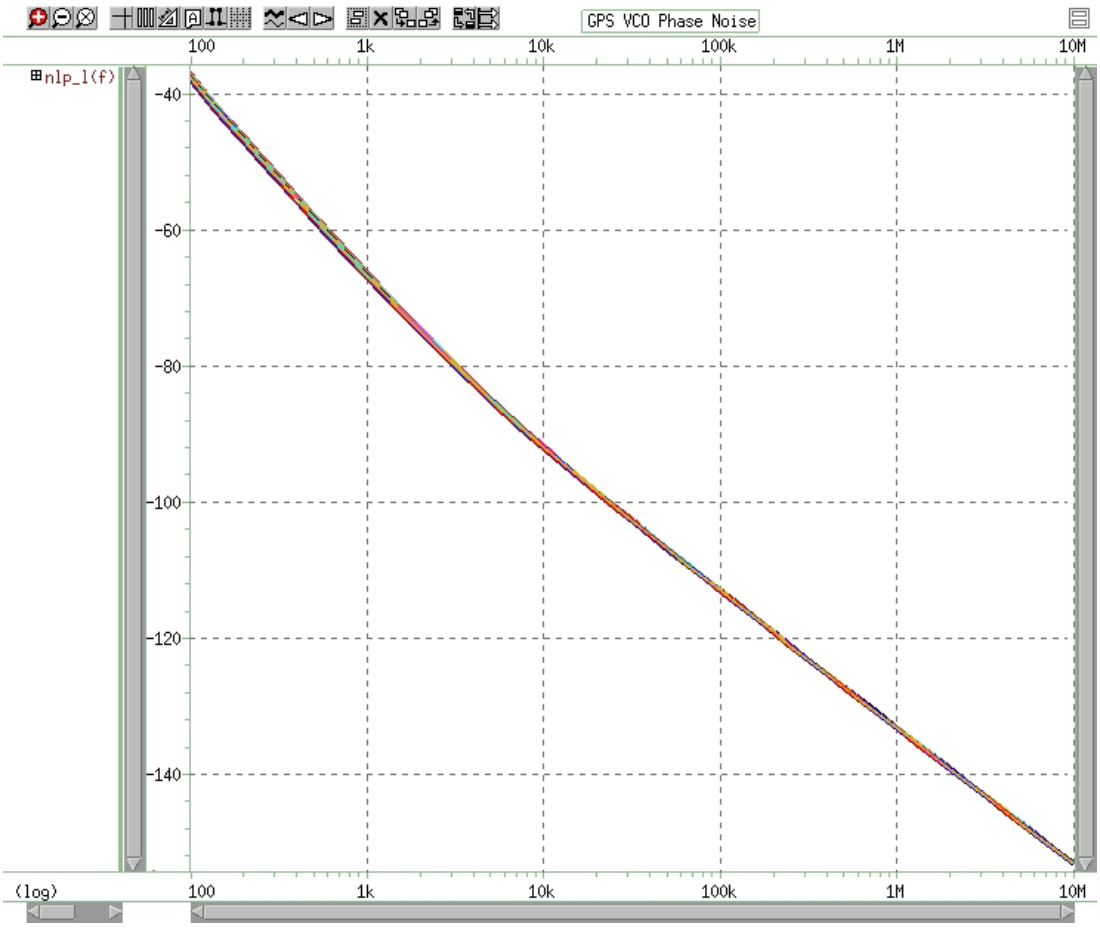


Figure 8 VCO Phase Noise Response

Example 6: Using Multi-Tone HB and HBAC Analyses for a Mixer

The example in this section shows how to use HSPICE RF to analyze a circuit driven by multiple input stimuli with different frequencies. Mixer circuits provide a typical example of this scenario: in this case, there might be two input signals (LO and RF), which are mixed to produce an IF output signal. In this case, HSPICE RF offers two options:

- Multi-tone HB analysis: specify the LO and RF base frequencies as two separate tones on the .HB command.
- Periodic AC analysis (HBAC): if one of the inputs is a small-signal, you can use a faster linear analysis to analyze its effect. For example, if a mixer's LO is a large signal, but RF is a small signal, a single-tone HB analysis using the LO frequency can be combined with HBAC in place of a 2-tone HB analysis.

To demonstrate both techniques, this example analyzes an ideal mixer built using behavioral elements. It is based on demonstration netlist *mix_tran.sp*, which is available in directory $*installdir*/demo/hspicerf/examples.

```
* Ideal mixer example: transient analysis
.OPTIONS POST
vlo lo 0 1.0 sin (1.0 0.5 1.0g 0 0 90)
rrf1 rf1 rf 1.0
g1 0 if cur='1.0*v(lo)*v(rf)' $ mixer element
c1 0 if q='1.0e-9*v(lo)*v(rf)' $ mixer element
rout if ifg 1.0
vctrl ifg 0 0.0
h1 out 0 vctrl 1.0 $ convert I to V
rh1 out 0 1.0
vrf rf1 0 sin (0 0.001 0.8GHz 0 0 114)
.tran 10p 10n
.opt sim_accuracy=100
.end
```

This example uses behavioral controlled current and charge sources to simulate a mixer. The LO signal is driven by a 0.5 Volt sinusoid at 1 GHz, and RF is driven by a 10mV signal at 800 MHz. The mixer output is the voltage at node out, v(out).

Two-tone HB Approach

To analyze this circuit using 2-tone HB, add:

- HB source for `LO`: add `HB 0.5 0 1 1` to the LO voltage source; this sets the amplitude to 0.5, no phase shift for the first harmonic of the first tone, which is 1 GHz.
- HB source for `RF`: add `HB 0.001 24 1 2` to the RF voltage source; this sets the amplitude to 0.001, 24 degrees phase shift for the first harmonic of the second tone (0.8 GHz).
- An `.HB` command specifying both tones: .hb tones=1g 0.8g nharms=6 3; only a small number of harmonics is required to resolve the signals.

The complete *mix_hb.sp* netlist for 2-tone HB analysis is then:

```
* Ideal mixer example: 2-tone HB analysis
.OPTIONS POST
vlo lo 0 1.0 sin (1.0 0.5 1.0g 0 0 90) HB 0.5 0 1 1
rrf1 rf1 rf 1.0
g1 0 if cur='1.0*v(lo)*v(rf)' $ mixer element
c1 0 if q='1.0e-9*v(lo)*v(rf)' $ mixer element
rout if ifg 1.0
vctrl ifg 0 0.0
h1 out 0 vctrl 1.0 $ convert I to V
rh1 out 0 1.0
vrf rf1 0 sin (0 0.001 0.8GHz 0 0 114) HB 0.001 24 1 2
.opt sim_accuracy=100
.hb tones=1g 0.8g nharms=6 3
.end
```

This example is available in directory $<installdir>/demo/hspicerf/examples.

HBAC Approach

To analyze this circuit using HBAC, start with the 2-tone HB analysis setup, and modify it as follows:

- Replace the `RF` HB signal with an HBAC signal: change `HB 0.001 24 1 2` to `HBAC 0.001 24`; this deactivates the source for HB and activates it for HBAC with the same magnitude and phase.
- Specify the frequency in the `.HBAC` command.
- Change the `.HB` command to single tone:

  ```
  .HB tones=1g nharms=6
  ```

  HBAC takes care of the second tone.
- Add a `.HBAC` command

```
.HBAC lin 1 0.8g 0.8g
```

This command runs an analysis at a single frequency point, 0.8 GHz. In general, HBAC analysis can sweep the RF frequency over a range of values.

The following is the complete *mix_hbac.sp* netlist for HBAC analysis of this simple mixer. This netlist also contains commands for performing periodic noise analysis. It is available in directory $*installdir*/demo/hspicerf/examples.

```
* Ideal mixer example: HBAC analysis
.OPTIONS POST
vlo lo 0 1.0 sin (1.0 0.5 1.0g 0 0 90)
+ HB 0.5 0 1 1
rrf1 rf1 rf 1.0
g1 0 if cur='1.0*v(lo)*v(rf)' $ mixer element
c1 0 if q='1.0e-9*v(lo)*v(rf)' $ mixer element
rout if ifg 1.0
vctrl ifg 0 0.0
h1 out 0 vctrl 1.0 $ convert I to V
rh1 out 0 1.0
vrf rf1 0 sin (0 0.001 0.8GHz 0 0 114)
+ HBAC 0.001 24
.opt sim_accuracy=100
.hb tones=1g nharms=6
.hbac lin 1 0.8g 0.8g
* Noise analysis
.hbnoise v(out) rrf1 lin 40 0.1g 4g
.print hbnoise onoise nf
.probe hbnoise onoise nf
.end
```

Comparing Results

After running all three netlists above, you will have generated 3 output files:

- mix_tran.tr0
- mix_hb.hb0
- mix_hbac.hb0

You can compare the results of the three analyses in Custom WaveView.

1. To run the netlists and start Custom WaveView, type:

```
hspicerf mix_tran.sp
hspicerf mix_hb.sp
hspicerf mix_hbac.sp
wv &
```

2. Use File > Import Waveform File and select the *mix_tran.tr0*, *mix_hb.hb0*, and *mix_hbac.hb0* files from the Open: Waveform Files dialog box. A histogram displays.
3. Select the v(out) signal from the *mix_hb.hb0* file in the signal browser. Double-click on the signal name or drag and drop the signal in the waveform. You should see a histogram similar to the one from the *mix_hb.hb0* file.
4. Convert the HB and HBAC histograms to time domain. In the waveform, right-click in the name area of the panel containing the signal v(out), left-click on the waveform label for v(out) from the *mix_hb.hb0* file. From the Panel menu, choose Signal 'v(out)' > To Time-Domain. To accept the defaults for range and interval, click OK in the Convert to Time domain window.
5. Repeat Step 4 for the v(out) signal from the *mix_hbac.hb0* file.
6. Use Waveview > New to open a new waveform.
7. Select the v(out) signal from the mix_tran.tr0 file in the signal browser. Double-click on the signal name or drag and drop the signal in the waveform.
8. Compare the time domain waveforms from the mix_hb.hb0 and mix_hbac.hb0 files with the time domain waveform from the mix_tran.tr0 file. In the file browser, click on IFT under derived waveforms. The signals 0|v(out) and 1|v(out) should appear in the signal browser. Select the 0|v(out) and 1|v(out) signals and drag and drop them in the waveform. All three time domain signals should be displayed in the same panel. The three signals are almost indistinguishable from each other.

You can also use HBAC to perform noise analysis on RF circuits by using the `.HBNOISE` command, which is included in the *mix_hbac.sp* netlist.

- The `.HBNOISE` command invokes noise analysis, identifying an output node where the noise is measured, an input noise source (in this case, rrf1) which serves as a reference for noise figure computation, and a frequency sweep for the noise analysis.
- The `.PRINT` and `.PROBE hbnoise` commands instruct HSPICE RF to save the output noise and noise figure at each frequency in the *mix_hbac.printpn0* and *mix_hbac.pn0* output files.

This ideal mixer is noiseless, except for the resistors at the input and output.

The mix_hbac.lis file contains detailed data on the individual noise source contributions of the resistors. You can view *mix_hbac.printpn0* to see the output noise and noise figure at each frequency. In WaveView, you can view

mix_hbac.pn0 to plot the output noise and noise figure data as a function of frequency.

Example 7: Using Shooting Newton Analysis on a Driven Phase Frequency Circuit and a Ring Oscillator

Introduction

While the Harmonic Balance (HB) algorithm represents the circuit's voltage and current waveforms as a Fourier series (a series of sinusoidal waveforms), the Shooting Newton (SN) algorithm provides analysis capability for digital logic circuits and RF components that require steady-state analysis, but operate with waveforms that tend to be square instead of sinusoidal.

Simple examples of using the Shooting Newton analysis functions are presented in this section:

- Driven Phase Frequency Detector Example
- Ring Oscillator Example

Shooting Newton Analysis Setup

To set up a time-domain, steady-state analysis, the HSPICE input netlist must contain:

- A `.SN` command to activate the analysis. The `.SN` command specifies:
- The expected period of the steady-state waveforms, which must match the period of any input waveforms. The period is specified in time domain units (seconds). Alternatively, you may specify a frequency in Hz.
- A time resolution, which is analogous to the transient analysis (`.TRAN`) command's TSTEP parameter and will affect the time step size selection. It also affects the number of frequency terms used in small-signal analyses,

such as periodic AC or noise analysis. The time resolution is typically specified in seconds but, alternatively, may be specified in the frequency domain as a number of harmonics.

- A transient initialization time that is used by HSPICE RF to run a basic transient simulation of this length before attempting Newton-Raphson iterations to converge on a steady-state solution. This parameter is optional. If it is not specified, the specified period is used as the initialization time. The initial transient analysis is used for circuit stabilization before the steady state solution is found. Larger initialization values typically result in convergence that is more robust.
- For oscillator circuits, a `.SNOSC` command is used to activate the analysis. The `.SNOSC` command specifies:
- The approximate frequency of oscillation specified either as a frequency (in Hertz) or as the time domain period.
- The number of high frequency harmonics. Alternatively, a time resolution in seconds can be specified.
- A transient initialization time that is used by HSPICE RF to run a basic transient simulation of this length before attempting Newton-Raphson iterations to converge on a steady state solution. This parameter is optional. If it is not specified, the period of the specified frequency of oscillation is used as the initialization time. For oscillators we recommend specifying a transient initialization time since the default initialization time is usually too short to effectively stabilize the circuit.
- A node at which to probe for oscillation conditions.
- If the tuning curve of a VCO is to be analyzed, the optional parameter MAXTRINITCYCLES can be specified.
- One or more signal sources for driving the circuit in SN analysis, if the circuit is driven. In the case of autonomous oscillator analysis, no signal source is required. The sources are required to be time domain sources and must match the period specified in the `.SN` command.

Driven Phase Frequency Detector Example

This example demonstrates the Shooting Newton-based analysis of a driven phase-frequency detector. Extracted portions of the input file are presented below. The complete *phasefreqdet.sp* input file for this example is located in the

following directory:
$installdir/demo/hspicerf/examples/

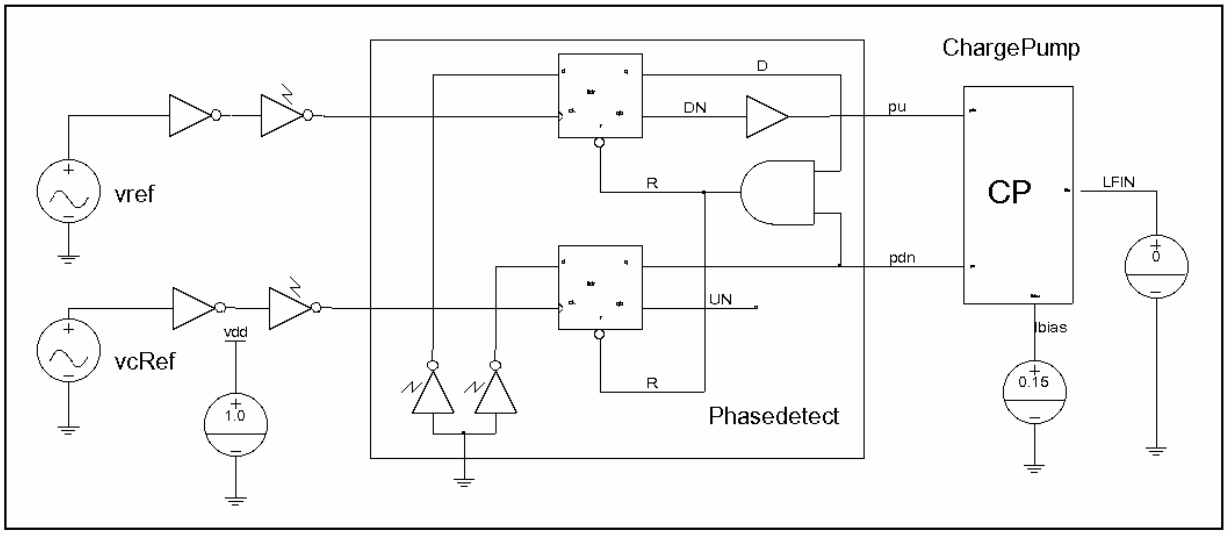


Figure 9 Driven Phase Frequency Detector

```
* Phase Frequency Detector Example
*
.global vdd gnd
.options wl post

* DC sources
vsup vdd 0 DC 1.0

* Reference signal (sine wave)
vref xin gnd DC 0 sin( 0.5 0.5 0.5g)
* Input buffers (square up Ref sine wave)
xfin1 xin fin1 inv
xfin2 fin1 FIN inv3

* Compare signal (sine wave)
vcRef cin gnd DC 0 sin( 0.5 0.5 0.5g 0.0 0.0 phase)
$ phase shift
* Input buffers (square up compare sine wave)
xcfin1 cin cfin1 inv
xcfin2 cfin1 cFIN inv3
*
** Phase/frequency detector
xPFD cFIN FIN pdn pu phasedet
** Chargepump
xCP LFIN Ibias pdn pu chargepump
** Bias voltage
vIbias Ibias gnd 0.15     $ Sets charge pump bias!
Vload LFIN 0 0

* Harmonic Balance Test Bench
*
.param phase=0.0     $ phase shift in degrees

.opt snaccuracy=30
.SN tres=10p period=2n SWEEP phase POI 5 0.0 22.5 45.0 67.5 90.0
.SNNOISE V(pu,pdn) Vref
+ DEC 21 100 10MEG      $ offset frequency sweep
+ [0,1]                 $ Take low frequency noise
*
.probe sn v(fin) v(cfin) v(pu) v(pdn) v(lfin) i(vibias)
.print snfd i(vload) i(vload) [0]
.probe snfd i(vload) i(vload) [0]
.probe SNNOISE  ONOISE
.print SNNOISE  ONOISE
.end
```

During the analysis, the phase of the input signal is swept between 0 degrees and 90 degrees using five equally spaced steps. This enables you to measure the phase detector gain at the output load. The `.SN` command syntax specifies specifies the expected period of the steady-state waveforms (2nS) and the time resolution (10pS) in the time domain.

A periodic, time-varying AC noise analysis based on the Shooting Newton algorithm is performed using the `.SNNOISE` command. The `.SNNOISE` analysis requires an output node (v(pu, pdn)) where the noise is to be measured, an input noise source (Vref) which serves as the reference for the noise computation and, a frequency sweep for the noise analysis. Optionally, an index term can be defined. The index term specifies the output frequency band at which the noise is evaluated. For this case, you will evaluate the low frequency noise of the phase frequency detector.

The time-domain signals v(cfin), v(fin), v(pu) and v(pdn), and v(lfin) are probed. The gain of the phase frequency detector can be found by probing the frequency domain value of v(lfin) at DC (frequency indices 0).

During the simulation, the simulation status is displayed on the screen. In addition to the screen display, more detailed status, cpu time, and memory usage information is also written to the *phasefreqdet.lis* file.

Viewing Results in Custom WaveView

You can view the time-domain, *phasefreqdet.sn0* file, the frequency domain, *phasefreqdet.snf0* file, and the noise results, *phasefreqdet.snpn0* file in Custom WaveView:

1. Enter **wv** *at the prompt* to start Custom WaveView.
2. The time domain results are used to show the input and output waveforms of the phase frequency detector. Use File > Import Waveform File to open the *phasefreqdet.sn0* file.
3. Select the input signals, v(cfin) and v(fin), and the output signals, v(pu) and v(pdn) from the signal browser. Drag and drop the selected signals in the waveform. Figure 10 on page 60 shows the waveforms for the selected input and output signals.

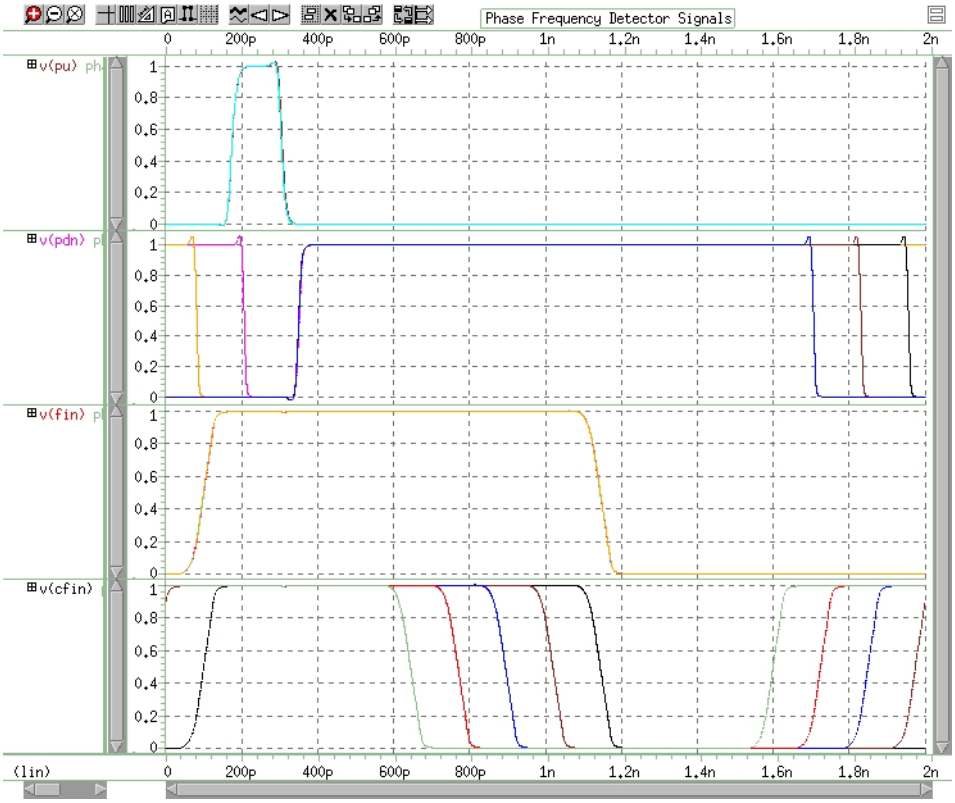


Figure 10 Phase Frequency Detector Signals

4. The frequency domain results are used to show the gain of the phase frequency detector. Use File > Import Waveform File to open the *phasefreqdet.snf0* file.
5. Use Waveview > New to open a new waveform.
6. Select the signal i(vload):(0) from the signal browser. Drag and drop the signal i(vload):(0) in the waveform. The signal is the DC component of the i(vload) signal spectrum. By default, the magnitude and phase of the load current are plotted. To measure the gain of the phase frequency detector verses phase, only the magnitude is required. Figure 11 on page 61 shows the gain of the phase frequency detector.

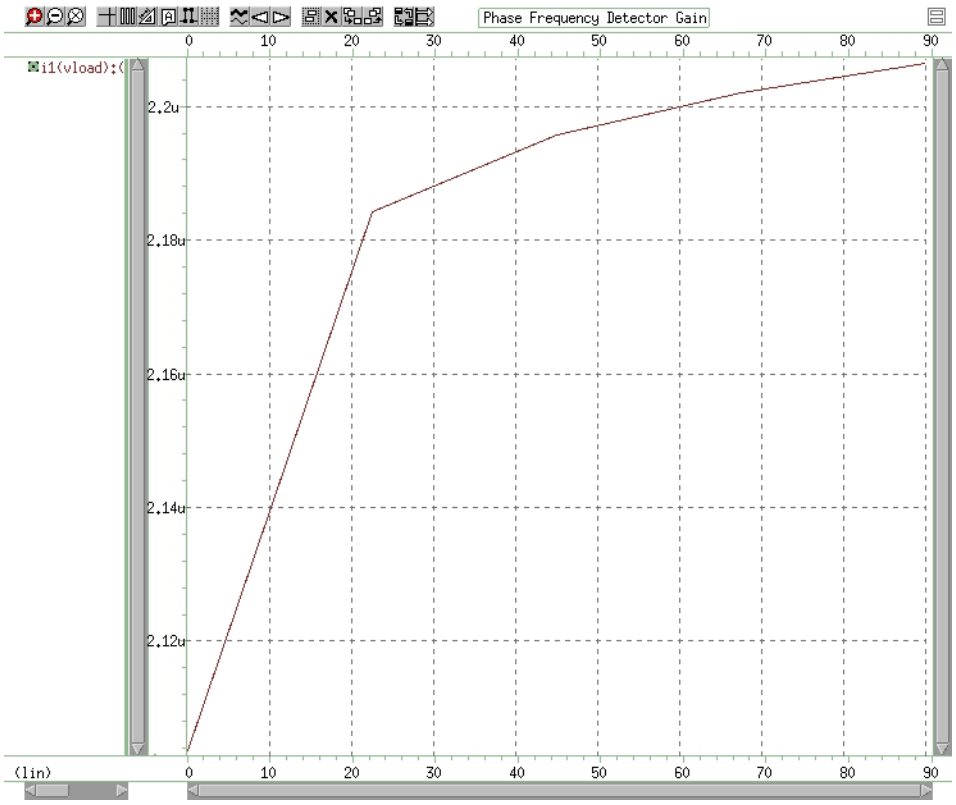


Figure 11 Phase Frequency Detector Gain

7. Next, plot the output noise of the phase frequency detector. Use File > Import Waveform File to open the *phasefreqdet.snpn0* file.*phasefreqdet.snpn0* file.
8. Use Waveview > New to open a new waveform.
9. Select the signal onoise() from the signal browser. Drag and drop the signal onoise() in the waveform. The noise results are shown in Figure 12 on page 62. This displays the noise at the output, v(pu, vpd) at each phase value swept in the `.SN` command.
10. Change the X-axis scale to log by left clicking on the X-axis and selecting Log Scale from the X-Axis menu.

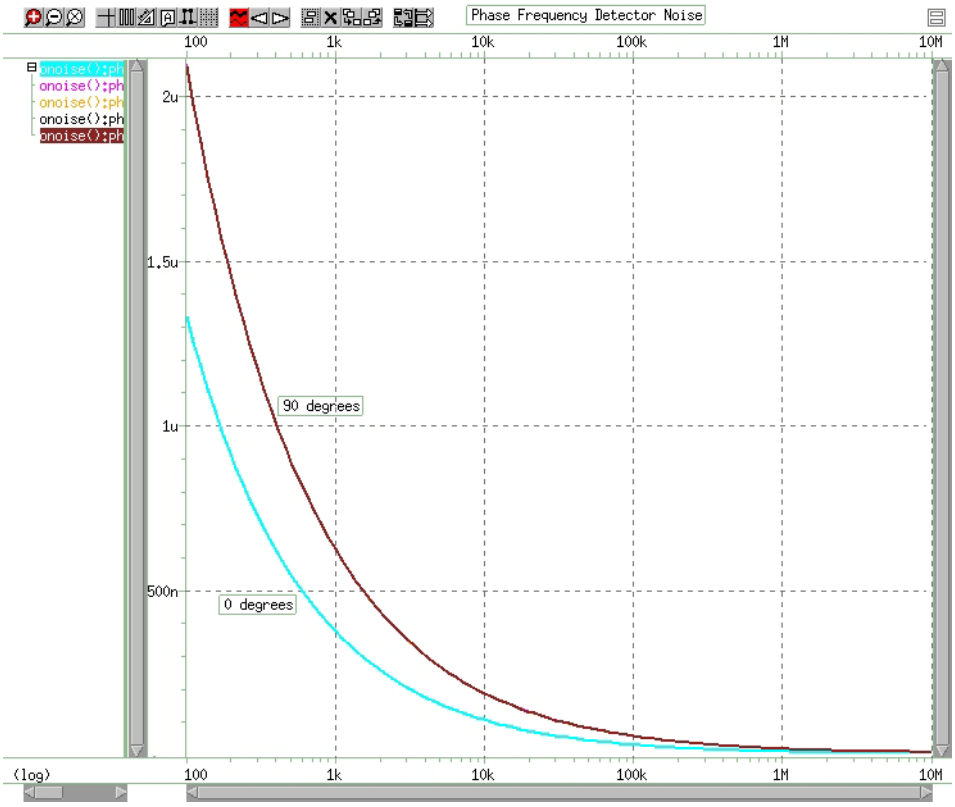


Figure 12 Phase Detector Output Noise

Ring Oscillator Example

The Shooting Newton algorithm provides fast and effective analysis for ring oscillators. The *ringoscSN.sp* input file for this example is located in the following directory:

$installdir/demo/hspicerf/examples/

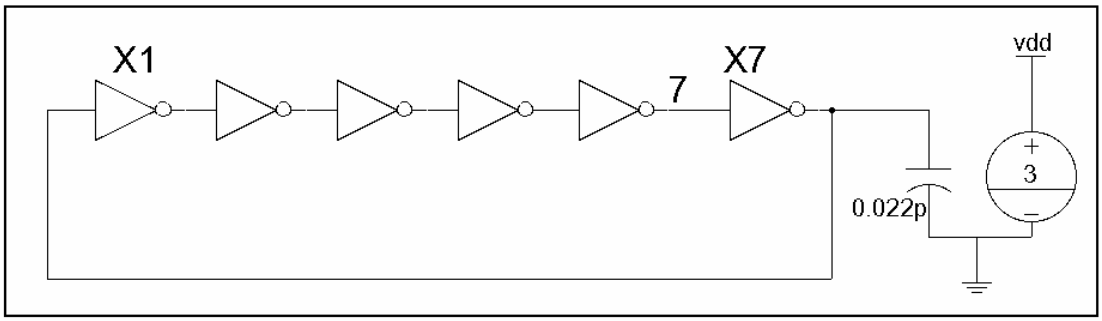


Figure 13 Ring Oscillator

```
.title ringosc

.subckt inv in out vdd
mn1 out in 0 0 nmos l=0.25u w=2u
mp1 out in vdd vdd pmos l=0.25u w=6u
.ends

vdd vdd 0 3
x1 1 2 vdd inv
x2 2 3 vdd inv
x3 3 4 vdd inv
x4 4 5 vdd inv
x5 5 6 vdd inv
x6 6 7 vdd inv
x7 7 1 vdd inv
c1 1 0 0.022p
.ic v(1)=3
.options post
.options snaccuracy=50

.snosc tones=335meg nharms=10 oscnode=1 trinit=10n
.phasenoise v(7) dec 10 100 10meg

.probe sn v(7)
.probe snfd v(7)
.print phasenoise phnoise v(7)
.probe phasenoise phnoise v(7)

.end
```

This analysis finds the oscillation frequency of the ring oscillator. Since the circuit is an oscillator, no input source is required. The oscillator is started by setting an initial condition at the input of the ring (node 1). In the `.SNOSC`

command, the node that the analysis will probe for oscillation conditions is specified, as well as the approximate frequency of oscillation. The number of harmonics to include in the analysis is specified, as well.

The phase noise characteristics of the oscillator are analyzed by using the `.PHASENOISE` command. The `.PHASENOISE` command requires that an output node, pair of nodes, or a two-port element and a frequency sweep be specified. The frequency sweep is used to calculate the phase noise analysis at the specified offset frequencies, measured from the oscillation carrier frequency. For this example phase noise analysis, the default Nonlinear Perturbation (NLP) method is used.

The signals v(7) will be probed in both the frequency and time domain. The measure statement is used to measure the fundamental frequency of the oscillator.

Simulation Status Output

During the simulation, the simulation status is displayed on the screen. In addition to the screen display, more detailed status, cpu time and memory usage information is also written to the *ringoscSN.lis* file.

Viewing Results in Custom WaveView

You can view the time-domain, *ringoscSN.sn0* file, the frequency domain, *ringoscSN.snf0* file, and the phase noise, *ringoscSN.snpn0* file in Custom WaveView.

1. Enter **wv** *at the prompt* to start Custom WaveView.
2. Use File > Import Waveform File to open the *ringoscSN.sn0* file.
3. Select the signal v(7) from the signal browser. Drag and drop the signal v(7) to the right side of waveform so that panels are opened in row / column format. The time domain trace shown at the right side of Figure 14 on page 65.
4. Use File > Import Waveform File to open the *ringoscSN.snf0* file.
5. Select the signal v(7) from the signal browser. Drag and drop the signal v(7) to the right side of waveform so that panels are opened in row / column format. The frequency domain spectrum is shown at the left side of Figure 14 on page 65.

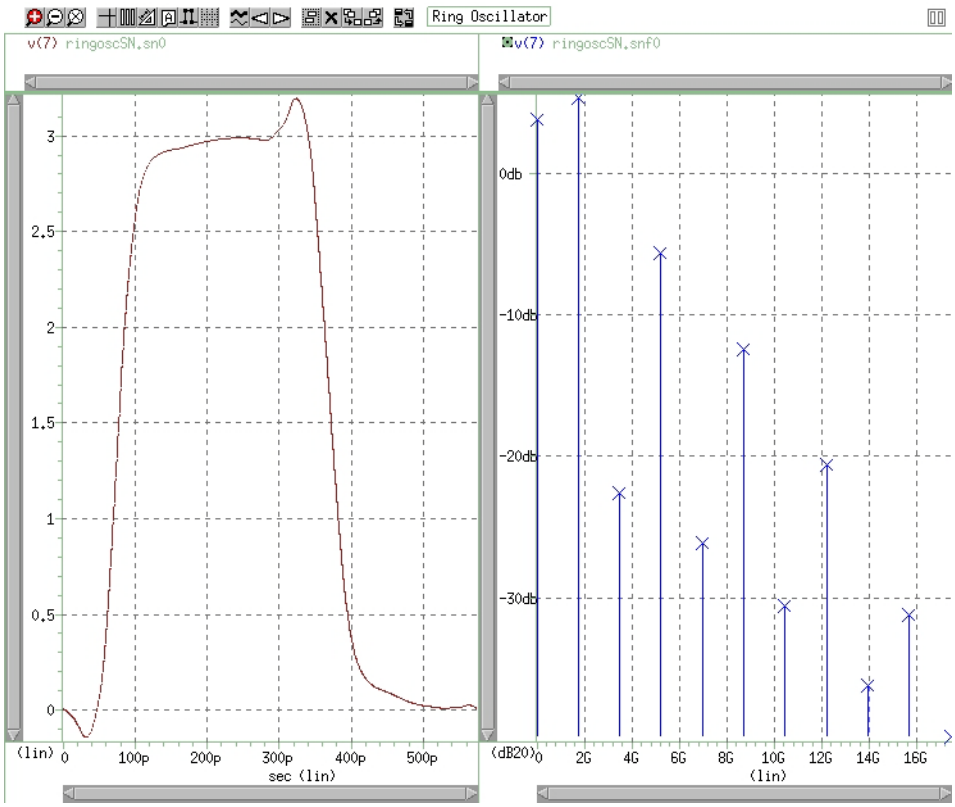


Figure 14 Ring Oscillator Output

6. Use File > Import Waveform File to open the *ringoscSN.snpn0* file.
7. Use Waveview > New to open a new waveform.
8. Select the signal nlp_I(f) from the signal browser. Drag and drop the signal nlp_I(f) signal in the waveform. Figure 15 on page 66 shows the resulting phase noise results for the oscillator.

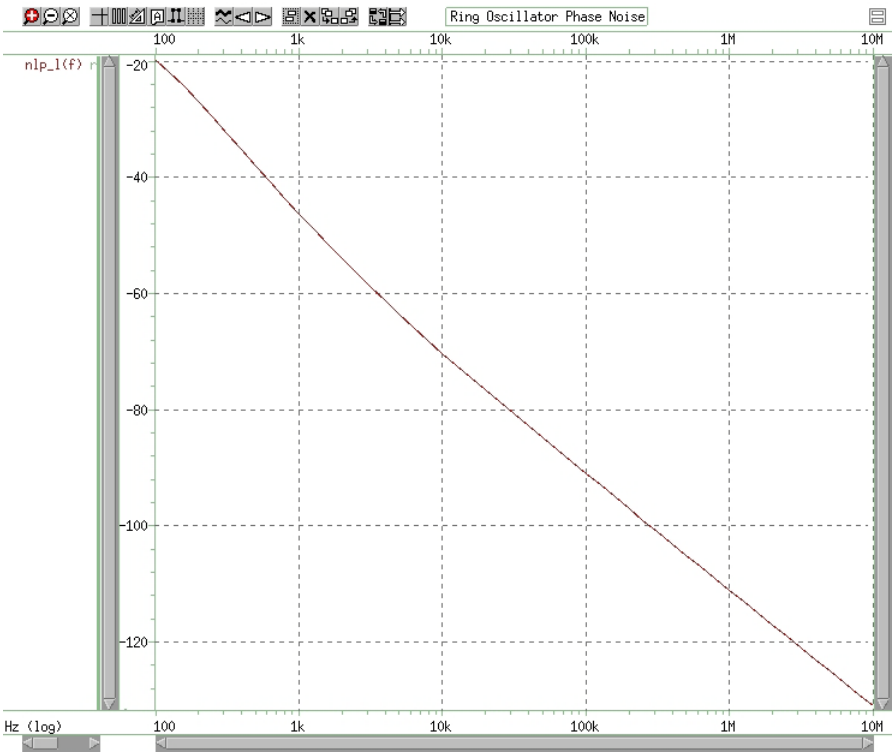


Figure 15 Ring Oscillator Phase Noise

Other Shooting Newton Analyses

The following Shooting Newton Analyses are also supported by HSPICE RF.

- `.SNFT` is equivalent to the `.FFT` command in transient (`.TRAN`) analysis. `.SNFT` uses Fourier transform to represent a time domain signal in the frequency domain. For more information, see Shooting Newton with Fourier Transform (.SNFT).
- `.SNAC` is used to perform a linear analysis of a driven (or nonautonomous) circuit, where the linear coefficients are modulated by a periodic, steady-state signal. The functionality is similar to the `.HBAC` command. For more information, see Shooting Newton AC Analysis (.SNAC).
- `.SNXF` is used to calculate transfer functions from an arbitrary number of small signal sources to a designated output in a circuit under periodic steady state conditions. For more information, see Shooting Newton Transfer Function Analysis (.SNXF).

RF Demonstration Input Files

The following is a listing of shipped demonstration files for illustrating HSPICE RF functionality. All of these example files are available at:

$*installdir*/demo/hspicerf/examples

File Name	Description
acpr.sp	Envelope simulation example
bjt.inc	Transistor model library used by osc.sp
cmos49_model.inc	Transistor model library used by example circuits
cmos90nmWflicker.lib	Transistor model library used by phasefreqdet.sp
gpsvco.sp	Oscillator and Phase Noise analysis example
gsmlna.sp	LNA Linear analysis example
gsmlnaIP3_A.sp	3rd order intercept point example
mix_hb.sp	Mixer HB analysis example
mix_hbac.sp	MIxer HBAC analysis example
mix_snac.sp	Mixer Shooting Newton AC example
mix_tran.sp	Mixer transient analysis example
osc.sp	Oscillator tuning curve and phase noise analysis example
pa.sp	Power amplifier HB analysis example
pfdcpGain.sp	Shooting Newton analysis example
phasefreqdet.sp	Shooting Newton and noise analysis example
ringoscSN.sp	Shooting Newton and Phase Noise analysis example
tsmc018.m	Transistor model library used by ringoscSN.sp

4

Testbench Elements

Describes the syntax for the specialized elements supported by HSPICE RF for high-frequency analysis and characterization.

Elements are local and sometimes customized instances of a device model specified in your design netlist. For discussion of the syntax for the basic elements (passive, multi-linear, port, and active available to both HSPICE RF and HSPICE), see Chapter 8, Elements in the *HSPICE User Guide: Simulation and Analysis*. Examples for HSPICE RF are noted in that chapter.

For descriptions of the standard device models on which elements (instances) are based, see the HSPICE Reference Manual: Elements and Device Models and the HSPICE Reference Manual: MOSFET Models. For signal integrity applications see the HSPICE User Manual: Signal Integrity.

HSPICE RF supports several specialized elements for high-frequency analysis and characterization.

The following topics are discussed in these sections.

- Values for Elements
- Ideal Transformer Format in HSPICE RF
- DC Block and Choke Elements
- Multi-Terminal Linear Elements
- Port Element
- Steady-State Voltage and Current Sources
- Steady-State HB Sources
- Phase Differences Between HB and SIN Sources
- Behavioral Noise Sources
- Function Approximations for Distributed Devices

- Complex Signal Sources and Stimuli
- SWEEPBLOCK in Sweep Analyses
- Clock Source with Random Jitter
- References

Values for Elements

HSPICE RF accepts equation-based resistors and capacitors. You can specify the value of a resistor or capacitor as an arbitrary equation, involving node voltages or variable parameters. Unlike HSPICE, you cannot use parameters to indirectly reference node voltages in HSPICE RF.

Ideal Transformer Format in HSPICE RF

The ideal transformer format simplifies modeling of baluns. Previously, baluns were modeled using mutual inductors (K elements) with the `IDEAL` keyword. Multiple L and K elements were needed for a given balun model. The ideal transformer model allows modeling of a balun using a single L element.

In the ideal transformer format, no absolute inductance or reluctance values are specified. Instead, the transformer's coupling characteristics are specified using inductor number-of-turns values. The behavior of the ideal transformer depends on ratios of the inductors' number of turns.

Syntax

`Lxxx` *`n1p n1n ... nNp nNn`* `TRANSFORMER_NT=(`*`nt1, ... , ntN`*`)`

Parameter	Description
Lxxx	Inductor element name. Must begin with L, followed by up to 1023 alphanumeric characters.
n1p n1n ... nNp nNn	Positive and negative terminal node names. The number of terminals must be even. Each pair of reports represents the location of an inductor.

Parameter	Description
TRANSFORMER_NT	Number of turns values. These parameters must match the number of inductors.

The ideal transformer element obeys the standard ideal transformer equations:

$$\frac{v_1}{nt_1} = \frac{v_2}{nt_2} = \ldots = \frac{v_N}{nt_N}$$

$$i_1 nt_1 + i_2 nt_2 + \ldots + i_N nt_N = 0$$

Example

```
L1 1 0 0 2 3 0 transformer_nt=(1,2,2)
```

DC Block and Choke Elements

In HSPICE RF, you can specify an INFINITY value for capacitors and inductors to model ideal DC block and choke elements. The following input syntax is for the DC block (ideal infinite capacitor):

Syntax

C*xxx node1 node2* [C=] INFINITY [IC=*val*]

HSPICE RF does not support any other capacitor parameters for DC block elements, because HSPICE RF assumes that the infinite capacitor value is independent of temperature and scaling factors. The DC block acts as an open circuit for all DC analyses. HSPICE RF calculates the DC voltage across the circuit's nodes. In all other (non-DC) analyses, a DC voltage source of this value represents the DC block (that is, HSPICE RF does not then allow dv/dt variations).

The following input syntax is for the Choke (ideal infinite inductor):

Syntax

L*xxx node1 node2* [L=] INFINITY [IC=*val*]

HSPICE RF does not support any other inductor parameters, because HSPICE RF assumes that the infinite inductance value is independent of temperature and scaling factors. The choke acts as a short circuit for all DC analyses. HSPICE RF calculates the DC current through the inductor. In all other (non-

DC) analyses, a DC current source of this value represents the choke (that is, HSPICE RF does not then allow di/dt variations).

Multi-Terminal Linear Elements

A multi-terminal linear element such as a transmission line is a passive element that connects any two conductors at any distance apart. One conductor sends the input signal through the transmission line, and the other conductor receives the output signal from the transmission line. The signal is voltage between the conductors that is transmitted from one end of the pair to the other end.

Scattering Parameter Data Element

All HSPICE and HSPICE RF analyses can use the S-element. For more information about S-parameters, see S-parameter Modeling Using the S-element in the *HSPICE Signal Integrity Guide*.

Frequency-Dependent Multi-Terminal (S-element)

When used with the generic frequency-domain model (`.MODEL SP`), an S-element is a convenient way to describe the behavior of a multi-terminal network.

The S-element describes a linear time-invariant system, and provides a series of data that describe the frequency response of the system. The S-element is particularly useful for high-frequency characterization of distributed passive structures. A common use of the S-element is in microwave circuits such as spiral inductors, because electronic devices in this frequency domain no longer act as they do in low frequencies. In this case, distributed system parameters must be considered. See the example below for an application of the state space stamping to generate a frequency invariant modified nodal analysis (NMA) matrix from frequency-dependent characteristics using the Shooting Newton (.SN) algorithm.

For scattering parameter element and model syntax, see S-element Syntax and S Model Syntax in the *HSPICE User Manual: Signal Integrity*.

Example

The following netlist and data file (*test.rfm*) show how the S-element “S1” uses the “STAMP=YSTS” configuration which invokes the state space stamping to

generate a frequency invariant modified nodal analysis (NMA) matrix from frequency-dependent characteristics. This stamping method allows the Shooting-Newton algorithm (.SN) to obtain the steady state. Note that unless RFM file input is given, the S-element first applies the rational function approximation (equivalent behavior to RATIONAL_FUNCTION=1) to the original S-parameters in order to generate the state space stamping.

```
======= main netlist =======
*** .SN with s-element example
P1 n1 gnd port=1 dc=1v ac=1v pulse(1 0 1n 1n 1n 10n 20n)
P2 n2 gnd port=2 dc=1v ac=1v pulse(1 0 1n 1n 1n 10n 20n)

S1 n1 n2 0 mname=s_model
.model s_model S n=2
+ rfmfile='test.rfm'
+ STAMP=YSTS
.SN tone=0.05Ghz nharms=32
.option post accurate
.end
```

The following is from the *.lis* file for this netlist.

```
======= rational function matrix file (test.rfm) ======
VERSION 200600 NPORT 2 MATRIX_TYPE Y SYMMETRIC PRECFAC 0.75 Z0 50 50
BEGIN 1 1

BEGIN_REAL 9
DC 2.10290261e-02
2.80562648113e+07  1.791888661818e+00
1.36806220992e+08 -5.313505935943e+01
1.16867967247e+09  2.840375731037e+06
1.23552099406e+09 -4.257158329976e+06
1.92568095149e+09  3.038955064913e+06
4.15005808751e+09 -8.058749095413e+06
1.00149288271e+10  3.846931398394e+06
2.27536895845e+10  1.702938150800e+05
3.54118199282e+10 -1.243885701867e+07

BEGIN_COMPLEX 5
5.53251427579e+05  1.28282249537e+06 -3.17377193705e-03 -
1.20935639131e-03
2.39642428296e+09  1.39710928734e+08 -1.99538130185e+07 -
6.93072640638e+07
2.41275272760e+09  4.88535891322e+09  2.92904966609e+04
4.08311621367e+04
9.49575839142e+08 -2.82753080087e+10 -1.69178467311e+05 -
1.42790736653e+04
    3.74702282735e+10 2.26461714292e+1
6.18960971035e+06  2.73309486084e+05 END BEGIN 2 2 DC 2.10290261e-
02
```

```
BEGIN REAL 9
2.80562648113e+07  1.79188866181e+00
1.36806220992e+08 -5.31350593594e+01
1.16867967247e+09  2.84037573103e+06
1.23552099406e+09 -4.25715832997e+06
1.92568095149e+09  3.03895506491e+06
4.15005808751e+09 -8.05874909541e+06
1.00149288271e+10  3.84693139839e+06
2.27536895845e+10  1.70293815080e+05
3.54118199282e+10 -1.24388570186e+07

BEGIN_COMPLEX 5
5.53251427579e+05  1.28282249537e+06 -3.17377193705e-03 -
1.20935639131e-03
2.39642428296e+09  1.39710928734e+08 -1.99538130185e+07 -
6.93072640638e+07
2.41275272760e+09  4.88535891322e+09  2.92904966609e+04
4.08311621367e+04
9.49575839142e+08 -2.82753080087e+10 -1.69178467311e+05 -
1.42790736653e+04  3.74702282735e+10  2.26461714292e+10
6.18960971035e+06  2.73309486084e+05
END
```

Port Element

The port element identifies the ports used in LIN analysis and behaves as either a noiseless impedance or a voltage source in series with the port impedance for all other analyses (DC, AC, or TRAN). Each port element requires a unique port number. If your design uses *N* port elements, your netlist must contain the sequential set of port numbers, 1 through *N*. For example, in a design containing 512 ports, you must number each port sequentially, 1 to 512.

Each port has an associated system impedance, `zo`. If you do not explicitly specify the system impedance, the default is 50 ohms.

- You can use this element as a pure terminating resistance or as a voltage or power source.
- You can use the `RDC`, `RAC`, `RHB`, `RHBAC`, and `RTRAN` values to override the port impedance value for a particular analysis.

The port element accepts transient waveforms AM, EXP, PULSE, PWL, SFFM, SIN, LFSR, and, for signal integrity usage, the PAT source (see Pattern Source).

Port Element Syntax

```
Pxxx p n port=portnumber
+ $ **** Port Impedence ****
+ [Z0=val]
+ $ **** Voltage or Power Information ********
+ [DC mag] [AC mag phase] [HBAC mag phase]
+ [HB mag phase harm tone modharm modtone]
+ [transient_waveform] [ENCODE=DW8B10B] [RD_INIT=0|1]
+ [TRANFORHB=[0|1]] [DCOPEN=[0|1]]
+ $ **** Power Switch ********
+ [power=[0|1|2|W|dbm]]
+ $ **** Source Impedance Overrides ********
[RDC=val] [RAC=val]
+ [RHBAC=val] [RHB=val] [RTRAN=val]
```

Parameter	Description
port=portnumber	The port number. Numbered sequentially beginning with 1 with no shared port numbers.
z0=*val*	Port impedance (Ohms). (Default: 50). Sets port characteristic impedance used for .LIN analysis, sets port termination impedance for other analyses, and also sets source impedance when the port element is used as a signal source.
DC *mag*	DC voltage or power source value.
AC *mag phase*	AC voltage or power source value.
HBAC *mag phase*	(HSPICE RF) HBAC voltage or power source value.

Parameter	Description
HB *mag* p*hase harm tone modharm modtone*	(HSPICE RF) HB voltage, current, or power source value. Multiple HB specifications with different harm, tone, modharm, and modtone values are allowed. ▪ phase is in degrees ▪ harm and tone are indices corresponding to the tones specified in the .HB statement. Indexing starts at 1 (corresponding to the first harmonic of a tone). ▪ modtone and modharm specify sources for multi-tone simulation. A source specifies a tone and a harmonic, and up to 1 offset tone and harmonic (modtone for tones and modharm for harmonics). The signal is then described as: V(or I)=mag*cos(2*pi* (harm*tone+modharm*modtone)*t + phase)
transient_waveform	(Transient analysis) Voltage or power source waveform. Any one of waveforms: AM, EXP, PULSE, PWL, SFFM, SIN, or PRBS. Multiple transient descriptions are not allowed.
ENCODE=DW8b10b	Keyword to specify 8b/10b encoding.
RD_INIT=0|1	Initial value of Running Disparity. The one bit memory that recalls the bias of the last unbalanced code word is called the Running Disparity.1: Specifies that a Running Disparity value of zero is synonymous with negative Running Disparity (?). ▪ 0: Specifies that a Running Disparity value of one is synonymous with negative Running Disparity (-) ▪ 1: Specifies that a Running Disparity value of one is synonymous with positive Running Disparity (+).

Parameter	Description
TRANFORHB=[0|1]	▪ (HSPICE RF) 0 (default): The transient description is ignored if an HB value is given or a DC value is given. If no DC or HB value is given and TRANFORHB=0, then HB analysis treats the source as a DC source, and the DC source value is the time=0 value. ▪ 1: HB analysis uses the transient description if its value is VMRF, SIN, PULSE, PWL, or LFSR. If the type is a non-repeating PWL source, then the time=infinity value is used as a DC analysis source value. For example, the following statement is treated as a DC source with value=1 for HB analysis: v1 1 0 PWL (0 0 1n 1 1u 1) + TRANFORHB=1 In contrast, the following statement is a 0V DC source: v1 1 0 PWL (0 0 1n 1 1u 1) + TRANFORHB=0 The following statement is treated as a periodic source with a 1us period that uses PWL values: v1 1 0 PWL (0 0 1n 1 0.999u 1 1u 0) R + TRANFORHB=1 To override the global TRANFORHB option, explicitly set TRANFORHB for a voltage or current source.
DCOPEN	Switch for open DC connection when DC mag is not set. ▪ 0 (default): P element behaves as an impedance termination. ▪ 1 : P element is considered an open circuit in DC operating point analysis. DCOPEN=1 is mainly used in .LIN analysis so the P element will not affect the self-biasing device under test by opening the termination at the operating point.
RDC=*val*	(DC analysis) Series resistance (overrides `z0`).
RAC=*val*	(AC analysis) Series resistance (overrides `z0`).
RHBAC=*val*	(HSPICE RF HBAC analysis) Series resistance (overrides `z0`).
RHB=*val*	(HSPICE RF HB analysis) Series resistance (overrides `z0`).
RTRAN=*val*	(Transient analysis) Series resistance (overrides `z0`).

Parameter	Description
power=[0 | 1 | 2 | W | dbm]	(HSPICE RF) power switch ▪ When 0 (default), element treated as a voltage or current source. ▪ When 1 or W, element treated as a power source, realized as a voltage source with a series impedance. In this case, the source value is interpreted as RMS available power in units of Watts. ▪ When 2 or dbm, element treated as a power source in series with the port impedance. Values are in dbms. You can use this parameter for transient analysis if the power source is either DC or SIN.

Example

For example, the following port element specifications identify a 2-port network with 50-ohm reference impedances between the “in” and “out” nodes.

```
P1 in gnd port=1 z0=50
P2 out gnd port=2 z0=50
```

Computing scattering parameters requires `z0` reference impedance values. The order of the `port` parameters (in the P-element) determines the order of the S, Y, and Z parameters. Unlike the `.NET` command, the `.LIN` command does not require you to insert additional sources into the circuit. To calculate the requested transfer parameters, HSPICE automatically inserts these sources as needed at the port terminals. You can define an unlimited number of ports.

Using the Port Element for Mixed-Mode Measurement

To measure mixed mode S-parameters you can use a port element with three terminals. Except for the number of external terminals, the syntax of the port element remains the same. The LIN analysis function internally sets the necessary drive mode (common/differential) of these mixed mode port elements. For analyses other than the LIN analysis (such as DC, AC, TRAN, and so on), the mixed-mode P-element acts as a differential driver that drives positive nodes with half of their specified voltage and the negative nodes with a negated half of the specified voltage. Figure 16 on page 79 shows the block diagram of the mixed mode port element.

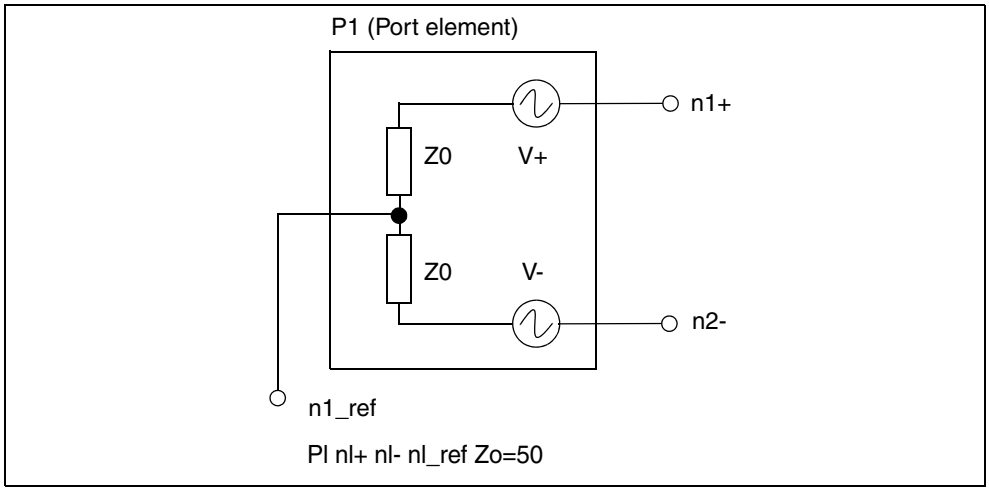


Figure 16 Mixed Mode Port Element

Steady-State Voltage and Current Sources

The I (current source) and V (voltage source) elements include extensions that allow you to use them as sources of steady-state sinusoidal signals for HB and HBAC analyses. When you use a power parameter to specify the available power, you can also use these elements as power sources.

For a general description of the I and V elements, see Power Sources in the *HSPICE User Guide: Simulation and Analysis*.

