cadence™

Virtuoso® Spectre® Circuit Simulator and Accelerated Parallel Simulator RF Analysis User Guide

Product Version 12.1
October 2012

© 1994–2012 Cadence Design Systems, Inc. All rights reserved.

Printed in the United States of America.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

MMSIM contains technology licensed from, and copyrighted by: C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh Â© 1979, J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson Â© 1988, J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling Â© 1990; University of Tennessee, Knoxville, TN and Oak Ridge National Laboratory, Oak Ridge, TN Â© 1992-1996; Brian Paul Â© 1999-2003; M. G. Johnson, Brisbane, Queensland, Australia Â© 1994; Kenneth S. Kundert and the University of California, 1111 Franklin St., Oakland, CA 94607-5200 Â© 1985-1988; Hewlett-Packard Company, 3000 Hanover Street, Palo Alto, CA 94304-1185 USA Â© 1994, Silicon Graphics Computer Systems, Inc., 1140 E. Arques Ave., Sunnyvale, CA 94085 Â© 1996-1997, Moscow Center for SPARC Technology, Moscow, Russia Â© 1997; Regents of the University of California, 1111 Franklin St., Oakland, CA 94607-5200 Â© 1990-1994, Sun Microsystems, Inc., 4150 Network Circle Santa Clara, CA 95054 USA Â© 1994-2000, Scriptics Corporation, and other parties Â© 1998-1999; Aladdin Enterprises, 35 Efal St., Kiryat Arye, Petach Tikva, Israel 49511 Â© 1999 and Jean-loup Gailly and Mark Adler Â© 1995-2005; RSA Security, Inc., 174 Middlesex Turnpike Bedford, MA 01730 Â© 2005

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522. All other trademarks are the property of their respective holders.

Restricted Permission: This publication is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this publication, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used only in accordance with a written agreement between Cadence and its customer.
2. The publication may not be modified in any way.
3. Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.
4. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

Patents: Cadence Product described in this document, is protected by U.S. Patents 5,610,847; 5,790,436; 5,812,431; 5,859,785; 5,949,992; 5,987,238; 6,088,523; 6,101,323; 6,151,698; 6,181,754; 6,260,176; 6,278,964; 6,349,272; 6,374,390; 6,493,849; 6,504,885; 6,618,837; 6,636,839; 6,778,025; 6,832,358; 6,851,097; 7,035,782; 7,085,700

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor

Contents

Preface . . . 27
Using License Queuing . . . 29
Suspending and Resuming Licenses . . . 29
Related Documents for Spectre . . . 29
Third Party Tools . . . 30
Typographic and Syntax Conventions . . . 30

1
Introduction . . . 33
Simulation Basics . . . 33
Fundamentals of RF Simulation . . . 33
Shooting Newton and Harmonic Balance algorithm . . . 33
Shooting Method . . . 34
Harmonic Balance Method . . . 34
SpectreRF Analyses . . . 34
Periodic Analyses . . . 34
Quasi-Periodic Analyses . . . 36
Envelope Analysis . . . 38
APS and Parasitic Reduction . . . 38
Co-simulation with Virtuoso AMS Designer . . . 39
Large vs. Small Signal Analysis . . . 40

2
Setting up a SpectreRF Simulation . . . 43
Using Virtuoso ADE . . . 43
A Testbench . . . 43
ADE . . . 43
Analysis form . . . 44
Direct Plot form . . . 44
Netlist Driven Flows . . . 44

3
Frequency Domain Analyses: Harmonic Balance 47
Overview of Simulation Capabilities 47
Large-Signal Harmonic Balance Overview 47
Example 49
Harmonic Balance Solves For Cosines 56
Setting Harmonics Automatically 60
Setting Harmonics Manually 61
Oversample Factor 62
Trading off Harmonics and Oversample Factor 68
Two Input Frequencies 70
Diamond Cut 71
Three Input Frequencies 73
Diamond Cut With Three Frequencies 74
Funnel Cut 75
Axis Cut 77
Aliasing 78
Sine Representation of Square Wave 85
Convergence 87
Errpreset 87
Iteration Limit 88
Number of Non-Sinusoidal Sources 90
Harmonic Balance Starting Point 90
More Capabilities 91
Save and Recover 91
Writehb and Readhb 92
Sweeps and Restart 92
Itres 93
Freqdivide 93
Oscillators 93
Pinnode 94
Probe-Based Method 95
Semi-Autonomous 95
Things to try to Achieve Convergence (Driven Circuits) 95
Things to try to Achieve Convergence (Oscillators) 96

Implementation in ADE . . . 98
Setting Frequencies, Harmonics, and Oversample . . . 98
Diamond Cut . . . 102
Funnel Cut . . . 103
Sweeps . . . 104
Freqdivide . . . 105
Oscillator Additions . . . 106
Probe-Based Method . . . 109
Semi-Autonomous . . . 111
Commonly Used Harmonic Balance Options (Driven Circuits and Oscillators) . . . 116
Convergence tab . . . 117
Accuracy Tab . . . 120
Output Tab . . . 121
Reuse Tab . . . 123
Misc Tab . . . 131
Oscillator Options . . . 133
Key Outputs in the Spectre Window . . . 134
Examples . . . 143
ADE Setup for 1-tone Harmonic Balance (hb) Analysis . . . 146
Plotting Spectral Content . . . 150
Plotting dBm . . . 151
Sweeps . . . 154
Plotting Currents . . . 154
Loadpull . . . 157
Power-Added Efficiency . . . 170
Oscillators . . . 172
Two Input Frequencies . . . 182
Compression Point . . . 192
IFFT . . . 195
Compression Vs Pout . . . 199
Example Using the Harmonic Port with Harmonic Balance . . . 200
Three Tone Harmonic Balance . . . 210
Large-Signal IP3 . . . 214
Multi-Sinusoid hb Analysis . . . 218
Harmonic Balance AC Analysis (hbAC) . . . 227
Small-signal Versus Large-Signal Analysis for Conversion Gain . . . 227

Example: Conversion Gain (Down conversion) . . . 228
Example: Conversion Gain (Up Conversion) . . . 236
Overview of Simulation Capabilities . . . 238
ADE Implementation . . . 247
Circuit, Input port Setting, and ADE Setup for all the Examples in This Section . . . 247
Using hbAC Analysis for Conversion Gain Measurement . . . 250
Commonly Used hbAC Options . . . 258
hbAC With Multiple hb Inputs . . . 263
Rapid IP3/IP2 . . . 273
Compression Distortion Summary . . . 278
IM2 Distortion Summary . . . 284
Modulated hbAC Analysis for an Oscillator . . . 286
Sampled hbAC Analysis for a Mixer . . . 293
Harmonic Balance Noise Analysis (hbnoise) . . . 301
Small-Signal Versus Large-Signal Analysis for Noise . . . 301
Overview of Simulation Capabilities . . . 302
Example . . . 303
Noise Output Near Zero Frequency . . . 309
Noise Output Near The First Harmonic . . . 311
Frequency Sweep . . . 311
Maximum Sideband . . . 312
Setting Harmonics and Sidebands . . . 312
Noise Separation . . . 312
Multiple hbnoise . . . 313
Hbnoise after Multi-Tone hb . . . 313
Modulated for Driven Circuits and Oscillators (AM and PM Noise) . . . 314
Jitter for Driven Circuits . . . 317
Timedomain for Driven Circuits . . . 318
Oscillators . . . 318
Modulated . . . 319
Timedomain . . . 319
FM Jitter . . . 319
PM Jitter . . . 320
Noise Calculations in the Simulator . . . 320
Noise Factor and Noise Figure . . . 320
Input-Referred Noise . . . 322

Noise transfer function . . . 323
Commonly Used Options . . . 323
Cyclo2txtfile Option . . . 326
Interpolation in Noise Files . . . 327
hbXF . . . 330
ADE Implementation . . . 331
Driven Circuits . . . 331
Oscillators . . . 350
Examples . . . 370
Driven Circuit Noise Setup . . . 370
Oscillators . . . 447
hbSP Analysis . . . 479
Overview of hbsp . . . 479
Example . . . 480

4
Single Input Large and Small-signal Analyses . . . 505
Overview of Simulation Capabilities . . . 505
Overview of Periodic Steady-State (pss) Analysis . . . 505
Example . . . 506
Harmonic Balance . . . 512
DC Principles . . . 517
Convergence . . . 519
Skipdc . . . 520
Transient Principles . . . 521
Convergence . . . 521
Relref . . . 521
Integration Methods . . . 526
Controlling Accuracy . . . 529
Shooting PSS . . . 533
Overview . . . 533
Oscillators . . . 534
Delay Time . . . 535
Piecewise Linear Sources for Power Supplies . . . 535
Tstab . . . 535

Beat Frequency . . . 536
Number of Harmonics . . . 536
Accuracy Defaults (errpreset) . . . 537
PSS Shooting Convergence . . . 538
ADE Implementation and Numerical Noise Floor . . . 539
Accuracy and Settings and Trade-Offs . . . 542
Which Engine Should be Selected? . . . 544
Important Outputs in the Spectre.out File . . . 545
Guidelines for Setting up Oscillators for Simulation . . . 548
Beat frequency . . . 548
Tstab . . . 548
Different ways to start the oscillator . . . 548
Saveinit- A way to save the startup waveform . . . 550
Oscillator Node- Used to estimate the frequency for the first pss iteration . . . 550
ADE Implementation . . . 551
Spectre Output File (Oscillators) . . . 554
PSS Options . . . 556
Convergence tab . . . 557
Accuracy Tab . . . 560
Output Tab . . . 576
Reuse Tab . . . 584
Misc Tab . . . 595
Help for Convergence Issues . . . 599
Examples . . . 602
Pss shooting with a single input to an amplifier or the LO applied in a mixer . . . 602
Plotting the output spectrum in dBm . . . 607
Same Example Using Harmonic Balance . . . 609
Loadpull . . . 611
Two Input Frequencies . . . 641
Sweeps . . . 645
Compression Point . . . 649
Voltage Gain . . . 651
Power Gain . . . 654
Power-Added Efficiency . . . 656
Compression Versus Output Power . . . 659
Power Gain Versus Output Power . . . 660

Plotting THD . . . 662
Large-signal IP3 . . . 664
Oscillator Simulation . . . 672
Plotting the Oscillator Startup Waveform . . . 675
Plotting the Oscillator Output Spectrum . . . 678
Additions for Oscillator Swept Tuning Voltage . . . 679
Plotting Output Power Versus Tuning Voltage . . . 683
Plotting Output Power Versus Output Frequency . . . 685
Plotting the Output Frequency Versus the Tuning Voltage . . . 688
Plotting the Modulation Sensitivity . . . 691
Periodic AC Analysis (PAC) . . . 695
Small-signal Versus Large-Signal Analysis for Conversion Gain . . . 695
Example: Conversion Gain (Down Conversion) . . . 696
Example: Conversion Gain (Up Conversion) . . . 704
PAC General Principles . . . 707
Overview of Simulation Capabilities . . . 712
ADE Implementation . . . 724
Circuit, Input port Setting, and ADE Setup for all the Examples in This Section . . . 724
Using PAC Analysis for Conversion Gain Measurement . . . 727
Sampled Circuits and Switched-capacitor Filters . . . 735
Commonly Used PAC Options . . . 736
PAC With Multiple PSS Inputs . . . 740
Rapid IP3/IP2 . . . 748
Compression Distortion Summary . . . 754
IM2 Distortion Summary . . . 762
Modulated PAC Analysis . . . 764
Sampled PAC Analysis . . . 772
Periodic Noise Analysis (Pnoise) . . . 780
Small-Signal Versus Large-Signal Analysis for Noise . . . 780
Overview of Simulation Capabilities . . . 781
Example . . . 782
Principles of Pnoise . . . 790
Noise Type=sources . . . 791
Noise Output Near The First Harmonic . . . 792
Frequency Sweep . . . 792
Maximum Sideband . . . 793

Full-Spectrum Pnoise . . . 794
Setting Harmonics and Sidebands . . . 795
Noise Separation . . . 795
Multiple Pnoise . . . 796
Noisetype = Modulated for Driven Circuits and Oscillators (AM and PM Noise) . . . 797
Jitter for Driven Circuits . . . 800
Timedomain for Driven Circuits . . . 801
Oscillators . . . 801
Modulated . . . 802
Timedomain . . . 802
FM Jitter . . . 802
PM Jitter . . . 803
Noise Calculations . . . 803
Phase Noise (Driven Circuit) . . . 803
Noise Factor and Noise Figure . . . 803
Input-Referred Noise . . . 807
Noise transfer function . . . 807
Commonly Used Pnoise Options . . . 808
Interpolation in Noise Files . . . 809
ADE Implementation . . . 820
General Notes . . . 820
Driven Circuits . . . 820
Oscillators . . . 841
Examples . . . 862
Driven Circuit Noise Setup . . . 862
Multiple pnoise . . . 905
Oscillators . . . 944
Periodic Stability Analysis (PSTB) . . . 1006
Parameters for PSTB Analysis . . . 1006
Sweep . . . 1008
Understanding Loop-Based and Device-Based Algorithms . . . 1008
PSTB Analysis . . . 1010
Periodic Transfer Function Analysis (PXF) . . . 1028
Parameters for PXF Analysis . . . 1030
Output Parameters . . . 1033
Probe Parameters . . . 1034

Output Parameters 1035
Modulation Parameters 1035
Sampled Small-Signal Analysis 1035
Swept PXF Analysis 1036
Conversion Gain Measurement 1037
Commonly Used Options 1039
Measuring AM to PM Conversion 1047
Periodic S-parameter Analysis (PSP) 1052
Parameters for PSP Analysis 1056
Setting up a PSP simulation - Input Frequency Range 1058
Setting up PSP - Output Frequency Range 1062
Commonly Used Options 1065

5
Multiple Input Large and Small-signal Analyses 1075

Quasi-Periodic Steady-State Analysis (QPSS) 1075
Comparing QPSS Analysis with PSS and PAC Analyses 1076
QPSS Parameters 1078
The errpreset Parameter in QPSS Analysis 1085
The Compression Parameter in QPSS Analysis 1086
Input source settings for QPSS 1087
Commonly Used QPSS Options 1093
QPSS Small Signal Analyses 1114
Quasi-Periodic Noise Analysis (QPnoise) 1114
QPnoise Output 1114
QPnoise Parameters 1120
Setting up the QPnoise analysis 1124
Quasi-Periodic AC Analysis (QPAC) 1142
QPAC Output Frequency and Sideband Vectors 1142
QPAC Parameters 1144
Input source settings for QPAC 1147
Quasi-Periodic S-parameter Analysis (QPSP) 1160
QPSP Output Frequencies and Sideband Vectors 1160
Input and Output Frequencies in QPSP 1162
Noise Analysis with QPSP 1165

Swept QPSP Analysis 1167
QPSP Parameters 1167
QPSP Setup 1170
Plotting S-parameter results. 1174
Plotting noise figure 1177
Quasi-Periodic Transfer Function Analysis (QPXF) 1179
QPXF Output Frequencies and Sideband Vectors 1179
Transfer Function Inputs 1182
Swept QPXF Analysis 1182
QPXF Parameters 1183
Setting up QPXF Analyses 1185

6
Envelope (ENVLP) Analysis 1191
Example 1192
Envelope Principles 1204
Setting up the Envelope Analysis in ADE 1217
Frequency Modulated Input Signals 1244
Commonly Used Options 1250
Autonomous ENVLP Analysis (Oscillators) 1267

7
Large Signal S-parameter Simulation (LSSP) Wizard 1285
Large Signal S-Parameters 1285
Large-Signal S-parameters for a Two-Port Circuit 1286
Measuring Large Signal S-parameters using SpectreRF 1288
References 1303

8
Simulating Noise and Jitter 1305
Analyzing Time-varying Noise 1305
Characterizing Time-Domain Noise 1305
Calculating Time Domain Noise 1308
Calculating Noise Correlation Coefficients 1311

Cyclostationary Noise Example 1312
Reference Information on Time-Varying Noise 1320
Thermal Noise 1321
Linear Systems and Noise 1327
Spectral Densities in Two Simple Circuits 1328
Time-Varying Systems and the Autocorrelation Function 1331
Time-Varying Systems and Frequency Correlations 1335
Summary 1340
Oscillator Noise Analysis 1342
Phase Noise Primer 1343
Models for Phase Noise 1346
Linear Time-Invariant (LTI) Models 1347
Linear Time-Varying (LTV) Models 1348
Amplitude Noise and Phase Noise in the Linear Model 1352
Details of the SpectreRF Calculation 1353
Calculating Phase Noise 1357
Setting Simulator Options 1357
Troubleshooting Phase Noise Calculations 1359
Known Limitations of the Simulator 1359
What Can Go Wrong 1360
Phase Noise Error Messages 1362
The tstab Parameter 1363
Frequently Asked Questions 1364
Further Reading 1369
References 1369
Measuring AM, PM and FM Conversion 1371
Derivation 1371
Positive Frequencies 1375
FM Modulation 1376
Simulation 1377
Results 1379
Conclusion 1382
References 1382

9
Noise-Aware PLL Flow . . . 1383

Noise-aware PLL Flow . . . 1383
Preparation . . . 1384
Using the Noise-Aware PLL flow in ADE . . . 1386
VCO Extraction . . . 1387
Extracting the VCO Macro-Model . . . 1389
Extracting the PFD-CP Macro-Model . . . 1413
PFD-CP Macro-model Extraction . . . 1418
Using PFD-CP macro-model . . . 1422
Divider Extraction . . . 1424
Divide Macro-Model Basis . . . 1424
Divider Macro-Model Extraction . . . 1425
Using Divider Macro-Model . . . 1427
PLL Simulation with Macro-Models . . . 1429
Sigma-Delta Modulator Macro-Model . . . 1432

10
AnalogLib Components Used in RF Simulation . . . 1435

The PORT Element . . . 1435
Capabilities of the port Component . . . 1436
Terminating the Port . . . 1436
Parameters for the Port Component . . . 1438
Port Parameters . . . 1438
Using the Harmonic Port with Harmonic Balance . . . 1439
Source type . . . 1449
DC Parameters . . . 1450
DC voltage . . . 1450
Pulse Waveform Parameters . . . 1451
PWL Waveform Parameters . . . 1465
Waveform Entry Method . . . 1467
Sinusoidal Waveform Parameters . . . 1490
Modulation Parameters . . . 1493
Display second sinusoid . . . 1508

Display multi sinusoid 1510
Exponential Waveform Parameters 1515
Bit Waveform Parameters 1517
Bit Waveform Examples 1520
PRBS Waveform Parameters 1526
Noise Parameters 1535
Small-Signal Parameters 1539
Temperature Effect Parameters 1541
Simulating Tabulated S-Parameters Using the Nport Component 1543
Convolution-Based Method 1549
Using the nport Component 1550
Controlling Rational Fit Accuracy 1564
Using the relerr and abserr Parameters 1565
Using the rational order(ratorder) Parameter 1567
Troubleshooting 1567
Assessing the Quality of the Rational Interpolation 1568
Model Reuse 1574
Dcblock, dcfeed, indq, and capq 1576
Dcfeed 1576
Dcblock 1580
indq 1583
capq 1592
Reference: S-Parameter Equations 1599
Network Parameters 1600
Equations for Network Parameters 1600
Two-Port Scalar Quantities 1602
Equations for Two-Port Scalar Quantities 1603
Two-Port Gain Quantities 1605
Equations for Two-Port Gain Calculations 1606
Two-Port Network Circles 1609
Equations for Two-Port Network Circle 1609
Equation for VSWR (Voltage Standing Wave Ratio) 1612
Equation for GD (group delay) 1613
References 1613

11 Cosimulation with MATLAB® and Simulink® 1615

Introduction to Cosimulation with MATLAB 1616
Software Requirements 1616
Setting Up and Running a Cosimulation 1616
Connecting the Coupler Block Into the System-Level Simulink Schematic 1617
Determining How You Want to Start and Run the Cosimulation 1621
Generating a Netlist for the Lower-Level Block 1621
- Preparing the Netlist When Using ADE 1621
- Preparing the Netlist Without Using a Graphical User Interface 1628

Running the Cosimulation 1630
- Starting the Two Applications Separately 1630
- Starting SpectreRF Manually and MATLAB Automatically 1631
- Starting MATLAB Manually and SpectreRF Automatically 1631

MATLAB Support Matrix 1632

A Design Workshops 1635

Setting Up Environment Variables and the Path Statement 1635
Using SpectreRF from the MMSIM Hierarchy 1635
Accessing the Most Current SpectreRF Documentation 1636
Creating a Local Editable Copy of the rfExamples Library 1636
the Cadence Libraries 1637
- Using the Library Path Editor 1637
- Using a UNIX Shell Window 1638

Setting Up For Simulation 1639
- Opening a Circuit in the Schematic Window 1639
- Choosing Simulator Options 1642
- Specifying Outputs to Save 1643
- Setting Up Model Libraries 1644
- Editing Design Variable Values 1645

Low-noise Amplifiers 1649
Analyses and Measurement Examples in this Chapter 1649
Simulating the lnaSimple Example 1649

Opening the InaSimple Circuit in the Schematic Window 1649
Choosing Simulator Options 1652
Setting Up Model Libraries 1654
Calculating Voltage Gain with PSS 1654
Setting Up the Simulation 1654
Editing the Schematic 1655
Setting Up the PSS Analysis 1656
Running the Simulation 1659
Plotting Voltage Gain 1659
Calculating Output Voltage Distribution with PSS 1662
Setting Up the Simulation 1663
Editing the Schematic 1663
Setting up the PSS Analysis 1665
Running the Simulation 1666
Plotting the Output Voltage Distribution 1666
S-parameter Analysis for Low Noise Amplifiers 1668
Setting Up the Simulation 1668
Editing the Schematic 1669
Setting up the S-parameter Analysis 1669
Running the Simulation 1671
Plotting S-Parameters 1671
Plotting the Voltage Standing Wave Ratio 1675
Linear Two-Port Noise Analysis with S-Parameters 1676
Setting Up the Simulation 1676
Editing the Schematic 1677
Setting up the S-parameter Analysis 1679
Running the Simulation 1681
Plotting the Noise Figure and Minimum Noise Figure 1682
Plotting the Equivalent Noise Resistance 1685
Plotting Load and Source Stability Circles 1687
Plotting the Noise Circles 1691
Noise Calculations with PSS and Pnoise 1694
Setting Up the Simulation 1694
Editing the Schematic 1695
Setting up the PSS and Pnoise Analyses 1697
Running the Simulation 1702

Plotting the Noise Calculations 1702
Printing the Noise Summary 1704
Plotting the 1dB Compression Point 1707
Setting Up the Simulation 1707
Editing the Schematic 1707
Setting up the PSS Analysis 1710
Running the Simulation 1713
Plotting the 1dB Compression Point 1713
Calculating the Third-Order Intercept Point with Swept PSS 1717
Setting Up the Simulation 1717
Editing the Schematic 1717
Setting up the Swept PSS Analysis 1718
Running the Simulation 1720
Plotting the Third-Order Intercept Point 1721
Calculating Conversion Gain and Power Supply Rejection with PSS and PXF 1724
Setting Up the Simulation 1724
Editing the Schematic 1725
Setting up the PSS and PXF Analyses 1725
Running the Simulation 1728
Plotting Conversion Gain and Power Supply Rejection 1729
Oscillators 1732
Phases of Autonomous PSS Analysis 1732
Phase Noise and Oscillators 1733
Starting and Stabilizing Oscillators 1733
The tline3oscRF Oscillator Circuit 1734
Simulating the tline3oscRF Oscillator Circuit 1734
Periodic Steady State and Phase Noise with PSS and Pnoise 1738
Setting Up the Simulation 1738
Setting Up the PSS Analysis 1739
Setting Up the Pnoise Analysis 1741
Running the Simulation 1744
Plotting the Fundamental Frequency 1745
Plotting the Periodic Steady State Solution 1748
Plotting the Phase Noise 1751
The oscDiff Circuit: A Balanced, Tunable Differential Oscillator 1754
Simulating the oscDiff Circuit 1755

Opening the oscDiff Circuit in the Simulation Window 1755
Choosing Simulator Options 1757
Setting Up Model Libraries 1758
Fundamental Frequency, Output Noise, and Phase Noise with PSS and Pnoise 1759
Setting Up the Simulation 1759
Editing Design Variables 1759
Setting Up the PSS Analysis 1760
Setting Up the Pnoise Analysis 1762
Running the Simulation 1764
Plotting the Fundamental Frequency 1765
Plotting the Output Noise and Phase Noise 1767
The Van der Pol Circuit: Measuring AM and PM Noise Separation 1771
Simulating the vdp_osc Circuit 1771
Opening the vdp_osc Circuit 1771
Editing Properties for the Inductor 1773
Opening the Simulation Window 1775
Measuring AM and PM Conversion with PSS and Pnoise 1776
Setting Up the Simulation 1777
Setting Up the PSS Analysis 1777
Setting Up the Pnoise Analysis 1780
Running the Simulation 1784
Measuring Jitter with PSS and Pnoise Jitter Analyses 1791
Opening the oscDiff Circuit 1792
Setting Up the Simulation 1796
Specifying Outputs to Save 1796
Setting Up the PSS Analysis 1797
Setting Up the Pnoise Analysis 1800
Running the Simulation 1803
Measuring Jitter 1804
Troubleshooting for Oscillator Circuits 1812
Mixers 1814
The ne600p Mixer Circuit 1815
Setting Up to Simulate the ne600p Mixer 1817
Opening the ne600p Mixer Circuit in the Schematic Window 1817
Choosing Simulator Options 1819
Setting Up Model Libraries 1821

Setting Design Variables 1822
Total Harmonic Distortion Measurement with PSS 1823
Setting Up the Simulation 1823
Editing the Schematic 1824
Setting Up the PSS Analysis 1826
Running the Simulation 1828
Plotting and Calculating Total Harmonic Distortion 1828
Compression Distortion Summary with PSS and PAC 1832
Setting Up the Simulation 1833
Editing the Schematic 1834
Setting up the PSS and PAC Analyses 1835
Setting Up the PAC Analysis 1837
Running the Simulation 1840
Printing the Compression Distortion Summary 1841
Rapid IP3 Measurement with PSS and PAC 1841
Setting Up the Simulation 1842
Editing the Schematic 1843
Setting up the PSS and PAC Analyses 1844
Plotting the Rapid IP3 Curve 1849
Noise Figure Measurement with PSS and Pnoise 1850
Setting Up the Simulation 1851
Editing the Schematic 1852
Setting up the PSS and Pnoise Analyses 1852
Running the Simulation 1856
Plotting the Noise Figure 1857
Noise Figure Measurement and Periodic S-parameter Plots with PSS and PSP 1859
Setting Up the Simulation 1860
Editing the Schematic 1861
Setting up the PSS and PSP Analyses 1862
Running the Simulation 1871
Plotting the Noise Figure 1871
Plotting Periodic S-Parameters 1873
Conversion Gain and Power Supply Rejection with PSS and PXF 1877
Setting Up the Simulation 1877
Editing the Schematic 1877
Setting Up the PSS and PXF Analyses 1878

Running the Simulation 1882
Plotting the Conversion Gain 1882
Plotting the Power Supply Rejection 1884
Calculating the 1 dB Compression Point with Swept PSS 1886
Setting Up the Simulation 1886
Editing the Schematic 1886
Setting Up the Swept PSS Analysis 1886
Running the Simulation 1889
Plotting the 1 dB Compression Point 1890
Third-Order Intercept Measurement with Swept PSS and PAC 1893
Setting Up the Simulation 1893
Editing the Schematic 1893
Setting Up the PSS and PAC Analyses 1895
Running the Simulation 1899
Plotting the IP3 Curve 1899
Intermodulation Distortion Measurement with QPSS 1902
Setting Up the Simulation 1902
Editing the Schematic 1902
Setting Up the QPSS Analysis 1903
Selecting Simulation Outputs 1907
Running the Simulation 1908
Plotting the Voltage and Power 1908
Noise Figure with QPSS and QPnoise 1912
Setting Up the Simulation 1913
Editing the Schematic 1913
Setting Up the QPSS and QPnoise Analyses 1914
Selecting Simulation Outputs 1920
Running the Simulation 1921
Plotting the Noise Figure 1921
Plotting the Output Noise 1923
IP3 and Compression Distortion Summary Measurements for a Mixer 1925
Measuring Rapid IP3 1927
Measuring Compression Distortion Summary 1929
Rapid IP2 and IM2 Distortion Measurements for a Gilbert Direct Conversion Mixer . . 1932
Measuring Rapid IP2 1934
Measuring IM2 Distortion Summary 1936

Rapid IP3 and Compression Distortion Summary Measurements for a Power Amplifier . . 1938
The Rapid IP3 Measurement 1941
Measuring Compression Distortion Summary 1943
Transmitters 1946
Envelope Analysis 1947
Opening the EF_example Circuit in the Schematic Window 1947
Opening the Simulation Window 1949
Setting Up the Model Libraries 1949
Editing PORT0 and PORT1 in the EF_example Schematic 1950
Setting Up an Envelope Analysis 1952
Looking at the Envelope results 1954
Following the Baseband Signal Changes Through an Ideal Circuit 1957
Following the Baseband Signal Changes Through a Non-Ideal Circuit 1962
Plotting the Complete Baseband Signal 1967
Plotting the Baseband Trajectory 1971
Measuring ACPR and PSD 1982
The ACPR Wizard 1982
Measuring ACPR 1986
Estimating PSD From the Direct Plot Form 1998
Reference Information for ACPR and PSD Calculations 2004
Measuring Load-Pull Contours and Load Reflection Coefficients 2015
Creating and Setting Up the Modified Circuit (EF_LoadPull) 2015
Setting Up and Running the PSS and Parametric Analyses 2027
Displaying Load Contours 2030
Moving to Differential Mode 2045
Using S-parameter Input Files 2045
Setting Up the EF_example Schematic for the First Simulation 2046
Adding Components to the Schematic 2050
Setting Up the sparamfirst Schematic 2053
Running the SP Simulation 2053
Setting Up and Running the Second sp Simulation 2055
Using the S-parameter Input File with a SpectreRF Envlp Analysis 2063
Measuring AM and PM Conversion with Modulated PAC, AC and PXF Analyses 2073
The Modulated Analysis Settings 2073
Creating the EF_AMP Circuit 2074

Setting Up the EF_AMP Circuit 2082
Edit the Variable Values 2083
Selecting Outputs To Save 2085
Setting Up and Running the PSS, PAC Modulated, and PXF Modulated Analyses 2088
Running the Simulations 2099
Plotting and Calculating PAC Modulated Results 2099
Plotting and Calculating PXF Modulated Results 2104
Measuring Jitter with PSS and Pnoise Analyses 2108
Setting Up the EF_AMP Circuit 2108
Opening the EF_AMP Circuit in the Schematic Window 2109
Opening the Simulation Window for the EF_AMP Circuit 2110
Setting Up and Running the PSS and Pnoise Analyses 2111
Running the Simulations 2117
Plotting the Jitter Measurement 2117

B
Top-down RF Design Methodology 2129

Top-Down Design of RF Systems 2130
Use Model for Top Down Design 2131
Baseband Modeling 2134
Example Comparing Baseband and Passband Models 2136
rfLib Library Overview 2148
Use Model and Design Example 2152
Opening a New Schematic Window 2153
Opening the Analog Environment 2154
Constructing the Baseband Model for the Receiver 2154
Setting Variable Values for the Receiver Schematic 2192
Setting Up and Running a Transient Analysis 2195
Examining the Results: Eye Diagram, Histogram, and Scatter Plot 2197
Computing Minimized RMS Noise Using the Optimizer 2209
Summarizing the Design Procedure 2226
Creating a Passband View of the Architectural Model 2226
Comparing Baseband and Passband Models 2230
Relationship Between Baseband and Passband Noise 2233
Introduction to Analysis 2234

Preparation Steps for Analyses 2235

C
Documents That Ship in the Software Hierarchy 2243

SpectreRF_simulink_example.pdf 2244
EnvelopeAN.pdf 2244
LSSP_AN.pdf 2245
MatlabWorkshop.pdf 2245
PLL_Jitter_AN.pdf 2246
NS_AN.pdf 2246
PSRR_Drv_AN.pdf 2246
readme.txt 2247
HB_AN.pdf 2247
LTJM_AN.pdf 2247
PerturbationAN.pdf 2248
PSRR_Osc_AN.pdf 2248
RF_Blocks_AN.pdf 2248
JitterAN.pdf 2249
MatlabAN.pdf 2249
PstbAN.pdf 2250

Index 2251

Preface

Virtuoso® Spectre® circuit simulator or Virtuoso® Accelerated Parallel Simulator (APS) base products with the Virtuoso MMSIM RF analysis option provide simulation capabilities for RFIC designers. The simulators:

- Support efficient calculation of the operating point, transfer function, noise, and signal distortion of common RF and communication circuits, such as mixers, LNA, oscillators, sample and holds, and switched capacitor filters.
- Support a multi-technology simulation (MTS) mode that enables the simulation of systems consisting of blocks designed with different processes, such as RF System-in-Package (SIP).

This user guide assumes that you are familiar with:

- RF circuit design
- SPICE simulation
- The Virtuoso® Analog Design Environment (ADE)

The Virtuoso® Spectre® Simulator, the high performance Virtuoso® APS, and the Virtuoso MMSIM RF analysis options are part of the MMSIM (multi-mode simulation) portfolio and are accessible using individual a la carte licenses or MMSIM tokens. Note that mixing of tokens and a la carte license is not allowed. Table -1 on page 27 and Table -2 on page 28 show the capabilities offered with the base products Virtuoso Spectre simulator or Virtuoso APS and the Virtuoso MMSIM RF analysis option.

Table -1 Virtuoso Spectre® Simulator with Virtuoso MMSIM RF Analysis Option

Features
Transient Noise Analysis
DC, small-signal analyses and transient
Monte Carlo, DC mismatch, Parametric sweep
RF Harmonic Balance

Features
RF Shooting Newton
Large-signal Analysis - Periodic and Quasi-periodic (HB, PSS and QPSS)
Noise Analysis - Periodic and Quasi-periodic (pnoise, QPNoise, sampled, jitter)
Small-signal Analysis - Periodic and Quasi-periodic (PAC, PXF, PSP, QPAC, QPXF, QPSP)
Periodic Stability Analysis (PSTB)
Envelope Analysis (AM, PM, FM, Autonomous)
Rapid Distortion Analyses - Perturbation-based IP2 and IP3
Co-simulation with Simulink® from The MathWorks
MMSIM Toolbox for MATLAB® from The MathWorks

Table -2 Virtuoso® Accelerated Parallel Simulator with Virtuoso MMSIM RF Analysis Option

Features
All analysis and features in Virtuoso Spectre
RF Harmonic Balance
RF Shooting Newton
Large-signal Analysis - Periodic and Quasi-periodic (HB, PSS and QPSS)
Noise Analysis - Periodic and Quasi-periodic (pnoise, QPNoise, sampled, jitter)
Small-signal Analysis - Periodic and Quasi-periodic (PAC, PXF, PSP, QPAC, QPXF, QPSP)
Periodic Stability Analysis (PSTB)
Envelope Analysis (AM, PM, FM, Autonomous)
Rapid Distortion Analyses - Perturbation-based IP2 and IP3

Cadence offers multi-core simulation up to 64 cores for RF analysis, enabled with the base product Virtuoso APS along with Virtuoso MMSIM RF analysis and the Virtuoso MMSIM CPU Accelerator options.

Please contact your account representative for more details on the licensing and packaging.

Using License Queuing

You can turn on license queuing by using the `lqtimeout` command line option:

```
spectre +lqtimeout time_in_seconds
```

If a license is not available when you begin a simulation job, the Spectre circuit simulator and APS wait in queue for a license for the specified time. If you specify the value 0 for this option, the Spectre circuit simulator waits indefinitely for a license. The `lqtimeout` option ha s no default value for the standalone Spectre circuit simulator. If you invoke Spectre through the Analog Design Environment, the default value for `lqtimeout` is 900 seconds.

You can use the `lqsleep` option to specify the interval (in seconds) at which the Spectre circuit simulator should check for license availability. The default value for `lqsleep` is 30 seconds.

```
spectre +lqsleep interval
```

For more information on any of the above options, see `spectre -h`.

Suspending and Resuming Licenses

You can direct Spectre and APS to release licenses when suspending a simulation job. This feature is aimed for users of simulation farms, where the licenses in use by a group of lower priority jobs may be needed for a group of higher priority jobs. To enable this feature, simply start Spectre with the `+lsuspend` command line option. In the Solaris environment, press `ctrl+z` to suspend the Spectre license. All licenses are checked in. To resume simulation, press `fg`. These keystrokes may not work if you have changed the default key bindings.

For information on tracking token licensing, see the *Virtuoso® Software Licensing and Configuration Guide*.

In Virtuoso® Analog Design Environment, the `lqtimeout` and `lqsleep` options are controlled by the following options:

```
spectre.envOpts lsuspend boolean t
spectre.envOpts licQueueTimeOut string "900"
spectre.envOpts licQueueSleep string "30"
```

Related Documents for Spectre

This user guide contains information about the functionality. The following documents provide more information about the RF analysis option with Virtuoso Spectre, Virtuoso APS, and related products.

- The Spectre circuit simulator is often run within the analog circuit design environment, under the Cadence design framework II. To see how the Spectre circuit simulator is run under the analog circuit design environment, read the *Virtuoso Analog Design Environment User Guide.*
- To learn more about specific parameters of components and analyses, consult the Spectre online help (`spectre -h`).
- To learn more about the equations used in the Spectre circuit simulator, consult the *Virtuoso Simulator Components and Device Models Manual*.
- The Spectre circuit simulator also includes a waveform display tool, Virtuoso Visualization and Analysis tool, to use to display simulation results. For more information about the tool, see the *Virtuoso Visualization and Analysis User Guide*.
- For more information about using the Spectre circuit simulator with Verilog-A, see the *Verilog-A Language Reference* manual.
- For more information about RF theory, see *Virtuoso Spectre Circuit Simulator RF Analysis Theory*.
- For more information about how you work with the design framework II interface, see *Cadence Design Framework II Help*.
- For more information about specific applications of Spectre analyses, see *The Designer's Guide to SPICE & Spectre*[1].

Third Party Tools

To view any `.swf` multimedia files, you need:

- Flash-enabled web browser, for example, Internet Explorer 5.0 or later, Netscape 6.0 or later, or Mozilla Firefox 1.6 or later. Alternatively, you can download Flash Player (version 6.0 or later) directly from the Adobe website.
- Speakers and a sound card installed on your computer for videos with audio.

Typographic and Syntax Conventions

This list describes the syntax conventions used for the Spectre circuit simulator.

1. Kundert, Kenneth S. *The Designer's Guide to SPICE & Spectre.* Boston: Kluwer Academic Publishers, 1995.

`literal`	Nonitalic words indicate keywords that you must enter literally. These keywords represent command (function, routine) or option names, filenames and paths, and any other sort of type-in commands.
`argument`	Words in italics indicate user-defined arguments for which you must substitute a name or a value. (The characters before the underscore (_) in the word indicate the data types that this argument can take. Names are case sensitive.
`|`	Vertical bars (OR-bars) separate possible choices for a single argument. They take precedence over any other character.
`[ ]`	Brackets denote optional arguments. When used with OR-bars, they enclose a list of choices. You can choose one argument from the list.
`{ }`	Braces are used with OR-bars and enclose a list of choices. You must choose one argument from the list.
`...`	Three dots (...) indicate that you can repeat the previous argument. If you use them with brackets, you can specify zero or more arguments. If they are used without brackets, you must specify at least one argument, but you can specify more.

The language requires many characters not included in the preceding list. You must enter required characters exactly as shown.

1

Introduction

Simulation Basics

Fundamentally, SpectreRF provides large-signal and small-signal analyses. Large-signal analyses PSS, QPSS, HB, and Envelope solve all the equations, and capture all the harmonics that are produced by the system. Large-signal effects like compression are taken into account. Because all the nonlinearities of the system need to be solved for, the simulation time is longer than the simulation time for small-signal analyses where the large-signal effects like compression are not solved for. The nonlinearity solved for in the large-signal analysis is taken into account, so frequency translation effects caused by the large signal are present in the small-signal analyses, thus conversion gain and noise are accurate for systems that translate frequencies.

Fundamentals of RF Simulation

SpectreRF provides solutions for single or multi tone periodic systems using periodic steady-state (PSS) or quasi-periodic steady state (QPSS), and the small-signal analyses provided for all three. Noise simulations that include frequency translation are provided using Pnoise, hbnoise, or QPnoise. To measure small-signal conversion gain, pac, QPAC, pxf, and QPXF analyses are provided. To measure IP3, rapid IP3 is provided. To measure power gain, PSP and QPSP are provided. This is an extension of S-Parameters for systems that translate frequency. Non-periodic systems can be simulated using envelope

Shooting Newton and Harmonic Balance algorithm

SpectreRF provides a choice of simulation engines between the traditional *Shooting Newton method* (or simply, *Shooting method*) and the *Harmonic Balance method* (HB) with most analyses. The Harmonic Balance engine complements the capabilities of the Shooting method.

The combination of a PSS or QPSS analysis using the Shooting method with a time-varying small-signal analyses is efficient for circuits that respond in a strongly nonlinear manner to

the LO or the clock. Consequently, you can use the SpectreRF simulations with the Shooting method to simulate strongly nonlinear circuits, such as switched-capacitor filters, switching mixers, chopper-stabilized amplifiers, PLL-based frequency multipliers, and sample-and-holds. You can use the SpectreRF simulations with the Harmonic Balance method to simulate circuits that produce a reasonably small number of harmonics, such as Low-Noise Amplifiers (LNAs) or Gilbert cell mixers.

Generally, PSS is used for circuits with a single input frequency. Either Shooting or Harmonic Balance can be used. Usually QPSS or HB is used for circuits that have more than one input frequency. QPSS runs much faster and requires less memory when Harmonic Balance is used as the engine. Use Shooting only for very nonlinear circuits like samplers with the sample clock and one or more inputs applied, or switched-capacitor circuits with the switch clock and one or more inputs applied.

Shooting Method

SpectreRF has traditionally used an engine known as the *Shooting method* [kundert90] to implement periodic and quasi-periodic analyses and the envelope analysis. The Shooting method is a time domain method.

Harmonic Balance Method

The Harmonic Balance engine supports frequency domain Harmonic Balance analyses. It provides efficient and robust simulation for circuits that produce a reasonable small number of harmonics. The Harmonic Balance engine is supported on the Solaris, Linux, HP and IBM platforms for both 32 and 64 bit architectures. See *Virtuoso Spectre Circuit Simulator RF Analysis Theory* for more information on the Harmonic Balance engine.

SpectreRF Analyses

The Virtuoso® Spectre® circuit simulator RF analysis (SpectreRF) analyses add capabilities to the Virtuoso Spectre circuit simulator, such as direct, efficient computation of steady-state solutions and simulation of circuits that translate frequency. You use the SpectreRF analyses in combination with the Fourier analysis capability of the Spectre circuit simulator and with the Verilog®-A behavioral modeling language.

Periodic Analyses

SpectreRF adds periodic large and small-signal analyses to Spectre simulation.

- **Periodic Steady-State Analysis, PSS (Large-Signal)**

  PSS captures the full non-linear behavior of the circuit usually with a single input applied to the circuit. Shooting or Harmonic Balance can be used for PSS.

- **Periodic AC Analysis, PAC (Small-Signal)**

  PAC allows the small-signal conversion gain and allows the small-signal IP3 to be measured.

- **Periodic S-parameter Analysis, PSP (Small-Signal)**

  PSP allows the small-signal power conversion gain to be measured, and allows the input match at (for example) the RF and the output match at the IF to be measured in the same run.

- **Periodic Transfer Function Analysis, PXF (Small-Signal)**

  PXF can be used to measure small-signal conversion gain to be measured and also PSRR with frequency translations can be measured.

- **Periodic Noise Analysis, Pnoise (Small-Signal)**

  Pnoise allows the noise of systems that translate frequencies to be measured accurately.

- **Periodic Stability Pstb** (**Small-Signal**)

  Pstb measures the open-loop gain and phase for systems where there is a large signal applied. The large signal can cause nonlinearity which might affect the stability of feedback systems.

- **Harmonic balance analysis, HB**

  HB calculates the harmonics and mixing products for all the inputs.

- **Harmonic Balance AC analysis, HBAC**

  HBAC allows the small-signal conversion gain and the small-signal IP3 to be calculated.

- **Harmonic Balance Noise, HBnoise**

  HBnoise allows the Pnoise of systems that translate frequencies to be calculated accurately. Hbxf is provided for the measurement of conversion gain from the sources in the circuit to the output specified for the noise analysis.

- **Harmonic Balance S-Parameter, Hbsp (Small-Signal)**

  Hbsp allows the small-signal power conversion gain to be measured, and allows the input match at (for example) the RF and the output match at the IF to be measured in the same run.

For details on the periodic analyses, see *Virtuoso Spectre Circuit Simulator RF Analysis Theory*.

Periodic Steady-State (PSS) analysis is a large-signal analysis that directly computes the periodic steady-state response of a circuit. With PSS, simulation times are independent of the time constants of the circuit, so PSS can quickly compute the steady-state response of circuits with long time constants, such as high-Q filters and oscillators. You can also sweep frequency or other variables using PSS.

Generally, PSS is used for circuits that have a single input frequency. Although it can simulate circuits with multiple inputs as long as fewer than 50 periods of the highest input frequency or less are simulated, it solves for all of the harmonics produced by the system. If you have a system that has 900MHz and 1GHz, and you want the harmonics though 3.5GHz, this requires 35 harmonics of 100MHz, and is a reasonable problem for PSS to simulate. If the system had 1GHz and 1GHz plus 1MHz, and you wanted harmonics through 3.5GHz, PSS would need to solve for 3,500 harmonics of 1MHz, which would take a very long time, and lots of computer memory. In this case, use QPSS or HB because these analyses only solve for the actual mixing products that are produced by the system. This is a much smaller number of harmonics, thus the runtime is much faster, and it is just as accurate.

After completing a PSS analysis, the SpectreRF simulator can model frequency conversion effects by performing one or more of the periodic small-signal analyses, Periodic AC analysis (PAC) (caused by the nonlinearity from the large signal applied to the circuit), Periodic S-parameter analysis (PSP), Periodic Transfer Function analysis (PXF), Periodic Stability (Pstb), and Periodic Noise analysis (Pnoise). The periodic small-signal analyses are similar to the Spectre L AC, SP, XF, STB, and Noise analyses, but you can apply the periodic small-signal analyses to periodically driven circuits that exhibit frequency conversion. Examples of important frequency conversion effects include conversion gain in mixers, noise in oscillators, and filtering using switched-capacitors.

Therefore, with periodic small-signal analyses you apply a small signal at a frequency that may be non commensurate (not harmonically related) to the small signal fundamental. This small signal is assumed to be small enough so that it is not distorted by the circuit.

Quasi-Periodic Analyses

SpectreRF adds quasi-periodic large and small-signal analyses to Spectre L simulation. Quasi-periodic means that even systems with input signals that don't have a rational number as the fundamental periodicity of the system can still be simulated. If for example, if a system has an input at 1GHz and an input which is a multiple of pi, this system can still be simulated with QPSS.

- **Quasi-Periodic Steady-State Analysis, QPSS (Large-Signal)**

QPSS captures the full non-linear behavior of the circuit with two or more inputs applied to the circuit. For most systems, Harmonic Balance is more efficient.

- **Quasi-Periodic AC Analysis, QPAC (Small-Signal)**

  QPAC allows the measurement of small-signal conversion gain. QPAC takes into account all the harmonics that are calculated by the QPSS analysis.

- **Quasi-Periodic S-parameter Analysis, QPSP (Small-Signal)**

  QPSP allows the small-signal power conversion gain to be measured, and allows the input match at (for example) the RF and the output match at the IF to be measured in the same run. It also allows the measurement of stability factor on amplifiers with large signals present at the input.

- **Quasi-Periodic Transfer Function Analysis, QPXF (Small-Signal)**

  QPXF can be used to measure small-signal conversion gain and also PSRR with frequency translations taken into account can be measured.

- **Quasi-Periodic Noise Analysis, QPnoise (Small-Signal)**

  QPnoise allows the noise of systems that translate frequencies to be measured accurately. All the harmonics of the tones applied in the QPSS are taken into account. Therefore, QPnoise is useful for measuring receiver desensitization when a blocker is applied at the RF input.

- **Quasi-Periodic S-parameter Analysis, QPSP (Small-signal)**

  QPSP allows the measurement of S-Parameters for circuits that have two or more inputs and that translate frequency. For example, S11 is at the input frequency, S22 is at the output frequency, and S21 measures the forward conversion gain.

For details on the quasi-periodic analyses, see *Virtuoso Spectre Circuit Simulator RF Analysis Theory*.

Quasi-Periodic Steady-State (QPSS) analysis, a large-signal analysis, is used for circuits with multiple large tones. With QPSS, you can model periodic distortion and include harmonic effects.

QPSS computes both a large signal, the periodic steady-state response of the circuit, and also the distortion effects of a specified number of moderate signals, including the distortion effects of the number of harmonics that you choose. In Shooting, one signal must be designated large. This signal should be the signal that causes the largest amount of distortion in the system, or the signal that is non-sinusoidal. In Harmonic Balance, tstab should be set to yes for this signal. With QPSS, you can apply one or more additional signals at frequencies not harmonically related to the large signal, and these signals can be large enough to create

distortion. Shooting requires that these signals be sinusoidal, but Harmonic Balance allows one additional signal to be a pulse signal.

Quasi-Periodic Noise (QPnoise) analysis is similar to a noise analysis, except that it includes frequency conversion and intermodulation effects. QPnoise analysis is useful for predicting the noise behavior of mixers, switched-capacitor filters and other periodically or quasi-periodically driven circuits. Quasi-periodic systems are systems that have an irrational number as one of the input frequencies. Because of this, there is no exact period of the output waveform.

The Quasi-Periodic AC (QPAC), Quasi-Periodic S-parameter (QPSP) and Quasi-Periodic Transfer Function (QPXF) analyses all work in a similar way as the Spectre L AC, SP and XF analyses except that they include the frequency translations that occur because of a large-signal LO or clock input.

Envelope Analysis

Envelope analysis allows RF circuit designers to efficiently and accurately predict the envelope response of the RF circuits used in communication systems. This allows the ACPR and EVM to be calculated for power amplifier and transmit modulators to be calculated. The new fast envelope capability in release 10.1 can run 100 times faster than traditional envelope with very little loss in accuracy.

For details on the Envelope analysis, see *Virtuoso Spectre Circuit Simulator RF Analysis Theory.*

APS and Parasitic Reduction

Parasitic reduction is used in post parasitic re-simulations. It is RC reduction only with L and K values are preserved. Reducing the RC networks reduces the number of nodes in the circuit, thereby reducing the number of matrix calculations, so the simulation runs faster. It takes multiple poles at high frequencies and reduces them to a smaller order RC network with lower pole frequencies. When RC reduction is selected with the default settings, the minimum pole frequency in the reduced netlist is 1GHz. This might cause errors for RF circuits because the pole might be below the operating frequency. When RF is selected, the minimum pole frequency is raised to 30GHz. This will result is less reduction, and thus longer simulation times compared to the default setting. Selecting Fmax allows the user to set the minimum pole frequency in GHz. If you want 10GHz as the minimum pole frequency set the `Fmax`(GHz) field to 10, and not 10G. This allows you to make trade-offs between accuracy and runtime for your needs.

APS makes the CMOS models more efficient and allowing multi-threading for the device current evaluation. This will be relatively more efficient on schematic-level circuits and adds the transient APS algorithms for DC and tstab solutions. It also allows multi-threading for all the matrix operations for Harmonic Balance and shooting and their small-signal analyses.

The processor affinity field should be used when you are running two or more jobs on the same multi-core machine. For example, if you have an 8 core machine and you are running two 4-thread jobs; assigning one job to cores 0 through 3 and the other to 4 though 7 saves time by having the operating system always use the same range of cores. This can save some time since the OS has less work to do. Note that if one of the cores you assign the job to already has a job, the entire simulation will run slowly because that one core will need to run two jobs by sharing time between them. The assumption is that all the threads will run on cores that are fully dedicated to the APS job.

Co-simulation with Virtuoso AMS Designer

SpectreRF co-simulation with Virtuoso AMS Designer enables the verification of full chip RF transceivers with mixed-signal baseband and digital control circuits. Using SpectreRF envelope analysis in AMS Designer, you can calculate the transient envelope response of RF circuits while simulating the rest of the system using digital event-driven simulation or mixed-

signal behavioral models. Co-simulation allows fast verification of communication systems in which the system architecture is highly dependent on the performance of the individual RF blocks. Using SpectreRF periodic and quasi-periodic steady state analysis in AMS Designer enables fast and accurate simulation of digitally calibrated RF blocks. The technology makes it possible to analyze the effect of digital control on the performance of RF circuits.

For more information about cosimulating with AMS Designer, see *Virtuoso AMS Environment User Guide.*

Large vs. Small Signal Analysis

SpectreRF provides a variety of time-varying small signal analysis for both periodic and quasi-periodic circuits. These small-signal analyses accurately model the frequency translation effects of time-varying circuits. Rather than using traditional small-signal analyses for circuits that exhibit no frequency translation, such as amplifiers and filters, you can simulate circuits that translate frequencies using time-varying small-signal analyses.

Circuits designed to translate from one frequency to another include mixers, samplers, frequency multipliers, phase-locked loops and parametric oscillators. Such circuits are commonly found in wireless communication systems.

Other circuits that translate energy between frequencies as a side effect include oscillators, switched-capacitor and switched-current filters, chopper-stabilized and parametric amplifiers, and sample-and-hold circuits. These circuits are found in both analog and RF circuits.

The quasi-periodic small-signal analyses accurately model the small signal characteristics of circuits with a quasi-periodic operating point, such as mixers with multiple LO frequencies or large RF inputs. The periodic small-signal analyses are more useful for circuits with a single input frequency.

Applying a time-varying small-signal analysis is a two-step process.

First, the simulator ignores the small input or noise signals while performing a PSS, QPSS or Hb analysis to compute the steady-state response to the remaining large-signals, such as the LO or the clock. The PSS or QPSS analysis is the full nonlinear solution for the circuit.

For each subsequent small-signal analysis, the simulator uses the nonlinear operating point computed by the PSS, HB, or QPSS analysis to predict the circuit response to a small sinusoid at an arbitrary frequency. You can perform any number of small-signal analyses after calculating the time-varying large-signal operating point.

The input signals for the small-signal analyses must be sufficiently small that the circuit does not respond to them in a significantly nonlinear fashion. You should use input signals that are

at least 10 dB smaller than the 1 dB compression point. This restriction does not apply to the signals you apply in the large-signal analysis.

This two-step process is widely applicable because most circuits that translate frequency react in a strongly nonlinear manner to one stimulus, usually either the LO or the clock, while they react in a nearly-linear manner to other stimuli such as the inputs. A mixer is a typical example. Its noise and conversion characteristics improve if it is discontinuously switched between two states by the LO, yet it must respond linearly to the input signal over a wide dynamic range.

To analyze a mixer with a small RF input and a single LO, you should use a PSS large-signal analysis using shooting or hb as the engine followed by one or more of the PAC, Pnoise, PSP or PXF small-signal analyses.

If the mixer has a small RF input and a large blocker as well as the LO, then a QPSS analysis would be the more appropriate large-signal analysis. Follow the QPSS analysis with one or more of the QPAC, QPNoise, QPSP or QPXF small-signal analyses for the RF input. The basic idea is this: If you have signals in your circuit that cause large-signal effects (like a blocker) this signal must be applied in a large-signal analysis like PSS for one input, or QPSS (or HB) for multiple inputs. For signals that are largely linear (like the conversion product of a receive mixer), you can use small-signal analyses which run faster. If the signals are largely sinusoidal, then hb and the small-signal analyses hbac, hbnoise, and hbsp can also be used.

Some circuits, such as frequency dividers, generate subharmonics. PSS can simulate the large-signal behavior of such circuits if you specify the period T to be that of the sub-harmonic. For other circuits, such as delta-sigma modulators, the periodically driven circuits respond chaotically, and you must use transient analysis rather than the PSS or QPSS analyses.

With the time-varying small-signal analyses such as QPAC or PXF, unlike traditional small-signal analyses such as AC or XF, there are many transfer functions between any single input and output due to mixing with harmonics. Usually, however, only one or two harmonics provide useful information. For example, when you analyze the down-conversion mixers found in receivers, you want to know about the transfer function that maps the input signal at the RF to the output signal at the IF, which is usually the LO minus the RF

2

Setting up a SpectreRF Simulation

Using Virtuoso ADE

Many people use ADE as a simulation environment because of the benefits it provides. The netlister netlists directly from the schematic. The direct plot form automates all of the standard RF measurements. Cross-probing to the schematic is provided. Easy set-up of the analyses to be run is provided. In ADE XL, multiple tests can be run at the same time. Monte Carlo and optimization are provided. In ADE GXL, you can estimate the parasitics before the layout is done to get a good estimate of the after-layout performance, and to identify nets that are very sensitive to layout parasitics.

A Testbench

A testbench is the schematic with the sources set up to provide the stimulus, and the different analyses to provide the measurements. In the schematic, make sure that the sources used for inputs and bias are at the top level, and have the circuit instantiated as a symbol (which could have levels of hierarchy below that). This allows these sources to be connected to the circuit but not have any connection to layout. Once the test setup has been made in ADE, that setup can be saved as an ADE state so that you don't have to start from scratch the next time you restart ADE.

ADE

ADE is the simulation environment for your circuit. once the schematic has been entered, ADE is started from the schematic. ADE L looks like the ADE from many years ago. It allows the setup of a single test, and allows parametric plots. It has three main sections: Variables, analysis setups, and calculator expressions to be plotted automatically after the simulation runs. The picture below shows a basic set of variables along with an HB and HBAC simulation setup.

ADE XL and GXL has a very different appearance, but allows the advanced analyses like yield optimization. ADE XL and GXL use several assistants for the different tasks to make the

setup easier. These assistants can be moved to your preferred places on the screen and saved. This allows your custom setup to be available later.

Analysis form

The *Choosing Analysis* form is provided for each analysis type you want to run. The *Choosing Analysis* form lists the sources in your circuit at the top. The number of harmonics and oversample factor are set in the edit fields just below the list. Accuracy is set by selecting liberal, moderate, or conservative. In general, to get a quick idea, choose liberal. For most simulations, choose moderate. When a more accurate simulation is desired, select conservative. If you have an oscillator, select the oscillator button. Sweeps can be set up in the form directly, or by using the parametric plot tool in ADE.

Each analysis has its own *Choosing Analysis* form where the appropriate information is entered for that analysis.

Direct Plot form

The *Direct Plot* form automates the plotting of signals. At the top, all the different analyses that were run are listed. Just below are the different things that can be plotted. In SpectreRF, both the time and frequency domain results are provided regardless of whether the simulation is done in the time domain using Shooting, or in the frequency domain using Harmonic Balance. Peak or RMS results are available, and when frequency domain results are present, modifiers are allowed so that dB or phase could be selected for the results.

When a selection is made in the schematic, the waveform tool displays the result.

Netlist Driven Flows

Netlist-driven flows start with a netlist. This might be done manually, or from a netlister that is developed in house. The netlister may or may not add the analysis statements to the netlist. One advantage to running from the netlist is the ability to alter a value and then run again, or to run multiple analyses of the same type with different settings.

To get spectre help for the netlist-driven flow, type spectre -h in a shell window. This gives a list of topics that have help available for, and then you might for example type spectre -h pss. When you do this, you will see the analysis statement and all the options along with all the values that are available.

When the netlist has the proper analysis statements, you can run spectre. Examples of this are shown below:

```
spectre netlist.scs
```

```
spectre +aps +mt=2 +log spectre.out netlist.scs (This runs APS with 2 threads and
     a log file spectre.out)
```

When Spectre runs, the results are saved in the <netlist_name>.raw directory. To view the results, type viva& (wavescan& in IC5141) in a shell window, and then plot the waveforms from the Results Browser and the Calculator.

3

Frequency Domain Analyses: Harmonic Balance

Overview of Simulation Capabilities

SpectreRF offers two different modes; Spectre and APS. Spectre uses traditional simulation techniques. APS makes the CMOS models more efficient and allows multi-threading for the device current evaluation. In addition, APS adds full parallel solving of all the matrices, thus speeding up the simulation even more. There is no loss in accuracy because there is no change in the actual equations that are simulated or in the options that control the accuracy of the simulation. These choices are available and are implemented for harmonic balance and periodic steady state simulations. Both large-signal harmonic balance and small-signal hbac, hbsp, and hbnoise analyses have the Spectre and APS implementations.

Large-Signal Harmonic Balance Overview

Harmonic balance is a large-signal analysis that solves in the frequency domain. It calculates the harmonics and mixing products of one or more inputs to the circuit. It takes into account all the large-signal effects in the circuit. Applications include measuring the harmonic content of a single input, calculating the large-signal IP3 of a power amplifier (with 2 input frequencies), and measuring the large-signal IP3 and IP2 of a transmit mixer (with 2 IF tones and an LO input). Harmonic balance can also provide a large-signal solution to the small-signal hbac, hbsp, and hbnoise analyses. More information on small-signal analyses is provided later in the chapter.

Harmonic balance is chosen for circuits that have near-sinusoidal signals and do not have high speed transitions. Although harmonic balance may work with circuits that have rapid transitions by setting lots of harmonics, it is likely that shooting (available in pss) will run faster. For LC oscillators and especially crystal oscillators, harmonic balance should be used. For ring oscillators, it is likely that shooting pss will be faster and more accurate.

The implementations of harmonic balance in the ADE *Choosing Analyses* form differ somewhat. The *Choosing Analyses* form has three choices that have harmonic balance

(hb) as a choice: hb, pss with the engine set to harmonic balance (hb), and qpss with the engine set to harmonic balance (hb). In hbhb and qpss when harmonic balance (hb) is selected as the engine, the harmonics of only the inputs and the mixing products are calculated. In pss, when harmonic balance is selected as the engine, all the harmonics of the inputs are calculated, even if the power in those harmonics is zero. For example, if there is an input at 2GHz and another at 2.1GHz, the harmonics of 100MHz are calculated. Note that in order to calculate through the fifth harmonic of the input at 2.1GHz, 105 harmonics need to be calculated. For this reason, pss is usually used for a single input frequency, and qpss or hb is usually used for multiple input frequencies. The hb analysis does not require any distinction. It always calculates the harmonics of the inputs, and if multiple inputs are present, it also calculates the mixing frequencies. Also, the hb analysis has an auto mode where all you do is provide the input frequency. Everything else is automatic. This chapter focuses on the hb selection in the ADE *Choosing Analyses* form.

Harmonic Balance provides a solution in the frequency domain. Things that have direct frequency domain representations like capacitors, transmission lines, or S-parameter descriptions go directly into the frequency domain solution. Devices and other nonlinearities need to be evaluated in the time domain, with the ifft, and fft used to translate between the domains. This is an iterative process. Each successive iteration produces a more accurate solution. In other words, an exact solution is never attained. The iterations continue until the answer is "close enough". Close enough is set by choosing an accuracy default in the *Choosing Analyses* form. Liberal, moderate, and conservative accuracy levels are provided. Choose the appropriate accuracy level based on your requirement. If you are calculating very small amplitude harmonics, you might need to choose conservative.

Example

Consider a behavioral low noise amplifier shown below:

The LNA is a behavioral LNA from rfLib. The properties list is shown below. Note that the gain is 20 dB, and the amplifier is only slightly nonlinear.

The input port (the source on the left side in the above schematic) has the properties shown below.

In general, give the signals names in the *Frequency name* property. In the example above, it is RF for the first frequency, and RF2 for the second. Giving names is required if you use the list of signals in the circuit in the harmonic balance Choosing Analyses form instead of

specifying the frequencies. The frequency is set to a variable name called frf and the amplitude in dBm is set to the variable name prf. Setting variables in these properties makes it easy to change the frequency or amplitude without needing to change the schematic. The variable frf is set to 2.45G and the variable prf is set to -30 in the variables section of ADE, as shown below.

ADE is used to set up the different analyses by clicking on the *Analysis - Choose* menu or by clicking the AC-DC-Tran icon ( ) located at the upper-right corner of the ADE window.

When you select *Analysis-Choose*, the *Choosing Analyses* form appears. At the top of the form, select the *hb* (harmonic balance) radio button.