I and V Element Syntax

```
Vxxx p n
+ $ **** Voltage or Power Information ********
+ [[dc] mag] [ac [mag [phase]]] [HBAC [mag [phase]]]
+[SNAC [mag [phase]]]
+ [hb [mag [phase [harm [tone [modharm [modtone]]]]]]]
+ [transient waveform] [TRANFORHB=[1|0]]

+ $ **** Power Switch ********
+ [power=[0 | 1 | W | dbm]] [z0=val] [rdc=val] [rac=val]
+ [RHBAC=val] [rhb=val] [rtran=val]
```

```
Ixxx p n
+ $ **** Current or Power Information ********
+ [[dc] mag] [ac [mag [phase]]] [HBAC [mag [phase]]]
+[SNAC [mag [phase]]]
+ [hb [mag [phase [harm [tone [modharm [modtone]]]]]]]
+ [transient waveform] [TRANFORHB=[1|0]]

+ $ **** Power Switch ********
+ [power=[0 | 1 | W | dbm]] [z0=val] [rdc=val] [rac=val]
+ [RHBAC=val] [rhb=val] [rtran=val]
```

Parameter	Description
[[dc] mag]	DC voltage or power source value. You don't need to specify DC explicitly (default=0).
[ac [mag [phase]]]	AC voltage or power source value.
[HBAC [mag [phase]]]	(HSPICE RF) HBAC voltage or power source value.
[SNAC [mag [phase]]]	(HSPICE RF) SNAC voltage or power source value.
[hb [mag [phase [harm [tone [modharm [modtone]]]]]]]	(HSPICE RF) HB voltage, current, or power source value. Multiple HB specifications with different harm, tone, modharm, and modtone values are allowed. ▪ *phase* is in degrees ▪ *harm* and *tone* are indices corresponding to the tones specified in the `.HB` statement. Indexing starts at 1 (corresponding to the first harmonic of a tone). ▪ *modtone* and *modharm* specify sources for multi-tone simulation. A source specifies a tone and a harmonic, and up to 1 offset tone and harmonic (modtone for tones and modharm for harmonics). The signal is then described as: V(or I) = mag*cos(2*pi* (harm*tone+modharm*modtone)*t + phase)
[transient waveform]	(Transient analysis) Any one of waveforms: AM, EXP, PULSE, PWL, SFFM, or SIN. Multiple transient descriptions are not allowed.

Parameter	Description
[power=[0 | 1 | W | dbm]]	(HSPICE RF) Power Switch ▪ When 0 (default), element treated as a voltage or current source. ▪ When 1 or W, element treated as a power source, realized as a voltage source with a series impedance. In this case, the source value is interpreted as RMS available power in units of Watts. ▪ When dbm, element treated as a power source in series with the port impedance. Values are in dbms. You can use this parameter for Transient analysis if the power source is either DC or SIN.
[z0=*val*]	(LIN analysis) System impedance used when converting to a power source, inserted in series with the voltage source. Currently, this only supports real impedance. ▪ When power=0, z0 defaults to 0. ▪ When power=1, z0 defaults to 50 ohms. You can also enter zo=*val*.
[rdc=*val*]	(DC analysis) Series resistance (overrides `z0`).
[rac=*val*]	(AC analysis) Series resistance (overrides `z0`).
[RHBAC=*val*]	(HSPICE RF HBAC analysis) Series resistance (overrides z0).
[rhb=*val*]	(HSPICE RF HB analysis) Series resistance (overrides z0).
[rtran=*val*]	(Transient analysis) Series resistance (overrides z0).

Parameter	Description
[TRANFORHB=[0|1]]	■ 0 (default): The transient description is ignored if an HB value is given or a DC value is given. If no DC or HB value is given and TRANFORHB=0, then HB treats the source as a DC source, and the DC source value is the time=0 value. ■ 1: HB analysis uses the transient description if its value is VMRF, SIN, PULSE, PWL, or LFSR. If the type is a non-repeating PWL source, then the time=infinity value is used as a DC source value. For example, the following statement is treated as a DC source with value=1 for HB: v1 1 0 PWL (0 0 1n 1 1u 1) TRANFORHB=1 In contrast, the following statement is a 0V DC source: v1 1 0 PWL (0 0 1n 1 1u 1) TRANFORHB=0 The following statement is treated as a periodic source with a 1us period that uses PWL values: v1 1 0 PWL (0 0 1n 1 0.999u 1 1u 0) R TRANFORHB=1 To override the global TRANFORHB option, explicitly set TRANFORHB for a V/I source.

Example 1

This example shows an HB source for a single tone analysis:

```
.hb tones=100MHz harms=7
```

I1 1 2 dc=1mA hb 3mA 0. 1 1

`I1` is a current source with a the following time-domain description:

```
I1=1mA + 3mA*cos(2*pi*1.e8*t)
```

Example 2

This example shows HB sources used for a two-tone analysis:

```
.hb tones=1.e9 1.1e9 intmodmax=5
Vin lo 0 dc=0. hb 1.5 90 1 1
```

Vrf rf 0 dc=0. hb 0.2 0 1 2

These sources have the following time-domain descriptions:

```
Vin=1.5*cos(2*pi*1.e9*t - 90*pi/180) V
```

Vrf = 0.2*cos(2*pi*1.1e9*t) V

Example 3

The following HB source uses a `modtone` and `modharms`:

```
.hb tones=2.e9 1.9e9 harms=5 5
```

Vm input gnd dc=0.5 hb 0.2 0. 1 1 -1 2

`Vm` has the following time-domain description:

Vm = 0.5 + cos(2*pi*1.e8*t)

Example 4

This example uses an HB source specified with a `SIN` source and `HBTRANINIT`.

```
.hb tone=1.e8 harms=7
```

Vt 1 2 SIN(0.1 1.0 2.e8 0. 0. 90) tranforhb=1

`Vt` is converted to the following HB source:

```
Vt 1 2 dc=0.1 hb 1.0 0.0 2 1
```

Example 5

This example shows a power source (the units are Watts).

```
.hb tones=1.1e9 harms=9
```

Pt Input Gnd power=1 Z0=50. 1m 0. 1 1

`Pt` delivers 1 mW of power through a 50 ohm impedance.

Steady-State HB Sources

The fundamental frequencies used with harmonic balance analysis are specified with the `.HB TONES` command. These frequencies can then be referenced by their integer indices when specifying steady-state signal sources. For example, the .HB specification given by the following line:

.HB TONES=1900MEG,1910MEG INTMODMAX=5

This specifies two fundamental frequencies: $f[tone = 1] = 1.9GHz$ and $f[tone = 2] = 1.91GHz$. Their mixing product at 10 MHz can then be referenced

using indices as $|f[2]-f[1]|$, while their 3rd order intermodulation product at 1.89 GHz can be referenced as $|2f[1]-f[2]|$.

Steady-state voltage and current sources are identified with the HB keyword according to

[HB [mag [phase [harm [tone [modharm [modtone]]]]]]]

The source is mathematically equivalent to a cosine signal source that follows the equation

$$A\cos(\omega t + \phi)$$

where

$$A = mag$$

$$\omega = 2\pi|harm \cdot f[tone] + modharm \cdot f[modtone]|$$

$$\phi = \frac{\pi}{180} \cdot phase$$

Values for tone and modtone (an optional modulating tone) must be non-negative integers that specify index values for the frequencies specified with the `.HB TONES` command. Values for `harm` (harmonic) and `modharm` (modulating tone harmonic) must be integers (negative values are OK) that specify harmonic indices.

Example 1

The following example is a 1.0 Volt (peak) steady-state cosine voltage source, which is at the fundamental HB frequency with zero phase and with a zero volt DC value:

```
Vsrc  in   gnd   DC  0  HB  1.0  0  1  1
```

Example 2

The following example is a steady-state cosine power source with 1.0mW available power, which is implemented with a Norton equivalent circuit and a 50 ohm input impedance:

```
Isrc  in   gnd   HB  1.0e-3  0  1  1  power=1 z0=50
```

Example 3

Five series voltage sources sum to produce a stimulus of five equally spaced frequencies at and above 2.44 GHz using modharm and modtone parameters. These are commensurate tones (an integer relation exists); therefore, you only need to specify two tones when invoking the HB analysis.

```
.param Vin=1.0
.param f0=2440MEG
.param deltaf=312.5K
.param fcenter='f0 + 2.0*deltaf'
Vrfa    in    ina    HB   'Vin'   0   1   1          $ 2.440625
   GHz
Vrfb    ina   inb    HB   'Vin'   0   1   1   -1   2   $
2.4403125   GHz
Vrfc    inb   inc    HB   'Vin'   0   1   1   -2   2   $
2.440   GHz
Vrfd    inc   ind    HB   'Vin'   0   1   1   +1   2   $
2.4409375   GHz
Vrfe    ind   gnd    HB   'Vin'   0   1   1   +2   2   $ 2.44125
   GHz
.HB tones=fcenter,deltaf intmodmax=5
```

Phase Differences Between HB and SIN Sources

The HB steady-state cosine source has a phase variation compared to the `TRAN` time-domain `SIN` source. The `SIN` source (with no offset, delay or damping) follows the equation:

Equation 1 $A\sin(\omega t + \phi)$

while the HB sources follow

Equation 2 $A\cos(\omega t + \phi)$

In order for the two sources to yield identical results it is necessary to align them by setting their phase values accordingly using:

Equation 3 $A\cos(\omega t + \phi) = A\sin(\omega t + \phi + 90^\circ)$

Equation 4 $A\sin(\omega t + \phi) = A\cos(\omega t + \phi - 90^\circ)$

To specify sources with matching phase for HB and TRAN analysis, use a convention similar to:

```
** Example #1 with equivalent HB and SIN sources
** SIN source is given +90 phase shift
.param freq1=2400MEG Vin=1.0
Vsrc in gnd DC 0 HB 'Vin' 0 1 1 SIN(0 'Vin' 'freq1' 0 0 90)
.HB tones=freq1 intmodmax=7
** Example #2 with equivalent HB and SIN sources
** HB source is given -90 phase shift to align with SIN
.param freq1=2400MEG Vin=1.0
Vsrc in gnd DC 0 HB 'Vin' -90 1 1 SIN(0 'Vin' 'freq1' 0)
.HB tones=freq1 intmodmax=7
** Example #3 with equivalent .HB and .TRAN sources
** SIN source is activated for HB using "TRANFORHB"
.param freq1=2400MEG Vin=1.0
Vsrc in gnd DC 0 SIN(0 'Vin' 'freq1' 0) TRANFORHB=1
.HB tones=freq1 intmodmax=7
```

Behavioral Noise Sources

In HSPICE RF, you can use the G-element to specify noise sources. Frequency domain noise analyses (`.NOISE`, `.HBNOISE`, and `.PHASENOISE`) take these noise sources into account.

You can attach noise sources to behavioral models. For example, you can use a G-element with the `VCCAP` parameter to model a varactor, which includes a noise model. You can also simulate effects such as substrate noise, including its effect on oscillator phase noise. You can also use this G element syntax to simulate behavioral descriptions of substrate noise during any frequency domain noise analysis, which includes phase noise analysis. For example,

```
gname node1 node2 noise='noise_equation'
gname node1 node2 node3 node4 noise='noise_equation'
```

The first line creates a simple two-terminal current noise source, whose value is described in A^2/(Hz). The output noise generated from this noise source is:

noise_equation*H

Where `H` is the transfer function from the terminal pair (node1,node2) to the circuit output, where HSPICE RF measures the output noise.

The second line produces a noise source correlation between the (node1,node2) and (node3,node4) terminal pairs. The resulting output noise is calculated as noise_equation*sqrt(H1*H2*); where,

- `H1` is the transfer function from (node1,node2) to the output
- `H2` is the transfer function from (node3,node4) to the output.

The noise_equation expression can involve node voltages and currents through voltage sources.

For the PAC phasenoise simulation to evaluate the frequency-dependent noise, the frequency-dependent noise factor in the phasenoise must be expressed in between the parentheses. For example:

```
gname node1 node2 noise = '(frequency_dependent_noise)*
 bias_dependent_noise'
```

This is only true when the total noise can be expressed in this form and when the frequency-dependent noise can be evaluated in the PAC phasenoise simulation. You can also input the behavioral noise source as a noise table with the help of predefined Table() function. The Table() function takes two formats:

- Noise table can be input directly through the Table() function. For example:

  ```
  gname node1 node2 noise = 'Table(arg1,f1,v1,f2,v2,......)'
  ```

- The *f1,v1,f2,v2,.....* parameters describe the noise table. When arg1 == f1, the function returns v1. The *arg1* can be an expression of either `HERTZ`, bias, or both. For example, `arg1 = 'HERTZ * 1.0E+3'`.
- The noise table can be input through a `.DATA` structure:

  ```
  .DATA d1
  + x y
  + f1 v1
  + f2 v2
  .ENDDATA

  gname node1 node2 noise = 'TABLE(arg1,d1)'
  ```

The `x`, `y` parameters in the `DATA` structure are two placeholder strings that can be set to whatever you prefer even if they are in conflict with other parameters in the netlist. The `arg1` parameter can be an expression of `HERTZ` and bias. When `arg1 == f2`, the function will return `v2`.

Using Noise Analysis Results as Input Noise Sources

SN phase noise and phase noise analyses can output simulation results as ASCII data in `*.printsnpn0` files for SNOSC and SNNOISE. By extending

the E and G voltage-controlled source syntax, the phase noise data in ASCII phase noise files can used as input for specifying behavioral noise sources

Usage Model

The syntax for the voltage controlled voltage (E) or current (G) source is as follows:

```
Exxx node1 node2 noisefile='filename' [mname='measname']
Gxxx node1 node2 noisefile='filename' [mname='measname']
```

Where,

`noisefile='filename'` is the name of the ASCII phase noise data file. The file name is typically designated as 'design.printsnpn0', for a .SNOSC phase noise analysis or .SNNOISE analysis. But it also supports .PHASENOISE, .HBNOISE, .NOISE, and .ACPHASENOISE outputs.

`mname='measname'` is used to select the appropriate noise measurement name to be taken from the **.printpn0* file.

measname can be one of the following:

- NLP_L(f) - selects the nlp_L(f) phase noise data in units of dBc/Hz
- PAC_L(f) - selects the pac_l(f) phase noise data in units of dBc/Hz
- BPN_L(f) - selects the bpn_l(f) phase noise data in units of dBc/Hz
- ONOISE - selects the onoise data based on .SNNOISE analysis

The following syntaxes are supported in both HSPICE RF and HSPICE:

- `Exxx n1 n2 noise data=dataname`
- `Exxx n1 n2 noise data=datablock`
- `Exxx n1 n2 noisefile='filename'`
- `Exxx n1 n2 noise='expression'`
- `Exxx n1 n2 noise='Table(arg1,f1,v1,f2,v2...)'`
- `Exxx n1 n2 noise='Table(arg1,dotDataBlockName)'`, where `dotDataBlockName` is the `.data` statement reference

Power Supply Current and Voltage Noise Sources

You can implement the power supply noise source with G and E elements. The G-element for the current noise source and the E-element for the voltage noise

source. As noise elements, they are two-terminal elements that represent a noise source connected between two specified nodes.

Syntax

Expression form

```
Gxxx node1 node2 noise='expression'
Exxx node1 node2 noise='expression'
```

The G noise element represents a noise current source and the E noise element represents a noise voltage source. The *xxx* parameter can be set with a value up to 1024 characters. The *node1* and *node2* are the positive and negative nodes that connect to the noise source. The noise expression can contain the bias, frequency, or other parameters.

Data form

```
Gxxx node1 node2 noise data=dataname
Exxx node1 node2 noise data=dataname
.data dataname
+ pname1 pname2
+ freq1 noise1
+ freq2 noise2
+ ...
.enddata
```

The data form defines a basic frequency-noise table. The `.DATA` statement contains two parameters: frequency and noise to specify the noise value at each frequency point. The unit for frequency is hertz, and the unit for noise is A^2/Hz (for G current noise source) or V^2/Hz (for E voltage noise source).

Example

The following netlist shows a 1000 ohm resistor (`g1`) using a G-element. The `g1noise` element, placed in parallel with the `g1` resistor, delivers the thermal noise expected from a resistor. The `r1` resistor is included for comparison: The noise due to `r1` should be the same as the noise due to `g1noise`.

```
* Resistor implemented using g-element
v1 1 0 1
r1 1 2 1k
g1 1 2 cur='v(1,2)*0.001'
g1noise 1 2
+ noise='4*1.3806266e-23*(TEMPER+273.15)*0.001'
rout 2 0 1meg
.ac lin 1 100 100
.noise v(2) v1 1
.end
```

Function Approximations for Distributed Devices

High-order rational function approximations constructed for distributed devices used at RF frequencies are obtained in the pole-residue form (also known as Foster canonical form). The popular method of recursive convolution also uses this form.

HSPICE supports the pole-residue form for its frequency-dependent controlled sources (G and E-elements). You can enter the pole-residue form directly without first converting to another form.

Foster Pole-Residue Form for Transconductance or Gain

The Foster pole-residue form for transconductance G(s) or gain E(s) has the form:

Equation 5

$$G(s) = k_0 + k_1 s + \sum_{i=1}^{N} \left(\frac{A_i}{s - p_i} + \frac{A_i^*}{s - p_i^*} \right)$$

Where,

- k_0, k_1 are real constants
- residues A_i and poles p_i are complex numbers (or real as a special case of complex
- asterisk (*) denotes the expression's complex conjugate

Advantages of Foster Form Modeling

The advantages of Foster canonical form modeling are:

- models high-order systems. It can theoretically model systems having infinite poles without numerical problems.
- equivalent to Laplace and Pole-zero models
- popular method of recursive convolution uses this form.

G and E-element Syntax

Transconductance G(s) form

```
Gxxx n+ n- FOSTER in+ in- k0 k1
+ (Re{A1}, Im{A1})/ (Re{p1}, Im{p1})
+ (Re{A2}, Im{A2})/ (Re{p2}, Im{p2})
+ (Re{A3}, Im{A3})/ (Re{p3}, Im{p3})
+ ...
```

Gain E(s) form

```
Exxx n+ n- FOSTER in+ in- k0 k1
+ (Re{A1}, Im{A1})/ (Re{p1}, Im{p1})
+ (Re{A2}, Im{A2})/ (Re{p2}, Im{p2})
+ (Re{A3}, Im{A3})/ (Re{p3}, Im{p3})
+ ...
```

In this syntax, parentheses, commas, and slashes are separators—they have the same meaning as a space. A pole-residue pair is represented by four numbers (real and imaginary part of the residue, then real and imaginary part of the pole).

You must make sure that Re[pi]<0; otherwise, the simulations will certainly diverge. Also, it is a good idea to assure passivity of the model (for an N-port admittance matrix Y, Re{Y} should be positive-definite), or the simulation is likely to diverge).

Example

To represent a G(s) in the form,

Equation 6

$$G(s) = 0.001 + 1 \times 10^{-12} s + \frac{0.0008}{s + 1 \times 10^{10}} + \frac{(0.001 - j0.006)}{s - (-1 \times 10^{8} + j1.8 \times 10^{10})} + \frac{(0.001 + j0.006)}{s - (-1 \times 10^{8} - j1.8 \times 10^{10})}$$

You would input:

```
G1 1 0 FOSTER 2 0 0.001 1e-12
+(0.0004, 0)/(-1e10, 0) (0.001, -0.006)/(-1e8, 1.8e10)
```

Note: In the case of a real poles, half the residue value is entered, because it's essentially applied twice. In the above example, the first pole-residue pair is real, but we still write it as “A1/(s-p1)+A1/(s-p1)”; therefore, 0.0004 is entered rather than 0.0008.

Complex Signal Sources and Stimuli

To predict radio-frequency integrated circuit (RFIC) performance, some analyses require simulations that use representative RF signal sources. Among the representative sources available in HSPICE RF is the complex modulated RF source. Also known as the *Vector Modulated* source, it allows digital modulation of an RF carrier using in-phase and quadrature components created from a binary data stream.

Vector-Modulated RF (VMRF) Source

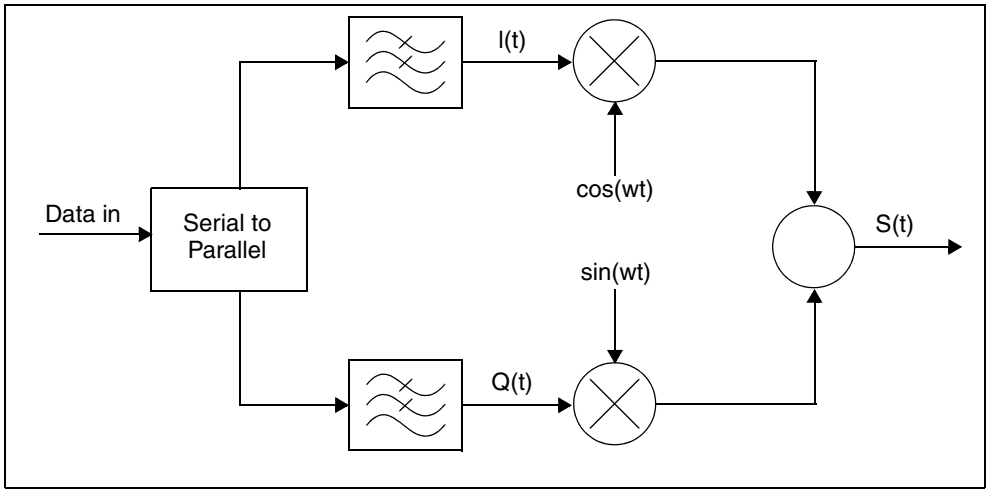
Digital RF waveforms are typically constructed by modulating an RF carrier with in-phase (I) and quadrature (Q) components. In HSPICE RF, this is accomplished using the Vector Modulated RF (VMRF) signal source.

The VMRF signal source function is supported both for independent voltage and current sources (V and I elements), and with controlled sources (E, F, G, and H elements).

- When used with independent sources, a baseband data stream can be input in binary or hexadecimal format, and the scheme used to divide the data into I and Q signals can be specified.
- With controlled VMRF sources, the modulating I and Q signals can be separately specified with other signal sources (such as a PWL source) and then used as control inputs into the VMRF source.

VMRF Implementation

The VMRF source is a mathematical implementation of the following block diagram:



The following equation calculates the time and frequency domain stimuli from the quadrature modulated signal sources:

Equation 7 $$s(t) = I(t)\cos(2\pi f_c t + \phi_0) - Q(t)\sin(2\pi fct + \phi_0)$$

The discrete ideal I (in-phase) and Q (quadrature) signal components are digital. Discrete values allow uniform scaling of the overall signal. HSPICE RF generates data streams for the I and Q signals based on interpreting the data string, breaking the data string into a binary representation, and then using the bit pairs to assign values for the I and Q data streams.

For BPSK (binary phase shift keying) modulation, the discrete signals are scaled so that $\sqrt{I^2 + Q^2} = 1$:

Data In	I Data	Q Data
0	$\frac{-1}{\sqrt{2}}$	$\frac{-1}{\sqrt{2}}$
1	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$

For QPSK (quadrature phase shift keying) modulation, the data stream is broken into bit pairs to form the correct I and Q values. This function is represented as the serial to parallel converter:

Data In	I Data	Q Data
00	$\frac{-1}{\sqrt{2}}$	$\frac{-1}{\sqrt{2}}$
01	$\frac{-1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$
10	$\frac{1}{\sqrt{2}}$	$\frac{-1}{\sqrt{2}}$
11	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$

To generate a continuous-time waveform, the VMRF source takes the resulting digital I and Q data streams and passes them through ideal filters. Rectangular and Nyquist (raised-cosine) filter options are available. The output waveforms are therefore band-limited according to the specified data rate.

Voltage and Current Source Elements

The V and I elements can include VMRF signal sources that you can use to generate BPSK and QPSK waveforms.

V and I Element Syntax

```
Vxxx n+ n- VMRF [()] AMP=sa FREQ=fc PHASE=ph MOD=MOD
+ FILTER=FIL FILCOEF=filpar RATE=Rb BITSTREAM=data
+ [TRANFORHB=0/1] [()]

Ixxx n+ n- VMRF [()] AMP=sa FREQ=fc PHASE=ph MOD=MOD
+ FILTER=FIL FILCOEF=filpar RATE=Rb BITSTREAM=data
+ [TRANFORHB=0/1] [()]
```

Parameter	Description
Vxxx	Independent voltage source.

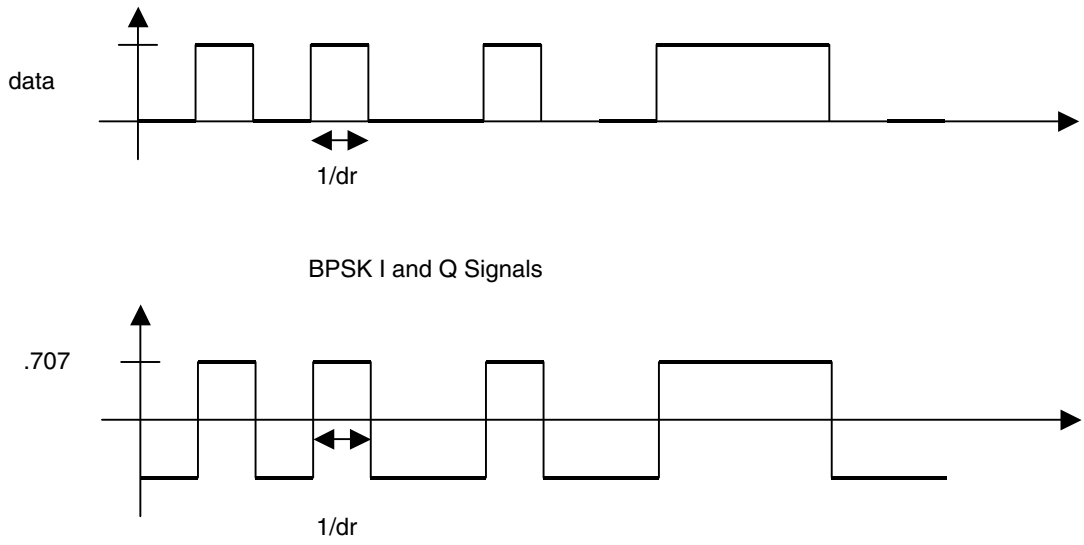
Parameter	Description
Ixxx	Independent current source.
n+ n-	Positive and negative controlled source connecting nodes.
VMRF	Keyword that identifies and activates the Vector Modulated RF signal source.
AMP	Signal amplitude (in volts or amps).
FREQ	Carrier frequency in hertz. Set fc=0.0 to generate baseband I/Q signals. For harmonic balance analysis, the frequency spacing must coincide with the .HB TONES settings.
PHASE	Carrier phase (in degrees). If fc=0.0, ▪ ph=0 and baseband I(t) is generated ▪ ph=-90 and baseband q(t) is generated ▪ Otherwise, $s(t) = I(t)\cos(\phi_0) - Q(t)\sin(\phi_0)$
MOD	One of the following keywords identifies the modulation method used to convert a digital stream of information to I(t) and Q(t) variations: ▪ BPSK (binary phase shift keying) ▪ QPSK (quadrature phase shift keying)
FILTER	One of the following keywords identifies the method used to filter the I and Q signals before modulating the RF carrier signal: ▪ COS (raised cosine Nyquist filter) ▪ RECT (rectangular filtering)
FILCOEF	Filter parameter for the COS filter: 0 ≤filpar ≤1
RATE	Bit rate for modulation (bits per second). ▪ For BPSK modulation, the data rate and the symbol rate are the same. ▪ For QPSK modulation, the symbol rate is half the data rate. The Rb value must be greater than zero.

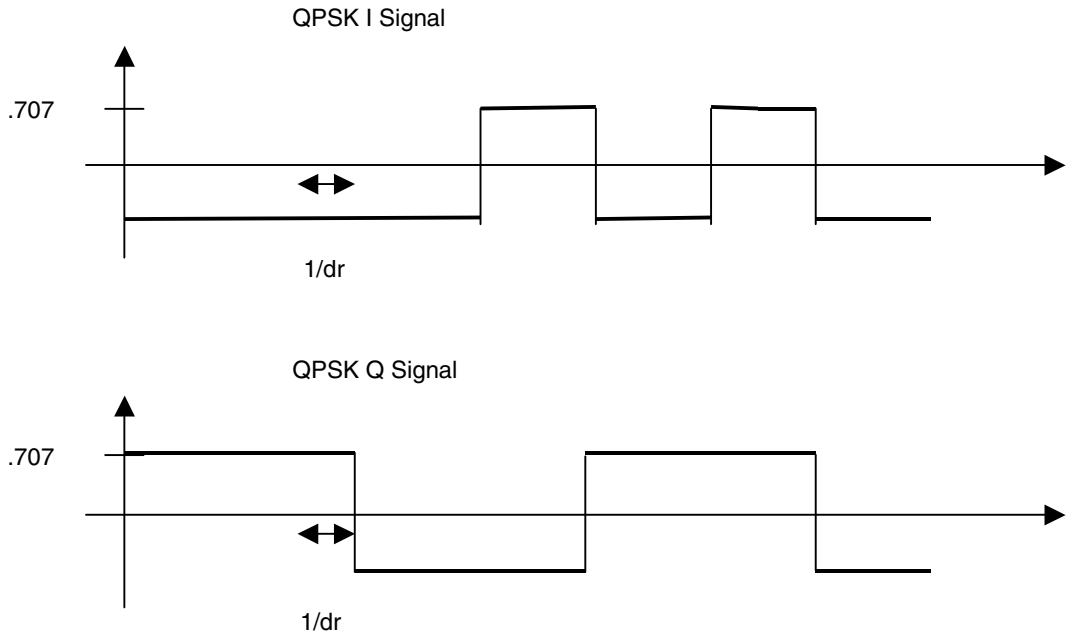
Parameter	Description
BITSTREAM	A binary (b) or hexadecimal (h) string that represents an input data stream. Valid data string characters are: ▪ 0 or 1 for binary (b) mode. ▪ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, a, b, c, d, e, or f for hexadecimal (h) mode. For example: ▪ 01010011b (binary) ▪ 0F647A30E9h (hexadecimal)

You can also use the standard V source and I source options for non-transient simulations (such as `DC=`*`val`* and `AC=`*`mag`*`,`*`ph`*) a with the VMRF source.

Example

```
BITSTREAM=01010010011100b
```





The `Rb` parameter represents the data rate. The associated symbol rate represents how fast the I and Q data streams change. The period for each bit of data is:

Equation 8 $$T_b = \frac{1}{R_b}$$

The symbol rate depends on whether you select `BPSK` or `QPSK` modulation:

- For `BPSK`, the symbol rate is the same as the data rate:

  $$_S R^{BPSK} = R_b$$

- For `QPSK` modulation, two bits are used to create each symbol so the symbol rate is half the data rate.

  $$_S R^{QPSK} = \frac{R_b}{2}$$

The period for each symbol is computed as:

Equation 9 $$T_s = \frac{1}{R_s}$$

This value is necessary for establishing the characteristics of Nyquist filters.

The following equation calculates the raised cosine (COS) filter response:

Equation 10

$$H_{rc}(f) = \int_0^{T_s} T_s cos^2\left[\frac{\pi T_s}{2\alpha}\left(|f| - \frac{1-\alpha}{2T_s}\right)\right]$$

$$|f| \le \frac{1-\alpha}{2T_s}$$

$$\frac{1-\alpha}{2T_s} \le |f| \le \frac{1+\alpha}{2T_s}$$

$$|f| > \frac{1+a}{2T_s}$$

The VMRF signal source is designed primarily for TRAN and HB analyses, and can generate baseband signals. You can also specify DC and AC values as with any other HSPICE signal source:

- In DC analysis, the VMRF source is a constant DC source.
- In AC analysis, the source is a short or an open, unless you specify an AC value.
- In HB analysis, you must specify `.OPTION TRANFORHB` on the source statement line. The `TRANFORHB` option supports the VMRF signal source as well as the SIN, PULSE, and PWL sources.

The VMRF quadrature signal source typically involves an HF carrier signal that is modulated with a baseband signal on a much different time scale. You must set source and simulation control parameters appropriately to avoid time-consuming simulations in both the time and frequency domains.

E, F, G, and H Element Statements

For E, F, G, and H elements, you can use the VMRF function to modulate I(t) and Q(t) signals with a RF carrier signal. The I and Q signal are driven by PWL sources that might be generated by an external tool, such as MATLAB. The PWL source accepts a text file containing time and voltage (or current) pairs.

When the VMRF function is used with controlled sources, it is anticipated that the in-phase (I) and quadrature (Q) signals are not digital, but continuous-time analog signals. The VMRF function therefore includes no filtering, and merely serves to create the complex modulation on the RF carrier.

```
Exxx n+ n- [VCVS] VMRF [() Iin+ Iin- Qin+ Qin- FREQ=fc
+ PHASE=ph [SCALE=A] ()]

Fxxx n+ n- [CCCS] VMRF [() VI VQ FREQ=fc PHASE=ph
+ [SCALE=A] ()]
```

```
Gxxx n+ n- [VCCS] VMRF [() Iin+ Iin- Qin+ Qin- FREQ=fc
+ PHASE=ph [SCALE=A] ())]

Hxxx n+ n- [CCVS] VMRF [() VI VQ FREQ=fc PHASE=ph
+ [SCALE=A] ())]
```

Parameter	Description
Exxx	Voltage-controlled voltage source.
Fxxx	Current-controlled current source.
G*xxx*	Voltage-controlled current source.
H*xxx*	Current-controlled current source.
VCVS	Keyword for voltage-controlled voltage source.
CCCS	Keyword for current-controlled current source.
VCCS	Keyword for voltage-controlled current source.
CCVS	Keyword for current-controlled current source.
n+ n-	Positive and negative controlled source connecting nodes.
VMRF	Keyword that identifies and activates the vector-modulated RF signal source.
Iin+ Iin-	Node names for input I(t) signal.
Qin+ Qin-	Node names for input Q(t) signal.
VI VQ	
FREQ	Carrier frequency in Hertz. Set fc=0.0 to generate baseband I/Q signals.
PHASE	Carrier phase (in degrees). If fc=0.0, ▪ ph=0 and baseband I(t) is generated ▪ ph=-90 and baseband Q(t) is generated
SCALE	Unit-less amplitude scaling parameter.

Example

```
Emod1 inp1 inn1 VMRF It_plus It_neg Qt_plus Qt_neg
+ freq=1g phase=0 scale=1.5
```

SWEEPBLOCK in Sweep Analyses

You can use the .SWEEPBLOCK statement to specify complicated sweeps. Sweeps affect:

- DC sweep analysis
- Parameter sweeps around TRAN, AC, or HB analyses
- Frequency values used in AC or HBAC analyses

Currently, HSPICE supports the following types of sweeps:

- Linear sweeps: sweeps a variable over an interval with a constant increment. The syntax is one of the following:
  - variable *start stop increment*
  - variable *lin npoints start stop*
- Logarithmic sweeps: sweeps a variable over an interval. To obtain each point, this sweep multiplies the previous point by a constant factor. You can specify the factor as a number of points per decade or octave as in:
  - variable *dec npoints start stop*
  - variable *oct npoints start stop*
- Point sweeps: a variable takes on specific values that you specify as a list. The syntax is:

```
variable poi npoints p1 p2 …
```

- Data sweeps: a .DATA statement identifies the swept variables and their values. The syntax is:

```
data=dataname
```

You can use the SWEEPBLOCK feature to combine linear, logarithmic, and point sweeps, which creates more complicated sets of values over which a variable is swept.

The .TRAN, .AC, .DC, and .HB commands can specify SWEEPBLOCK=*blockname* as a sweep instead of LIN, DEC, OCT, and so forth.

Also, you can use SWEEPBLOCK for frequency sweeps with the .AC, .HBAC, .PHASENOISE, and .HBNOISE commands.

All commands that can use SWEEPBLOCK must refer to the SWEEPBLOCK sweep type. In addition, you must specify SWEEPBLOCK as one of the syntax types allowed for frequency sweeps with the .HBAC, .PHASENOISE, and .HBNOISE commands.

Input Syntax

The SWEEPBLOCK feature creates a sweep whose set of values is the union of a set of linear, logarithmic, and point sweeps. To specify the set of values in the SWEEPBLOCK, use the .SWEEPBLOCK command. This command also assigns a name to the SWEEPBLOCK. For example,

```
.SWEEPBLOCK swblockname sweepspec [sweepspec
+ [sweepspec […]]]]
```

You can use SWEEPBLOCK to specify DC sweeps, parameter sweeps, AC and HBAC frequency sweeps, or wherever HSPICE accepts sweeps.

You can specify an unlimited number of *sweepspec* parameters. Each sweepspec can specify a linear, logarithmic, or point sweep by using one of the following forms:

```
start stop increment
lin npoints start stop
dec npoints start stop
oct npoints start stop
poi npoints p1 p2 …
```

Example

The following example specifies a logarithmic sweep from 1 to 1e9 with more resolution from 1e6 to 1e7:

```
.sweepblock freqsweep dec 10 1 1g dec 1000 1meg 10meg
```

Using SWEEPBLOCK in a DC Parameter Sweep

To use the sweepblock in a DC parameter sweep, use the following syntax:

```
.DC sweepspec [sweepspec [sweepspec]]
```

Each *sweepspec* can be a linear, logarithmic, point, or data sweep, or it can be in the form:

```
variable SWEEPBLOCK=swblockname
```

The SWEEPBLOCK syntax sweeps the specified variable over the values contained in the SWEEPBLOCK.

Example

```
.dc vin1 0 5 0.1 vin2 sweepblock=vin2vals
```

Using in Parameter Sweeps in TRAN, AC, and HB Analyses

To use the sweepblock in parameter sweeps on .TRAN, .AC, and .HB commands, and any other commands that allow parameter sweeps, use the following syntax:

```
variable sweepblock=swblockname
```

Example 1

.tran 1n 100n sweep rout sweepblock=rvals

AC and HBAC analysis frequency sweeps can use sweepblock=*swblockname* to specify the frequency values.

Example 2

.ac sweepblock=freqsweep

Limitations

- You cannot use recursive SWEEPBLOCK specifications. That is, a .SWEEPBLOCK command cannot refer to another SWEEPBLOCK to build its list of values.
- You cannot include data sweeps in a .SWEEPBLOCK statement.

Clock Source with Random Jitter

In many applications involving signal integrity, RF, analog, and mixed-signal design, it is desirable to have an ideal signal source, such as a sine wave or square wave, that also includes a non-ideal random drift in phase (jitter). Such a source is useful for representing non-ideal clock sources during time-domain transient simulation. Modeling jitter in this way can be used to examine eye-

diagram behavior or study how jitter may propagate through a circuit or system. A source with jitter is useful for representing non-ideal clock sources during time-domain transient simulation.

The PERJITTER option allows you to add periodic jitter to SIN, COS and PULSE time domain sources.

Syntax of SIN, COS, and Pulse Sources

The syntax of SIN source is:

```
Vxxx n+ n- SIN [() vo va [freq [td [q [j ]]]] [)]
+ [PERJITTER=val SEED=val]]
Ixxx n+ n- SIN [() vo va [freq [td [q [j ]]]] [)]
+ [PERJITTER=val SEED=val]]
```

Parameter	Description
Vxxx	Independent voltage source.
Ixxx	Independent current source.
PERJITTER	Period jitter
PWL	Keyword for piecewise linear.
PWLFILE	Text file containing the PWL data consisting of time and voltage (or current) pairs. This file should not contain a header row, unless it is a comment. The PWL source data is obtained by extracting col1 and col2 from the file.
col1, [*col2*]	Time values are in col1 and voltage (or current) values are in col2. By default, col1=1 and col2=2.
R	Repeat function. When an argument is not specified, the source repeats from the beginning of the function. The argument repeated is the time, in seconds, which specifies the start point of the waveform being repeat. The repeat time must be less than the greatest time point in the file.
TD	Time delay, in seconds, of the PWL function.
options	Any standard V or I source options.

The sine wave behavior following the td time delay now becomes

Equation 11
$$V(t) + e^{-(t-t_d)\cdot\ \theta} = V_0 + V_a \cdot \left[\sin 2\pi f_0(t-t_d) + \frac{\pi}{180}\varphi + \phi t - d_d\right]$$

The Syntax of COS source is:

```
Vxxx n+ n- COS [() vo va [freq [td [q][j]] [)]
+ [PERJITTER=val SEED=val]
Ixxx n+ n- COS [() vo va [freq [td [q] [j]] [)]
+ [PERJITTER=val SEED=val]
```

The new cosine wave becomes

Equation 12
$$V(t) + e^{-(t-t_d)\cdot\ \theta} = V_0 + V_a \cdot \left[\cos 2\pi f_0(t-t_d+x(t)) + \frac{\pi}{180}\varphi\right]$$

The syntax for the PULSE source is:

```
Vxxx n+ n- PU[LSE] [()v1 v2 [td [tr] [tf] [pw] [per] [)]
+ [PERJITTER=val SEED=val]
Ixxx n+ n- PU[LSE] [()v1 v2 [td [tr] [tf] [pw] [per] [)]
+ [PERJITTER=val SEED=val]
```

The effect of jitter on the PULSE source results in random shifts of the rise and fall transitions that takes place at

RISE edge: $td + n\cdot\ T_0 \leq t \leq td + tr + n\cdot\ T_0$

FALL edge: $td + pw + n\cdot\ T_0 \leq t \leq td + pw + tf + n\cdot\ T_0$

The jitter effect is equivalent to introducing random shifts in the period T_0 consistent with the 1st order jitter model based on Period Jitter.

A Gaussian random number generator computes the time deviation $x(t)$ after each leading edge of the clock sources. For flexibility, the SEED parameter (integer) is supported for generating different random number sequences when different SEED integers are used for initialization. SEED does not set a fixed time deviation. It only changes the sequence of random samples. By HSPICE (Monte Carlo) convention, the default value for SEED is 1.

An interpretation of PERJITTER is to view it as causing each period of the PULSE/SIN/COS to be a random variable T_j, where period T_j will have a Gaussian distribution about the (mean) given period value of T_0. The standard deviation of this Gaussian is the PERJITTER value (it is considered RMS period jitter), which results in a bell curve distribution centered about period T_0.

Apply the following considerations when using PERJITTER:

- T_j should be forced to be between: $0 < T_j < 2 \cdot \ T_0$, since period cannot go negative, and the curve should be symmetrical.
- It is reasonable to require that $2 \cdot \ \text{PERJITTER} < T_0$. Otherwise, the jitter would result in very large period changes, and many would be $T_j < 0$.
- To establish a waveform reference, the first period should be T_0 (i.e., no jitter in the first period). This helps to establish good eye diagrams.

Example

As an alternative to using a Verilog-A module, you can generate a pseudo-random binary sequence (PRBS) using the following steps:

1. Construct your usual linear feedback shift register (LFSR) generator.
2. Construct a matching (T,tr,tf) PULSE source as a clock, but add jitter to it with the PERJITTER keyword.
3. Use the PULSE source to gate (buffer) the LFSR output (through an ideal AND gate, VCCS, and so forth).

References

[1] L.J. Greenstein and M.Shafi, *Microwave Digital Radio*, IEEE Press, 1988.

[2] N. Sheikholeslami and P. Kabal, “A Family of Nyquist Filters Based on Generalized Raised-Cosine Spectra,” *Proceedings of the 19th Biennial Symposium on Communications* (Kingston, Ontario), pages 131-135, June 1998.

[3] IEEE Standard Definitions of Physical Quantities for Fundamental Frequency and Time Metrology - Random Instabilities, IEEE Std. 1139-1999.

[4] A. van der Ziel, Noise in Solid State Devices and Circuits, John Wiley & Sons, © 1986.

[5] A. Demir, A. Mehrotra, and J. Roychowdhury, “Phase noise in oscillators: A unifying theory and numerical methods for characterization,” IEEE Trans. Circuits Syst. I, vol. 47, pp. 655-674, May 2000.

[6] A. Hajimiri, S. Limotyrakis, and T.H. Lee, “Jitter and phase noise in ring oscillators,” IEEE J. Solid-State Circuits, vol. 34, no. 6, pp. 790-804, June 1999.

[7] Jitter Analysis Techniques for High Data Rates, Application Note 1432, Agilent Technologies, Feb. 2003.[6] Characterization of Clocks and Oscillators, NIST Technical Note 1337, National Institute of Standards and Technology.

5

Steady-State Harmonic Balance Analysis

Describes how to use harmonic balance analysis for frequency-driven, steady-state analysis.

HSPICE RF provides several analyses that support the simulation and analysis of radio-frequency integrated circuits (RFICs). These analyses provide simulation capabilities that are either much more difficult to perform, or are not practically possible by using standard HSPICE analyses. The RF analyses include:

- Harmonic Balance (HB) for frequency-domain, steady-state analysis, see Harmonic Balance Analysis on page 108.
- Shooting Newton (SN) for frequency or time domain steady state analysis, see Chapter 6, Steady-State Shooting Newton Analysis plus spectrum analysis specific to the SN analysis (see Shooting Newton with Fourier Transform (.SNFT).
- Harmonic Balance oscillator analysis (HBOSC), see Harmonic Balance Oscillator Analysis (.HBOSC) on page 144.
- Shooting Newton oscillator analysis (SNOSC), see Oscillator Analysis Using Shooting Newton (.SNOSC) on page 158.
- Harmonic Balance AC (HBAC) for periodic AC analysis, see Chapter 8, Large Signal Periodic AC, Transfer Function, and Noise Analyses, Multitone Harmonic Balance AC Analysis (.HBAC) on page 189.
- Shooting Newton AC analysis, see Shooting Newton AC Analysis (.SNAC) on page 195.
- Harmonic Balance Noise (HBNOISE) for periodic, time-varying AC noise analysis (see Chapter 8, Large Signal Periodic AC, Transfer Function, and Noise Analyses).

- Shooting Newton noise analysis, see Shooting Newton Noise Analysis (.SNNOISE) on page 209.
- Harmonic balance transfer functions, see Multitone Harmonic Balance Transfer Function Analysis (.HBXF) on page 226.
- Shooting Newton transfer functions, see Shooting Newton Transfer Function Analysis (.SNXF) on page 230
- Frequency translation S-parameter extraction for describing N-port circuits that exhibit frequency translation effects (see Frequency Translation S-Parameter (HBLIN) Extraction on page 256).
- Envelope Analysis (ENV) (see Chapter 11, Envelope Analysis).

You can use steady-state analysis on a circuit if it contains only DC and periodic sources. These analyses assume that all “start-up” transients have completely died out with only the steady-state response remaining. Sources that are not periodic or DC are treated as zero-valued in these analyses.

Harmonic Balance Analysis

Harmonic balance analysis (HB) is a frequency-domain, steady-state analysis technique. In HSPICE RF, you can use this analysis technique on a circuit that is excited by DC and periodic sources of one or more fundamental tones. The solution that HB finds is a set of phasors for each harmonic signal in the circuit. You can think of this solution as a set of truncated Fourier series. HSPICE RF allows you to specify the solution spectrum to use in an analysis. HB analysis then finds the set of phasors at these frequencies that describes the circuit response. The result is a set of complex valued Fourier series coefficients that represent the waveforms at each node in the circuit.

Linear circuit elements are evaluated in the frequency domain, while nonlinear elements are evaluated in the time domain. The nonlinear response is then transformed to the frequency domain where it is added to (or “balanced” with) the linear response. The resulting composite response satisfies KCL and KVL (Kirchoff's current and voltage laws) when the circuit solution is found.

Typical applications include performing intermodulation analysis, oscillator analysis, and gain compression analysis, on amplifiers and mixers. HB analysis also serves as a starting point for periodic AC and noise analyses.

Harmonic Balance Equations

We can write Kirchoff's current law in the time domain as:

Equation 13 $$f(v,\ t)\ =\ i(v(t)) + \frac{d}{dt}q(v(t)) + \int_{-\infty}^{t} y(t-\tau)v(\tau)d\tau + i_s(t)\ =\ 0$$

- *i(v(t))* represents the resistive currents from nonlinear devices
- *q* represents the charges from nonlinear devices
- *y* represents the admittance of the linear devices in the circuit
- i_s represents the vector of independent current sources
- *v* is a variable that represents the circuit unknowns, both node voltages and branch currents, and f(v,t) is an error term that goes to zero when Kirchoff's current law is satisfied.

Transforming this equation to the frequency domain results in:

Equation 14 $$F(V)\ =\ I(V) + \Omega Q(V) + Y(\omega)V + I_s\ =\ 0$$

Note: Time-differentiation is transformed to multiplication by $j\omega$ terms (which make up the Ω matrix) in the frequency domain. The convolution integral is transformed to a simple multiplication. The Y matrix is the circuit's modified nodal admittance matrix.

All terms above are vectors, representing the circuit response at each analysis frequency.

The following equation shows the vector of (complex-valued) unknowns in the frequency domain for a circuit with *K* analysis frequencies and *N* unknowns.

Equation 15 $$V=\ \left[V_{(1,\ 0)}\ V_{(1,\ 1)}\ \cdots V_{(1,\ K-1)}\ V_{(2,\ 0)}\ \cdots V_{(N,\ K-1)}\right]$$

HSPICE RF finds the unknown vector (*V*), which satisfies the system of nonlinear equations shown in the equation above. This is done via the Newton-Raphson technique by using either a direct solver to factor the Jacobian matrix, or an indirect solver. The indirect solver available in HSPICE RF is the Generalized Minimum Residual (GMRES) Solver, a Krylov technique, and uses a matrix-implicit algorithm.

Features Supported

HB supports the following features:

- All existing HSPICE RF models.
- Unlimited number of independent input tones.
- Sources with multiple HB specifications.
- SIN, PULSE, VMRF, and PWL sources with `TRANFORHB=1`.

Prerequisites and Limitations

The following prerequisites and limitations apply to HB:

- Requires one `.HB` statement.
- Treats sources without a DC, HB, or TRANFORHB description as a zero-value for HB unless the sources have a transient description, in which case, the `time=0` value is used as a DC value.

Input Syntax

Without `SS_TONE`

```
.HB TONES=F1 [F2 ... FN] [SUBHARMS=SH]
+ [NHARMS=H1, [H2 ...HN] [INTMODMAX=n]
+ [SWEEP parameter_sweep]
```

With `SS_TONE`

```
.HB TONES=F1 [F2 ... FN]
+ [NHARMS=H1, [H2 ...HN] [INTMODMAX=n]
+ [SS_TONE=n] [SWEEP parameter_sweep]
```

Parameter	Description
TONES	Fundamental frequencies.
SUBHARMS	Allows subharmonics in the analysis spectrum. The minimum non-DC frequency in the analysis spectrum is f/subharms, where f is the frequency of oscillation.

Parameter	Description
NHARMS	Number of harmonics to use for each tone. Must have the same number of entries as TONES. You must specify NHARMS, INTMODMAX, or both.
INTMODMAX	INTMODMAX is the maximum intermodulation product order that you can specify in the analysis spectrum. You must specify NHARMS, INTMODMAX, or both.
SS_TONE	Small-signal tone number for HBLIN analysis. The value must be an integer number. The default value is 0, indicating that no small signal tone is specified. For additional information, see Frequency Translation S-Parameter (HBLIN) Extraction on page 256.
SWEEP	Type of sweep. You can sweep up to three variables. You can specify either LIN, DEC, OCT, POI, SWEEPBLOCK, DATA, OPTIMIZE, or MONTE. Specify the nsteps, start, and stop frequencies using the following syntax for each type of sweep: ■ LIN *nsteps start stop* ■ DEC *nsteps start stop* ■ OCT *nsteps start stop* ■ POI *nsteps freq_values* ■ SWEEPBLOCK *nsteps freq1 freq2 ... freqn* ■ DATA=*dataname* ■ OPTIMIZE=OPT*xxx* ■ MONTE=*val*

HB Analysis Spectrum

The `NHARMS` and `INTMODMAX` input parameters define the spectrum.

- If `INTMODMAX=N`, the spectrum consists of all f=a*f_1 + b*f_2 + ... + n*fn frequencies so that f>=0 and |a|+|b|+...+|n|<=N. The a,b,...,n coefficients are integers with absolute value <=N.
- If you do not specify `INTMODMAX`, it defaults to the largest value in the `NHARMS` list.
- If entries in the `NHARMS` list are > `INTMODMAX`, HSPICE RF adds the m*f_k frequencies to the spectrum, where f_k is the corresponding tone, and m is a value <= the `NHARMS` entry.

Example 1

```
.hb tones=f1, f2 intmodmax=1
```

The resulting HB analysis spectrum={dc, f_1, f_2}

Example 2

```
.hb tones=f1, f2 intmodmax=2
```

The resulting HB analysis spectrum={dc, f_1, f_2, f_1+f_2, f_1-f_2, $2*f_1$, $2*f_2$}

Example 3

```
.hb tones=f1, f2 intmodmax=3
```

The resulting HB analysis spectrum={dc, f_1, f_2, f_1+f_2, f_1-f_2, $2*f_1$, $2*f_2$, $2*f_1+f_2$, $2*f_1-f_2$, $2*f_2+f_1$, $2*f_2-f_1$, $3*f_1$, $3*f_2$}

Example 4

```
.hb tones=f1, f2 nharms=2,2
```

The resulting HB analysis spectrum={dc, f_1, f_2, f_1+f_2, f_1-f_2, $2*f_1$, $2*f_2$}

Example 5

```
hb tones=f1, f2 nharms=2,2 intmodmax=3
```

The resulting HB analysis spectrum={dc, f_1, f_2, f_1+f_2, f_1-f_2, $2*f_1$, $2*f_2$, $2*f_1-f_2$, $2*f_1+f_2$, $2*f_2-f_1$, $2*f_2+f_1$}

Example 6

```
.hb tones=f1, f2 nharms=5,5 intmodmax=3
```

The resulting HB analysis spectrum={dc, f_1, f_2, f_1+f_2, f_1-f_2, $2*f_1$, $2*f_2$, $2*f_1-f_2$, $2*f_1+f_2$, $2*f_2-f_1$, $2*f_2+f_1$, $3*f_1$, $3*f_2$, $4*f_1$, $4*f_2$, $5*f_1$, $5*f_2$}

HB Analysis Options

The following table lists the `.OPTION` command options specific to HB analysis.

Table 4 HB Analysis Options

Option	Description
HBCONTINUE	Specifies whether to use the sweep solution from the previous simulation as the initial guess for the present simulation. ▪ HBCONTINUE=1 (default): Use solution from previous simulation as the initial guess. ▪ HBCONTINUE=0: Start each simulation in a sweep from the DC solution.
HBJREUSE	Controls when to recalculate the Jacobian matrix: ▪ HBJREUSE=0 recalculates the Jacobian matrix at each iteration. ▪ HBJREUSE=1 reuses the Jacobian matrix for several iterations, if the error is sufficiently reduced. The default is 0 if HBSOLVER=1 or 2, or 1 if HBSOLVER=0.
HBJREUSETOL	Determines when to recalculate Jacobian matrix (if HBJREUSE=1). The percentage by which HSPICE RF must reduce the error from the last iteration so you can use the Jacobian matrix for the next iteration. Must be a real number, between 0 and 1. The default is 0.05.
HBKRYLOVDIM	Dimension of the Krylov subspace that the Krylov solver uses. Must be an integer, greater than zero. Default is 40.
HBKRYLOVTOL	The error tolerance for the Krylov solver. Must be a real number, greater than zero. The default is 0.01.
HBLINESEARCHFAC	The line search factor. If Newton iteration produces a new vector of HB unknowns with a higher error than the last iteration, then scale the update step by HBLINESEARCHFAC, and try again. Must be a real number, between 0 and 1. The default is 0.35.

Table 4 HB Analysis Options (Continued)

Option	Description
HBMAXITER (or HB_MAXITER)	Specifies the maximum number of Newton-Raphson iterations that the HB engine performs. Analysis stops when the number of iterations reaches this value. The default is 10000.
HBKRYLOVMAXITER (or HB_KRYLOV_MAXITER	Specifies the maximum number of GMRES solver iterations performed by the HB engine.
HBSOLVER	Specifies a preconditioner to solve nonlinear circuits. ▪ HBSOLVER=0: invokes the direct solver. ▪ HBSOLVER=1 (default): invokes the matrix-free Krylov solver. ▪ HBSOLVER=2: invokes the two-level hybrid time-frequency domain solver.
HBTOL	The absolute error tolerance for determining convergence. Must be a real number that is greater than zero. The default is 1.e-9.
LOADHB	LOADHB='filename' loads the state variable information contained in the specified file. These values are used to initialize the HB simulation.
SAVEHB	SAVEHB='filename' saves the final state (that is, the no sweep point or the steady state of the first sweep point) variable values from a HB simulation in the specified file. This file can be loaded as the starting point for another simulation by using a LOADHB option.
TRANFORHB	▪ TRANFORHB=1: forces HB to recognize V/I sources that include SIN, PULSE, VMRF, and PWL transient descriptions, and to use them in analysis. However, if the source also has an HB description, analysis uses the HB description instead. ▪ TRANFORHB=0: forces HB to ignore transient descriptions of V/I sources, and to use only HB descriptions. To override this option, specify TRANFORHB in the source description.

Harmonic Balance Output Measurements

This section explains the harmonic balance output measurements you receive after HSPICE runs an HB simulation.