In the *Choosing Analyses* form, the frequencies, the number of harmonics, oversample factor and accuracy are set. Oversample factor will be discussed later in the chapter. The

Fundamental Frequency field can either have a number like 2.4G, or it can have the variable name that is used to set the frequency in the input port as shown above.

Run transient has three settings: Run Automatically, Yes, and No. Run automatically means that the transient will be run before the harmonic balance algorithm. In this mode, the transient starts with a small number of periods of the fundamental frequency for Tone1 as the transient stop time. As the transient analysis runs, the waveform is analyzed to see if steady-state has been reached. When steady-state is reached, a Fourier transform is performed on the last cycle of the transient waveform, and this is used as the starting point of the harmonic balance iterations. If steady-state has not been reached in this small number of periods, the transient is extended in time up to a maximum of 250 periods for a driven circuit, and 500 periods for an oscillator. If steady-state has not been detected in the last period of the transient, other continuation methods will be tried in an attempt to achieve convergence.

Run Automatically also sets the number of harmonics in the *Choosing Analyses* form for Tone1 to auto by default. The transient analysis waveforms at all the nodes in the circuit are analyzed, and based on the waveforms, the number of harmonics that are needed for the simulation to be accurate is set by the simulator. This can be manually overridden by setting a number of harmonics manually.

Setting Run transient to Yes requires you to enter a stop time for the initial transient analysis in the Stop Time (tstab) field. This mode allows the steady-state detection to be selected or not. If it is selected, whenever steady-state is detected in the transient analysis, it will terminate, run the fft, and start the harmonic balance iterations. In this mode, the stop time for the transient analysis cannot be automatically extended. If Detect Steady-State is toggled off, the transient will run to the specified stop time without checking for steady-state.

When Run transient is yes, the number of harmonics field remains blank by default. You can either enter the word auto, or you can manually set the number of harmonics by specifying a number. When auto is set, the transient analysis waveforms at all the nodes in the circuit are analyzed, and based on the waveforms, the number of harmonics that are needed for the simulation to be accurate is set by the simulator.

Setting Run Transient to No causes the harmonic balance algorithm to start from the DC solution without transient assist.

When you have completed the *Choosing Analyses* form, click *OK* and run the simulation. You can run the simulation by selecting *Simulation - Netlist and Run*, or by selecting the green arrow on the right side of the ADE window.

When the simulation completes, select *Results - Direct Plot - Main Form*. The *Direct Plot Form* is displayed.

At any time, the next thing that needs to be accomplished is shown at the bottom of the form.

The *Analysis* section at the top of the form displays the list of different analyses that were run. In this example, only *hb* results are available.

The *Function* selection is where the type of data that is desired to be plotted is selected. *Power* is shown above.

To get the power based on the voltage in a net, *Net(specify R)* is chosen from the drop-down list.

Next select Magnitude, which gives power in Watts, dB10, which gives dB with respect to 1 Watt, or dBm, which gives dB with respect to 1 milliwatt. from the *Modifier* section. *dBm* is shown above.

The next step is to select the net in the schematic. The waveform window appears with the selected result in it. Both the input and output nets are shown. A marker has been positioned at the first harmonic in both traces. Note that the input level is almost exactly -30 as set in the ADE, and the output is 20dB larger than the input as set in the behavioral LNA. The harmonics at about -350dBm represent the numerical noise floor of the simulation.

Harmonic Balance Solves For Cosines

In order to be compatible with other harmonic balance simulators, the convention in Spectre harmonic balance is to solve for cosines rather than sines. From a practical point of view, this means that the phase that is solved in harmonic balance is 90 degrees different compared to

shooting. It also means that in order to get a zero degree solution in harmonic balance, the phase of the input source needs to be 90 degrees.

The circuit below is used for this example.

In the properties list for the input port, the phase has been set to 90 degrees. This is the phase of the sinusoid that is produced by the input source.

Cell Name
port
View Name
symbol
Instance Name
PORT0
Add
Delete
Modify
User Property
Master Value
Local Value
Display
IvsIgnore
TRUE
CDF Parameter
Value
Display
Port mode
Normal
HarmonicPort
Resistance
50 Ohms
Reactance
Port number
DC voltage
Source type
sine
Frequency name 1
RF
Frequency 1
frf Hz
Amplitude 1 (Vpk)
Amplitude 1 (dBm)
prf
Phase for Sinusoid 1
90
Sine DC level
Delay time
Display second sinusoid
Frequency name 2
RF2
Frequency 2
frf+100M Hz
Amplitude 2 (Vpk)
Amplitude 2 (dBm)
prf
Phase for Sinusoid 2
Display multi sinusoid
Display modulation params
Display small signal params
PAC Magnitude
1 V
PAC Magnitude (dBm)
PAC phase
AC Magnitude
AC phase
XF Magnitude
off

The signal *in1*, which is the input to the behavioral amplifier has been plotted below. The power in dBm is unaffected by the phase, but you can see that the phase that is calculated by Spectre is zero degrees. In harmonic balance, if you want a reference phase of zero, you must set the phase on the input source to 90 degrees.

Setting Harmonics Automatically

Harmonic balance can now set harmonics automatically for the signal that causes the most distortion. This is recommended in the general case. To enable this, select *Decide automatically* or *Yes* for the Run Transient selection in the Transient-Aided Options. This single action will cause a transient analysis to be run until steady-state is detected, and then from the transient analysis, the number of harmonics for Tone1 (when Frequencies is selected) or for the tone that has tstab enabled (when Names is selected).

Setting Harmonics Manually

Consider an amplifier with a single input. In this case the input and its harmonics need to be calculated. In order to get the correct solution, both positive and negative frequencies need to be calculated. With harmonic balance, input is required as to how many harmonics should be calculated. If you set 3 harmonics, then -3. -2, -1, 0, 1, 2, and 3 times the input frequency needs to be calculated, as shown below.

The number of harmonics to be calculated is (2*number of harmonics) +1. The runtime and memory consumption are proportional to the number of unknowns (all the harmonics at all the nodes) that need to be solved. In this case, seven unknowns need to be solved at each node.

In order to make sure that you have enough harmonics, start with an estimate based on the power level and the harmonic content of the input to the circuit. If the input is sinusoidal at fairly low power, try about 3 harmonics. If the amplitude is large, try about 7 harmonics. Next, run the simulation and plot the results. Position a marker at the desired measurement. Now increase the number of harmonics, and re-run the simulation and re-plot the waveform. If the measurement did not change, then the original estimate was enough, and you might be able to reduce the number of harmonics. Use the smallest number of harmonics that are required in order to have a minimum runtime.

If the input is a pulse wave at high amplitude, start at about 15 harmonics and an oversample factor of 4. Oversample factor will be discussed later in the chapter. The simulator takes the fft of the input signal, and uses multiple sources in series with the values and phases set to the value calculated by the fft.

When square waves are present in the circuit, the minimum number of harmonics should be set to the period of the square wave divided by the risetime of the square wave. Very sharp edges require many harmonics in order to be accurate in the time domain. If the time-domain waveform is less important than the frequency domain content, then a smaller number of harmonics can be specified along with an oversample factor of 4 or 8. For more information, refer to the Oversample Factor section of this chapter.

When piecewise linear waveforms are used as power supplies to ramp the power up at time zero, specify only two points. The first point should have time set to zero, and the voltage

should be about 80% to 90% of the actual supply voltage. The second point should have the time set equal to about half the period of the operating frequency, and the voltage set to the supply voltage. If the simulation time exceeds the second time in the PWL setup, it will retain that value for the rest of the simulation. More importantly, for any SpectreRF large-signal analysis, the system does not become periodic until after the last timepoint in the PWL file. If you put a point at 1 second and the power supply voltage, the system will need to be simulated for one full second in the tstab interval using the transient algorithm before the harmonic balance simulation can run. This can take a long time.

When periodic piecewise linear inputs are used, the more non-sinusoidal the waveform is, the more harmonics you need and also the higher oversample factor needs to be in order to accurately simulate the system. Oversample factor is explained in the following section. Start with an estimate of how many harmonics would be required to represent the waveform in the frequency domain, and then re-run the simulation with more harmonics to see if things changed. Oversample factor also needs to be increased. Sine waves need oversample=1. Square waves need 4 or 8. The more nonsinusoidal the waveform is, the higher the oversample factor needs to be. Note that even with a sinusoid applied, the currents may be very nonsinusoidal. Think of the currents in a diode mixer with a sinusoidal LO input. In this case, an oversample factor of 4 or 8 is likely to be required.

Up to two non-sinusoidal sources are allowed in the circuit. The first can be any periodic signal type, like pulse, exponential, and periodic piecewise linear. The second non-sinusoidal source is limited to being a pulse waveform. The rest of the inputs must be sinusoidal.

Oversample Factor

Note that fft and ifft are used to translate between time and frequency domains for the nonlinear evaluation. It is the nature of the ifft and fft to require more samples than the absolute minimum required in the time domain to get a correct answer for the harmonics in the frequency domain when the waveform is nonsinusoidal.

The circuit below has a square wave source on the top, and a sine wave source on the bottom. The load for both is a resistor.

There is no nonlinearity at all in the circuit. Therefore, the harmonics of the sinusoid and the square wave can be analyzed using the transient analysis and the DFT function in the calculator. The DFT function uses an fft algorithm to calculate the peak value of each

harmonic. The transient waveforms are shown below. Both signals have exactly 1 volt peak amplitudes.

In the following figure, you can see that the fft of the sine wave is essentially the same for 4, 16, 64, and 256 samples of the transient waveform. The waveforms are shown in the top-left

sub window. The number of samples can be read in the legend at the top-left corner of each sub window.

All the readings are in pico-dB or femto-dB. Sine waves do not need extra timepoints to calculate the correct amplitude for the harmonic.

Shown below are the ffts of the square wave with 4, 8, 16, 32, 64, 128, and 256 samples. The number of samples can be read at the upper-left corner of each sub window in the legend.

Note that with four samples, the value is 0dBV, which is 1 volt. With 256 samples, the result is 2.09777dB. This is almost 2.1dB larger than with four samples and is obtained just by increasing the number of samples. With 32 samples, you can get almost the right answer. This is a factor of eight times the number of samples required to calculate two harmonics. In other words, if two harmonics are set, an oversample factor of at least eight is required to get the correct answer for the square wave.

When 8192 points are used, only a minor change is observed as shown below.

Trading off Harmonics and Oversample Factor

The risetime of the above square wave is 1/500th the period of the square wave. Technically, the period divided by the risetime or 500 harmonics are required to obtain the correct answer. The hb solution for the waveform (from an ifft) and in the frequency domain are shown below.

The large number of harmonics requires a long runtime when real circuits are simulated. Because of the large number of harmonics, shooting pss is likely to run faster than hb for this level of accuracy.

If accuracy in the time domain can be compromised a bit, setting a relatively small number of harmonics with an oversample factor of 8 can get reasonably close to the solution above. In the below example, 5 harmonics with an oversample factor of 8 have been used.

Note that the answer in the frequency domain is only a few thousandths of a dB different than the answer with 500 harmonics. The waveform in the time domain (from an ifft) is very different than the transient waveform. If you only need an answer in the frequency domain, setting a small number of harmonics along with a high oversample factor will result in much faster runtimes with little compromise in accuracy in the frequency domain when non-sinusoidal waveforms are present.

The basic strategy with non-sinusoidal waveforms is to start with an estimate of how many harmonics might be required, and set the oversample factor to 8. Run the simulation. Reduce the number of harmonics by about 50% and run the simulation again. If the answer does not change, reduce the number of harmonics again. Do the same with oversample. You are

looking for the lowest runtime that also produces the correct answer, which will usually occur with a relatively large number for oversample factor, and a relatively small number of harmonics.

Every circuit is a little bit different in terms of the number of harmonics and oversample factor, but the principles are the same.

Two Input Frequencies

Consider an amplifier with two input frequencies. Because the circuit is nonlinear, intermodulation distortion is created. A lot more harmonics are produced by the circuit and need to be solved in the simulation. As a result, the runtime is longer and more memory is required. The harmonics to be calculated can be seen graphically, as shown below. Three harmonics are specified for both signals. The horizontal axis represents the first frequency and has symbols at -3*input frequency, -2*input frequency, -1*input frequency, the DC level, +1*input frequency, +2*input frequency, and +3*input frequency. Similarly, the vertical axis has symbols at the frequencies of the harmonics for the second input signal. The first symbol, 45 degrees from the origin, is the mixing product `Frequency1` + `Frequency2`. The figure below shows several different mixing products.

Note that the actual frequencies are not represented well by the chart. If F1 is 1GHz, and F2 is 1.1GHz, then F1+F2 is 2.1GHz and this is the first point at +45 degrees. F1-F2 is 100MHz, and is represented by the first point at -45 degrees.

This is called a rectangular cut, which is the default setting in harmonic balance. Note that ((2*number of harmonics on tone 1) +1) * ((2*number of harmonics on tone 2) +1) harmonics need to be calculated. In this case 49 unknowns need to be solved at each node of the circuit.

Note that the diagonal corners are sixth order terms. For example, the upper-right corner is 3*F1 + 3*F2. Since it is a sixth order term, it is likely that the amount of power that the circuit generates is small, and therefore can be removed from the solution space, thereby speeding up the simulation with minimal loss in accuracy.

Diamond Cut

SpectreRF has a diamond cut available where the high order terms can be removed from the simulation. The solution space is shown below for a 4th order diamond cut. The solutions in red are in the solution space. The solutions in black are not. The figure below assumes that 7 harmonics are specified for `Frequency 1`, and 3 for `Frequency 2`.

It is called a diamond cut because graphically, it looks like a diamond. The rectangular cut would have 15*7 or 105 harmonics in it. This cut has 39 harmonics in it. Instead of solving 105 unknowns at each node, 39 unknowns are solved at each node, therefore, the simulation runs considerably faster and takes less memory. All the fourth order and lower order terms are in the solution. The fifth order and higher order terms are excluded. As long as there is not much power in those harmonics, the solution will remain accurate.

Diamond cut is useful for circuits where the distortion is similar in the circuit from all the input frequencies. An example is an IP3 simulation for a power amplifier. Because the power level of both inputs is the same, both signals cause similar distortion.

The order of the cut can be set in the ADE *Choosing Analyses* form. Setting the order of the cut is similar to setting the number of harmonics. The more the distortion in the circuit, the larger the order needs to be for accuracy. Start with an estimate of what the circuit might produce, and run the simulation. Take the measurement. Raise the order, re-run, and re-plot. If the measurement changed significantly, then the order needs to be raised again. If the measurement did not change, try a smaller order. Use the smallest order that produces accurate results.

Funnel cut is also available. Funnel cut is a diamond cut with the addition of the harmonics on the axis for all the tones. This cut is useful for systems where there is a difference in the distortion from the different inputs. One example would be a mixer with a large amplitude LO, and an RF tone. In this case, because the LO power is large, all the harmonics of the LO are desired, but the mixing products are limited to lower order. This reduces aliasing with the addition of only a small number of harmonics. A diagram is shown below.

2 input frequencies. Funnel Cut. Red Xs are calculated.

Three Input Frequencies

Now imagine a mixer with an LO and two RF signals. Here, there are multiple planes of harmonics that need to be calculated, as shown in the figure below. Only 2 harmonics of the RF tones and one harmonic of the LO are shown in the figure so the planes can be visualized.

Note that ((2*number of harmonics on tone 1) +1) * ((2*number of harmonics on tone 2) +1) * ((2*number of harmonics on tone 3) +1) harmonics need to be calculated. If three harmonics were specified for all 3 tones, 7*7*7 or 343 harmonics need to be calculated at each node. Four input frequencies is a practical maximum for most circuits.

Diamond Cut With Three Frequencies

When the diamond frequency cut is applied, the red solution shown below is calculated when the maximum order is set to two. On the zero plane for the LO, a diamond that goes to the second harmonic of both tones is included. In the -1 and +1 planes for the LO, the diamond is drawn with the order reduced by one. When the number of LO harmonics is larger, which is the general case, on the 0 (zero) LO plane, the diamond goes through the order specified in the `maximorder` parameter. For the +1 and -1 planes, the order is reduced by one. For the +2 and -2 planes, it is reduced by one more. This continues on the rest of the planes.

Now imagine that there were more LO harmonics. More planes would be created. If the maximum order was set to 4, on the center plane, the included elements would go through the fourth order. For the first plane above and below, harmonics through the third order would be calculated. For the second plane above and below, harmonics through the second order would be included. For the third plane above and below, only the first harmonic would be calculated. For any other planes, no harmonics would be calculated.

You can see from the figure that fewer harmonics are calculated when the cut is applied and that the diamond is extended vertically. Fewer harmonics are calculated in the upper and lower planes than in the center plane. In this way, many harmonics are pruned from the solution space, and the simulation is sped up in the process. Two harmonics for the LO is artificially small in order to understand the concept. Usually, between 3 and 15 harmonics are chosen in most circuits.

Funnel Cut

Note that in the following example, because of the high LO power, many LO harmonics would be created in the circuit, and therefore, might cause incorrect results if the higher harmonics of the LO were not included in the solution. Funnel cut takes the diamond cut, and adds harmonics along the axes for all the inputs up to the maximum specified by the number of harmonics for each tone. This is shown in the figure below. Note that the signal names on the

axes are changed from the previous figure. This is done to allow the diagram to fit on the page.

Note that the harmonics along the axes only are added. In this case, the LO harmonics are added. The funnel cut is provided for circuits like mixers where the distortion is different from the different inputs. The idea is that the maximum order of the diamond might be 4 or 5 which allows accurate calculation of the mixing products and still calculating enough harmonics of the signal(s) that causes relatively more distortion. If the diamond cut were used, a higher maximum order is necessary to allow the high order harmonics of the high distortion signal to be calculated which includes many more harmonics to be solved than the funnel cut.

Axis Cut

A fourth cut is available called Axis cut that only calculates the harmonics of the inputs themselves. No mixing products are calculated at all. This cut is only useful for getting a quick idea of the amplitudes of the individual tones. It is shown in the figure below.

Aliasing

The reason for setting the number of harmonics and the order of different cuts is because of aliasing. If the actual system produces power outside the solution in the higher harmonics, that power will appear in the solution you specified. In the example above with the axis cut, because no mixing products are calculated at all, all that power shows up as increased power somewhere in the harmonics that are actually calculated. It is not predictable where that power may be aliased to.

Aliasing needs to be checked, as can be seen in the following pictures.

The figure above displays the spectral calculation of an amplifier with 900MHz and 920MHz applied at the same level. Note the level of the harmonics at 880MHz, 900MHz, 920MHz, and 940MHz. In this solution, two harmonics were set for both tones.

When the number of harmonics in the solution is raised to three for each input, the aliasing is less because there is less power in the uncalculated harmonics of the system. Note that in this case, the measurements for the harmonics at 900MHz and 920MHz changed only by 0.01dB. If you only need a measurement of the output power at the main output frequencies, 2 harmonics are enough. The measurements at 880MHz and 940MHz have changed by 0.268dB and 0.263dB respectively. This might or might not be accurate enough for an IP3 measurement depending on your accuracy requirement. The solution is shown below.

In order to check if the solution has enough harmonics, even more harmonics need to be run. Below is the solution with five harmonics on each tone. Note that the levels have changed by 0.01 dB for both tones at 880MHz and 940MHz. Three harmonics are enough in this case.

Although the solution changed slightly with more harmonics, it is difficult to justify the extra runtime for a 0.01 dB change.

When the harmonics are raised to nine, the solution changes by less than 0.001dB as compared to five. To repeat, three harmonics is enough in this case. There was no significant change when the number of harmonics was raised above three.

The solution with nine harmonics is shown below.

Below is a slightly different view of harmonics. Harmonic balance is supported in pss, where all of the harmonics starting from zero through the highest harmonic are calculated.

Note: This is not suggested for most applications. This is just to understand the concepts only.

The first view shows the solution with slightly more harmonics than the number needed to calculate the upper third order product at 940MHz. The circuit is an amplifier with inputs at 900MHz and 920MHz.

The third order products are at 880MHz and 940MHz.

Note that there are significant harmonics from about 380MHz through 520MHz. These harmonics are not produced by the circuit. These harmonics are produced because the power in higher harmonics is significant, and that power shows up in the solution with the number of harmonics set up for this run. The take away from this is that if you do not calculate enough harmonics in the solution space, random and unpredictable errors are introduced in the solution specified by you.

When the number of harmonics are raised, the solution changes. The solution below has 205 harmonics and appears to have no aliasing. Note that the first order terms at 900MHz and

920MHz have only changed slightly, but that the third order terms have changed by a significant amount.

However, when the number of harmonics is raised to 210 (only 5 more harmonics), the picture changes. Here, it is apparent that there aliasing is still going on, and it is a bit difficult to infer which harmonics are real and which are produced by aliasing.

This is shown below.

It is only when 1005 harmonics are taken that the aliasing stops because the levels of the higher harmonics are on the order of the numerical noise floor.

Sine Representation of Square Wave

In harmonic balance, a square wave is approximated by a series of sinusoids at the odd harmonics. The actual waveform that the circuit sees is shown from a transient simulation in the waveform window shown below. The top trace has the first and third harmonic. The second trace contains all the harmonics through the 11th harmonic. The third trace contains all the harmonics through the 19th harmonic. The bottom trace has 25 harmonics. Note that as more harmonics are added, a closer approximation is made to the square wave. The flat part of the waveform gets flatter and the edge gets steeper.

Note that this is done for all non-sinusoidal waveforms by taking the fft of the waveform specified through the harmonic number and then stacking the voltage sources in series with the values set to the value from the fft.

Convergence

Like all harmonic balance simulators, an iterative approach is used to solve the circuit equations. In an iterative approach, there is always another iteration that causes the solution to become more accurate. You stop iterating as soon as the error in the solution becomes small. As a result, the solution is not exact.

To indicate progress in the Spectre output, two metrics are provided.

- `Resd_norm` measures the absolute error in the solution.
- `Delta_norm` measures the change from the last iteration.

Convergence is achieved when the sum of the currents at all the nodes is near zero, and there is only a small motion between the last iteration and the current iteration. When both `resd_norm` and `delta_norm` are less than one, the simulation has converged and the iterations stop.

The absolute accuracy is affected by the product of `reltol * residualtol`. The change that is allowed form iteration-to-iteration is affected by the product of `reltol *lteratio * steadyratio`.

Reltol is a global option, which is an option that affects all analyses that are run. This option can be set in ADE by selecting *Simulation - Options - Analog*. The default for reltol depends on the setting of `liberal`, `moderate`, or `conservative`, whether the circuit is driven or is an oscillator, and the number of inputs to the circuit. The *Residualtol* option is located just below the *reltol* option in the form, and is a multiplier to reltol. Normally residualtol is not used in SpectreRF. Lteratio depends on the setting of `liberal`, `moderate`, or `conservative` for the errpreset option.

Errpreset

Errpreset is shown in the ADE implementation section. Normally, the option *Iteratio* is not set away from the default for harmonic balance. *Steadyratio* is a harmonic balance option, and defaults to 1.0. Setting a larger *steadyratio* value allows a larger change from iteration-to-iteration and still allows convergence. Following are the default values for *reltol*, *Iteratio*, and *steadyratio*.

1 input driven circuit	Reltol	Lteratio	Steadyratio
Liberal	1e-3	3.5	1.0
Moderate	1e-3	3.5	1.0
Conservative	1e-4	10	1.0

multiple input driven circuit or oscillator	Reltol	Lteratio	Steadyratio
Liberal	1e-3	3.5	1.0
Moderate	1e-4	3.5	1.0
Conservative	1e-5	10	1.0

Note that the values for the *reltol* option are the maximum values for the frequency domain iterations only. For transient assist, the value of the *reltol* option in the global options form is used. If the *reltol* option is set to a value smaller than the values in the table, that value will be used. If the *reltol* option is set to a value higher than the values in the table, the values in the table will be used.

In some cases, the `resd_norm` (a measure of absolute error) in the Spectre output window will be below one, but the delta norm (a measure of the change between iterations) remains above one. In this case, the solution varies from iteration to iteration more than the amount allowed. This is usually caused by the small-amplitude harmonics changing relatively a lot, but since they are small, allowing a large delta is acceptable. Look at the delta norm to see the range it is in, and then set the *steadyratio* option slightly larger than that and re-run the simulation.

Iteration Limit

All iterative approaches have the possibility of not converging. In hb, there is a default limit of 100 iterations. This can be changed by setting the `maxperiods` option.

Note: If the iteration limit is reached without convergence, the solution point for the last iteration is written to the data file and is available for plotting. Make sure that the simulation actually converged by checking the Spectre output window. If you are running a sweep, the sweep will stop when the error is incurred.

The message is located at the end of the output, and is shown below.

```
/usr2/rdavis/demos/ic61/spectrerf_aps_workshop/simulation/DVCO_RFX_tb2/spectre/schematic/ps
File  Help                                                                    cadence

********** iter = 98 **********
Damping Factor is 0.3
Delta Norm=1.61e+03  at node I24.R1:1  harm=(1 0)
Resd Norm=2.34e+03  at node I12.C4.node3  harm=(1 0)
Frequency= 5.1047e+09 Hz, delta f= 5.94e+07

********** iter = 99 **********
Frequency Damped
Damping Factor is 0.1
Delta Norm=5.21e+03  at node I24.R1:1  harm=(1 0)
Resd Norm=2.88e+03  at node I12.C4.node3  harm=(1 0)
Frequency= 5.0026e+09 Hz, delta f= -1.02e+08

********** iter = 100 **********
Delta Norm=3.49e+03  at node I24.R1:1  harm=(1 0)
Resd Norm=2.31e+03  at node I12.C2.node3  harm=(1 0)
Frequency= 5.1144e+09 Hz, delta f= 1.12e+08

Error found by spectre during Harmonic Balance Steady State Analysis `hb'.
    ERROR (SPCRTRF-15278):  Maximum number of iterations reached. Simulation failed to converge.


*************************************************
Fundamental frequency is 5.11438 GHz.
*************************************************

CPU time=13 s

Analysis `hb' was terminated prematurely due to an error.

Notice from spectre.
    49 notices suppressed.
    49 warnings suppressed.

modelParameter: writing model parameter values to rawfile.
element: writing instance parameter values to rawfile.
outputParameter: writing output parameter values to rawfile.
designParamVals: writing netlist parameters to rawfile.
primitives: writing primitives to rawfile.
subckts: writing subcircuits to rawfile.
12
```

You can see from *resd_norm* at the last iteration, that the error is significant. (it is much greater than one.) This answer cannot be trusted. To allow more iterations, set the `maxperiods` option to a number greater than 100.

Number of Non-Sinusoidal Sources

Up to two non-sinusoidal sources are allowed in the circuit. The signal that causes the largest number of harmonics or is periodic piecewise linear in the circuit should be *Tone1* when *Tones* are set to *Frequencies*. *Tones* is a choice in the *Choosing Analyses* form. If *Tones* is set to *Names*, enable tstab in the list for this signal. The second non-sinusoidal source is limited to being a pulse waveform

All the rest of the sources in the circuit must be sinusoidal. When *Frequencies* is selected, there is a limit of four input frequencies in the circuit. *Names* does not have this limitation, but four input frequencies is a practical upper limit because the number of harmonics to solve explodes.

When *Names* is selected, the sources in the circuit need to have entries in the *Frequency name 1* or *Frequency name 2* properties in the signal sources in the circuit. When assigning names, make sure that if you have multiple inputs at a single frequency, like there might be for a differential circuit, the names should be the same. This enables SpectreRF to consider the two signals as one. If you have frequencies that are integer multiples of each other, these should have the same name too. In this case, the actual frequency will be the lower of the two frequencies. Make sure that enough harmonics of the low frequency are specified to incorporate enough harmonics for the high frequency input to prevent aliasing.

Harmonic Balance Starting Point

Harmonic balance has traditionally started from the DC solution for the first iteration in the frequency domain. This is not the default for SpectreRF. The default is to run a short transient until steady-state is reached, and then switch to harmonic balance where the number of harmonics is selected automatically based on the transient analysis waveforms.

Starting from a DC point is a reasonable starting point for driven circuits. The input signal causes harmonics to be created and solved in the frequency domain. Transient aided harmonic balance is also available. This is accomplished by selecting Decide automatically or Yes for run transient. If Yes is selected, setting a simulation time in the stop time property is needed. Transient aided runs a transient analysis on the signal that you select for the time specified, and then runs a single period of the input. A Fourier transform is performed, and the result is used as the starting point for the frequency domain iterations. Because the starting point is closer to the actual solution, it usually requires fewer frequency domain iterations at the expense of running the transient analysis at the start. It is frequently faster to run a short transient and then iterate in the frequency domain. The only way to determine which is faster is to try both ways and see which one finishes first. Only one signal can have transient assist. When *Tones* is set to *Names*, the signal with tstab set to *yes* can have transient assist. Choose the signal that produces the largest amount of distortion for transient

assist. When *Tones* is set to *Frequencies*, only *Tone1* can have transient assist. *Tones* is an option in the *Choosing Analyses* form.

Transient assist can also be used to improve convergence. It works because the starting point in the frequency domain is closer to the real solution than the DC solution, which is the default. Anytime an iterative approach is used, the closer the starting point is to the real solution, the higher the probability of convergence.

In some cases, very lengthy transient simulations are needed to produce convergence. There are two ways you might use to shorten this time. The first way is to specify a filename in the *writehb* option on the first run, and then use the *readhb* option on subsequent analyses. The *writehb* option writes out every relevant parameter to a file so that when *readhb* is specified, the simulation can start from the file. The second way to reduce the time is to save the state near the end of the transient assist (tstab) interval, and then use that state to restart the transient assist with a much shorter simulation time before going into the frequency domain iterations. This is accomplished using the *savefile*, *saveperiod*, and *savetime* options.

Both of these ways allow circuit values to be changed when restarted, but the topology of the circuit cannot be altered. When the topology is altered, the transient assist needs to start over. If a value is changed, the change causes an instantaneous discontinuity in the solution and because of this, if the change is too large, there will be convergence failures.

More Capabilities

Save and Recover

Save and recover writes and restores the full internal state of the transient analysis that is used in the tstab interval so that a circuit can be restarted from that state to save time in a simulation that has a long startup time, or in the case where a longer tstab is needed for the transient assist. For now, savefiles that are made using the transient analysis are not usable as a recover file in harmonic balance. The savefile must be created from the tstab interval in harmonic balance or pss.

To save a file, set the *savefile* option to a filename. If you are working from a netlist, the file will be created in the directory you are running from. If you are in the ADE environment, the file is located in the `netlist` directory. This is located in the *simulation/<circuit_name>/<simulator_name>/<view_name>/netlist* directory. The simulation directory defaults to your home directory, but it can also be set by the administrators of the Cadence tools. In most cases, this directory will be shown in the project directory of the *Setup - Simulator/Directory/Host* menu in ADE. If you are unable to locate this directory, contact your system administrator to find the location. Spectre is the simulator name, so if you had a circuit called

`abc`, and you started ADE from the schematic view, the `netlist` directory would be located at *~/simulation/abc/spectre/schematic/netlist*.

Next, set either *saveperiod* or type in a list of times in the *savetimes* field. Do not specify both at the same time. If tstab is set to 50n, *saveperiod* is set to 10n, and the *savefile* option is set to *save_file*, when the simulation reaches 10n in the tstab interval the file *save_file* is created in the `netlist` directory. When 20n is reached, *save_file* is overwritten with the data at 20n. This continues until the last 10n interval is reached in the tstab interval.

Alternatively, you can type 10n 25n 40n 50n in the *savetimes* field. This would create four files beginning with *save_file* and ending with the time for that specific point. The exact format of this file changes occasionally, so look in the `netlist` directory to get the actual filename you want to start from.

To recover, you specify the filename you want to recover from, and set tstab to a larger time than the time in the file. You can also change the list of times in the *savetimes* option to save more times after the time you recovered from.

Writehb and Readhb

Setting *writehb* causes the entire harmonic balance solution to be written to a file that can be used later. When readhb is set to a previously saved *writehb* file, the solution is read from the file, and then follow-on small-signal analyses can be performed. Note that *readhb* and *writehb* cannot be set at the same time.

Readhb can also be used as a starting point for when more input frequencies are supplied to the circuit. If you have a 2-tone solution, and then you add a third tone, The two tones in the solution are used as a starting point for the 3-tone solution which must iterate to achieve convergence. Because the starting point contains the solution for two signals, this is a better starting point than using transient aided where the solution to only one tone is available for the first iteration.

Sweeps and Restart

Sweeps are available both in the hb *Choosing Analyses* form and from the parametric tool in ADE. Sweeps automatically use the previous solution as the starting point for the next member of the sweep. If convergence difficulties are encountered, the sweep parameter will be automatically stepped from the value that converged to the value that needs to be measured. When Monte Carlo encounters convergence difficulties, several things are tried to achieve convergence. If you want to disable these automatic features, set the restart option to yes.

Itres

Itres controls the precision of the solution at the first iteration in the frequency domain. Harmonic balance starts with the DC solution as the starting point which causes the solution at the first iteration to be inaccurate. Because of this, the first solution is calculated without much precision (number of digits in the solution) in order to save time. Follow-on iterations have more precision. The final solution has full precision. Note that if you use transient-aided harmonic balance, the starting point in the frequency domain can be quite good, and so itres should be manually set to 1e-2. This forces enough digits in the mantissa to be solved so that the solution differs from a solution with all the digits of the mantissa by less than 1 part in 100. If the circuit is nearly linear (for example an LNA) increasing the precision by setting itres to 1e-2 can also reduce the total number of iterations. Regardless of the setting of itres, as the iterations progress, the precision is automatically increased so that the final solution is accurate. The default is 0.1, which allows a 10% error on the first iteration.

Freqdivide

If the circuit has a frequency divider, specify the divide ratio in the *freqdivide* property. This needs to be an integer. Make sure that oversample factor is set in this case as discussed earlier. Also, make sure that transient-assisted is used, and set the transient part to at least the time needed for the divider to count through its sequence once. The effect of this is that the fundamental frequency is divided down by the freqdivide ratio. Make sure that enough harmonics are specified so that at least the fifth harmonic of the input signal is calculated. If *freqdivide* is set to four, make sure that at least 20 harmonics are specified for that signal.

Oscillators

For oscillators, a DC starting point is not a good starting point. The frequency and amplitude of the harmonics needs to be solved and the DC state has no information at all about this. There is no signal source, so the oscillator usually just does not start. The default in ADE is to estimate the oscillatory frequency and amplitude, and then run a transient analysis until steady-state is reached. Upon reaching steady-state, the number of harmonics needed for the oscillator are calculated by evaluating the waveform at every node in the oscillator, and then running harmonic balance to solve for the frequency and amplitude at each node. This is a good starting point for LC, crystal, or mems oscillators.

If the oscillator is a ring oscillator, apply the initial conditions to force one stage of the ring to a high or low state. Set Run transient to Decide automatically. If desired, turn off *Calculate initial conditions automatically*, which will only work for a conventional feedback oscillator.

When using transient assist manually (called the tstab interval) for oscillators, set *Run Transient* to *Yes*. Specify a time for tstab, on the order of twenty periods of oscillation. Select

the check box for Detect Steady State. This will enable the transient, and automatically switch to harmonic balance when steady-state is reached. In this mode, tstab cannot be lengthened automatically like it can be when Decide Automatically is selected.

In many cases, it is desirable to see the startup waveform, which is not saved by default. If you want to see the startup behavior, select *yes* for the *Save Initial Transient Results (saveinit)* option.

When oscillators are simulated, a node or pair of nodes (if it is differential) needs to be declared to the simulator. This pair of nodes is used to get an estimate of the oscillation frequency at the end of the transient assist period. At the end of the transient assist period, 5 periods of the estimated operating frequency are simulated. In these 5 periods, the simulator looks for the frequency divided output at every node in the circuit and if it finds this behavior, it automatically adjusts the frequency. The frequency is estimated based on these last 5 cycles. A Fourier transform is performed on the last single cycle, and this is the starting point in the frequency domain. If frequency dividers are in the circuit, the estimated frequency of the oscillator is recommended to be specified in the *Choosing Analyses* form, and the frequency divide ratio should be specified in the freqdivide option.

Solving the frequency in addition to the amplitudes and phases adds another unknown to the solution set, so it usually takes more iterations to reach a solution compared to a driven circuit.

Pinnode

In a driven circuit, all the phases and amplitudes of the harmonics are defined by the input signal. There is no input for an oscillator, so a pinnode must be selected. The first harmonic phase of that node is pinned (held constant), and then the amplitude of all the nodes and relative phases of all the other nodes are calculated using the pinnode phase being held constant. SpectreRF has an algorithm that automatically selects the pinnode, and it is usually reliable. If convergence is an issue, try setting the pinnode manually in the options form to one of the nodes in the resonator.

To set the pinnode manually, select the *Options* button at the bottom of the hb *Choosing Analyses* form and select the convergence tab (*IC61 only*). In the options field, do not enter a slash(/) at the beginning of the expression. Just enter the hierarchical node name similar to this example: `I12.I1.osc_resonator`. In the schematic, IC61 shows the instance name of the block in the tab for that block, so it is a bit easier to get the hierarchical names.

Probe-Based Method

The normal method of solving for an oscillator allows the amplitudes, phases, and frequencies to be solved simultaneously. A second method called the probe-based method iterates for the frequency solution in the outer loop and the amplitude and phase solution in the inner loop. It places a voltage source between a pair of nodes you specify, which should be in the resonator. If you have a single-ended oscillator, specify one node only. If the second node is left blank, it will be connected to the global ground node automatically. The voltage source converts the oscillator into a driven circuit, which converges easier. The source amplitude, phase, and the frequency are adjusted until the current in the source is near zero. Do not use the probe-based method and the pinnode option together, since the probe-based method also requires a pinnode to be selected. *Harmonic Index* and *Magnitude* do not need to be specified for the probe-based method. The *harmonic index* defaults to 1, which is suggested, and the *Magnitude* defaults to a small amplitude. If you specify *Magnitude*, make sure that the magnitude you specify (in volts RMS) is smaller than the amplitude of the actual oscillations, or the method itself will fail.

Semi-Autonomous

Semi-autonomous is provided to allow the simulation of circuits that have periodic sources and oscillators in the same circuit. It is only supported in the hb *Choosing Analyses* form. Typical applications are for when you have a receiver with an RF tone and an oscillator for the LO so that all the oscillator noise is taken into account for the receiver noise simulation. Another application is for identifying the phase noise and spurious response of an oscillator when power supply ripple is applied.

The simulator combines the driven and oscillator capability in one analysis. To set it up is similar to a multi-tone simulation, just select the *Oscillator* check box and specify the estimate of the oscillator frequency in the *osc!* section of the form. The probe-based method for oscillators is not available for semi-autonomous simulations.

Things to try to Achieve Convergence (Driven Circuits)

Try running transient-assisted by setting the tstab option. Ensure that the signal that causes the most distortion is in *Tone 1* (when *Frequencies* is selected) or has tstab enabled (when *Names* is selected). Transient assist can help on any circuit.

If there is an S-Parameter block in your circuit, make sure that causality is set to *fmax* on every instance of the nport. Starting with MMSIM 11.1 this is the default. The effect of this is to enforce causality for the DC and tstab interval. If this selection is not made, even the DC solution can be affected, sometimes producing results that can be larger than the power

supply voltage. Setting causality to *fmax* makes sure that the DC and tstab intervals have a causal model. Refer to the Simulating S-Parameters chapter for a description about the nport component. Also, make sure that every port of the nport has a connection to something in the circuit. Leaving a terminal open can be very bad for convergence.

Look at the startup waveform to see if it is settled. Make sure that tstab is long enough. Look at the starting waveform, and ensure that the amplitude is reasonable for your circuit. To improve convergence, the waveform must be pretty close to settled at the end of the tstab interval where the Fourier transform is made.

Use enough harmonics. If your system produces square waves, make sure that at least five harmonics are calculated. In many cases, many more harmonics than that are required.

Try setting the oversample factor to a higher value. This should be set to four or eight for circuits that have square waves. Setting it to two or four improves the accuracy of the Fourier transforms used in harmonic balance (hb) even for nearly sinusoidal circuits.

Try setting itres to 1e-2. This forces a solution that is more precise on the first iteration, thus improving the chance of convergence.

Try more maxperiods if `resd_norm` is small (near one). This indicates that the circuit is almost converged, and might converge more if more iterations are allowed. The default maxperiods is 100.

If the residual norm is below one, but the delta norm is greater than one, set *steadyratio* just a little higher than the value in which the delta norm is running. The effect is to allow more delta from iteration to iteration and still allow convergence. Only do this if the `resd_norm` is less than one, which indicates that the absolute accuracy check is good.

Try moderate accuracy. Sometimes, loosening the tolerances allows a circuit to converge. Note that this does reduce the accuracy of the solution the simulator solves for.

Things to try to Achieve Convergence (Oscillators)

Try everything listed above. In addition:

Try setting *reltol* to 1e-5 and *vabstol* to 3e-8. These options can be found in the global options form in ADE. Select *Simulation* - *Options* - *Analog* to open the global options form. Making *reltol* and *vabstol* smaller makes the initial transient much more accurate by reducing the error caused by iterating to each solution and the numerical integration error. By providing a more accurate estimate of the frequency and amplitude of the waveform, convergence becomes easier.

Try setting a pinnode manually. While the algorithm that is used in SpectreRF is usually reliable, if the oscillator does not converge, set the pinnode to one of the nodes in the resonator.

Select the *Calculate initial conditions automatically* check box if you have an LC oscillator. This causes an initial estimate of the frequency and amplitude to occur at the start of the tstab interval. Add transient assist so that the harmonics can be calculated for the starting point in the frequency domain.

Try the probe based method. This method adds a voltage source to the circuit so it becomes a driven circuit. Because the circuit is driven, convergence is easier. The frequency amplitude and the phase of the source are adjusted so that the current of the source becomes zero. When this happens, the source has no effect on the circuit.

Try setting an initial condition (current) in the inductor of the resonator. Choose a value that approximates the steady-state peak value for the oscillatory condition.

Implementation in ADE

Setting Frequencies, Harmonics, and Oversample

Setting Harmonics and Transient Assist Automatically

The default in ADE is to run transient assist until steady-state is just reached, and then choose the number of harmonics for the first tone automatically, and then proceed with solving in the frequency domain.

The defaults are highlighted above. This configuration is recommended for all hb analyses.

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/D
File  Help                                                    cadence

    hb: time = 5.369 ns      (88 %), step = 2.083 ps     (500 m%)
    hb: time = 5.39 ns       (93 %), step = 2.083 ps     (500 m%)
    hb: time = 5.41 ns       (98 %), step = 2.083 ps     (500 m%)

Notice from spectre during Harmonic Balance Steady State Analysis `hb'
    Auto harmonic calculation has chosen 3 harmonics for tone-1.
    Solution size = 20450, itres = 0.1, max GMRES iteration = 80
    Use new Arnoldi routine


==============================
     Harmonic balance
  hbhomotopy=tone (1-tone)
==============================
Important HB parameters:
    RelTol=1.00e-05
    abstol(I)=1.00e-12 A
    abstol(V)=1.00e-06 V
    residualtol=1.00e+00
    lteratio=1.00e+01
    steadyratio=1.00e+00
    maxperiods=100


********** initial residual **********
Resd Norm=2.11e+01  at node I0.I28.I6.nt1  harm=(1 0)

********** iter = 1 **********
Delta Norm=3.01e+01  at node I9:bal_p_flow  harm=(1 0)
Resd Norm=3.50e-02  at node I0.I31.NM13:int_s  harm=(2 0)

10  HelpAction
```

Spectre will calculate the number of harmonics that are needed and report the number in the spectre output file, as highlighted above.

Tones = Frequencies

1. For sinusoidal sources, set *Oversample Factor* to 1.
2. For square waves, set *Oversample Factor* to 2, 4, or 8. Use the smallest number that provides accurate results.

3. If there are multiple sources with a common integer multiple frequency in your circuit, specify the highest common frequency. For example, if 1GHz and 1.5GHz are present in the circuit, set *Tone1* or *Tone2* frequency to 500MHz. In this case, make sure you take enough harmonics of 500MHz to get an accurate result at 1.5GHz.

Important

Specify the signal that causes the most distortion in the circuit as Tone1. This is the tone that can have transient assist, and selecting the signal with the most distortion as Tone1 improves the accuracy of harmonic balance.

Tones = Names

1. Select a line in the sources list.

   The fields (highlighted below) are automatically populated with the values of the selected line in the sources list.

2. Change the number of harmonics under the *Mxham* column (auto in the figure above).
3. Change the oversample factor under the *Ovsap* column (1 in the figure above).
4. Set only one of the inputs to have transient assist by setting *Tsta*b to *yes*.

Important

Make sure you enable tstab for the signal that has the largest amount of distortion in the circuit. This will make harmonic balance more accurate.

5. If there are multiple sources in your circuit that have integer frequency relationships, set the *Frequency name 1* property in all the sources in the schematic to the same name. This will cause all the sources to be treated as solving for harmonics of the highest common frequency. For example, if 2.4GHz and 3.6GHz were present in the circuit, a single-tone simulation would be run that solves the harmonics of 1.2GHz. Make sure that you use enough harmonics so that the enough harmonics of 3.6 GHz are solved to prevent aliasing errors.

Diamond Cut

This is used when the number of significant harmonics produced by the circuit are similar for all the inputs. This is usually the case for a large-signal IP3 measurement on an amplifier.

In the *Choosing Analyses* form:

1. Choose *Select* from the Harmonics drop-down list.
2. Select *diamond*.
3. Type a reasonable value in the *MaximOrder (int)* field. The value is usually between 3 and 7.

Funnel Cut

This is used when the number of significant harmonics produced by the circuit are different for one or more inputs. This is usually the case for a large-signal IP3 measurement on a mixer (The LO usually produces more harmonics than the RF input signals).

In the *Choosing Analyses* form:

1. Choose *Select* from the Harmonics drop-down list.
2. Select *funnel*.
3. Type a reasonable value in the *MaximOrder (int)* field. The value is usually between 3 and 9.

Sweeps

You can sweep up to three things in the hb *Choosing Analyses* form.

- Each can be a variable, temperature, component parameter, or model parameter.
- Choose what you want to sweep from the drop-down list.
- Set the start and stop range for the sweep in the *Start* and *Stop* fields.
- You can add points off the grid in the *Add Specific Points* field.

Freqdivide

If you set freqdivide, make sure that you have at least five harmonics at the input frequency as an absolute minimum. You must also set the *Oversample Factor* between 2 and 8.

Normally, when there are square waves in the circuit, for accurate results for the transient waveform, the number of harmonics needs to be at least the period of the square wave divided by the rise or fall-time, whichever is shorter.

Oscillator Additions

In the *Choosing Analyses* form, select `Names` or `Frequencies`.

Tones = Frequencies

1. If you have an oscillator, select *Oscillator*.

   Tone 1 will automatically change to *osc!*.

   The oscillator node location in the circuit only needs to have a signal on it.

2. Select *Decide automatically* from the *Run transient?* drop-down list. This will cause an estimate of the oscillator frequency and amplitude to be run, then the transient runs in the tstab interval, but only to the point of steady-state. Once steady-state is reached, a Fourier transform is calculated, and the frequency domain iterations begin.

3. Set the frequencies, number of harmonics, and oversample factor.

The *Fundamental Frequency* field should have a value between 0.5 and 1.5 times the actual frequency of oscillation. If *Run transient?* is set to *Decide automatically* or *Yes*, *auto* is set for the number of harmonics. At the end of the transient in the tstab interval, the number of harmonics will be set automatically based on the waveforms in the tstab interval.

4. Select *conservative* for oscillators.

Note: You should always select *conservative* for oscillators.

Tones = Names

1. If you have an oscillator, select *Oscillator*. The system will display a popup window that reminds you to provide an estimate for the oscillation field.

The oscillator node location in the circuit only needs to have a signal on it.

2. Specify a value in the *Expr* field that is between 0.5 and 1.5 times the actual frequency of oscillation.

3. Select *Decide Automatically* from the *Run transient?* drop-down list. This will cause tstab to be run only until steady-state is reached, where a Fourier transform is calculated, and the frequency domain iterations in harmonic balance begin.

4. *Decide automatically* also evaluates the Fourier transform at the end of the tstab interval, and sets the harmonics based on the actual spectrum. The word *auto* appears in the *Mxharm* field.

5. Select *conservative* accuracy for oscillators.

Note: You should always select *conservative* for oscillators.

Probe-Based Method

Note: Earlier, this used to be called the two-tier method.

In the *Choosing Analyses* form:

1. Choose *Frequencies* or *Names*.

2. Select *Use the probe-based solution method.*

Note: When the probe based method is used, the only choice is to set the *Run transient?* option to *No*.

3. Select an oscillator node that is inside the feedback system for *Pinnode+*.

4. If you have a differential circuit, select the differential node at the mirror location in the feedback system for *Pinnode-*.

5. Leave the *Harmonic Index* and *Magnitude* fields blank.

Semi-Autonomous

Semi-Autonomous allows an oscillator and a periodic source to be simulated at the same time.

In the *Choosing Analyses* form, select *Names* or *Frequencies*.

Frequencies

1. Select *Oscillator*.

2. Select the number of tones as desired.
3. Type an estimate for the *Fundamental Frequency* in the *osc!* column.
4. Type the frequencies for the periodic sources in the other columns.
5. Set the number of harmonics and the oversample factor as required. If *Run transient?* is set to *Decide automatically* or *Yes*, then *auto* is entered for the number of harmonics in the first tone only. This can be set manually, if desired. The number of harmonics for the rest of the tones must be set manually.
6. If there is a frequency divider after the oscillator in the circuit, set the divider ratio in the *Freqdivide Ratio for Tone 1* field. Remember to set oversample factor and enough harmonics when this is set.

7. If you want to reduce the number of harmonics set, choose *Select* from the *Harmonics* drop-down list and set *MaximOrder*.

8. Select *conservative*.

9. If you want to view the startup waveform, select *yes* for *Save Initial Transient Results (saveinit)*.

10. Specify the oscillator nodes in the circuit.

Names

1. Select *Oscillator*. The system will display a popup window that reminds you to supply an estimate for the *Expr* field.

2. Specify the frequency in the *Expr* field.

3. Select Decide automatically from the *Run Transient?* drop-down list. This will cause Spectre to run a transient in the tstab interval until steady-state is reached. Then, a Fourier transform is calculated, and the frequency domain iterations begin.

4. Set the harmonics and the oversample factor as required. If *Run transient?* is set to *Decide automatically* or *Yes*, then *auto* is entered for the number of harmonics in the first tone only. This can be set manually, if desired. The number of harmonics for the rest of the tones must be set manually.

5. Click *Change, or* select another field, or use the <*Tab*> key.

6. If there is a frequency divider after the oscillator in the circuit, set the divider ratio in the *Freqdivide Ratio for Tone 1* field. Remember to set oversample factor and enough harmonics when this is set.

7. If you want to reduce the number of harmonics set, choose *Select* from the *Harmonics* drop-down list.

8. Select *Conservative*.

9. If you want to view the startup waveform, select *yes* for *Save Initial Transient Results (saveinit)*.

10. Specify the oscillator nodes in the circuit. This node can be any node in the circuit where the oscillations are present.

Commonly Used Harmonic Balance Options (Driven Circuits and Oscillators)

The *Harmonic Balance Options* form deliberately omits uncommonly used options. In the unlikely case that you need options that are available in pss-harmonic balance, you will need to determine the `option name=value` pairs and enter this information in the *additionalParams* field in the *Misc* tab. The easiest way to get these `option name=value` pairs is to set the option in a dummy pss analysis, use ADE to create the netlist, and then highlight the text in the netlist reader. This text can be entered in the *additionalParams* field by clicking the center mouse button.

The options are divided into five tabs: *Convergence*, *Accuracy*, *Output*, *Reuse*, and *Misc* (miscellaneous).

The options are shown in the order that they appear in the *Harmonic Balance Options* form in ADE.

Convergence tab

Maxperiods

This specifies the maximum number of iterations for the hb algorithm. The default is 100. If this option is set less than 100, it will be ignored.

Itres

This option specifies the precision for the solution of the first iteration in harmonic balance. For subsequent iterations, the precision is increased so that when the iterations end, full precision is used. The default is 0.1, which allows up to 10% error in the first iteration. If you have a linear or near linear circuit like an LNA, lower itres to the 1e-2 to 1e-4 range so that

the solution for the first iteration is more precise. Fewer iterations will be required for the solution to converge.

Ic

Initial conditions can be specified graphically by selecting *Simulation - Convergence Aids - Initial Condition* in the ADE. Initial conditions can also be specified from a file using the *readic* property. Capacitors and inductors have the initial condition properties in the property list for the component. For capacitors, this is an initial voltage across the capacitor, and for inductors, it is an initial current in the inductor. The default is to observe all the initial conditions in the DC analysis that is used as the time-zero timepoint in the tstab interval. The initial conditions force a voltage or current to be present in the time-zero solution. The initial conditions are released for the rest of the transient simulation in the tstab interval. The *ic* option controls the initial conditions that should be observed in the time-zero timepoint. *all* is the default. *dev* means that only the initial conditions on capacitors and inductors are observed. *node* means that only the initial conditions on a node are observed. *dc* means that no initial conditions are observed. The example below shows the ic option set to *node*.

Skipdc

In some cases, the time-zero timepoint DC analysis does not converge. Instead of stopping the simulation, *skipdc* allows the tstab simulation to continue using an assumed solution for the time-zero timepoint. The default is *no* and a DC analysis is run to get the initial timepoint. *yes* means skip the DC solution, and proceed directly to the tstab simulation. All the nodes with initial conditions specified start at the initial condition value. Nodes with batteries start at the battery voltage. Nodes with no initial conditions start at zero volts. For `skipdc=yes`, the signal sources start as specified immediately in the tstab simulation. *sigrampup* uses the same assumptions for the starting voltages as *yes* does, but the start time is set to negative one tenth of the stop time for the tstab interval. At this time, the signal source time-varying part starts at zero and linearly ramps up to the full value at time = zero. After time = zero, the sources have the full amplitude time-varying part. The example below shows the *skipdc* option set to *sigrampup*.

Readic (Shooting and Harmonic Balance)

This specifies an ascii file that contains two columns to be read as initial conditions. The first column is the node name. The second column is the voltage value. If the entry in the *readic* field does not start with / (slash), the entry is located in the netlist directory. To find the netlist directory, select *Setup- Simulator/Directory/Host* in ADE. Look in the *Project Directory* field for the location of the simulation directory. Navigate to that directory and then to the `<Circuit Name>/spectre/<schematic or config>/netlist` directory. You can also click (...) and browse to the directory. An example is shown below.

Readns

This specifies an ascii file with the same format as an ic file that is used as nodesets for the time-zero DC solution. Nodesets do not force a voltage to be held for the time-zero solution. Instead, they are a way of speeding up the time-zero calculation. As a suggestion, set the *write* option and the *readns* option to the same filename. The *write* option writes the time-zero solution to a file. When this is used as a starting point, many fewer iterations are needed for the time-zero point to converge. An example is shown below.

Cmin

cmin can be used to improve convergence in the tstab and shooting intervals for shooting and for the tstab interval for harmonic balance. If a value is set for *cmin*, a capacitor with this value is added to every node with the other terminal of the capacitor connected to the global ground node. If a 10f to 50f capacitor is added, this prevents instantaneous changes from occurring from timepoint to timepoint, thus improving the convergence at the cost of adding non-physical capacitors to the circuit. An example with 10 femtoFarads is shown below. Note that if 10 is entered, a 10 Farad capacitor is added from every node to ground. Remember the multiplier in the entry. An example is shown below.

Accuracy Tab

The *Accuracy* tab is split into two categories, *TIME STEP PARAMETERS* and *INTEGRATION METHOD PARAMETERS*.

Maxstep

This is the maximum allowable timestep for the transient analysis in the tstab interval.

Tstabmethod

This option controls the integration method in the tstab interval. The default value is determined by the setting of *errpreset* and whether the circuit is driven or is an oscillator. For *moderate* and *conservative* with a driven circuit, the default is *gear2only*. For *liberal* it is *traponly*. Generally, *gear2only* is preferred because of the absence of trapezoidal ringing inherent in the *trap* (trapezoidal) method. The default for *tstabmethod* is *traponly* for oscillators. *traponly* is used for oscillators because with the defaults, the tstab interval uses relatively loose convergence options in order to speed up the simulation. This causes relatively long timesteps which for the *gear2* method cause noticeable numerical damping. *traponly* does not numerically damp, therefore, it provides a better estimate of the oscillating frequency and amplitude. An example with *tstabmethod* set to *gear2only* is shown below.

Output Tab

Annotate

This option is currently not implemented.

Save

The default is *allpub*. This saves all the public voltages at all levels of the hierarchy in the schematic. It excludes the internal nodes of the device models. *all* adds the internal nodes of all the devices. *lvl* saves all the nodes including the internal nodes of the devices through the level of hierarchy set in the *nestlvl* option. *lvlpub* is like *lvl*, but it does not save the internal nodes. *selected* saves only the nodes that are specifically saved. In ADE, this is accomplished by selecting *Outputs - To Be Saved - Select On Schematic* and then selecting the nodes and terminals specifically in the schematic. At the netlist level, this is accomplished by using a save statement with a list of names to be saved.

Nestlvl (Shooting and Harmonic Balance)

If *save* is set to *lvl* or *lvlpub*, this controls the maximum level of hierarchy to be saved. If *nestlvl* is *1*, only the top level is saved. If *nestlvl* is 2, the top level and the next level down are saved. The value for *nestlvl* can be any integer. The example below shows *lvlpub* selected that saves two levels of hierarchy.

Compression (Shooting and Harmonic Balance)

Normally, this should not be used for RF simulation. It is not digital compression. For RF simulation, where the input is sinusoidal, the size of the results file will normally double. It is useful only for circuits that are predominantly square wave.

Reuse Tab

Writehb

This option specifies that the full internal state of the harmonic balance analysis be written out to the file specified in the option. An example is shown below.

Readhb

This option specifies the file to be read in as a starting point for the next harmonic balance analysis. If nothing is changed in the circuit, then only one iteration is performed and small-signal analyses can be run with a much faster time for the hb analysis. Changes can be made to the circuit or analysis options as long as the topology of the circuit stays the same. The

changes introduce a discontinuity at the beginning of the hb analysis, which might be large enough to cause hb not to converge.

An example of *readhb* is shown below.

```
readhb    hb_state_file
```

Saveclock

saveclock saves the state of the simulation in the tstab interval at the clock time interval in seconds specified by *saveclock*. When the clock time has passed in the tstab interval, the file specified by the option *savefile* is created. When subsequent clock time intervals have passed, the file is overwritten. Use only one of the save options at a time. The example below shows *saveclock* set to three minutes.

```
saveclock    180
```

Saveperiod

saveperiod saves the state of the simulation in the tstab interval at the simulation time interval in seconds specified by *saveperiod*. When the simulation time has passed in the tstab interval, the file specified by the option savefile is created. When subsequent simulation time intervals have passed, the file is overwritten. Use only one of the save options at a time. The example below shows *saveperiod* set to 25 nanoseconds.

```
saveperiod    25n
```

Savetime

savetime is a list of times in the tstab interval where the state of the simulation is written out. The list should be specified with spaces between the entries. The information is written out with the filename specified in the *savefile* option with extensions for the time after that. Use only one of the save options at a time.

The example below shows four times in the tstab interval to save.

```
savetime    50n 100n 125n 150n
```

Savefile

This is the file name to write out the tstab state information to. If you do not specify a filename that starts with the slash (/) character, the file is stored in the *netlist* directory. To locate the *netlist* directory, first select *Setup - Simulator/Directory/Host* menu in ADE, and read the path to the project directory. This is the path to the *simulation* directory. In the simulation directory, navigate to the `<circuit_name>`*/spectre/<schematic or config>/netlist* directory.

Note: Do not specify a relative pathname like *./save_file/run1*.

Recover (Shooting and Harmonic Balance)

recover specifies the file name to recover the tstab simulation from. If *saveperiod* or *saveclock* are used to make the savefile, then the same name as specified in *savefile* is used. If *savetimes* has a list of times specified, then several files are created at the times specified in the list. Click (...) and browse to the netlist directory where the savefiles are shown.

Select one from the list, and click *Open*. This adds the full path to the *recover* option, as shown below. Also note that in the example below, the *savefile* option is still set, and times after the restart times are specified. This is specifically allowed. As long as tstab is 200n or larger, the files at 175n and 200n will be added and can be reused later.

Now change tstab to a later time. In this case, 200n is set.

When the simulation runs, the savefiles are created, as shown below.

The waveform from the original tstab and the restarted tstab overlay is shown below.

Even in the zoomed-in view, the waveforms overlay, as shown below.

Misc Tab

New Initial Value For Each Point (restart)

Restart is an option that controls the behavior of the simulator in a sweep or in Monte Carlo. If restart is yes, then the simulation restarts from scratch for every sweep point. This can cause excessive runtimes and/or convergence difficulties. When restart is no, the previous member of a sweep or Monte Carlo is used as the starting point for this analysis. If this point does not converge, a variety of methods are applied automatically to try to get convergence. These methods include parameter sweeping, backing off the source power and then ramping it back up in increments, and running a tstab simulation.

AdditionalParams

This is a field for `<option_name>=value` statements. Multiple `<option_name>=value` statements are allowed with a space between them. Generally, this is used to unlock the beta features, however, there is one case where this field might be useful for an option that is not in the GUI.

In some cases, in the tstab interval, the timestep collapses to near zero. The symptom is that the percent complete stays at the same value for many reporting intervals in the Spectre output window. In this case, there might be a discontinuity in one of the models.

In transient simulation, two things can reduce the timestep; having too much numerical integration error or taking too many iterations at a single timestep can cause the timestep to be cut. Spectre does not report what is causing the timestep to become small. One way to eliminate the possibility of numerical integration error (which is called local truncation error in

the simulator) as the cause of reducing the timestep is to set *lteratio* to 1e9. The disadvantage of this is that it disables the normal method of timestep control, therefore, *maxstep* must be set to maintain accuracy.

Another way to accomplish this is to set *lteminstep* to a value of about the stop time divided by 1e5. This allows the normal method of timestep control, but if for some reason the numerical integration error wants to cut the timestep too small, the error is ignored. *lteminstep* is the way to specify this. To do this in the ADE, type `lteminstep=<Stop_Time>/1e5`. If you use `lteratio=1e9`, or if you set *lteminstep* and you still have very small timesteps, try increasing *maxiters* in the *Convergence* tab to between 40 and 100. When *lteratio* or *lteminstep* is set and you still have timestep problems, there is likely a discontinuity in either a device model or in a Verilog-A model. An example is shown below.

Oscillator Options

1. *pinnode* can be set manually. Do not start the name with a slash. If you have a hierarchical name, it will usually be in the form *I1.I10.out1*. If you use the probe-based method, do not specify any entry here.

2. The *save osc ppv* option is used with the PLL methodology. Refer to Noise-Aware PLL Flow on page 1383 for the PLL methodology.

Key Outputs in the Spectre Window

First, the circuit inventory is displayed. In addition, you also get a message that the linear oscic is being used. This is controlled by the *Calculate initial conditions (ic) automatically* check box.

Next, you get the output that either indicates that the linear oscic worked, and the estimate of the frequency, or that it failed, and is running tstab.

The time of the tstab interval is reported. If *Run transient?* is set to *Decide automatically*, you might see one or more messages that indicate the tstab interval is being extended. This usually is only seen for oscillators.

For oscillators, the tstab interval includes the time set in tstab plus five periods of the frequency estimate from linear oscic. If linear oscic is not used, it includes five periods of the frequency you estimated in the *Choosing Analyses* form.

The list of the options that are actually used for this phase is listed.

The transient output from the tstab interval is next.