Harmonic Balance Signal Representation

The HB cosine sources can be interpreted in real/imaginary and polar formats according to:

Equation 16

$$\begin{aligned} v(t) &= A\cos(\alpha t+\phi) = Re\{Ae^{j(\alpha t+\phi)}\} = Re\{Ae^{j\phi}e^{j\omega t}\} \\ &= Re\{Ae^{j\phi}[\cos(\alpha t)+j\sin(\alpha t)]\} \\ &= Re\{[V_R+jV_I][\cos(\alpha t)+j\sin(\alpha t)]\} \\ &= V_R\cos(\alpha t)-V_I\sin(at) \\ &= A\cos(\phi)\cos(\alpha t)-A\sin(\phi)\sin(\alpha t) \end{aligned}$$

Note that real/imaginary and polar formats are related with the standard convention:

Equation 17

$$\begin{aligned} V_R+jV_I &= Ae^{j\phi} \\ V_R &= A\cos(\phi) \\ V_I &= A\sin(\phi) \\ A &= \sqrt{V_R^2+V_I^2} \\ \tan\phi &= \frac{V_I}{V_R} \end{aligned}$$

The result of HB analysis is a complex voltage (current) spectrum at each circuit node (or specified branch). Let a[i] be the real part and b[i] be the imaginary part of the complex voltage at the ith frequency index. Conversion to a steady-state time-domain waveform is given by the Fourier series expansion:

Equation 18

$$v(t) = a[0] + a[1]*\cos(2\pi f[1]*t) - b[1]*\sin(2\pi f[1]*t) + a[2]*\cos(2\pi f[2]*t) - b[2]*\sin(2\pi f[2]*t) + a[3]*\cos(2\pi f[3]*t) - b[3]*\sin(2\pi f[3]*t) + \ldots + a[N]*\cos(2\pi f[N]*t) - b[N]*\sin(2\pi f[N]*t)$$

Where:

- v(t) is the resulting time domain waveform.
- *N*+1 is the total number of harmonics (including DC) in the frequency domain spectrum of the *.hb0 file (the zero-th data point represents DC).
- a[i] is the real component at the ith frequency
- b[i] is the imaginary component at the ith frequency
- f[i] is the ith frequency value (with i=0 representing the zero frequency DC term). These frequencies need not be harmonically related.

This frequency domain (Fourier coefficient) representation can be converted into a steady-state time domain waveform output representation by using the `.PRINT` or `.PROBE HBTRAN` output option or by invoking the To Time Domain function on complex spectra within Custom WaveView.

Output Syntax

This section describes the syntax for the HB `.PRINT` and `.PROBE` statements.

.PRINT and .PROBE Statements

```
.PRINT HB TYPE(NODES or ELEM)[INDICES]
```

```
.PROBE HB TYPE(NODES or ELEM)[INDICES]
```

Parameter	Description
TYPE(NODES or ELEM)	Specifies a harmonic type node or element. TYPE can be one of the following: ■ Voltage type – V = voltage magnitude and phase in degrees VR = real component VI = imaginary component VM = magnitude VP - Phase in degrees VPD - Phase in degrees VPR - Phase in radians VDB - dB units VDBM - dB relative to 1 mV ■ Current type – I = current magnitude and phase in degrees IR = real component II = imaginary component IM = magnitude IP - Phase in degrees IPD - Phase in degrees IPR - Phase in radians IDB - dB units IDBM - dB relative to 1 mV ■ Power type – P ■ Frequency type – 'HERTZ[i]' (for single tone analysis), 'HERTZ[i][j]' (for two-tone analysis) , 'HERTZ[i][j][k]' (for 3-tone analysis), etc. You must specify the harmonic index integer for the HERTZ keyword. The frequency of the specified harmonics is dumped. NODES or ELEM can be one of the following: ■ Voltage type – a single node name (n1), or a pair of node names, (n1,n2) ■ Current type – an element name (elemname) ■ Power type – a resistor (resistorname) or port (portname) element name.

Parameter	Description
INDICES	Index to tones in the form [n1, n2, ..., nN], where nj is the index of the HB tone and the HB statement contains N tones. If INDICES is used, then wildcards are not supported.

HB data can be transformed into the time domain and output using the following syntax:

```
.PRINT hbtran ov1 [ov2 ... ]
.PROBE hbtran ov1 [ov2 ... ]
```

Where `ov1 ...` are the output variables to print or probe.

Outputting Phase Noise Source as ASCII Data Files Using *.printpn0

HB phase noise and phase noise analyses can output simulation results as ASCII data in `*.printpn0` files for HBOSC and HBNOISE. By extending the E and G voltage-controlled source syntax, the phase noise data in ASCII phase noise files can used as input for specifying behavioral noise sources

Usage Model

The syntax for the voltage controlled voltage (E) or current (G) source is as follows:

```
Exxx node1 node2 noise file='filename' [mname='measname']
Gxxx node1 node2 noise file='filename' [mname='measname']
```

Where `file='filename'` is the name of the ASCII phase noise data file. The file name is typically designated as 'design.printpn0', for a .HBOSC phase noise analysis or .HBNOISE analysis.

`mname='measname'` is used to select the appropriate noise measurement name to be taken from the **.printpn0* file.

measname can be one of the following:

- NLP_L(f) selects the nlp_L(f) phase noise data in units of dBc/Hz
- PAC_L(f) - selects the pac_l(f) phase noise data in units of dBc/Hz
- BPN_L(f) - selects the bpn_l(f) phase noise data in units of dBc/Hz
- ONOISE - selects the onoise data based on .HBNOISE or .SNNOISE analysis

Calculating Power Measurements After HB Analyses

Two types of power measurements are available: dissipated power in resistors and delivered power to port elements. The following subtle differences between these two measurements are described in this section.

Power Dissipated in a Resistor

All power calculations make use of the fundamental phasor power relationship given as the following equation, where voltage V and current I are complex phasors given in peak values (not rms, nor peak-to-peak):

Equation 19
$$P_{rms} = \frac{1}{2} Re\{VI^*\}$$

In the case of a simple resistor, its current and voltage are related according to $V_n=I_nR$. The power dissipated in a resistor of (real) value R at frequency index n is then given by:

Equation 20
$$P_{rms}(resistor)[n] = \frac{|V_n|^2}{2R}$$

Power Delivered to a Port Element

The port element can be either a source or sink for power. You can use a special calculation that computes the power flowing into a port element even if the port element itself is the source of that power. In the following figure is a port element connected to a circuit (the port element may or may not include a voltage source).

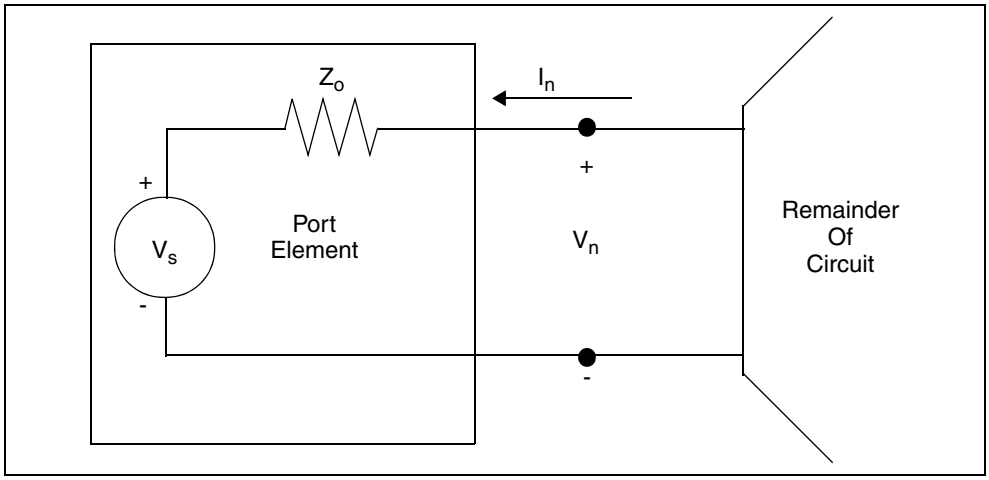


Figure 17 Port Element

Let V_n be the (peak) voltage across the terminals of the port element (at frequency index n). Let I_n be the (peak) current into the (1st) terminal of the port element (at frequency index n). Let Z_o be the impedance value of the z0 port element. Then, the power wave flowing into the terminals of the port element (at frequency index *n*) can be computed according to:

Equation 21
$$P_{in}[n] = \frac{1}{2}\left|\frac{V_n + Z_o I_n}{2\sqrt{Z_o}}\right|^2$$

This power expression remains valid whether or not the port element includes an internal voltage source at the same frequency. If the port element includes a voltage source at the same frequency, you can use this power calculation to compute the magnitude of the related large-signal scattering parameters.

If you expand the preceding formula, the power delivered to a port element with (real) impedance Z_o is given by

Equation 22
$$P_{rms}(port)[n] = \frac{1}{2}\left\{\frac{|V_n|^2 + Z_o^2 |I_n|^2}{4Z_o} + \frac{1}{2}Re\{V_n I_n^* \}\right\}$$

This power value represents the power incident upon and delivered to the port element's load impedance (Zo) due to other power sources in the circuit, and due to reflections of its own generated power.

If the port element is used as a load resistor (no internal source), the preceding equation reduces to that for the simple resistor.

If you used the port element as a power source (with non-zero available power, i.e. a non-zero V_s) and it is terminated in a matched load (Zo), the port power measurement returns 0 W, because no power is reflected.

You can request power measurements in the form of complete spectra or in the form of scalar quantities that represent power at a particular element. To request a complete power spectrum, use the following syntax.

```
.PRINT HB P(Elem)
.PROBE HB P(Elem)
```

To request a power value at a particular frequency tone, use the following syntax:

```
.PRINT HB P(Elem)[<n1<,n2<n3,...>>>]
.PROBE HB P(Elem)[<n1<,n2<,n3,...>>>]
```

The `Elem` is the name of either a Resistor (R) or Port (P) element, and *n1*,*n2*, and *n3* are integer indices used for selecting a particular frequency in the Harmonic Balance output spectrum.

Example 1

Prints a table of the RMS power (spectrum) dissipated by resistor R1.

```
.PRINT HB P(R1)
```

Example 2

Outputs the RMS power dissipated by resistor `R1` at the fundamental HB analysis frequency following a one-tone analysis.

```
.PROBE HB P(R1)[1]
```

Example 3

Prints the power dissipated by resistor `R1` at DC following a one-tone analysis.

```
.PRINT HB P(R1)[0]
```

Example 4

Outputs the RMS power dissipated by resistor `R1` at the (low-side) 3rd order intermodulation product following an HB two-tone analysis.

```
.PROBE HB P(R1)[2,-1]
```

Example 5

Prints the RMS power dissipated by resistor `R1` at the (high-side) 3rd order intermodulation product following an HB two-tone analysis.

```
.PRINT HB P(R1)[-1,2]
```

Example 6

Outputs the RMS power (spectrum) delivered to port element Pload.

```
.PROBE HB P(Pload)
```

Example 7

Prints the RMS power delivered to port element `Pload` at the fundamental HB analysis frequency following a one-tone analysis.

```
.PRINT HB P(Pload)[1] $
```

Example 8

Outputs the RMS power delivered to port element Pload at the (low-side) 3rd order intermodulation product following an HB two-tone analysis.

```
.PROBE HB P(Pload)[2,-1]
```

Calculations for Time-Domain Output

In addition to a frequency-domain output, HB analysis also supports a time-domain output. The equivalent time-domain waveform is generated according to the Fourier series expansion given by

Equation 23

$$V(n1)@\text{time } t = \text{SUM}_{\text{OVER}m}(\text{REAL}V(n1)[m]) \bullet \cos""(\Omega[m] \bullet t) - \text{IMAG}(\text{V}(n1)[m]\sin\Omega[m \bullet t] \quad \bullet t$$

Where *m* starts from 0 to the number of frequency points in the HB simulation.

The output syntax is

```
.PRINT [HBTRAN | HBTR] V(n1)
.PROBE [HBTRAN | HBTR] V(n1)
```

The output time ranges from 0 to twice the period of the smallest frequency in the HB spectra.

Minimizing Gibbs Phenomenon

You can use the HB_GIBBS option for HBTRAN output to minimize Gibbs' phenomenon that may occur in transforming a square-wave signal from the

frequency domain to the time domain. The syntax is `.OPTION HB_GIBBS=`*n* (defaults to zero, which is equivalent to not using it at all).

The result is that the HBTRAN waveforms are filtered by a $(\sin c(x))^N$ function before being transformed to the time domain via FFT. This option applies only to single-tone output. For example:

```
.option hb_gibbs = 2
...
.print hbtran v(2)
```

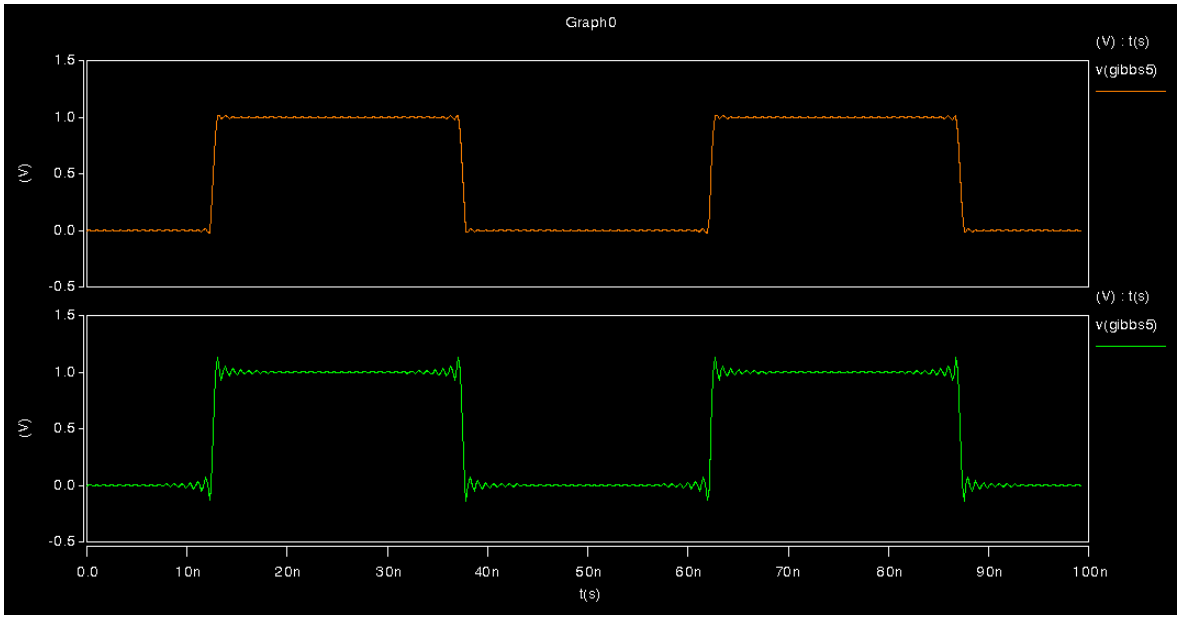


Figure 18 Upper square-wave signal shows HB_GIBBS = 2, while the lower shows the option = 0

Output Examples

```
.PRINT HB P(rload)       $ RMS power (spectrum)
                         $ dissipated at the rload resistor
.PROBE HB V(n1,v2)       $ Differential voltage (spectrum)
                         $ between the n1,n2 nodes
.PRINT HB VP(out)[1]     $ Phase of voltage at the out
                         $ node, at the fundamental
                         $ frequency
.PROBE HB P(Pout)[2,-1] $ RMS power delivered to the Pout
                         $ port, at third-order intermod
.PRINT HBTRAN V(n1)      $ Voltage at n1 in time domain
.PROBE HBTRAN V(n1, n2) $ Differential voltages between n1
                         $ and n2 node in time domain.
```

Using .MEASURE with .HB Analyses

- For transient analysis (TRAN), the independent variable for calculating .MEASURE is time.
- For AC analysis, the independent variable for calculating .MEASURE is frequency.
- However, as with DC analysis, the use of a .MEASURE command is peculiar for HB analysis, because it has no obvious independent variable.

In HSPICE RF, the independent variable for HB .MEASURE analysis is the first swept variable specified in the .HB simulation control statement. This variable can be anything: frequency, power, voltage, current, a component value, and so on.

Example 1

For the following .HB simulation control statement, the independent variable is the swept tone frequency, and the .MEASURE command values return results based on this frequency sweep:

```
* HARMONIC BALANCE tone-frequency sweep for amplifier
.param freq1=1.91e9 power=1e-3
.HB tones=freq1 nharms=10 sweep freq1 LIN 10 1.91e9 2.0e9
.MEASURE HB Patf0 FIND P(Rload)[1] AT=1.95e9 $ Power at
+ f0=1.95Ghz
.MEASURE HB Frq1W WHEN P(Rload)[1]=1. $ freq1 @ 1 Watt
.MEASURE HB BW1W TRIG AT=1.92e9 TARG P(Rload)[1] VAL=1.
+ CROSS=2 $ 1 Watt bandwidth
.MEASURE HB MaxPwr MAX P(Rload)[1] FROM=1.91e9 TO=2.0e9
+ $ Finds max output power
.MEASURE HB MinPwr MIN P(Rload)[1] FROM=1.91e9 TO=2.0e9
+ $ Finds min output power
```

Example 2

In the following example, the independent variable is the *power* variable, and the .MEASURE values return results based on the power sweep. Units are in Watts.

```
* HARMONIC BALANCE power sweep for amplifier
.param freq1=1.91e9 power=1e-3
.HB tones=freq1 nharms=10 sweep power DEC 10 1e-6 1e-3
.MEASURE HB Pat1uW FIND P(Rload)[1] AT=1e-6 $ Pout at 1uW
.MEASURE HB Pin1W WHEN P(Rload)[1]=1. $ Pin @ 1 Watt Pout
.MEASURE HB Prange1W TRIG AT=1.92e9 TARG P(Rload)[1] VAL=1.
+ CROSS=2    $ 1W oper. range

.MEASURE HB ssGain DERIV P(Rload)[1] AT=1e-5
+ $ relative power gain at 10uW input
.MEASURE HB Gain3rd DERIV P(Rload)[3] AT=1e-5
+ $ 3rd harmonic gain at 10uW input
.MEASURE HB PAE1W FIND '(P(Rload)[1]-power)/P(Vdc)[0]'
+ WHEN P(Rload)[1]=1 $ PAE at 1 Watt output
```

Example 3

In this example, the independent variable is again the *power* variable, and the .MEASURE values return results based on the power sweep. This is a two-tone sweep, where both input frequency sources are at the same power level in Watts.

```
* HARMONIC BALANCE two-tone sweep for amplifier
* An IP3 calculation is made at 10uW in the sweep
.param freq1=1.91e9 freq2=1.91e9 power=1e-3
.HB tones=freq1,freq2 nharms=6,6 sweep power DEC 10 1e-6 1e-3
.MEASURE HB Pf1dBm FIND '10.*LOG(P(Rload)[1,0]/1.e-3)'
+ AT=1e-5 $ P(f1) at 10uW input
.MEASURE HB P2f1_f2dBm FIND '10.*LOG(P(Rload)[2,-1]/1.e-3)'
+ AT=1e-5 $ P(2f1-f2) at 10uW input
.MEASURE HB OIP3dBm PARAM = '0.5*(3.*Pf1dBm-P2f1_f2dBm)'
.MEASURE HB IIP3dBm PARAM = 'OIP3dBm-Pf1dBm+20.0'
.MEASURE HB AM2PM DERIV VP(outp,outn)[1] AT=1e-5
+ $ AM to PM Conversion in Deg/Watt
```

If you do not specify an HB sweep, then .MEASURE assumes a single-valued independent variable sweep.

You can apply the measurements to current, voltage, and power waveforms. The independent variable for measurements is the swept variable (such as power), not the frequency axis corresponding to a single HB steady state point.

HSPICE RF also supports the .MEASURE [HBTRAN | HBTR] . . . syntax. Similar to the .PROBE and .PRINT HBTR statements in the section Calculations for Time-Domain Output on page 122, a .MEASURE HBTR statement is applied on the signals obtained in the same way. Moreover, like a .MEASURE statement in transient analysis, the independent variable in a .MEASURE HBTR statement is time.

HSPICE RF optimization can read the data from .MEASURE HB and .MEASURE HBTR statements. The optimization syntax in HSPICE RF is identical to that in the HSPICE. Due to the difference in the independent variable between the .MEASURE HB and .MEASURE HBTR statements, these two types of measurements cannot be mixed in a HSPICE RF optimization. But a .MEASURE HBTR statement can be combined with a .MEASURE PHASENOISE statement (see Measuring Phase Noise with .MEASURE PHASENOISE on page 172) and a .MEASURE HBNOISE statement (see Measuring HBNOISE Analyses with .MEASURE on page 206) in a HSPICE RF optimization flow.

HB Output Data Files

The results of an HB analysis are complex spectral components at each frequency point. The a[i] is the real part, and b[i] is the imaginary part of the complex voltage at frequency index i. The conversion to a steady state time-domain is then given by the Fourier series expansion.

An HB analysis produces these output data files:

- Output from the `.PRINT HB` statement is written to a .printhb# file.
  - The header contains the large signal fundamental frequencies.
  - The columns of data are labeled as `HERTZ`, followed by frequency indices, and then the output variable names.
  - The sum of the frequency indices, multiplied by the corresponding fundamental frequencies, add up to the frequency in the first column.
- Output from the `.PROBE HB` statement is written to a .hb# file. It is in the same format as the HSPICE transient analysis .tr# file. Besides the output waveform, it contains the information of harmonic indices and basic tone frequencies.
- Output from the `.PRINT HBTRAN` statement is written to a .printhr# file. The format is identical to a .print# file.
- Output from the `.PROBE HBTRAN` statement is written to a .hr# file. The format is identical to a *.tr#* file.
- Reported performance log statistics are written to a .lis file:
  - Name of HB data file.
  - Simulation time:

    DC operating point (op) time

    HB time

    Total simulation time
  - Memory used
  - Size of matrix (nodes * harmonics)
  - Final HB residual error

Errors and Warnings

Table 5 lists the errors messages and Table 6 on page 129 lists the warning messages.

Table 5 HB Analysis Error Messages

File	Description
HB_ERR.1	Harmonic numbers must be positive non-zero.
HB_ERR.2	No .hb frequencies given.
HB_ERR.3	Negative frequency given.
HB_ERR.4	Number of harmonics should be greater than zero.
HB_ERR.5	Different number of tones, nharms.
HB_ERR.6	Bad probe node format for oscillator analysis.
HB_ERR.7	Bad format for FSPTS.
HB_ERR.8	Bad .hb keyword.
HB_ERR.9	Tones must be specified for .hb analysis.
HB_ERR.10	Nharms or intmodmax must be specified for .hb analysis.
HB_ERR.11	Source harmonic out of range.
HB_ERR.12	Source named in the tones list is not defined.
HB_ERR.13	Source named in the tones list does not have TRANFORHB specified.
HB_ERR.14	Source named in the tones list has no transient description.
HB_ERR.15	Source named in the tones list must be HB, SIN, PULSE, PWL, or VMRF.
HB_ERR.16	Tone specification for the source is inconsistent with its frequency.
HB_ERR.17	HB oscillator analysis has reached the NULL solution.

Table 5 HB Analysis Error Messages (Continued)

File	Description
HB_ERR.18	Bad subharms format.
HB_ERR.19	Modtone may not be set to the same value as tone.

Table 6 HB Analysis Warning Messages

File	Description
HB_WARN.1	.hb multiply defined. Last one will be used.
HB_WARN.2	Tone specified for V/I source not specified in .HB command.
HB_WARN.3	HB convergence not achieved.
HB_WARN.4	Source specifies both HB and transient description. HB description will be used.
HB_WARN.5	Source specifies exponential decay. HB will ignore it.
HB_WARN.6	Source specifies a non-positive frequency.
HB_WARN.7	Source does not fit the HB spectrum.
HB_WARN.8	Source cannot be used with the TRANFORHB option.
HB_WARN.9	Frequency not found from transient analysis
HB_WARN.10	.hb/.hbosc will be ignored due to .env/.envosc.
HB_WARN.11	HBTRANINIT does not support more than one input tone.

References

[1] S. Maas, *Nonlinear Microwave Circuits*, Chapter 3, IEEE Press, 1997.

[2] R. Gilmore and M.B. Steer, “Nonlinear Circuit Analysis Using the Method of Harmonic Balance - A Review of the Art, Part I, Introductory Concepts.” *International Journal of Microwave and Millimeter-wave Computer-Aided Engineering*, Volume 1, No. 1, pages 22-37, 1991.

[3] R. Gilmore and M.B. Steer, “Nonlinear Circuit Analysis Using the Method of Harmonic Balance - A Review of the Art. Part II. Advanced Concepts.” *International Journal of Microwave and Millimeter-wave Computer-Aided Engineering*, Volume 1, No. 2, pages 159-180, 1991.

[4] V. Rizzoli, F. Mastri, F. Sgallari, G. Spaletta, “Harmonic-Balance Simulation of Strongly Nonlinear Very Large-Size Microwave Circuits by Inexact Newton Methods,” *MTT-S Digest*, pages 1357-1360, 1996.

[5] S. Skaggs, *Efficient Harmonic Balance Modeling of Large Microwave Circuits*, Ph.D. thesis, North Carolina State University, 1999.

[6] R.S. Carson, *High-Frequency Amplifiers*, 2nd Edition, John Wiley & Sons, 1982

[7] S.Y. Liao, *Microwave Circuit Analysis and Amplifier Design*, Prentice-Hall, 1987.

[8] J. Roychowdhury, D. Long, P. Feldmann, “Cyclostationary Noise Analysis of Large RF Circuits with Multitone Excitations”, *IEEE JSCC*, volume 33, number 3, March 1998.

[9] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, 1995.

[10] J. Roychowdhury, D. Long, and P. Feldmann, “Cyclostationary Noise Analysis of Large RF Circuits with Multitone Excitations,” *IEEE Journal of Solid-State Circuits*, volume 33, pages 324–336, March 1998.

[11] K. Kurakawa, “Power waves and the Scattering Matrix,” IEEE Trans. *Microwave Theory Tech.*, vol. MTT-13, pp. 194-202, March 1965.

6

Steady-State Shooting Newton Analysis

Describes HSPICE RF steady-state time domain analysis based on a Shooting-Newton algorithm.

These topics are covered in the following sections:

- SN Steady-State Time Domain Analysis
- SN Analysis Syntax
- SN Analysis Output
- Shooting Newton with Fourier Transform (.SNFT)

SN Steady-State Time Domain Analysis

An advanced Shooting Newton (SN) algorithm provides additional performance and functionality to HSPICE RF for time-domain, steady-state analysis.

Shooting-Newton adds analysis capabilities for PLL components, digital circuits/logic, such as ring oscillators, frequency dividers, phase/frequency detectors (PFDs), and for other digital logic circuits and RF components that require steady-state analysis, but operate with waveforms that are more square wave than sinusoidal.

The Shooting-Newton algorithm effectively analyzes applications including:

- Ring oscillators (see Chapter 7, Oscillator and Phase Noise Analysis)
- Frequency dividers (prescalers)
- Mixer conversion gain
- Phase-frequency detectors (PFDs)
- Mixer noise figure

Functionality includes:

- Both driven and oscillator (autonomous) analyses
- Time Domain or Frequency analysis based on advanced Shooting Newton algorithm
- Spectrum analysis specific to the SN analysis (see Shooting Newton with Fourier Transform (.SNFT) on page 137)
- Shooting Newton-based AC analysis (SNAC) (see Shooting Newton AC Analysis (.SNAC) on page 195)
- Shooting Newton-based noise analysis (SNNOISE) (see Oscillator Analysis Using Shooting Newton (.SNOSC) on page 158)
- Shooting Newton-based phase noise analysis (PHASENOISE) (see Oscillator and Phase Noise Analysis on page 143)

SN Analysis Syntax

Shooting Newton provides two syntaxes. Syntax #1 is recommended when you are using/making Time Domain sources and measurements (for example, going from .TRAN to .SN). Syntax #2 effectively supports Frequency Domain sources and measurements (and should be used, for example, when going from .HB to .SN).

Syntax #1

```
.SN TRES=Tr PERIOD=T [TRINIT=Ti]
+ [SWEEP parameter_sweep] [MAXTRINITCYCLES=integer]
```

or, Syntax #2

```
.SN TONE=F1 NHARMS=N [TRINIT=Ti]
+ [SWEEP parameter_sweep] [MAXTRINITCYCLES=integer]
```

where

Parameter	Description
TRES	The time resolution to be computed for the steady-state waveforms (in seconds).

Parameter	Description
PERIOD	The expected period T (seconds) of the steady-state waveforms. Enter an approximate value when using for oscillator analysis. The period of the steady-state waveform may be entered either as PERIOD or its reciprocal, TONE.
TONE	The fundamental frequency (in Hz).
NHARMS	Specifies the number of high-frequency harmonic components to include in the analysis. NHARMS defaults to PERIOD/TRES rounded to nearest integer. NHARMS is required to run subsequent SNAC, SNNOISE, SNXF, and PHASENOISE analyses. When using Syntax #1, NHARMS is computed automatically as NHARMS=Round(PERIOD/TRES).
TRINIT	This is the transient initialization time. If not specified, the transient initialization time will be equal to the period (for Syntax 1) or the reciprocal of the tone (for Syntax 2).
SWEEP	Specifies the parameter sweep. As in all main analyses in HSPICE RF such as .TRAN, .HB, etc., you can specify LIN, DEC, OCT, POI, SWEEPBLOCK, DATA, MONTE, or OPTIMIZE.
MAXTRINITCYCLES	Stops SN stabilization simulation and frequency detection when the simulator detects that maxtrinitcycles have been reached in the oscnode signal, or when time=trinit, whichever comes first. Minimum cycles is 1.

Options

In addition to all .TRAN options, .SN analysis supports the following options.

Option	Default	Description
.OPTION SNMAXITER | SN_MAXITER=*integer*	40	Maximum number, Shooting-Newton iterations

Option	Default	Description
.OPTION SNCONTINUE= 1|2	1	Specifies whether to use the sweep solution from the previous simulation as the initial guess for the present simulation. ▪ SNCONTINUE=1 (default): Use solution from previous simulation as the initial guess. ▪ HBCONTINUE=0: Start each simulation in a sweep from the DC solution.
.OPTION SNACCURACY=*integer*	10	Similar to the sim_accuracy definition in.TRAN, i.e., larger values of snaccuracy result in a more accurate solution but may require more time points. Because Shooting-Newton must store derivative information at every time point, the memory requirements may be significant if the number of time points is very large. The maximum integer value is 50.
.OPTION LOADSNINIT="*filename*"		Loads the operating point saved at the end of SN initialization which is used as initial conditions for the Shooting-Newton method.
.OPTION SAVESNINIT="*filename*"		Saves the operating point at the end of SN initialization (sninit).

SN Analysis Output

The output from .SN analysis is generated in both time and frequency domains.

The time domain output variables are the same as for standard transient analysis:

- individual nodal voltages: *V*(*n1* [,*n2*])
- branch currents: *I*(*Vxx*)
- element power dissipation: *In*(*element*)

It is also possible to output the results from Shooting Newton analysis in terms of complex, frequency-domain output variables. This output format is activated by using the “SNFD” keyword in the output syntax.

For output in the frequency domain, the syntax is identical to the Harmonic Balance output syntax:

```
.PRINT SNFD TYPE(NODES | ELEM)[INDICES]
.PROBE SNFD TYPE(NODES | ELEM)[INDICES]
```

Parameter Description

TYPE Specifies a harmonic type node or element. TYPE can be one of the following:

- Voltage type –
- V = voltage magnitude and phase in degrees
- VR = real component
- VI = imaginary component
- VM = magnitude
- VP - Phase in degrees
- VPD - Phase in degrees
- VPR - Phase in radians
- VDB - dB units
- VDBM - dB relative to 1 mV

- Current type –
- I = current magnitude and phase in degrees
- IR = real component
- II = imaginary component
- IM = magnitude
- IP - Phase in degrees
- IPD - Phase in degrees
- IPR - Phase in radians
- IDB - dB units
- IDBM - dB relative to 1 mV
- Power type – P
- Frequency type –
- `hertz[index], hertz[index1, index2, ...]`

You must specify the harmonic index for the hertz variable. The frequency of the specified harmonics is dumped.

Parameter Description

`NODES | ELEM` can be any of the following:

- Voltage type – a single node name (n1), or a pair of node names, (n1,n2)
- Current type – an element name (elemname)
- Power type – a resistor (resistorname) or port (portname) element name
- INDEX n1, is the harmonic index of the SNFD tone. Index is limited to the single tone associated with the SN analysis.

Output Files

The time domain data are output to *printsn0* and *.sn0* files. Frequency domain data are output to *.printsnf0* and *.snf0* files.

Output Format

The format for time domain output is the same as standard transient analysis. For frequency domain output, the format is similar to HB. The main difference is that Shooting Newton output in the frequency domain is single tone only.

The results of an SN analysis are complex spectral components at each frequency point. The a[i] is the real part, and b[i] is the imaginary part of the

complex voltage at frequency index i. The conversion to a steady state time-domain is then given by the Fourier series expansion.

An SN analysis produces these output data files:

- Output from the `.PRINT SN` statement is written to a *.printsn#* file.
  - The header contains the large signal fundamental frequencies.
  - The columns of data are labeled as `HERTZ`, followed by frequency indices, and then the output variable names.
  - The sum of the frequency indices, multiplied by the corresponding fundamental frequencies, add up to the frequency in the first column.
- Output from the `.PROBE SN` statement is written to a *.sn#* file in the same format as the HSPICE transient analysis .tr# file. It contains the information of harmonic indices and basic tone frequencies plus the output waveform.
- Reported performance log statistics are written to a *.lis* file:
  - Name of SN data file.
  - Simulation time:

    DC operating point (op) time

    SN time

    Total simulation time
  - Memory used
  - Size of matrix (nodes * harmonics)
  - Final SN residual error

Shooting Newton with Fourier Transform (.SNFT)

The .SNFT command is to the .SN analysis what .FFT is to the TRAN analysis, a means to provide spectrum analysis. Spectrum analysis represents a time-domain signal, within the frequency domain. .SNFT uses the Fourier transform: a Discrete Fourier Transform (DFT) uses sequences of time values to determine the frequency content of analog signals, in circuit simulation. The `.SNFT` statement uses the internal time point values.

By default, the `.SNFT` statement uses a second-order interpolation to obtain waveform samples, based on the number of points that you specify.

You can use windowing functions to reduce the effects of waveform truncation on the spectral content. You can also use the `.SNFT` command to specify:

- output format
- frequency
- number of harmonics
- total harmonic distortion (THD)

.SNFT Input Syntax

The .SNFT command an take arguments with either alphanumeric or numerics and expressions.

Syntax # 1 Alphanumeric input

```
.SNFT output_var [START=val] [STOP=val]
+ [NP=value] [FORMAT=keyword]
+ [WINDOW=keyword] [ALFA=val]
+ [FREQ=val] [FMIN=val] [FMAX=val]
```

Syntax #2 Numerics and expressions

```
.SNFT output_var [START=param_expr1] [STOP=param_expr2]
+ [NP=param_expr3] [FORMAT=keyword]
+ [WINDOW=keyword] [ALFA=param_expr4]
+ [FREQ=param_expr5] [FMIN=param_expr6] [FMAX=param_expr7]
```

Arguments

Argument	Description
output_var	Can be any valid output variable, such as voltage, current, or power.
START	Start of the output variable waveform to analyze. Defaults to the START value in the .SN statement, which defaults to 0.
FROM	An alias for START in .SNFT statements.
STOP	End of the output variable waveform to analyze. Defaults to the TSTOP value in the .SN statement.
TO	An alias for STOP, in .SNFT statements.

Argument	Description
NP	Number of points to use in the SNFT analysis. NP must be a power of 2. If NP is not a power of 2, HSPICE automatically adjusts it to the closest higher number that is a power of 2. The default is 1024.
FORMAT	Specifies the output format: ■ NORM= normalized magnitude (default) ■ UNORM=unnormalized magnitude
WINDOW	Specifies the window type to use: ■ RECT=simple rectangular truncation window (default). ■ BART=Bartlett (triangular) window. ■ HANN=Hanning window. ■ HAMM=Hamming window. ■ BLACK=Blackman window. ■ HARRIS=Blackman-Harris window. ■ GAUSS=Gaussian window. ■ KAISER=Kaiser-Bessel window.
ALFA	Parameter to use in GAUSS and KAISER windows to control the highest side-lobe level, bandwidth, and so on. 1.0 <= ALFA <= 20.0 The default is 3.0
FREQ	Frequency to analyze. If FREQ is non-zero, the output lists only the harmonics of this frequency, based on FMIN and FMAX. HSPICE also prints the THD for these harmonics. The default is 0.0 (Hz).
FMIN	Minimum frequency for which HSPICE prints SNFT output into the listing file. THD calculations also use this frequency. T=(STOP-START) The default is 1.0/T (Hz).
FMAX	Maximum frequency for which HSPICE prints SNFT output into the listing file. THD calculations also use this frequency. The default is 0.5*NP*FM IN (Hz).

Example 1

```
.SNFT v(1)
.SNFT v(1,2) np=1024 start=0.3m stop=0.5m freq=5.0k
+ window=kaiser alfa=2.5
.SNFT I(rload) start=0m to=2.0m fmin=100k fmax=120k
+ format=unorm
.SNFT par('v(1) + v(2)') from=0.2u stop=1.2u
+ window=harris
```

Example 2

```
.SNFT v(1) np=1024
.SNFT v(2) np=1024
```

This example generates an .snft0 file for the SNFT of v(1) and an .snft1 file for the SNFT of v(2).

.SN Signal Sources

.SN analysis assumes that all stimuli are periodic with period T. If the circuit is driven with more than one periodic stimulus, then the frequencies must be all co-periodic and T must match the common period or some integer multiple of it. The .SN analysis only supports .tran (time-domain) periodic signal sources. (Refer to the .tran analysis for a detailed documentation on transient signal sources).

.SN Reported Performance Log Statistics

The following performance statistics are displayed:

- DC operating time
- Initial transient time (including user's time for circuit stabilization)
- Total simulation time
- SN time
- Total simulation time
- Memory used
- Final SN convergence residual error
- The value of the computed frequency if the circuit is autonomous

Errors/Warnings

Error messages are displayed with convergence recommendations in cases of non-convergence within the maximum number of Shooting-Newton iterations.

Error messages are displayed for software errors such as segmentation violations, and abort conditions such as:

- unrecognized format, i.e., unrecognized V/I source
- faulty input values, i.e., wrong sign, out of range value
- unspecified values, i.e., unspecified tone
- inconsistent values, i.e., non-commensurable tones
- duplicate values, i.e., same entry, given more than one, the last one is always taken

Limitations and Assumptions

True distributed components (such as ideal delays or transmission lines) are not supported; components with hidden states are not supported.

Example

For a demonstration of using Shooting Newton analysis you can run the *pdfcpGain.sp* file shipped with the HSPICE RF distribution, located in the directory $*installdir*/hspicerf/examples. This example performs analysis on a D-flipflop phase frequency divider with charge pump, implemented in 50nm technology. The example is configured to measure the gain (volts per degree) of the DFF PFD and tri-state output combination.

7

Oscillator and Phase Noise Analysis

Describes how to use HSPICE RF to perform oscillator and phase noise analysis on oscillator circuits.

These topics are covered in the following sections:

- Harmonic Balance or Shooting Newton for Oscillator Analysis
- Harmonic Balance Oscillator Analysis (.HBOSC)
- Input Syntax for Harmonic Balance Oscillator Analysis
- HB Simulation of Ring Oscillators
- HBOSC Analysis Using Transient Initialization
- Oscillator Analysis Using Shooting Newton (.SNOSC)
- Phase Noise Analysis (.PHASENOISE)
- Accumulated Jitter Measurement for Closed Loop PLL Analysis
- Small-Signal Phase-Domain Noise Analysis (.ACPHASENOISE)

Harmonic Balance or Shooting Newton for Oscillator Analysis

Oscillator Classification

Oscillators can be divided into two main categories:

1. Ring oscillators: These oscillators tend to have low Q and operate based on delay of digital cells such as inverters. Ring oscillators have strong nonlinear behavior and output signals are often square-wave-like. Ring oscillators can be analyzed in either the frequency domain using Harmonic Balance analysis or in the time domain using Shooting Newton analysis.
2. Harmonic oscillators: Common harmonic oscillators are LC and crystal oscillators. These oscillators tend to have a high Q, making it difficult to find the oscillation frequency. Their behavior tends to be only mildly nonlinear and their output signals tend to be close to purely sinusoidal. Harmonic Balance analysis is most effective for analyzing harmonic oscillators.

HSPICE RF includes special analysis algorithms for finding the steady-state solution for oscillator circuits. In oscillators, there are no driving sources that set the frequencies of operation, but rather the fundamental oscillation frequency is one of the unknowns that is being solved by the simulator. HSPICE RF provides two approaches: either harmonic balance or analysis based on the Shooting Newton algorithm.

The following sections are presented in this chapter:

- Harmonic Balance Oscillator Analysis (.HBOSC)
- Oscillator Analysis Using Shooting Newton (.SNOSC)
- Phase Noise Analysis (.PHASENOISE)

Harmonic Balance Oscillator Analysis (.HBOSC)

Because the frequency of oscillation is not determined by the frequencies of driving sources, a slightly different set of nonlinear equations are solved during the simulation and are as shown in the following equation:

Equation 24 $$F(V,\ \omega_0) = I(V,\ \omega_0) + \Omega Q(V,\ \omega_0) + Y(\omega_0)V + I_s$$

HSPICE harmonic balance oscillator analysis (.HBOSC) adds the fundamental frequency of oscillation to the list of unknown circuit quantities. To accommodate the extra unknown, the phase (or equivalently, the imaginary part) of one unknown variable (generally a node voltage) is set to zero. The phases of all circuit quantities are relative to the phase, at this reference node (referred to as the "PROBENODE").

Additionally, the HBOSC analysis tries to avoid the "degenerate solution," where all non-DC quantities are zero. Although this is a valid solution of the above equation (it is the correct solution, if the circuit does not oscillate), HBOSC analysis might find this solution incorrectly, if the algorithm starts from a bad initial solution.

The HBOSC analysis follows a technique similar to that described by Ngoya, et al, which uses an internally-applied voltage probe to find the oscillation voltage and frequency. The source resistance of this probe is a short circuit at the oscillation frequency, and an open circuit otherwise. HSPICE RF uses a two-tier Newton approach to find a non-zero probe voltage, which results in zero probe current.

The DC solution is used as a starting point for the HB analysis of the oscillator circuit. In addition to the DC solution, initial values for both the oscillation frequency and the probe voltage are needed. HBOSC analysis calculates the small-signal admittance that the voltage probe sees over a range of frequencies in an attempt to find potential oscillation frequencies. Oscillation is likely to occur where the real part of the probe current is negative, and the imaginary part is zero. You can use the FSPTS parameter to specify the frequency search. You must also supply an initial guess for the large signal probe voltage. A value of one-half the supply voltage is often a good starting point.

Input Syntax for Harmonic Balance Oscillator Analysis

The input syntax for HBOSC analysis supports two different formats, depending on whether the PROBENODE location is specified using a circuit element (current source) or using the HBOSC PROBENODE parameters:

Syntax #1

```
.HBOSC TONE=F1 NHARMS=H1
+ PROBENODE=N1,N2,VP
+[FSPTS=NUM, MIN, MAX] [STABILITY=(-2|-1|0|1|2)]
+[SWEEP PARAMETER_SWEEP] [SUBHARMS=I]
```

Syntax #2 (Uses current source to set PROBENODE)

```
ISRC N1 N2 HBOSCVPROBE=VP
.HBOSC TONE=F1 NHARMS=H1
+[FSPTS=NUM, MIN, MAX] [STABILITY=(-2|-1|0|1|2)]
+[SWEEP PARAMETER_SWEEP] [SUBHARMS=I]
```

Parameter	Description
TONE	Approximate value for oscillation frequency (Hz). The search for an exact oscillation frequency begins from this value, unless you specify an FSPTS range or transient initialization (see HB Simulation of Ring Oscillators on page 150 for more information).
NHARMS	Number of harmonics to use for oscillator HB analysis.
PROBENODE	Circuit nodes that are probed for oscillation conditions. ■ N1 and N2 are the positive and negative nodes for a voltage probe inserted in the circuit to search for oscillation conditions. ■ VP is the initial probe voltage value (one-half the supply voltage is a suggested value). The phase of the probe voltage is forced to zero; all other phases are relative to the probe phase. HSPICE RF uses this probe to calculate small-signal admittance for the initial frequency estimates. It should be connected near the “heart” of the oscillator (near resonators, inside the ring of a ring oscillator, etc.). Note: The PROBENODE pins and approximate voltage value can also be set by using a zero amp current source that uses the HBOSCVPROBE keyword.
HBOSCVPROBE=VP	Sets PROBENODE parameters with a separate current source element. If a current source with HBOSCVPROBE is used, the PROBENODE parameter and its values need not be included in the .HBOSC command.
FSPTS	Specifies the frequency search points that HSPICE RF uses in its initial small-signal frequency search to find an oscillation frequency. Optional, but recommended for high-Q and most LC oscillators. If the circuit is a ring oscillator, see HB Simulation of Ring Oscillators on page 150 for more information on how to use the HBTRANINIT option. ■ NUM is an integer. ■ MIN and MAX are frequency values in units of Hz. If the FSPTS analysis finds an approximate oscillation frequency, the TONE parameter may be ignored.

Parameter	Description
STABILITY	When used with FSPTS, activates the additional oscillator stability analyses depending on the following values: ■ 0: A single point oscillator frequency-search stability analysis is performed. The FSPTS search is executed, and the first successful linear oscillation frequency value found is used as the starting point for the two-tier Newton nonlinear oscillator analysis. The probenode vp value specified is used as the starting amplitude for the Newton solver. ■ 1: (default) A single point oscillator frequency-search stability analysis, plus an estimate of oscillator amplitude, is performed. The FSPTS search is executed, and the first successful linear oscillation frequency value found is used as the starting point for the two-tier Newton nonlinear oscillator analysis. An additional analysis for automatically estimating the probenode amplitude is also performed, and this value is used as the starting amplitude for the two-tier Newton solver. ■ –1: A single point oscillator frequency-search stability analysis, plus an estimate of oscillator amplitude, is performed. The FSPTS search is executed, and the first successful linear oscillation frequency value found is accurately computed and reported. An additional analysis for automatically estimating the probenode amplitude is also performed, and this value is also reported. The analysis aborts without attempting the two-tier Newton nonlinear oscillator analysis. By using STABILITY=–1, a check can be made if any linear oscillation conditions are found, before attempting the nonlinear oscillator analysis. ■ 2: A multi-point frequency-search stability analysis is performed. The FSPTS search is executed, and all successful linear oscillation frequency values found over the entire FSPTS search range are reported. For each potential oscillation frequency found, an additional analysis for estimating the probenode amplitude is also performed. All frequency and amplitude values are reported. The frequency value that has the largest predicted amplitude is used as the starting point for the two-tier Newton nonlinear oscillator analysis. ■ –2: A multi-point frequency-search stability analysis is performed. The FSPTS search is executed, and all successful linear oscillation frequency values found over the entire FSPTS search range are reported. For each potential oscillation frequency found, an additional analysis for estimating the probenode amplitude is also performed. All frequency and amplitude values are reported. The analysis aborts without attempting the two-tier Newton nonlinear oscillator analysis. By using STABILITY=–2, a check can be made if any linear oscillation conditions are found, before attempting the nonlinear oscillator analysis.

Parameter	Description
SWEEP	Specifies the type of sweep. You can sweep up to three variables. You can specify either LIN, DEC, OCT, POI, SWEEPBLOCK, DATA, OPTIMIZE, or MONTE. Specify the nsteps, start, and stop frequencies using the following syntax for each type of sweep: ■ LIN *nsteps start stop* ■ DEC *nsteps start stop* ■ OCT *nsteps start stop* ■ POI *nsteps freq_values* ■ SWEEPBLOCK *nsteps freq1 freq2 ... freqn* ■ DATA=*dataname* ■ OPTIMIZE=OPT*xxx* ■ MONTE=*val*
SUBHARMS	Allows subharmonics in the analysis spectrum. The minimum non-DC frequency in the analysis spectrum is f/subharms, where f is the frequency of oscillation. Use this option if your oscillator circuit includes a divider or prescaler that will result in frequency terms that are subharmonics of the fundamental oscillation frequency

Note: You can specify either `.OPTION HBTRANPTS` or `.OPTION HBTRANSTEP`, but not both.

Simulation Strategies for Harmonic Oscillators

Since harmonic oscillators tend to have high Q, they are more sensitive than ring oscillators to frequency and amplitude guess. The high Q also means that HBTRANINIT is not usually helpful, because it takes too long for transient simulation to settle close enough to steady state. For these oscillators, FSPTS tends to work well, because the behavior is close to linear.

The recommended setup for harmonic oscillators is:

- Set up .HBOSC with FSPTS. Higher Q oscillators may need more FSPTS points.
- Choose a node directly connected to the oscillator or crystal as the PROBENODE.

- The number of harmonics specified by nharms can be fairly small, perhaps 10, unless some digital circuitry is included in the simulation.
- Do not use HBTRANINIT.

  Because the challenges related to ring oscillators and harmonic oscillators are so different, we approach them with separate simulation strategies.

.HBOSC Examples

Example 1

```
.HBOSC tone=900MEG nharms=9 probenode=gate,gnd,0.65
```

Performs an oscillator analysis, searching for frequencies in the vicinity of 900 MHz. This example uses nine harmonics with the probe inserted between the `gate` and `gnd` nodes. The probe voltage estimate is `0.65` V.

Example 2

```
.HBOSC tone=2400MEG nharms=11
+ probenode=drainP,drainN,1.0 fspts=20,2100MEG,2700MEG
```

Performs an oscillator analysis, searching for frequencies in the vicinity of 2.4 GHz. This example uses `11` harmonics with the probe inserted between the `drainP` and `drainN` nodes. The probe voltage estimate is `1.0` V.

Example 3

Another means to define the probenode information is through a zero-current source. The following two methods define an equivalent `.HBOSC` command:

- Method 1:

  ```
  .HBOSC tone = 2.4G nharms = 10
  + probenode = drainP, drainN, 1.0
  + fspts = 20, 2.1G, 2.7G
  ```

- Method 2:

  ```
  ISRC drainP drainN 0 HBOSCVPROBE = 1.0
  .HBOSC tone = 2.4G nharms = 10
  + fspts = 20, 2.1G, 2.7G
  ```

In Method 2, the `PROBENODE` information is defined by a current source in the circuit. Only one such current source is needed, and its current must be 0.0 with the HBOSC `PROBENODE` voltage defined through its `HBOSCVPROBE` property.

HB Simulation of Ring Oscillators

Ring oscillators require a slightly different simulation approach in HB. Since their oscillation is due to the inherent delay in the inverters of the ring, they are best modeled in the time domain and not in the frequency domain.

In addition, ring oscillator waveforms frequently approach square waves, which require a large number of harmonics to be described in the frequency domain. An accurate initial guess is important if they are going to be simulated accurately with HB.

The HSPICE RF HBOSC analysis typically starts from the DC solution and looks for potential resonances in the linear portion of the circuit to determine the initial guess for the oscillation frequency. However, these resonances generally do not exist in ring oscillators, which do not contain linear resonant elements.

HB analysis provides a second method of obtaining a good initial guess for the oscillation frequency, which is specifically intended for ring oscillators. Instead of starting from the results of a DC analysis, this method starts from the result of a transient analysis. This method is called *Transient Initialization* and also provides a good initial guess for all the voltages and currents in the circuit.

The recommended setup for ring oscillators is therefore:

- Set up .HBOSC without FSPTS.
- Choose one of the nodes in the ring as the PROBENODE.
- Since ring oscillators tend to have square-wave-like output signals which have significant high frequency content, a relatively large value, perhaps 50, for nharms is recommended. Ring oscillators with more stages tend to need more harmonics.
- Set HBTRANINIT to a value that represents ~5-10 oscillator periods, and make sure that you include an .ic command or other transient analysis setup to start the oscillator in transient simulation. Longer HBTRANINIT times may result in faster HBOSC convergence, at the expense of additional CPU time spent on HBTRANINIT.

HBOSC Analysis Using Transient Initialization

To perform an HBOSC analysis, use the following options in your HSPICE RF netlist.

Table 7 HBOSC Analysis Options for Transient Initialization

Keyword	Description
HBTRANINIT = *time*	Tells HB to use transient analysis to initialize all state variables. *time* is when the circuit has reached (or is near) steady-state. Default = 0.
HBTRANPTS = *npts*	*npts* specifies the number of points per period for converting the time-domain data results from transient analysis, into the frequency domain. *npts* must be an integer greater than 0. The units are in nharms (nh). Default=4*nh. This option is relevant only if you set .OPTION HBTRANINIT.
HBTRANSTEP = *stepsize*	*stepsize* specifies the step size for the transient analysis. The default is 1/(4*nh*f0), where nh is the nharms value and f0 is the oscillation frequency. This option is relevant only if you set .OPTION HBTRANINIT.
HBTRANFREQSEARCH = *1|0*	If HBTRANFREQSEARCH=1 (default), then HB analysis calculates the oscillation frequency from the transient analysis. Otherwise, HB analysis assumes that the period is 1/*f*, where *f* is the frequency specified in the tones description.

You must also either specify the initial conditions or add a PWL or PULSE source to start the oscillator for transient analysis. This source should provide a brief stimulus, and then return to zero. HB analysis effectively ignores this type of source, treating it as zero-valued.

This method does the following:

1. If `HBTRANFREQSEARCH=1`, transient analysis runs for several periods, attempting to determine the oscillation frequency from the probe voltage signal.
2. Transient analysis continues until the time specified in `HBTRANINIT`.
3. Stores the values of all state variables over the last period of the transient analysis.
4. Transforms the state variables to the frequency domain by using a Fast Fourier Transform (FFT) to establish an initial guess for HB oscillator analysis.
5. Starts the standard HB oscillator analysis.

Additional .HBOSC Analysis Options

Oscillator analysis will make use of all standard HB analysis options as listed in the following table. In addition, the following options are specifically for oscillator applications.

Table 8 HBOSC Analysis Options for Oscillator Applications

Parameter	Description
HBFREQABSTOL	An additional convergence criterion for oscillator analysis. HBFREQABSTOL is the maximum absolute change in frequency between solver iterations for convergence. Default is 1 Hz.
HBFREQRELTOL	An additional convergence criterion for oscillator analysis. HBFREQRELTOL is the maximum relative change in frequency between solver iterations for convergence. Default is 1.e-9.
HBPROBETOL	HBOSC analysis tries to find a probe voltage at which the probe current is less than HBPROBETOL. This option defaults to the value of HBTOL, which defaults to 1.e-9.
HBOSCMAXITER (or HBOSC_MAXITER)	Maximum number of outer-loop iterations for HBOSC analysis. It defaults to 10000.

.HBOSC Output Syntax

The output syntax for .HBOSC analysis is identical to that for HB analysis (see Chapter 5, Steady-State Harmonic Balance Analysis). To output the final frequency of oscillation, use the `HERTZ` keyword. For example, `HERTZ[1]` identifies the fundamental frequency of oscillation.

Note: For `PROBENODE = n1 n2 vp`, where `vp` is a voltage, units must be given in volts.

See also Outputting Phase Noise Source as ASCII Data Files Using *.printpn0.

Troubleshooting Convergence Problems

This section lists the most common causes of convergence problems, how to recognize them, and resolve them.

The HSPICE RF harmonic balance oscillator analysis is a two-tier iterative analysis, consisting of “inner loop” and “outer loop” iterations. In the outer loop iteration, HBOSC will iterate to reduce the “probe error” which is reported for each outer loop iteration. Each outer loop iteration involves a non-autonomous Harmonic Balance (HB) circuit solution; this non-autonomous solve is referred to as the inner loop iteration.

If HBOSC has inner loop convergence problems, the simulation may get stuck on the first outer loop iteration or you may see warning messages such as:

```
Warning: HB_WARN.3: Final HB residual value > HB_TOL.
Rank of HB Jacobian = 155
Warning: HB_WARN.3: HB convergence failure in non-autonomous HB.
```

For each outer loop iteration, the probe voltage and probe frequency are listed. If an outer loop convergence problem occurs, you may see the following:

- Decreasing probe voltage values.
- Wildly fluctuating values of probe frequency.

  ```
  Osc probe : voltage = 0.218234 frequency =
  6.240794122744832e+09
  ```

- A warning message which indicates that the oscillator simulation has reached a non-oscillating DC solution.

  ```
  Warning: HB_ERR.18: HB oscillator analysis has reached the
  NULL solution.
  ```

General Convergence Issues

Probe Node Location

Since convergence is sensitive to the probe node location, convergence problems can often be tracked to this setting.

A common scenario is that the oscillator's output signal is passed through one or more buffers, and the designer may think of the buffer output as the oscillator output. A frequent mistake is to place the probe node at the output of the buffer, but in fact, this will cause HB convergence problems. In this case, the probe node should be moved to the oscillator part of the circuit. Often, it is necessary to select an internal node of a subcircuit to achieve this.

The most typical symptom of this problem is inner loop convergence failure. As mentioned above, for ring oscillators, always choose a node that is part of the ring, i.e., connecting two stages of the ring; for harmonic oscillators, choose a node close to the oscillator.

Incorrect Source Values

If the original netlist was set up to simulate the oscillator in transient analysis, some voltage or current sources may have transient descriptions (e.g., PWL) in order to start the oscillator. For example, a voltage supply may be ramped to simulate a power-up to start the oscillator:

```
Vvdd vdd 0 PWL (0 0 1n 3)
```

In this case, the user would like HBOSC to use 3 as the voltage source value, but HBOSC will use 0 because the explicit DC value of the source is used for Harmonic Balance. HSPICE RF tries to interpret your sources intelligently but, in some cases it may not be able to determine what you intended.

For the above example, there are a few ways to ensure that HSPICE RF correctly interprets the source.

- Remove the explicit DC value. If only a transient description is given, HB will use the time=infinity value of the source.
- Add TRANFORHB=1

  ```
  Vvdd vdd 0 PWL (0 0 1n 3) TRANFORHB=1
  ```

  The TRANFORHB=1 keyword will cause HB to use the transient analysis description in HB and HBOSC.
- Add an explicit HB value

```
Vvdd vdd 0 PWL (0 0 1n 3) HB 3 0 0
```

This causes HB to treat the source as a 3V DC source (0th harmonic is specified). If a HB value is given, HSPICE RF will ignore the PWL description and use “HB 3 0 0” (amplitude=3, phase=0, harmonic=0) instead. The PWL description will still be used for HBTRANINIT.

Incorrect source values usually result in the following:

- High residual value after HBTRANINT. Usually, HBTRANINIT should produce a good starting point for HB or HBOSC. Typical residuals after HBTRANINIT are 1e-4 or 1e-5. If the initial residual printed immediately after HBTRANINIT is done is high, there may be a source problem. In the VDD ramping example above, you might see a residual value of 3.Outer loop may converge to DC solution because incorrect source values result in a non-oscillatory circuit:

  ```
  Warning: HB_ER.18: HB oscillator analysis has reached the NULL
  solution.
  ```

- In some cases, inner loop non-convergence may occur.

GMRES Convergence

When the default value for .option HBSOLVER (=1) is set, HSPICE RF uses a GMRES iterative solver to solve the linear systems that arise on each inner loop Newton-Raphson step. If GMRES does not solve the linear systems accurately enough, then the inner loop may not converge.

The GMRES solver is controlled by two options:

- HBKRYLOVTOL: relative tolerance for GMRES solver. Default is 0.01, or 1%. For some circuits, setting this option will help inner loop convergence:

  ```
  .option HBKRYLOVTOL=1e-3
  ```

- HBKRYLOVDIM: dimension of Krylov subspace used in GMRES iteration. Also controls maximum number of GMRES iterations. In HSPICE RF's *.lis* file, the number of GMRES iterations taken for each Newton-Raphson step is listed. If that number is equal to HBKRYLOVDIM, convergence may be improved by increasing HBKRYLOVDIM. Example:

  ```
  .option HBKRYLOVDIM=80
  ```

The symptom for GMRES convergence difficulty is always inner loop convergence failure, or slow inner loop convergence. If this problem occurs, the inner loop convergence is often good until the residual reaches a fairly low value like 1e-8 or 1e-7, and then stagnates.

Accuracy of Initial Guess

Both inner loop and outer loop convergence improves significantly if the starting point or initial guess of the iterative method is good.

Outer Loop Convergence

For outer loop convergence, the initial guess consists simply of the oscillation frequency and first harmonic amplitude at the probe node location. If inner loop convergence is successful but outer loop convergence is not, then a better frequency or amplitude guess may be needed.

If HBTRANINIT is being used, then the accuracy of the initial guess can be improved by one of the following methods:

- Increase the HBTRANINIT time, simply by increasing the value of the HBTRANINIT option.
- Increase the HBTRANINIT accuracy. You can increase the transient analysis accuracy by setting .option DELMAX or .option SIM_ACCURACY. For example, you may set

  ```
  .option SIM_ACCURACY=10 HBTOL=1e-8
  ```

  Note that SIM_ACCURACY will simultaneously tighten transient and HB accuracy tolerances. If you want HB accuracy to remain unaffected, you may also want to set HBTOL as in the example above.
- Increase accuracy of time domain to frequency domain conversion of HBTRANINIT results, by increasing HBTRANPTS or equivalently, decreasing HBTRANSTEP. For example:

  ```
  .option HBTRANSTEP=1p
  ```

If FSPTS is being used, you can increase the number of points. Sometimes, it is best to supply a guess manually by removing FSPTS and adjusting the TONES value.

If HBTRANINIT is not used, you may be able to improve convergence by manually adjusting the PROBENODE amplitude guess.

To evaluate the effectiveness of your option settings, look at the “probe error” reported after the first outer loop iteration:

```
Iteration 1
```

```
Osc probe : voltage = 0.2 frequency = 5.980000000000000e+09
hb residual = 7.628260e-10
Rank of HB Jacobian = 9102
Probe error = 0.000154462
dv = -0.0411324 df =-2.30464430e+08
```

A smaller probe error value indicates a better initial guess.

Inner Loop Convergence

If inner loop convergence is a problem, it may be because the initial voltage waveform values are not close to the solution. The only way to improve the voltage values is by using HBTRANINIT. While this will not work well for harmonic oscillators, it does work well for ring oscillators. You can improve the accuracy of HBTRANINIT as described in the outer loop convergence section above.

If the initial residual is large after HBTRANINIT, you may want to check to make sure that the voltage and current sources are consistent between HB and transient analysis.

Insufficient Number of Harmonics

If the number of harmonics specified is too small to represent the signals present in the circuit, you may see either convergence problems in either the inner or outer loop, or the solution may converge to an unreasonable frequency value.

It is difficult to know when the number of harmonics is insufficient, but if suspected, it is a simple experiment to increase the value of NHARMS. If convergence was achieved and the number of harmonics is large enough, then the magnitude of the spectral data for all signals should significantly decay with increasing frequency. If the spectral data for node voltages has not decayed at the highest harmonics included in the simulation, an increase in the value of NHARMS is recommended.

Presence of Frequency Divider

If a frequency divider is present and not accounted for by the SUBHARMS setting, convergence is not possible because the Harmonic Balance spectrum does not include the necessary low frequency components. As a result, you will encounter inner loop convergence failure. When debugging HBOSC convergence problems, it is necessary to rule out the possibility of presence of frequency dividers early in the process.

If a frequency divider is present, you can simulate the circuit if you set SUBHARMS to the largest frequency division present in the circuit. If a

frequency divider is present, it is almost always necessary to use the HBTRANINIT option to achieve convergence.

To get optimal performance, it is recommend that you set

```
.option HBSOLVER=2
```

This activates a hybrid time/frequency-domain preconditioner which is particularly effective on frequency dividers.

Oscillator Analysis Using Shooting Newton (.SNOSC)

The analysis described in Chapter 6, Steady-State Shooting Newton Analysis also provides a very effective means for finding the steady-state for oscillator circuits.

Ring oscillators are best suited for time domain analysis using Shooting Newton because they tend to:

- have a low Q
- operate based on digital delays
- have strongly nonlinear behavior
- output signals that are piece-wise-linear or square-wave-like

HBOSC is superior for sinusoidal waveforms. As with the Harmonic Balance approach, the goal is to solve for the additional unknown oscillation frequency. This is accomplished in Shooting Newton by considering the period of the waveform as an additional unknown, and solving the boundary conditions at the waveform endpoints that coincide with steady-state operation. As with regular Shooting Newton analysis, input may be specified in terms of time or frequency values.

Syntax #1

```
.SNOSC TONE=F1 NHARMS=H1 [TRINIT=Ti] OSCNODE=N1
+[MAXTRINITCYCLES=N][SWEEP PARAMETER_SWEEP]
```

Syntax #2

```
.SNOSC TRES=Tr PERIOD=Tp [TRINIT=Tr] OSCNODE=N1
+[MAXTRINITCYCLES=I] SWEEP PARAMETER_SWEEP
```

Parameter	Description
TONE	Approximate value for oscillation frequency (Hz). The search for an exact oscillation frequency begins from this value.
NHARMS	Number of harmonics to be used for oscillator SN analysis.
OSCNODE	Node used to probe for oscillation conditions. This node is automatically analyzed to search for periodic behavior near the TONE or PERIOD value specified.
TRINIT	This the transient initialization time. If not specified, the transient initialization time will be equal to the period (for Syntax 1) or the reciprocal of the tone (for Syntax 2). For oscillators, we recommend specifying a transient initialization time since the default initialization time is usually too short to effectively stabilize the circuit.
MAXTRINITCYCLES	Stops SN stabilization simulation and frequency detection when the simulator detects that MAXTRINITCYCLES have been reached in the `oscnode` signal, or when time=`trinit`, whichever comes first. Minimum cycles is 1. The MAXTRINITCYCLES parameter is optional.
TRES	`TRES` is the time resolution to be computed for the steady-state waveforms (in seconds). The period of the steady-state waveform may be entered either as `PERIOD` or its reciprocal, `TONE`.
PERIOD	`PERIOD` is the expected period T (seconds) of the steady-state waveforms. Enter an approximate value when using for oscillator analysis.

Parameter	Description
SWEEP	Specifies the type of sweep. You can sweep up to three variables. You can specify either LIN, DEC, OCT, POI, SWEEPBLOCK, DATA, OPTIMIZE, or MONTE. SWEEP is an optional parameter. Specify the nsteps, start, and stop frequencies using the following syntax for each type of sweep: ▪ LIN *nsteps start stop* ▪ DEC *nsteps start stop* ▪ OCT *nsteps start stop* ▪ POI *nsteps freq_values* ▪ SWEEPBLOCK *nsteps freq1 freq2 ... freqn* ▪ DATA=*dataname* ▪ OPTIMIZE=OPT*xxx* ▪ MONTE=*val*

Example 1

```
.SNOSC tone=900MEG nharms=9 trinit=10n oscnode=gate
```

Performs an oscillator analysis, searching for periodic behavior after an initial transient analysis of 10 ns. This example uses nine harmonics while searching for an oscillation at the gate node.

Example 2

```
.SNOSC tone=2400MEG nharms=11 trinit=20n oscnode=drainP
```

Performs an oscillator analysis, searching for frequencies in the vicinity of 2.4 GHz. This example uses 11 harmonics and a search at the drainP.

.SNOSC Output Syntax

The output syntax for .SNOSC analysis is identical to that for SN analysis (see Chapter 6, Steady-State Shooting Newton Analysis). To output the final frequency of oscillation, use the `HERTZ` keyword. For example, `HERTZ[1]` identifies the fundamental frequency of oscillation.

See also Using Noise Analysis Results as Input Noise Sources.

Phase Noise Analysis (.PHASENOISE)

The following topics are covered in this section:

- Phase Noise Analysis Overview
- Identifying Phase Noise Spurious Signals
- PHASENOISE Input Syntax
- Phase Noise Algorithms
- PHASENOISE Output Syntax
- Phase Noise Analysis Options
- Measuring Phase Noise with .MEASURE PHASENOISE
- Amplitude Modulation/Phase Modulation Separation

Phase Noise Analysis Overview

.PHASENOISE analysis is designed for use with autonomous oscillators. Phase Noise analysis requires first running either harmonic balance (HBOSC) or Shooting Newton (SNOSC) analysis, and then PHASENOISE analysis. The PHASENOISE analysis itself is identical whether you run SNOSC or HBOSC.

Figure 19 on page 161 shows a simple free-running oscillator, which includes a port with injected current.

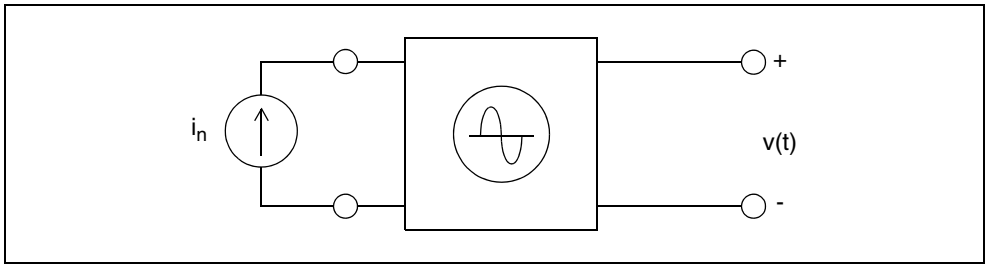


Figure 19 Oscillator with Injected Current

An ideal oscillator would be insensitive to perturbations with a fixed amplitude, frequency, and phase represented by:

Equation 25 $v(t) = A\cos[\omega_0 t + \phi_0]$

A noisy oscillator has amplitude and phase fluctuations we can write as:

Equation 26 $v(t) = A(t)\cos[\omega_0 t + \phi(t)]$

In the preceding equation:

- *A(t)* is the time varying amplitude for the noisy oscillator.
- $\phi(t)$ is the time varying phase for the noisy oscillator.
- ω_0 is the frequency of oscillation.

In most applications, the phase noise is of particular interest, because it represents frequency fluctuations about the fundamental, which you cannot remove. These fluctuations are random processes, and are typically expressed in terms of their power spectral density. For most oscillators, the phase noise is a low-frequency modulation that creates sidebands in the oscillator's spectrum, about ω_0 .

For example, the following equation represents a simple sinusoidal variation in the phase:

Equation 27 $v(t) = A\cos[\omega_0 t + \theta_p \sin\omega_m t]$

- θ_p is the peak phase deviation, specified as $\theta_p = \Delta\omega / \omega_m$
- $\Delta\omega$ is the peak angular frequency deviation.

For $\theta_p \ll 1$, the following equation approximates the output:

Equation 28 $$v(t) = A\left\{\cos(\omega_0 t) - \frac{\theta_p}{2}[\cos(\omega_0 + \omega_p)t - \cos(\omega_0 + \omega_m)t]\right\}$$

That is, when the peak phase deviation is small, the result is frequency components on each side of the fundamental with amplitude $\theta_p / 2$.

The Single-Sideband Phase Noise $L(f_m)$ is the ratio of noise power to carrier power in a 1Hz bandwidth, at offset $\omega_m = 2\pi f_m$, which in this case can be written as:

Equation 29 $$L(f_m) = \left(\frac{V_{sb}}{A}\right)^2 = \frac{\theta_p^2}{4} = \frac{\theta_{rms}^{\ 2}}{2}$$

This model for oscillator noise shows that sidebands about the fundamental, due to noise, are directly related to the spectrum of the phase fluctuations $\theta(t)$. The more general definition of phase noise relates it to the spectral density of phase fluctuations, i.

Equation 30 $$S_\phi(\omega_m) = \frac{\theta_p^2}{2} = 2L(f_m)$$

HSPICE RF uses several sophisticated analysis techniques for computing the power spectrum of the phase variations to yield the phase noise response. This information can be used to predict the spectrum of the oscillator about the fundamental frequency, and also used to predict its random jitter characteristics.

Any .PHASENOISE analysis will result in the calculation of a curve fit for a power-law model according to:

Equation 31 $$L(f) = 10 \cdot \log\left\{\frac{a3}{f^3} + \frac{a2}{f^2} + \frac{a1}{f} + a0\right\}$$

The coefficients $a3$, $a2$, $a1$, and $a0$ are reported in the *.lis* file.

The *.lis* file includes a table which models the phase noise, including the behavioral model fit and its fit error.

```
|-------------------------------------------------------------|
| L(f) = 10*log10( a3/f^(2+ef) + a2/f^2 + a1/f^(ef) + a0 ) dBc/Hz|
| a3 = 0.000000e+00                                           |
| a2 = 3.165111e-02                                           |
| a1 = 0.000000e+00                                           |
| a0 = 0.000000e+00                                           |
| ef = 1.000000e+00                                           |
| Average fit error = 1.6185e+00 dB                           |
| Maximum fit error = 8.4862e+00 dB @ 1.0000e+07 Hz           |
```

Identifying Phase Noise Spurious Signals

Realistic phase noise responses include spurs. Spurs are contributions to the phase noise that result from deterministic signals present within the circuit. In most cases, the spurs are very small signals and do not interfere with the

steady-state operation of the oscillator, but do add energy to the output spectrum of the oscillator. The energy that the spurs adds may need to be included in jitter measurements. The phase noise spurs feature adds an additional analysis option that can predict the spurious contributions to the jitter.

To activate the new phase noise spur analysis, use the SPURIOUS keyword in the .PHASENOISE command. An additional .HBAC analysis is performed that predicts the spurious contributions to the phase noise.

A voltage or current source can be used to add spurious signals to an oscillator circuit. The keyword SPUR identifies the spurious signal.

Syntax

```
Exxxx n1 n2 … [SPUR mag phase freq] … $ voltage spur
Gxxxx n1 n2 … [SPUR mag phase freq] … $ current spur
```

Where,

- *mag* is the amplitude in volts or amps
- *phase* is the phase in degrees
- *freq* is the frequency in Hz

The source is equivalent to a steady-state sinusoidal source at the specified amplitude, phase, and frequency values. It is only used for the spurious analysis and is ignored by all other analyses. The SPUR keyword is can be combined into a source used for other analyses. It is recommended, however, that SPUR sources be added as separate sources.

PHASENOISE Input Syntax

```
.PHASENOISE output frequency_sweep [method=0|1|2]
+ [carrierindex=int] [listfreq=(frequencies|none|all)]
+ [listcount=val] [listfloor=val] [listsources=on|off]
+ [spurious=0|1]
```

Parameter	Description
output	An output node, pair of nodes, or 2-terminal element. HSPICE RF references phase noise calculations to this node (or pair of nodes). Specify a pair of nodes as V(n+,n-). If you specify only one node, V(n+), then HSPICE RF assumes that the second node is ground. You can also specify a 2-terminal element.

Parameter	Description
frequency_sweep	A sweep of type LIN, OCT, DEC, POI, or SWEEPBLOCK. Specify the type, nsteps, and start and stop time for each sweep type, where: ■ type = Frequency sweep type, such as OCT, DEC, or LIN. ■ nsteps = Number of steps per decade or total number of steps. ■ start = Starting frequency. ■ stop = Ending frequency. The four parameters determine the offset frequency sweep about the carrier used for the phase noise analysis. LIN *type nsteps start stop*OCT *type nsteps start stop*DEC *type nsteps start stop*POI *type nsteps start stop*SWEEPBLOCK *freq1 freq2 ... freqn*
method	■ METHOD=0 (default) selects the Nonlinear Perturbation (NLP) algorithm, which is used for low-offset frequencies. ■ METHOD=1 selects the Periodic AC (PAC) algorithm, which is used for high-offset frequencies. ■ METHOD=2 selects the Broadband Phase Noise (BPN) algorithm, which you can use to span low and high offset frequencies. You can use METHOD to specify any single method. See the section on Phasenoise Algorithms below for a more detailed discussion on using the METHOD parameter.
carrierindex	Optional. Specifies the harmonic index of the carrier at which HSPICE RF computes the phase noise. The phase noise output is normalized to this carrier harmonic. Default=1.

Parameter	Description
listfreq	Dumps the element phase noise value to the .lis file. You can specify which frequencies the element phase noise value dumps. The frequencies must match the sweep_frequency values defined in the parameter_sweep, otherwise they are ignored. In the element phase noise output, the elements that contribute the largest phase noise are dumped first. The frequency values can be specified with the NONE or ALL keyword, which either dumps no frequencies or every frequency defined in the parameter_sweep. Frequency values must be enclosed in parentheses. For example: ▪ listfreq=(none) ▪ listfreq=(all) ▪ listfreq=(1.0G) ▪ listfreq=(1.0G, 2.0G) The default value is the first frequency value.
listcount	Dumps the element phase noise value to the .lis file, which is sorted from the largest to smallest value. You do not need to dump every noise element; instead, you can define listcount to dump the number of element phase-noise frequencies. For example, listcount=5 means that only the top 5 noise contributors are dumped. The default value is 20.
listfloor	Dumps the element phase noise value to the .lis file and defines a minimum meaningful noise value (in dBc/Hz units). Only those elements with phase-noise values larger than the listfloor value are dumped. For example, listfloor=-200 means that all noise values below -200 (dBc/Hz) are not dumped. The default value is -300 dBc/Hz.
listsources	Dumps the element phase-noise value to the .lis file. When the element has multiple noise sources, such as a level 54 MOSFET, which contains the thermal, shot, and 1/f noise sources. When dumping the element phase-noise value, you can decide if you need to dump the contribution from each noise source. You can specify either ON or OFF: ON dumps the contribution from each noise source and OFF does not. The default value is OFF.
spurious	Selects phase noise spur analysis 0 = No spurious analysis (default) 1 = Activates additional SPUR source analysis

Phase Noise Algorithms

HSPICE RF provides three algorithms for oscillator phasenoise: nonlinear perturbation, periodic AC, and broadband calculations. These algorithms are selected by setting the METHOD parameter to 0, 1, or 2, respectively.

Each algorithm has their regions of validity and computational efficiency, so some thought is necessary to obtain meaningful results from a PHASENOISE simulation. For each algorithm, the region of validity depends on the particular circuit being simulated. However, there are some general rules that can be applied to oscillator types (that is, ring or harmonic) so that a valid region can be identified. And there are techniques that can be used to check validity of your simulation results.

Nonlinear Perturbation Algorithm

The nonlinear perturbation (NLP) algorithm, which is the default selection, is typically the fastest computation, but is valid only in a region close to the carrier. Generally, you will want to use this algorithm if you interested in phasenoise close to the carrier and do not need to determine a noise floor. NLP computation time is almost independent of the number of frequency points in the phasenoise frequency sweep.

Periodic AC Algorithm

The periodic AC (PAC) algorithm is valid in a region away from the carrier and is slower than the NLP algorithm. The PAC algorithm is used for getting phasenoise in the far carrier region and when you need to determine a noise floor.

The computation time for the PAC algorithm is approximately linearly dependent on the number of frequency points in the phasenoise frequency sweep. If you are using the PAC algorithm, you should try to minimize the number of points in the sweep.

Another issue is that the PAC algorithm becomes more ill-conditioned as you approach the carrier. This means that you may have to generate a steady-state solution with more harmonics to get an accurate simulation as you get closer to the carrier. So, if you find that the PAC is rolling off at close-in frequencies, you should rerun HB analysis with a larger number of harmonics. Although, typically, you will not see improvements in PAC accuracy beyond more than about 100-200 harmonics.

Early in your testing, the best way to verify that NLP and PAC are giving accurate results is to run both algorithms over a broad frequency range and check that the curves have some range in frequency where they overlap. Typically, you will see the NLP curve rolling off at 20 to 30 dB/decade as frequency increases, characteristic of white noise or 1/f noise behavior. Also, the PAC curve will at first be flat or even noisy close to the carrier. At some point though, you will see this curve match the NLP roll-off.

The lowest frequency at which the curves overlap defines the point, f_{PAC} above which the PAC algorithm is valid. Sometimes, by increasing the number of HB harmonics, it is possible to move f_{PAC} to lower frequencies. The highest frequency at which the curves overlap defines the point, f_{NLP} below which the NLP algorithm is valid. A rough rule of thumb is that $f_{PAC} = f_o/Q$, where f_o is the carrier frequency and Q is the oscillator Q-value. This implies that for high-Q oscillators, such as crystal and some harmonic oscillators, that PAC will be accurate to values quite close to the carrier.

Broadband Phase Noise Algorithm

The Broadband Phase Noise (BPN) algorithm allows phase noise simulation over a broad frequency range. The BPN algorithm runs both the NLP and PAC algorithms and then connects them in the overlap region to generate a single phase noise curve. This algorithm is ideal for verifying the NLP and PAC accuracy regions and when you require a phase noise response over a broad frequency range.

PHASENOISE Output Syntax

HSPICE RF supports the output of the phase noise as well as the phase noise due to a specified element. In addition, you can output phase noise due to the specified noise source types. In addition, you can use specialized keywords to output phase noise due to the specified noise source types, as described below.

Specified Element

```
.PRINT PHASENOISE phnoise phnoise(element_name)
.PROBE PHASENOISE phnoise phnoise(element_name)
```

In this syntax, *phnoise* is the phase noise parameter.

The `.PHASENOISE` statement outputs raw data to the **.pn#* and **.printpn#* files. HSPICE RF outputs the *phnoise* data in decibels, relative to the carrier

signal, per hertz, across the output nodes in the .PHASENOISE statement. The data plot is a function of the offset frequency. Units are in dBc/Hz.

- If you use the NLP algorithm (METHOD=0) default, HSPICE RF calculates only the phase noise component.
- If you use the PAC algorithm (METHOD=1), HSPICE RF sums both the phase and amplitude noise components to show the total noise at the output.
- If you use the BPN algorithm (METHOD=2), HSPICE RF adds both the phase and amplitude noise components together to show the total noise at the output. HSPICE RF outputs phnoise to the .pn# file if you set .OPTION POST.

Element phase noise can also be analyzed through the .PRINT and .PROBE statements, which the previous syntax shows. A single *phnoise* keyword specifies the phase noise for the whole circuit, and the phnoise(*element_name*) specifies the phase-noise value of the specified element.

Example 1

```
.HBOSC TONE=900MEG NHARMS=9
+ PROBENODE=gate,gnd,0.65
.PHASENOISE V(gate,gnd) DEC 10 100 1.0e7
+ METHOD=0 CARRIERINDEX=1 $use NLP algorithm
```

This example performs an oscillator analysis, searching for frequencies in the vicinity of 900 MHz, followed by a phase noise analysis at frequency offsets from 100 Hz to 10 MHz.

Example 2

```
.HBOSC TONE=2400MEG NHARMS=11
+ PROBENODE=drainP,drainN,1.0
+ FSPTS=20,2100MEG,2700MEG
+ SWEEP Vtune 0.0 5.0 0.2
.PHASENOISE V(drainP,drainN) DEC 10 100 1.0e7
+ METHOD=0 CARRIERINDEX=1 $use NLP algorithm
```

This example performs a VCO analysis, searching for frequencies in the vicinity of 2.4 GHz. This example uses eleven harmonics, and sweeps the VCO tuning voltage from 0 to 5 V. HSPICE RF uses the nonlinear perturbation (NLP) algorithm to perform a phase noise analysis about the fundamental frequency for each tuning voltage value.

Frequency-Dependent and Frequency-Independent Sources

The phnoise_fdep keyword variable will collect all frequency-dependent noise sources' contributions to the phase noise.

The `phnoise_findep` keyword variable will collect all frequency independent noise sources' contributions.

```
.print phasenoise phnoise_fdep
.print phasenoise phnoise_findep
```

Frequency and Bias Dependencies

- The following syntax is frequency-independent and bias-dependent:

  ```
  .print phasenoise phnoise_cyclo
  ```

  Also acceptable is:

  ```
  .print phasenoise phnoise_cyclostationary
  ```

  Where:

  `cyclo` or `cyclostationary` means anything bias-dependent.

- The following syntax is frequency-independent and bias-independent:

  ```
  .print phasenoise phnoise_stationary
  ```

- The following syntax is bias-independent and frequency-dependent:

  ```
  .print phasenoise phnoise_flicker
  ```

- The following syntax is frequency-dependent and bias-dependent:

  ```
  .print phasenoise phnoise_cycloflicker
  ```

  Also acceptable is:

  ```
  .print phasenoise phnoise_cyclostationaryflicker
  ```

- The `phnoise_fdep` is a combination of `phnoise_flicker` and `phnoise_cycloflicker`.
- The `phnoise_findep` is a combination of `phnoise_stationary` and `phnoise_cyclostationary`.

A Noise *type* suffix can be added to each of these noise terms, `phnoise`, `la`, `ltotal`, `onoise`, to select specific noise-type components:

Table 9 Summary of Noise Type Dependences

Noise type	frequency-dependent	bias-dependent
`phnoise_stationary`	No	No
`phnoise_cyclostationary`	No	Yes

Table 9 Summary of Noise Type Dependences

Noise type	frequency-dependent	bias-dependent
`phnoise_flicker`	Yes	No
`phnoise_cycloflicker`	Yes	Yes
`phnoise_fdep`	is the union of `phnoise_Flicker` and `phnoise_cycloflicker` noise types	
`phnoise_findep`	is the union of `phnoise_stationary` and `phnoise_cyclostationary` noise types	

Where: Noise_term can be `phnoise`, `la`, `ltotal`, `onoise` (see Amplitude Modulation/Phase Modulation Separation later in this chapter).

See also Using Noise Analysis Results as Input Noise Sources.

Phase Noise Analysis Options

Table 10 lists the control options specific to PHASENOISE applications.

Table 10 PHASENOISE Analysis Options

Parameter	Description
BPNMATCHTOL=*val*	Determines the minimum required match between the NLP and PAC phase noise algorithms. An acceptable range is 0.05dB to 5dB. The default is 0.5dB.
PHASENOISEKRYLOVDIM (or PHASENOISE_KRYLOV_DIM	Specifies the dimension of the Krylov subspace that the Krylov solver uses. This must be an integer greater than zero. The default is 500.
PHASENOISEKRYLOVITER (or PHASENOISE_KRYLOV_ITER)	Specifies the maximum number of Krylov iterations that the phase noise Krylov solver takes. Analysis stops when the number of iterations reaches this value. The default is 1000.
PHASENOISETOL	Specifies the error tolerance for the phase noise solver. This must be a real number greater than zero. The default is 1e-8.

Table 10 PHASENOISE Analysis Options (Continued)

Parameter	Description
PHNOISE_LORENTZ=*val*	Turns on a Lorentzian model for the phase noise analysis. ▪ *val*=0: uses a linear approximation to a Lorentzian model and avoids phasenoise values >0dB for low offsets ▪ *val*=1 (default): applies a Lorentzian model to all noise sources ▪ *val*=2: applies a Lorentzian model to all non-frequency dependent noise sources ▪ *val*=3: Lorentzian model applied to white noise source, Gaussian model applied to flicker noise sources.
PHNOISEAMPM=1	Turns on amplitude modulation/phase modulation separation. See Amplitude Modulation/Phase Modulation Separation for details.

Measuring Phase Noise with .MEASURE PHASENOISE

The HSPICE RF optimization flow can read the measured data from a `.MEASURE PHASENOISE` analysis. This flow can be combined in the HSPICE RF optimization routine with a `.MEASURE HBTR` analysis (see Using .MEASURE with .HB Analyses on page 124) and a `.MEASURE HBNOISE` analysis (see Measuring HBNOISE Analyses with .MEASURE on page 206). The `.MEASURE PHASENOISE` syntax supports the following measurements:

- FIND

  ```
  .MEASURE PHASENOISE result FIND phnoise at = IFB_value
  ```

  — yields the result of a variable value at a specific input frequency band (IFB) point. For example:

  ```
  .MEASURE PHASENOISE np1 find PHNOISE at=100K
  ```

- WHEN

  ```
  .MEASURE PHASENOISE result WHEN phnoise=value
  ```

  —yields the input frequency point at a specific phnoise value. For example:

```
.MEASURE PHASENOISE fcorn1 when PHNOISE=-120
```

- RMS, average, min, max, and peak-to-peak

  ```
  .MEASURE PHASENOISE result func phnoise
  + [FROM = IFB1] [TO = IFB2]
  ```

  —yields the average, RMS, minimum, maximum, or peak-to-peak value of the phase noise from frequency `IFB1` to frequency `IFB2`, where the value of `func` can be RMS, AVG, MIN, MAX or PP. If `FROM` and `TO` are not specified, the value will be calculated over the frequency range specified in the `.PHASENOISE` command. For example:

  ```
  .measure PHASENOISE agn1 AVG phnoise from=100k to=10meg
  ```

- Integral evaluation

  ```
  .MEASURE PHASENOISE result INTEGRAL phnoise
  + [FROM = IFB1] [TO = IFB2]
  ```

  —integrates the phase noise value from the `IFB1` frequency to the `IFB2` frequency. For example:

  ```
  .MEASURE PHASENOISE rns1 INTEGRAL phnoise from=50k to 500k
  ```

- Derivative evaluation

  ```
  .MEASURE PHASENOISE result DERIVATIVE phnoise AT = IFB1
  ```

  —finds the derivative of phase noise at the `IFB1` frequency point. For example:

  ```
  .MEASURE PHASENOISE fdn1 DERIVATIVE phnoise at=10meg
  ```

Note: `.MEASURE PHASENOISE` cannot contain an expression that uses a phasenoise variable as an argument. You also cannot use `.MEASURE PHASENOISE` for error measurement and expression evaluation of the `.PHASENOISE` command.

See also, the .MEASURE PHASENOISE command in the *HSPICE Reference Manual: Commands and Control Options*.

Amplitude Modulation/Phase Modulation Separation

Amplitude Modulation (AM) and Phase Modulation (PM) components of the total noise can be separated by calculating components in-phase (AM

component) and in quadrature (PM component) with the carrier using PAC and BPN PHASENOISE analysis. Output and measure syntax is used to separate AM/PM noise.

This feature is turned on by setting the `.OPTION PHNOISEAMPM=1` (see .OPTION PHNOISEAMPM in the *HSPICE Reference Manual: Commands and Control Options*. See also, Important Note for AM/PM Users at the end of this section.

Keywords for AM/PM separations are:

- Phase Modulation term only: `phnoise`
- Amplitude Modulation term only: `la`
- Total phase noise: `ltotal`
- Voltage noise: `onoise`

```
.PROBE PHASENOISE phnoise [la] [ltotal] [onoise]
.PRINT PHASENOISE phnoise [la] [ltotal] [onoise]
```

A Noise type suffix can be added to each of these noise terms, `phnoise`, `la`, `ltotal`, `onoise`, to select specific noise-type components:

Table 11 Summary of Noise_term

Noise type	frequency-dependent	bias-dependent
*Noise_term*_phnoise_stationary	No	No
*Noise_term*_phnoise_cyclostationary	No	Yes
*Noise_term*_phnoise_flicker	Yes	No
*Noise_term*_phnoise_cycloflicker	Yes	Yes
*Noise_term*_phnoise_fdep	The union of phnoise_Flicker and phnoise_cycloflicker noise types	
*Noise_term*_phnoise_findep	The union of phnoise_stationary and phnoise_cyclostationary noise types	

Where: Noise_term can be `phnoise`, `la`, `ltotal`, `onoise`.

Example

```
.PROBE PHASENOISE phnoise_cyclostationary
```

This example outputs the phase modulation noise associated only with Cyclostationary sources (i.e., sources that are bias dependent, but not frequency dependent).

Noise Element output is of the form Noise_term (element_name), where Noise_term can be `phnoise`, `la`, `ltotal`, `onoise`, and element_name is a valid netlist element name.

Output File Format

- File **.printpn#*: Writes output from the `.PRINT` statement when using HB to obtain the steady state solution.
- File **.pn#*: Writes output from the `.PROBE` statement when using HB to obtain the steady state solution.
- File **.printsnpn#*: Writes output from the `.PRINT` statement when using SN to obtain the steady state solution.
- File **.snpn#*: Writes output from the `.PROBE` statement when using SN to obtain the steady state solution.

Noise source contributions are listed sequentially and are controlled by the `.PHASENOISE` command line parameters Listfreq, ListCount, Listfloor, Listsources. A noise list block will be generated for each output parameter specified in the `.PRINT`/`.PROBE` statement e.g., `phnoise`, `la`, `ltotal`, `onoise`.

.MEASURE Syntax and File Format

`.MEASURE PHASENOISE` extends output variables to the set: `am[noise]` `pm[noise]`

Measure File Format

- File **.mpn#:* Writes output from the .MEASURE statement when using HB to obtain the steady state solution.
- File **.msnpn#:* Writes output from the .MEASURE statement when using SN to obtain the steady state solution.

Interpreting Phase Noise Analysis Results

A typical phase noise plot consists of a line, which drops off as a function of frequency, at a slope of -20dbc/decade where white noise dominates, or -30dbc/decade where flicker noise dominates. At very low offset frequencies, the phase noise rolls off at according to a Lorentzian shape, such that it never exceeds 0 dbc/Hz even for very low offset frequencies. The 0 dbc/Hz value represents the power of the carrier oscillation, at 0 offset frequency. At very

high offset frequencies, the slope can deviate from -20 dbc/decade due to the existence of a noise floor or a circuit feedback effect.

Numerical methods for phase noise analysis have limitations. The main limitation in the PAC phase noise algorithm is that it rolls off too quickly at low offset frequencies. In the low frequency region, NLP should be trusted. The main limitation of the NLP algorithm is that it does not cover all high frequency effects, so PAC should be trusted in the high frequency region.

The BPN algorithm attempts to combine the NLP and PAC results to generate a single result that is valid for all offset frequencies. It may fail if it cannot identify an overlap region where NLP and PAC results match. If no overlap region can be found, the user should attempt to increase nharms on the .HBOSC command, as this will increase the accuracy of both algorithms, especially PAC. PAC accuracy is more sensitive to nharms than NLP.

If the phase noise results are suspected to be inaccurate, check the following:

1. Is the .HBOSC steady state solution fully converged?

   Explanation: The NLP or PAC small-signal noise analysis requires a highly accurate steady state solution.

2. Is the phase noise analysis fully converged?

   Explanation: Phase noise analysis uses a GMRES iterative linear solver. If this iterative solver reaches its iteration limit before it is fully converged, the results are not reliable. Check the number of Krylov iterations that the phase noise analysis required. If it took the maximum number of iterations (as set by PHASENOISE_KRYLOV_ITR, default=1000), then the results may not be fully converged and should not be trusted.
   The options PHASENOISE_KRYLOV_DIM, PHASENOISE_KRYLOV_TOL, and PHASENOISE_KRYLOV_ITR can be used to control the GMRES solver. You can increase PHASENOISE_KRYLOV_DIM to improve convergence rate at the expense of memory, or increase PHASENOISE_KRYLOV_ITR to allow more iterations.

Important Note for AM/PM Users

There are discrepancies that may occur between this feature and the traditional PAC phase noise analysis in HSPICE RF. Total phase noise (i.e., `ltotal`) is the sum of two terms, the amplitude modulation (am) and phase modulation (phnoise). Traditionally, PAC phase noise reports the `phnoise` (phase modulation component) and `ltotal` (total phase noise) terms as identical, with the assumption that the am term (amplitude modulation component) was zero.

The PAC phase noise am/pm feature described in this section separately calculates the am and phnoise components. All phase noise measurements involving either PAC or BPN will be affected. In most cases the differences between `PHNOISEAMPM=1` and `=0` will be small unless you expect a significant AM component. You may see a slight decrease in the new phase noise (phase modulation) component compared to the old calculation.

For example, the random jitter calculations HSPICE RF uses are accurate only when they involve the PM component of phase noise. A small error may be introduced when they are based on `ltotal`, or AM noise is included.

NLP phase noise (`method=0`) only calculates the phnoise component and is not affected by the am/pm option.

BPN phase noise (`method=2`) is affected in that the far side component is derived from the PAC phase noise.

Workaround: Make sure that `.OPTION PHNOISEAMPM=0` (the default) to assure that Periodic AC phase noise amplitude modulation (AM) component is set to zero to maintain backward compatibility and traditional results for phnoise and jitter, for example.

Accumulated Jitter Measurement for Closed Loop PLL Analysis

Enhancements to HSPICE RF include considerable support for a variety of jitter measurements. Many of these are important in a PLL flow, where the HBOSC or SNOSC analyses are used to compute a phase noise response for an oscillator or VCO, and the resulting random jitter is derived from phase noise. In the PLL methodology, other HSPICE RF analyses are used to compute phase noise contributions from the other PLL building block. A closed loop analysis is then performed, using phase-domain models for both signal and noise responses, that takes into account the noise contributions from all such blocks. To complete this flow is the ability to compute “Accumulated Jitter” or “Timing Jitter” for the closed loop PLL. The accumulated jitter response is essentially an integral transformation of the closed-loop PLL response. The following sections show how accumulated jitter also be measured directly from the phase noise output of an open loop oscillator analysis.

In the PLL flow, the closed loop phase noise must be interpreted from the results of a linear HSPICE RF .AC/.NOISE analysis. The following sections describe a capability that allows direct measurement of accumulated jitter from

the results of this closed loop noise analysis, without any special interpretation of the results.

Jitter Measurements from Phase Noise

These topics are discussed in the following sections:

- Jitter Definitions
- Jitter Output Syntax
- .MEASURE Statements for Jitter
- Peak-to-Peak Jitter

Jitter Definitions

HSPICE RF provides several random jitter (RJ) measurements. This section defines, describes, and compares the various jitter measurements provided. Random jitter measurements are derived from the results of an HSPICE RF phase noise analysis. The relationships between phase noise and the random jitter measurements are presented here, and their means for calculation. The types of random jitter measurements include: Timing, Phase, Period, Tracking, Long-Term, and Cycle-to-Cycle Jitter.

Timing jitter is a measurement of oscillator uncertainty in the time domain. For clock applications, time domain measurements are preferable, since most specifications of concern involve time domain values.

Timing jitter is the standard deviation of the timing uncertainty, which is a function of the auto-correlation function in the power spectrum of the phase variations. Timing Jitter is the square root of the variance (standard deviation squared) of the timing uncertainty between two clock edges separated by an interval given by $\tau = N \cdot T_o$, where T_o is the ideal clock period. It can be written as a function of the auto-correlation function of the power spectrum of phase variations as:

Equation 32
$$\sigma_{TIE}^2(\tau) = \frac{2}{\omega_o^2}[R_\phi(0) - R_\phi(\tau)]$$

where TIE refers to the Time Interval Error. This measurement is known as Timing Jitter, Accumulated Jitter, or N-Cycle Jitter, since it represents the jitter that may accumulate over an interval of many periods.

The Weiner-Khintchine Theorem [1] relates the auto correlation function to the power spectrum of phase variations as in the following equation:

Equation 33
$$R_\phi(\tau) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S_\phi(\omega) e^{j\omega\tau} d\omega = 2\int_0^{\infty} L(f)\cos(2\pi f\tau)df$$

where $S_\phi(\omega)$ is the double-sided power spectrum of phase variations, and $L(f)$ is the single-sideband phase noise. The auto-correlation for $\tau = 0$ is given by

Equation 34
$$R_\phi(0) \equiv \phi_{rms}^2 = 2\int_0^{\infty} L(f)df$$

which defines ϕ_{rms} in HSPICE RF known as RMS Phase Jitter.

Using the identity $2\sin^2\alpha = 1 - \cos 2\alpha$ we can then write:

Equation 35
$$\sigma_{TIE}^2(\tau) = \frac{8}{\omega_o^2}\int_0^{\infty} L(f)\sin^2(\pi f\tau)df$$

to enable currently supported HSPICE RF jitter measurements to be written as:

Equation 36
$$\phi_{rms} = \sigma_{ph}\cdot\omega_0 = \sqrt{2\int_0^{\infty} L(f)df}$$ "RMS Phase Jitter"

$$\sigma_{TIE}(\tau) = \frac{2}{\omega_0}\sqrt{2\int_0^{\infty} L(f)sin^2(\pi f\tau)df}$$ "Timing (Time Interval Error) Jitter"

From these definitions, several other key jitter measurements can be derived, including Period Jitter, Tracking Jitter, Long-Term Jitter, and Cycle-to-Cycle Jitter.

Period Jitter is equivalent to the value for Timing Jitter for a one period interval. We therefore have:

Equation 37 $$\sigma_{PER} = \sigma_{TIE}(T_0) = \frac{2}{\omega_0}\sqrt{2\int_0^{\infty} L(f)sin^2(\pi f T_0)df}$$ "Period Jitter"

Tracking Jitter is equivalent to the value (in units of seconds) for RMS Phase Jitter, or:

Equation 38 $$\sigma_{tr} = \sigma_{ph} = \frac{\phi_{rms}}{\omega_0} = \frac{1}{\omega_0}\sqrt{2\int_0^{\infty} L(f)df}$$ "Tracking Jitter"

Long-Term Jitter is equivalent to $\sqrt{2}$ times the Tracking Jitter, i.e.:

Equation 39 $$\sigma_{\Delta T \to \infty} = \sigma_{TIE}(\tau \to \infty) = \sqrt{2}\frac{\phi_{rms}}{\omega_0} = \frac{2}{\omega_0}\sqrt{\int_0^{\infty} L(f)\,df}$$ "Long-Term Jitter"

Cycle-to-Cycle Jitter is based on the difference between adjacent Period Jitter measurements. It is given by:

Equation 40 $$\sigma_{CTC} = \sqrt{4\sigma_{PER}^2 - \sigma_{TIE}^2(2T_0)}$$ "Cycle-to-Cycle Jitter"

In general, each of the above calculations must be performed carefully over limits of integration to accurately calculate jitter expressions based on the finite frequency limits provided for the phase noise analysis. Linear interpolation is used, but the phase noise generally follows more of a power law expansion.

Jitter Output Syntax

The timing jitter calculations are derived from the results of phase noise analysis. The phase noise output syntax supports the `JITTER` keyword as an output keyword in addition to the `PHNOISE` keyword.

```
.PRINT PHASENOISE PHNOISE JITTER
.PROBE PHASENOISE PHNOISE JITTER
```

If the `JITTER` keyword is present, the `.PHASENOISE` statement also outputs the raw jitter data to *.jt0 and *.printjt0 data files. The PHNOISE data is given in units of dBc/Hz, i.e., dB relative to the carrier, per Hz, across the output nodes specified by the PHASENOISE statement. The data plot is a function of offset frequency. If the JITTER keyword is present, .PHASENOISE outputs the TIE Timing Jitter (Accumulated Jitter) data to *.jt0 and *.printjt0 data files. These

data are plotted as a function of time in units of seconds. The jitter calculations make use of some of the parameters given in the .PHASENOISE syntax (see PHASENOISE Input Syntax for the syntax and examples.).

The time samples for timing jitter output make use of the same number of points as the phase noise frequency sweep specification.

The output of timing jitter information uses a corresponding time sampling derived via:

Equation 41

$$\tau_1 = \frac{1}{T_0},\ \tau_2 = \frac{2}{T_0},\ ...,\tau_N = \frac{N}{T_0}$$

.MEASURE Statements for Jitter

The jitter-specific `.MEASURE` statements specify the jitter keywords as follows. (For discussion of the BER parameter, see below.)

```
.MEASURE PHASENOISE Jname PERJITTER phnoise
+ [UNITS=(sec|rad|UI)] [BER=val]
.MEASURE PHASENOISE Jname CTCJITTER phnoise
+ [UNITS=(sec|rad|UI)] [BER=val]
.MEASURE PHASENOISE Jname RMSJITTER phnoise
+ [FROM start_frequency] [TO end_frequency]
+ [UNITS=(sec|rad|UI)] [BER=val]
.MEASURE PHASENOISE Jname PHJITTER phnoise
+ [FROM start_frequency] [TO end_frequency]
+ [UNITS=(sec|rad|UI)] [BER=val]
.MEASURE PHASENOISE Jname TRJITTER phnoise
+ [FROM start_frequency] [TO end_frequency]
+ [UNITS=(sec|rad|UI)] [BER=val]
.MEASURE PHASENOISE Jname LTJITTER phnoise
+ [FROM start_frequency] [TO end_frequency]
+ [UNITS=(sec|rad|UI)] [BER=val]
```

RMSJITTER, PHJITTER, and TRJITTER are synonymous measurements, all based on the calculations described related to the RMS Phase Jitter value in units of seconds given by $\sigma_{ph} = \phi_{rms} / \omega_0$. These measurements allow control of the integration range using the FROM and TO parameters. The measurements for PERJITTER, and CTCJITTER use the full offset frequency sweep range given for the phase noise analysis to compute values (the FROM and TO parameters are ignored if entered).

As given currently in HSPICE RF, the frequency intervals can be modified for these jitter calculations (if desired, although not recommended), and UNITS can be selected between seconds, radians, and Unit Intervals. The following table specifies the calculation used for units=seconds for each jitter measurement.

MEASURE name	Calculation used (Units=sec)
RMSJITTER	$\sigma_{ph} = \phi_{rms} / \omega_0$
PHJITTER	$\sigma_{ph} = \phi_{rms} / \omega_0$
TRJITTER	$\sigma_{ph} = \phi_{rms} / \omega_0$
PERJITTER	σ_{PER}
LTJITTER	$\sigma_{\Delta T \to \infty} = \sqrt{2}\phi_{rms} / \omega_o$
CTCJITTER	σ_{CTC}

Example

```
.meas phasenoise rj RMSJITTER phnoise from 1K to 100K
+ units = rad
```

Peak-to-Peak Jitter

As noted in .MEASURE Statements for Jitter, an additional BER (Bit Error Rate) parameter is supported. This parameter allows you to convert any jitter value from an RMS value into a Peak-to-Peak value. The RMS jitter values correspond to a 1-sigma standard deviation value for the Gaussian distribution of the jitter. Peak-to-peak values represent the full span of the Gaussian distribution. Since this span is theoretically unbounded for truly random distributions, the conversion to peak-to-peak values has to be interpreted as spanning some number of sigma values. You can arrive at this number (i.e., “sigma multiplier”) by specifying a corresponding Bit Error Rate.

The term “BER” corresponds to the unitless Bit Error Rate that allows for this conversion. The following table shows some example conversions from various

BER values into a “sigma multiplier” value which corresponds to the number of sigma standard deviations in converting from RMS to peak-to-peak values:

Bit Error Rate	Sigma Multiplier
10^{-3}	6.180
10^{-4}	7.438
10^{-5}	8.530
10^{-6}	9.507
10^{-7}	10.399
10^{-8}	11.224
10^{-9}	11.996
10^{-10}	12.723
10^{-11}	13.412
10^{-12}	14.069
10^{-13}	14.698
10^{-14}	15.301
10^{-15}	15.883
10^{-16}	16.444

These conversions are done in accordance with the relationship:

Equation 42
$$\frac{1}{2} erfc \frac{\alpha}{2 \cdot \sqrt{2}} = \mathrm{BER}$$

where, $erfc$ is the complementary error function, and α is the Sigma Multiplier. Support for peak-to-peak conversions is included for a continuous range of

BER values from $10^{-16} \leq \alpha \leq 10^{-3}$ (and some values extrapolated outside this range).

Specification of the BER parameter results in the output of the Peak to Peak jitter value, and not the RMS value. Labels for the measurements show appropriate “rms” and “p-p” labels. A BER parameter set to BER=0 is equivalent to having no parameter, and only results in the RMS calculation.

Errors/Warnings

Error handling and recovery is exercised to capture obvious errors in input specifications. The following error checking is performed:

- Calculations are be performed if oscillator or phase noise analysis fails.
- ERROR if L(f) > 1 over any part of the frequency sweep (non-dB form).
- ERROR if L(f) < 0 over any part of the frequency sweep (non-dB form).
- Error if any time or frequency samples are negative values.
- ERROR if BER < 0 for any Jitter measurement.
- WARNING if BER > 1 for any Jitter measurement.
- WARNING if f0 < 10 Hz. Message: “Jitter calculations may be ineffective for offset frequencies under 10 Hz.”

Small-Signal Phase-Domain Noise Analysis (.ACPHASENOISE)

To see the influence that oscillator or VCO phase noise can have on a system where it is present, it is necessary to perform a phase-domain analysis where the circuit variables are phase, and the input noise stimuli are phase noise. This is the purpose of the .ACPHASENOISE analysis in HSPICE RF.

This type of analysis is critical, for example, in analyzing the effects of noise in a phase-locked loop (PLL). In a PLL design flow, the HBOSC or SNOSC analyses are used to compute a phase noise response for an oscillator or VCO. HSPICE RF analyses can be used to compute phase noise contributions from the other PLL building blocks. A closed loop PLL analysis can then be performed by using phase-domain models for both signal and noise responses, where the noise contributions from all blocks are input as phase noise stimuli. Such an analysis can be performed to determine the PLL closed-loop phase

noise, based on the contributions of each block, determined by an open loop analysis.

AC Phase Noise Analysis Syntax

A small-signal phase-domain analysis, such as that useful for PLL noise analyses, can use a methodology based on small-signal .AC and .NOISE analyses, where the circuit analyzed is a phase-domain modeled circuit (not voltage/current domain), and the results are therefore interpreted accordingly as phase variables. The variables of the associated noise analysis must therefore interpreted as phase noise variables. A normal .AC analysis command can be used to activate the small-signal analysis and specify its LIN/OCT/DEC/POI or SWEEPBLOCK frequency sweep. However, to properly interpret signal and noise quantities as phase variables, the following command properly formats output in terms of phase noise variables. The syntax is:

```
.ACPHASENOISE output input [interval] carrier=freq
+ [listfreq=(frequencies|none|all)]
+ [listcount=val] [listfloor=val]
+ [listsources=(1|0)]
```

where:

- *output* Node voltage or branch current output variable
- *input* Independent source used as input reference
- *interval* Number of intervals for which to dump jitter and noise summary information
- *freq* Frequency (in Hz) of the fundamental carrier upon which the noise transformations are based
- list… List parameters usage is consistent with other HSPICE noise analyses

ACPHASENOISE Analysis .PRINT/.PROBE Syntax

The unique aspect of the .ACPHASENOISE analysis is that it allows the small signal noise calculation results to be interpreted as phase noise values. The available .print/.probe measurements reflect this. The .print/.probe output syntax are the "JITTER" and "PHNOISE" keywords consistent with the HSPICE RF .phasenoise analysis, namely:

```
.PRINT ACPHASENOISE PHNOISE JITTER
.PROBE ACPHASENOISE PHNOISE JITTER
```

As with the .PHASENOISE analysis, the .ACPHASENOISE analysis outputs raw data to **.pn0* and **.printpn0* files. The PHNOISE data is given in units of dBc/Hz, i.e., dB relative to the carrier, per Hz, across the output nodes specified by the .ACPHASENOISE statement. The data plot is a function of offset frequency. If the "JITTER" keyword is present, .ACPHASENOISE also outputs the accumulated TIE jitter data to **.jt0* and **.printjt0* data files. These data are plotted as a function of time in units of seconds. The Timing Jitter data itself has units of seconds. The timing jitter calculations make use of the parameters given in the .ACPHASENOISE syntax, such as "freq" and "interval".

Se also Using Noise Analysis Results as Input Noise Sources.

.MEASURE Support for ACPHASENOISE

Single valued jitter measurements are available from .MEASURE statements. Examples include period jitter, cycle-to-cycle jitter, and phase jitter measurements, respectively, as shown below:

```
.MEASURE ACPHASENOISE Jname PERJITTER phnoise
+ [UNITS=(sec|rad|UI)] [BER=val]

.MEASURE ACPHASENOISE Jname CTCJITTER phnoise
+ [UNITS=(sec|rad|UI)] [BER=val

.MEASURE ACPHASENOISE Jname PHJITTER phnoise
+ [FROM start_frequency [TO end_frequency]
+ [UNITS=(sec|rad|UI)] [BER=val]
```

Errors and Warnings

Error checking of values are consistent with that used for the .PHASENOISE analysis.

References

[1] E. Ngoya, A. Suarez, R. Sommet, R. Quere, “Steady State Analysis of Free or Forced Oscillators by Harmonic Balance and Stability Investigation of Periodic and Quasi-Periodic Regimes,” *International Journal of Microwave and Millimeter-Wave Computer-Aided Engineering*, Volume 5, Number 3, pages 210-223 (1995)

[2] C.R. Chang, M.B. Steer, S. Martin, E. Reese, “Computer-Aided Analysis of Free-Running Microwave Oscillators,” *IEEE Trans. on Microwave Theory and Techniques*, Volume 39, No. 10, pages 1735-1745, October 1991.

[3] G.D. Vendelin, *Design of Amplifiers and Oscillators by the S-Parameter Method*, John Wiley & Sons, 1982

[4] A. Demir, A. Mehrotra, J. Roychowdhury, “Phase Noise in Oscillators: A Unifying Theory and Numerical Methods for Characterization” in Proc. IEEE DAC, pages 26-31, June 1998.