```
/usr2/rdavis/demos/ic61/spectrerf_aps_workshop/simulation/oscillat
File  Help                                                    cadence

Linear IC: estimated frequency is 1.75949e+09 Hz

================================
`hb': time = (0 s -> 7.63158 ns)
================================
Important parameter values in tstab integration:
    start = 0 s
    outputstart = 0 s
    stop = 7.63158 ns
    step = 5.52632 ps
    maxstep = 21.0526 ps
    ic = all
    skipdc = no
    reltol = 1e-03
    abstol(V) = 1 uV
    abstol(I) = 1 pA
    temp = 27 C
    tnom = 27 C
    tempeffects = all
    method = traponly
    lteratio = 3.5
    relref = sigglobal
    cmin = 0 F
    gmin = 1 pS

    hb: time = 202.6 ps    (2.65 %), step = 11.98 ps     (157 m%)
    hb: time = 214.5 ps    (2.81 %), step = 11.93 ps     (156 m%)
    hb: time = 216.6 ps    (2.84 %), step = 2.121 ps    (27.8 m%)
    hb: time = 220.9 ps    (2.89 %), step = 4.243 ps    (55.6 m%)
    hb: time = 229.4 ps    (3.01 %), step = 8.485 ps     (111 m%)
    hb: time = 235.7 ps    (3.09 %), step = 6.366 ps    (83.4 m%)
    hb: time = 241.6 ps    (3.17 %), step = 5.84 ps     (76.5 m%)
    hb: time = 248.6 ps    (3.26 %), step = 7.052 ps    (92.4 m%)
    hb: time = 255 ps      (3.34 %), step = 6.414 ps      (84 m%)
    hb: time = 265.9 ps    (3.48 %), step = 10.84 ps     (142 m%)
    hb: time = 274.4 ps     (3.6 %), step = 8.578 ps     (112 m%)
    hb: time = 286 ps      (3.75 %), step = 11.54 ps     (151 m%)
    hb: time = 297.8 ps     (3.9 %), step = 11.83 ps     (155 m%)

6  HelpAction
```

Next, one cycle of the oscillator is simulated.

```
/usr2/rdavis/demos/ic61/spectrerf_aps_workshop/simulation/oscillat
File  Help                                                    cadence

Linear IC: estimated frequency is 1.75949e+09 Hz

==================================
`hb': time = (0 s -> 7.63158 ns)
==================================
Important parameter values in tstab integration:
    start = 0 s
    outputstart = 0 s
    stop = 7.63158 ns
    step = 5.52632 ps
    maxstep = 21.0526 ps
    ic = all
    skipdc = no
    reltol = 1e-03
    abstol(V) = 1 uV
    abstol(I) = 1 pA
    temp = 27 C
    tnom = 27 C
    tempeffects = all
    method = traponly
    lteratio = 3.5
    relref = sigglobal
    cmin = 0 F
    gmin = 1 pS

    hb: time = 202.6 ps    (2.65 %), step = 11.98 ps     (157 m%)
    hb: time = 214.5 ps    (2.81 %), step = 11.93 ps     (156 m%)
    hb: time = 216.6 ps    (2.84 %), step = 2.121 ps    (27.8 m%)
    hb: time = 220.9 ps    (2.89 %), step = 4.243 ps    (55.6 m%)
    hb: time = 229.4 ps    (3.01 %), step = 8.485 ps     (111 m%)
    hb: time = 235.7 ps    (3.09 %), step = 6.366 ps    (83.4 m%)
    hb: time = 241.6 ps    (3.17 %), step = 5.84 ps     (76.5 m%)
    hb: time = 248.6 ps    (3.26 %), step = 7.052 ps    (92.4 m%)
    hb: time = 255 ps      (3.34 %), step = 6.414 ps      (84 m%)
    hb: time = 265.9 ps    (3.48 %), step = 10.84 ps     (142 m%)
    hb: time = 274.4 ps     (3.6 %), step = 8.578 ps     (112 m%)
    hb: time = 286 ps      (3.75 %), step = 11.54 ps     (151 m%)
    hb: time = 297.8 ps     (3.9 %), step = 11.83 ps     (155 m%)

6  HelpAction
```

At the end of the one cycle, a Fourier transform is done to serve as the starting point in the frequency domain, and a node is picked for the pinnode.

The frequency domain iterations start.

```
/usr2/rdavis/demos/ic61/spectrerf_aps_workshop/simulation/oscillat
File  Help                                                    cadence

    hb: time = 8.175 ns     (100 %), step = 3.396 ps     (625 m%)
Pinning node: 13, harm: 1, name: Q0:int_e, value: (0.844012, -0.33509

==============================
     Harmonic balance
==============================
Important HB parameters:
    RelTol=1.00e-05
    abstol(I)=1.00e-12 A
    abstol(V)=1.00e-06 V
    ResidualTol=1.00e+00
    LTE_Ratio=1.00e+01
    Steady_Ratio=1.00e+00
    MaxNewtonIter=100


********** initial residual **********
Resd Norm=5.51e+04  at node Q0:int_b  harm=(10)

********** iter = 1 **********
Delta Norm=2.27e+02  at node Q0:int_e  harm=(0)
Resd Norm=1.96e+04  at node Q0:int_b  harm=(10)
Frequency= 1.8405e+09 Hz, delta f= 4.70e+03

********** iter = 2 **********
Delta Norm=2.54e+02  at node IPRB0:in  harm=(1)
Resd Norm=3.46e+03  at node Q0:int_b  harm=(8)
Frequency= 1.8327e+09 Hz, delta f= -7.81e+06

********** iter = 3 **********
Delta Norm=3.55e+01  at node IPRB0:in  harm=(6)
Resd Norm=5.55e+02  at node Q0:int_b  harm=(10)

6  HelpAction
```

The iterations continue until the delta norm and the residual norm are both less than one.

The frequency is reported. The times and memory use are reported.

The circuit inventory, pin node, and linear oscic are reported.

The linear oscic frequency is reported.

The options used for this part of the simulation are reported.

The probe-based method inserts a source connected between the pinnodes. The amplitude and frequency are adjusted until the current is near zero.

If the probe-based method is used, the initial starting point is determined. The outer loop iterates on the frequency and the inner loop iterates on the amplitude.

The iterations continue until the source current is near zero, the sum of the currents at all the nodes is near zero, and the delta between iterations is near zero.

The tolerances are set by using the errpreset setting (liberal, moderate, conservative).

The final frequency is reported. The time and memory statistics are shown.

Examples

RF source setup for all examples in this chapter

The *Edit Object Properties* dialog for the port that is used as the RF source is shown below.

Property
Value
Display
Library Name
analogLib
Cell Name
port
View Name
symbol
Instance Name
RF
off
Add
Delete
Modify
User Property
Master Value
Local Value
Display
IvsIgnore
TRUE
CDF Parameter
Value
Display
Port mode
Normal
HarmonicPort
Resistance
50 Ohms
Reactance
Port number
1
DC voltage
Source type
sine
Frequency name 1
frf1
Frequency 1
frf1 Hz
Amplitude 1 (Vpk)
V
Amplitude 1 (dBm)
prf
Phase for Sinusoid 1
Sine DC level
Delay time
Display second sinusoid
Frequency name 2
frf2
Frequency 2
frf2 Hz
Amplitude 2 (Vpk)
Amplitude 2 (dBm)
prf
Phase for Sinusoid 2
Display multi sinusoid
Display modulation params
Display small signal params
Display temperature params
Display noise parameters
Multiplier
OK
Cancel
Apply
Defaults
Previous
Next
Help

In the *Edit Object Properties* dialog:

1. If you select *Names* in the *Choosing Analyse*s form, specify a name in the *Frequency name 1* and *Frequency name 2* fields.
2. Set the first sinusoid frequency to a variable name in the *Frequency 1* field.
3. Set the *Amplitude 1 (dBm)* field to a variable name.
4. Set the *Frequency* 2 field to a variable name.
5. Set the *Amplitude 2(dBm)* field to a variable name used to set the first frequency amplitude, if you want the amplitudes to match. If you do not want the amplitudes to match, you can use a different variable name.
6. Select the *Display small signal params* check box.
7. Set the *PAC Magnitude* field to 1 volt.

Note: Using variables in the entries enables you to change the properties without needing to change the schematic. Although this is useful mostly for ADE L, it works in ADE XL also. ADE XL allows any property in the schematic to be varied directly without changing the schematic.

ADE Setup for 1-tone Harmonic Balance (hb) Analysis

The Analog Design Environment (ADE) window is shown below.

Note: The variable name *frf1* is set to `900M` which sets the first frequency in the input port to 900MHz. The variable *prf* sets the input power for the RF input. The second frequency is zero, which disables that input.

To set the 1-tone hb analysis:

1. Open the *Choosing Analyses* form from the ADE window.

2. Select *hb*.

3. You can specify either the frequencies of the inputs, or get a list of the source names and frequencies in the schematic.

Frequencies is select by default for hb analysis.

4. *Decide automatically* is the default for *Run Transient?*. This is suggested. Except for the frequency, this will set everything to automatic. If you select *Yes*, you will have the option of setting the tstab (transient-aided) time, and the option to automatically manage the tstab stop time.

5. Select *1* to set the number of tones to 1.

6. Set the frequencies, number of harmonics, and oversample factor.

   auto calculates the number of harmonics that are needed based on the Fourier Transform at the end of the tstab interval, and is available for the *Decide automatically* and *Yes* selections for *Run transient?*. If the circuit has square wave signals and you are setting harmonics manually, set the number of harmonics to at least the period of the signal divided by the risetime and set the oversample factor to between 2 and 8.

7. Select the *moderate* check box.

   If you need more accuracy, select the *conservative* check box.

8. When you are done with the setup, run the simulation.

1. When *Names* is selected, all the sources in the circuit are read into the table along with their frequencies. In this case, there is one source with the frequency set to 900M using the variable *frf1*.

2. To make changes, select the line and set the number of harmonics and the oversample factor in the fields just below.

3. Click *Change,* select another field, or press the <*Tab*> key.

4. When you are done, run the simulation.

   Spectre will calculate the number of harmonics and display the result in the Spectre output window.

Plotting Spectral Content

1. In the ADE window, select *Results - Direct Plot - Main Form*.

2. Select the *hb* radio button from the *Analysis* section.
3. Select the Voltage radio button from the *Function* section.
4. Select the *spectrum* radio button from the *Sweep* section.
5. Select the *rms* radio button.

6. Select the *dB20* radio button from the *Modifier* section.
7. If you want this signal to plot automatically when a new simulation is run, select *Add To Outputs*.
8. Select the node that you want to plot on the schematic.

The output harmonics are shown in the waveform tool.

The dB20 modifier is dB with respect to 1 volt, or dBV.

To position the markers, move your cursor near the object you want to read, and type *m*.

Plotting dBm

On the *Direct Plot Form*:

1. Select *hb* from the *Analysis* section.

2. Select the *Power* radio button from the *Function* section.
3. Select the *Net (specify R)* option from the drop-down list.
4. Select *dBm* from the *Modifier* section.
5. Select the appropriate net on the schematic.

6. Click *OK*.

The output spectrum in dBm is displayed.

To position the markers, move your cursor near the thing you want to read, and type *m*.

Sweeps

1. To enable sweeps, select the *Sweep* check box in the *Choosing Analyses* form.

2. Select what you want to sweep from the drop-down list. In the figure above, *variable* is selected.
3. Click *Select Design Variable* and select the variable that you want to sweep.
4. Specify the sweep range in the *Start* and *Stop* fields.
5. If desired, select the *Add Specific Points* check box and specify the additional points.

   Separate the entries with a space.

Plotting Currents

Currents can be plotted from the terminals of the sources in the circuit only. If you want to plot a current in a branch of your circuit, add an iprobe to the circuit in series with the branch. An iprobe is a zero volt voltage source that also allows the current in that branch to be solved.

To plot the currents, select the *Current* radio button from the *Function* section in the *Direct Plot Form* and click *OK*.

Virtuoso (R) Visualization & Analysis XL
File Edit View Graph Axis Trace Marker Measurements Tools Window Browser Help
cadence
Layout: Auto
Workspace: Classic
RF_APS DLNA1_RFT_tb1 schematic
Harmonic Balance Response
Name
... h=1; hb dB20(
-30.0
-35.0
-40.0
-45.0
-50.0
-55.0
peak (dB)
-30.0
-25.0
-20.0
-15.0
-10.0
-5.0
prf
mouse L:
M:
R:
6(11)
no error

Loadpull

For a loadpull simulation, add a triplexer (for setting the reflection and angle for up to three harmonics) or the ten_plexer (for up to 10 harmonics) as the load to the circuit. These are in the `rfLib` library.

Normally, properties are used to set the reflection coefficient magnitude and phase for the harmonics that are present in the triplexer or ten_plexer. The example below shows the ten_plexer, and has 0 (zero) set for all the reflection coefficients and angles.

Descend-read into the triplexer or ten_plexer. Triplexer is shown. The top bandpass filter is for the first harmonic. The middle is the second harmonic. The bottom is the third harmonic.

1. Add two variables in ADE.

The example above shows `mag` and `theta` variables added. These variables should not be used to set any component values in the circuit.

Triplexer from `rfLib` directory contains three filters for the first, second, and third harmonics and portAdapters with port loads for each of the three filters.

2. Open the pss or hb *Choose Analyses* form.

3. Select the *Loadpull* checkbox.

4. Click the *Select* button on the right of the greyed out *Load Instance* field.

5. In the triplexer schematic, select the topmost port on the right side of the schematic. if you want a loadpull for the first harmonic. Select the next port down for a loadpull of the second harmonic. Select the third port down for the third harmonic.

   The instance name is entered in the *Load Instance* field. Direct entry by typing is not allowed for this field.

6. Click the *Variable* button on the right of the *rho (Z)* field. Select the variable you entered in ADE.

   This variable will be used to set the value of the reflection coefficient in the port.

7. Specify a sweep range, and set the spacing.

8. When completed, click *OK*.

   The values are displayed in the *Choosing Analyses* form.

9. Click the *Variable* button on the right of the *phi (Z)* field. This sets the angle of the reflection coefficient.

10. In the *Select Design Variable* window, select the second variable you entered and specify the sweep limits.

11. When completed, click *OK*.

The *Choosing Analyses* form is updated.

12. Specify the system resistance in the *Z0* field.

The *Choosing Analyses* form should appear similar to the one below.

Choosing Analyses -- Virtuoso® Analog Design Environn
Analysis
tran
dc
ac
noise
xf
sens
dcmatch
stb
pz
sp
envlp
pss
pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
hbsp
Harmonic Balance Analysis
Transient-Aided Options
Run transient?
Decide automatically
Detect Steady State
Stop Time(tstab)
auto
Save Initial Transient Results (saveinit)
no
yes
Tones
Frequencies
Names
Number of Tones
1
2
3
4
Tone 1
Fundamental Frequency
2.45G
Number of Harmonics
auto
Oversample Factor
1
Freqdivide Ratio for Tone 1
Harmonics
Default
Accuracy Defaults (errpreset)
conservative
moderate
liberal
Oscillator
Sweep
Loadpull
Load Instance
/I8/PORT14
Select
rho (Z)
mag
{1m:991m}step:110m
Variable ...
phi (Z)
theta
{-180:160}step:20
Variable ...
Z0
50
Enabled
Options...
OK
Cancel
Defaults
Apply
Help

13. When completed, click *OK* and run the simulation in ADE.
14. In ADE, select *Results - Direct Plot - Main Form*.

There are two methods of plotting loadpull curves.

First Method

1. Select *pss or hb results*.

2. Select *Power Contours*.

3. Select *Magnitude*, *dB10*, or *dBm*.

4. Specify 2 more than the number of contour lines you want.

   The smallest and largest power occurs at a single point which is plotted, but are very difficult to see in the waveform tool.

5. Select the *Close Contours* check box. This causes the contours from the loadpull to be displayed as circles or ellipses, depending on the distortion of the circuit.

6. Select the first harmonic.

7. Select the top terminal of the top port in the triplexer.

The power contours appear.

Second Method

1. Select *pss* or *hb results*.

2. Select *Power*.

3. Select *variable*.

4. Select *Magnitude*, *dB10*, or *dBm*.

5. Select the *harmonic number*.

6. Select *Loadpull Contour*.

7. Specify 2 more than the number of curves you want.
8. Select the *Closed Contours* check box. This causes the contours from the loadpull to be displayed as circles or ellipses, depending on the distortion of the circuit.
9. Select the top terminal of the top port of the triplexer circuit.

   The loadpull curves appear in the waveform tool.

Power-Added Efficiency

1. Set up an hb run as desired. A sweep is shown below.

2. Run the simulation.

3. When the simulation completes, select *Results - Direct Plot - Main Form*. The *Direct Plot Form* is displayed.

4. Select *Power Added Eff.*

5. Set the number of DC sources in the circuit you want to include for the PAE measurement.
6. Select the frequency of the output.
7. Select the top terminal of the load in the schematic.
8. Select the top terminal of the input port.
9. Select the positive terminal of the DC source that supplies power to the amplifier. If you set multiple sources, select the positive terminals of all the sources.

   The PAE is plotted in the waveform tool.

Oscillators

On the *Choosing Analyses* form:

1. Set the *Run Transient?* selection appropriately.

2. For oscillators, selecting the *Detect Steady State* check box is strongly recommended. This will run tstab until steady-state is reached, and then switch to harmonic balance. When *Decide automatically* is set for *Run Transient?*, *Detect Steady State* is always on.
3. Specify a frequency that is within 0.5 times the actual oscillation frequency and 1.5 times this frequency.
4. Set the harmonics and the oversample factor as needed for your oscillator. *auto* harmonics is available when *Run Transient?* is set to *Decide automatically* or *yes*.
5. Always select *conservative*.

   For LC and crystal oscillators, no further action is necessary. For ring oscillators, specify the initial conditions in ADE to force one stage to the high or low state.
6. If you want to see the startup waveform, select *yes* for saveinit.
7. Select any net with the oscillator signal present for the oscillator node.
8. If you want the output frequency versus tuning voltage, define an appropriate sweep for your oscillator.
9. Run the simulation.

Oscillator Tuning Range

To plot the tuning range, on the *Direct Plot Form*:

1. Click the *Harmonic Frequency* radio button.
2. Select the *Add to Outputs* check box.
3. Select the desired harmonic number to plot from the *Harmonic Frequency* field.
4. Click *Plot*.

The tuning range plot appears.

Plotting Modulation Sensitivity

From plotting the oscillator frequency as shown below, an expression is added in the ADE window.

1. Double-click the expression.

The *Setting Outputs* window is displayed.

2. Click *Open*.

The Calculator starts with the expression in the buffer. This expression will be different for different circuits.

3. Set the plotting mode to *New Subwindow*.
4. Select the *deriv* function.

The expression changes in the buffer.

5. Click the *Plot* icon.

The modulation sensitivity appears in the new subwindow.

Two Input Frequencies

To enable an additional frequency, set the variable in ADE to the frequency you want instead of 0 (zero). In the window below, the variable frf1 has been set to 1.901G.

To set the 2-tone hb analysis:

1. Open the *Choosing Analyses* form from the Analog Design Environment (ADE) window.

pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
hbsp
Harmonic Balance Analysis
Transient-Aided Options
Run transient?
Decide automatically
Detect Steady State
Stop Time(tstab)
auto
Save Initial Transient Results (saveinit)
no
yes
Tones
Frequencies
Names
Number of Tones
1
2
3
4
Tone 1
Tone 2
Fundamental Frequency
1.9G
1.901G
Number of Harmonics
auto
3
Oversample Factor
1
1
Tone 1 be LO or signal which causes the most nonlinearity.
Freqdivide Ratio for Tone 1
Harmonics
Default
Accuracy Defaults (errpreset)
conservative
moderate
liberal
Oscillator
Sweep
1
Variable
Frequency Variable?
no
yes
Variable Name
prf
Select Design Variable
Sweep Range
Start-Stop
Center-Span
Start
-30
Stop
-10
Sweep Type
Linear
Logarithmic
Step Size
Number of Steps
5
Add Specific Points
-8 -7.25 -6.5 -5.75

2. Decide whether you want to set the harmonics manually or automatically for the first tone. For auto tstab, and auto selection of harmonics for the first tone, select *Decide automatically* from the *Run transient?* drop-down list.
3. You can specify either the frequencies of the inputs, or obtain a list of the source names and frequencies in the schematic. *Frequencies* are shown above.
4. Click the appropriate radio button to select the number of inputs.
5. Set the frequencies, number of harmonics, and oversample factor. For an automatic calculation of the number of harmonics for the first tone, set the number of harmonics to auto in Tone1.
6. If you have a frequency divider in your circuit, set the divide ratio. This assumes that the Tone1 input will be divided in frequency.
7. If you want to enable a frequency cut, choose *Select* from the *Harmonics* drop-down list box and set a value for *MaximOrder*.
8. Set the accuracy you want for the analysis.
9. If you want to manually set transient-aided hbhb, select *Yes* from the *Run Transient?* drop-down list and set a time for the transient tin the *Stop Time (tstab)* field. If you want to see the startup waveform, select *yes* for saveinit.

10. Enable sweeps by checking the *Sweep* checkbox, or use the parametric plot tool in ADE. In the *Choosing Analyses* form, you can specify up to three variables. In the parametric plot tool, the number of variables is unlimited.
11. Set the variable name to be swept.
12. Set the sweep range and step information.
13. If you want to run specific points, check the appropriate check box at the left side of the *Add Specific Points* field.

The form below shows the hb *Choosing Analyses* form with *Names* selected. Only the differences are shown from the previous section.

14. When *Names* is selected, ADE reads the schematic and lists all the sources in the circuit.

The *Expr* field lists the entry that sets the frequency, and the values field evaluates the variable expression and lists the value.

Mxharm lists the number of harmonics for that tone. *auto* is allowed if *Decide automatically* or *Yes* are selected from the *Run transient?* drop-down list.

Ovrsap lists the oversample factor for that tone.

Tstab specifies which tone is eligible for transient-aided hb.

Important

Tstab should be enabled for the tone that causes the most distortion in the circuit.

15. To make changes, select the line and then edit the entries in the fields just below.

16. When you are done with your edits, click *Change,* select another field, or press the <*Tab*> key.

17. When you are done with the setup, run the simulation.

18. When the simulation completes, select *Results - Direct Plot - Main Form* in the ADE environment.

The form looks as below:

The *Plotting Mode* drop-down list box has the following options:

- *Append* - Add the new plot to an existing plot in the selected subwindow.
- *Replace* - Replace the data in the selected subwindow.
- *New SubWin* - Open a new subwindow and plot the data.
- *New Win* - Open a new window and plot the data.

19. Select the analysis type you want to plot from the list.

The figure below shows three different analyses. *_mt* means multiple tone.

20. Select the specific item you want to plot from the list in the *Function* section. This example shows plotting the voltage on a node.

21. Different items can be plotted by using the *Select* drop-down list. You can choose from a number of different things by clicking the *Select* drop-down list. For example, *Differential nets*.

22. If you want to plot the spectrum at a specific power level, select *spectrum* or *frequency*. If you want to see a swept measurement of a specific harmonic, select *variable*. If you want to plot the time-domain waveform at a specific power level, select *ifft*.

23. For a harmonic plot, select *peak* or *rms*.

24. Select the data type you want to plot. If you want *dBm*, select *Power* in the *Function* section.

25. Select the desired harmonic or power level (as appropriate for your selection in the sweep section)

26. If you want to add an expression to the ADE outputs section so that it plots automatically after a simulation, select *Add To Outputs*.

At any time when filling the *Direct Plot Form*, appropriate directions on what to do are given at the bottom of the form. In this example, a variable value needs to be selected in the variables table.

This is the wide range display of all the harmonics. *For more information on the waveform tool, see the documentation for Wavescan (IC5.1) or ViVA (IC6.1).*

This is the narrow range harmonic display. Click the right mouse button to zoom.

Compression Point

Follow the steps below to get the compression point curve on the *Direct Plot Form*:

1. Select *Compression Point*.

2. Select the desired value (port, net, or differential net) from the *Select* drop-down list.

3. Select whether you want the output power or the gain plotted on the Y axis as a function of the input power from the *Format* drop-down list.

4. Select the desired amount of compression in the *Gain Compression (dB)* field. For small amounts of compression you might need to use conservative in the hb *Choosing Analyses* form.
5. Specify the desired level of input power to begin drawing the 1dB down ideal 1dB per dB curve. The default is to drop down the amount of compression specified at the lowest power point of the sweep.
6. Select the desired output-referred or input-referred compression point.
7. Select the desired frequency of the output harmonic.

This is the compression plot. The ideal 1dB per dB curve is drawn and the compression point is shown at the upper right of the plot.

IFFT

To get the time-domain waveform, select the following values on the *Direct Plot Form*:

1. Select *ifft*.
2. For multi tone simulations, the default is to filter out the higher harmonics of the hb analysis in order to improve the plotting speed.
3. Select the desired input power for the ifft plot.

This results in a good waveform that plots quickly. Only the lower frequency harmonics are present in this plot.

To get the best waveform, select the *Complete ifft* checkbox. The plotting time can be considerable for 3 and 4 tone simulations.

Note that not all the harmonics are present in the upper (default) waveform. You choose if you want a quick idea or the plot with all the harmonics. If you want the higher order harmonics in

the waveform, select the *Complete ifft* checkbox in the *Direct Plot Form*. The full waveform and the zoomed-in views are shown below.

Virtuoso (R) Visualization & Analysis XL
File Edit View Graph Axis Trace Marker Measurements Tools Window Browser Help
cādence
Layout: Auto
Workspace: Classic
WorkshopLib db_mixer schematic
Name
...1 prf=-30; hb_n
...1 prf=-30; hb_n
2.56
2.555
2.55
2.545
2.54
2.535
2.53
(V)
597.0
598.0
599.0
600.0
601.0
602.0
603.0
time (ns)
mouse L:
M:
R:
16(30)
RMB zoom reset to xy mode.

Compression Vs Pout

Another thing you can plot from a sweep is the compression versus the output power. To see this, select *Comp Vs Pout*.

Several different types of data can be selected. For example, a net, or differential nets.

The zero compression input power is set in the field as shown below. The default is the lowest power point of the sweep.

The frequency of the harmonic to be plotted is selected here.

Select the port in the circuit.

The output is displayed as below.

Example Using the Harmonic Port with Harmonic Balance

The port component has a feature called the harmonic port that allows the specification of different impedances at different harmonic numbers. This is avaialble for harmonic balance large-signal analyses only, and does not work with any other analysis.

For this example, a 2.4GHz LNA is the example circuit.

The input and output ports are set up normally, as shown in the properties forms below.

PORT1 is the input and PORT2 is the output.

One Input Frequency

The harmonic balance *Choosing Analyses* form has a single input applied.

To use the harmonic port, select *Harmonic Port* on the *Edit Object Properties* form. In this case, the port being used as a load in the circuit is selected. The *Edit harmonic port* window is displayed.

For this setup, a single input frequency is applied to the circuit. In this case, harmonic numbers 1, 2, 3, 4, and 5 exist in the simulation result. In the *Harmonic* column, you enter the harmonic number. In the *Resistance* column, you enter the resistance. In the *Reactance* column, you enter the series reactance in ohms. Positive reactance adds inductance, and negative reactance adds capacitance. Multiple entries are allowed,

therefore, different impedances can be set for each harmonic. The resistance or reactance can be left blank, which is interpreted as zero resistance or reactance.

Here is an example where the third harmonic is set to 1 ohm in series with +j1mohm. This should lower the amplitude of the third harmonic.

The top trace is the normal port, and the bottom is the trace with the third harmonic set to 1ohm in series with +j1milliohm. Note that the thrd harmonic is smaller.

Two Input Frequencies

Now the harmonic balance *Choosing Analyses* form has two frequencies.

Also, the diamond cut is set with a very large *MaximOrder*. The reason for doing this is to see the harmonic numbers that corrrespond to the frequencies produced by the circuit. Each frequency has 2 harmonic numbers that indicate which harmonics are mixing with which to produce a specific output frequency. For example, take the 2.44GHz term. The harmonic number is 1 0. The frequency that is produced is 1*2.44G + 0*2.46G. Now look at the intermod at 2.42GHz. This index is 2 -1. The frequency is 2*2.44G - 1*2.46G. These are the harmonic numbers that can be set for the harmonic port. Wild cards (*) are not allowed in the harmonic number specification.

Note that if there were three input frequencies, there would be three numbers that specify every output frequency.

In this case, if the harmonic was specified as 3, there would be an error produced by spectre because there is a single index in the harmonic, but there are 2 indices for all the harmonics of the existing simulation. The number of numbers in the harmonic field must match the number of input freqeuncies in the circuit.

The above example sets the impedance of the third order intermods to be much lower than 50 ohms. If the source is in series, setting the resistance and/or reactance to a large number can block harmonics you do not want. If the port is in parallel like this one, setting small

resistance and reactance values can short out an undesired harmonic. This can be used for debugging the circuit, or for setting values to maximize power-added efficiency.

The top trace above is from the normal port, and the bottom trace is from the harmonic port. The third order harmonics are much smaller because they are essentially shorted out.

Three Tone Harmonic Balance

Generally, a 3 tone harmonic balance simulation is used to calculate large-signal IP3 for transmit applications. If you need a small-signal IP3 measurement, see Rapid IP3 in the hbac section of this chapter. For a small-signal IP3 measurement, Rapid IP3 is much faster and is usually more accurate than a 3 tone harmonic balance simulation.

In a 3-tone simulation, first decide if you want to remove the higher order terms from the solution. If you remove the high order harmonics from the simulation, it can run faster because fewer harmonics are calculated.

On the Choosing Analyses form:

1. To trim the high-order harmonics, choose *Select* from the *Harmonics* drop-down list.

   Funnel is recommended for systems where one or more signals cause a lot of distortion compared to the rest of the signals. Generally, these systems are mixers or sampling circuits that have large amplitude LO or sampling clock signals. *Funnel* calculates the harmonics whose order is less than that specified in *MaximOrder* from the simulation and adds the harmonics of the LO and RF tones but not the higher order mixing products.

   Funnel cuts the harmonics whose order is higher than that specified in *MaximOrder*, but it does not cut the lower-order harmonics of the mixing products. For example, you might set funnel and *MaximOrder* to six. This calculates the sixth order and lower mixing products, but allows the higher order (up to the 10 harmonics specified for Tone1) LO harmonics to be calculated. This is useful for mixer simulations where you know that the LO produces relatively more harmonics than you want with *MaximOrder*. Setting *maximOrder* to six is the recommended starting point for a large-signal IP3 measurement in a down-convert mixer.

2. To see the actual harmonic frequencies to be run in a list, type a frequency range in the fields next to *Harmonics* drop-down list.

3. Click *OK* and start the simulation.

Large-Signal IP3

To plot a large-signal IP3, in the *Direct Plot Form*:

1. Select *IPN Curves*.
2. Select the *Variable Sweep* radio button.
3. Select the input power level you want to do the third order projection from. The default is the lowest power of the sweep.
4. Select Input or Output referred from the drop-down list.
5. Select the order from the *Order* drop-down list. *3rd* order is the default.
6. Select the 2nd or 3rd order frequency in the *2nd Order Harmonic or 3rd Order Harmonic* fields.
7. Select the first order frequency in the *1st Order Harmonic field*.

The IP3 calculation (Input Referred IP3 = 4.71 dBm in the plot below) looks like it is tied to the last datapoint of the first order curve. It is tied there so it will be displayed on the screen. It actually calculates the intercept point.

The extrapolation point (ep = -50 in the plot below) is the input power level that the IP3 calculation is done from. This is the input power level where the ideal curves and the measured data overlay.

In the IP3 plot for a small-signal IP3 measurement, look for the ideal curves and the data to nearly overlay each other on the third order curve for at least a 10dB range of input power. The plot above shows the data and the ideal curves being essentially the same from -50 to about -15 dBm.

The bottom right corner of the plot shows the following in the order as listed below:

- Output of the circuit.
- Third order frequency.

- First order frequency.

Note that the above IP3 is really a small-signal IP3 measurement. For a much faster IP3 calculation use Rapid IP3 as documented in the hbac and pac sections. For a large-signal IP3 calculation, set the extrapolation point to the desired input power for the large-signal measurement as shown below in the direct plot form.

Now either select replot, or select the output port in the schematic.

Note that the IP3 calculation has changed from 4.76dBm to 2.17dBm.

Multi-Sinusoid hb Analysis

The port component has been enhanced to support the generation of up to nine sinusoids simultaneously. To do this, *Frequency* 2 needs to be set to the spacing frequency. All the sinusoids need to have equal spacing frequencies. *Frequency 1* in the port needs to be set to the center frequency of the multiple output frequencies. The center frequency needs to be one of the frequencies that is generated. For example, imagine that 1GHz, 1.01GHz, and 1.02GHz need to be generated. The center frequency in this case is 1.01GHz and the

spacing is 10MHz. Now imagine that 1GHz, 1.01GHz, 1.02GHz, and 1.03GHz need to be generated. Now the center frequency could be either 1.01GHz or 1.02GHz.

Once the center frequency and spacing are defined, click *Display Multi Sinusoid*. Next, specify the number of carriers you want to generate. There is a maximum limit of nine in one port. If you need more than nine tones, add more ports in series, all with the same center frequency and spacing. Specify the frequencies you need to generate in the additional ports. Divide the port resistance desired by the number of ports you have in series. Note that when you do this, you will need to scale the power in dBm for all the tones in the multi sinusoid because the voltage is set in the port assuming there is a match with the port resistance on the port output. In the case of stacking, the overall resistance is the desired port resistance.

Important

The phase unit for the multi-sinusoid outputs is radians, not degrees.

Once the input frequencies are defined, fields are displayed to specify the frequency and amplitude of each sinusoid to be generated. All the frequencies must be either the center frequency specified in the *Frequency* 1 field, or *Frequency 1* with integer spacing frequency offsets from frequency one.

Below is an example of the port setup to generate three signals.

Note that the frequency 2.45G is not generated for this test. For a composite triple beat measurement, one of the frequencies will be missing. For that frequency, just do not specify it.

When the hb analysis is set up in ADE for an amplifier, specify only two input frequencies. The center frequency and the spacing frequency should be specified in the *Fundamental Frequency* field.

For harmonics, set the total number of possible carriers in the total frequency range. In this case, there are three signals, but there is a fourth possible harmonic at 2.45GHz. In this case, the number of harmonics needs to be four for both tones.

Since all the inputs are sinusoids, *Oversample Factor* is set to one.

An example is shown below.

Now run the simulation. The output node of the circuit is plotted.

An example with five input frequencies is shown below.

If an LNA-Mixer combination is to be simulated, then there are three tones for this simulation: the center frequency, the spacing frequency, and the LO frequency. For the center and spacing frequencies, five carriers are possible, so five harmonics are set. Ten harmonics are set for the LO. The hb *Choosing Analyses* form is shown below.

Now run the simulation. The down-converted output of the circuit is shown below.

Harmonic Balance AC Analysis (hbAC)

Small-signal Versus Large-Signal Analysis for Conversion Gain

One way to measure conversion gain is to apply the LO and the RF signals in harmonic balance (hb) and calculate the full large-signal response at each RF input frequency, and then plot the output as a function of the sweep frequency. While this works, because all the large-signal effects and many harmonics are calculated, the required time for the simulation can be significant.

The alternative is to use a small-signal approach similar to the AC analysis. The AC analysis runs considerably faster than running the transient at multiple frequencies because it ignores the large-signal effects. In circuits with frequency translation, the AC analysis is unusable because the calculation is based on the DC bias point. In hbac analysis, instead of using the DC bias point, the harmonic balance solution with just the LO is used as the basis of the calculation. The large-signal analysis only calculates the harmonics of a single input, and no mixing products, and then hbac is run after the hb analysis. The overall time savings can be very significant.

The hb solution with the LO calculates the amplitude of the harmonics and it characterizes the nonlinearity of the system. Therefore, in hbac, the output mixing products of the input signal mixing with the LO harmonics can be calculated. Because the signal is considered to be small-signal, harmonics of the hbac test signal are not calculated. Like AC analysis, hbac allows the input to be swept in frequency in order to calculate a frequency response of the output mixing products.

For example, think of a perfectly linear circuit with a single input applied. Because the circuit is perfectly linear, the output spectrum that would be calculated by the hb analysis would have only the first harmonic in it. Because the circuit is linear, if a second input was applied, no frequency translation can occur.

Now imagine a real circuit with a single small input amplitude signal at 1GHz. Because the circuit has nonlinear devices, the output will contain 1GHz and the integer multiples of 1 GHz. Because the input signal is small, the amplitude of the harmonics is small, and if a second input were applied at 1.1 GHz, that signal would be very weakly converted when it is mixed with the harmonics of the circuit. Certainly, 1.1GHz would appear at the output with no mixing. 100MHz and 2.1GHz are produced by mixing with the 1GHz fundamental frequency. 900MHz and 3.1GHz are produced when 1.1GHz mixes with the second harmonic of the 1GHz input. 1.1GHz mixes with the other harmonics as well, producing a wide range of output frequencies from a single input frequency.

Now imagine that the 1GHz signal is large. The output harmonics above the first harmonic are much larger. If the input went up 10dB, the first harmonic goes up 10dB, the second

harmonic goes up 20dB, the third harmonic goes up 30dB, and so on. If a second input at 1.1GHz were applied to the circuit, this mixes with the harmonics of 1GHz and more strongly mixes up and down in frequency.

Therefore, the harmonic content of the hb analysis contains information about how strongly a circuit translates frequencies, and this is used by the small-signal analyses hbac, hbnoise, and hbsp in order to calculate the small-signal conversion gain. This section is about hbac analysis.

Example: Conversion Gain (Down conversion)

Consider the double-balanced diode mixer shown below.

The RF input source has the frequency and amplitude set to variable names in order to allow changing the test setup without needing to change the circuit. This is suggested for all the sources used in the circuit that might need to be changed.

The property list is shown below.

The easiest way to get the variables list into the ADE is to select *Variables -> Copy From Cellview*. To set the values, select the value field to the right of the variable, type the desired value, and press *Enter*.

The LO frequency is 1GHz and its amplitude is 1.4 volts peak. This is +13 dBm. The RF frequency is 1.1GHz, but it is disabled because the amplitude is set to 0 (zero). Setting either the frequency or the amplitude to zero disables the production of that waveform from the port.

The hb analysis needs to be set up first. The example below shows a single input at 1GHz with 51 harmonics. For more information, see the harmonic balance section.

Next, the hbAC analysis needs to be set up. Like AC, the frequency range specified in the form is the input frequency range. *Maximum sideband* should be left blank so that all the mixing products from all the harmonic balance harmonics are present in the output of hbac.

Once the analyses are set up, run the simulation. When the simulation completes, select *Results - Direct Plot - Main Form*.

The example below shows a typical plot.

When the net in the schematic is selected, the waveform is displayed.

Note that all the short line segments are the mixing products that are produced when mixed with the harmonics in the harmonic balance large-signal analysis.

Usually, the frequencies that are produced for the down-converted product are desired. In the *Direct Plot Form*, select *sideband* instead of *spectrum*, and then select the desired

frequency range from the list at the bottom of the form. Only that mixing product will be displayed, as shown below. The conversion loss is about 6dB.

Example: Conversion Gain (Up Conversion)

The process is similar for an up-conversion mixer. In this case the input frequency is low, as shown below.

In the *Direct Plot Form*, both the outputs above and below the LO frequency might be desired. Select the first frequency, and click the second while holding the <*Ctrl*> key. Now select the node in the circuit and the results will plot.

Note that the conversion loss is almost identical for both mixing products.

Overview of Simulation Capabilities

Specialized Analysis=None

Down Conversion: Input Near the First harmonic

At the bottom of the hbac *Choosing Analyses* form, there is a selection called *Specialized Analyses*. The default is *None*, and this is for measuring the conversion gain that is the average conversion gain with the large signal applied to the circuit (usually the LO). The mixing products that are produced are called sidebands. The sideband number is the harmonic number that is being mixed with to provide the output. A positive number means

that the output mixing product is higher in frequency than the input frequency specified in the *Choosing Analyses* form. A negative number means that the output is lower in frequency than the input frequency. A diagram is shown below.

We tend to think of positive frequencies. The default is to reflect negative frequencies to positive frequencies.

The sideband number is the harmonic that the input mixes with. Negative numbers are downconverted in frequency, and positive numbers are upconverted in frequency.

Again, in hbac, a single input frequency is applied at a time, and then the output mixing products are calculated through the highest harmonic of the harmonic balance simulation specified by the *Maximum sideband* parameter. Leaving that parameter blank is recommended, and this calculates the mixing products for all the harmonics in hb. There is no run-time penalty for calculating all the mixing products. There is a runtime penalty for increasing the number of harmonics in the hb analysis.

Note that the input causes harmonics and the output mixing products cause harmonics. Consider a mixer with a 1GHz LO and a 1.1GHz input. All the LO harmonics are calculated by the hb analysis. When 1.1GHz mixes with 1GHz, an output at 100MHz is produced. Because the circuit is nonlinear, harmonics of 100MHz are produced. hbac is a small-signal analysis, and the assumption is that those harmonics do not matter. They are not calculated.

In practical circuits, this limitation is not usually significant because generally, we work with the output frequency itself, not its harmonics.

Up-conversion: Input Near Zero Frequency

In a similar manner, up-conversion is shown below.

Frequency Sweep

This analysis is like AC, where the input frequency is swept. The input frequency range is specified in the hbac *Choosing Analyses* form.

In addition to the frequency, there is a selection called *Sweep Type*. The choices are *relative* and *absolute*. When *absolute* is selected, the frequency range is used directly with no frequency conversion. When *relative* is selected, a *Relative Harmonic* field appears. The frequency sweep is shifted by the specified frequency of the harmonic in the harmonic balance analysis. For example, assume that the harmonic balance is at 1GHz, and a log sweep is desired above the first harmonic of the hb analysis. In this case, you need to select *relative* sweep, and specify 1 in the *Relative Harmonic* field. Next, type 1K to 100M for the frequency range with a log sweep and 3 to 5 points per decade.

The sweep will start at 1G + 1K, and sweep to 1G + 100M using log spacing from 1K to 100M. This is illustrated below.

To use a log sweep below the first harmonic, use the same frequency range, and type -1 as the relative harmonic. This is illustrated below.

Maximum Sideband

Conceptually, an infinite number of mixing products are produced when a single input is applied to a nonlinear system. From the practical point of view, usually only a small number of mixing products need to be measured.

There is a property in the *Choosing Analyses* form called *Maximum sideband* that is used to define how many mixing products should be calculated. If *Maximum sideband* is zero, the hbac analysis will only contain the frequency that appears without mixing or aliasing of any type. This is not equivalent to a linear AC analysis because hbac takes into account the instantaneously varying nature of the LO signal because of the large signal being present in the hb analysis.

When *Maximum sideband* is one, the outputs without frequency translation and the outputs that mix or alias with the first harmonic are present in the hbac analysis. When *Maximum sideband* is 10, the hbac analysis includes mixing through the 10th harmonic of the harmonic balance analysis. Usually the number of harmonics in the hb analysis and the maximum number of sidebands should agree. This is easily accomplished by leaving the *Maximum sideband* field blank in the *Choosing Analyses* form in ADE.

Setting Harmonics and Sidebands

The process of setting harmonics and sidebands is similar to setting the maximum number of harmonics for the harmonic balance simulation. Start with an estimate based on the harmonic content of the input signal(s) and the nonlinearity of the circuit. Then run a simulation. Raise harmonics and sidebands and run again. If the conversion gain measurement did not change, then the original number of harmonics and sidebands might be able to be reduced. If it did change, then more are needed. Getting a stable hbac output may take more or fewer harmonics and sidebands than the harmonic balance simulation by itself.

Modulated

Modulated hbac is typically used for measuring AM to PM measurements in driven circuits and oscillators. In this case, the input is usually the power supply ripple at low frequency, and the desired output is near the first harmonic of the output. Hbac apples a modulated input signal to the circuit, and is able to measure a modulated output signal from the circuit. The types of modulation are Single Side Band (SSB), AM, and PM.

A diagram is shown below for ssb input.

Note that although the input occurs at one frequency, the AM output can be measured directly from the single input.

When AM is applied at the input, the following diagram is shown.

The difference is that in addition to the input at low positive frequency, another input (to form an AM signal) is applied at low negative input. This second input adds to the output signal near the first harmonic.

For PM, the negative input amplitude is negative, as shown below.

By running all three inputs, the SSB/AM/PM input to SSB/AM/PM output can be calculated. This is a setup choice in the *Choosing Analyse*s form.

In the *Choosing Analyses* form, the input frequency range, and the output harmonic must also be specified.

All the plot functions are implemented in the *Direct Plot Form*.

Rapid IP2/IP3 (AC and PAC)

Note that Rapid IP2 and Rapid IP3 in hbac can only be selected when a single tone hb analysis is run. This would be the LO signal for a mixer. hbAC and AC add the ability to have multiple input frequencies. The Rapid IP2 and Rapid IP3 that is available in the AC analysis form allows the calculation of IP2 and IP3 for amplifiers, or other circuits where there is no frequency translation. In hbac, Rapid IP3 and Rapid IP2 are provided for circuits that translate frequencies, whether that translation is an up or down conversion. Note that hbac and ac are small-signal analyses, so the limitation is that the measurement is a small-signal IP3. If you need a large-signal IP2 or IP3 measurement, use a large-signal analysis like hb or qpss to make the measurement. Note that when you are using Rapid IP2 or Rapid IP3 in the AC analysis, either a SpectreRF license is required, or an additional mmsim token is required to run the analysis.

hbAC and AC Rapid IP3 also add the ability to calculate the harmonics and mixing products that are generated by the two input frequencies. Make sure that the input level is in the small-signal range. Rapid IP2 and IP3 have exceptionally small numerical noise floors, therefore, it is recommended that you specify a level that is too small, rather than too large.

The plot functions for Rapid IP2 and IP3 are implemented in the *Direct Plot Form*. Select *Results - Direct Plot - Main Form*. In the *Direct Plot Form*, select *ac* or *hbac* from the *Analysis* section. Next, select *Rapid IP2* or *Rapid IP3* (depending on which was set up) from the *Function* section, and click *Plot*. The X and Y points are shown for the intercept. The X value is input-referred IP2/IP3, and the Y is output-referred IP2/IP3.

Compression and IP2 distortion summary

The distortion summaries measure distortion in an amplifier in AC or in circuits that have frequency translation in hbac. The idea is to find which devices in a signal path contribute relatively more or less to the distortion. Since IP3 and compression are related mathematically, the compression distortion summary provides information about which devices contribute the most amount of third order intermodulation distortion. The IP2 distortion summary measures which devices contribute to the second order distortion. Note that when you are using the distortion summaries in the AC analysis, either a SpectreRF license is required, or an additional mmsim token is required to run the analysis.

Both distortion summaries use the *PAC Magnitude* property in the input port. Make sure that the PAC Magnitude (in volts or dBm) is actually in the small-signal range, or the results will not be accurate. This is because the hbac results do not take into account the large-signal limits like slew-rate or power supply voltage.

The distortion summaries provide a list output, not a plot output. Accordingly, instead of selecting *Results - Direct Plot - Main Form*, select *Results - Print - hbAC Distortion Summary*.

For the compression distortion, you get the actual gain with the distortion of each device by itself divided by the ideal small-signal gain provided by the hbac analysis. In addition, the amplitude of the first three harmonics of the linear output frequency is calculated.

For the IM2 distortion summary, you get the total IM2 output amplitude, and the amplitude that results from having only a single device in the circuit.

Sampled

Sampled is provided in order to calculate the instantaneous conversion gain at a voltage threshold crossing in the ifft or at specific timepoints in the ifft. This is useful for measuring

the conversion gain from the power supply to the oscillator output at a threshold crossing, or at specific times in the LO waveform to the output of a mixer.

ADE Implementation

Circuit, Input port Setting, and ADE Setup for all the Examples in This Section

Consider the following circuit:

This above circuit is an LNA and a mixer for a 2.4GHz application that is implemented in CMOS. The input source (a port in Cadence terminology) has the values for its frequency and amplitude set to variable names. The PAC magnitude property is set to 1 volt in order to allow easy conversion gain measurements. The reason for the variables is that the frequency and amplitude can be changed in the ADE environment without changing the schematic.

The *Edit Object Properties* form for the input port is shown below.

The ADE has the variables set to appropriate values, as shown below.

Note that the variables *fb1* and *fb2* (which set the input frequency of the input port) are set to zero, which effectively disables the RF input. Only the LO signal is applied to the circuit.

Using hbAC Analysis for Conversion Gain Measurement

Set up the hb analysis with just the large signal that causes the frequency translation to occur. For more information, see the harmonic balance section.

Set up the hbac Choosing Analyses form as follows:

1. Specify the input frequency range.
2. For hbac, or pac where pss is set to the Harmonic Balance engine, the *Maximum Sideband* field should be left blank in which case, all the mixing products are calculated for all the harmonics in the hb analysis.
3. For a normal conversion gain measurement, set the *Specialized Analysis* field to *None*.

For a conversion gain measurement, set the PAC magnitude to 1 volt on the *Edit Object Properties* form for the RF input port as shown earlier. This is always volts peak.

4. Now run the simulation. When the simulation completes, select *Results - Direct Plot - Main Form*.

   The *Direct Plot Form* is displayed.

The harmonic balance analysis characterizes the nonlinearity of the system as a series of harmonics. Hbac injects an input frequency, and because the system nonlinearity has been characterized by the harmonic balance simulation, the input tone mixes with the harmonics of the LO to produce outputs at multple frequencies. These mixing products are calculated for mixing the hbAC input frequency range with the hb harmonics. Because the amplitude of the RF input signal is assumed to be small, the harmonics of the hbac input and harmonics of the output mixing products are not calculated.

To measure the outputs you desire, place a marker at the desired output frequency of interest and read the conversion gain, as shown below.

The low frequency conversion gain is just over 19dB.

In the *Direct Plot Form*:

1. To plot just the desired output mixing product, select *sideband* instead of *spectrum*, and select the desired output mixing product form the list, as shown below.

2. Now select the output node, or click the *Replot* button in the *Direct Plot Form*. The waveform tool displays the output, as shown below.

3. Position a marker by moving the cursor to the desired output frequency and type *m*. The marker appears, as shown in the figure above.

4. To position the marker at an exact frequency, place it as above, and then move your cursor over the intercept point on the trace. Click and hold the right mouse button, move

to *Marker Properties*, and release the mouse button. The *Point Marker Properties* window appears, as shown below.

5. Type the frequency in the *Position* field, and click *OK* or *Apply*.

Commonly Used hbAC Options

Freqaxis

freqaxis specifies whether you want to see the negative frequency axis or not.

The analysis calculates the output frequencies and amplitudes, so *out* and *absout* are reasonable choices. *out* displays the negative frequency axis. *absout* (absolute value of the output) displays positive output frequencies. This is the default. Below is the default of *absout*.

Next is the output with freqaxis set to *out*. This is the same data as the above plot except the negative output frequencies have not been reflected up to the positive frequencies.

If you select *in*, all the outputs at the different frequencies are plotted on the same input frequency range scale. This is not recommended. The default is *absout*.

Tolerance

Leave this option at the default value.

Hbac uses an iterative solver to calculate the output amplitudes. Any iterative solver needs an error tolerance to specify when to stop iterating because the solution is accurate enough, and the tolerance option specifies that accuracy for hbac. For driven circuits, the default is 1e-6. For oscillators, the default is 1e-4.

Lnsolver

Leave this option at the default. Each hbac sideband is internally calculated from a matrix. This matrix is solved using an iterative solver. Several different algorithms are provided for the iterative solver. *gmres* is the default because the accuracy of each iteration inherently increases at the next iteration. Convergence is generally good as well. The other solvers may require less memory, but they are less robust for convergence and may suffer from false convergence. Their use is not suggested.

Considerable knowledge is needed to understand the differences between these methods. For more information, refer to the books on linear algebra theory.

gmres is the Generalized Minimum RESidual method.

qmr is the Quasi-Minimal Residual method.

bicgstab is the STABilized BI-Conjugate Gradient method.

resgmres is the REStarted Generalized Minimal RESidual method.

Resgmrescycle

Leave this option at the default. This option is used when resgmres is set for the linear solver. For the resgmres linear solver, there are six different options.

Hbprecond_solver

This option has two settings; basicsolver and autoset. Basicsolver is the only solver available in standard Spectre. Autoset is the default solver when APS is used. This solver is faster, but occasionally stagnates. When stagnation is detected, APS automatically switches to basicsolver, and prints a message in the Spectre output window.

Annotate

This option controls the amount of output in the spectre output window. Selecting title produces the least amount of detail, and selecting steps gives the most amount of detail. the default is status.

Save

The default is *allpub*. This saves all the public voltages at all levels of the hierarchy in the schematic. It excludes the internal nodes of the device models. *all* adds the internal nodes of

all the devices. *lvl* saves all the nodes including the internal nodes of the devices through the level of hierarchy set in the *nestlvl* option. *lvlpub* is like *lvl*, but it does not save the internal nodes. *selected* saves only the nodes that are specifically saved. In ADE, this is accomplished using the *Outputs - To Be Saved - Select On Schematic* and then selecting the nodes and terminals specifically in the schematic. At the netlist level, this is accomplished by using a save statement with a list of names to be saved.

Nestlvl

If *save* is set to *lvl* or *lvlpub*, this option controls the maximum level of hierarchy to be saved. If *nestlvl* is *1*, only the top level is saved. If *nestlvl* is 2, the top level and the next level down are saved. The value for *nestlvl* can be any integer. The example below shows *lvlpub* selected that saves two levels of hierarchy.

AdditionalParams

additionalParams is typically used for new features that are being beta tested early in the release cycle where there is not a gui selection available yet.

hbAC With Multiple hb Inputs

Harmonic balance is available in the hb form and the pss and qpss forms when harmonic balance is selected for the engine. Hbac is available in the hbac form and in the pac and qpac forms when the engine is set to harmonic balance in pss and qpss. In the example below, hb and hbac were used. The LO is set to 2.4GHz as before. To set up the second (RF) input for the harmonic balance analysis, the variable `fb1` was set to `2.5G` in the ADE window, as

shown below. This variable sets the frequency of the first source in the RF port component. In this case, the LO is at 2.4GHz and the RF is at 2.5GHz.

The harmonic balance analysis was set up to calculate the LO harmonics automatically, and three RF harmonics. For more information on Harmonic balance, please see the Harmonic Balance section at the beginning of this chapter.

The hbac form is set up exactly the same as the hbac form was for the first example in the ADE implementation section.

Now the analysis was run. The harmonic balance output plot below shows that because there were more input frequencies, many more harmonics are produced by the circuit, which is as expected.

Because there are many more harmonics to mix with, many more mixing products are produced by hbac, as shown below.

The mixing products below -200dB are at the numerical noise floor of the simulation.

The next figure shows a zoomed in area near zero frequency. Only the mixing products larger than -75 dB are shown.

If the output frequency were 50MHz, note that there are three different values. Which one is correct? Actually, they all are. They are produced by mixing with different harmonics from the large-signal analysis.

To identify which harmonics are being mixed with, select the top trace. On the left side of the display an entry is highlighted. The example below shows the top trace selected. (It is bold. It is difficult to see in the figure below, but in the waveform tool it is much easier.)

On the right side of the legend, it reads (-1 0) The first number is the harmonic number of the tone with tstab set, or is listed as *Tone1* in the hb *Choosing Analyses* form. This is LO in this example. The second number is the RF input.

-1 0 means that the input frequency is mixed down with the first harmonic of the LO, and not mixed with the RF. The input frequency range in the hbac (qpac) *Choosing Analyses* form is 2.40001G to 2.8G. If you take 2.40001G as a second small-signal input (that is what hbac does) and mix it down with the first harmonic of the LO at 2.4G, you get very near zero. When you mix 2.8G down with 2.4G, you get 400M. So this component starts at near zero, and sweeps up to 400M as the input sweeps from 2.40001G to 2.8G.

Now select the second trace down, as shown below.

The legend reads 1 -2.

2.40001G mixes up with the first harmonic of the LO, and then mixes down with the second harmonic of the RF. So, 2.40001G + 2.4G - 2*2.5G is very near -200M. As the input sweeps up in frequency, the output becomes less negative, passes through zero, and then sweeps up to +200M. Because the negative frequencies are reflected to the positive frequency domain, the resultant response looks like a less than sign (<) in the above plot.

Note that the ambiguity goes away when the frequency sweep on the input source is restricted to a smaller range.

Rapid IP3/IP2

Note: The first discussion is about Rapid IP3.

Rapid IP3 and Rapid IP2 are available in the AC and the hbac *Choosing Analyses* forms. Rapid IP3 in the AC form is for measuring IP3 of amplifiers where there is no frequency translation, and Rapid IP3 in the hbac form allows the measurement of IP3 where frequency translations are present. Rapid IP3 and IP2 measure small-signal IP3 and IP2.

1. First set up the hb form with just the LO applied, as shown below. For more information, see the harmonic balance section.

2. Select the hbac analysis. At the bottom of the form, in the *Specialized Analysis* section, select *Rapid IP3* or *Rapid IP2*, depending on what you want to measure. The entries for the forms are identical, except for the frequency of the intermodulation product. For

switched FET mixers, make sure that the psp model is used. Incorrect results will be produced for BSIM3 and BSIM4 models due to a limitation in these models.

3. Define the input port, input frequencies, input power, the frequencies of the linear and second or third order products, and the output node. An example setup for Rapid IP3 is shown below. Make sure the input power is in the small-signal range for your circuit.

4. Leave the *Maximum Non-linear Harmonics* field blank.
5. Run the simulation. When the simulation is complete, select *Results - Direct Plot - Main Form*.

6. Select *hbac* results.
7. Select *Rapid IP3* or *RapidIP2*, whichever was run.
8. Click *Plot*.

The waveform is displayed. The marker shown at the intercept point shows the input and output-referred IP3 or IP2.

Compression Distortion Summary

Note that the *Compression Distortion Summary* is available in the AC and the hbac *Choosing Analyses* forms. *Compression Distortion Summary* in the AC form is for measuring Compression Distortion of amplifiers where there is no frequency translation, and *Compression Distortion Summary* in the hbac form allows the measurement of Compression Distortion where frequency translations are present.

With the Compression Distortion summary, you can see which components in a frequency translating circuit cause relatively more and less distortion.

Note that compression and IP3 are related mathematically, so this analysis gives information about which components contribute more and less to the IP3 measurement as well as which components contribute to compression.

1. First set up the hb analysis with just the LO applied, as shown below. For more information, see the harmonic balance section.

2. Now select the hbac analysis.

3. Select *Compression Distortion Summary* from the *Specialized Analyses* drop-down list.

   The following notice appears to remind you to set the PAC amplitude parameter in the source in the small-signal range.

4. If you leave the *Contributor Instances* field blank, all the nonlinear devices in your circuit will be run. If you want a subset, click *Select*, and select the instances tht you want on the schematic.
5. Specify the input frequency in the *Frequency of Input Source* field.
6. Specify the output frequency in the *Frequency of Linear Output Signal* field.
7. Leave the *Maximum Non-Linear Harmonics* field blank.
8. Select the output node in the circuit.

The completed form should appear similar to the one shown below.

9. In the *Edit Object Properties* form, make sure that the *PAC magnitude (dBm)* is set in the small-signal range. In this example, it is set to `-30Bm`.

10. If you make changes to your schematic that you want for the distortion summary, but you do not want to save them to the circuit file, make the desired changes and then select *Check - Current Cellview* in the schematic window. Later, if you want to save them, just select *Check* and *Save*. If you do not want them, quit the schematic window, and do not save the changes.

ADE will netlist from the checked view, so it will get the changes.

11. When you are done, run the simulation.

12. When the simulation completes, select *Results - Print - hbAC Distortion Summary*.

The summary is displayed below:

Results Display Window

Window Expressions Info Help cadence

HBAC Compression Distortion Summary

Results in hbac_distortion

Instance	Distortion(dB)	Nonlinear Mag at 1st 2nd & 3rd harm of linear freq (V) freq=1e+06	freq=2e+06	freq=3e+06
Total	-381.694m	3.9925m	30.4208f	954.021u
I33.I1	-365.073m	3.79223m	25.993f	952.687u
I33.I1.I0	-230.353m	2.88156m	26.2349f	669.734u
I33.I1.I0.NM1	-49.3797m	867.699u	1.79563m	166.042u
I33.I1.I0.NM0	-49.3797m	867.699u	1.79563m	166.042u
I33.I1.I1.NM0	-54.8752m	773.258u	1.86273m	173.055u
I33.I1.I1.NM1	-54.8752m	773.258u	1.86273m	173.055u
I33.I0.NM11	-6.98423m	125.421u	32.7355u	1.88953u
I33.I0.NM9	-6.98423m	125.421u	32.7355u	1.88953u
I33.I1.I0.PM5	-10.3919m	116.7u	273.215u	30.5835u
I33.I1.I0.PM4	-10.3918m	116.7u	273.212u	30.5835u
I33.I1.I0.PM2	-10.3918m	116.7u	273.212u	30.5835u
I33.I1.I1.PM2	3.3283m	102.038u	259.683u	28.7359u
I33.I1.I1.PM4	3.3283m	102.038u	259.683u	28.7359u
I33.I1.I1.PM3	3.32827m	102.037u	259.68u	28.7357u
I33.I1.I1.PM5	3.32827m	102.037u	259.68u	28.7357u
I33.I1.I0.PM0	-3.50823m	91.4258u	468.653u	27.7093u
I33.I1.I0.PM1	-3.50823m	91.4258u	468.653u	27.7093u
I33.I1.I1.PM1	-6.88681m	83.6277u	473.987u	28.0417u
I33.I1.I1.PM0	-6.88681m	83.6277u	473.987u	28.0417u
I33.I0.NM10	-1.25223m	15.6818u	6.8781u	67.5971n
I33.I0.NM12	-1.25223m	15.6818u	6.8781u	67.5971n
I33.I1.I0.I2.NM33	-2.08113u	36.5858n	4.65397p	66.061n
I33.I1.I0.I2.NM3	-2.08113u	36.5858n	4.65404p	66.0609n
I33.I1.I1.I2.NM3	-2.71335u	35.6473n	4.63098p	66.0817n
I33.I1.I1.I2.NM33	-2.71335u	35.6473n	4.63105p	66.0817n
I33.I1.I1.I2.PM12	-60.5371n	1.73096n	45.4684f	4.01256n
I33.I1.I1.I2.PM11	-60.5367n	1.73095n	45.4681f	4.01255n
I33.I1.I0.I2.PM11	152.725n	1.73065n	52.246f	4.01162n
I33.I1.I0.I2.PM12	152.724n	1.73064n	52.2458f	4.0116n
I33.I1.I0.I2.NM1	595.725p	41.67p	445.221f	6.74277n
I33.I1.I0.I2.NM2	595.725p	41.67p	445.216f	6.74277n
I33.I1.I1.I2.NM2	2.81827n	29.9802p	701.941f	6.74427n
I33.I1.I1.I2.NM1	2.81827n	29.9802p	701.938f	6.74427n
I33.I1.I0.I2.NM0	-258.965p	2.8209p	14.9016f	155.755p
I33.I1.I0.I2.NM32	-258.965p	2.8209p	14.8961f	155.755p
I33.I1.I1.I2.NM32	36.7197p	1.85554p	27.5315f	155.711p
I33.I1.I1.I2.NM0	36.7197p	1.85554p	27.5324f	155.711p

- The first column is the instance name in the schematic.
- The second column is the gain with the signal applied divided by the ideal gain.

Gain compression is shown as a negative number. Gain expansion is shown as a positive number.

- The third, fourth, and fifth columns are the amplitudes of the first, second, and third harmonic of the linear output frequency with just the nonlinearity of that single device taken into account.

IM2 Distortion Summary

The steps for the IM2 distortion summary are very similar to the Compression distortion summary in the previous section. The differences are noted below.

- The IM2 distortion summary provides the output voltage (magnitude, real, and imaginary parts) of the 2nd order distortion signal for each component in the schematic by itself.
- The IM2 distortion summary does not provide a selection mechanism for the devices in the circuit. All the devices are run in the IM2 distortion summary.
- The output format is slightly different. The first column is the instance name in the schematic. The second column is the magnitude of the IM2 product that is produced by

that single component in volts. The third column has the real and imaginary parts of the voltage. A sample is shown below.