[5] A. Demir, A. Mehrotra, and J. Roychowdhury, “Phase Noise in Oscillators: A Unifying Theory and Numerical Methods for Characterization,” *IEEE Trans. Circuits System I*, Volume 47, pages 655–674, May 2000.

[6] A. van der Ziel, Noise in Solid State Devices and Circuits, John Wiley & Sons, 1986.

[7] A. Hajimiri, S. Limotyrakis, and T.H. Lee, “Jitter and phase noise in ring oscillators,” IEEE J. Solid-State Circuits, vol. 34, no. 6, pp. 790-804, June 1999.

[8] Jitter Analysis Techniques for High Data Rates, Application Note 1432, Agilent Technologies, Feb. 2003.

[9] Characterization of Clocks and Oscillators, NIST Technical Note 1337, National Institute of Standards and Technology.

[10] G.V. Klimovitch, “Near-carrier oscillator spectrum due to flicker and white noise,” Proc. ISCAS 2000 (Geneva), May 2000.

8

Large Signal Periodic AC, Transfer Function, and Noise Analyses

Describes how to use both harmonic balance-based and Shooting Newton-based AC, and transfer function analyses, as well as nonlinear, steady-state noise analysis.

The following topics are presented in this section:

- Multitone Harmonic Balance AC Analysis (.HBAC)
- Shooting Newton AC Analysis (.SNAC)
- Multitone Harmonic Balance Noise (.HBNOISE)
- Shooting Newton Noise Analysis (.SNNOISE)
- Periodic Time-Dependent Noise Analysis (.PTDNOISE)
- Multitone Harmonic Balance Transfer Function Analysis (.HBXF)
- Shooting Newton Transfer Function Analysis (.SNXF)

Multitone Harmonic Balance AC Analysis (.HBAC)

You use the `.HBAC` (Harmonic Balance AC) statement for analyzing linear behavior in large-signal periodic systems. The `.HBAC` statement uses a periodic AC (PAC) algorithm to perform linear analysis of autonomous (oscillator) or nonautonomous (driven) circuits, where the linear coefficients are modulated by a periodic, steady-state signal.

Multitone HBAC analysis extends single-tone HBAC to quasi-periodic systems with more than one periodic, steady-state tone. One application of multitone HBAC is to more efficiently determine mixer conversion gain under the

influence of a strong interfering signal than is possible by running a swept three-tone HB simulation.

The following sections discuss these topics:

- Prerequisites and Limitations
- Input Syntax
- Output Syntax
- HBAC Output Data Files
- Errors and Warnings

Prerequisites and Limitations

The following prerequisites and limitations apply to HBAC:

- Requires one and only one `.HBAC` statement. If you use multiple `.HBAC` statements, HSPICE RF uses only the last `.HBAC` statement.
- Requires one and only one `.HB` statement.
- Supports arbitrary number of tones.
- Requires placing the parameter sweep in the `.HB` statement.
- Requires at least one HB source.
- Requires at least one HBAC source.
- Supports unlimited number of HB and HBAC sources.
- The requested maximum harmonic in a `.PROBE` or `.PRINT` statement must be less than or equal to half the number of harmonics specified in harmonic balance (that is, max_harm <= num_hb_harms / 2).

Input Syntax

```
.HBAC frequency_sweep
```

Parameter	Description
frequency_sweep	Frequency sweep range for the input signal (also referred to as the input frequency band (IFB) or fin). You can specify LIN, DEC, OCT, POI, or SWEEPBLOCK. Specify the nsteps, start, and stop frequencies using the following syntax for each type of sweep: ▪ LIN *nsteps start stop* ▪ DEC *nsteps start stop* ▪ OCT *nsteps start stop* ▪ POI *nsteps freq_values* ▪ SWEEPBLOCK *nsteps freq1 freq2 ... freqn* ▪ DATA=*dataname*

HBAC Analysis Options

The following options directly relate to a HBAC analysis and override the corresponding PAC options if specified in the netlist:

- `.OPTION HBACTOL`, default = 1x10-8, Range = 1x10-14 to Infinity
- `.OPTION HBACKRYLOVDIM`, default = 300, Range = 1 to Infinity
- `.OPTION HBACKRYLOVITER | HBAC_KRYLOV_ITER`, default = 1000, Range = 1 to Infinity

If these parameters are not specified, then the following conditions apply:

- If `HBACTOL` > `HBTOL`, then `HBACTOL` = `HBTOL`
- If `HBACKRYLOVDIM` < `HBKRYLOVDIM`, then `HBACKRYLOVDIM` = `HBKRYLOVDIM`

Output Syntax

This section describes the syntax for the HBAC `.PRINT` and `.PROBE` statements. These statements are similar to those used for HB analysis.

.PRINT and .PROBE Statements

```
.PRINT HB TYPE(NODES | ELEM)[INDICES]
```

```
.PROBE HB TYPE(NODES | ELEM)[INDICES]
```

Parameter	Description
TYPE	Specifies a harmonic type node or element. TYPE can be one of the following: ▪ Voltage type – V = voltage magnitude and phase in degrees VR = real component VI = imaginary component VM = magnitude VP - Phase in degrees VPD - Phase in degrees VPR - Phase in radians VDB - dB units VDBM - dB relative to 1 mV ▪ Current type – I = current magnitude and phase in degrees IR = real component II = imaginary component IM = magnitude IP - Phase in degrees IPD - Phase in degrees IPR - Phase in radians IDB - dB units IDBM - dB relative to 1 mV ▪ Power type – P ▪ Frequency type – hertz[index], hertz[index1, index2, ...] You must specify the harmonic index for the hertz variable. The frequency of the specified harmonics is dumped.
NODES | ELEM	NODES or ELEM can be one of the following: ▪ Voltage type – a single node name (n1), or a pair of node names, (n1,n2) ▪ Current type – an element name (elemname) ▪ Power type – a resistor (resistorname) or port (portname) element name ▪ Frequency type – the harmonic index for the hertz variable. The frequency of the specified harmonics is dumped.

Parameter	Description
INDICES	Index to tones in the form [n1, n2, ..., nK, +/-1]. ■ nj is the index of the j-th HB tone and the .HB statement contains K tones ■ +/-1 is the index of the HBAC tone Wildcards are not supported if this parameter is used. You can transform HB data into the time domain and output by using the following syntax:.PRINT HBTRAN ov1 [ov2 ... ovN].PROBE HBTRAN ov1 [ov2 ... ovN]. See TYPE above for voltage and current type definitions.

HBAC Output Data Files

An HBAC analysis produces these output data files:

- Output from the `.PRINT` statement is written to a .printhb# file. This data is against the IFB points.
  - The header contains the large-signal fundamental and the range of small-signal frequencies.
  - The columns of data are labeled as F(Hz), followed by the output variable names. Each variable name has the associated mixing pair value appended.

    All *N* variable names and all *M* mixing pair values are printed for each swept small-signal frequency value (a total of N*M for each frequency value).
- Output from the `.PROBE` statement is written to a .hb# file. This data is against the IFB points.
- Reported performance log statistics are written to a .lis file:
  - Number of nodes
  - Number of FFT points
  - Number of equations
  - Memory in use
  - CPU time
  - Maximum Krylov iterations

- Maximum Krylov dimension
- Target GMRES residual
- GMRES residual
- Actual Krylov iterations taken
- Frequency (swept input frequency values).

Errors and Warnings

The following error and warning messages are used when HSPICE encounters a problem with a HBAC analysis.

Error Messages

HBAC frequency sweep includes negative frequencies. HBAC allows only frequencies that are greater than or equal to zero.

No HB statement is specified (error at parser). HBAC requires an HB statement to generate the steady-state solution.

Warning Messages

More than one HBAC statement (warning at parser). HSPICE RF uses only the last HBAC statement in the netlist.

No HBAC sources are specified (error at parser). HBAC requires at least one HBAC source.

GMRES Convergence Failure. When GMRES (Generalized Minimum Residual) reaches the maximum number of iterations and the residual is greater than the specified tolerance. The HBAC analysis generates a warning and then continue as if the data were valid. This warning reports the following information:

- Final GMRES Residual
- Target GMRES Residual
- Maximum Krylov Iterations
- Actual Krylov Iterations taken

HBAC Example

The following example is shipped with the HSPICE RF distribution as *mix_hbac.sp* and is available in directory: $*installdir*/demo/hspicerf/examples.

```
* Test HBAC: ideal I,Q mixer
.OPTIONS PROBE
.OPTIONS POST=2
vlo lo 0 1.0 cos(1.0 0.5 1g) TRANFORHB=1 $ Periodic, Large-Signal
Input
rlo lo 0 1.0
rrf rf 0 1.0 $ Noise source
rrf1 rf1 rf 1.0 $ Noise source
g1 0 if cur='1.0*v(lo)*v(rf)' $ mixer element
c1 0 if q='1.0e-9*v(lo)*v(rf)' $ mixer element
rout if ifg 1.0
vctrl ifg 0 0.0
h1 out 0 vctrl 1.0
rh1 out 0 1.0
vrf rf1 0 hbac .001 0 $ Small signal source
.hb tones=1.0g nharms=3
.hbac lin 1 0.8g 0.8g
.print hb v(rf1) v(lo) v(out)
.probe hb v(rf1) v(lo) v(out)
.measure hb vout1 find v(out)[1,-1] at=0.8g
.end
```

Shooting Newton AC Analysis (.SNAC)

You use the Shooting Newton AC (`.SNAC`) statement for analyzing linear behavior in large-signal periodic systems. The `.SNAC` statement uses a periodic AC (PAC) and Shooting Newton algorithm to perform linear analysis of nonautonomous (driven) circuits, where the linear coefficients are modulated by a periodic, steady-state signal.

The following section describes the periodic AC analysis based on a Shooting Newton algorithm. This functionality is similar to the Harmonic Balance (HBAC) for periodic AC analysis.

The following section discuss these topics:

- Prerequisites and Limitations
- Input Syntax

- Output Syntax
- SNAC Output Data Files
- Errors and Warnings
- SNAC Example

Prerequisites and Limitations

The following prerequisites and limitations apply to SNAC:

- Requires one and only one .SNAC statement. If you use multiple .SNAC statements, HSPICE RF uses only the last .SNAC statement.
- Requires one and only one .SN statement.
- Requires placing the parameter sweep in the .SN statement.
- Requires at least one Periodic source.
- Limited to simulations that can be reduced to a single tone SN analysis.
- Supports unlimited number of sources.
- The requested maximum harmonic in a .PROBE or .PRINT statement must be less than or equal to half the number of harmonics specified in the SN statement (that is, max_harm $\leq$ nharms / 2).

Input Syntax

```
.SNAC frequency_sweep
```

Parameter	Description
frequency_sweep	Frequency sweep range for the input signal (also referred to as the input frequency band (IFB) or fin). You can specify LIN, DEC, OCT, POI, or SWEEPBLOCK. Specify the nsteps, start, and stop frequencies using the following syntax for each type of sweep: ▪ LIN *nsteps start stop* ▪ DEC *nsteps start stop* ▪ OCT *nsteps start stop* ▪ POI *nsteps freq_values* ▪ SWEEPBLOCK *nsteps freq1 freq2 ... freqn* ▪ DATA=*dataname*

Output Syntax

This section describes the syntax for the SNAC .PRINT and .PROBE statements. These statements are similar to those used for HB analysis.

.PRINT and .PROBE Statements

```
.PRINT SN TYPE(NODES | ELEM)[INDICES]
.PROBE SN TYPE(NODES | ELEM)[INDICES]
```

Parameter	Description
TYPE	Specifies a harmonic type node or element. TYPE can be one of the following: ▪ Voltage type – V = voltage magnitude and phase in degrees VR = real component VI = imaginary component VM = magnitude VP - Phase in degrees VPD - Phase in degrees VPR - Phase in radians VDB - dB units VDBM - dB relative to 1 mV ▪ Current type – I = current magnitude and phase in degrees IR = real component II = imaginary component IM = magnitude IP - Phase in degrees IPD - Phase in degrees IPR - Phase in radians IDB - dB units IDBM - dB relative to 1 mV ▪ Power type – P ▪ Frequency type – hertz[index], hertz[index1, index2, ...] You must specify the harmonic index for the hertz variable. The frequency of the specified harmonics is dumped.

Parameter	Description
NODES | ELEM	NODES or ELEM can be one of the following: ■ Voltage type – a single node name (n1), or a pair of node names, (n1,n2) ■ Current type – an element name (elemname) ■ Power type – a resistor (resistorname) or port (portname) element name ■ Frequency type – the harmonic index for the hertz variable. The frequency of the specified harmonics is dumped.
INDICES	Index to tones in the form [n1, +/-1]. ■ n1 is the index of the SN tone ■ +/-1 is the index of the SNAC tone Wildcards are not supported if this parameter is used. You can transform SN data into the time domain and output by using the following syntax:.PRINT SNTRAN ov1 [ov2 ... ovN]. PROBE SNTRAN ov1 [ov2 ... ovN]. See TYPE above for voltage and current type definitions.

SNAC Output Data Files

A SNAC analysis produces these output data files:

- Output from the `.PRINT` statement is written to a *.printsnac#* file.
- This data is against the IFB points.
- The header contains the large-signal fundamental and the range of small-signal frequencies.
- The columns of data are labeled as F(Hz), followed by the output variable names. Each variable name has the associated mixing pair value appended. All N variable names and all M mixing pair values are printed for each swept small-signal frequency value (a total of N*M for each frequency value).
- Output from the `.PROBE` statement is written to a *.snac#* file.

Reported performance log statistics are written to a *.lis* file:

- Number of nodes
- Number of FFT points
- Number of equations

- Memory in use
- CPU time
- Maximum Krylov iterations
- Maximum Krylov dimension
- Target GMRES residual
- GMRES residual
- Actual Krylov iterations taken
- Frequency (swept input frequency values)

Errors and Warnings

The following error and warning messages are used when HSPICE encounters a problem with a SNAC analysis.

Error Messages

SNAC frequency sweep includes negative frequencies. SNAC allows only frequencies that are greater than or equal to zero.

No SN statement is specified (error at parser). SNAC requires an SN statement to generate the steady-state solution.

Warning Messages

More than one SNAC statement (warning at parser). HSPICE RF uses only the last SNAC statement in the netlist.

No SNAC sources are specified (error at parser). SNAC requires at least one SNAC source.

GMRES Convergence Failure. When GMRES (Generalized Minimum Residual) reaches the maximum number of iterations and the residual is greater than the specified tolerance. The SNAC analysis generates a warning and then continue as if the data were valid. This warning reports the following information:

- Final GMRES Residual
- Target GMRES Residual

- Maximum Krylov Iterations
- Actual Krylov Iterations taken

SNAC Example

The following example is shipped with the HSPICE RF distribution as *mix_snac.sp* and is available in directory: $*installdir*/demo/hspicerf/examples.

```
* Test SNAC: ideal I,Q mixer -rrd
.OPTIONS PROBE
.OPTIONS POST=2
$.OPTIONS snmaxiter=100
.OPTIONS SNACCURACY=5
vlo lo 0 1.0 cos(1.0 0.5 1g) $ Periodic, Large-Signal SN Input
rlo lo 0 1.0
rrf rf 0 1.0 $ Noise source
rrf1 rf1 rf 1.0 $ Noise source
g1 0 if cur='1.0*v(lo)*v(rf)' $ mixer element
c1 0 if q='1.0e-9*v(lo)*v(rf)' $ mixer element
rout if ifg 1.0
vctrl ifg 0 0.0
h1 out 0 vctrl 1.0
rh1 out 0 1.0
vrf rf1 0 snac .001 24.0 $ Small signal for SNAC with 1-tone
SN Input
.sn tones=1.0g nharms=3
.snac lin 1 0.8g 0.8g
.print sn v(rf1) v(lo) v(out)
.print snfd v(rf1) v(lo) v(out)
.print snac v(rf1) v(lo) v(out)
.measure snac vout1 find v(out)[1,-1] at=0.8g
.measure snac vout2 find v(out)[0,1] at=0.8g
.measure snac vout3 find v(out)[1,1] at=0.8g
.measure sn vlo1 find v(lo) at=0.5n
.measure sn vlo2 find v(lo) at=1n
.measure snfd vlo3 find v(lo)[1] at=1
.end
```

Phase Noise and Buffer Chains

Phase noise is specific to oscillators. However, for buffer chains you can do periodic noise analysis. Phase noise that may be contributed by buffer,

amplifier, divider, or multiplier circuits is often referred to as *residual phase noise*. There are three commands in HSPICE RF that can apply help you predict such noise: .HBNOISE, .SNNOISE, and .PTDNOISE. These analyses compute the noise spectral density at an output variable taking into consideration modulation effects.

.HBNOISE and .SNNOISE compute the average noise over one period, assuming your circuit is driven by a periodic input signal.

.PTDNOISE computes the noise at one specific time point within the period, or it can compute the noise as a function of time over one period. .PTDNOISE can also convert the noise to a “jitter” value, which would be an uncertainty in the timing as opposed to the uncertainty in the output voltage.

See the following sections for detailed information on these commands:

- Multitone Harmonic Balance Noise (.HBNOISE)
- Shooting Newton Noise Analysis (.SNNOISE)
- Periodic Time-Dependent Noise Analysis (.PTDNOISE)

Multitone Harmonic Balance Noise (.HBNOISE)

An HBNOISE (Harmonic Balance noise) analysis simulates the noise behavior in periodic systems and is designed for use with driven circuits. It employs a Periodic AC (PAC) algorithm to perform noise analysis of nonautonomous (driven) circuits under periodic, steady-state tone conditions. This can be extended to quasi-periodic systems having more than one periodic, steady-state tone. One application for a multitone HBNOISE analysis is determining mixer noise figures under the influence of a strong interfering signal.

The PAC method simulates noise assuming that the stationary noise sources and/or the transfer function from the noise source to a specific output are periodically modulated.

- The modulated noise source (thermal, shot, or flicker) is modeled as a cyclostationary noise source.
- A PAC algorithm solves the modulated transfer function.
- You can also use the HBNOISE PAC method with correlated noise sources, including the MOSFET level 9 and level 11 models, and the behavioral noise source in the G-element (Voltage Dependent Current Source).

You use the `.HBNOISE` statement to perform a Periodic Noise Analysis.

The following sections discuss these topics:

- Supported Features
- Input Syntax
- Output Syntax
- Output Data Files
- Measuring HBNOISE Analyses with .MEASURE
- Errors and Warnings
- HBNOISE Example

Supported Features

HBNOISE supports the following features:

- All existing HSPICE RF noise models.
- Uses more than one single-tone, harmonic balance to generate the steady-state solution.
- Unlimited number of HB sources (using the same tone, possibly multiple harmonics).
- Includes stationary, cyclostationary, frequency-dependent, and correlated noise effects.
- Swept parameter analysis.
- Results are independent of the number of HBAC sources in the netlist.

Prerequisites and Limitations

The following prerequisites and limitations apply to HBNOISE:

- Requires one `.HB` statement (which determines the steady-state solution).
- Requires at least one HB source or one TRANFORHB source.
- Requires placing the parameter sweep in the `.HB` statement.
- The requested maximum harmonic in `.HBNOISE` must be less than or equal to half the number of harmonics used in harmonic balance (that is, *max_harm* <= *num_hb_harms*/2).

Input Syntax

```
.HBNOISE [output] [insrc] [parameter_sweep]
+ [n1, n2, ..., nk,+/-1]
+ [listfreq=(frequencies|none|all)] [listcount=val]
+ [listfloor=val] [listsources=on|off]
```

Parameter	Description
output	Output node, pair of nodes, or 2-terminal element. HSPICE RF references equivalent noise output to this node (or pair of nodes). Specify a pair of nodes as V(n+,n-). If you specify only one node, V(n+), then HSPICE RF assumes that the second node is ground. You can also specify a 2-terminal element name that refers to an existing element in the netlist.
insrc	An input source. If this is a resistor, HSPICE RF uses it as a reference noise source to determine the noise figure. If the resistance value is 0, the result is an infinite noise figure.
parameter_sweep	Frequency sweep range for the input signal. Also referred to as the input frequency band (IFB) or *fin*). You can specify LIN, DEC, OCT, POI, SWEEPBLOCK, DATA, MONTE, or OPTIMIZE sweeps. Specify the nsteps, start, and stop frequencies using the following syntax for each type of sweep: ▪ LIN *nsteps start stop* ▪ DEC *nsteps start stop* ▪ OCT *nsteps start stop* ▪ POI *nsteps freq_values* ▪ SWEEPBLOCK *nsteps freq1 freq2 ... freqn*

Parameter	Description
n1,n2,...,nk, +/-1	Index term defining the output frequency band (OFB or *fout*) at which the noise is evaluated. Generally, fout=ABS(n1*f+n2*f2+...+nk*fk+/-fin) Where: ▪ f1,f2,...,fk are the first through k-th steady-state tones determined from the harmonic balance solution ▪ fin is the IFB defined by *parameter_sweep*. The default index term is [1,1,...1,-1]. For a single tone analysis, the default mode is consistent with simulating a low-side, down conversion mixer where the RF signal is specified by the IFB and the noise is measured at a down-converted frequency that the OFB specifies. In general, you can use the [n1,n2,...,nk,+/-1] index term to specify an arbitrary offset. The noise figure measurement is also dependent on this index term.

listfreq	Prints the element noise value to the .lis file. You can specify at which frequencies the element noise value is printed. The frequencies must match the sweep_frequency values defined in the *parameter_sweep*, otherwise they are ignored.
	
	In the element noise output, the elements that contribute the largest noise are printed first. The frequency values can be specified with the NONE or ALL keyword, which either prints no frequencies or every frequency defined in *parameter_sweep*. Frequency values must be enclosed in parentheses. For example:`listfreq=(none)`
	`listfreq=(all)`
	`listfreq=(1.0G)`
	`listfreq=(1.0G, 2.0G)` The default value is NONE.

listcount	Prints the element noise value to the .lis file, which is sorted from the largest to smallest value. You do not need to print every noise element; instead, you can define `listcount` to print the number of element noise frequencies. For example, `listcount=5` means that only the top 5 noise contributors are printed. The default value is 1.

Parameter	Description
listfloor	Prints the element noise value to the *.lis* file and defines a minimum meaningful noise value (in $V/Hz^{1/2}$ units). Only those elements with noise values larger than `listfloor` are printed. The default value is 1.0e-14 $V/Hz^{1/2}$.
listsources	Prints the element noise value to the *.lis* file when the element has multiple noise sources, such as a FET, which contains the thermal, shot, and 1/f noise sources. You can specify either ON or OFF: ON Prints the contribution from each noise source and OFF does not. The default value is OFF.

Output Syntax

The HSPICE RF HB and SN noise analyses can output the output noise (onoise), noise figures (NF, SSNF and DSNF) and, the input referred noise (inoise). This section describes the syntax for the HBNOISE `.PRINT` and `.PROBE` statements.

.PRINT and .PROBE Statements

```
.PRINT HBNOISE [ONOISE] [NF] [SSNF] [DSNF] [INOISE]
.PROBE HBNOISE [ONOISE] [NF] [SSNF] [DSNF] [INOISE]
```

Parameter	Description
ONOISE	Outputs the voltage noise at the output frequency band (OFB) across the output nodes in the .HBNOISE statement. The data is plotted as a function of the input frequency band (IFB) points. Units are in $V/Hz^{1/2}$. Simulation ignores ONOISE when applied to autonomous circuits.
NF SSNF	NF and SSNF both output a single-side band noise figure as a function of the IFB points: NF = SSNF = 10 Log(SSF) Single side-band noise factor, SSF = {(Total Noise at output, at OFB, originating from all frequencies) - (Load Noise originating from OFB)} / (Input Source Noise originating from IFB).

Parameter	Description
DSNF	DSNF outputs a double side-band noise figure as a function of the IFB points. DSNF = 10 Log(DSF) Double side-band noise factor, DSF = {(Total Noise at output, at the OFB, originating from all frequencies) - (Load Noise originating from the OFB)} / (Input Source Noise originating from the IFB and from the image of IFB).
INOISE	Outputs input referred noise which can be printed, probed, or measured.

Output Data Files

An HBNOISE analysis produces these output data files:

- Output from the `.PRINT` statement is written to a .printpn# file.
- Output from the `.PROBE` statement is written to a .pn# file.

  Both the *.printpn# and *.pn# files output data against the input frequency band points.
- Standard output information is written to a .lis file:
  - simulation time
  - HBNOISE linear solver method
  - HBNOISE simulation time
  - total simulation time

See also Using Noise Analysis Results as Input Noise Sources.

Measuring HBNOISE Analyses with .MEASURE

Note: A `.MEASURE HBNOISE` statement cannot contain an expression that uses a `HBNOISE` variable as an argument. Also, you cannot use a `.MEASURE HBNOISE` statement for error measurement and expression evaluation of `HBNOISE`.

The .MEASURE HBNOISE syntax supports several types of measurements:

- Find-when

```
.MEASURE HBNOISE result FIND out_var1
+ AT = Input_Frequency_Band value
```

  The previous measurement yields the result of a variable value at a specific IFB point.

```
.MEASURE HBNOISE result FIND out_var1
+ WHEN out_var2 = out_var3
```

  The previous measurement yields the result at the input frequency point when *out_var2 == out_var3*.

```
.MEASURE HBNOISE result WHEN out_var2 = out_var3
```

  The previous measurement yields the input frequency point when *out_var2 == out_var3*.

- Average, RMS, min, max, and peak-to-peak

```
.MEASURE HBNOISE result [RMS] out_var [FROM = IFB1]
+ [TO = IFB2]
```

- Integral evaluation

```
.MEASURE HBNOISE result INTEGRAL out_var
+ [FROM = IFB1] [TO = IFB2]
```

  This measurement integrates the *out_var* value from the IFB1 frequency to the IFB2 frequency.

- Derivative evaluation

```
.MEASURE HBNOISE result DERIVATIVE out_var AT = IFB1
```

  This measurement finds the derivative of *out_var* at the IFB1 frequency point.

  Note: .MEASURE HBNOISE cannot contain an expression that uses an *hbnoise* variable as an argument. You also cannot use .MEASURE HBNOISE for error measurement and expression evaluation of HBNOISE.

- Input referred noise

```
.MEASURE [HBNOISE|SNNOISE] result FIND inoise
+ AT = IFB_value
```

This measurement yields the result of the input referred noise at a specific input frequency band point.

```
.MEASURE [HBNOISE|SNNOISE] result FIND inoise
+ WHEN out_var2 = out_var3
```

This measurement yields the result at the input frequency point when out_var2 == out_var3.

```
.MEASURE HBNOISE result func inoise [FROM = IFB1]
+ [TO = IFB2]
```

Where `func` is one of the following measurement types:

- AVG (average): Calculates the area under the inoise curve, divided by the periods of interest.
- MAX (maximum): Reports the maximum value of inoise over the specified interval.
- MIN (minimum): Reports the minimum value of inoise over the specified interval.
- PP (peak-to-peak): Reports the maximum value, minus the minimum value of inoise over the specified interval.
- RMS (root mean squared): Calculates the square root of the area under the inoise curve, divided by the period of interest.

```
.MEASURE HBNOISE result INTEGRAL inoise
+ [FROM =IFB1] [TO = IFB2]
```

This measurement integrates the inoise value from the IFB1 frequency to the IFB2 frequency.

```
.MEASURE HBNOISE result DERIVATIVE inoise AT = IFB1
```

This measurement finds the derivative of inoise at the IFB1 frequency point.

The HSPICE RF optimization flow can read the measured data from a `.MEASURE HBNOISE` analysis. This flow can be combined in the HSPICE RF optimization routine with a `.MEASURE HBTR` analysis (see Using .MEASURE with .HB Analyses on page 124) and a `.MEASURE PHASENOISE` analysis (see Measuring Phase Noise with .MEASURE PHASENOISE on page 172).

Errors and Warnings

HBNOISE Errors

See the list of HBAC Errors and Warnings on page 194.

HBNOISE Example

This example performs an HB analysis, then runs an HBNOISE analysis over a range of frequencies, from 9.0e8 to 9.2e8 Hz. Simulation outputs the output noise at V(out) and the single side-band noise figure versus IFB, from 1e8 to 1.2e8 Hz, to the *.pn0 file. The netlist for this example is shown immediately following.

```
$$*-Ideal mixer + noise source
$ prints total noise at the output (1.57156p V/sqrt-Hz),
$ single-sideband noise figure, (3.01 dB)
$ double-sideband noise figure. (0 dB)
.OPTION PROBE
.OPTION POST=2
vlo lo 0 0.0 hb 1.0 0 1 1$ Periodic, HB Input
Ilo lo 0 0
rsrc rfin rf1 1.0$ Noise source
c1 0 if q='1.0e-9*v(lo)*v(rfin)' $ mixer element
g1 0 if cur='1.0*v(lo)*v(rfin)'  $ mixer element
rout if 0 1.0
vrf rf1 0 $ hbac 2.0 0.0
.hb tones=1.0g nharms=4 $ sweep mval 1 2 1
.HBNOISE rout rsrc lin 11 0.90g 0.92g
.print HBNOISE onoise ssnf dsnf
.probe HBNOISE onoise ssnf dsnf
.end
```

Shooting Newton Noise Analysis (.SNNOISE)

A SNNOISE (Shooting Newton noise) analysis simulates the noise behavior in periodic systems. It uses a Periodic AC (PAC) algorithm to perform noise analysis of nonautonomous (driven) circuits under periodic, steady-state tone conditions. SNNOISE is similar to the HBNOISE analysis.

The PAC method simulates noise assuming that the stationary noise sources and/or the transfer function from the noise source to a specific output are periodically modulated.

- The modulated noise source (thermal, shot, or flicker) is modeled as a cyclostationary noise source.
- A PAC algorithm solves the modulated transfer function.
- You can also use the SNNOISE PAC method with correlated noise sources, including the MOSFET Level 9 and Level 11 models, and the behavioral noise source in the G-element (Voltage Dependent Current Source).

You use the `.SNNOISE` statement to perform a Periodic Noise Analysis.

The following sections discuss these topics:

- Supported Features
- Input Syntax
- Output Syntax
- Output Data Files
- Measuring SNNOISE Analyses with .MEASURE
- SNNOISE Analysis Example

Supported Features

SNNOISE supports the following features:

- All existing HSPICE RF noise models.
- Uses Shooting Newton to generate the steady-state solution.
- Unlimited number of sources.
- Includes stationary, cyclostationary, frequency-dependent, and correlated noise effects.
- Swept parameter analysis.
- Results are independent of the number of SNAC sources in the netlist.

Prerequisites and Limitations

The following prerequisites and limitations apply to SNNOISE:

- Requires one `.SN` statement (which determines the steady-state solution).
- Requires at least one Periodic source. Does not recognize HB sources.
- Requires placing the parameter `sweep` in the `.SN` statement.

Input Syntax

```
.SNNOISE [output] [insrc] [parameter_sweep]
+ [n1+/-1]
+ [listfreq=(frequencies|none|all)] [listcount=val]
+ [listfloor=val] [listsources=on|off]
```

Parameter	Description
output	Can be an output node, pair of nodes, or 2-terminal element. HSPICE RF references the equivalent noise output to this node (or pair of nodes). Specify a pair of nodes as V(n+,n-). If you specify only one node, V(n+), then HSPICE RF assumes the second node is ground. You can also specify a 2-terminal element name that refers to an existing element in the netlist.
insrc	An input source. If this is a resistor, HSPICE RF uses it as a reference noise source to determine the noise figure. If the resistance value is 0, the result is an infinite noise figure.
parameter_sweep	Frequency sweep range for the input signal. Also referred to as the input frequency band (IFB) or *fin*). You can specify LIN, DEC, OCT, POI, SWEEPBLOCK, DATA, MONTE, or OPTIMIZE sweeps. Specify the nsteps, start, and stop frequencies using the following syntax for each type of sweep: ▪ LIN *nsteps start stop* ▪ DEC *nsteps start stop* ▪ OCT *nsteps start stop* ▪ POI *nsteps freq_values* ▪ SWEEPBLOCK *nsteps freq1 freq2 ... freqn*

Parameter	Description
n1,+/-1	Index term defining the output frequency band (OFB or *fout*) at which the noise is evaluated. Generally, fout=ABS(n1*f1+/-fin) Where: f1 is the fundamental harmonic (tone) determined in the Shooting Newton analysis n1 is the associated harmonic multiplier n1,n2,...,nk are the associated harmonic multipliers; n1 can be any non-negative integer ≤ nharm defined in the .SN statement; +/-1 is fixed, either +1 or -1 fin is the IFB defined by *parameter_sweep*. The default index term is [1,-1]. For a single tone analysis, the default mode is consistent with simulating a low-side, down conversion mixer where the RF signal is specified by the IFB and the noise is measured at a down-converted frequency that the OFB specifies. In general, you can use the [n1,+/-1] index term to specify an arbitrary offset. The noise figure measurement is also dependent on this index term. See Specifying Variant Indicesbelow. (See alsoMeasuring SNNOISE Analyses with .MEASURE.)

listfreq	Prints the element noise value to the .lis file. You can specify at which frequencies the element noise value is printed. The frequencies must match the sweep_frequency values defined in the *parameter_sweep*, otherwise they are ignored.
	
	In the element noise output, the elements that contribute the largest noise are printed first. The frequency values can be specified with the NONE or ALL keyword, which either prints no frequencies or every frequency defined in *parameter_sweep*. Frequency values must be enclosed in parentheses. For example:`listfreq=(none)`
	`listfreq=(all)`
	`listfreq=(1.0G)`
	`listfreq=(1.0G, 2.0G)`
	The default value is NONE.

Parameter	Description
listcount	Prints the element noise value to the .lis file, which is sorted from the largest to smallest value. You do not need to print every noise element; instead, you can define `listcount` to print the number of element noise frequencies. For example, `listcount=5` means that only the top 5 noise contributors are printed. The default value is 1.
listfloor	Prints the element noise value to the *.lis* file and defines a minimum meaningful noise value (in $V/Hz^{1/2}$ units). Only those elements with noise values larger than `listfloor` are printed. The default value is 1.0e-14 $V/Hz^{1/2}$.
listsources	Prints the element noise value to the *.lis* file when the element has multiple noise sources, such as a FET, which contains the thermal, shot, and 1/f noise sources. You can specify either ON or OFF: ON Prints the contribution from each noise source and OFF does not. The default value is OFF.

Specifying Variant Indices

.SNNOISE is the appropriate HSPICE RF analysis for the computation of noise at the output of a sample and hold circuit. When using .SNNOISE, you need to specify the indices as [0,1] instead of the default [1,-1]. When you specify the indices as [0,1], you get results that are “what you see is what you get” with respect to the frequency sweep specified in the .SNNOISE command. There are two more important things to consider when using .SNNOISE. You must use enough harmonics to resolve the clock edge and you will want a high density of SN time points. It is recommended that the number of time points be between 2 and 20 times the number of harmonics used. To increase the density of the time points during the .SN analysis, you can use the option DELMAX to specify a maximum time step.

For example, you can use the following settings:

```
.OPTION SNACCURACY=50
.OPTION DELMAX=5p

.SN TONES=5e6 nharms=2000 trinit=800n
.SNNOISE v(vo) vsin [0,1] dec 100 1e6 1e9
```

This example uses 100 points per decade for the frequency sweep instead of 1000 points per decade. This is to speed up the simulation. It does not affect the accuracy of the results.

Output Syntax

This section describes the syntax for the SNNOISE `.PRINT` and `.PROBE` statements.

.PRINT and .PROBE Statements

```
.PRINT SNNOISE [ONOISE] [NF] [SSNF] [DSNF] [INOISE]
.PROBE SNNOISE [ONOISE] [NF] [SSNF] [DSNF] [INOISE]
```

Parameter	Description
ONOISE	Outputs the voltage noise at the output frequency band (OFB) across the output nodes in the .SNNOISE statement. The data is plotted as a function of the input frequency band (IFB) points. Units are in $V/Hz^{1/2}$. Simulation ignores ONOISE when applied to autonomous circuits.
NF *SSNF*	NF and SSNF both output a single-side band noise figure as a function of the IFB points: NF = SSNF = 10 Log(SSF) Single side-band noise factor, SSF = {(Total Noise at output, at OFB, originating from all frequencies) - (Load Noise originating from OFB)} / (Input Source Noise originating from IFB).
DSNF	DSNF outputs a double side-band noise figure as a function of the IFB points. DSNF = 10 Log(DSF) Double side-band noise factor, DSF = {(Total Noise at output, at the OFB, originating from all frequencies) - (Load Noise originating from the OFB)} / (Input Source Noise originating from the IFB and from the image of IFB).
INOISE	Outputs input referred noise which can be printed, probed, or measured.

Output Data Files

An SNNOISE analysis produces these output data files:

- Output from the .PRINT statement is written to a .printsnpn# file.
- Output from the .PROBE statement is written to a *.snpn#* file.

  Both the *.printsnpn# and *.pn# files output data against the input frequency band points.
- Standard output information is written to a .lis file:
  - simulation time
  - SNNOISE linear solver method
  - SNNOISE simulation time
  - total simulation time

See also Using Noise Analysis Results as Input Noise Sources.

Measuring SNNOISE Analyses with .MEASURE

Note: A .MEASURE SNNOISE statement cannot contain an expression that uses a SNNOISE variable as an argument. Also, you cannot use a .MEASURE SNNOISE statement for error measurement and expression evaluation of SNNOISE.

The .MEASURE SNNOISE syntax supports four types of measurements:

- Find-when

```
.MEASURE SNNOISE result FIND out_var1
+ At = Input_Frequency_Band value
```

  The previous measurement yields the result of a variable value at a specific IFB point.

```
.MEASURE SNNOISE result FIND out_var1
+ WHEN out_var2 = out_var3
```

  The previous measurement yields the result at the input frequency point when *out_var2* == *out_var3*.

```
.MEASURE SNNOISE result WHEN out_var2 = out_var3
```

The previous measurement yields the input frequency point when *out_var2 == out_var3*.

- Average, RMS, min, max, and peak-to-peak

```
.MEASURE SNNOISE result [RMS] out_var [FROM = IFB1]
+ [TO = IFB2]
```

- Integral evaluation

```
.MEASURE SNNOISE result INTEGRAL out_var
+ [FROM = IFB1] [TO = IFB2]
```

This measurement integrates the *out_var* value from the IFB1 frequency to the IFB2 frequency.

- Derivative evaluation

```
.MEASURE SNNOISE result DERIVATIVE out_var AT = IFB1
```

This measurement finds the derivative of *out_var* at the IFB1 frequency point.

Note: `.MEASURE SNNOISE` cannot contain an expression that uses an *hbnoise* variable as an argument. You also cannot use `.MEASURE SNNOISE` for error measurement and expression evaluation of `SNNOISE`.

- Input referred noise

```
.MEASURE SNNOISE result FIND inoise
+ AT = IFB_value
```

This measurement yields the result of the input referred noise at a specific input frequency band point.

```
.MEASURE SNNOISE result FIND inoise
+ WHEN out_var2 = out_var3
```

This measurement yields the result at the input frequency point when out_var2 == out_var3.

```
.MEASURE SNNOISE result func inoise [FROM = IFB1]
+ [TO = IFB2]
```

Where `func` is one of the following measurement types:

- AVG (average): Calculates the area under the inoise curve, divided by the periods of interest.

- MAX (maximum): Reports the maximum value of inoise over the specified interval.
- MIN (minimum): Reports the minimum value of inoise over the specified interval.
- PP (peak-to-peak): Reports the maximum value, minus the minimum value of inoise over the specified interval.
- RMS (root mean squared): Calculates the square root of the area under the inoise curve, divided by the period of interest.

```
.MEASURE SNNOISE result INTEGRAL inoise
+ [FROM =IFB1] [TO = IFB2]
```

This measurement integrates the inoise value from the IFB1 frequency to the IFB2 frequency.

```
.MEASURE SNNOISE result DERIVATIVE inoise AT = IFB1
```

This measurement finds the derivative of inoise at the IFB1 frequency point.

SNNOISE Analysis Example

This example performs an SN analysis, then runs an SNNOISE analysis over a range of frequencies, from 9.0e8 to 9.2e8 Hz. Simulation outputs the output noise at V(out) and the single side-band noise figure versus IFB, from 9.0e8 to 9.2e8 Hz, to the *.pn0 file. The netlist for this example is shown immediately following.

```
*
$$*-Ideal mixer + noise source
$ prints total noise PSD at the output (2.47e-20 V^2) when q=0
$ single-sideband noise figure, (3.01 dB)
$ double-sideband noise figure. (0 dB)
.OPTION PROBE
.OPTION POST=2
vlo lo 0 0.0 cos (0 1.0 1.0g 0 0 0)
Ilo lo 0 0
rsrc rfin rf1 1.0$ Noise source
g1 0 if cur='1.0*v(lo)*v(rfin)' $ mixer element
c1 0 if q='1.0e-9*v(lo)*v(rfin)' $ mixer element
rout if 0 1.0
vrf rf1 0 $ hbac 2.0 0.0
.option delmax=0.002n
.SN tones=1G   nharms=4 trstab=10n
.SNNOISE rout rsrc lin 11 0.90g 0.92g
.probe SNNOISE onoise ssnf dsnf
.print SNNOISE onoise ssnf dsnf
.end
```

Periodic Time-Dependent Noise Analysis (.PTDNOISE)

While HBNOISE and SNNOISE calculate a time-averaged power spectral density, there are applications where a characterization of the time-dependence of the noise is required. These applications include computation of jitter associated with a noisy signal crossing a threshold and computation of the noise associated with discretization of an analog signal, which computes the noise in a periodically driven circuit at a point in time. Periodic Time-Dependent noise analysis (PTDNOISE) calculates the noise spectrum and the total noise at a point in time. Jitter in a digital threshold circuit can then be determined from the total noise and the digital signal slew rate.

Circuits driven by large periodic signals produce cyclostationary noise, that is, the noise characteristics are periodic in time. Cyclostationary noise can be characterized in several ways, with the particular application determining which is appropriate.[9] The time-average power spectral density (PSD) ignores frequency correlations in the noise, but is adequate when the fundamental frequency of the cyclostationary noise is much larger than the bandwidth of interest. The time-average PSD is calculated in the HBNOISE/SNNOISE analyses. [10]

The harmonic power spectral density (HPSD) or equivalently, the auto-correlation function, R(t1,t2), contains the correlation information between noise sidebands that is necessary to build behavioral cyclostationary noise sources and to separate the amplitude modulation (AM) and phase modulation (PM) noise components. (See Amplitude Modulation/Phase Modulation Separation for more information.

The time-dependent power spectral density (TDPSD) can be integrated over frequency to yield the time-dependent noise (TDN). TDN can then be used to determine jitter associated with a noisy signal crossing a threshold. PTDNOISE analysis allows the calculation of TDPSD, TDN, and jitter. In addition, you can calculate both the time-domain power spectral density (TDSN) and the integrated noise (time-dependent noise, TDN) at multiple time points.

By measuring the jitter associated with a noisy signal crossing a threshold, jitter is modeled by displacing the time in a noise free signal v(t) with a stochastic process j(t).

Equation 43 $$V_{jitter}(t) = v(t + j(t))$$

We can also determine the voltage at this node including the time-dependent noise n(t):

Equation 44 $$Vn(t) = v(t) + n(t)$$

by equating these two representations, expanding in a Taylor series, and dropping higher order terms, as follows:

Equation 45 $$V(t) + n(t) = v(t + j(t)) = v(t) + dv(t)/\ dt \cdot\ j(t) + ..$$

Equation 46 $$N(t) = dv(t)/\ dt \cdot\ j(t)$$

In terms of variances, jitter is then defined as:

Equation 47 $$Var(j(t)) = n^2(t)/\ (dv(t)/\ dt)^2$$

The following sections discuss these topics:

- PTDNOISE Input Syntax
- PTDNOISE Output Syntax and File Format
- Error Handling and Warnings
- Usage Example

PTDNOISE Input Syntax

```
.PTDNOISE output time_value [time_delta]
+ frequency_sweep
+ [listfreq=(frequencies|none|all)]
+ [listcount=val] [listfloor=val]
+ [listsources=on|off]
```

Parameter	Description
output	Is an output node, pair of nodes, or 2-terminal elements. HSPICE RF references the equivalent noise output to this node (or pair of nodes). Specify a pair of nodes as V(n+,n-); only one node as V(n+, n-). If you specify only one node, V(n+), then HSPICE RF assumes the second node is ground. You can also specify a 2-terminal element name that refers to an existing element in the netlist.
time_value	Time point at which time domain noise is evaluated. Specify either a time point explicitly, such as: TIME=value, where value is either numerical or a parameter name or a .MEASURE name associated with a time domain .MEASURE command located in the netlist. PTDNOISE uses the time point generated from the .MEASURE command to evaluate the noise characteristics. This is useful if you want to evaluate noise or jitter when a signal reaches some threshold value.
time_delta	Time point at which time domain noise is evaluated. Specify either a time point explicitly, such as: TIME=value, where value is either numerical or a parameter name or a .MEASURE name associated with a time domain .MEASURE command located in the netlist. PTDNOISE uses the time point generated from the .MEASURE command to evaluate the noise characteristics. This is useful if you want to evaluate noise or jitter when a signal reaches some threshold value.

Parameter	Description
frequency_sweep	Frequency sweep range for the output noise spectrum. The upper and lower limits also specify the integral range in calculating the integrated noise value. Specify LIN,DEC, OCT, POI, SWEEPBLOCK, DATA sweeps. Specify the nsteps, start, and stop frequencies using the following syntax for each type of sweep: ▪ LIN *nsteps start stop* ▪ DEC*nsteps start stop* ▪ OCT *nsteps start stop*POI *nsteps freq_values* ▪ SWEEPBLOCK *nsteps freq1 freq2 ... freqn* ▪ DATA data_name
listfreq	Prints the element noise value to the *.lis* file. This information is only printed if a noise spectrum is requested in a PRINT or PROBE statement. (See PTDNOISE Output Syntax and File Format.) You can specify which frequencies the element noise is printed. The frequencies must match the sweep_frequency values defined in the *frequency_sweep*, otherwise they are ignored. In the element noise output, the elements that contribute the largest noise are printed first. The frequency values can be specified with the NONE or ALL keyword, which either prints no frequencies or every frequency defined in *frequency_sweep*. Frequency values must be enclosed in parentheses. For example: ▪ listfreq=(none) ▪ listfreq=(all) ▪ listfreq=(1.0) ▪ listfreq=(1.0G, 2.0G) The default value is NONE.
listcount	Prints the element noise value to the *.lis* file, which is sorted from the largest to smallest value. You do not need to print every noise element; instead, you can define `listcount` to print the number of element noise frequencies. For example, `listcount=5` means that only the top 5 noise contributors are printed. The default value is 1.
listfloor	Prints the element noise value to the *.lis* file and defines a minimum meaningful noise value (in $V/Hz^{1/2}$ units). Only those elements with noise values larger than `listfloor` are printed. The default value is 1.0e-14 $V/Hz^{1/2}$.

Parameter	Description
listsources	Prints the element noise value to the *.lis* file when the element has multiple noise sources, such as a MOSFET, which contains the thermal, shot, and 1/f noise sources. You can specify either ON or OFF: ON prints the contribution from each noise source and OFF does not. The default value is OFF.

PTDNOISE Output Syntax and File Format

PTDNOISE output syntax allows for the output of a single parameter: onoise, where, onoise is the noise voltage spectral density at each frequency point specified by the frequency_sweep keyword for the time points specified by the TIME keyword. The units are $V\sqrt{Hz}$.

```
.PROBE PTDNOISE onoise
.PRINT PTDNOISE onoise
```

Parameter	Units	Description
onoise	$V\sqrt{Hz}$	Noise voltage spectral density at each frequency point specified by *frequency_sweep* at the time point specified by *time_value*

Output File Format

The following PTDNOISE output files are generated depending on the user input:

File	Description
*.printptn#	Writes output from the .PRINT statement when using HB to obtain the steady state solution
*.ptn#	Writes output from the .PROBE statement when using HB to obtain the steady state solution
*.printsnptn#	Reports output from the .PRINT statement when using SN to obtain the steady state solution.
*.snptn#	Writes output from the .PROBE statement when using SN to obtain the steady state solution.

File	Description
*.lis	Standard output file *.lis contains the following information: ■ Performance Statistics Log ■ Number of Nodes ■ Number of FFT Points ■ Number of Equations ■ Memory in use ■ Maximum Krylov iterations ■ Maximum Krylov Dimension ■ Target GMRES Residual ■ Gmres Residual ■ Actual Krylov Iterations taken ■ Frequency (swept input frequency values) Noise source contributions are listed sequentially and are controlled by the PTDNOISE command line parameters: listtime, listfreq, listcount, listfloor, and listsources.

.MEASURE Syntax and File Format

The syntax for .MEASURE PTDNOISE is:

```
.MEASURE PTDNOISE result [integnoise|jitter|slewrate]
```

.MEASURE PTDNOISE allows for the measurement of these parameters: integnoise, time-point, tdelta-value, slewrate, and strobed jitter.

Parameter	Units	Description
integnoise	V	Voltage noise integrated over a frequency range specified by frequency_range at the time point specified by TIME=*val*.
slewrate	v/sec	Output signal slewrate at the time point specified by TIME=*val*.
jitter	sec	Calculated from the noise voltage (integrated over the frequency range specified by frequency_range), divided by the slew rate at the same node(s), at the time point specified by TIME=*val*.

Note: .MEASURE PTDNOISE is ignored when a TIME=*sweep* is specified in the netlist and a warning message is issued.

Measure File Format

File	Description
*.msnptn#	Contains output from the .MEASURE statement when using .SN to obtain the steady state solution.

Error Handling and Warnings

Error messages are generated under the following circumstances:

- PTDNOISE frequency sweep includes negative frequencies. PTDNOISE allows only frequencies that are greater than or equal to zero.
- PTDNOISE time sweep includes negative times. PTDNOISE allows only time points that are greater than or equal to zero.
- No SN statement is specified (error at parser). PTDNOISE requires an SN statement to generate the steady-state solution.
- Incorrect match to .MEASURE statement.

A warning is issued for a PTDNOISE convergence failure. When the gmres solver reaches the maximum number of iterations and the residual is greater than the specified tolerance, PTDNOISE generates a warning and then continue as if the data were valid. The Warning reports the following information:

- Final GMRES Residual
- Target GMRES Residual
- Maximum Krylov Iterations
- Actual Krylov Iterations taken

Usage Example

The following test case illustrates the PTDNOISE analysis for a simple inverter.

```
* Simple RC + Inverter - rcInvPTDNoise.sp
* rrd Jan 03, 2007
* Simulates PSD(t,f) of a simple inverter
* sweep time points
.param f0 = 5.0e8
.sn tones=f0 nharms=4 trinit=10n
.PTDNOISE v(out1) TIME=lin 3 0 2n TDELTA=.1n dec 5 1e5 1e10
+ listfreq=(1e6,1e8)
+ listcount=1
+ listsources=ON

.MEASURE PTDNOISE strobejit STROBEJITTER onoise FROM = 1e4 TO =
1e10
$.measure SN t1 trig AT=0 targ v(out1) val=1.5 fall=1

.opt post
.probe ptdnoise onoise
.print ptdnoise onoise
.probe sn v(out1)

vd    vdd  0   3.0
.global vdd
vgate in0  0   COS(1.5 1.4 'f0'  0 0 0)
rin   in0 in1 50
rout  out1  0    .1g

xo1 in1 out1 inv

.subckt inv in out
m1 out in 0 0 n l=350e-9 w=4.5e-6
m2 out in vdd vdd p l=350e-9 w=4.5e-6
.ends

.MODEL N NMOS
+Level= 49 Tnom=27.0 version =3.1 TLEVC= 1
*
***

.MODEL P PMOS
+Level= 49 Tnom=27.0 version =3.1 TLEVC= 1

.end
```

Multitone Harmonic Balance Transfer Function Analysis (.HBXF)

The `.HBXF` command calculates the transfer function from a given source in the circuit to a designated output. Frequency conversion is calculated from the input frequencies to a single output frequency that is specified with the command. The relationship between the `.HBXF` command and the input/output is expressed in the following equation:

Equation 48

$$Y_m(j\omega_0) = \sum_{\omega \varepsilon W} HBXF_{m,n}(j\omega_0, j(\omega + \Delta\omega)) \cdot X_n(j(\omega + \Delta\omega))$$

Where:

- $HBXF_{m,n}(j\omega_0, j(\omega + \Delta\omega))$ is the transfer function from input port n to the output port m
- W is the set of all possible harmonics
- $\omega + \Delta\omega$ is the input frequency
- $\Delta\omega$ is the offset frequency
- m is the output node number
- n is the input node number
- ω_0 is the output frequency
- Y is the output (voltage or current)
- X is the input (voltage or current)

The following sections discuss these topics:

- Supported Features
- Input Syntax
- Output Syntax
- Output Data Files
- Example
- HBXF Test Listing

Supported Features

The .HBXF command supports the following features:

- All existing HSPICE RF models and elements
- Sweep parameter analysis
- Unlimited number of HB sources

Prerequisites and Limitations

The following prerequisites and limitations apply to the .HBXF command:

- Only one .HBXF statement is required. If you use multiple .HBXF statements, HSPICE RF only uses the last .HBXF statement.
- At least one .HB statement is required, which determines the steady-state solution.
- Parameter sweeps must be placed in .HB statements.

Input Syntax

.HBXF *out_var freq_sweep*

Parameter	Description
out_var	Specify i(2_*port_elem*) or V(*n1*[,*n2*])

Parameter	Description
freq_sweep	Frequency sweep range for the input signal (also referred to as the input frequency band (IFB or fin)). A sweep of type LIN, DEC, OCT, POI, or SWEEPBLOCK. Specify the nsteps, start, and stop frequencies using the following syntax for each type of sweep: ■ LIN *nsteps start stop* ■ DEC *nsteps start stop* ■ OCT *nsteps start stop* ■ POI *nsteps freq_values* ■ SWEEPBLOCK = *BlockName* Specify the frequency sweep range for the output signal. HSPICE RF determines the offset frequency in the input sidebands; for example, f1 = abs(fout - k*f0) s.t. f1<=f0/2 The f0 is the steady-state fundamental tone, and f1 is the input frequency.

Output Syntax

This section describes the syntax for the HBXF `.PRINT` and `.PROBE` statements.

.PRINT and .PROBE Statements

```
.PRINT HBXF TYPE (NODES | ELEM)
.PROBE HBXF TYPE (NODES | ELEM)
```

Parameter	Description
TYPE	TYPE can be one of the following: ■ TFV = existing source ■ TFI = placeholder value for the current source attached to the given node. The transfer function is computed on the output variables and input current or voltage.

Parameter	Description
NODES | ELEM	NODES or ELEM can be one of the following: ■ Voltage type – a single node name (n1), or a pair of node names, (n1,n2) ■ Current type – an element name (elemname) ■ Power type – a resistor (resistorname) or port (portname) element name.

Output Data Files

An HBXF calculation produces these output data files:

- Output from the `.PRINT` statement is written to a *.printxf#* file.
  - The output is in ohms, siemens, or undesignated units, and the header in the output file is Z(..). Y(..) or GAIN(..).
- Output from the `.PROBE` statement is written to an *.xf#* file.
- Reported performance log statistics are written to a .lis file:
  - HBXF CPU time
  - HBXF peak memory usage

Example

Based on the HB analysis, the following example computes the trans-impedance from `isrc` to `v(1)`.

```
.hb tones=1e9 nharms=4
.hbxf  v(1) lin 10 1e8 1.2e8
.print hbxf tfv(isrc)  tfi(n3)
```

HBXF Test Listing

```
* Test HBXF: nonlinear order-2 poly equation
.OPTIONS  PROBE
.OPTIONS POST=2
vlo lo 0 cos(0 1.0 1g 0 0) tranforhb=1
rlo lo 0 50
vrf1 rf1 0 0
rrf1 rf1 0 50
E1 out 0 POLY(2) lo 0 rf1 0  0 1 1 1 10 1
rout out 0 50
.hb tones=1g nharms=5
.hbxf v(out) lin 2 100meg 200meg
.print hb v(out) v(rf1) v(lo)
.print hbxf tfv(vrf1) tfv(vlo)
.end
```

Shooting Newton Transfer Function Analysis (.SNXF)

The `.SNXF` command calculates transfer functions from an arbitrary number of small signal sources to a designated output in a circuit under periodic steady state conditions. Frequency conversion is calculated from multiple input frequencies to a single output at a single frequency that is specified on the command line.

Prerequisites and Limitations

The following prerequisites and limitations apply to the .SNXF command:

- Only one .SNXF statement is required. If you use multiple .SNXF statements, HSPICE RF only uses the last one issued.
- At least one .SN statement is required, which determines the steady-state solution.
- Parameter sweeps must be placed in .SN statements.