```
HBAC IM2 Distortion Summary

Results in        hbac_im2distortion

Instance          Distortion(V)[Magnitude Complex]
Total             401.43f    complex(-400.233f, -30.973f)
/I33/I1/I0/PM5    1.3848m    complex(-1.3848m, -2.83457u)
/I33/I1/I0/PM3    1.3848m    complex(1.3848m, 2.83457u)
/I33/I1/I0/PM4    1.38479m    complex(1.38479m, 2.83456u)
/I33/I1/I0/PM2    1.38479m    complex(-1.38479m, -2.83456u)
/I33/I1/I1/PM2    1.37327m    complex(-1.37327m, -2.82163u)
/I33/I1/I1/PM4    1.37327m    complex(1.37327m, 2.82163u)
/I33/I1/I1/PM3    1.37327m    complex(1.37327m, 2.82162u)
/I33/I1/I1/PM5    1.37327m    complex(-1.37327m, -2.82162u)
/I33/I1/I1/I2/NM33    154.322n    complex(113.004n, -105.097n)
/I33/I1/I1/I2/NM3    154.322n    complex(-113.003n, 105.097n)
/I33/I1/I0/I2/NM3    150.693n    complex(-111.65n, 101.206n)
/I33/I1/I0/I2/NM33    150.693n    complex(111.649n, -101.207n)
/I33/I1/I1/I2/PM12    9.1563n    complex(7.15786n, -5.70989n)
/I33/I1/I1/I2/PM11    9.15628n    complex(-7.15786n, 5.70986n)
/I33/I1/I0/I2/PM11    9.12122n    complex(-7.28648n, 5.48669n)
/I33/I1/I0/I2/PM12    9.12118n    complex(7.28648n, -5.48664n)
/I33/I1/I1/NM0    4.42456n    complex(-4.42455n, -9.36531p)
/I33/I1/I1/NM1    4.42456n    complex(4.42455n, 9.36527p)
/I33/I1/I0/NM0    4.22614n    complex(4.22614n, 8.5599p)
/I33/I1/I0/NM1    4.22614n    complex(-4.22613n, -8.55975p)
/I33/I1/I1/PM1    2.66145n    complex(-2.66145n, -5.45575p)
/I33/I1/I1/PM0    2.66143n    complex(2.66142n, 5.4525p)
/I33/I1/I0/PM1    1.24769n    complex(1.24769n, 2.75393p)
/I33/I1/I0/PM0    1.24766n    complex(-1.24765n, -2.74832p)
/I33/I0/NM9       358.96p    complex(-358.957p, -1.40184p)
/I33/I0/NM11      358.957p    complex(358.954p, 1.40182p)
/I33/I1/I1/I2/NM2    178.011p    complex(-160.898p, 76.1576p)
/I33/I1/I1/I2/NM1    178.011p    complex(160.898p, -76.1575p)
/I33/I1/I0/I2/NM2    101.771p    complex(67.2487p, 76.3869p)
/I33/I1/I0/I2/NM1    101.771p    complex(-67.2486p, -76.387p)
/I33/I1/I0/I2/NM32    83.8968p    complex(83.8604p, 2.47293p)
/I33/I1/I0/I2/NM0    83.8968p    complex(-83.8603p, -2.47289p)
/I33/I1/I1/I2/NM0    55.5677p    complex(-55.5163p, -2.3902p)
/I33/I1/I1/I2/NM32    55.5677p    complex(55.5163p, 2.39017p)
/I33/I0/NM12      22.2028p    complex(22.2027p, 53.3724f)
/I33/I0/NM10      22.195p    complex(-22.1949p, -53.3496f)
```

Modulated hbAC Analysis for an Oscillator

Modulated hbac analysis is usually applied to oscillators to measure AM to PM conversion from low-frequencies on the power supply (ripple) to the output near the first harmonic. It can also be applied to digital circuits for the same purpose. Although the shape of the transfer function is different, the steps are the same for both applications. Note that a relatively large number of harmonics and an oversample factor of at least 4 is needed for simulating the digital circuit.

In the circuit, make sure that any PAC magnitude terms are removed on the input source (if there is one), and set the PAC Magnitude to 1 volt in the source that is used as the power supply. If you do not want to save the change permanently, just select *Check - Current View* in the schematic.

This example is for an oscillator. The circuit is shown below:

In ADE, set the values for the power supply voltage and for the tuning voltage.

Set up the harmonic balance analysis For more information, see the Harmonic Balance section at the beginning of this chapter.

In the hbac *Choosing Analyses* form, first select *Modulated* in the *Specialized Analyses* section, as shown below.

Note that this causes the *Sweep Type* to be set to *relative*. Since the gain from power supply ripple to the output is desired, the relative harmonic number is zero, and then the desired frequency range is entered. *Maximum sideband* can be left blank, or it can match the hb harmonics, as shown above.

Next, the output harmonics can be set. Note that a list of harmonics can be entered. The easiest way is to click the *Choose* button to the right of the field and select the desired frequencies from a list in the *Choose Harmonic* window, as shown below.

The same process is used to set the input frequency, but only one frequency range is allowed.

Once the analysis has been set up, run the simulation. When the simulation completes, select *Results - Direct Plot - Main Form*.

The input defaults to AM. Select PM for the output. Since many output frequencies were calculated, you need to select the one you want to plot. In this example, the output just above the first harmonic is selected. Once this is done, select a net in the schematic.

The output is displayed. To make a measurement at a specific frequency, position a marker. The measurement is in radians per volt of input on the supply that has PAC magnitude set.

Sampled hbAC Analysis for a Mixer

hbAC sampled is for measuring the conversion gain at a specific instantaneous time in the ifft waveform. The first step is to determine an appropriate threshold for the measurement by setting up the hb analysis, as shown below.

Now run the analysis. When the analysis completes, select *Results - Direct Plot - Main Form* in ADE and plot the time-domain waveform, as shown below.

The waveform is shown below.

Note that 2.5 volts is a good threshold voltage.

Now set up the hbac *Choosing Analyses* form.

At the top of the form, set up the input frequency range like normal.

1. At the bottom of the form, select *Sampled*. Usually the gain to a node is desired, therefore, select *voltage*.

2. Click *Select* located to the right of the *Net+* field and select the node in the schematic.
3. Specify the desired threshold voltage based on the ifft from the hb analysis
4. Choose an option from the *Crossing Direction* drop-down list. In the example, *all* is selected.
5. If you need more than 16 crossing points, type the number you need in the *Maximum Samples* field.
6. If you want to sample at specific times from the ifft waveform, you can type in the times in the *Additional Timepoints* field.

7. Now run the simulation. When the simulation completes, open the *Direct Plot Form* and set it up, as shown below.

8. Select *hbac* results.

9. Select *Voltage*.

10. Select *spectrum*.

11. Select *dB20*.

12. Select the time you want from the *Variable Value (eventtime)* list.

13. Select the output node in the schematic.

The waveform is plotted, as shown below.

Because the sweep was log spaced, the X Axis of the waveform tool is set to log. To zoom in to the frequency of interest, double-click one of the numbers on the X Axis, and type in the output frequency range of interest. This is shown below.

To measure the conversion gain to the output, move the tracking cursor to the output frequency and read the gain.

Harmonic Balance Noise Analysis (hbnoise)

Small-Signal Versus Large-Signal Analysis for Noise

In most cases, noise is a small-signal problem in the real world. Consider an amplifier. One approach is to calculate the small-signal gain from the noise sources to the output, then calculate the frequency response of the noise source, and then add up the noise power at the output. This is done in the linear noise analysis. This runs quickly because the gain calculation just takes the tangent at the DC operating point for everything in the circuit. This ignores large-signal effects like slew-rate limits and clipping. Because noise in most circuits is actually very small compared to the output levels, this approach works well. Because it is a linear approach, frequency translations cannot be calculated.

When the input to the amplifier gets large enough that it starts to create nonlinearity, the frequency translations in the circuit need to be taken into account. Noise near the harmonics can mix up or down to the output frequency. In this case, the large signal needs to be applied in a large-signal analysis like harmonic balance (hb) where the nonlinearity and harmonic levels can be calculated. After the hb runs, hbnoise runs and is similar to linear noise except the frequency translations of the circuit are taken into account. Instead of calculating a single gain from the noise source to the output, both the amplitude of the noise and the gain for the noise sources at the significant harmonics needs to be calculated, and each noise source now has several contributions at the output from several different noise frequencies. The basic strategy is to apply the signals that cause nonlinearity in the hb analysis, and follow that with an hbnoise calculation.

This same approach can be taken for mixers where the LO signal deliberately introduces nonlinearity for the purpose of frequency translation. The LO signal is applied to the mixer in the hb analysis, and hbnoise follows that.

This works with the mixer by itself, or in combination with other circuits, like an LNA, mixer, and a baseband amplifier.

Many designs need to tolerate a large amplitude blocking signal. Apply the LO and the blocking signal (and not the RF signal) in the hb analysis and follow with hbnoise. The blocker power can be swept in the hb analysis and hbnoise run after each sweep value for the blocker power to measure the desensitization.

For large-signal noise problems, or for systems that are not periodic, transient noise is provided. For more information on this, refer to the *Virtuoso® Spectre® Circuit Simulator and Accelerated Parallel Simulator User Guide*. This approach requires specifying a maximum noise frequency to be analyzed, which in RF systems is usually quite high. Two timepoints are created in the period of noisefmax. The transient needs to be run over many noise points in order to characterize the noise present in the circuit and needs post-

processing to see the results in the frequency domain. This usually requires much more time than using small-signal noise analyses.

Overview of Simulation Capabilities

The fundamental quantity calculated in hbnoise is the output noise of the circuit with a large signal applied and including the noise translations caused by the nonlinearity of the circuit. In order to run hbnoise, a harmonic balance (hb) analysis needs to be run first in order to calculate the nonlinearity of the circuit and the amplitude of the harmonics that are present to mix with. Hbnoise takes this information from the hb analysis and then for each noise source it calculates the transfer functions from all the frequencies specified in the *Choosing Analyses* form. Then it calculates the frequency response of the noise source itself. Next, it calculates the total noise power at the output for that source at all the noise frequencies that contribute at the output. At the end, it adds up all the noise power from all the sources at all the frequencies. Input-referred calculations require the selection of the input frequency range by setting the reference sideband. This selection is necessary because there are many transfer functions from the input to the output caused by mixing or aliasing from the different harmonics of the system. The passband frequency is needed for the noise figure calculations.

Note that hbnoise is a small-signal calculation that works for periodic systems. If the noise is large-signal, or the system is not periodic, transient noise should be used.

Example

Consider the double-balanced diode mixer below where a single-sideband noise figure measurement is desired.

The input source has variables set to define the frequency and amplitude of the two large-signal outputs. This is done so that the frequency and the amplitude can be set in the ADE window without changing the schematic.

The property list for the input port is shown below.

The variables have values set in the ADE variables section. The easiest way to get the variables list is to generate a netlist from the ADE or use *Variables - Copy From CellView*. This will cause all the variable names to be entered in the variables section. Just click in the value field for each variable and enter a value. Press *Enter* when you are done. In the figure below, the LO frequency set by the variable flo is 1GHz. The variable `vrf` which sets the linear amplitude of the large-signal RF inputs is zero. This disables the large-signal waveform. This circuit just has the signal that causes the frequency translations to occur (the LO signal) applied. Note that 1.4 volts peak is +13 dBm, which is a common drive level for a passive diode mixer.

The hb analysis needs to be run first in order to measure the amplitude of the harmonics and to characterize the nonlinearity of the system. From this, the noise folding from all the noise sources can be calculated in the hbnoise analysis. For more information on the setup of the hb form, please refer to the hb large-signal analysis section.

In the form below, the 1GHz LO is solved using automatic harmonics.

The noise output frequency and the number of noise translations to be calculated are set up in the hbnoise form. In the example below, the output frequency is near baseband. Leaving

the *Maximum sideband* field blank causes noise translations from all the harmonics in the hb analysis to be calculated.

The reference sideband selects the passband frequency for the noise figure calculation.

Once both forms are set up, the simulation is run. To plot outputs, select *Results - Direct Plot - Main Form* in the ADE window. The top of the form lists all the analyses that were run.

Select *hbnoise* where the noise figure is calculated. Next, select *NF* because that is what is desired to plot. If the noise figure over a bandwidth is desired, select *Integrated Over Bandwidth*, and specify the frequency range you desire for the measurement.

When you are done, click *Plot*.

The waveform window plots the noise figure and the noise figure over the range you specified to be displayed, as shown below.

Noise Output Near Zero Frequency

First, the hb analysis is run to calculate the nonlinearity and the harmonics. Then hbnoise is run to calculate the noise. In all systems there is a maximum number of noise translations that need to be taken into account. Either the system harmonics become so small that the amount of noise translation becomes minimal or the devices themselves have finite noise bandwidths.

These noise calculations from mixing or aliasing with the different harmonics are called sidebands.

In the diagram above the X axis is frequency. The top represents the harmonics that are calculated by the hb analysis. The bottom represents the noise output frequency. The arrows show the various mixing paths to the output frequency. Hbnoise calculates the noise contribution at the output with all these mixing paths in place. Black is the noise that occurs without mixing. Purple is the noise that mixes with the first harmonic. Blue is the noise that mixes with the second harmonic. Red is the noise that mixes with the third harmonic.

In hbnoise, the output frequency (which is usually a range of frequencies) is specified along with the highest harmonic to calculate the mixing products.

Noise Output Near The First Harmonic

Here the noise output frequency is shifted up to just above the first harmonic frequency. Black is the noise that occurs without mixing. Purple is the noise that mixes with the first harmonic. Blue is the noise that mixes with the second harmonic. Note that the noise frequency just below the first harmonic mixes with the second harmonic to provide the output just above the first harmonic. Red is the noise that mixes with the third harmonic. Note that the second term that mixes with the third harmonic is just above the fourth harmonic of the hb analysis which is not shown. The noise term shown with a dotted line is mixing with the fourth harmonic of the system.

Frequency Sweep

Since the fundamental quantity that is calculated in hbnoise is the output noise, the frequency range defined in the *Choosing Analyses* form defines an output frequency range to calculate the noise for. For driven circuits, the default is to sweep an absolute frequency range. Oscillators will be covered later.

For an up-conversion system, sometimes a log frequency range is desired near one of the harmonics of the system. In that case, select relative sweep, and specify a harmonic number. In the output frequency range, type in the desired offset frequency from that harmonic. Note that this is only available for frequencies above the desired harmonic number because to get this below the harmonic requires taking the log of a negative number, which does not exist.

When automatic is selected for sweep type, 100 total points will be calculated. In most cases, fewer points are needed so it is usually better to select linear or log and then specify the sweep you desire. Typically 3 to 5 points per decade or about 20 total points are needed.

Maximum Sideband

There is a property in the *Choosing Analyses* form called *Maximum sideband* that is used to define how many mixing products should be calculated. If *Maximum sideband* is zero, the noise analysis will only contain the noise that appears without mixing or aliasing of any type. This is not equivalent to a linear noise analysis because hbnoise takes into account the instantaneously varying nature of the noise because of the large signal being present in the hb analysis.

When *Maximum sideband* is one, then the noise without frequency translation and the noise that mixes or aliases with the first harmonic is present in the noise analysis. When *Maximum sideband* is 10, the noise analysis includes noise mixing through the 10th harmonic of the harmonic balance analysis. Usually the number of harmonics in the hb analysis and the maximum number of sidebands should agree. This is easily accomplished by leaving the *Maximum sideband* field blank in the *Choosing Analyses* form in ADE.

Setting Harmonics and Sidebands

The process of setting harmonics and sidebands is similar to setting the maximum number of harmonics for the harmonic balance simulation. Start with an estimate based on the harmonic content of the input signal(s) and the nonlinearity of the circuit. Then run a simulation. Raise harmonics and sidebands and run again. If the noise measurement did not change, then the original number of harmonics and sidebands might be able to be reduced. If it did change, then more are needed. Getting a stable hbnoise output may take more or fewer harmonics and sidebands than the harmonic balance simulation by itself.

Noise Separation

In addition to the total output noise, the individual noise contributions can be plotted if noise separation is selected in the *Choosing Analyses* form. More information about the noise separation will be provided later. Noise separation is available when the noise type is set to sources only. The things that can be plotted are:

1. Total noise at the output from each individual input frequency.
2. The noise at the output from each instance name with all noise mechanisms included for mixing from an individual input frequency.

3. The noise at the output from each instance name with all noise mechanisms broken out separately for mixing from an individual input frequency.
4. The current noise at the noise instance for each instance name with all noise mechanisms included for mixing from an individual input frequency.
5. The noise current in the instance from each instance name with all noise mechanisms broken out separately for mixing from an individual input frequency.
6. The gain to the output from each individual instance with all the individual noise sources broken out separately for mixing from an individual input frequency.

Multiple hbnoise

Some systems contain multiple points in the circuit where the noise measurement is desired. When the *Noise Type* is *sources*, multiple hbnoise analyses are allowed. At the top of the hbnoise *Choosing Analyses* form, an option called *Multiple hbnoise* is provided. When this option is selected, up to six hbnoise analyses can be made after a single hb analysis.

Hbnoise after Multi-Tone hb

When multiple input frequencies are present in the circuit, there are more frequencies in the large-signal analysis to mix with. Not only are the harmonics of both tones present, the intermodulation products are present as well. In this case, the noise analysis becomes more involved because the transfer functions from all the frequencies present in the large-signal have to be calculated for each noise source. Also, the meaning of *Maximum sideband* slightly changes. If 5 were selected, all the noise translations would be calculated through the 5th harmonic of the tone with tstab enabled. In the harmonic balance *Choosing Analyses* form it is *Tone1* when *Frequencies* is selected, and the tone with tstab set to *yes* in the *Names* list. It is recommended that the largest number of harmonics for any tone in the hb list be used as maximum sideband. This can easily be accomplished by leaving the number of sidebands field being left blank.

Note that noise type=sources and noisetype=modulated are the only available choices for a 2-tone driven circuit.

Modulated for Driven Circuits and Oscillators (AM and PM Noise)

Modulated is provided so that the AM and PM components of noise can be separated out. When the *Noise Type* is *sources*, the total noise (AM + PM) is calculated. To illustrate AM noise, see the phasor diagram below.

This diagram illustrates a periodic signal with AM noise. The black phasor is the output signal, and AM noise is a random vector that varies the amplitude of the signal.

PM noise is shown in the phasor diagram below.

In this case, the amplitude is constant, but the noise is a random vector perpendicular to the signal phasor. The timing of the signal varies a bit from cycle to cycle.

Most circuits produce a combination of AM and PM noise. This concept is shown below.

In this case, a random vector at any angle and amplitude is produced that varies at each instant of time. If the uncertainty creates a round shape, the AM component and the PM component are the same size.

In many circuits, AM or PM noise may dominate. If the PM noise dominates, the circle turns into an ellipse that is large in the perpendicular direction than in the parallel direction. This is shown below.

Modulated also provides the total noise in the frequencies above the output frequency (the USB in the *Direct Plot Form*) and the noise below the output frequency (the LSB in the *Direct Plot Form*).

Jitter for Driven Circuits

Jitter is limited to single input frequency circuits. Jitter is usually applied to digital circuits where the average noise over the entire waveform is not appropriate. The noise in the high state and in the low state does not matter because the output voltage is so far from the voltage that triggers the load that it does not matter. The only time the noise matters is at the exact time where the trigger event is generated. Also, the noise in the low and high state is quite small because one device is completely off, and the other device is driven on. It is basically acting as a resistor. The device is not very good as an amplifier because the output is pulled hard into a rail. As the state changes, both devices are on, and they are relatively better amplifiers, so the noise is likely to be larger in the switching interval. In jitter for driven circuits, the voltage threshold for the noise measurement, and the selection of the rising edge, the falling edge, or both is made. At the threshold voltage specified, a noise measurement is

made at that instant. Once the noise voltage is known, the timing uncertainty can be determined from the slew rate. Since the RMS noise voltage is calculated, the timing uncertainty is one standard deviation from the ideal crossing time. This is illustrated below.

Timedomain for Driven Circuits

Timedomain is available for single input frequency circuits only. This is a variation of the jitter analysis explained in the previous section. In this analysis, instead of specifying a voltage, a list of times from the time-domain waveform is supplied. A noise analysis is performed at those instants in time, and the noise voltage is calculated at each timepoint. The conversion from noise voltage to time-domain jitter is a manual step.

Oscillators

Sources

For oscillators, the hbnoise frequency range defaults to *relative*. Specify the harmonic number as appropriate for the system you are simulating. If you are simulating an oscillator by itself, then the harmonic number is likely to be 1. If you have an oscillator and a diode frequency doubler, then the harmonic number is likely to be 2. If you have an oscillator with a frequency divider, in the hb form, you should specify the approximate frequency of oscillation for the oscillator itself, and set the freqdivide property to the divider ratio. This has the effect of dropping the first harmonic frequency by the divide ratio. In the hbnoise form, if the noise is desired on the frequency divided output, then the relative harmonic is 1. If the noise is desired at the output of the oscillator, the relative harmonic number is the divide ratio. The meaning of relative is to take the frequency of harmonic number specified and add to it the frequencies specified in the Choosing Analyses form. If the oscillator had a 1GHz output, and

the hbnoise had 1M relative to the first harmonic specified, the actual output frequency is 1G + 1M, or 1001M.

Oscillators do not need noise figure calculations and are not typically driven, so the input source should be set to *none*. The output is usually a voltage, and an output node is typically selected.

Hbnoise is a small-signal analysis and is not limited by large-signal effects such as clipping or slew-rate limits. As a result, at low offset frequency, the phase noise might be significantly greater than 0dBC/Hz. This indicates that the noise is larger than the oscillations, which is not physically possible. If you want to see the phase noise curve level off at low frequency, set the *lorentzian* option to *yes*.

Phase noise is the total output noise divided by the harmonic specified as the relative harmonic for the node specified for the noise measurement. If you want to see the AM and PM components of noise, or you want to measure jitter, or you want to make an instantaneous point in time noise measurement, set the *Noise Type* to *modulated* or *jitter*.

Modulated

Modulated is provided so that the AM and PM components of noise can be separated out. When the *Noise Type* is *sources*, the total noise (AM + PM) is calculated. For a diagram to understand the concepts, see Modulated for Driven Circuits and Oscillators (AM and PM Noise) on page 314

Timedomain

This is usually applied to oscillators that drive digital dividers where the average noise over the entire waveform is not appropriate. The noise above and below the threshold does not matter because the output voltage is so far from the voltage that triggers the load that it does not matter. The only time the noise matters is at the exact time where the trigger event is generated. In time-domain for oscillators, a list of times from the waveform is supplied and a noise measurement is made at that list of times. Once the noise voltage is known, the timing uncertainty can be determined from the slew rate. The noise calculated from this analysis provides a relative measure of the noise at different times. For a quantitative measurement, use PM jitter. For a diagram that illustrates this, see Jitter for Driven Circuits on page 317.

FM Jitter

FM jitter calculates a standard phase noise measurement, and also the AM and PM components from the modulated analysis, and adds the ability to integrate the phase noise

curve to calculate the cycle jitter or the cycle-to-cycle jitter. The jitter calculations are integrated into the *Direct Plot Form* in ADE. All the measurements are averaged over the oscillator cycle.

PM Jitter

This is like the time-domain jitter measurement. You specify a threshold voltage and the timing jitter is calculated. In PM Jitter, a quantitative jitter measurement can be made from the direct plot form. For a diagram on how this works, see Jitter for Driven Circuits on page 317.

Noise Calculations in the Simulator

Noise Factor and Noise Figure

When the hbnoise analysis is set up with voltage selected instead of probe for the input, noise figure and noise factor cannot be calculated because the system resistance specified in the port component is not available to the simulator. Using a port as the input probe is required to obtain noise figure and noise factor calculations.

In the *Direct Plot Form*, noise factor and noise figure are both single-sideband measurements.

Noise factor is:

```
((All the noise power at the output from all the components with all the frequency
translations) - (Noise power at the output from the load))/Noise power at the output
from the resistance of the input port from the passband frequency only.
```

In the *Choosing Analyses* form, the input port is identified by the instance name. The port component is a voltage source in series with a resistor and is considered as a single element. No access is provided to the results for the internal node where the resistor and voltage source join.

In order to determine the passband frequency, the reference sideband must be supplied. This specifies the passband frequency for the noise figure calculation. The easiest way to select the reference sideband is to use the *Select from list* feature in the hbnoise *Choosing Analyses* form.

The noise from the load is normally defined by selecting the port or resistor that is serving as the load for the circuit in the Choosing Analyses form.

Noise figure is:

```
10*log(10) (Noise Factor)
```

Noise factor (IEEE) is similar to the noise figure measurement above. If you subtract from the numerator all the noise from the input port at all frequencies except the passband, you have the IEEE noise factor.

As mentioned earlier, in order to calculate the IEEE noise factor, a port component must be used as the input. In the *Choosing Analyses* form, the input port is identified by the instance name. The port component is a voltage source in series with a resistor and is considered as a single element. No access is provided to the results for the internal node where the resistor and voltage source join.

In order to determine the passband frequency, the reference sideband must be supplied. This specifies the passband frequency for the noise figure calculation. The easiest way to select the reference sideband is to use the *Select from list* feature in the *Choosing Analyses* form.

The noise from the load is normally defined by selecting the port or resistor that is serving as the load for the circuit in the *Choosing Analyses* form.

Noise Factor IEEE is:

```
((All the noise power at the output from all the components with all the frequency
translations) - (Noise power at the output from the resistor in the input port at
all frequencies except for the passband) - (Noise power at the output from the
load))/Noise power at the output from the resistance of the input port from the
passband frequency only.
```

Noise figure IEEE is:

```
10*log(10) (IEEE Noise Factor)
```

Noise factor DSB is similar to the single sideband noise factor above, except in the denominator, the noise power from the input port at the image frequency is added.

Noise factor (DSB) is:

```
((All the noise power at the output from all the components with all the frequency
translations) - (Noise power at the output from the load))/Noise power at the output
from the resistance of the input port from the passband and image frequencies only.
```

As mentioned earlier, in order to calculate the DSB noise factor, a port component must be used as the input. In the *Choosing Analyses* form, the input port is identified by the instance name. The port component is a voltage source in series with a resistor and is considered as a single element. No access is provided to the results for the internal node where the resistor and voltage source join.

In order to determine the passband frequency, the reference sideband must be supplied. This specifies the passband frequency and by inference the image frequency for the noise figure

calculation. The easiest way to select the reference sideband is to use the *Select from list* feature in the *Choosing Analyses* form.

The noise from the load is normally defined by selecting the port or resistor that is serving as the load for the circuit in the *Choosing Analyses* form.

Noise figure (DSB) is:

```
10*log(10) (Noise Factor (DSB))
```

Summary:

`No` = total output noise
`Ns` = noise at the output due to the input probe at the passband frequency (the source)
`Nsi` = noise at the output due to the image harmonic at the source
`Nso` = noise at the output due to harmonics other than input at the source
`Nl` = noise at the output due to the output probe (the load)
`IRN` = input referred noise
`G` = gain of the circuit
`F` = noise factor
`NF` = noise figure
`Fdsb` = double sideband noise factor
`NFdsb` = double sideband noise figure
`Fieee` = IEEE single sideband noise factor
`NFieee` = IEEE single sideband noise figure

then,

```
IRN = sqrt(No^2/G^2)
F = (No^2 - Nl^2)/Ns^2
NF = 10*log10(F)
Fdsb = (No^2 - Nl^2)/(Ns^2+Nsi^2)
NFdsb = 10*log10(Fdsb)
Fieee = (No^2 - Nl^2 - Nso^2)/Ns^2
NFieee = 10*log10(Fieee)
```

When the hbnoise analysis is set up with voltage selected instead of probe, noise figure cannot be calculated.

Input-Referred Noise

When the hbnoise analysis is set up with the input source being a port, input-referred noise may not be exact. In this case, the input-referred noise is referred to the voltage source that is inside the port component. The amplitude of the noise is divided by two, which assumes a perfect match. If an accurate input-referred noise is desired, set the *Input Source* to *voltage*, and select the input node. This will disable the noise figure calculation.

Noise transfer function

Hbnoise calculates the forward gain from the passband to the output and saves it. In the *Direct Plot Form*, this is called *Transfer Function*, and is in linear volts at the output divided by volts at the input.

Commonly Used Options

Tolerance

Leave this option at the default value.

Hbnoise uses an iterative solver to calculate the output amplitudes. Any iterative solver needs an error tolerance to specify when to stop iterating because the solution is accurate enough, and the tolerance option specifies that accuracy for hbnoise. For driven circuits, the default is 1e-6. For oscillators, the default is 1e-4.

Lnsolver

Leave this option at the default. Each hbnoise sideband is internally calculated from a matrix. This matrix is solved using an iterative solver. Several different algorithms are provided for the iterative solver. *gmres* is the default because the accuracy of each iteration inherently increases at the next iteration. Convergence is generally good as well. The other solvers may require less memory, but they are less robust for convergence and may suffer from false convergence. Their use is not suggested.

Considerable knowledge is needed to understand the differences between these methods. For more information, refer to the books on linear algebra theory.

gmres is the Generalized Minimum RESidual method.

qmr is the Quasi-Minimal Residual method.

bicgstab is the STABilized BI-Conjugate Gradient method.

resgmres is the REStarted Generalized Minimal RESidual method.

Resgmrescycle

Leave this option at the default. This option is active when resgmres is set for the lnsolver. For the resgmres linear solver, there are six different options.

Hbprecond_solver

The basic solver is the only solver available in standard Spectre. Autoset is the default solver when APS is used. This solver is faster, but occasionally stagnates. When stagnation is detected, APS automatically switches to the basic solver, and prints a message in the Spectre output window.

Annotate

This option controls the amount of output in the spectre output window. Selecting *title* produces the least amount of detail, and selecting *steps* gives the most amount of detail. The options between produce more details as you move from left ot right.

Stimuli

This option is not implemented in the *Direct Plot Form* in ADE. It is used for the hbxf analysis that runs when an hbnoise is run. The default is *sources* which calculates the forward conversion gain in hbxf from the sources in the circuit only. *nodes_and_terminals* also calculates the conversion gain from all the nodes in the circuit to the output. To display these results in ADE, you must use the results browser and the calculator.

Save

The default is *allpub*. This saves all the public voltages at all levels of the hierarchy in the schematic. It excludes the internal nodes of the device models. *all* adds the internal nodes of all the devices. *lvl* saves all the nodes including the internal nodes of the devices through the level of hierarchy set in the *nestlvl* option. *lvlpub* is like *lvl*, but it does not save the internal nodes. *selected* saves only the nodes that are specifically saved. In ADE, this is accomplished using *Outputs - To Be Saved - Select On Schematic* and then selecting the nodes and terminals specifically in the schematic. At the netlist level, this is accomplished by using a save statement with a list of names to be saved.

Nestlvl

If *save* is set to *lvl* or *lvlpub*, this controls the maximum level of hierarchy to be saved. If *nestlvl* is *1*, only the top level is saved. If *nestlvl* is 2, the top level and the next level down are saved. The value for *nestlvl* can be any integer. The example below shows *lvlpub* selected that saves two levels of hierarchy.

Saveallsidebands

Leave this option at the default value of *no*. If you want to see the noise contributors from the different noise input frequencies, select *yes* for noise separation in the hbnoise *Choosing Analyses* form.

Lorentzian

This option is documented in the oscillator section. It only affects oscillators.

AdditionalParams

This is a field for `<option_name>=value` statements. Multiple `<option_name>=value` statements are allowed with a space between them. Generally, this is used to unlock the beta features.

Cyclo2txtfile Option

Select the *cyclo2txtfile* option if you want the simulator to create a file that has all the time-averaged noise information in it. When this option is set, hbnoise runs a noise analysis at all the sideband frequencies using the sweep information from the hbnoise form. The file is created in the `psf` directory and has the filename `hbnoise.pnoise_hbnoise.data`. This option is only available when a single input frequency is applied to the circuit in the hb analysis. Copy this file to another directory if you want to save it. The `psf` directory contents are deleted at the start of each simulation.

The `psf` directory is located either in the `simulation/<circuit_name>/spectre/schematic` directory or in the `<circuit_name>/spectre/config` directory. The simulation directory defaults to your home directory and can be set by your system administrator. Contact your system administrator to find the simulation directory if you are unfamiliar with the location

For more information regarding the other hbnoise options, type `spectre -h hbnoise` in a shell window.

The output file has two columns. The left column is the frequency, and the right column is the noise voltage squared. Below is an example of the file.