The following sections discuss these topics:

- Input Syntax
- Output Syntax
- Output Data Files

- Example
- SNXF Test Listing

Input Syntax

`.SNXF` *`out_var freq_sweep`*

Parameter Description

Parameter	Description
out_var	Specify `i(2_`*`port_elem`*`)` or `V(`*`n1`*`[,`*`n2`*`])`
freq_sweep	Frequency sweep range for the input signal (also referred to as the input frequency band (IFB or fin)). A sweep of type LIN, DEC, OCT, POI, or SWEEPBLOCK. Specify the nsteps, start, and stop frequencies using the following syntax for each type of sweep: ▪ LIN *nsteps start stop* ▪ DEC *nsteps start stop* ▪ OCT *nsteps start stop* ▪ POI *nsteps freq_values* ▪ SWEEPBLOCK = *BlockName* Specify the frequency sweep range for the output signal. HSPICE RF determines the offset frequency in the input sidebands Fin, where Fin = abs(n*F0 +/- Fout). F0 is the steady-state fundamental tone, and Fout is the output frequency. SNXF then generates the transfer functions from all of the input sidebands (the Fin values) to the output frequency Fout.

Output Syntax

This section describes the syntax for the SNXF `.PRINT` and `.PROBE` statements.

.PRINT and .PROBE Statements

```
.PRINT SNXF TYPE(NODES | ELEM)
.PROBE SNXF TYPE(NODES | ELEM)
```

Parameter Description

TYPE can be one of the following:

- TFV = existing source
- TFI = placeholder value for the current source attached to the given node.

The transfer function is computed on the output variables and input current or voltage .NODES | ELEM NODES or ELEM can be one of the following:

- Voltage type – a single node name (n1), or a pair of node names, (n1,n2)
- Current type – an element name (elemname)
- Power type – a resistor (resistorname) or port (portname) element name

Output Data Files

An SNXF calculation produces these output data files:

- Output from the .PRINT statement is written to a *.printsnxf#* file. The output is in ohms, siemens, or undesignated units, and the header in the output file is Z(..). Y(..) or GAIN(..).
- Output from the .PROBE statement is written to a *.snxf#* file.

Reported performance log statistics are written to a *.lis* file:

- SNXF CPU time
- SNXF peak memory usage

Example

Based on the SN analysis, the following example computes the transimpedance from isrc to v(1).

```
.SN tones=1e9 nharms=4
.SNXF v(1) lin 10 1e8 1.2e8
print SNXF TFV(isrc) TFI(n3)
```

SNXF Test Listing

```
* Test SNXF: nonlinear order-2 poly equation
.OPTIONS  PROBE
.OPTIONS POST=2
vlo lo 0 cos(0 1.0 1g 0 0)
rlo lo 0 50
vrf1 rf1 0 0
rrf1 rf1 0 50
E1 out 0 POLY(2) lo 0 rf1 0  0 1 1 1 10 1
rout out 0 50
.opt delmax=.01n
.sn tones=1g nharms=5
.snxf v(out) lin 2 100meg 200meg
.print sn v(out) v(rf1) v(lo)
.print snxf tfv(vrf1) tfv(vlo)
.end
```

References

[1] S. Maas, *Nonlinear Microwave Circuits*, Chapter 3, IEEE Press, 1997.

[2] R. Gilmore and M.B. Steer, “Nonlinear Circuit Analysis Using the Method of Harmonic Balance - A Review of the Art, Part I, Introductory Concepts.” *International Journal of Microwave and Millimeter-wave Computer-Aided Engineering*, Volume 1, No. 1, pages 22-37, 1991.

[3] R. Gilmore and M.B. Steer, “Nonlinear Circuit Analysis Using the Method of Harmonic Balance - A Review of the Art. Part II. Advanced Concepts.” *International Journal of Microwave and Millimeter-wave Computer-Aided Engineering*, Volume 1, No. 2, pages 159-180, 1991.

[4] V. Rizzoli, F. Mastri, F. Sgallari, G. Spaletta, “Harmonic-Balance Simulation of Strongly Nonlinear Very Large-Size Microwave Circuits by Inexact Newton Methods,” *MTT-S Digest*, pages 1357-1360, 1996.

[5] S. Skaggs, *Efficient Harmonic Balance Modeling of Large Microwave Circuits*, Ph.D. thesis, North Carolina State University, 1999.

[6] R.S. Carson, *High-Frequency Amplifiers*, 2nd Edition, John Wiley & Sons, 1982

[7] S.Y. Liao, *Microwave Circuit Analysis and Amplifier Design*, Prentice-Hall, 1987.

[8] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, 1995.

[9] J. Roychowdhury, D. Long, and P. Feldmann, “Cyclostationary Noise Analysis of Large RF Circuits with Multitone Excitations,” *IEEE Journal of Solid-State Circuits*, volume 33, pages 324–336, March 1998.

[10] A. Demir, A. Sangiovanni-Vincentelli, “Analysis and Simulation of Noise in Nonlinear Electronic Circuits and Systems”, Kluwer Academic, 1998.

9

Transient Noise Analysis

Describes the HSPICE and HSPICE RF solutions to perform transient noise analysis and compute noise statistics and their variation over time for circuits driven with non-periodic waveforms.

Transient noise analysis shows the effect of noise on the signal magnitude. It is also useful to see how noise effects the timing of the signal. From the transient noise analysis results, jitter can be measured. The two jitter measurements are time interval error (TIE) and autocorrelation function. TIE measures the time-shift behavior relative to a reference signal. The autocorrelation function is used for tracking the relative time-shift behavior of the signal.

The chapter describes two approaches:

- Monte Carlo (default), where the device noise is modeled as uncorrelated random signal sources to predict the statistical characteristics of the circuit performance due to device noise.
- Stochastic Differential Equation (SDE), for advanced users, which makes a direct prediction of the actual statistics of the output waveforms.

HSPICE and HSPICE RF include several different algorithms for understanding circuit behavior based on noise generated internally by electronic devices and thermal noise. PHASENOISE analysis computes the effects noisy elements have on the output spectrum of oscillators. HBNOISE and SNNOISE analyses compute the small-signal variations that noise can create about large-signal steady-state operating conditions. Periodic time-domain noise (PTDNOISE)

analysis computes the noise statistics of a periodic signal, and how those statistics vary with time over the period of the steady-state signal.

Table 12 .TRANNOISE Analysis Supported Platforms

Linux RHEL	Linux SUSE	Sun	Windows
Yes	Yes	Yes	Yes

The following topics are discussed in these sections:

- Overview of HSPICE/HSPICE RF Transient Noise Analysis
- Modeling Frequency-Dependent Noise Sources
- Monte Carlo Noise Analysis
- Stochastic Differential Equation (SDE) Analysis
- Setting up a .TRANNOISE Analysis
- Jitter Measurements from .TRANNOISE Analysis Results
- Correlating Noise Results: .TRANNOISE (Monte Carlo) and .NOISE
- Error Handling, Error Recovery, Status Reporting
- References

Overview of HSPICE/HSPICE RF Transient Noise Analysis

A variety of noise measurements are desirable from circuit simulation. The traditional SPICE `.noise` analysis provides a measurement of the RMS noise voltage at an output node as a function of frequency. This RMS value is akin to a measurement of the standard deviation of an equivalent Gaussian distribution of noise present at the output node of interest due to the contributing random noise sources within the circuit. The `.noise` analysis is a small-signal analysis giving an output noise (onoise) value over the `.ac` frequency range. More advanced examples of noise measurements include Phase Noise and Timing Jitter. Timing Jitter, in particular, is a measurement of a clock or oscillator's random noise over a time interval. It represents the standard deviation (or variance) of the timing uncertainty (i.e. the random drift of the clock edges) as a function of time. It is therefore a time-domain noise measurement that is typically evaluated at each cycle or half-cycle. For some clocks, oscillators, and

PLLs, this measurement can be separated into time-independent and time-dependent contributions, and written as

Equation 49
$$\sigma^2(\tau) = \frac{2}{\omega_0^2}[R_\phi(0) - R_\phi(\tau)]$$

where $\sigma^2(\tau)$ represents the time-dependent variance, τ is the time advance, and $R_\phi(\tau)$ is the autocorrelation function related to the power spectrum of phase variations as

Equation 50
$$R_\phi(\tau) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S_\phi(\omega) e^{j\omega\tau} d\omega$$

These special relationships allow the computation of timing jitter from the results of `.phasenoise` analysis, since we can assume $L(f) \cong S_\phi(f)$, and therefore derive the time-varying noise from the frequency-domain phase noise simulation solution.

Other measurements of time-dependent or time-varying noise are also desirable for circuits other than clocks and oscillators, and for situations other than steady-state operation. The purpose of such measurements is usually similar: derive useful information on the time-varying statistical behavior of the circuit due to its internal noise sources.

Transient Noise Analysis is essentially a typical `.TRAN` simulation, but with all random noise sources within the circuit activated as contributing signal sources. Monte Carlo Noise analysis is a transient noise simulation approach that uses uncorrelated random signal sources for device noise in such a way that all noise signals can be seeded uniquely but repeatable from run-to-run to predict the statistical characteristics of circuit performance due to device noise. The resulting outputs are typically examined using histogram plots to measure the statistical behavior.

Techniques Available in HSPICE

HSPICE provides the following techniques for transient noise analysis:

1. Single-run Monte Carlo: (default) A single transient analysis that includes time-varying noise contributions to all output waveforms. With ergodic systems, the statistics of output variables are observable over time.

2. Multi-run Monte Carlo: Multiple transient analyses, each including time-varying noise, with unique seeding from run to run, form an ensemble of simulations. Output waveform statistics are observable across the ensemble at specific time points. For techniques 1 and 2 see Monte Carlo Noise Analysis.
3. Stochastic Differential Equation (SDE) Dynamic noise statistics are computed in the form of a time-varying covariance matrix. SDE techniques allow the output of a variance waveform for any output signal. Variance waveforms can be used to construct probability density plots. See Stochastic Differential Equation (SDE) Analysis.

Single-run Monte Carlo features include:

- Simulation includes statistically accurate noise source contributions from all devices.
- Adjustable bandwidth for noise source frequency responses.
- Fastest approach possible; based on single `.TRAN` analysis.
- Effective for ergodic simulations (viewing statistical variations over time).
- Use the following process for Single Monte Carlo Trannoise analysis
  - Run as typical `.TRAN` analysis.
  - Post process waveforms to measure resulting noise effects.
  - Use `FMIN` to set low-frequency flicker noise limits.
  - Use `FMAX` to set maximum noise source waveform bandwidth and ensure Nyquist sampling.
  - Vary seed values to re-run simulations with uniquely different noise waveforms.

SDE and Multi-run Monte Carlo approaches are useful for characterizing statistics at specific time events.

- SDE approach gives probability density information.
- Multi-run Monte Carlo provides data ensembles for histogram generation.

Modeling Frequency-Dependent Noise Sources

Transient noise techniques require that noise sources be modeled in the time domain. These techniques model device noise sources in terms of standard

white Gaussian noise processes. Let the standard white Gaussian noise process be written as $\zeta(t)$.

White noise source models can be calculated in terms of intensity functions. For example, if the (frequency dependent) noise from a time varying conductance is given by

Equation 51 $$\overline{i_n^2} = 4kTG(t)\Delta f$$

...then this can be modeled in terms of intensity as the random time-domain current given by

Equation 52 $$j_n = \sqrt{2kTG(t)}\zeta(t)$$

In the case of flicker noise, it is necessary to create the 1/f power spectrum of flicker noise sources by filtering a $\zeta(t)$ process. This can be accomplished with the following rational function transfer relationship:

Equation 53 $$H(s) = \prod_{i=1}^{N}\frac{(s+\omega_{zi})}{(s+\omega_{pi})}$$

Note that since all flicker sources can be modeled over the same frequency range (i.e., bandwidth), this same transfer function can be used for all sources. Tests have shown that the above fit is very accurate using one pole/zero per octave, and with reasonable fits using one pole/zero for every two octaves, or even with one pole/zero per decade. This fitting algorithm is therefore frequency range-specific, which is why the parameters FMAX and FMIN are used for specifying frequency ranges for Transient Noise Analysis similarly to the modeling of the frequency-dependent S-element.

Monte Carlo Noise Analysis

You can use two methods of Monte Carlo analysis to determine transient noise, single-sample (default) and multiple-sample:

Single-sample Monte Carlo method:

- Using single-sample Monte Carlo method
- Single transient analysis
- Time-varying noise contributes to all output variables
- Assuming the circuit is ergodic*, the statistics of output variables are observable over time
- With post-processing, results can be shown as TIE (Timing Interval Error) jitter and jitter spectrum

* *Ergodic – relating to a process in which every sequence or sizable sample is equally representative of the whole.*

Multiple-sample Monte Carlo method:

- Multiple transient analyses
- An *ensemble* of simulations is created, with enough samples to adequately observe the output waveform statistics due to noise
- Results are shown as histogram of statistical behavior

The equations for Monte Carlo transient noise analysis define the Monte Carlo approach:

Let the time-domain system of equations for transient analysis be given by the MNA system described by the following vector state equation:

Equation 54 $\mathrm{f}(\dot{x},\ \mathrm{x},\ t)\ =\ 0$

Consider this the noise-less system that is solved during a normal transient analysis. Transient noise analysis can be considered a similar analysis where we now inject noise from all device noise models to give the modified equation:

Equation 55 $\mathrm{f}(\dot{x},\ \mathrm{x},\ t)\ =\ -\mathrm{j}_n(t)$

This system reflects the additional noise sources added and can be solved in the same manner as transient analysis. However, if we consider each noise source to be related to a white Gaussian noise function, it is clear that we must create an ***ensemble*** of waveforms for our unknown *x(t)* vector in order to predict the statistics of the outputs. This type of simulation therefore involves generating multiple uncorrelated noise source waveforms for all noise sources, and then running multiple simulations in a Monte Carlo fashion.This approach is also known as Monte-Carlo Noise Simulation. This method cannot directly measure the statistics of output signals due to input noise (as with .NOISE), but

instead models noise sources as independent time-domain stimuli. Generating statistical information in this approach requires running a plurality of simulations over a variety of random noise-source sequences (to create an ensemble of output waveforms) and then analyzing the statistics of the ensemble using histograms of other plots. When the system behavior can be considered ***ergodic***, it is possible to run a single very long duration simulation in order to capture the statistical variations of the output signal over time (as with an eye-diagram). Monte-Carlo modeling of the noise sources can be done with a sum of sinusoids with random phases [2], or using random number generators with the appropriate statistical distributions[3]. An advantage of the Monte-Carlo approach is its ability to capture very nonlinear noise behaviors. This is useful, for example, when the responses of circuits with noise are known to have non-Gaussian variations about their noise-less simulations.

Stochastic Differential Equation (SDE) Analysis

Transient Noise Analysis is very useful for predicting waveform statistics at particular time points. One method to accomplish this is to run the multi-sample Monte Carlo approach, and then use the ensemble of simulation results to generate histograms at the time points of interest. Histograms that measure vertical distributions reveal voltage noise. Histograms that measure horizontal time-shift distributions reveal jitter. In those cases where it is desirable to have a very accurate distribution curve, it may be necessary to simulate with a large number of Monte Carlo samples. SDE analysis can provide a more efficient alternative in such cases. Instead of requiring a large ensemble of results, SDE performs special calculations that directly predict the statistics of the output waveforms.

Let the time-domain signal resulting from a regular transient analysis (i.e., with noise ignored) for a specific output node be written as $v^{s}_{out}(t)$.

Let the time-domain signal resulting from an analysis with signal and noise for the same output node be written as $v^{s+n}_{out}(t)$.

We can define the noise voltage component $v^{s}_{out}(t)$ to be:

Equation 56 $$v^{n}_{out}(t) = v^{s+n}_{out}(t) - v^{s}_{out}(t)$$

We can define a variance equal to the expected value of this noise component at a given time to be:

Equation 57 $$\sigma_n^2(t) = \overline{v_{out}^n(t) \cdot v_{out}^n(t)}$$

If we assume that the noise variations are small, we can create a linear Stochastic Differential Equation (SDE) for the noise contribution vector $x_n(t)$ (on entry in the vector being the output noise $v_{out}^n(t)$). This SDE may be formulated in terms of time-varying coefficient matrices that are evaluated for a normal transient analysis, which are functions of the noise-free solution vector $x_s(t)$ and derived from the noise-free transient analysis computations [1]:

Equation 58
$$A(t)x_n + C(t)\dot{x}_n + B(t)v \cong 0$$
$$x_n(0) = x_{n,0}$$

We can then create and solve a linear ordinary differential equation (ODE) system for the time-varying noise correlation matrix $K(t) = \overline{x_n(t)x_n(t)^T}$:

Equation 59 $$\dot{K}(t) = E(t)K(t) + F(t)F(t)^T$$

where $E(t)$ and $F(t)$ are derived from $A(t)$, $B(t)$, and $C(t)$, and T indicates the transpose operation.

In general, simultaneously solving for both the deterministic and stochastic differential equations can therefore give us the complete time-dependent output signal waveform vector $x_s(t)$, as well as the complete time-varying noise correlation/covariance matrix $K(t) = \overline{x_n(t)x_n(t)^T}$. The entries in this matrix represent time-dependent variance values $\sigma_n^2(t)$ for output signals $v_{out}^n(t)$. This output can be interpreted and plotted as a time-varying RMS noise voltage waveforms for $v^n{}_{RMS}(t) = \sqrt{\sigma_n^2(t)}$. The results of such a transient + noise analysis include the usual deterministic transient analysis waveforms, including the mean voltage output $v_{out}^s(t)$, and also its (stochastic) time-varying RMS noise component $v^n{}_{RMS}(t)$. In this sense, the SDE analysis method provides typical SPICE output waveforms for circuit unknowns, plus the additional

waveform representing the time-varying statistics of the circuit. The .TRANNOISE SDE method therefore activates this special analysis and makes this output available to the user.

Setting up a .TRANNOISE Analysis

The transient noise analysis requires an accompanying .tran analysis which determines the time-sampling, matrix solutions, and deterministic output waveforms. The .TRANNOISE command is used to activate transient noise and to compute the additional noise variables. Note that this is consistent with how .NOISE computes additional noise outputs when added to an .AC analysis.]

The following sections discuss these topics:

- Input Syntax
- Monte Carlo Output Data
- SDE Output Data
- Monte Carlo Examples
- SDE Examples

Input Syntax

Monte Carlo Single Sample Approach

```
.TRANNOISE output [METHOD=MC] [SEED=val]
+ [AUTOCORRELATION=0|1|off|on]
+ [FMIN=val] [FMAX=val] [SCALE=val]
```

Monte Carlo Multi-Sample Approach

```
.TRANNOISE output
+ [METHOD=MC] [SEED=val] [SAMPLES=val]
+ [AUTOCORRELATION=0|1|off|on]
+ [FMIN=val][FMAX=val][SCALE=val]
```

SDE Approach

```
.TRANNOISE output METHOD=SDE
+ [AUTOCORRELATION=0|1|off|on|]
+ [TIME=(all|val)]
+ [FMIN=val] [FMAX=val] [SCALE=val]
```

Keyword	Description
output	(Required) Output node, pair of nodes, or 2-terminal element. Noise calculations are referenced to this node (or node pair). Specify a pair of nodes as V(n+,n-). If you specify only one node, V(n+), then HSPICE RF reads the second node as ground. If you specify a 2-terminal element, the noise voltage across this element is treated as the output.
METHOD=MC | SDE	Specifies Monte Carlo or SDE transient noise analysis method. The default, or, if METHOD is not specified, is the single-sample Monte Carlo method. Specifying METHOD=SDE is required to select the transient noise analysis SDE method. METHOD=MC | SDE is position independent.
SAMPLES=*val*	Specifies the number of Monte Carlo samples to use for the analysis. The default, or if SAMPLES is not specified is 1, the single-sample Monte Carlo method. For the multi-sample Monte Carlo method, SAMPLES must be specified as greater than 1.
SEED=*val*	(Optional) Specifies the beginning simulation sample. Default=2, if value for SEED is not specified. Setting SEED=1 causes a noiseless simulation to be performed.
TIME	(Optional) Used to specify additional time points (breakpoints) where time-domain noise should be evaluated in addition to those time points that will be evaluated as part of the normal time-stepping algorithm. Use this parameter to force noise evaluation at important time points of interest (such as rising/falling edges). ▪ TIME=all: (default) causes time-domain noise ONOISE values to be computed and available for output at all time points selected by the .TRAN command time-step algorithm. ▪ TIME=val: Specifies a single additional time point at which time domain noise is measured. The value can be numeric or a parameter. A .TRAN analysis at this time point will be forced. Note that time-domain noise calculations require an accompanying .TRAN analysis at each time point. The TIME parameter may therefore add transient analysis time-points (breakpoints) as needed while values given outside the range of the .TRAN command constraints are ignored

Keyword	Description
AUTOCORRELATION	(Optional) Used to enable the autocorrelation function calculation at the specified output. ▪ AUTOCORRELATION=0 (OFF) - (default) Autocorrelation function is not calculated. ▪ AUTOCORRELATION=1 (ON) - Autocorrelation function is calculated at the specified output.
FMIN	(Optional) Base frequency used for modeling frequency dependent noise sources. Sets low-frequency flicker noise limit for contributing noise sources. (Default: 1/TSTOP) See Note below.
FMAX	(Optional) Maximum frequency used for modeling frequency dependent noise sources. Sets maximum noise source waveform bandwidth and ensures Nyquist sampling. Default: 1/TSTEP; See Note below.
SCALE	Scale factor that can be applied to uniformly amplify the intensity of all device noise sources to exaggerate their contributions. Default: 1.0

Note: FMAX has a dramatic effect on TRANNOISE, since it controls the amount of energy each noise source is allowed to emit. Therefore, huge values of FMAX (like 100G) can result in huge instantaneous noise levels. FMIN sets the low frequency limit for flicker noise, and therefore controls the energy in flicker noise sources. You can expect some significant differences with FMAX and FMIN changes: Noise power will increase linearly with FMAX; flicker noise power can scale as 1/FMIN.

Monte Carlo Output Data

The `.TRANNOISE` analysis outputs raw data to the **.tr0* and **.printtr0* files consistent with a Monte Carlo transient analysis. Data in these output files is organized according to the sample number which is indicated as the Monte Carlo index.

The first sample (index=1) is used to create a noise-free simulation, i.e., all noise sources are disabled for this simulation. Beginning with index=2, all subsequent simulations use unique random number seeding to create unique simulation results due to noise.

SDE Output Data

The `.TRANNOISE` SDE analysis outputs raw data to the **.tr0* and **.printtr0* files consistent with transient analysis. The `.PRINT/.PROBE` output syntax supports the following measurements:

```
.print trannoise ONOISE ONOISE(M) VRMS(n1[,n2])
.probe trannoise ONOISE ONOISE(M) VRMS(n1[,n2])
```

where:

- The ONOISE and ONOISE(M) outputs are the same. They represent noise voltage or current at the node or branch specified by the `output` keyword. The ONOISE represents the RMS noise voltage component (square root of the variance), units in Volts, of the noise at the specified output present in addition to the noise-less transient voltage.
- VRMS: The output of RMS noise voltages at other nodes (i.e., the output for general nodal noise voltage values).

Note that the actual instantaneous output voltage is the sum of the signal plus noise components:

Equation 60 $$v_{out}^{s+n}(t) = v_{out}^{s}(t) + v_{out}^{n}(t)$$

Where the noise component $v_{out}^{n}(t)$ has an assumed Gaussian distribution (in x) as:

$$\frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-x^2}{2\sigma^2}}$$

And the output signal $v_{out}^{s}(t)$ is that resulting from the (deterministic) `.TRAN` analysis. The time-varying RMS noise voltage waveforms (i.e., for `onoise`) are related to the variance at the specified outputs as given by $v_{RMS}^{n}(t) = \sqrt{\sigma_n^2(t)}$ where: $\sigma_n^2(t) = \overline{v_{out}^{n}(t) \cdot v_{out}^{n}(t)}$.

Monte Carlo Examples

The following example generates 30 Monte Carlo noise simulations beginning with a noiseless (index=1) simulation.

```
.TRANNOISE v(out) SAMPLES=30
```

The following example generates 20 monte carlo noise simulations starting with the seed value (i.e., index) of 31 for the first simulation.

```
.TRANNOISE v(out) SAMPLES=20 SEED=31
```

The following example generates a single noise simulation, with seed value of 50, with all noise sources amplified by a factor of 10.

```
.TRANNOISE v(out) SEED=50 SCALE=10.0
```

SDE Examples

Example 1. SDE method with maximum frequency of 5GHz.

```
.TRANNOISE v(7) METHOD=SDE FMAX=5g
```

Example 2. Activates SDE noise analysis, and dumps the `ONOISE` output to the **.tr0* file:

```
.TRANNOISE v(out) METHOD SDE
.PROBE TRANNOISE ONOISE
```

Example 3. Activates SDE noise analysis, placing a lower bound on flicker noise to be 10kHz, and an upper bound on all noise power at 100MHz:

```
.TRANNOISE v(out) METHOD=SDE FMIN=10k FMAX=100MEG
```

Jitter Measurements from .TRANNOISE Analysis Results

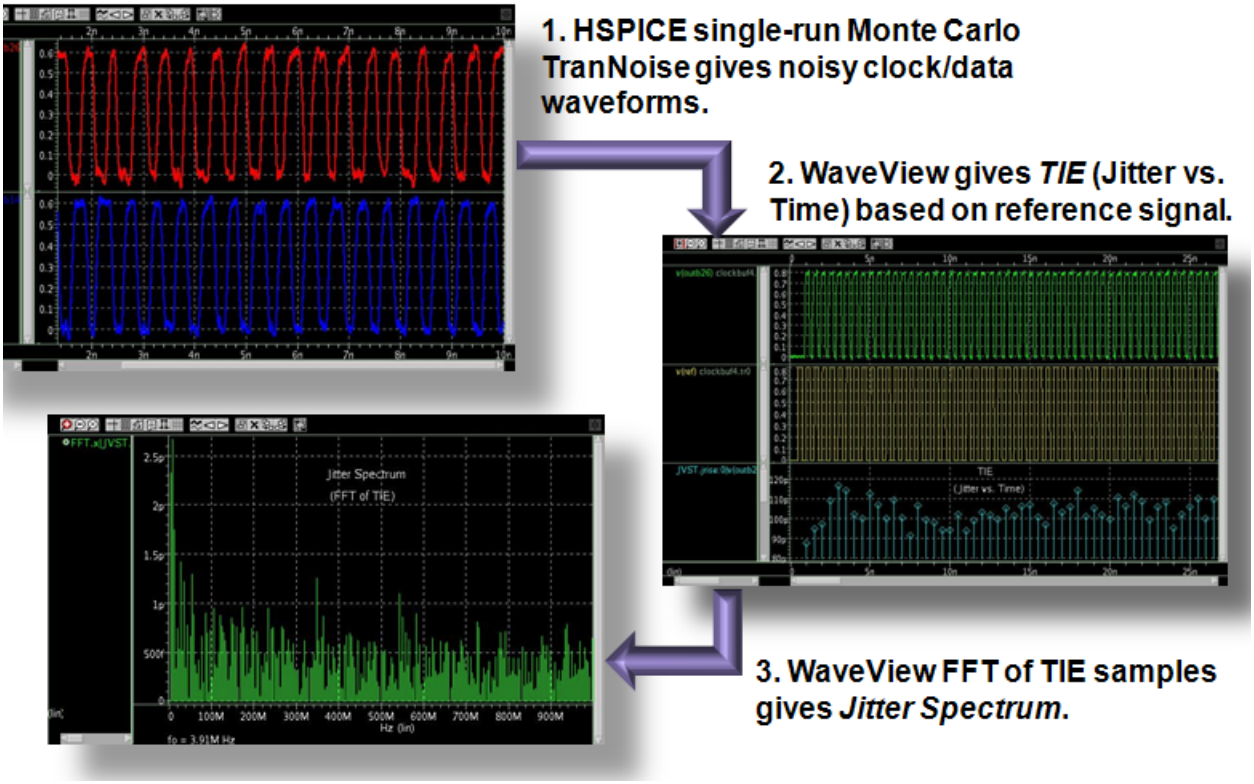
While transient noise analysis shows the effect of noise on the signal magnitude, it is also useful to see how the noise affects the timing of the signal. From the transient noise analysis results, jitter can be measured. The two jitter measurements are *time interval error* or TIE and *autocorrelation function*. TIE measures the time-shift behavior relative to a reference signal and is best

measured using WaveView. The autocorrelation function tracks the relative time-shift behavior of the signal.

Output Data Files

The output data from the autocorrelation function calculations is output to the ASCII formatted **.trnz#* file.

Measure TIE Jitter and Jitter Spectrum Example



The following clock buffer example circuit uses a single run Monte Carlo transient noise analysis. Post-processing is performed using WaveView.

Analysis Setup Details

```
* Transient Noise Analysis Example
.option post probe
.option runlvl=5
.paramfreq=2000MEG period='1.0/freq'
* Reference Clock
Vsrcref gndDC 0 PULSE (0.0 'vdd'
+'0.975*period' '0.05*period' '0.05*period'
+ '0.45*period' 'period')
* Clock Buffer Circuit
:
* Analysis Setup
.tran'0.05*period' '520*period'
.trannoisev(outb26) SWEEP MONTE=1 FIRSTRUN=2
+ FMAX=50G SCALE=10
.probe tranv(ref) v(outa) v(outb14) v(outb26)
.end
```

Figure 20 Single-run Monte Carlo Transient Noise gives noisy clock/data waveforms

Figure 21 WaveView: Jitter vs. Time measurement to get Timing Interval Error (TIE) Jitter vs. Time. Measurement based on the reference signal

Figure 22 FFT of the TIE Jitter samples show the Jitter Spectrum

Correlating Noise Results: .TRANNOISE (Monte Carlo) and .NOISE

Comparing noise results between `.TRANNOISE` and `.NOISE` simulations requires some careful setups:

1. Pay attention to the bandwidth (i.e., frequency sweep) that you are using for `.noise`. When you run `.trannoise`, it will always be bandwidth-limited based on the time-stepping (tsteps) and time interval (tstop) of the simulation. You want the same bandwidth to be reflected in the `.noise`, as well. (The reason why will be clear below.)
2. If possible, test your circuit for natural bandwidth limitations, i.e., the `onoise` output rolls off substantially beyond some frequency. This is true of post-layout circuits that have some capacitance at every node. If your circuit has natural BW limits, then item #1 above isn't so critical.
3. Run your `.noise` simulation, and monitor the Total Output Noise Voltage in units of Volts. This is the integrated `ONOISE` over the bandwidth of interest. `ONOISE` values are in Volts-per-unit-sqrt(Hertz). To compare with the time domain, you need to know the Hertz. The (integrated) total output noise voltage is dumped to the .lis file.
4. Set up your `.trannoise` simulation such that the noise will not alter the large-signal behavior of the circuit. `.NOISE` is a small-signal simulation. Try to keep `.trannoise` signals small (e.g., no big PULSE sources) so the comparisons will be valid. Note that with `.trannoise`, you don't need an input signal, as the noise will be the signal.
5. Use the `.trannoise` command; set FMAX to match the integration bandwidth (freq range) used for your `.NOISE` analysis. To see low frequency range (flicker effects), your .tran command may need a big TSTOP value. Do not set the FMIN parameter, since this sets the noise generation. You still need a big TSTOP to observe this generated noise.
6. To compare with a single-run Monte Carlo simulation, create a `.measure` command to measure RMS voltage for the same node(s) you used for `.noise` output. If you can't avoid a nonlinear transient startup for your circuit, you may need a FROM and TO for this `.measure` to look past it. Your `.measure` results will match your Total Output Noise Voltage if the only signal at this node is due to noise.

7. To compare with a multi-run Monte Carlo simulation, `.measure` the voltage at a particular time point of interest. When you run the simulation, the Monte Carlo report gives you results you can compare with `.noise`. You may need many samples to get stable statistical results for a single time point.

Error Handling, Error Recovery, Status Reporting

The following error checks are made with transient noise analysis:

- Verify that the specified output node exists.
- Verify non-negative integer values for “list” entries.

Note: Invalid values for FMIN and FMAX will be ignored and default values will be used.

The SDE solving can be substantially slower than transient analysis. Status reporting includes noise analysis progress.

References

[1] A. Demir, E.W.Y. Liu, A.L. Sangiovanni-Vincentelli, “Time-domain non-Monte Carlo noise simulation for nonlinear dynamic circuits with arbitrary excitations,” IEEE Trans. CAD, vol. 15, no. 5. May 1996.

[2] P. Bolcato and R. Poujois, “A new approach for noise simulation in transient analysis”, Proc. IEEE Int. Symp. Circuits Syst., pp. 887-890, May 1992.

[3] J. A. McNeill, “Jitter in ring oscillators,” Ph.D. dissertation, Boston University, 1994.

[4] M. Okumura and H. Tanimoto, “A time-domain method for numerical noise analysis of oscillators”, Proc. IEEE Asia Pacific Design Automation Conference, 1997.

[5] A. van der Ziel, Noise in Solid State Devices and Circuits, John Wiley & Sons, 1986.

[6] A. Hajimiri, S. Limotyrakis, and T.H. Lee, “Jitter and phase noise in ring oscillators,” IEEE J. Solid-State Circuits, vol. 34, no. 6, pp. 790-804, June 1999.

10

S-parameter Analyses

Describes how to do frequency translation and large-signal S-parameter extraction, as well as noise parameter calculation.

These topics are covered in the following sections:

- Frequency Translation S-Parameter (HBLIN) Extraction on page 256
- Large-Signal S-parameter (HBLSP) Analysis on page 263

This chapter discusses various techniques supported in HSPICE RF for extracting circuit scattering parameters. Since RF circuits can operate under large-signal and small-signal conditions, there are several types of scattering parameters that are useful to measure.

Linear small-signal scattering parameters represent the RF frequency-domain transfer characteristics for a circuit that is operating at its DC bias condition, but the stimulus and response signals are sufficiently small that they do not influence the operating point. This type of analysis is performed using the `.LIN` analysis, which is supported in both HSPICE and HSPICE RF. For information on doing small-signal S-parameter analysis (.LIN), please see (Linear Network Parameter Analysis) in the *HSPICE User Guide: Simulation and Analysis*.

In the case of RF mixers and receiver front-ends, some of the input and output frequencies of interest involve a frequency translation. This translation is intentional and caused by nonlinear mixing in the circuit due to devices being driven by large-signal periodic waveforms. This type of scattering parameter analysis therefore must begin by solving the large-signal periodic response, and then finding the small-signal behavior about this large-signal operating point. This capability is provided by the .HBLIN analysis, which has setup and analysis control options similar to .LIN, but is capable of extracting S-parameters about a large-signal periodic steady-state operating point.

In the case of circuits such as power amplifiers, the extraction of scattering parameters is also important, but the circuit stimulus and response signals may

themselves be large-signal periodic waveforms. And, it can be important to analyze how these S-parameter vary as a function of input power levels. This capability is provided by the .HBLSP Large-Signal S-parameter analysis, which uses large-signal stimulus signals for the S-parameter extractions.

Frequency Translation S-Parameter (HBLIN) Extraction

Frequency translation scattering parameter (S-parameter) extraction is used to describe N-port circuits that exhibit frequency translation effects, such as mixers. The analysis is similar to the existing LIN analysis, except that the circuit is first linearized about a periodically varying operating point instead of a simple DC operating point. After the linearization, the S-parameters between circuit ports that convert signals from one frequency band to another are calculated.

You use the `.HBLIN` statement to extract frequency translation S-parameters and noise figures.

Frequency translation S-parameter describes the capability of a periodically linear time varying systems to shift signals in frequency. The S-parameters for a frequency translation system are similar to the S-parameters of a linear-time-varying system, it is defined as:

$$b = S \cdot a$$

Equation 61

$$S_{\mathrm{i,j;m,n}}(\omega) = \frac{b_{\mathrm{i,m}}(\omega)}{a_{\mathrm{j,n}}(\omega)}$$

$$a_{\mathrm{k} \neq j,\ p \neq n}(\omega) = 0$$

The incident waves, $a_{i,n}(\omega)$, and reflected waves, $b_{i,n}(\omega)$, are defined by using these equations:

Equation 62

$$a_{i,n}(\omega) = \frac{V_i(\omega + n\omega_0) + Z_{0i} I_i(\omega + n\omega_0)}{2\sqrt{Z_{0i}}}$$

$$b_{i,n}(\omega) = \frac{V_i(\omega + n\omega_0) - Z_{0i} I_i(\omega + n\omega_0)}{2\sqrt{Z_{0i}}}$$

Where,

- ω_0 is the fundamental frequency (tone).
- n is a signed integer.
- i is the port number.
- $a_{i,n}(\omega)$ is the input wave at the frequency $\omega + n\omega_0$ on the i[th] port.
- $b_{i,n}(\omega)$ is the reflected wave at the frequency $\omega + n\omega_0$ on the i[th] port.
- $V_i(\omega + n\omega_0)$ is the Fourier coefficient at the frequency $\omega + n\omega_0$ of the voltage at port i.
- $I_i(\omega + \omega n_0)$ is the Fourier coefficient at the frequency $\omega + n\omega_0$ of the current at port i.
- Z_{0i} is the reference impedance at port i.
- V and I definitions are Fourier coefficients rather than phasors.

For a multi-tone analysis, it can be expressed as:

$$b = S \cdot a$$

Equation 63

$$S_{i,j;m_1 \ldots m_N, n_1, n_2 \ldots n_N}(\omega) = \frac{b_{i,m_1,m_2 \ldots m_N}(\omega)}{a_{j,n_1,n_2 \ldots n_N}(\omega)}$$

$$a_{k,p_1,p_2 \ldots p_N | k \neq j, \nabla p_q \neq n_q}(\omega) = 0$$

Equation 64

Where,

- ω_j is the i[th] tone.

The frequency translate S-parameters are calculated by applying different $n_j (j = 1 \sim N)$ to different ports.

Limitations

The HBLIN analysis has these known limitations:

$$a_{i,n_1,n_2...n_N}(\omega) = \frac{V_i\left(\omega + \sum_{j=1}^{N} n_j\omega_j\right) + Z_{0i}I_i\left(\omega + \sum_{j=1}^{N} n_j\omega_j\right)}{2\sqrt{Z_{oi}}}$$

$$b_{i,n_1,n_2...n_N}(\omega) = \frac{V_i\left(\omega + \sum_{j=1}^{N} n_j\omega_j\right) - Z_{0i}I_i\left(\omega + \sum_{j=1}^{N} n_j\omega_j\right)}{2\sqrt{Z_{oi}}}$$

- Noise parameters are not calculated for mixed-mode operation.
- Only the S-parameters corresponding to the set of frequencies specified at each port are extracted.
- Multiple small-signal tones are not supported.
- The port (P) element impedance cannot be specified as complex.

HB Analysis

An HB analysis is required prior to an HBLIN analysis. To extract the frequency translation S-parameters, a sweep of the small-signal tone is necessary. You can identify the small-signal tone sweep in the `.HBLIN` command or in the `.HB` command together with a `SS_TONE` specification.

For additional information regarding HB analysis, see Harmonic Balance Analysis on page 108.

Port Element

You must use a port (P) element as the termination at each port of the system. To indicate the frequency band that the S-parameters are extracted from, it is necessary to specify a harmonic index for each P-element.

Port Element Syntax

Without SS_TONE

```
Pxxx p n n_ref PORT=portnumber
+ [HBLIN = [H1, H2, ... HN, +/-1]] ...
```

With SS_TONE

```
Pxxx p n n_ref [PORT=portnumber]
+ [HBLIN = [H1, H2, ... +/-1 ... HN]] ...
```

Parameter	Description
n_ref	Reference node used when a mixed-mode port is specified.
PORT	The port number. Numbered sequentially beginning with 1 with no shared port numbers.
HBLIN	Integer vector that specifies the harmonic index corresponding to the tones defined in the .HB command. The +/-1 term corresponds to the small-signal tone specified by SS_TONE in the .HB command. If there is no SS_TONE in the .HB command, the +/-1 term must be at the last entry of HBLIN vector.

HBLIN Analysis

You use the .HBLIN statement to extract frequency translation S-parameters and noise figures.

Input Syntax

Without SS_TONE

```
.HBLIN frequency_sweep
+ [NOISECALC = [1|0|yes|no]] [FILENAME=file_name]
+ [DATAFORMAT = [ri|ma|db]]
+ [MIXEDMODE2PORT = [dd|cc|cd|dc|sd|sc|cs|ds]]
```

With SS_TONE

```
.HBLIN [NOISECALC = [1|0|yes|no]] [FILENAME=file_name]
+ [DATAFORMAT = [ri|ma|db]]
+ [MIXEDMODE2PORT = [dd|cc|cd|dc|sd|sc|cs|ds]]
```

Parameter	Description
frequency_sweep	Frequency sweep range for the input signal (also referred to as the input frequency band (IFB) or fin). You can specify LIN, DEC, OCT, POI, or SWEEPBLOCK. Specify the nsteps, start, and stop frequencies using the following syntax for each type of sweep: ■ LIN *nsteps start stop* ■ DEC *nsteps start stop* ■ OCT *nsteps start stop* ■ POI *nsteps freq_values* ■ SWEEPBLOCK *nsteps freq1 freq2 ... freqn* ■ DATA=*dataname*
NOISECALC	Enables calculating the noise figure. The default is no (0).
FILENAME	Specifies the output file name for the extracted S-parameters or the object name after the -o command-line option. The default is the netlist file name.
DATAFORMAT	Specifies the format of the output data file. ■ dataformat=RI, real-imaginary. ■ dataformat=MA, magnitude-phase. This is the default format for Touchstone files. ■ dataformat=DB, DB(magnitude)-phase.

Parameter	Description
MIXEDMODE2PORT	Describes the mixed-mode data map of output mixed mode S-parameter matrix. The availability and default value for this keyword depends on the first two port (P element) configuration as follows: ■ case 1: p1=p2=single-ended (standard-mode P element) available: ss default: ss ■ case 2: p1=p2=balanced (mixed-mode P element) available: dd, cd, dc, cc default: dd ■ case 3: p1=balanced p2=single-ended available: ds, cs default: ds ■ case 4: p1=single p2=balanced available: sd, sc default: sd

Example 1

Single-tone analysis with frequency translation. In this example, the 2-port S-parameters from RF (1G-del_f) to IF (del_f) are extracted. The LO signal is specified by normal voltage source Vlo. The frequency on port 1 is in the RF band, 1G-del_f, and the frequency on port 2 is in the IF band, del_f. The IF band is swept from 0- to 100-MHz. The results are output to file ex1.s2p.

```
p1 RFin gnd port=1 HBLIN=(1,-1)
p2 IFout gnd port=2 HBLIN=(0,1)
Vlo LOin gnd DC 0 HB 2.5 0 1 1
.HB tones=1G harms=5
.HBLIN lin 5 0 100meg noisecalc=no filename=ex1
+ dataformat=ma
```

Example 2

Another single-tone analysis with frequency translation example. In this example, the 3-port S-parameters are extracted. Port 3 provides the periodic large signal. The frequency on port 1 is del_f, the frequency on port 2 is 1G*2-del_f, and the frequency on port 3 is 1G*1+del_f. The small-signal frequency is swept from 0 to 100MHz. HBNOISE calculation is required. The results are output to file ex2.s3p.

```
p1 1 0 port=1 HBLIN=(0, 1)
p2 2 0 port=2 HBLIN=(2, -1)
p3 3 0 port=3 hb 0.5 0 1 1 HBLIN=(1, 1)
.HB tones=1G harms=5
.HBLIN lin 5 0 100meg noisecalc=yes filename=ex2
```

Output Syntax

This section describes the syntax for the `HBLIN .PRINT` and `.PROBE` statements.

.PRINT and .PROBE Statements

```
.PRINT HBLIN Smn | Smn(TYPE) | S(m,n) | S(m,n)(TYPE)
.PROBE HBLIN Smn | Smn(TYPE) | S(m, n) | S(m, n)(TYPE)
.PRINT HBLIN SXYmn | SXYmn(TYPE) | SXY(m,n) | SXY(m,n)(TYPE)
.PROBE HBLIN SXYmn | SXYmn(TYPE) | SXY(m, n) | SXY(m, n)(TYPE)
.PRINT HBLIN NF SSNF DSNF
.PROBE HBLIN NF SSNF SSNF
```

Parameter	Description
Smn | Smn(TYPE) | S(m,n) | S(m,n)(TYPE) SXYmn | SXYmn(TYPE) | SXY(m,n) | SXY(m,n)(TYPE)	Complex 2-port parameters. Where: ▪ m = 1 or 2 ▪ n = 1 or 2 ▪ X and Y are used for mixed-mode S-parameter output. The values for X and Y can be D (differential), C (common), or S (single-end). ▪ TYPE = R, I, M, P, PD, D, DB, or DBM R = real I = imaginary M = magnitude P = PD = phase in degrees D = DB = decibels DBM = decibels per 1.0e-3

Parameter	Description
NF SSNF	NF and SSNF both output a single-side band noise figure as a function of the IFB points: NF = SSNF = 10 Log(SSF) Single side-band noise factor, SSF = {(Total Noise at output, at OFB, originating from all frequencies) - (Load Noise originating from OFB)} / (Input Source Noise originating from IFB).
DSNF	DSNF outputs a double side-band noise figure as a function of the IFB points. DSNF = 10 Log(DSF) Double side-band noise factor, DSF = {(Total Noise at output, at the OFB, originating from all frequencies) - (Load Noise originating from the OFB)} / (Input Source Noise originating from the IFB and from the image of IFB).

Output Data Files

An HBLIN analysis produces these output data files:

- The S-parameters from the `.PRINT` statement are written to a *.printhl#* file.
- The extracted S-parameters from the `.PROBE` statement are written to a *.hl#* file.

Large-Signal S-parameter (HBLSP) Analysis

An HBLSP analysis provides three kinds of analyses for periodically-driven nonlinear circuits, such as those that employ power amplifiers and filters:

- Two-port power-dependant (large-signal) S-parameter extraction
- Two-port small-signal S-parameter extraction
- Two-port small-signal noise parameter calculation

Unlike small-signal S-parameters, which are based on linear analysis, power-dependent S-parameters are based on harmonic balance simulation. Its solution accounts for nonlinear effects such as compression and variation in power levels.

The definition for power-dependent S-parameters is similar to that for small-signal parameters. Power-dependent S-parameters are defined as the ratio of reflected and incident waves by using this equation:

b = S * a ; S[i, j]=b[i,n]/a[j,n] when a[k,n](k!=j)=0

The incident waves, a[i, n], and reflected waves, b[i, n], are defined by using these equations:

$a[i, n] = (V[i](n*W_0) + Z_o[i] * I[i](n*W_0)) / (2 * sqrt(Z_o[i]))$

$b[i, n] = (V[i](n*W_0) - Z_o[i] * I[i](n*W_0)) / (2 * sqrt(Z_o[i]))$

Where:

- W_0 is the fundamental frequency (tone).
- n is a signed integer.
- i is the port number.
- a[i, n] is the input wave at the frequency $n*W_0$ on the i^{th} port.
- b[i, n] is the reflected wave at the frequency $n*W_0$ on the i^{th} port.
- $V[i](n*W_0)$ is the Fourier coefficient at the frequency $n*W_0$ of the voltage at port i.
- $I[i](n*W_0)$ is the Fourier coefficient at the frequency $n*W_0$ of the current at port i.
- $Z_o[i]$ is the reference impedance at port i.

An HBLSP analysis only extracts the S-parameters on the first harmonic (that is, n=1).

Limitations

The HBLSP analysis has these known limitations:

- Power-dependent S-parameter extraction is a 2-port analysis only. Multiport power-dependent S-parameters are not currently supported.
- The intermodulation data block (`IMTDATA`) in the *.p2d#* file is not supported.
- The internal impedance of the P (port) Element can only be a real value. Complex impedance values are not supported.

Input Syntax

```
.HBLSP NHARMS=nh [POWERUNIT=[dbm|watt]]
+ [SSPCALC=[1|0|YES|NO]] [NOISECALC=[1|0|YES|NO]]
+ [FILENAME=file_name] [DATAFORMAT=[ri|ma|db]]
+ FREQSWEEP freq_sweep POWERSWEEP power_sweep
```

Parameter	Description
NHARMS	Number of harmonics in the HB analysis triggered by the `.HBLSP` statement.
POWERUNIT	Power unit. Default is watt.
SSPCALC	Extract small-signal S-parameters. Default is 0 (NO).
NOISECALC	Perform small-signal 2-port noise analysis. Default is 0 (NO).
FILENAME	Output data .p2d# filename. Default is the netlist name or the object name after the -o command-line option.
DATAFORMAT	Format of the output data file. Default is ma (magnitude, angle).
FREQSWEEP	Frequency sweep specification. A sweep of type LIN, DEC, OCT, POI, or SWEEPBLOCK. Specify the *nsteps*, *start*, and *stop* times using the following syntax for each type of sweep: ▪ LIN *nsteps start stop* ▪ DEC *nsteps start stop* ▪ OCT *nsteps start stop* ▪ POI *nsteps freq_values* ▪ SWEEPBLOCK=*blockname* This keyword must appear before the POWERSWEEP keyword.

Parameter	Description
POWERSWEEP	Power sweep specification. A sweep of type LIN, DEC, OCT,POI, or SWEEPBLOCK. Specify the nsteps, start, and stop frequencies using the following syntax for each type of sweep: ▪ LIN *nsteps start stop* ▪ DEC *nsteps start stop* ▪ OCT *nsteps start stop* ▪ POI *nsteps power_values* ▪ SWEEPBLOCK=*blockname* This keyword must follow the FREQSWEEP keyword.

Note: The `FREQSWEEP` and `POWERSWEEP` keywords must appear at the end of an `.HBLSP` statement.

Examples

Example 1 does 2-port single-tone, power-dependent S-parameter extraction, without frequency translation:

- Frequency sweep: The fundamental tone is swept from 0 to 1G
- Power sweep: The power input at port 1 is swept from 6 to 10 Watts.
- Five harmonics are required for the HB analysis. Large-signal S-parameters are extracted on the first harmonic.
- Five harmonics are required in the HBLSP triggered HB analysis.
- The DC value in p1 statement is used to set DC bias, which is used to perform small-signal analyses.
- Small-signal S-parameters are required extracted.
- Small-signal two-port noise analysis is required.
- The data will be output to the ex1.p2d file.

Example 1 2-Port, Single Tone

```
p1 1 0 port=1 dc=1v
p2 2 0 port=2
.hblsp nharms=5 powerunit = watt
+ sspcalc=1 noisecalc=1 filename=ex1
+ freqsweep lin 5 0 1G powersweep lin 5 6 10
```

Example 2 generates large scale S-parameters as a function of input for a differential equalizer.

Example 2 4-Port Network

```
* hblsp example
.opt post
p1 n1 0 port=1 ac=1
p2 n2 0 port=2
*** put your DUT
R1 n1 n2 10***
.hblsp nharms=5
+ freqsweep lin 4 1k 10k
+ powersweep lin 2 5 10
.end
```

Output Syntax

This section describes the syntax for the `HBLSP .PRINT` and `.PROBE` statements. These statements only support S and noise parameter outputs. Node voltage, branch current, and all other parameters are not supported in `HBLSP .PRINT` and `.PROBE` statements.

.PRINT and .PROBE Statements

```
.PRINT HBLSP Smn | Smn(TYPE) | S(m, n) | S(m, n)(TYPE)
+ ...small signal 2-port noise params...
.PROBE HBLSP Smn | Smn(TYPE) | S(m, n) | S(m, n)(TYPE)
+ ...small signal 2-port noise params...
```

Parameter	Description
Smn | Smn(TYPE) | S(m,n) | S(m,n)(TYPE)	Complex 2-port parameters. Where: ▪ m = 1 or 2 ▪ n = 1 or 2 ▪ TYPE = R, I, M, P, PD, D, DB, or DBM R = real I = imaginary M = magnitude P = PD = phase in degrees D = DB = decibels DBM = decibels per 1.0e-3

Parameter	Description
... small signal 2-port noise parameters ...	G_AS | NF | RN | YOPT | GAMMA_OPT | NFMIN | VN2 | ZCOR | GN | RHON | YCOR | ZOPT | IN2 For a description of these parameters, see Linear Network Parameter Analysis in the *HSPICE User Guide: Simulation and Analysis*.

Output Data Files

An HBLSP analysis produces these output data files:

- The large-signal S-parameters from the `.PRINT` statement are written to a *.printls#* file.
- The small-signal S-parameters from the `.PRINT` statement are written to a *.printss#* file.
- The large-signal S-parameters from the `.PROBE` statement are written to a *.ls#* file.
- The small-signal S-parameters from the `.PROBE` statement are written to a .ss# file.
- The extracted large- and small-signal S and noise parameters are written to a .p2d# file.

The large- and small-signal S-parameters from the `.PROBE` statement are viewable in Custom WaveView.

11

Envelope Analysis

Describes how to use envelope simulation.

These topics are covered in the following sections:

- Envelope Simulation
- Envelope Analysis Commands
- Nonautonomous Form
- Oscillator Analysis Form
- Fast Fourier Transform Form
- Output Syntax

Envelope Simulation

Envelope simulation combines features of time- and frequency-domain analysis. Harmonic Balance (HB) solves for a static set of phasors for all the circuit state variables, as shown in this equation:

Equation 65

$$v(t) = a_0 + \sum_{i=1}^{N} [a_i \cos \omega_i t + b_i \sin \omega_i t]$$

In contrast, envelope analysis finds a dynamic, time-dependent set of phasors, as this equation shows:

Equation 66
$$v(t) = a_0(\hat{t}) + \sum_{i=1}^{N} [a_i(\hat{t})\cos\omega_i t + b_i(\hat{t})\sin\omega_i t]$$

Thus, in envelope simulation, each signal is described by the evolving spectrum. Envelope analysis is generally used on circuits excited by signals with significantly different timescales. An HB simulation is performed at each point in time of the slower-moving ($\hat{t}$) timescale. In this way, for example, a 2-tone HB simulation can be converted into a series of related 1-tone simulations where the transient analysis proceeds on the ($\hat{t}$) timescale, and 1-tone HB simulations are performed with the higher frequency tone as the fundamental frequency.

In HSPICE RF, any voltage or current source identified as a HB source either in a V or I element statement, or by an `.OPTION TRANFORHB` command, is used for HB simulations at each point in $\hat{t}$ time. All other sources are associated with the transient timescale. Also, the input waveforms can be represented in the frequency domain as RF carriers modulated by an envelope by identifying a VMRF signal source in a V or I element statement. The amplitude and phase values of the sampled envelope are used as the input signal for HB analysis.

Some typical applications for envelope simulation are amplifier spectral regrowth, adjacent channel power ration (ACPR), and oscillator startup and shutdown analyses.

Envelope Analysis Commands

This section describes those commands specific to envelope analysis. These commands are:

- Standard envelope simulation (`.ENV`)
- Oscillator simulation, both startup and shutdown (`.ENVOSC`)
- Envelope Fast Fourier Transform (`.ENVFFT`)

Nonautonomous Form

`.ENV TONES=`*`f1`* `[`*`f2...fn`*`] NHARMS=`*`h1`* `[`*`h2...hn`*`]`

```
+ ENV_STEP=tstep ENV_STOP=tstop
```

Parameter	Description
TONES	Carrier frequencies, in hertz.
NHARMS	Number of harmonics.
ENV_STEP	Envelope step size, in seconds.
ENV_STOP	Envelope stop time, in seconds.

Description

You use the `.ENV` command to do standard envelope simulation. The simulation proceeds just as it does in standard transient simulation, starting at `time=0` and continuing until `time=env_stop`. An HB analysis is performed at each step in time. You can use Backward-Euler (BE), trapezoidal (TRAP), or level-2 Gear (GEAR) integration.

Recommended option settings are:

- For BE integration, set `.OPTION SIM_ORDER=1`.
- For TRAP, set `.OPTION SIM_ORDER=2` (default) `METHOD=TRAP` (default).
- For GEAR, set `.OPTION SIM_ORDER=2` (default) `METHOD=GEAR`.

Example

```
.env tones=1e9 nharms=6 env_step=10n env_stop=1u
```

Oscillator Analysis Form

```
.ENVOSC TONE=f1 NHARMS=h1 ENV_STEP=tstep ENV_STOP=tstop
+ PROBENODE=n1,n2,vosc <FSPTS=num, min, max>
```

Parameter	Description
TONE	Carrier frequencies, in hertz.
NHARMS	Number of harmonics.
ENV_STEP	Envelope step size, in seconds.
ENV_STOP	Envelope stop time, in seconds.

Parameter	Description
PROBENODE	Defines the nodes used for oscillator conditions and the initial probe voltage value.
FSPTS	Specifies the frequency search points used in the initial small-signal frequency search. Usage depends on oscillator type.

Description

You use the .ENVOSC command to do envelope simulation for oscillator startup or shutdown.

Oscillator startup or shutdown analysis with this command must be helped along by converting a bias source from a DC description to a PWL description that either:

- Starts at a low value that supports oscillation and ramps up to a final value (startup simulation)
- Starts at the DC value and ramps down to zero (shutdown simulation).

In addition to solving for the state variables at each envelope time point, the .ENVOSC command also solves for the frequency. This command is intended to be applied to high-Q oscillators that take a long time to reach steady-state. For these circuits, standard transient analysis is too costly. Low-Q oscillators, such as typical ring oscillators, are more efficiently simulated with standard transient analysis.

Example

```
.envosc tone=250Meg nharms=10 env_step=20n env_stop=10u
+ probenode=v5,0,1.25
```

Fast Fourier Transform Form

```
.ENVFFT output_var [NP=val] [FORMAT=keyword]
+[<WINDOW=keyword] [ALFA=val]
```

Parameter	Description
output_var	Any valid output variable.
NP	The number of points to use in the FFT analysis. NP must be a power of 2. If not a power of 2, then it is automatically adjusted to the closest higher number that is a power of 2. The default is 1024.

Parameter	Description
FORMAT	Specifies the output format: NORM= normalized magnitude UNORM=unnormalized magnitude (default)
WINDOW	Specifies the window type to use: RECT=simple rectangular truncation window (default) BART=Bartlett (triangular) window HANN=Hanning window HAMM=Hamming window BLACK=Blackman window HARRIS=Blackman-Harris window GAUSS=Gaussian window KAISER=Kaiser-Bessel window
ALFA	Controls the highest side-lobe level and bandwidth for GAUSS and KAISER windows. The default is 3.0.

Description

You use the `.ENVFFT` command to perform Fast fourier Transform (FFT) on envelope output. This command is similar to the `.FFT` command. The only difference is that transformation is performed on real data with the `.FFT` command, and with the `.ENVFFT` command, the data being transformed is complex. You usually want to do this for a specific harmonic of a voltage, current, or power signal.

Example

```
.envfft v(out)[1]
```

Output Syntax

The results from envelope simulation can be made available through the `.PRINT`, `.PROBE`, and `.MEASURE` commands. This section describes the basic syntax you can use for this purpose.

.PRINT or .PROBE

You can print or probe envelope simulation results by using the following commands:

```
.PRINT ENV ov1 <ov2... >
.PROBE ENV ov1 <ov2... >
```

Where `ov1...` are the output variables to print or probe.

.MEASURE

In HSPICE RF, the independent variable for envelope simulation is the first tone. Otherwise and except for the analysis type, the `.MEASURE` statement syntax is the same as the syntax for HB; for example,

```
.MEASURE ENV result ...
```

Envelope Output Data File Format

The results of envelope simulations are written to *.ev# data files by the `.PROBE` statement. The format of an *.ev# data file is equivalent to an *.hb# data file with the addition of one fundamental parameter sweep that represents the slowly-varying time-envelope variation $\hat{t}$ of the Fourier coefficients and frequencies. You can recognize this swept parameter" in the *.ev# file by the keyword `env_time`.

Each row in the tabulated data of an *.ev# file includes values for identifying frequency information, the complex data for the output variables, and information on the envelope time sweep. For example, the header for a data file dump for output variables v(in) and v(out) that follow a 2-tone envelope analysis, have entries for:

```
hertz  v(in)  v(out)  n0  f0  n1  f1  sweep  env_time  $&%#
```

Which result in data blocks with floating point values following:

```
env_time[0]
f[0]  a[0]{v(in)}  b[0] {v(in)}  a[0] {v(out)}  b[0] {v(out)}  n0
f0  n1  f1
f[1]  a[1]{v(in)}  b[1] {v(in)}  a[1] {v(out)}  b[1] {v(out)}  n0
f0  n1  f1
...
f[N]  a[N]{v(in)}  b[N] {v(in)}  a[N] {v(out)}  b[N] {v(out)}  n0
f0  n1  f1

env_time[1]
f[0]  a[0]{v(in)}  b[0] {v(in)}  a[0] {v(out)}  b[0] {v(out)}  n0
f0  n1  f1
f[1]  a[1]{v(in)}  b[1] {v(in)}  a[1] {v(out)}  b[1] {v(out)}  n0
f0  n1  f1
...
f[N]  a[N]{v(in)}  b[N] {v(in)}  a[N] {v(out)}  b[N] {v(out)}  n0
f0  n1  f1

...

env_time[M-1]
f[0]  a[0]{v(in)}  b[0] {v(in)}  a[0] {v(out)}  b[0] {v(out)}  n0
f0  n1  f1
f[1]  a[1]{v(in)}  b[1] {v(in)}  a[1] {v(out)}  b[1] {v(out)}  n0
f0  n1  f1
...
f[N]  a[N]{v(in)}  b[N] {v(in)}  a[N] {v(out)}  b[N] {v(out)}  n0
f0  n1  f1
```

Where there are M data blocks corresponding to M envelope time points, with each block containing N+1 rows for the frequency data. The units for the `env_time` sweep are seconds.

12

Post-Layout Analysis

Describes the post-layout analysis flow, including post-layout back-annotation, DSPF and SPEF files, linear acceleration, check statements, and power analysis.

These topics are covered in the following sections:

- Post-Layout Back-Annotation
- Linear Acceleration Control Options Summary

Post-Layout Back-Annotation

A traditional, straightforward, “brute-force” flow runs an RC extraction tool that produces a detailed standard parasitic format (DSPF) file. DSPF is the standard format for transferring RC parasitic information. This traditional flow then feeds this DSPF file into the circuit simulation tool for post-layout simulation.

A key problem is that the DSPF file is flat. Accurately simulating a complete design, such as an SRAM or an on-chip cache, is a waste of workstation memory, disc space usage, and simulation runtime. Because this DSPF file is flat, control and analysis are limited.

- How do you set different options for different blocks for better trade-off between speed and accuracy?
- How do you perform a power analysis on a flat netlist to check the power consumption?

- This traditional flow flattens all nodes after extraction so it is more difficult to compare the delay before and after extraction.
- This traditional flow can also stress the limits of an extraction tool so reliability also becomes an issue.

HSPICE RF provides a flow that solves all of these problems.

- Star-RCXT generates a hierarchical Layout Versus Schematic (LVS) ideal netlist, and flat information about RC parasitics in a DSPF or (standard parasitic exchange format (SPEF) file.
- HSPICE RF uses the hybrid flat-hierarchical approach to back-annotate the RC parasitics, from the DSPF or SPEF file, into the hierarchical LVS ideal netlist.

Using the hierarchical LVS ideal netlist cuts simulation runtime and CPU memory usage. Because HSPICE RF uses the hierarchical LVS ideal netlist as the top-level netlist, you can fully control the netlist. For example:

- You can set different modes to different blocks for better accuracy and speed trade-off.
- You can run power analysis, based on the hierarchical LVS ideal netlist, to determine the power consumption of each block. If you use the hierarchical LVS ideal netlist, you can reuse all post-processing statements from the pre-layout simulation for the post-layout simulation. This saves time, and the capacity of the verification tool is not stressed so reliability is higher.

HSPICE RF supports only the XREF:COMPLETE flow and the XREF:NO flow from Star-RCXT. Refer to the *Star-RCXT User Guide* for more information about the XREF flow.

To generate a hierarchical LVS ideal netlist with Star-RCXT, include the following options in the Star-RCXT command file.

```
*** for XREF:NO flow ***
NETLIST_IDEAL_SPICE_FILE: ideal_spice_netlist.sp
NETLIST_IDEAL_SPICE_TYPE: layout
NETLIST_IDEAL_SPICE_HIER:YES

*** for XREF:COMPLETE flow ***
NETLIST_IDEAL_SPICE_FILE: ideal_spice_netlist.sp
NETLIST_IDEAL_SPICE_TYPE: schematic
NETLIST_IDEAL_SPICE_HIER:YES
```

Note: Before version 2002.2, Star-RCXT used `NETLIST_IDEAL_SPICE_SKIP_CELLS` to generate the hierarchical ideal SPICE netlist. HSPICE RF can still simulate post-layout designs using the brute-force flow, but the post-layout flow is preferable in HSPICE RF.

HSPICE RF supports these post-layout flows to address your post-layout simulation needs.

- Standard Post-Layout Flow
- Selective Post-Layout Flow
- Additional Post-Layout Options

Standard Post-Layout Flow

Use this flow mainly for analog or mixed signal design, and high-coverage verification runs when you need to back-annotate RC parasitics into the hierarchical LVS ideal netlist. In this flow, HSPICE RF expands all nets from the DSPF or SPEF file. To expand only selected nets, use see Selective Post-Layout Flow on page 283.

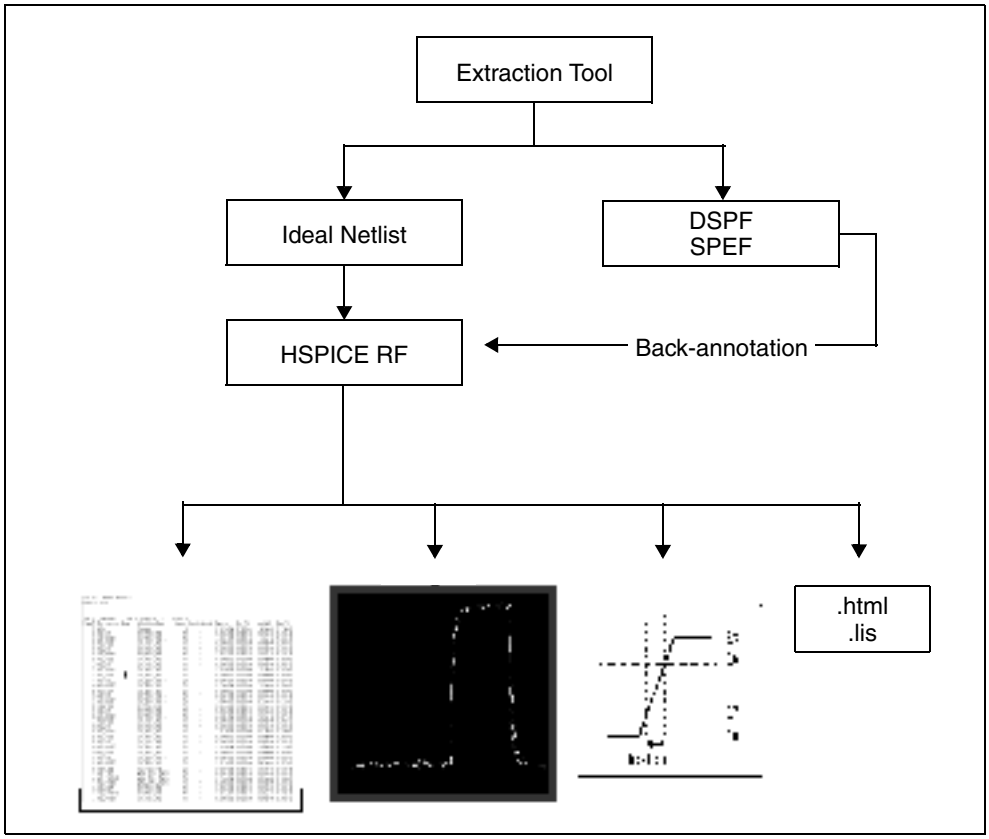


Figure 23 Standard Post-Layout Flow

Standard Post-Layout Flow Control Options

The standard post-layout flow options are `SIM_DSPF` and `SIM_SPEF`. Include one of these options in your netlist. For example,

```
.OPTION SIM_DSPF="[scope] dspf_filename"
.OPTION SIM_SPEF="spec_filename"
```

In the `SIM_DSPF` syntax, `scope` can be a subcircuit definition or an instance. If you do not specify `scope`, it defaults to the top-level definition. HSPICE RF requires both a DSPF file and an ideal netlist. Only flat DSPF files are supported; hierarchy statements, such as `.SUBCKT` and .x1, are ignored.

Very large circuits generate very large DSPF files; this is when using either the `SIM_DSPF` or the `SIM_DSPF_ACTIVE` option can really improve performance.

You can specify a DSPF file in the `SIM_SPEF` option, or a SPEF file in the `SIM_DSPF` option. The `scope` function is not supported in the SPEF format.

For descriptions and usage examples, see .OPTION SIM_DSPF and .OPTION SIM_SPEF in the *HSPICE Reference Manual: Commands and Control Options*.

Example

```
$  models
.MODEL p pmos
.MODEL n nmos
.INCLUDE add4.dspf
.OPTION SIM_DSPF="add4.dspf"
.VEC "dspf_adder.vec"
.TRAN 1n 5u
vdd vdd 0 3.3
.OPTION POST
.END
```

SIM_DSPF With SIM_LA Option

The SIM_DSPF option accelerates the simulation by more than 100%. By using the SIM_LA option at the same time, you can further reduce the total CPU time:

```
$  models
.MODEL p pmos
.MODEL n nmos
.INCLUDE add4.dspf
.OPTION SIM_DSPF="add4.dspf"
.OPTION SIM_LA=PACT
.VEC "dspf_adder.vec"
.TRAN 1n 5u
vdd vdd 0 3.3
.OPTION POST
.END
```

To expand only active nodes, such as those that move, include the SIM_DSPF_ACTIVE option in your netlist. For example:

```
.OPTION SIM_DSPF_ACTIVE="active_net_filename"
```

This option is most effective when used with a large design—for example, over 5K transistors. Smaller designs lose some of the performance gain, due to internal overhead processing.

For syntax and description of SIM_DSPF_ACTIVE option, see .OPTION SIM_DSPF_ACTIVE in the *HSPICE Reference Manual: Commands and Control Options*.

When you have included the appropriate control option, run HSPICE RF, using the ideal netlist.

The structure of a DSPF file is:

```
*|DSPF 1.0
*|DESIGN "demo"
*|Date "October 6, 1998"
...
.SUBCKT < name > < pins >
* Net Section
C1 ...
R1 ...
...
* Instance Section
...
.ENDS
```

Selective Post-Layout Flow

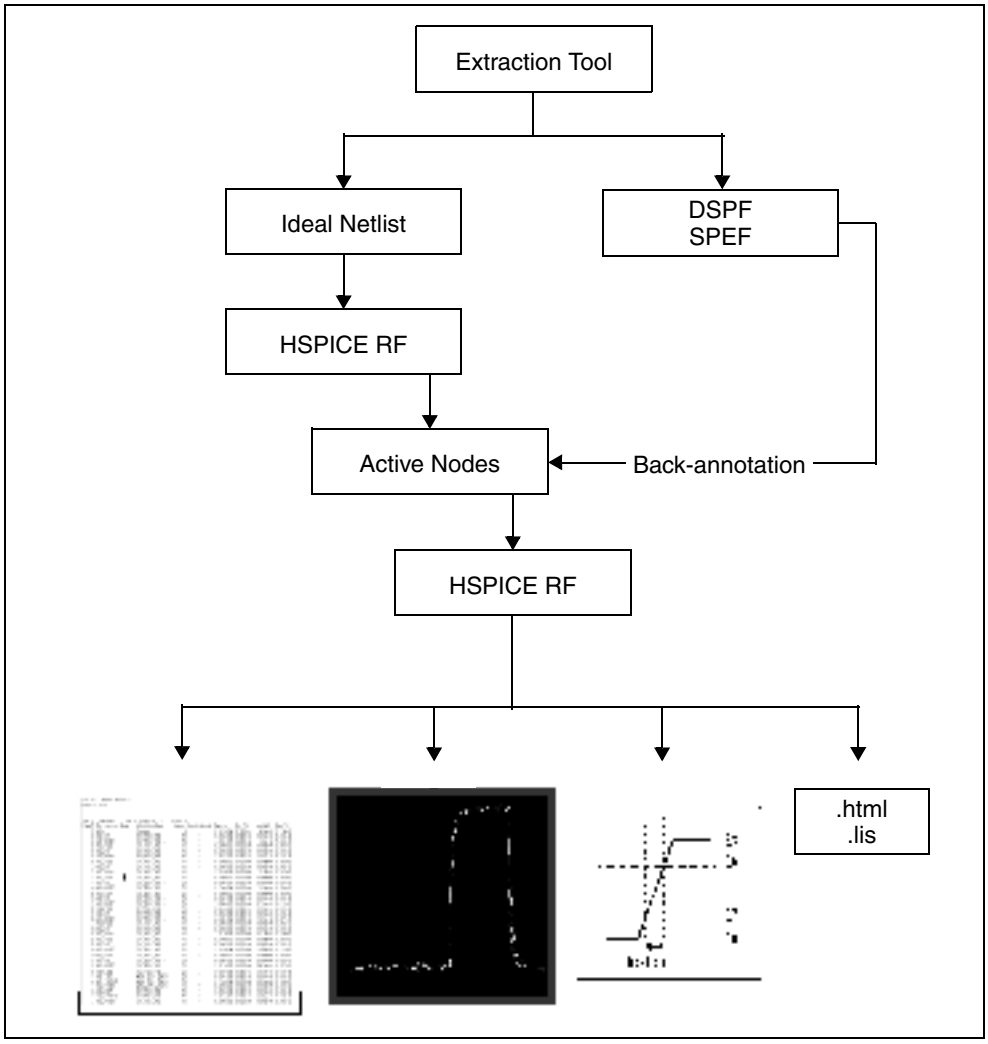


Figure 24 Selective Post-Layout Flow

You can use the selective post-layout flow to simulate a post-layout design for a memory or digital circuit, and for a corner-point verification run. Instead of back-annotating all RC parasitics into the ideal netlist, the selective post-layout flow automatically detects and back-annotates only active parasitics, into the hierarchical LVS ideal netlist. For a high-latency design, the selective post-layout flow is an order of magnitude faster than the standard post-layout flow.

Note: The selective post-layout flow applies only to RF transient analyses and cannot be used with other analyses such as DC, AC, or HB.

Selective Post-Layout Flow Control Options

To invoke the selective post-layout flow, include one of the options listed in Table 13 in your netlist.

Table 13 Selective Post-Layout Flow Options

Syntax	Description
SIM_DSPF_ACTIVE -or- SIM_SPEF_ACTIVE	HSPICE RF performs a preliminary verification run to determine the activity of the nodes and generates two ASCII files: active_node.rc and active_node.rcxt. These files save all active node information in both Star-RC format and Star-RCXT format. By default, a node is considered active if the voltage varies by more than 0.1V. To change this value, use the SIM_DSPF_VTOL or SIM_SPEF_VTOL option. For descriptions and usage examples, see OPTION SIM_DSPF_ACTIVE and .OPTION SIM_SPEF_ACTIVE in the *HSPICE Reference Manual: Commands and Control Options*.

Table 13 Selective Post-Layout Flow Options (Continued)

Syntax	Description
SIM_DSPF_VTOL -or- SIM_SPEF_VTOL	HSPICE RF performs a second simulation run by using the *active_node* file, the DSPF or SPEF file, and the hierarchical LVS ideal netlist to back-annotate only active portions of the circuit. If a net is latent, then HSPICE RF does not expand the net. This saves simulation runtime and memory. ▪ *value* is the tolerance of the voltage change. ▪ *scopen* can be a subcircuit definition (which has an @ prefix), or a subcircuit instance. By default, HSPICE RF performs only one iteration of the second simulation run. Use the SIM_DSPF_MAX_ITER or SIM_SPEF_MAX_ITER option to change it. For descriptions and usage examples, see .OPTION SIM_DSPF_VTOL and .OPTION SIM_SPEF_VTOL in the *HSPICE Reference Manual: Commands and Control Options*.
SIM_DSPF_MAX_ITER -or- SIM_SPEF_MAX_ITER	value is the maximum number of iterations for the second simulation run. Some of the latent nets might turn active after the first iteration of the second run. In this case: ▪ Resimulate the netlist to ensure the accuracy of the post-layout simulation. ▪ Use SIM_DSPF_MAX_ITER or SIM_SPEF_MAX_ITER to set the maximum number of iterations for the second run. If the active_node remains the same after the second simulation run, HSPICE RF ignores these options. For descriptions and usage examples, see .OPTION SIM_DSPF_MAX_ITER and .OPTION SIM_SPEF_MAX_ITER in the *HSPICE Reference Manual: Commands and Control Options*.

Additional Post-Layout Options

Other post-layout options are listed in Table 14.

Table 14 Additional Post-Layout Options

Syntax	Description
SIM_DSPF_RAIL -or- SIM_SPEF_RAIL	By default, HSPICE RF does not back-annotate parasitics of the power-net. To back-annotate power-net parasitics, include one of these options in the netlist. Default=OFF. ON expands nets in a power rail as it expands all nets.
SIM_DSPF_SCALER SIM_SPEF_SCALER -or- SIM_DSPF_SCALEC SIM_SPEF_SCALEC	Scales the resistance or capacitance values. ■ scaleR is the scale factor for resistance ■ scaleC is the scale factor for capacitance.
SIM_DSPF_LUMPCAPS -or- SIM_SPEF_LUMPCAPS	If HSPICE RF cannot back-annotate an instance in a net because one or more instances are missing in the hierarchical LVS ideal netlist, then by default HSPICE RF does not evaluate the net. Instead of ignoring all parasitic information for this net, HSPICE RF includes these options to connect a lumped capacitor with a value equal to the net capacitance to this net. Default = ON adds lumped capacitance; ignores other net contents.
SIM_DSPF_INSERROR -or- SIM_SPEF_INSERROR	HSPICE RF supports options to skip the unmatched instance, and continue the evaluation of the next instance. The default is OFF. ON skips unmatched instances and continues the evaluation.

Table 14 Additional Post-Layout Options (Continued)

Syntax	Description
SIM_SPEF_PARVALUE	This option affects only values in a SPEF file that have triplet format: *float:float:float*, which this option interprets as *best:average:worst*. In such cases: ▪ If SIM_SPEF_PARVALUE=1, HSPICE RF uses best. ▪ If SIM_SPEF_PARVALUE=2 (default), HSPICE RF uses average. ▪ If SIM_SPEF_PARVALUE=3, HSPICE RF uses worst.

Unsupported SPEF Options

HSPICE RF does not yet support the following IEEE-481 SPEF options:

- Hierarchical SPEF definition (multiple SPEF files connected with a hierarchical definition):
- `*DEFINE` and `*PDEFINE`
- `*R_NET` and `*R_PNET` definition
- `*D_PNET` definition.

Selective Extraction Flow

Use the selective extraction flow if disk space is limited. Especially use this option when simulating a full-chip post-layout design, where block latency is high. HSPICE RF feedbacks the active net information to Star-RCXT to extract only the active parasitic.

The major advantage of this flow is a smaller DSPF or SPEF file, which saves disk space.

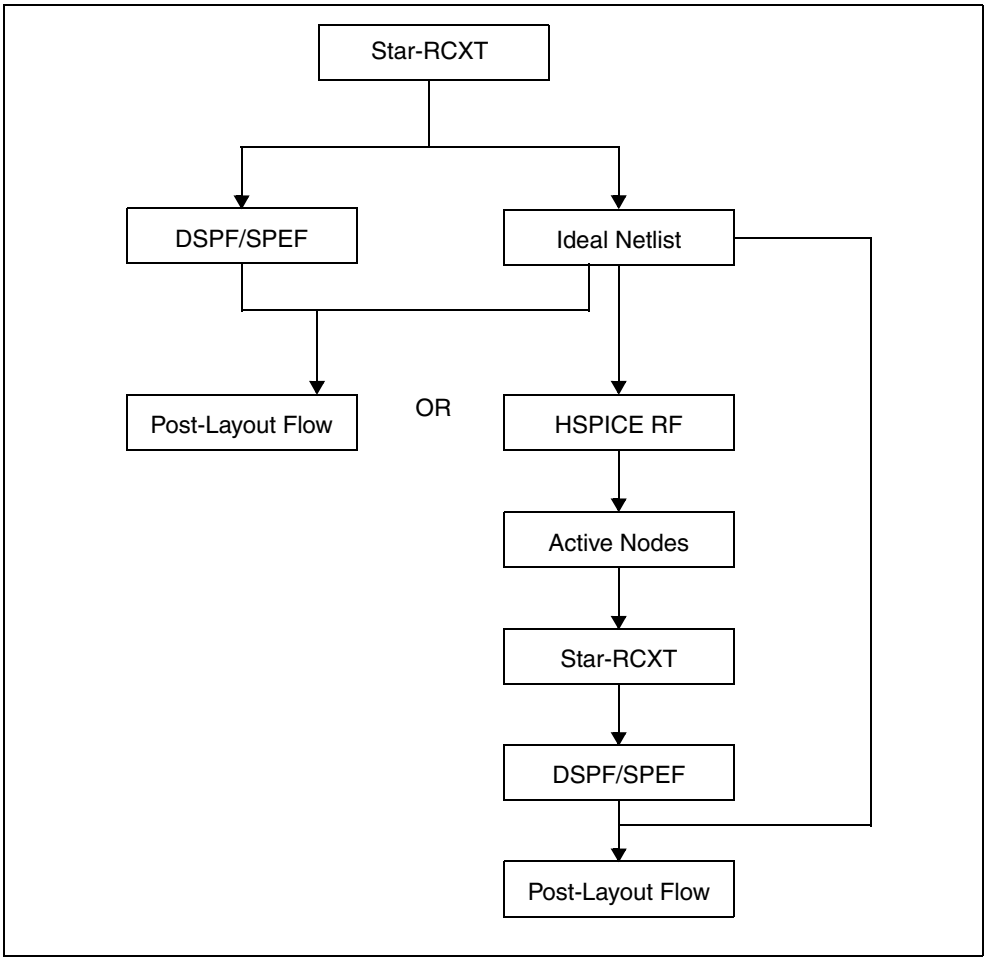


Figure 25 Selective Extraction Flow

Note: HSPICE RF generates an active node file in both Star-RC and Star-RCXT format. It then expands the active node file to the Star-RCXT command file to extract only active parasitics.

Overview of DSPF Files

In general, an SPF (Standard Parasitic Format) file describes interconnect delay and loading, due to parasitic resistance and capacitance. DSPF (Detailed Standard Parasitic Format) is a specific type of SPF file that describes the actual parasitic resistance and capacitance components of a net. DSPF is a

standard output format commonly used in many parasitic extraction tools, including Star-RCXT. The HSPICE RF circuit simulator can read DSPF files.

DSPF File Structure

The DSPF standard is published by Open Verilog International (OVI). For information about how to obtain the complete DSPF specification, or any other documents from OVI, see:

http://www.ovi.org/document.html

The OVI DSPF specification requires the following file structure in a DSPF file. Parameters in {braces} are optional:

```
DSPF_file : :=

*|DSPF{version}
{*|DESIGN design_name}
{*|DATE date}
{*|VENDOR vendor}
{*|PROGRAM program_name}
{*|VERSION program_version}
{*|DIVIDER divider}
{*|DELIMITER delimiter}

.SUBCKT
   *|GROUND_NET
      {path divider} net_name
   *|NET {path divider} net_name ||
         {path divider} instance_name ||
         pin_name
      net_capacitance

      *|P (pin_name pin_type
         pinCap
            {resistance {unit} {O}
            capacitance {unit} {F}}
         {x_coordinate y_coordinate})

      ||

      *|I {path divider} instance_name
            delimiter pin_name
         {path divider} instance_name
         pin_name pin_type
         pinCap
            {resistance {unit} {O}
            capacitance {unit}{F}}
         {x_coordinate y_coordinate}

      *|S ({path divider} net_name ||
         {path divider} instance_name
            delimiter pin_name ||
         pin_name
         instance_number
         {x_coordinate y_coordinate})
      capacitor_statements
      resistor_statements
   subcircuit_call_statements
.ENDS
```

{ .END}

Table 15 DSPF Parameters

Parameter	Definition
*|DSPF	Specifies that the file is in DSPF format.
{version}	Version number of the DSPF specification (optional).
*|	Words that start with *| are keywords.
||	Or (use the option either preceding or following ||). For example, *|P || *I means you can use either the *|P option or the *|I option.
design_name	Name of your circuit design (optional).
date	Date and time when a parasitic extraction tool (such as Star-RCXT) generated the DSPF file (optional).
vendor	Name of the vendor (such as Synopsys) whose tools you used to generate the DSPF file (optional).
program_name	Name of the program (such as Star-RCXT) that generated the DSPF file (optional).
program_version	Version number of the program that generated the DSPF file (optional).
divider	Character that divides levels of hierarchy in a circuit path (optional). If you do not define this parameter, the default hierarchy divider is a slash (/). For example, X1/X2 indicates that X2 is a subcircuit of the X1 circuit.
delimiter	Character used to separate the name of an instance and a pin in a concatenated instance pin name, or a net name and a sub-node number in a concatenated sub-node name. If you do not define this parameter, the default delimiter is a colon (:).
path	Hierarchical path to a net, instance, or pin, within a circuit.
net_name	Name of a net in a circuit or subcircuit.
instance_name	Name of an instance of a subcircuit.

Table 15 DSPF Parameters (Continued)

Parameter	Definition
pin_name	Name of a pin on an instance of a subcircuit.
pinCap	Capacitance of a pin.
pin_type	■ I (input) ■ O (output) ■ B (bidirectional) ■ X (don't care) ■ S (switch) ■ J (jumper)
resistance	Resistance on a pin in ohms for input (I), output (O), or bidirectional (B) pins. You can use resistance-capacitance (RC) pairs to model pin characteristics by using a higher-order equivalent RC ladder circuit than a single capacitor model. For example: C0 {R1 C1 R2 C2...}. Attaching RC pairs increases the order of the equivalent circuit from the first (C0) order. For X, S, and J pin types, simulation ignores this generalized capacitance value, but you should insert a 0 value as a place-holder for format integrity. The resistance value can be a real number or an exponent (optionally followed by a real number). You can enter an O (ohms) after the value.
capacitance	Capacitance on a pin in farads for input (I), output (O), or bidirectional (B) pins. Use as part of a resistance-capacitance (RC) pair. Optionally enter an F (farads) after the value.
unit	■ K (kilo) ■ M (milli) ■ U (micro) ■ N (nano) ■ P (pico) ■ F (femto)
x_coordinate	Location of a pin relative to the x (horizontal) axis.
y_coordinate	Location of a pin relative to the y (vertical) axis.

Table 15 DSPF Parameters (Continued)

Parameter	Definition
capacitor_ statements	SPICE-type statements that define capacitors in the subcircuit.
resistor_ statements	SPICE-type statements that define resistors in the subcircuit.
subcircuit_call_ statements	Statements that call the subcircuit from higher-level circuits.
.END	Marks the end of the file (optional).

DSPF File Example

```
*|DSPF 1.0
*|DESIGN "my_circuit"
*|DATE June 15, 2002 14:12:43
*|VENDOR "Synopsys"
*|PROGRAM "Star-RC"
*|VERSION "Star-RCXT 2002.2"
*|DIVIDER /
*|DELIMITER :
.SUBCKT BUFFER OUT IN
* Description of Nets
*GROUND_NET VSS
*|NET IN 1.221451PF
*|P(IN 1 0.0 0 10)
*|I(DF1:A DF1 A I 0.0PF 10.0 10.0)
*|I(DF1:B DF1 B I 0.0PF 10 0 20.0)
*|S(IN:1 5.0 10.0)(IN:2 5.0 20.0)
   C1 IN VSS 0.117763PF
   C2 IN:1 VSS 0.276325PF
   C3 IN:2 VSS 0.286325PF
   C4 DF1:A VSS 0.270519PF
   C5 DF1:B VSS 0.270519PF
   R20 IN N:1 1.70333E00
   R21 IN:1 DF1:A 1.29167E-01
   R22 IN:1 IN:2 1.29167E-01
   R23 IN:2 DF1:B 1.70333E-01
*|NET BF 0.287069PF
*|I(DF1:C DF1 C O 0.0PF 12.0 15.0)
*|I(INV1:IN INV1 IN I 0.0PF 30.0 15.0)
   C6 DF1:C VSS 0.208719PF
   C7 INV1:IN VSS 0.783500PF
   R24 DF1:C INV1:IN 1.80833E-01
*|NET OUT 0.148478PF
*|S(OUT:1 45.0 15.0)
*|P(OUT O 0.0PF 50.0 5.0)
*|I(INV1:OUT INV1 OUT O 0.0PF 40.0 15.0)
   C8 INV1:OUT VSS 0.147069PF
   C9 OUT:1 VSS 0.632813PF
   C10 OUT VSS 0.776250PF
   R25 INV1:OUT OUT:1 3.11000E00
   R26 OUT:1 OUT 3.03333E00

* Description of Instances
XDF1 DF1:A DF1:B DF1:C DFF
XINV1 INV1:IN INV1:OUT INV
.ENDS
.END
```

Overview of SPEF Files

The Standard Parasitics Exchange Format (SPEF) file structure is described in IEEE standard *IEEE-1481*. For information about how to obtain the complete SPEC (*IEEE-1481*) specification, or any other documents from IEEE, see:

http://www.ieee.org/products/onlinepubs/stand/standards.html

SPEF File Structure

The IEEE-1481 specification requires the following file structure in a SPEF file. Parameters in [brackets] are optional:

```
SPEF_file : :=

*SPEF version
*DESIGN design_name
*DATE date
*VENDOR vendor
*PROGRAM program_name
*VERSION program_version
*DESIGN_FLOW flow_type {flow_type}
*DIVIDER divider
*DELIMITER delimiter
*BUS_DELIMITER bus_prefix bus_suffix
*T_UNIT time_unit NS|PS
*C_UNIT capacitance_unit FF|PF
*R_UNIT resistance_unit OHM|KOHM
*L_UNIT inductance_unit HENRY|MH|UH

[*NAME_MAP name_index name_id|bit|path|name|physical_ref]
[*POWER_NETS logical_power_net physical_power_net ...]
[*GROUND_NETS ground_net ...]
[*PORTS logical_port I|B|O
   *C coordinate ...
   *L par_value
   *S rising_slew falling_slew [low_threshold high_threshold]
   *D cell_type]
[*PHYSICAL_PORTS [physical_instance delimiter]
   physical_port I|B|O
   *C coordinate ...
   *L par_value
   *S rising_slew falling_slew [low_threshold high_threshold]
   *D cell_type]

[*DEFINE logical_instance design_name |
 *PDEFINE physical_instance design_name]

*D_NET net_path total_capacitance
   [*V routing_confidence]
   [*CONN
      *P [logical_instance delimiter] logical_port|physical_port
         I|B|O
         *C coordinate ...
         *L par_value
         *S rising_slew falling_slew
            [low_threshold high_threshold]
         *D cell_type
      |
      *I [physical_instance delimiter] logical_pin|physical_node
         I|B|O
```

```
          *C coordinate ...
          *L par_value
          *S rising_slew falling_slew
             [low_threshold high_threshold]
          *D cell_type
       *N net_name delimiter net_number coordinate
    [*CAP cap_id node1 [node2] capacitance]
    [*RES res_id node1 node2 resistance]
    [*INDUC induc_id node1 node2 inductance]
*END
```

Table 16 SPEF Parameters

Parameter	Definition
*SPEF	Specifies that the file is in SPEF format.
{version}	Version number of the SPEF specification, such as “IEEE 1481-1998”.
*	Words that start with an asterisk (*) are keywords.
|	Or. For example, NS|PS means choose either nanoseconds or picoseconds as the time units.
design_name	Name of your circuit design.
date	Date and time when a parasitic extraction tool (such as Star-RCXT) generated the SPEF file.
vendor	Name of the vendor (such as Synopsys) whose tools you used to generate the SPEF file (optional).
program_name	Name of the program (such as Star-RCXT) that generated the SPEF file.
program_version	Version number of the program that generated the SPEF file.

Table 16 SPEF Parameters (Continued)

Parameter	Definition
flow_type	One or more of the following flow types: ▪ EXTERNAL_LOADS: The SPEF file defines all external loads (if any). If you do not specify this flow type, then some or all external loads are not defined in this SPEF file. If HSPICE RF cannot find external load data outside the SPEF file, it reports an error. ▪ EXTERNAL_SLEWS: The SPEF file defines all external slews (if any). If you do not specify this flow type, then some or all external slews are not defined in this SPEF file. If HSPICE RF cannot find external slew data outside the SPEF file, it reports an error. ▪ FULL_CONNECTIVITY: A SPEF file defines all net connectivity. If you do not specify this flow type, then some or all net connectivity is not defined in this SPEF file. If HSPICE RF cannot find connectivity data outside the SPEF file, it issues an error. This flow does not look for presence or absence of power and ground nets, or any other nets that do not correspond to the logical netlist. If a SPEC file includes FULL_CONNECTIVITY and MISSING_NETS, HSPICE RF reports an error. ▪ MISSING_NETS: If any logical nets are not defined in the netlist, HSPICE RF merges missing parasitic data from another source. If it does not find another source, HSPICE RF rereads the netlist and estimates the missing parasitics. This flow does not look for presence or absence of power and ground nets, or any other nets that do not correspond to the logical netlist. If you use FULL_CONNECTIVITY and MISSING_NETS in the same SPEF file, HSPICE RF reports an error. ▪ NETLIST_TYPE_VERILOG, NETLIST_TYPE_VHDL87, NETLIST_TYPE_VHDL93, or NETLIST_TYPE_EDIF: Specifies the type of naming conventions used in the SPEF file. If you specify more than one format in one SPEF file, HSPICE RF reports an error. ▪ ROUTING_CONFIDENCE *positive_integer*: Specifies a default routing confidence value for all nets in the SPEF file. ▪ ROUTING_CONFIDENCE_ENTRY *positive_integer* character_string: Specifies one or more characters that represent additional routing confidence values, which you can assign to nets in the SPEF file.

Table 16 SPEF Parameters (Continued)

Parameter	Definition
flow_type (continued)	▪ NAME_SCOPE LOCAL|FLAT: Specifies whether paths in the SPEF file are LOCAL (relative to the current SPEF file) or FLAT (relative to the top level of your circuit design). ▪ SLEW_THRESHOLDS low high: Specifies low and high default input slew thresholds for your circuit design as a percentage of the voltage level for the input pin. ▪ PIN_CAP NONE|INPUT_OUTPUT|INPUT_ONLY: Specifies the type of pin capacitance to include when calculating the total capacitance for all nets in the SPEF file, either no capacitance, all input and output capacitances, or only input capacitances.
divider	Character used to divide levels of hierarchy in a circuit path name. Must be one of the following characters: . / : | For example, X1/X2 means that X2 is a subcircuit of the X1 circuit.
delimiter	Character used to separate the name of an instance and a pin in a concatenated instance pin name. Must be one of these characters: . / : |
bus_prefix bus_suffix	Delimiter characters that precede and follow a bus bit or an arrayed instance number. If these characters are not matching pairs, HSPICE RF reports an error. Valid bus delimiter prefix and suffix character pairs are brackets “[ ]”, braces “{ }”, parentheses “( )”, or angle brackets “< >”>
time_unit	A positive number. For example, 10 PS means use time units of 10 picoseconds. 5 NS means use time units of 5 nanoseconds.
capacitance_unit	A positive number. For example, 10 PF means capacitance units of 10 picofarads. 5 FF means use capacitance units of 5 femtoseconds.
resistance_unit	Positive number. For example, 10 OHM sets resistance units to 10 ohms. 5 KOHM sets resistance units to 5 kilo ohms.
inductance_unit	A positive number. For example, 10 HENRY means use inductance units of 10 henries. 5 MH means use inductance units of 5 millihenries. 2 UH means use inductance units of 2 micro-henries.

Table 16 SPEF Parameters (Continued)

Parameter	Definition
name_index	Name used throughout a SPEF file. To reduce file space, you can map other names to this name.
name_id|bit|path|name| physical_ref	A name identifier, bit, path, name, or physical reference to map to the name_index.
logical_power_net	Logical path (or logical path index) to a power net.
physical_power_net	Physical path (or physical path index) to a power net. You can specify multiple *logical_power_net physical_power_net* pairs.
ground_net	Name of a net to use as a ground net. You can specify multiple ground net names.
logical_port	Logical name of an input, output, or bidirectional port.
coordinate	Geometric location of a logical or physical port.
par_value	Either a single float value, or a triplet in float:float:float form.
rising_slew	Rising slew of the waveform for the port. T_UNIT defines the time unit for the waveform.
falling_slew	Rising slew of the waveform for the port. T_UNIT defines the time unit for the waveform.
low_threshold	Low voltage threshold as a percentage of the port's input voltage. Can bed one float value or a triplet in float:float:float form.
high_threshold	High voltage threshold as a percentage of the input voltage for the port. Either a single *float* value or a triplet in *float*:*float*:*float* form.
cell_type	Type of cell that drives the port. If you do not know the cell type, use the reserved word UNKNOWN_DRIVER as the cell type.
physical_port	Physical name of an input, output, or bidirectional port.

Table 16 SPEF Parameters (Continued)

Parameter	Definition
logical_instance	Logical name of a subcircuit in your *design_name* circuit design. You can specify more than one logical_instance. Whenever you specify a logical instance name, you must set NAME_SCOPE to FLAT. If you connect a logical net to a physical port, HSPICE RF reports an error.
physical_instance	Physical name of a subcircuit in your *design_name* circuit design. You can specify more than one physical_instance. Whenever you specify a physical instance name, you must set NAME_SCOPE to FLAT. If you connect a physical net to a logical port, HSPICE RF reports an error.
routing_confidence	One of the following positive integers: ▪ 10: Statistical wire load model. ▪ 20: Physical wire load model. ▪ 30: Physical partitions with locations, no cell placement. ▪ 40: Estimated cell placement with Steiner tree-based route. ▪ 50: Estimated cell placement with global route. ▪ 60: Final cell placement with Steiner route. ▪ 70: Final cell placement with global route. ▪ 80: Final cell placement, final route, 2d extraction. ▪ 90: Final cell placement, final route, 2.5d extraction. ▪ 100: Final cell placement, final route, 3d extraction.
logical_pin	Logical name of a pin.
physical_node	Physical name of a node.
net_name	Name of a net in a circuit or subcircuit.
cap_id	Unique identifier for capacitance between two specific nodes.
res_id	Unique identifier for resistance between two specific nodes.
induc_id	Unique identifier for inductance between two specific nodes.
node1	First of two nodes, between which you are specifying a capacitance, resistance, or inductance value.

Table 16 SPEF Parameters (Continued)

Parameter	Definition
node2	Second of two nodes, between which you are specifying a capacitance, resistance, or inductance value. For a capacitance value, if you do not specify a second node name, HSPICE RF assumes that the second node is ground.
capacitance	Specifies the capacitance value assigned to a *cap_id* identifier. *capacitance_unit* defines the units of capacitance. For example, if you set capacitance to 5 and capacitance_unit to 10 PF, then the actual capacitance value is 50 picoFarads.
resistance	Specifies the resistance value assigned to a *res_id* identifier. *resistance_unit* defines the units of resistance. For example, if you set *resistance* to 5 and *resistance_unit* to 5 KOHM, then the actual resistance value is 25 kilo ohms.
inductance	Specifies the resistance value assigned to an *induc_id* identifier. *inductance_unit* defines the units of inductance. For example, if you set *inductance* to 6 and *inductance_unit* to 2 UH, then the actual inductance value is 12 microhenries.

SPEF File Example

```
*SPEF "IEEE 1481-1998"
*DESIGN   "My_design"
*DATE   "11:26:34 Friday June 28, 2002"
*VENDOR   "Synopsys, Inc."
*PROGRAM   "Star-RCXT"
*VERSION   "2002.2."
*DESIGN_FLOW   "EXTERNAL_LOADS" "EXTERNAL_SLEWS" "MISSING_NETS"
*DIVIDER   /
*DELIMITER   :
*BUS_DELIMITER   [ ]
*T_UNIT   1 NS
*C_UNIT   1 PF
*R_UNIT   1 OHM
*L_UNIT   1 HENRY

*POWER_NETS   VDD
*GND_NETS   VSS

*PORTS
CONTROL O *L 30 *S 0 0
FARLOAD O *L 30 *S 0 0
INVX1FNTC_IN I *L 30 *S 5 5
NEARLOAD O *L 30 *S 0 0
TREE O *L 30 *S 0 0
```

If you use triplet format, the above section would look like this:

```
*PORTS
CONTROL O *L 30:30:30 *S 0:0:0 0:0:0
FARLOAD O *L 30:30:30 *S 0:0:0 0:0:0
INVX1FNTC_IN I *L 30:30:30 *S 5:5:5 5:5:5
NEARLOAD O *L 30:30:30 *S 0:0:0 0:0:0
TREE O *L 30:30:30 *S 0:0:0 0:0:0
```

This triplet formatting principle applies to the rest of this example.

```
*D_NET INVX1FNTC_IN 0.033
*CONN
*P INVX1FNTC_IN I
*I FL_1281:A *L 0.033
*END
*D_NET INVX1FNTC 2.033341

*CONN
*I FL_1281:X O *L 0.0
*I I1184:A I *L 0.343
*I FL_1000:A I *L 0.343
*I NL_1000:A I *L 0.343
*I TR_1000:A I *L 0.343

*CAP
216 FL_1000:A 0.346393
217 I1184:A 0.344053
218 INVX1FNTC_IN 0
219 INVX1FNTC_IN:10 0.154198
220 INVX1FNTC_IN:11 0.117827
221 INVX1FNTC_IN:12 0.463063
222 INVX1FNTC_IN:13 0.0384381
223 INVX1FNTC_IN:14 0.00246845
224 INVX1FNTC_IN:15 0.00350198
225 INVX1FNTC_IN:16 0.00226712
226 INVX1FNTC_IN:17 0.0426184
227 INVX1FNTC_IN:18 0.0209701
228 INVX1FNTC_IN:2 0.0699292
229 INVX1FNTC_IN:20 0.019987
230 INVX1FNTC_IN:21 0.0110279
231 INVX1FNTC_IN:24 0.0192603
232 INVX1FNTC_IN:25 0.0141824
233 INVX1FNTC_IN:3 0.0520437
234 INVX1FNTC_IN:4 0.0527105
235 INVX1FNTC_IN:5 0.1184749
236 INVX1FNTC_IN:6 0.0468458
237 INVX1FNTC_IN:7 0.0391578
238 INVX1FNTC_IN:8 0.0113856
239 INVX1FNTC_IN:9 0.0142528
240 NL_1000:A 0.344804
241 TR_000:A 0.34506

*RES
152 INVX1FNTC_IN INVX1FNTC_IN:18 8.39117
153 INVX1FNTC_IN INVX1FNTC_IN:5 25.1397
154 INVX1FNTC_IN:11 INVX1FNTC_IN:20 4.59517
155 INVX1FNTC_IN:12 INVX1FNTC_IN:13 3.688
156 INVX1FNTC_IN:13 INVX1FNTC_IN:17 25.102
```

```
157 INVX1FNTC_IN:14 INVX1FNTC_IN:16 0.0856444
158 INVX1FNTC_IN:14 NL_1000:A 0.804
159 INVX1FNTC_IN:15 INVX1FNTC_IN:16 1.73764
160 INVX1FNTC_IN:15 INVX1FNTC_IN:24 0.307175
161 INVX1FNTC_IN:17 INVX1FNTC_IN:25 5.65517
162 INVX1FNTC_IN:18 FL_1000:A 1/36317
163 INVX1FNTC_IN:2 INVX1FNTC_IN:4 6.95371
164 INVX1FNTC_IN:2 INVX1FNTC_IN:5 50.9942
165 INVX1FNTC_IN: INVX1FNTC_IN:21 4.71035
166 INVX1FNTC_IN: I1184:A 0.403175
167 INVX1FNTC_IN: TR_1000:A 0.923175
168 INVX1FNTC_IN: INVX1FNTC_IN:12 31.7256
169 INVX1FNTC_IN: INVX1FNTC_IN:4 11.9254
170 INVX1FNTC_IN: INVX1FNTC_IN:7 25.3618
171 INVX1FNTC_IN: INVX1FNTC_IN:6 23.3057
172 INVX1FNTC_IN: INVX1FNTC_IN:24 8.64717
173 INVX1FNTC_IN: INVX1FNTC_IN:8 7.46529
174 INVX1FNTC_IN: INVX1FNTC_IN:10 2.04729
175 INVX1FNTC_IN: INVX1FNTC_IN:10 10.8533
176 INVX1FNTC_IN: INVX1FNTC_IN:11 1.05164

*END

*D_NET NE_794 1.98538

*CONN
*I NL_1039:X O *L 0 *D INVX
*I NL_2039:A I *L 0.343
*I NL_1040:A I *L 0.343

*CAP
3387 NE_794 0
3388 NE_794:1 0.0792492
3389 NE_794:10 0.0789158
3390 NE_794:11 0.0789991
3391 NE_794:12 0.0789991
3392 NE_794:13 0.0792992
3393 NE_794:14 0.00093352
3394 NE_794:15 0.00063346
3395 NE_794:16 0.0792992
3396 NE_794:17 0.80116
3397 NE_794:18 0.80116
3398 NE_794:19 0.00125452
3399 NE_794:2 0.0789158
3400 NE_794:20 0.00336991
3401 NE_794:21 0.00668512
3402 NE_794:23 0.00294932
3403 NE_794:25 0.00259882
```

```
3404 NE_794:26 0.00184653
3405 NE_794:3 0.0789158
3406 NE_794:4 0.0796826
3407 NE_794:5 0.0796826
3408 NE_794:6 0.0789991
3409 NE_794:7 0.0789991
3410 NE_794:8 0.0793992
3411 NE_794:9 0.0789158
3412 NL_1039:X 0.00871972
3413 NL_1040:A 0.344453
3414 NL_2039:A 0.343427

*RES
2879 NE_794:1 NE_794:13 66.1953
2880 NE_794:1 NE_794:2 0.311289
2881 NE_794:11 NE_794:12 0.311289
2882 NE_794:13 NE_794:14 0.353289
2883 NE_794:14 NE_794:19 0.365644
2884 NE_794:15 NE_794:16 0.227289
2885 NE_794:15 NE_794:20 0.239644
2886 NE_794:17 NE_794:18 0.14
2887 NE_794:19 NE_794:21 0.0511746
2888 NE_794:2 NE_794:9 65.9153
2889 NE_794:20 NE_794:23 1.15117
2890 NE_794:21 NL_1039:X 3.01917
2891 NE_794:25 NE_794:26 0.166349
2892 NE_794:26 NL_1040:A 0.651175
2893 NE_794:3 NE_794:10 65.9153
2894 NE_794:3 NE_794:4 0.311289
2895 NE_794:4 NE_794:17 66.5437
2896 NE_794:5 NE_794:18 66.5437
2897 NE_794:5 NE_794:6 0.311289
2898 NE_794:6 NE_794:11 65.98853
2899 NE_794:7 NE_794:12 65.9853
2900 NE_794:7 NE_794:8 0.311289
2901 NE_794:8 NE_794:16 66.3213
2902 NE_794:9 NE_794:10 0.311289
2903 NL_1039:X NE_794:25 1.00317
2904 NL_2039:A NE_794:23 0.171175

*END
```

Linear Acceleration

Linear acceleration, by using the SIM_LA option, accelerates the simulation of circuits that include large linear RC networks. To achieve this acceleration, HSPICE RF reduces all matrices that represent RC networks. The result is a smaller matrix that maintains the original port behavior, yet achieves significant savings in memory and computation. Thus, the SIM_LA option is ideal for circuits with large numbers of resistors and capacitors, such as clock trees, power lines, or substrate networks.

In general, the RC elements are separated into their own network. The nodes shared by both main circuit elements (including .PRINT, .PROBE, and .MEASURE statements), and RC elements. are the port nodes of the RC network,. All other RC nodes are internal nodes. The currents flowing into the port nodes are a frequency-dependent function of the voltages at those nodes.

The multiport admittance of a network represents this relationship.

- The SIM_LA option formulates matrices to represent multiport admittance.
- Then, to eliminate as many internal nodes as possible, it reduces the size of these matrices, while preserving the admittance, otherwise known as port node behavior.
- The amount of reduction depends on the *f*0 upper frequency, the threshold frequency where SIM_LA preserves the admittance. This is shown graphically in Figure 26.

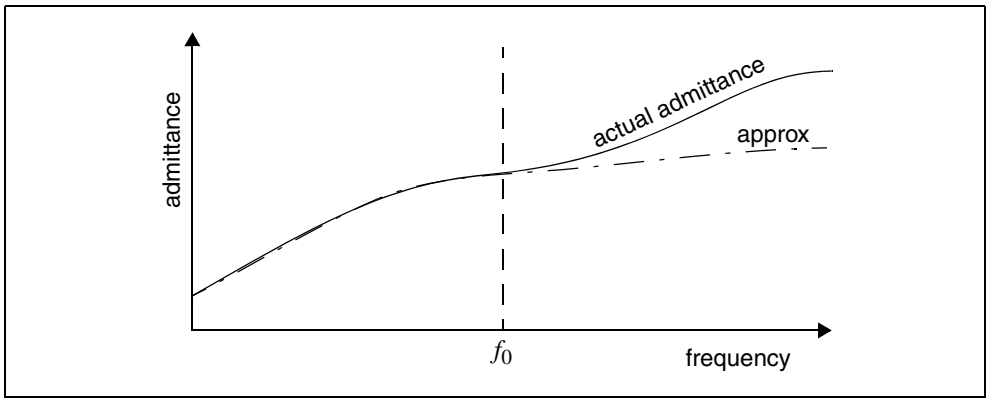


Figure 26 Multiport Admittance vs. Frequency

The SIM_LA option is very effective for post-layout simulation, because of the volume of parasitics. For frequencies below *f*0, the *approx* signal matches that of the original admittance. Above f_0, the two waveforms diverge, but presumably the higher frequencies are not of interest. The lower the *f*0 frequency, the greater the amount of reduction.

For the syntax and description of this control option, see .OPTION SIM_LA in the *HSPICE Reference Manual: Commands and Control Options*.

You can choose one of two algorithms, explained in the following sections:

- PACT Algorithm
- PI Algorithm

PACT Algorithm

The PACT (Pole Analysis via Congruence Transforms) algorithm reduces the RC networks in a well-conditioned manner, while preserving network stability.

- The transform preserves the first two moments of admittance at DC (slope and offset), so that DC behavior is correct (see Figure 27).
- The algorithm preserves enough low-frequency poles from the original network to maintain the circuit behavior up to a specified maximum frequency *f*0, within the specified tolerance.

This approach is the most accurate of the two algorithms, and is the default.

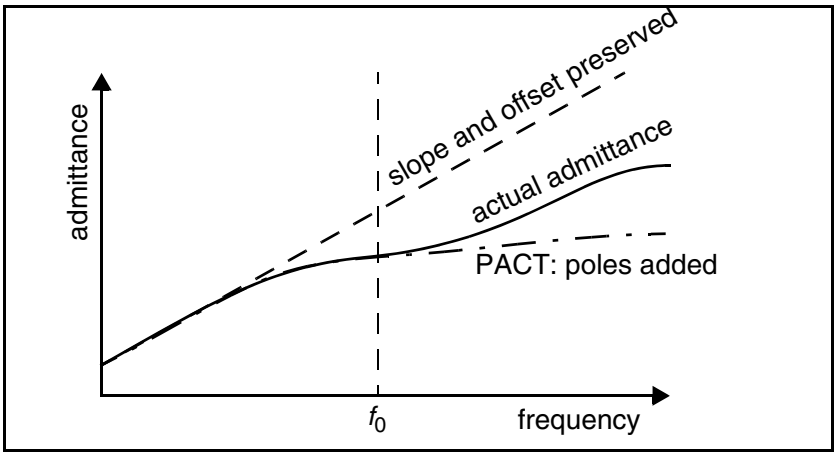


Figure 27 PACT Algorithm

PI Algorithm

This algorithm creates a *pi* model of the RC network.

- For a two-port, the *pi* model reduced network consists of:
  - a resistor connecting the two ports, and
  - a capacitor connecting each port to ground

  The result resembles the Greek letter pi.
- For a general multiport, `SIM_LA` preserves the DC admittance between the ports, and the total capacitance that connects the ports to ground. All floating capacitances are lumped to ground.

Linear Acceleration Control Options Summary

In addition to `.OPTION SIM_LA`, other options are available to control the maximum resistance and minimum capacitance values to preserve, and to limit the operating parameters of the PACT algorithm. Table 17 on page 309 contains a summary of these control options. For the syntax and descriptions of these options, see the respective sections in the HSPICE and RF Netlist Simulation Control Options in the *HSPICE Reference Manual: Commands and Control Options*.

Table 17 PACT Options

Syntax	Description
.OPTION SIM_LA=PACT | PI	Activates linear matrix reduction and selects between two methods.
.OPTION SIM_LA_FREQ=<value>	Upper frequency where you need accuracy preserved. *value* is the upper frequency for which the PACT algorithm preserves accuracy. If *value* is 0, PACT drops all capacitors, because only DC is of interest. The maximum frequency required for accurate reduction depends on both the technology of the circuit and the time scale of interest. In general, the faster the circuit, the higher the maximum frequency. The default is 1GHz.

Table 17 PACT Options (Continued)

Syntax	Description
.OPTION SIM_LA_MAXR=<value>	Maximum resistance for linear matrix reduction. *value* is the maximum resistance preserved in the reduction. SIM_LA assumes that any resistor greater than *value* has an infinite resistance, and drops the resistor after reduction finishes. The default is 1e15 ohms.
.OPTION SIM_LA_MINC=<value>	Minimum capacitance for linear matrix reduction. *value* is the minimum capacitance preserved in the reduction. After reduction completes, SIM_LA lumps any capacitor smaller than *value* to ground. The default is 1e-16 farads.
.OPTION SIM_LA_MINMODE= ON|OFF	Reduces the number of nodes instead of the number of elements.
.OPTION SIM_LA_TIME=<value>	Minimum time for which accuracy must be preserved. *value* is the minimum switching time for which the PACT algorithm preserves accuracy. HSPICE RF does not accurately represent waveforms that occur more rapidly than this time. SIM_LA_TIME is simply the dual of SIM_LA_FREQ. The default is equivalent to setting LA_FREQ=1 GHz. The default is 1ns.
.OPTION SIM_LA_TOL=<value>	Error tolerance for the PACT algorithm. *value* is the error tolerance for the PACT algorithm, is between 0.0 and 1.0. The default is 0.05.

Example

In this example, the circuit has a typical risetime of 1ns. Set the maximum frequency to 1 GHz, or set the minimum switching time to 1ns.

```
.OPTION SIM_LA_FREQ = 1GHz
-or-
.OPTION SIM_LA_TIME = 1ns
```

However, if spikes occur in 0.1ns, HSPICE will not accurately simulate them. To capture the behavior of the spikes, use:

```
.OPTION SIM_LA_FREQ = 10GHz
-or-
.OPTION SIM_LA_TIME = 0.1ns
```

Note: Higher frequencies (smaller times) increase accuracy, but only up to the minimum time step used in HSPICE.

13

Using HSPICE with HSPICE RF

Describes how various analysis features differ in HSPICE RF as compared to standard HSPICE.

This first section of this chapter describes topics related to transient analysis and the other section describe other differences between HSPICE and HSPICE RF.

These topics are covered in the following sections

- RF Numerical Integration Algorithm Control
- RF Transient Analysis Accuracy Control
- RF Transient Analysis Output File Formats
- Compressing Analog Files

RF Numerical Integration Algorithm Control

In HSPICE RF, you can select either the Backward-Euler or Trapezoidal integration algorithm. Each of these algorithms has its own advantages and disadvantages for specific circuit types. For pre-charging simulation or timing critical simulation, the Trapezoidal algorithm usually improves accuracy.

You use the `SIM_ORDER` option to control the amount of Backward-Euler (BE) to mix with the Trapezoidal (TRAP) method for hybrid integration. For example,

```
.OPTION SIM_ORDER=x
```

Setting `SIM_ORDER` to its lowest value selects Backward-Euler integration algorithm, and setting it to its highest value selects Trapezoidal integration.

For the syntax and description of this control option, see .OPTION SIM_ORDER in the *HSPICE Reference Manual: Commands and Control Options*.

RF Transient Analysis Accuracy Control

The default time step method in HSPICE RF mixes timestep algorithms Trapezoidal and second-order Gear (Gear-2). This yields a more accurate scheme than Trapezoidal or Backward-Euler. Also, detection of numerical oscillations inserts fewer Backward-Euler steps than in previous HSPICE versions.

The following sections discuss these topics:

- .OPTION SIM_ACCURACY
- Algorithm Control

.OPTION SIM_ACCURACY

You use the `SIM_ACCURACY` option to modify the size of timesteps in HSPICE RF. For example,

`.OPTION SIM_ACCURACY=`*`val`*

A timestep is a time interval at which you evaluate a signal. HSPICE RF discretely expresses the time continuum as a series of points. At each point or timestep, a circuit simulator evaluates the corresponding voltage or current value of a signal. Thus, a resulting signal waveform is a series of individual data points; connecting these points results in a smooth curve.

You can apply different accuracy settings to different blocks or time intervals. The syntax to set accuracy on a block, instance, or time interval is similar to the settings used for a power supply.

Note: An `.OPTION SIM_ACCURACY` takes precedence over an `.OPTION ACCURATE`.

For the syntax and description of this control option, see .OPTION SIM_ACCURACY in the *HSPICE Reference Manual: Commands and Control Options*.

Algorithm Control

In HSPICE RF, you can select the Backward-Euler, Trapezoidal, Gear, or hybrid method algorithms. Each of these algorithms has its own advantages and disadvantages for specific circuit types. These methods have tradeoffs related to accuracy, avoidance of numerical oscillations, and numerical damping of circuit oscillations. For pre-charging simulation or timing critical simulations, the Trapezoidal algorithm usually improves accuracy.

.OPTION METHOD

You use the `METHOD` option to select a numeric integration method for a transient analysis.

HSPICE RF supports three basic timestep algorithms: Trapezoidal (`TRAP`), second-order Gear (Gear-2), and Backward-Euler (BE). Backward-Euler is the same as first-order Gear. Also, HSPICE RF supports a hybrid algorithm (`TRAPGEAR`), which is a mixture of the three basic algorithms.

HSPICE RF contains an algorithm for auto-detection of numerical oscillations commonly encountered with trapezoidal integration. If HSPICE RF detects such oscillations, it inserts BE steps, but not more than one BE step for every 10 time steps. To turn off auto-detection, use the `PURETP` option.

The `TRAPGEAR` method, combining 90% trapezoidal with 10% Gear-2. HSPICE RF inserts BE steps, when the simulator encounters a breakpoint, or when the auto-detection algorithm finds numerical oscillations.

For the syntax and description of this control option, see .OPTION METHOD in the *HSPICE Reference Manual: Commands and Control Options*.

.OPTION MAXORD

You use the `MAXORD` option to select the maximum order of integration for the `GEAR` method. Either the first-order Gear (Backward-Euler), or the second-order Gear (Gear-2) integration method.

For the syntax and description of this control option, see .OPTION MAXORD in the *HSPICE Reference Manual: Commands and Control Options*.

.OPTION SIM_ORDER

You use the SIM_ORDER option to control the amount of Backward-Euler (BE) to mix with the Trapezoidal method for hybrid integration. This option affects time stepping when you set .OPTION METHOD to TRAP or TRAPGEAR.

For the syntax and description of this control option, see .OPTION SIM_ORDER in the *HSPICE Reference Manual: Commands and Control Options*.

.OPTION SIM_TG_THETA

You use the SIM_TG_THETA option to control the amount of Gear-2 method to mix with trapezoidal integration for the hybrid TRAPGEAR method.

For the syntax and description of this control option, see .OPTION SIM_TG_THETA in the *HSPICE Reference Manual: Commands and Control Options*.

.OPTION SIM_TRAP

You use the SIM_TRAP option to change the default SIM_TG_THETA to 0, so that method=trapgear acts like METHOD=TRAP.

For the syntax and description of this control option, see .OPTION SIM_TRAP in the *HSPICE Reference Manual: Commands and Control Options*.

.OPTION PURETP

You use the PURETP option to turn off insertion of Backward-Euler (BE) steps due to auto-detection of numerical oscillations.

For the syntax and description of this control option, see .OPTION PURETP in the *HSPICE Reference Manual: Commands and Control Options*.

.OPTION SIM_OSC_DETECT_TOL

You use the SIM_OSC_DETECT_TOL option to specify the tolerance for detecting numerical oscillations. If HSPICE RF detects numerical oscillations, it inserts Backward-Euler (BE) steps. Smaller values of this tolerance result in fewer BE steps.

For the syntax and description of this control option, see .OPTION SIM_OSC_DETECT_TOL in the *HSPICE Reference Manual: Commands and Control Options*.

RF Transient Analysis Output File Formats

The default output format for transient analysis in HSPICE RF is the same as in HSPICE: the .tr0 file format. See Transient Analysis in the *HSPICE User Guide: Simulation and Analysis*. HSPICE RF supports these output formats, which are described in this section:

- Tabulated Data Output
- WDB Output Format
- NW Output Format
- VCD Output Format
- turboWave Output Format (tw)
- Undertow Output Format (ut)
- CSDF Output Format

If your netlist includes an unsupported output format, HSPICE RF prints a warning message, indicating that the selected format is unsupported. HSPICE RF then automatically defaults the output to TR0 format.

Waveform Results

The waveform results of Monte Carlo runs are expected to be different in HSPICE vs. RF because the results depend on the exact sequence in which the random numbers are used, and it is not practical to make them the same. However, if you run a large number of samples, the statistical results from HSPICE and RF should converge to the same values.

Tabulated Data Output

HSPICE RF outputs all analog waveforms specified in a `.PRINT` statement. HSPICE RF saves these waveforms as ASCII tabulated data, into a file with the .PRINT extension.

To display waveforms graphically, Custom WaveView can directly read the tabulated data. For more information about Custom WaveView, see the *CustomExplorer and Custom WaveView User Guide*.

Note: Tabulated data excludes waveforms specified in `.PROBE` statements.

WDB Output Format

You can use the waveform database (WDB) output format in `.OPTION POST`. It was developed for maximum efficiency. The output file is *.wdb#. For example, to output to a *.wdb# file, enter:

```
.OPTION POST=wdba
```

Signals across multiple hierarchies, that map to the same node, are named together. They also share the same waveform data.

You can also set up the database so that Custom WaveView extracts one signal at a time. This means that Custom WaveView does not need to read the entire output file to display a single waveform.

The WDB format was designed to make accessing waveform data faster and more efficient. It is a true database so the waveform browser does not have to load the complete waveform file for you to view a single signal. This feature is especially useful if the size of the waveform file is several gigabytes.

Furthermore, the WDB format is usually more compact than XP and NW (described later in this section). However, if the NW file is already very small, then WDB offers little advantage in size or speed.

You can compress WDB files. For additional information, see Compressing Analog Files on page 320.

TR Output Format

HSPICE RF stores simulation results for analysis by using the Custom WaveView graphical interface method. For example, these commands output a *.tr# file in TR format:

- `.OPTION POST=1` saves the results in binary format
- `.OPTION POST=2` saves the results in ASCII format.

NW Output Format

HSPICE RF outputs the NW format to a file with the .nw# extension. You need a Synopsys waveform display tool to process a file in NW format. For example, to output to a *.nw# file, enter:

```
.OPTION POST=nw
```

You can compress NW files. For additional information, see Compressing Analog Files on page 320.

VCD Output Format

To output your waveforms from HSPICE RF in VCD (Value Change Dump) format, set the `VCD` option in conjunction with the `.LPRINT` statement. For example,

```
.OPTION VCD
.LPRINT (0.5 4.5) v(0) v(2) v(6)
```

.LPRINT Statement

You use the `.LPRINT` statement to produce output in VCD file format from transient analysis. For example,

```
.LPRINT (v1,v2) output_varable_list
```

For additional information, see .LPRINT in the *HSPICE Reference Manual: Commands and Control Options*.

turboWave Output Format

To use turboWave output format TW, enter:

```
.OPTION POST=tw
```

This format supports analog compression as described in Compressing Analog Files on page 320.

Undertow Output Format

To use Veritools Undertow output format UT, enter:

```
.OPTION POST=ut
```

This format supports analog compression as described in Compressing Analog Files on page 320.

The waveform list in UT format now displays in a hierarchical structure, rather than one flat level as in previous versions.

CSDF Output Format

To use CSDF output format CSDF, enter:

```
.OPTION POST=csdf
.OPTION csdf [overrides .OPTION POST setting]
```

Compressing Analog Files

Analog compression eliminates unnecessary data points from a HSPICE RF voltage or current waveform to reduce the size of the waveform file.

Eliminating Voltage Datapoints

You use the SIM_DELTAV option to determine the selection criteria for HSPICE RF voltage waveforms in WDB or NW format. For example,

```
.OPTION SIM_DELTAV=<value>
```

During simulation, HSPICE RF checks whether the value of the *X* signal at the *n* timestep changes by more than the SIM_DELTAV option, from its previous value at the *n-1* timestep.

- If yes, then HSPICE RF saves the new data point.
- Otherwise, this new data point is lost.

Typically such an algorithm yields a reduced file size with minimal resolution loss as long as you set an appropriate SIM_DELTAV value. If a value for the SIM_DELTAV option is too large, the waveform degrades.

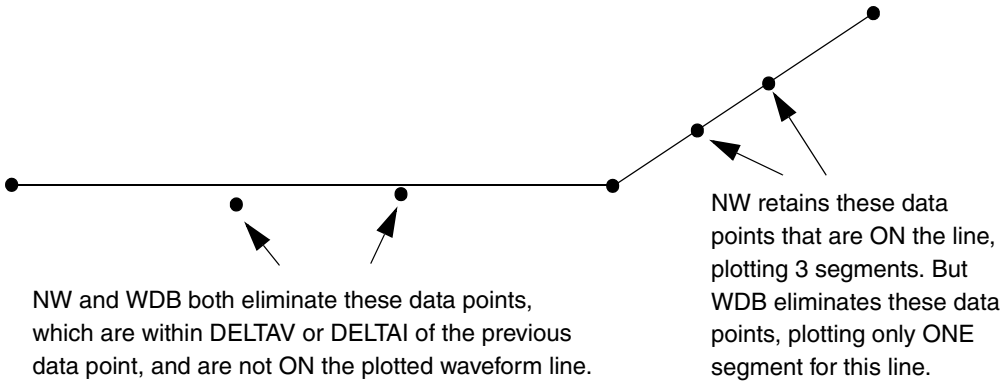


Figure 28 Analog Compression Formats

For a additional information, see .OPTION SIM_DELTAV in the *HSPICE Reference Manual: Commands and Control Options*.

Eliminating Current Datapoints

You use the `SIM_DELTAI` option to determine the selection criteria for HSPICE RF current waveforms in WDB or NW format. For example,

```
.OPTION SIM_DELTAI=val
```

For a additional information, see .OPTION SIM_DELTAI in the *HSPICE Reference Manual: Commands and Control Options*.

14

Advanced Features

Describes how to invoke HSPICE RF and how to perform advanced tasks, including redirecting input and output.

HSPICE RF accepts a netlist file from standard input and delivers the ASCII text simulation results to an HTML file or to stdout. Error and warning messages are forwarded to standard error output.

This chapter describes how to do this as well as how to invoke HSPICE RF and redirect input and output.

These topics are covered in the following sections:

- Creating a Configuration File
- Using Wildcards in HSPICE RF
- Limiting Output Data Size
- Probing Subcircuit Currents
- Generating Measurement Output Files
- Optimization
- Using CHECK Statements
- POWER DC Analysis
- Detecting and Reporting Surge Currents

Creating a Configuration File

You can create a configuration file, called *.hspicerf*, to customize your HSPICE RF simulation. HSPICE RF first searches for *.hspicerf* in your current working

directory, then in your home directory as defined by `$HOME`. The configuration options listed in Table 18 are available for your use.

Table 18 Configuration File Options

Keyword	Description	Example
flush_waveform	Flushes a waveform. If you do not specify a percentage, then the default value is 20%.	`flush_waveform percent%`
ground_floating_ node	Uses .IC statements to set floating nodes in a circuit to ground. You can select three options for grounding floating nodes: ■ If set to `1`, grounds only floating nodes (gates, bulk, control nodes, non-rail bulk) that are included in the .IC set. ■ If set to 2, adds unconnected terminals to this set. ■ If set to 3, uses .IC statements to ground all floating nodes, including dangling terminals.	`ground_floating_ node 1`
hier_delimiter	Changes the delimiter for subcircuit hierarchies from "." to the specified symbol.	`hier_delimiter /`
html	Stores all HSPICE RF output in HTML format.	`htmlhspicerf test` This example creates a file named test.html in the current directory.
integer_node	Removes leading zeros from node names. For example, HSPICE RF considers 0002 and 2 to be the same node. Without this keyword, 0002 and 2 are two separate nodes.	`integer_node`

Table 18 Configuration File Options (Continued)

Keyword	Description	Example
max_waveform_size	Automatically limits the waveform file size. ■ If the number is less than 5000, HSPICE RF resets it to 2G. ■ If you do not set the number, HSPICE RF uses the default, 2G. ■ If you do not set the line, the file size has no limit.	`max_waveform_ size 2000000000`
negative_td	Allows negative time delay input in `pwl` (piecewise linear with repeat), `pl` (piecewise linear), `exp` (exponential, rising time delay only), `sin` (damped sinusoidal), `pulse` (trapezoidal pulse), and `am` (amplitude modulation) formats.	If you do not set `negative_td`, a negative time delay defaults to zero.
port_element_ voltage_ matchload	Allows the alternate Port element definition. A Port element consists of a voltage source in series with a resistor. For the explanation that follows, let the user-specified DC, AC, or transient value of the Port element be V, and let the voltage across the overall port element be Vp. By default, HSPICE RF will set the internal voltage source value to V. The value of Vp will be lower than V, depending on the internal impedance and the network's input impedance. With the alternate definition, the internal voltage source value is adjusted to 2*V, so that Vp=V when the Port element's impedance is matched with the network input impedance. The actual value of Vp will still depend on the port and network impedances.	`port_element_ voltage_ matchload`
rcxt_divider	Defines the hierarchy delimiter in the active nodes file in RCXT format.	`rcxt_divider /`
skip_nrd_nrs	Directs HSPICE RF to consider transistors with matching geometries (except for NRD and NRS) as identical for pre-characterization purposes.	`skip_nrd_nrs`

Table 18 Configuration File Options (Continued)

Keyword	Description	Example
unit_atto	Activates detection of the “atto.” unit. Otherwise, HSPICE RF assumes that “a” represents “amperes.”	`unit_atto`
v_supply	Changes the default voltage supply range for characterization.	`v_supply 3`
wildcard_left_range	Begins range expression.	`wildcard_left_range [`
wildcard_match_all	Matches any group of characters.	`wildcard_match_all *`
wildcard_match_one	Matches any single character.	`wildcard_match_one ?`
wildcard_right_range	Ends range expression.	`wildcard_right_range ]`

Note: For more information about wildcards, see Using Wildcards in HSPICE RF on page 327.

Inserting Comments in a .hspice File

To insert comments into your .hspicerf file, include a number sign character (#) as the first character in a line. For example, this configuration file shows how to use comments in a *.hspicerf* file:

```
# sample configuration file
# the next line of code changes the delimiter
# for subcircuit hierarchies from "," to "^"
hier_delimiter ^
# the next line of code matches any groups of "*" characters
wildcard_match_all *
# the next line of code matches one "?" character
wildcard_match_one ?
# the next line of code begins the range expression with
# the "[" character
wildcard_left_range [
# the next line of code ends the range expression with
# the "]" character
wildcard_right_range ]
```

Using Wildcards in HSPICE RF

You can use wildcards to match node names. HSPICE RF uses wildcards somewhat differently than standard HSPICE.

Before using wildcards, you must define the wildcard configuration in a .hspicerf file. For example, you can define the following wildcards in a .hspicerf file:

```
file .hspicerf
wildcard_match_one     ?
wildcard_match_all     *
wildcard_left_range    [
wildcard_right_range   ]
```

The `.PRINT`, `.PROBE`, `.LPRINT`, and `.CHECK` statements support wildcards in HSPICE RF.

For more information about using wildcards in an HSPICE configuration file, see Using Wildcards in PRINT and PROBE Statements in the *HSPICE User Guide: Simulation and Analysis*.

Limiting Output Data Size

For multi-million transistor simulations, an unrestricted waveform file can grow to several gigabytes in size. The file becomes unreadable in some waveform viewers, and requires excessive space on the hard drive.

This section describes options that limit the number of nodes output to the waveform file to reduce the file size. HSPICE RF supports the following options to control the output:

- SIM_POSTTOP Option
- SIM_POSTSKIP Option
- SIM_POSTAT Option
- SIM_POSTDOWN Option
- SIM_POSTSCOPE Option

SIM_POSTTOP Option

You use the `SIM_POSTTOP` option to limit the data written to your waveform file to data from only the top *n* level nodes. This option outputs instances up to *n* levels deep. For example,

```
.OPTION SIM_POSTTOP=<n>
```

Note: To enable the waveform display interface, you also need the `POST` option.

For additional information, see .OPTION SIM_POSTTOP in the *HSPICE and HSPICE RF Command Reference*.

SIM_POSTSKIP Option

You use the `SIM_POSTSKIP` to have the `SIM_POSTTOP` option skip any instances and their children that the subckt_definition defines. For example,

```
.OPTION SIM_POSTSKIP=<subckt_definition>
```

For additional information, see .OPTION SIM_POSTSKIP in the *HSPICE and HSPICE RF Command Reference*.

SIM_POSTAT Option

You use the SIM_POSTAT

option to limit the waveform output to only the nodes in the specified subcircuit instance. For example,

```
.OPTION SIM_POSTAT=<instance>
```

This option can be used in conjunction with the SIM_POSTTOP option and when present, has precedence over the SIM_POSTSKIP option.

For additional information, see .OPTION SIM_POSTAT in the *HSPICE and HSPICE RF Command Reference*.

SIM_POSTDOWN Option

You use the SIM_POSTDOWN option to include an instance and all children of that instance in the output. For example,

```
.OPTION SIM_POSTDOWN=<instance>
```

It can be used in conjunction with the SIM_POSTTOP option and when present, has precedence over the SIM_POSTSKIP option.

For additional information, see .OPTION SIM_POSTDOWN in the HSPICE and HSPICE RF Command Reference.

SIM_POSTSCOPE Option

You use the SIM_POSTSCOPE option to specify the signal types to probe from within a scope. For example,

```
.OPTION SIM_POSTSCOPE=net|port|all
```

For additional information, see .OPTION SIM_POSTSCOPE in the *HSPICE and HSPICE RF Command Reference*.

Probing Subcircuit Currents

To provide subcircuit power probing utilities, HSPICE RF uses the X() and X0() extended output variables. You can use these X variables in .PROBE, .PRINT, or .MEASURE statements.

The following syntax is for the output variable X():

```
X (subcircuit_node_path)
X0 (subcircuit_node_path)
```

subcircuit_node_path specifies the subcircuit path and the subcircuit node name definition. The node must be either an *external* node in a subcircuit definition or a global node.

X() returns the total current flowing into a subcircuit branch, including all lower subcircuit hierarchies. X0() returns only current flowing into a subcircuit branch, minus any current flowing into lower subcircuit hierarchies. Figure 29 on page 330 illustrates the difference between the X() and X0 () variables.

The dotted line boxes represent subcircuits, and the black circles are the external nodes. The X(X1.vc1) path returns the current of the X1subcircuit, through the *vc1* node, including the current to the X1.X1 and X1.X2 subcircuits as represented by the white (black outlined) arrows. In contrast, X0(X1.vc2) returns only the current flowing through vc2 to the top level of the X1 subcircuit as shown by the black arrows.

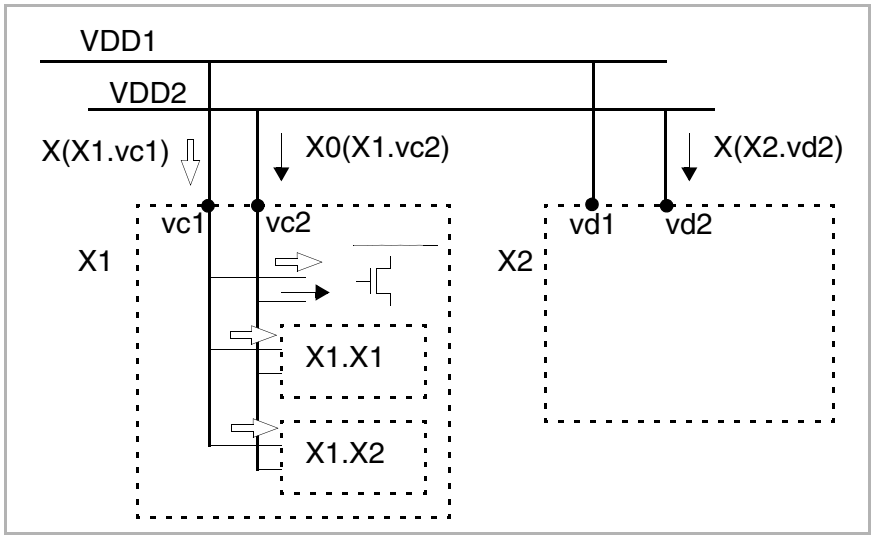


Figure 29 Probing Subcircuit Currents

Example 1

In this example, the first five lines constitute the definition of the `sb1` subcircuit with external nodes named `node1`, `node2`, and `clr`. The line beginning with `X1` is an instance of `sb1` with nodes named;

- `11` (references `node1`)
- `12` (references `node2`)
- `0` (references `clr`)

```
.subckt sb1 node1 node2 clr
* subckt elements
R1 node1 node2 1K
C1 clr node1 1U
.ends
* subcircuit instance
X1 11 12 0 sb1
.PRINT X(X1.node1) 'X(X1.clr) + I(X1.R1)'
```

To find the current flowing into node `11` of the `X1` subcircuit instance, this example uses the `X()` variable. HSPICE RF maps node `11` to the `node1` external node as shown in the first part of the `.PRINT` statement.

The latter half of the `.PRINT` statement illustrates that you can combine the `X()` variable with `I()` variables.

Example 2

In this example, the `X()` variable finds the current through the *in* node of the `S1` subcircuit.

```
.subckt S1 in out
R1 in inp 1K
C1 inp 0 1u
R2 in out 1K
.PROBE X(in)
.ends
```

Generating Measurement Output Files

You can make all of the same measurements with the `.MEASURE` statement in HSPICE RF as you can in HSPICE.

The results of the `.MEASURE` statements appear in a file with one of the following filename extensions:

- .mt# for measurements in transient analysis
- .ms# for measurements in DC analysis
- .ma# for measurements in AC analysis
- .mb# for measurements in HB analysis
- .mp# for measurements in HBNOISE analysis

For more information about .MEASURE statements, see HSPICE and HSPICE RF Netlist Commands in the *HSPICE Reference Manual: Commands and Control Options*.

Optimization

Like HSPICE, HSPICE RF employs an incremental optimization technique. This technique solves the DC parameters first, then the AC parameters, and finally the transient parameters.

To perform optimization, create an input netlist file that specifies:

- Optimization parameters with upper and lower boundary values along with an initial guess.
- An AC, DC, TRAN, HB, or HBOSC optimization statement.
- An optimization model statement.
- Optimization measurement statements for optimization parameters.

If you provide the input netlist file, optimization specifications, limits, and initial guess, then the optimizer reiterates the simulation until it finds an optimized solution.

Usage Notes and Examples

- Optimization works for TRAN, AC, DC, HB, HBOSC, and HBAC analyses.
- You can add the GOAL options in every meaningful .MEASURE statement, like FIND-WHEN, FIND-AT, and so forth.
- A data sweep is not required to be defined in the .HB statement for HB optimization to use the measured result from .MEASURE HBNOISE, PHASENOISE, or HBTRAN statements. Therefore, parameter sweep is not supported for this type of optimization.
- Optimize multiple parameters with multiple goals by selecting .MODEL OPT LEVEL=0 (modified Lavenberg-Marquardt method).
- Optimize single parameters in single measurement situations by selecting .MODEL OPT LEVEL=1 (bisection method).
- Examples
  - Setting optimization parameters

```
.param W=opt1(231u, 100u, 800u)
.param Rs=opt1(10,8,20)
```

- Optimization analysis statement

```
.HB tones=2.25g 2.5g nharms=6,3
 + sweep Pin:dbm -30 0 2
 + sweep optimize = opt1
 + results = gain $measure result to tune the parameters
 + model= optmod1
```

- Selecting an optimization model

```
.model optmod1 opt level=1 $Bisection method
 + itropt=40 relin=1e-4 relout=1e-6 $ accuracy settings
```

- Measurement statements to tune the optimization parameters

```
.measure HB vif find vdb(if+)[-1,1] at 10e-6
.measure HB vrf find vdb(rf+)[0,1] at 10e-6
.measure HB gain=param('vif-vrf') goal=-2
```

- Measurement statement to find the fundamental frequency from HB analysis

```
.measure HB frequency_max FIND 'HERTZ[1]' at=0
```

Optimizing AC, DC. and TRAN Analyses

The HSPICE syntax is followed for optimizing AC, DC. and TRAN analyses. The required statements are:

- Optimization `.PARAM` statement

  `.PARAM` *ParamName*`=OPT`*xxx*`(`*Init*`,`*LoLim*`,`*HiLim*`)`

- Optimizing `.TRAN` statement

  `.TRAN` *tincr1 tstop1* `[`*tincr2 tstop2 ... tincrN tstopN*`]`
  `+ SWEEP OPTIMIZE=OPT`*xxx* `RESULTS=`*measname* `MODEL=`*optmod*

- Optimizing `.MODEL` statement

  `.MODEL` *mname* `OPT LEVEL=[0|1]`

  Where:

  - 0 specifies the Modified Levenberg-Marquardt method. You would use this setting with multiple optimization parameters and goals.

- 1 specifies the Bisection method. You would use this setting with one optimization parameter.

Optimizing HB Analysis

- Optimization with HB Measurements
- Optimization with HBNOISE, PHASENOISE, or HBTRAN Measurements

Optimization with HB Measurements

The required statements are:

- Analysis statement

```
.HB TONES=f1[f2 ... fn] [NHARMS=h1 [,h2 ... hn]]
+ SWEEP parameter_sweep OPTIMIZE=OPTxxx RESULT=measname
+ MODEL=mname
```

- Measure statement

```
.MEASURE HB measname FIND out_var1 AT=val GOAL=val
```

Optimization with HBNOISE, PHASENOISE, or HBTRAN Measurements

The required statements are:

- Analysis statement

```
.HB TONES=f1[f2 ... fn] [NHARMS=h1 [,h2 ... hn]]
+ SWEEP OPTIMIZE=OPTxxx RESULT=measname MODEL=mname
```

For example,

```
.HBOSC tones=1g nharms = 5 optimize = opt1
+ result = y1, y2 model = m1
   .model m1 opt level=0
   .PHASENOISE dec 1 1k 1g
   .meas phasenoise y1 find phnoise at 10k goal = -150dbc
   .meas phasenoise y2 RMSJITTER phnoise units = sec goal =
1.0e-12
```

- Measure statement

```
.MEASURE HBNOISE measname FIND out_var1 AT=val GOAL=val
.MEASURE PHASENOISE measname FIND out_var1 AT=val
+ GOAL=val
.MEASURE HBTRAN measname FIND out_var1 AT=val GOAL=val
```

Optimizing HBOSC Analysis

- Optimization with HB Measurements
- Optimization with HBNOISE, PHASENOISE, or HBTRAN Measurements

Optimization with HB Measurements

The required statements are:

- Analysis statement

```
.HBOSC TONES=f1 [f2 ... fn] [NHARMS=h1 [,h2 ... hn]]
+ SWEEP parameter_sweep OPTIMIZE=OPTxxx RESULT=measname
+ MODEL=mname
```

- Measure statement

```
.MEASURE HB measname FIND out_var1 AT=val GOAL=val
```

Optimization with HBNOISE, PHASENOISE, or HBTRAN Measurements

The required statements are:

- Analysis statement

```
.HBOSC TONES=f1 [f2 ... fn] [NHARMS=h1 [,h2 ... hn]]
+ SWEEP OPTIMIZE=OPTxxx RESULT=measname MODEL=mname
```

For example,

```
.HBOSC tones=1g nharms = 5 sweep x 1 5 1 optimize = opt1
+ result = y1, y2 model = m1
   .model m1 opt level=0
   .PHASENOISE dec 1 1k 1g
   .meas phasenoise y1 find phnoise at 10k goal = -150dbc
   .meas phasenoise y2 RMSJITTER phnoise units = sec goal =
1.0e-12
Measure statement—
.MEASURE HBNOISE measname FIND out_var1 AT=val GOAL=val
.MEASURE PHASENOISE measname FIND out_var1 AT=val
+ GOAL=val
.MEASURE HBTRAN measname FIND out_var1 AT=val GOAL=val
```

Optimization with HBNOISE, PHASENOISE or HBTRAN measurements must not be used in combination with HB measurement optimization as shown in Optimization with HB Measurements.

Using CHECK Statements

The CHECK statements in HSPICE RF offer the following instrumentation:

- Setting Global Hi/Lo Levels
- Slew, Rise, and Fall Conditions
- Edge Timing Verification
- Setup and Hold Verification
- IR Drop Detection

The results of these statements appear in a file with an .err extension. To prevent creating unwieldy files, HSPICE RF reports only the first 10 violations for a particular check in the .err file.

Setting Global Hi/Lo Levels

You use the `.CHECK GLOBAL_LEVEL` statement to globally set the desired high and low definitions for all CHECK statements. For example,

```
.CHECK GLOBAL_LEVEL (hi lo hi_th lo_th)
```

Values for hi, lo, and the thresholds are defined by using this statement.

For syntax and description of this statement, see .CHECK GLOBAL_LEVEL in the *HSPICE Reference Manual: Commands and Control Options*.

Slew, Rise, and Fall Conditions

You use the `.CHECK SLEW` statement to verify that a slew rate occurs within the specified window of time. For example,

```
.CHECK SLEW (min max) node1 [node2 ...] [(hi lo hi_th lo_th)]
```

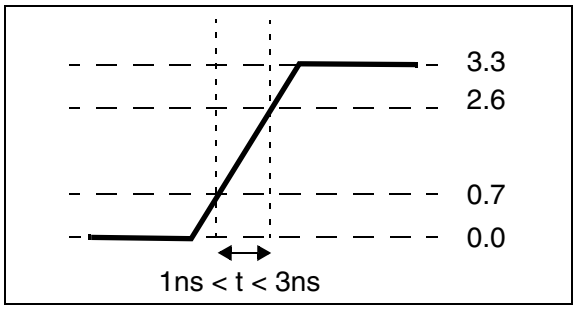


Figure 30 SLEW Example

For syntax and description of this statement, see .CHECK SLEW in the *HSPICE Reference Manual: Commands and Control Options*.

You use the `.CHECK RISE` statement to verify that a rise time occurs within the specified window of time. For example,

```
.CHECK RISE (min max) node1 [node2 ...] [(hi lo hi_th lo_th)]
```

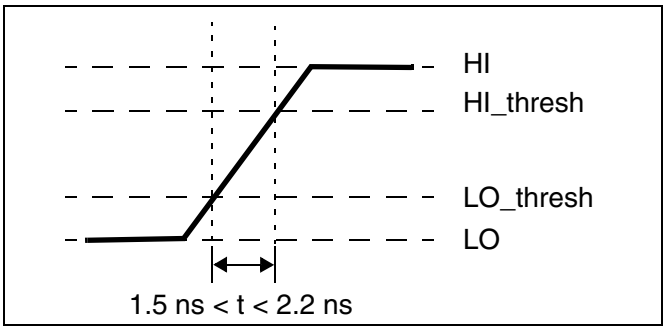


Figure 31 RISE Time Example

For syntax and description of this statement, see .CHECK RISE in the *HSPICE and HSPICE RF Command Reference*.

You use the `.CHECK FALL` statement to verify that a fall time occurs within the specified window of time. For example,

```
.CHECK FALL (min max) node1 [node2 ...>] [(hi lo hi_th lo_th)]
```

For syntax and description of this statement, see .CHECK FALL in the *HSPICE Reference Manual: Commands and Control Options*.

Edge Timing Verification

The edge condition verifies that a triggering event provokes an appropriate RISE or FALL action, within the specified time window. You use the `.CHECK EDGE` statement to verify this condition. For example,

```
.CHECK EDGE (ref RISE|FALL min max RISE|FALL)
+ node1 [node2 . . . ] [(hi lo hi_th low_th)]
```

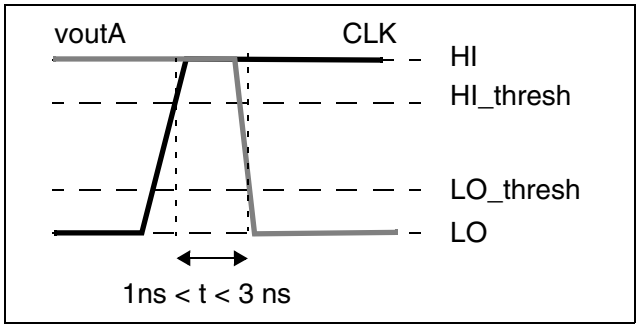


Figure 32 EDGE Example

For syntax and description of this statement, see .CHECK EDGE in the *HSPICE and HSPICE RF Command Reference*.

Setup and Hold Verification

You use the `.CHECK SETUP` and `.CHECK HOLD` statements to ensure that specified signals do not switch for a specified period of time. For example,

```
.CHECK SETUP (ref RISE|FALL duration RISE|FALL) node1
+[node2 . . . ] [(hi lo hi_th low_th)]
.CHECK HOLD (ref RISE|FALL duration RISE|FALL) node1
+[node2 . . . ] [(hi lo hi_th low_th)]
```

- For a SETUP condition, this is the minimum time before the triggering event, during which the specified nodes cannot rise or fall.

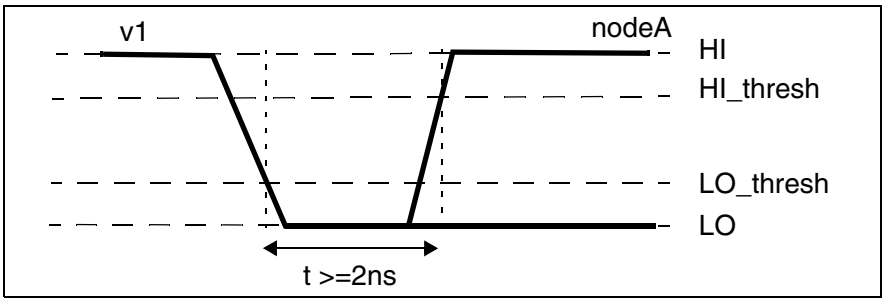


Figure 33 SETUP Example

For syntax and description of this statement, see .CHECK SETUP in the *HSPICE Reference Manual: Commands and Control Options*.

- For a HOLD condition, this is minimum time required after the triggering event, before the specified nodes can rise or fall.

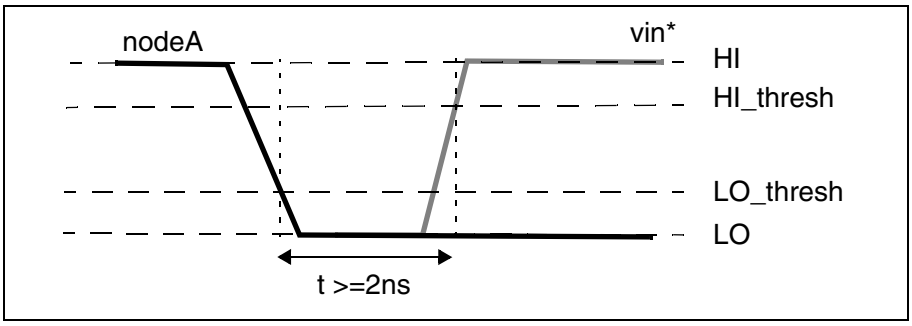


Figure 34 HOLD Example

For syntax and description of this statement, see .CHECK HOLD in the *HSPICE and HSPICE RF Command Reference*.

IR Drop Detection

You use the `.CHECK IRDROP` statement to verify that the IR drop does not exceed, or does not fall below, a specified value for a specified duration. For example,

```
.CHECK IRDROP ( volt_val time ) node1 [node2 . . . ]
+ [( hi lo hi_th low_th )]
```

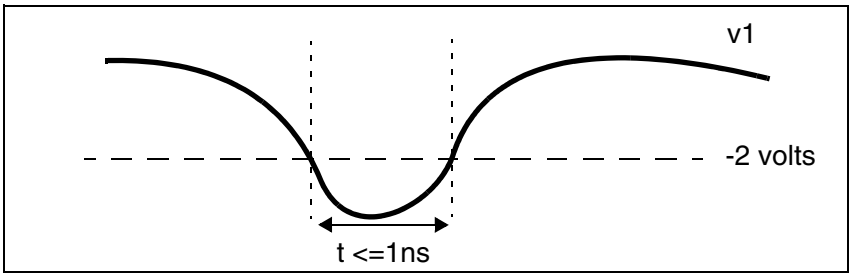


Figure 35 IR Drop Example

For syntax and description of this statement, see .CHECK IRDROP in the *HSPICE Reference Manual: Commands and Control Options*.

POWER DC Analysis

You use the `.POWERDC` (standby current) statement to calculate the DC leakage current of a design hierarchy. For example,

```
.POWERDC keyword subckt_name1...
```

This statement creates a table that lists the measurements of the AVG, MAX, and MIN values for the current of every instance in the subcircuit. This table also lists the sum of the power of each port in the subcircuit.

You use the `SIM_POWERDC_HSPICE` option to increase the accuracy of operating point (OP) calculations.

Or for even higher accuracy in operating point calculations, you use the `SIM_POWERDC_ACCURACY` option.

For syntax and description of this statement and options, see .POWERDC, .OPTION SIM_POWERDC_ACCURACY, or .OPTION

SIM_POWERDC_HSPICE in the *HSPICE Reference Manual: Commands and Control Options.*

Power DC Analysis Output Format

```
*** Leakage Current Result ***
Subckt Name=XXX
Instance Name       Port   Max(A)   Min(A)   Avg(A)
Total Power          Max(W)   Min(W)   Avg(W)
NOTE:    Power=Sum{Ii * Vi}
Subckt Name=XXX
Instance Name       Port   Max(A)   Min(A)   Avg(A)
Total Power          Max(W)   Min(W)   Avg(W)
```

Example

```
.global vdd vss
.powerdc all
x1 in1 mid1 inv
x2 mid1 out1 inv
.subckt inv in out
mn out in vss vss nch
mp vdd in out vdd pch
.ends
.end
```

(Output)

```
*** Leakage Current Result ***
Subckt Name=Top Level
Instance Name      Port   Max(A)   Min(A)   Avg(A)
   x1   in      .......
   x1   out      .......
   x2   in      .......
   x2   out      .......
   Total Power         .......
Subckt Name=inv
Instance Name      Port   Max(A)   Min(A)   Avg(A)
   mn   d      .......
   mn   g      .......
   mn   s      .......
   mn   b      .......
   mp   d      .......
   mp   g      .......
   mp   s      .......
   mp   b      .......
   Total Power         .......
```

POWER Analysis

The .POWER statement in HSPICE RF creates a table, which by default contains the measurements for AVG, RMS, MAX, and MIN for every signal specified. For example,

```
.POWER signals [REF=vname FROM=start_time TO=end_time]
```

By default, the scope of these measurements are set from 0 to the maximum timepoint specified in the .TRAN statement.

For syntax and description of .POWER statement, see .POWER in the *HSPICE and HSPICE RF Command Reference*.

Example 1

In this example, no simulation start and stop time is specified for the x1.in signal, so the simulation scope for this signal runs from the start (0ps) to the last .tran time (100ps).

```
.power x1.in
.tran 4ps 100ps
```

Example 2

You can use the FROM and TO times to specify a separate measurement start and stop time for each signal. In this example:

- The scope for simulating the x2.in signal is from 20ps to 80ps.
- The scope for simulating the x0.in signal is from 30ps to 70ps.

```
.param myendtime=80ps
.power x2.in REF=a123 from=20ps to=80ps
.power x0.in REF=abc from=30ps to='myendtime - 10ps'
```

Setting Default Start and Stop Times

In addition to using FROM and TO times in a .POWER statement, you can also use the SIM_POWERSTART and SIM_POWERSTOP options with .POWER statements to specify default start and stop times for measuring signals during simulation. These times apply to all signals that do not have their own defined FROM and TO measurement times. For example,

```
.OPTION SIM_POWERSTART=time
.OPTION SIM_POWERSTOP=time
```

These options control the power measurement scope; the default is for the entire run.

For syntax and description of these options, see .OPTION SIM_POWERSTART or .OPTION SIM_POWERSTOP in the *HSPICE and HSPICE RF Command Reference*.

Controlling Power Analysis Waveform Dumps

You use the `SIM_POWERPOST` option to control power analysis waveform dumping. For example,

```
.OPTION SIM_POWERPOST=ON|OFF
```

Considering the potentially enormous number of signals, there is no waveform dumping by default for the signals in the `.POWER` statement. Setting `SIM_POWERPOST=ON` turns on power analysis waveform dumping.

Detecting and Reporting Surge Currents

The `.SURGE` statement in HSPICE RF automatically detects and reports a current surge that exceeds the specified surge tolerance. For example,

```
.SURGE surge_threshold surge_width node1 [node2 .... noden]
```

This statement reports any current surge that is greater than *surge_threshold* for a duration of more than *surge_width*.

For additional information, see .SURGE in the *HSPICE Reference Manual: Commands and Control Options*.

Index

A

accuracy control 314
algorithm
  linear acceleration 308
  nonlinear perturbation 167
  numerical integration 313, 315
  periodic AC 167
amplifier 20, 25
amplifier, IP3 29
amplitude modulation (AM) 173
AM-PM separation 173
analysis
  phase noise 161
  time domain steady-state 131
autocorrelation function 235
autocorrelation function, jitter measurement 247

B

Backward-Euler
  algorithm 313, 315
  integration 313, 315
block elements 71
broadband phasenoise 168
broadband phasenoise algorithm 168

C

.CHECK EDGE statement 338
.CHECK FALL statement 338
.CHECK GLOBAL_LEVEL statement 336
.CHECK HOLD statement 338
.CHECK IRDROP statement 340
.CHECK RISE statement 337
.CHECK SETUP statement 338
.CHECK SLEW statement 337
choke elements 71
circuit description syntax 14
clock source, random jitter 102
CMOS GPS VCO 41
Colpitts oscillator 37
.command
  .PRINT ENV 273
command
  .PROBE ENV 273
commands
  hspicerf 13
  PTDNOISE 218
comparing results 53
config file
  hspicerf 323
configuration file 323
  example 326
configuration options
  flush_waveform 324
  ground_floating_ node 324
  hier_delimiter 324
  html 324
  integer_node 324
  max_waveform_size 325
  negative_td 325
  port_element_ voltage_ matchload 325
  rcxt_divider 325
  unit_atto 326
  v_supply 326
  wildcard_left_range 326
  wildcard_match_all 326
  wildcard_match_one 326
  wildcard_right_range 326
Custom WaveView 16

D

DC block elements 71
demo files
  67
demo files, RF 67
demonstration files, RF 67
demonstration input files 67
Detailed Standard Parasitic Format *See* DSPF
device model cards 33
DSPF
  expansion 288
  file structure 282

E

edge condition 338
element parameters
  linear inductors 70
.ENV statement 270
Envelope Analysis (ENV) 269
envelope simulation 269
.ENVFFT 272
.ENVFFT statement 272
.ENVOSC 271
.ENVOSC statement 271
ergodic simulations 238
example
  configuration file 326
examples, RF tutorials 19
Extended output variables 329

F

fall time
  verification 338
files
  .hl# 263
  hspicerf 323
  .ls# 268
  .p2d# 268
  .printhl# 263
  .printls# 268
  .printss# 268
  .ss# 268
files, output 14
flags 323
flush_waveform configuration option 324
format
  output
    DSPF 281
format, output
  NW 318
  WDB 318
Foster pole-residue form
  E element 91
  G element 91
frequency-dependent noise sorces, modeling 238

G

generating output 14
ground_floating_ node configuration option 324

H

Harmonic Balance (HB) 107
  analysis spectrum 111
  equations 109
  errors 128
  options 113
  oscillator analysis 144
  output 116
  syntax 110
  warnings 128
.HB
  for HBLIN 258
HB analysis
  IP3 amplifier 29
  power amplifier 25
HB_GIBBS option 122
HBAC 52, 189
  errors 194, 199, 252
  example 52
  output 191, 273
  output data files 193
  syntax 191
  warnings 194, 199, 252
HBAC analysis
  mixer 50
.HBLIN 256, 259
  limitations 257
  output syntax 262
.HBLSP 263
  example 266
  input syntax 265
  limitations 264
  output data files 263, 268
  output syntax 267
.HBOSC
  options 152
HBOSC analysis
  Colpitts oscillator 37
  VCO 41
.HBOSC statement 143
HBXF
  command 226
hier_delimiter configuration option 324
.hl# file 263
hold time verification 338
hspicerf command 13
hspicerf file 323

hspicerf test 324
html configuration option 324

I

ideal transformer 70
inductors
  node names 70
input files demonstration 67
input files, demo examples 67
integer_node configuration option 324
invoking HSPICE RF 13
IR drop
  checking 340

J

jitter
  random, with clock source 102
jitter measurements, transient noise 247
jitter, random, clock source 102

L

large-signal S parameter extraction 263
LIN analysis 20
linear
  acceleration 307
  matrix reduction 307
linear elements
  elements, linear 72
low noise amplifier 20
.LPRINT statement 319
.ls# file 268

M

max_waveform_size configuration option 325
.measure 274
.MEASURE ENV command 274
mixer 50
model cards 33
Monte Carlo, TRANNOISE method 235
multi-tone HB analysis
  mixer 50

N

negative_td configuration option 325
noise
  .HBNOISE 201, 209
noise parameter extraction
  small-signal 263
nonlinear perturbation algorithm 167
numerical integration 313, 315
NW output format 318

O

optimization 332
  syntax 332
.OPTION
  MAXORD 315
  PURETP 316
  SIM_ACCURACY 314
  SIM_DELTAI 321
  SIM_DELTAV 320
  SIM_DSPF 280
  SIM_DSPF_ACTIVE 280, 284
  SIM_DSPF_INSERROR 286
  SIM_DSPF_LUMPCAPS 286
  SIM_DSPF_MAX_ITER 285
  SIM_DSPF_RAIL 286
  SIM_DSPF_SCALEC 286
  SIM_DSPF_SCALER 286
  SIM_DSPF_VTOL 285
  SIM_LA 281, 307, 309
  SIM_LA_FREQ 309
  SIM_LA_MAXR 310
  SIM_LA_MINC 310
  SIM_LA_MINMODE 310
  SIM_LA_TIME 310
  SIM_LA_TOL 310
  SIM_ORDER 313, 316
  SIM_OSC_DETECT_TOL 316
  SIM_POSTAT 328
  SIM_POSTDOWN 329
  SIM_POSTSCOPE 329
  SIM_POSTSKIP 328
  SIM_POWERDC_ACCURACY 340
  SIM_POWERDC_HSPICE 340
  SIM_POWERPOST 343
  SIM_POWERSTART 342
  SIM_SPEF 280
  SIM_SPEF_ACTIVE 284
  SIM_SPEF_INSERROR 286
  SIM_SPEF_LUMPCAPS 286
  SIM_SPEF_MAX_ITER 285

SIM_SPEF_PARVALUE 287
SIM_SPEF_RAIL 286
SIM_SPEF_SCALEC 286
SIM_SPEF_SCALER 286
SIM_SPEF_VTOL 285
SIM_TG_THETA 316
SIM_TRAP 316
.OPTION HB_GIBBS 122
options, configuration file 324
oscillator
HB analysis 144
phase noise 161
oscillator example 37
output
files 14
format
DSPF 289
NW 318
tabulated data 317
WDB 318
generating 14
restricting 327

P

.p2d# file 268
periodic AC algorithm 167
periodic pime-dependent noise analysis 218
phase modulation (PM) 173
phase noise 161
phase noise analysis 161
PHASENOISE 161, 164
PHASENOISE algorithms 167
PI (linear acceleration) algorithm 309
PLL, jitter measurements 177
port_element_voltage_matchload configuration option 325
power amplifier 25
power amplifier IP3 29
.POWER statement 342
.POWERDC statement 340
.PRINT ENV command 273
.printhl# file 263
.printls# file 268
.printss# file 268
.PROBE command 273
Probing Subcircuit currents 329
PTDNOISE
input syntax 220
.MEASURE 223
output file format 222
output syntax 222
overview 218
syntax 220
PTDNOISE command 218

R

rcxt_divider configuration option 325
restricting output 327
results 53
reusing simulation output 319, 340, 342
RF
demo files 67
tutorial examples 19
RFdemo files 67
rise time
example 337
verify 337

S

S parameter extraction
large-signal 263
power-dependent 255
small-signal 263
SDE 235, 241
separating AM-PM noise 173
SETUP time verification 338
Shooting Newton
driven phase frequency circuit example 55
overview 131
ring oscillator example 62
SIM_ACCURACY option 314
SIM_ACTIVE option 280, 284, 285, 286, 287
SIM_DELTAI option 321
SIM_DELTAV option 320
SIM_DSPF option 280, 281, 313, 314, 320
SIM_DSPF_ACTIVE option 280, 284
SIM_DSPF_INSERROR option 286
SIM_DSPF_LUMPCAPS option 286
SIM_DSPF_MAX_ITER option 285
SIM_DSPF_RAIL option 286
SIM_DSPF_SCALEC option 286
SIM_DSPF_SCALER option 286

SIM_DSPF_VTOL option 285
SIM_LA option 281, 307, 309
SIM_LA_FREQ option 309
SIM_LA_MAXR option 310
SIM_LA_MINC option 310
SIM_LA_MINMODE option 310
SIM_LA_TIME option 310
SIM_LA_TOL option 310
SIM_ORDER option 313
SIM_POSTAT option 328
SIM_POSTDOWN option 329
SIM_POSTSCOPE option 329
SIM_POSTSKIP option 328
SIM_POWERDC_ACCURACY option 340
SIM_POWERED_HSPICE option 340
SIM_POWERPOST option 343
SIM_POWERSTART option 342
SIM_SPEF option 280
SIM_SPEF_ACTIVE option 284
SIM_SPEF_INSERROR option 286
SIM_SPEF_LUMPCAPS option 286
SIM_SPEF_MAX_ITER option 285
SIM_SPEF_PARVALUE option 287
SIM_SPEF_RAIL option 286
SIM_SPEF_SCALEC option 286
SIM_SPEF_SCALER option 286
SIM_SPEF_VTOL option 285
simulation engine 2
skip_nrd_nrs configuration option
  configuration options
    skip_nrd_nrs 325
slew rate
  example 337
  verification 337
small-signal noise parameter extraction 263
small-signal S parameter extraction 263
SN steady-state time domain analysis 131
SNAC
  input syntax 195
  output data files 198
SNFT 137
SNNOISE 209
  input syntax 211
  output data files 215
SNOSC 158
SNXF 230
spur (spurious) signals 163
.ss# file 268
starting
  hspicerf 13
statement 271, 272
  .ENV 270
  .HBOSC 143
statements
  .CHECK EDGE 338
  .CHECK FALL 338
  .CHECK GLOBAL_LEVEL 336
  .CHECK HOLD 338
  .CHECK IRDROP 340
  .CHECK RISE 337
  .CHECK SETUP 338
  .CHECK SLEW 337
  .HBXF 226
  .LPRINT 319
  .POWER 342
  .POWERDC 340
  .PTDNOISE 218
  .SURGE 343
  .TRAN 342
steady state time domain analysis, Shooting Newton 131
Stochastic Differential Equation, TRANNOISE method 235
strobejitter 223
subcircuit
  probing currents 329
.SURGE statement 343

T

tabulated data output 317
TIE 235
TIE, jitter mreasurement 247
time domain steady state analysis 131
time interval error (TIE) 235
.TRAN statement 342
TRANNOISE
  jitter measurements 235
  Monte Carlo method 239
  SDE method 241
transformer, ideal 70
transient noise jitter measurements 247
Trapezoidal (TRAP) integration algorithm 313, 315
tutorial 19

overview 2
simulation engine 2
two-tone HB 51

U

unit_atto configuration option 326

V

v_supply configuration option 326
VCD format 318
VCO 41
vector-modualted RF 92
vector-modulated RF
E element 98
F element 98
G element 98
H element 98
I element 94
implementation 92
V element 94
Verilog-A support 6
VMRF, See vector-modulated RF 92

W

waveform viewing 16
WaveView 16
WDB format 318
wildcard uses 327
wildcard_left_range configuration option 326
wildcard_match_all configuration option 326
wildcard_match_one configuration option 326
wildcard_right_range configuration option 326

X

X() variable 329