```
Terminal
File  Edit  View  Terminal  Tabs  Help
1.0000000002384186e+03     4.1976179202311482e-20
1.5848931927680969e+03     1.6713526184450628e-20
2.5118864316940308e+03     6.6541397418897690e-21
3.9810717058181763e+03     2.6490872073399725e-21
6.3095734450817108e+03     1.0545957949468125e-21
1.0000000000238419e+04     4.1981471199230932e-22
1.5848931924819946e+04     1.6711642040673237e-22
2.5118864315271378e+04     6.6548863771520372e-23
3.9810717055559158e+04     2.6594618704367568e-23
6.3095734448194504e+04     1.0893235490275314e-23
1.0000000000023842e+05     5.1584490781642718e-24
1.5848931924629211e+05     4.2122011260108464e-24
2.5118864315128326e+05     7.3817813521682905e-24
3.9810717055368423e+05     1.8376642299234070e-23
6.3095734448051453e+05     7.2002344421267964e-23
1.0000000000002384e+06     1.3844360510239424e-21
1.5848931924612522e+06     8.2447867935178601e-25
2.5118864315097332e+06     1.8906195812299195e-23
3.9810717055351734e+06     8.4013044161565058e-23
6.3095734448022842e+06     2.4460759370357001e-22
1.0000000000000238e+07     6.0508315175955095e-22
1.5848931924611330e+07     1.3563078334928723e-21
2.5118864315096140e+07     2.6606482609752016e-21
"hbnoise.pnoise_hbnoise.data" [readonly] 651L, 32550C
```

Interpolation in Noise Files

All independent sources have a property called Noise file name. This is an ascii file with two columns. The left column is frequency, and the right column is noise voltage squared. This

file is read for all the small-signal noise analyses. Between the points in the file, an interpolation is made.

Both curves in the above figure use the `cyclo2txt` output file for an oscillator at about 1.85GHz. Just the noise in the file is plotted. The top curve has a noise sweep to just less than one half the oscillation frequency. The bottom curve has a sweep to 10 MHz relative to the first harmonic. There were 10 harmonics in the harmonic balance simulation and 10 sidebands in the hbnoise analysis.

Because the noise analysis was referenced to the first harmonic, the +10 sideband starts at +10*1.85GHz + 1.85GHz, or 20.35GHz, and goes to about 20.36GHz for the 10MHz sweep, and 21.25GHz for the 900MHz sweep. Note that the data near 20GHz in both starts at about -120dBV and drops. The minus 10 sideband starts at -10*1.85GHz + 1.85GHz, or -16.65GHz to -16.64GHz for the 10MHz sweep and -15.75GHz for the 900MHz sweep. Note that there is no data on the bottom side of the 10th and 11th harmonic. The data in between the frequency points in the file is interpolated. That is the reason for the large rise for the last 2 noise spikes.

The spikes from the 900MHz sweep is narrower because the data goes to a higher frequency near 17GHz and 19GHz.

The figure below shows the curves overlaid on the same strip.

Note that the interpolation above and below the actual data in the 10MHz sweep (the blue trace) causes much more noise than the real system produces. Make sure that your noise sweep is sufficient for your requirements.

hbXF

Note that in the noise simulation, the forward transfer functions need to be calculated for all the noise sources in the circuit at all the noise frequencies that mix to the output frequency. When hb analysis is run and hbnoise is also run, the forward conversion gain from the sources in the circuit to the output specified in the hbnoise form is also calculated and presented in the *Direct Plot Form* as hbxf results.

Note that this is small-signal conversion gain. The large-signal effects like slew-rate and clipping limits are ignored for the hbxf calculation. If you need a large-signal measurement, you must use a large-signal analysis like hb, pss or qpss to do the calculation.

In the figure above, the top spectrum represents the harmonics created by the hb large-signal analysis. The negative and positive frequencies are shown and the DC component is in the center. The arrows show the transfer functions from the different input frequencies to the output frequency that is shown on the bottom axis. hbXF takes the maximum sideband property from the hbnoise setup form, and calculates the specified sidebands, which are really input to output conversion gain transfer functions.

The figure above illustrates a down-conversion case where the output is at low frequency.

The sideband numbers indicate which harmonic number is being mixed for the transfer function. Zero means that there is no frequency translation. One is the mixing product that occurs with a frequency shift of +1 times the hb harmonic frequency. Mixing frequencies to the left have negative numbers because the input frequency that causes the output are more negative than the output frequency.

For example, imagine a large-signal input at 1GHz in the hb analysis and an output frequency of 100MHz to be calculated. With no mixing, there is a gain from the sources in the circuit to the output of the circuit at 100MHz. Only the transfer function from 100MHz to 100MHz is included in this calculation. This is called the zero sideband because there is zero frequency shift from the input to the output.

The input frequencies that mix with the first harmonic at 1GHz to provide an output at 100MHz are at -900MHz and 1.1GHz. The frequency that is below the zero sideband is the -1 sideband and the frequency above the zero sideband is the +1 sideband.

Similarly, there are terms that mix with the second and third harmonics.

Usually, we think of positive frequencies only. This is shown above.

ADE Implementation

This section jumps from theory to practical with examples on how to use the individual analysis types that have been covered. Driven circuits and oscillators will be shown.

Driven Circuits

Noise Type = Sources

On the *Choosing Analyses* form:

1. First, set up the harmonic balance analysis. For more information, see the Harmonic Balance section at the beginning of this chapter.

2. In the hbnoise *Choosing Analyses* form, specify the output frequency range of your circuit.

Note: Since the fundamental quantity that is calculated by the noise analysis is the noise at the output, the frequency range is always the output frequency range.

3. Set about 3 to 5 points per decade in the *Points Per Decade* field.

4. Leave the *Maximum sideband* field blank or set it to the same number as the number of harmonics in the hb analysis. This specifies that the noise that mixes with all the hb harmonics is calculated.

5. Specify the output of the circuit. When the output is set to *probe*, a resistor or a port can be selected and this noise will be subtracted from the noise figure calculation.

6. Select *port* from the *Input Source* drop-down list for noise figure calculation.

7. Choose *Select from list* from the drop-down list for reference sideband, which specifies the input frequency range for the noise figure calculation.

8. Select the passband frequency range from the list.

9. Select *sources* from the *Noise Type* drop-down list to calculate the average noise power at the output averaged over the entire time of one cycle of the input waveform.

10. Select *yes* for *Noise Separation* to enable plotting of the individual noise contributors in the circuit. Noise separation is only available when *Noise Type* is set to *sources*.

Multiple hbnoise

When *Noise type* is set to *sources*, multiple hbnoise analyses can be run after a single hb analysis. This is useful for LNA-Mixer combinations, or in any circuit where there are multiple points in the circuit where noise is important.

To run multiple analyses after a single hb:

1. First, set up the harmonic balance simulation. for moire information, see the harmonic balance section at the beginning of this chapter.

2. Select the *Multiple hbnoise* check box located at the top of the hbnoise *Choosing Analyses* form.

3. Set up all the noise analyses you want to run, as shown below.

Noise Type = Modulated

Modulated allows the measurement of the AM and PM components of noise separately.

First, set up the hb analysis as appropriate for your circuit. For more information, see the harmonic balance section of this chapter.

On the hbnoise *Choosing Analyses* form:

1. Set up an appropriate frequency range of interest.

2. Select *modulated* from the *Noise Type* drop-down list.

3. Run the simulation. Note that modulated runs both the positive and negative frequencies.

```
/lan/csv/rfrd_v2/user/rdavis/rdavis/demos/ic61/docs_database/simul
File  Help                                                    cadence
*****************************************************************
HB Noise Analysis `hbnoise_mod1': freq = (125 kHz -> 1.25 GHz)
*****************************************************************
Input frequency range = (125 kHz -> 1.25 GHz).
    hbnoise_mod1: freq = 222.3 kHz   (6.25 %), step = 97.28 kHz    (6.
    hbnoise_mod1: freq = 395.3 kHz   (12.5 %), step = 173 kHz      (6.
    hbnoise_mod1: freq = 702.9 kHz   (18.8 %), step = 307.6 kHz    (6.
    hbnoise_mod1: freq = 1.25 MHz      (25 %), step = 547.1 kHz    (6.
    hbnoise_mod1: freq = 2.223 MHz   (31.2 %), step = 972.8 kHz    (6.
    hbnoise_mod1: freq = 3.953 MHz   (37.5 %), step = 1.73 MHz     (6.
    hbnoise_mod1: freq = 7.029 MHz   (43.8 %), step = 3.076 MHz    (6.
    hbnoise_mod1: freq = 12.5 MHz      (50 %), step = 5.471 MHz    (6.
    hbnoise_mod1: freq = 22.23 MHz   (56.2 %), step = 9.728 MHz    (6.
    hbnoise_mod1: freq = 39.53 MHz   (62.5 %), step = 17.3 MHz     (6.
    hbnoise_mod1: freq = 70.29 MHz   (68.8 %), step = 30.76 MHz    (6.
    hbnoise_mod1: freq = 125 MHz       (75 %), step = 54.71 MHz    (6.
    hbnoise_mod1: freq = 222.3 MHz   (81.2 %), step = 97.28 MHz    (6.
    hbnoise_mod1: freq = 395.3 MHz   (87.5 %), step = 173 MHz      (6.
    hbnoise_mod1: freq = 702.9 MHz   (93.8 %), step = 307.6 MHz    (6.
    hbnoise_mod1: freq = 1.25 GHz     (100 %), step = 547.1 MHz    (6.
Total time required for hbnoise analysis `hbnoise_mod1': CPU = 2.82957
Time accumulated: CPU = 9.32858 s, elapsed = 8.20055 s.
Peak resident memory used = 47.3 Mbytes.


*****************************************************************
HB Noise Analysis `hbnoise_mod2': freq = (-125 kHz -> -1.25 GHz)
*****************************************************************
Input frequency range = (125 kHz -> 1.25 GHz).
    hbnoise_mod2: freq = -222.3 kHz  (6.25 %), step = -97.28 kHz   (6.
    hbnoise_mod2: freq = -395.3 kHz  (12.5 %), step = -173 kHz     (6.
    hbnoise_mod2: freq = -702.9 kHz  (18.8 %), step = -307.6 kHz   (6.
    hbnoise_mod2: freq = -1.25 MHz     (25 %), step = -547.1 kHz   (6.
    hbnoise_mod2: freq = -2.223 MHz  (31.2 %), step = -972.8 kHz   (6.
    hbnoise_mod2: freq = -3.953 MHz  (37.5 %), step = -1.73 MHz    (6.
    hbnoise_mod2: freq = -7.029 MHz  (43.8 %), step = -3.076 MHz   (6.
    hbnoise_mod2: freq = -12.5 MHz     (50 %), step = -5.471 MHz   (6.
    hbnoise_mod2: freq = -22.23 MHz  (56.2 %), step = -9.728 MHz   (6.
    hbnoise_mod2: freq = -39.53 MHz  (62.5 %), step = -17.3 MHz    (6.
    hbnoise_mod2: freq = -70.29 MHz  (68.8 %), step = -30.76 MHz   (6.
    hbnoise_mod2: freq = -125 MHz      (75 %), step = -54.71 MHz   (6.
    hbnoise_mod2: freq = -222.3 MHz  (81.2 %), step = -97.28 MHz   (6.
    hbnoise_mod2: freq = -395.3 MHz  (87.5 %), step = -173 MHz     (6.
    hbnoise_mod2: freq = -702.9 MHz  (93.8 %), step = -307.6 MHz   (6.
    hbnoise_mod2: freq = -1.25 GHz    (100 %), step = -547.1 MHz   (6.
Total time required for hbnoise analysis `hbnoise_mod2': CPU = 13.148
Time accumulated: CPU = 22.4766 s, elapsed = 16.9117 s.
Peak resident memory used = 57.4 Mbytes.

modelParameter: writing model parameter values to rawfile.
element: writing instance parameter values to rawfile.
outputParameter: writing output parameter values to rawfile.
40  HelpAction
```

Now plot the Output noise using the *Direct Plot Form*.

On the *Direct Plot Form*:

1. Select *Output Noise* from the *Function* section.

2. Select *AM* or *PM* from the *Noise Type* radio buttons. The *AM* noise is shown below.

3. Click *Plot*.

On the plot, note that the AM noise is 3dB larger than the output noise. This is because the output noise is derived from the positive frequencies only, and the AM and PM noise is derived using the positive and negative frequencies.

Noise Type = Jitter

Driven jitter is usually used for measuring the noise in a digital circuit at a specific threshold crossing. Accordingly, the hb *Choose Analyses* form has oversample factor set to 4, as shown below. Note that *Number of Harmonics* is set to *auto*. *auto* sets the number of

harmonics that are needed automatically. You do not need to go through the trial and error process of setting harmonics manually.

When Spectre runs with *Decide automatically* and *auto harms*, the tstab interval can be remarkably short, as shown below. In addition, the number of harmonics is set high enough to have less than 1% total aliasing power in the result.

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/4_
File  Help                                                    cādence

==============================
`hb': time = (0 s -> 20.4 ns)
==============================
Important parameter values in tstab integration:
    start = 0 s
    outputstart = 0 s
    stop = 20.4 ns
    step = 20.4 ps
    maxstep = 16 ps
    ic = all
    useprevic = no
    skipdc = no
    reltol = 1e-03
    abstol(V) = 1 uV
    abstol(I) = 1 pA
    temp = 27 C
    tnom = 27 C
    tempeffects = all
    method = traponly
    lteratio = 3.5
    relref = sigglobal
    cmin = 0 F
    gmin = 1 pS
    rabsshort = 1 mOhm

    hb: time = 516.1 ps     (2.53 %), step = 7.08 ps     (34.7 m%)
Steady-state is detected after 1.2 ns.

Tstab: runs at least 200 timesteps per cycle,     MaxStep=2e-12

==============================
`hb': time = (1.2 ns -> 1.6 ns)
==============================
    hb: time = 1.21 ns     (2.57 %), step = 276.5 fs    (69.1 m%)
    hb: time = 1.231 ns    (7.82 %), step = 2 ps         (500 m%)
    hb: time = 1.251 ns    (12.8 %), step = 2 ps         (500 m%)
    hb: time = 1.271 ns    (17.8 %), step = 2 ps         (500 m%)
    hb: time = 1.291 ns    (22.6 %), step = 2 ps         (500 m%)
    hb: time = 1.311 ns    (27.6 %), step = 2 ps         (500 m%)
    hb: time = 1.331 ns    (32.6 %), step = 2 ps         (500 m%)
    hb: time = 1.351 ns    (37.6 %), step = 2 ps         (500 m%)
    hb: time = 1.371 ns    (42.6 %), step = 2 ps         (500 m%)
    hb: time = 1.391 ns    (47.6 %), step = 2 ps         (500 m%)
    hb: time = 1.41 ns     (52.6 %), step = 358.9 fs    (89.7 m%)
    hb: time = 1.43 ns     (57.6 %), step = 2 ps         (500 m%)
    hb: time = 1.45 ns     (62.6 %), step = 2 ps         (500 m%)
    hb: time = 1.47 ns     (67.6 %), step = 1.878 ps     (469 m%)
    hb: time = 1.49 ns     (72.5 %), step = 2 ps         (500 m%)
    hb: time = 1.51 ns     (77.5 %), step = 2 ps         (500 m%)
    hb: time = 1.53 ns     (82.5 %), step = 2 ps         (500 m%)
    hb: time = 1.55 ns     (87.5 %), step = 2 ps         (500 m%)
    hb: time = 1.57 ns     (92.5 %), step = 2 ps         (500 m%)
    hb: time = 1.59 ns     (97.5 %), step = 2 ps         (500 m%)

MultiThread info: 1 new work threads created

Notice from spectre during periodic steady state analysis `hb'.
    Auto harmonic calculation has chosen 33 harmonics.
    Solution size = 612, itres = 0.001, max GMRES iteration = 80
11
```

An appropriate threshold is needed, which can be determined by plotting the time domain waveform from the hb *Direct Plot Form*. The waveform is an ifft of the harmonics that were solved in harmonic balance. The waveform below shows an appropriate threshold value to be 1.65V.

On the hbnoise *Choosing Analyses* form, the stop frequency is half the input frequency. The start frequency is four decades below that. About five points per decade is usually sufficient. *Maximum sideband* should be left blank. The *Output* field is set to *voltage*, and the *Input Source* field is set to *none* because usually, only the output jitter is needed. The threshold is set appropriately. The *Crossing Direction* option can be set to *all*, *rise*, or *fall*.

Choosing Analyses – Virtuoso® Analog Design Environn
Analysis
tran
dc
ac
noise
xf
sens
dcmatch
stb
pz
sp
envlp
pss
pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
hbsp
Harmonic Balance Noise Analysis
Sweeptype
relative
Relative Harmonic
0
Output Frequency Sweep Range (Hz)
Start-Stop
Start
125K
Stop
1.25G
Sweep Type
Logarithmic
Points Per Decade
Number of Steps
5
Add Specific Points
Sidebands
Maximum sideband
When using hb engine, default value is harms of 1st tone.
Output
voltage
Positive Output Node
/Out1
Select
Negative Output Node
Select
Input Source
none
Do Noise
Noise Type
jitter
jitter: jitter measurement at the output
PM jitter for driven circuit
Signal
Threshold Value
Crossing Direction
/Out1
1.65
all
Enabled
Options...
OK
Cancel
Defaults
Apply
Help

When the simulation completes, use the *Direct Plot Form* to plot the jitter. For driven circuits, select *Jee*, which is edge-to-edge jitter. The event time is the time in the ifft for the large-signal hb simulation. In the below example, 77p is the rising edge, and 275p is the falling edge.

The output noise frequency response and the edge-to-edge jitter are displayed. The jee measurement is one standard deviation for the expected jitter.

Noise Type = Timedomain

On the hbnoise *Choosing Analyses* form:

1. First, set up the harmonic balance simulation. For more information on harmonic balance, see the Harmonic Balance section at the beginning of this chapter.

2. In the hbnoise *Choosing Analyses* form, *relative* is forced in the *Sweeptype* drop-down list.

3. Specify 0 in the *Relative Harmonic* field.
4. Set the stop frequency to half the input hb frequency.

5. Set the start frequency to four decades smaller than the stop frequency.
6. Leave the *Maximum sideband* field blank.

7. Select *voltage* from the *Output* drop-down list.

8. Specify the output node in the *Positive Output Node* field.
9. Select *none* from the *Input Source* drop-down list.

10. Select *timedomain* from the *Noise Type* drop-down list.

11. Select the *Number of Points* check box and specify 1 in the text field.

12. Select the *Add Specific Points* check box and specify the time values for the desired instantaneous noise measurements. Separate the entries with a space.

Oscillators

Noise Type = Sources

The settings are the same as driven `noise type = sources` with the exception of being a relative sweep by default. Since the phase noise of the oscillator is near one of the harmonics, *Sweeptype* is set to *relative* by default. A harmonic number is provided for the case where there is a frequency multiplier or a frequency divider on the output of the oscillator. Noise type = sources also has the ability to plot the noise contributors from mixing with each harmonic when noise separation is selected. There is an example of noise separation later in this chapter.

The hbnoise setup is similar to driven except that the relative harmonic is specified.

First, set up the harmonic balance analysis. For more information, see the Harmonic Balance section at the beginning of this chapter.

On the hbnoise *Choosing Analyses* form:

1. Select *relative* from the *Sweeptype* drop-down list.

2. Specify the appropriate harmonic number for your design. In this example, the first harmonic is specified in the *Relative Harmonic* field.
3. The frequency range should be the relative frequency range.

4. Leave the *Maximum sideband* field blank.

5. Select *voltage* from the *Output* drop-down list.

6. Specify the output node in the *Positive Output Node* field.
7. Select *none* from the *Input Source* drop-down list.

8. Select *sources* from the *Noise Type* drop-down list. This gives an averaged noise measurement.

9. Select *yes* for *Noise Separation* if you want to plot the individual noise contributors.

Noise Type = Modulated

This is the same as driven noise type = modulated (see Noise Type = Modulated on page 337). The AM and PM components of phase noise are available for plotting. Generally jitter is the preferred setting because it calculates the modulated result and it allows the calculation of jitter for the oscillator output. For examples, see Oscillators on page 447.

Noise Type = Timedomain

Remember that harmonic balance works in the frequency domain. To calculate a time-domain waveform, an ifft is required to convert the frequency-domain harmonics to a time-domain waveform. `Noise Type = Timedomain` provides a relative measurement of the noise at the individual points of the ifft. For a quantitative measurement, use the PM Jitter measurement.

First, set up the harmonic balance analysis. For more information, see the harmonic balance section at the beginning of this chapter.

On the hbnoise *Choosing Analyses* form:

1. Select *timedomain* from the *Noise Type* drop-down list.

2. Select the *Noise Skip Count* check box and type 0 in the text box.
3. Run the simulation.
4. In ADE, select *Results - Direct Plot - Main Form*.

5. Select hb results.
6. Select Time.
7. Select the output net(s) in the schematic. The output waveform is displayed.

8. Select *tdnoise* results.

9. Select *HB Integ Output Noise*.

10. Type the frequency range form the hbnoise *Choosing Analyses* form.

11. Click *Plot*.

12. Position markers by moving your mouse cursor near the point you want to measure, and type m in the time-domain output as desired. In the above example, markers were placed at the peaks and valleys of the noise curve. Markers were then placed on the waveform to see where the noise is relatively large and relatively small. The largest noise occurs just above the zero crossing of the waveform.

Noise Type = Jitter

Oscillator Phase Noise Measurement (FM Jitter)

This selection allows for measurement of the cycle jitter, the cycle-to-cycle jitter, and the AM and PM components of phase noise. This is an averaged noise measurement over one cycle of the oscillator.

First, set up the harmonic balance analysis. For more information, see the harmonic balance section at the beginning of this chapter.

On the hbnoise *Choosing Analyses* form:

1. Select *Jitter* from the *Noise Type* drop-down list.

2. Select the *FM* radio button.
3. Run the simulation.
4. In the ADE, select *Results - Direct Plot - Main Form*.

5. Select *hbnoise* results.
6. Select *Phase Noise*.

7. Click *Plot*. The single sideband phase noise is plotted. Note that the Lorentzian option is set to *yes*, which causes the phase noise to flatten out at low offset frequencies.

8. Select *hbnoise jitter* results.

9. Select *Phase Noise*.

10. Click *Plot*. Note that this is double-sideband phase noise, and it does not level off at low frequency.

11. Select *hbnoise modulated* results.

12. Select *PM* from the *Noise Type* section.

13. Click *Plot*. Note that for this oscillator, the PM component overlays the double-sideband phase noise, which indicates that the PM noise dominates.

14. To plot the cycle or cycle-to-cycle jitter, select *hbnoise jitter* results.

15. Select *Jc* or *Jcc*.

16. *Click* Plot. The cycle jitter result is plotted as a label in the waveform tool.

PM Jitter

This allows for instantaneous measurement of jitter at individual voltages in the ifft waveform.

First, set up the harmonic balance analysis. For more information, see the harmonic balance section at the beginning of this chapter.

On the hbnoise *Choosing Analyses* form:

1. Select *jitter* from the *Noise Type* drop-down list.

2. Specify the voltage for instantaneous noise measurement in the *Threshold Value* field. If you are not sure of the appropriate voltage, run the hb analysis, and plot the time-domain waveform for the output node(s).

3. Select a value from the *Crossing Direction* drop-down list.

Examples

Driven Circuit Noise Setup

First, set up the harmonic balance analysis. For more information, see the Harmonic Balance section at the beginning of this chapter.

On the hbnoise *Choosing Analyses* form:

1. Specify the output frequency range of your circuit.

2. If you use log spacing, set about 3 to 5 points per decade in the *Points Per Decade* field.

3. In hbnoise, and pnoise or qpnoise when Harmonic Balance is set for the pss or qpss engine, you can leave the *Maximum sideband* field blank. Then, all the mixing products from the large-signal analysis will be considered.

4. Select the resistor or port component used for the load resistance. This excludes the noise from this component for the noise figure calculation, but keeps it for the total noise output calculation.

5. If you want a noise figure calculation, you must use a port as the input for the circuit. Set the instance name.

6. The easiest way to select the reference sideband is to set it to choose *Select from list*.

7. Specify the passband frequency range that you want for the noise figure and input-referred noise calculations.

side	Frequencies		1.9G
u	1K	100M	0
l	1.8G	1.9G	-1
u	1.9G	2G	1

8. If you want to see the noise contributions from the individual frequencies and the different device noise calculations individually on the output of the noise analysis, enable noise separation.

9. When you are done with the setup, run the simulation.

Noise Summary

The noise summary provides information about the devices that contribute noise to the output. This information is available any time an hbnoise simulation completes.

In the ADE, select *Results - Print - Noise Summary*. If noise separation is enabled, there will be two choices at the very top. `Hbnoise_src` has information about the noise currents at the individual noise sources at the noise source. Hbnoise has information about noise at the output of the circuit.

If you want the noise contributors at a single frequency, select *spot noise*. If you want to have the noise integrated over a frequency range, select *integrated noise*.

If you want noise in volts, set the *noise unit* to *V.*

Generally, *Include All Types* should be selected. If you just want the noise from specific types of noise generators, you can select them from the list.

Truncate just applies to the list. The total input and output-referred noise at the bottom of the noise summary output always includes all the noise from everything in the circuit.

The figure below is the noise summary window with the sort being noise contributors. Then there are three windows that show the three different sorted outputs.

The output below shows the nosie summary sorted by noise contributors. The largest noise contributor is on the top of the list.

Results Display Window

Window Expressions Info Help cadence

Device	Param	Noise Contribution	% Of Total
/PORT1	rn	8.10808e-06	54.08
/I33/I0/NM12	id	2.731e-06	6.14
/I33/I0/NM10	id	2.731e-06	6.14
I33.I0.I28.I5.rs1	rn	1.73613e-06	2.48
I33.I0.I28.I11.rs1	rn	1.73613e-06	2.48
I33.I0.I28.I5.rs2	rn	1.70083e-06	2.38
I33.I0.I28.I11.rs2	rn	1.70083e-06	2.38
/I33/I0/R7/R3	rs	1.63603e-06	2.20
/I33/I0/R0/R3	rs	1.63603e-06	2.20
/I33/I1/I1/PM3	fn	1.30005e-06	1.39
/I33/I1/I1/PM5	fn	1.30005e-06	1.39
/I33/I1/I1/PM2	fn	1.30005e-06	1.39
/I33/I1/I1/PM4	fn	1.30005e-06	1.39
/I33/I1/I0/PM2	fn	1.19941e-06	1.18
/I33/I1/I0/PM4	fn	1.19941e-06	1.18
/I33/I1/I0/PM3	fn	1.19941e-06	1.18
/I33/I1/I0/PM5	fn	1.19941e-06	1.18
/PORT2	rn	1.15608e-06	1.10
I33.I0.I28.I5.rs11	rn	8.45348e-07	0.59
I33.I0.I28.I11.rs11	rn	8.45348e-07	0.59
I33.I0.I28.I5.rsbp1	rn	8.29624e-07	0.57
I33.I0.I28.I11.rsbp1	rn	8.29624e-07	0.57
I33.I0.I28.I5.rs22	rn	8.17196e-07	0.55
I33.I0.I28.I11.rs22	rn	8.17196e-07	0.55
I33.I0.I28.I5.rsbt1	rn	5.89468e-07	0.29
I33.I0.I28.I11.rsbt1	rn	5.89468e-07	0.29
/I33/I0/NM11	id	5.4465e-07	0.24
/I33/I0/NM9	id	5.4465e-07	0.24
/I33/I0/NM10	fn	5.3688e-07	0.24
/I33/I0/NM12	fn	5.3688e-07	0.24
/I33/I1/I0/NM1	id	4.09442e-07	0.14
/I33/I1/I0/NM0	id	4.09442e-07	0.14
/I33/I1/I0/PM1	id	4.0939e-07	0.14
/I33/I1/I0/PM0	id	4.0939e-07	0.14
/I33/I1/I1/NM0	id	3.81267e-07	0.12
/I33/I1/I1/NM1	id	3.81267e-07	0.12
/I33/I1/I1/PM0	id	3.80694e-07	0.12
/I33/I1/I1/PM1	id	3.80694e-07	0.12
I33.I0.I28.I9.rs2	rn	3.45878e-07	0.10
I33.I0.I28.I8.rs2	rn	3.45878e-07	0.10

Integrated Noise Summary (in V) Sorted By Noise Contributors
Total Summarized Noise = 1.10252e-05
Total Input Referred Noise = 1.21795e-06
The above noise summary info is for hbnoise data

46

The output below shows the noise summary sorted by composite noise. The largest total noise instance is on the top of the list, and the individual noise contributors for that component are shown below the instance name.

Results Display Window

Window Expressions Info Help cadence

Device	% Of Total	Inp Ref Noise Param		Noise Contribution
/PORT1	54.08	8.95665e-07	rn	8.10808e-06
			total	8.10808e-06
			ext_file_noise	0
/I33/I0/NM12	6.38	3.07632e-07	total	2.78486e-06
			id	2.731e-06
			fn	5.3688e-07
			rs	9.3991e-08
			rd	6.91542e-09
/I33/I0/NM10	6.38	3.07632e-07	total	2.78486e-06
			id	2.731e-06
			fn	5.3688e-07
			rs	9.3991e-08
			rd	6.91542e-09
I33.I0.I28.I5.rs1	2.48	1.91783e-07	rn	1.73613e-06
			total	1.73613e-06
			fn	0
I33.I0.I28.I11.rs1	2.48	1.91783e-07	rn	1.73613e-06
			total	1.73613e-06
			fn	0
I33.I0.I28.I5.rs2	2.38	1.87883e-07	rn	1.70083e-06
			total	1.70083e-06
			fn	0
I33.I0.I28.I11.rs2	2.38	1.87883e-07	rn	1.70083e-06
			total	1.70083e-06
			fn	0
/I33/I0/R7/R3	2.20	1.80726e-07	rs	1.63603e-06
			total	1.63603e-06
			id1	0
			id2	0
/I33/I0/R0/R3	2.20	1.80726e-07	rs	1.63603e-06
			total	1.63603e-06
			id1	0
			id2	0
/I33/I1/I1/PM3	1.41	1.38883e-07	total	1.30801e-06
			fn	1.30005e-06
			id	1.44033e-07
			rs	3.19789e-09
			rd	1.23572e-10
/I33/I1/I1/PM5	1.41	1.38883e-07	total	1.30801e-06
			fn	1.30005e-06
			id	1.44033e-07
			rs	3.19789e-09
			rd	1.23572e-10
/I33/I1/I1/PM2	1.41	1.38882e-07	total	1.30801e-06
			fn	1.30005e-06
			id	1.44033e-07
			rs	3.19789e-09
			rd	1.23572e-10
/I33/I1/I1/PM4	1.41	1.38882e-07	total	1.30801e-06
			fn	1.30005e-06
			id	1.44033e-07
			rs	3.19789e-09
			rd	1.23572e-10
/I33/I1/I0/PM2	1.20	1.28267e-07	total	1.20794e-06
			fn	1.19941e-06
			id	1.43282e-07
			rs	3.11628e-09
			rd	1.15133e-10
/I33/I1/I0/PM4	1.20	1.28267e-07	total	1.20794e-06
			fn	1.19941e-06
			id	1.43282e-07
			rs	3.11628e-09
			rd	1.15133e-10

Integrated Noise Summary (in V) Sorted By Device Composite Noise
Total Summarized Noise = 1.10252e-05
Total Input Referred Noise = 1.21795e-06
The above noise summary info is for hbnoise data

47 Clear

The output below shows the nosie summary sorted by device name. The devices are listed in alphabetical order.

Results Display Window

Window Expressions Info Help cadence

Device	% Of Total	Inp Ref Noise Param		Noise Contribution
/I33/I0/NM10	6.38	3.07632e-07	total	2.78486e-06
			fn	5.3688e-07
			id	2.731e-06
			rs	9.3991e-08
			rd	6.91542e-09
/I33/I0/NM12	6.38	3.07632e-07	total	2.78486e-06
			fn	5.3688e-07
			id	2.731e-06
			rs	9.3991e-08
			rd	6.91542e-09
/I33/I0/R0/R3	2.20	1.80726e-07	total	1.63603e-06
			id2	0
			id1	0
			rs	1.63603e-06
/I33/I0/R7/R3	2.20	1.80726e-07	total	1.63603e-06
			id2	0
			id1	0
			rs	1.63603e-06
/I33/I1/I0/PM2	1.20	1.28267e-07	total	1.20794e-06
			fn	1.19941e-06
			id	1.43282e-07
			rs	3.11628e-09
			rd	1.15133e-10
/I33/I1/I0/PM4	1.20	1.28267e-07	total	1.20794e-06
			fn	1.19941e-06
			id	1.43282e-07
			rs	3.11628e-09
			rd	1.15133e-10
/I33/I1/I1/PM2	1.41	1.38882e-07	total	1.30801e-06
			fn	1.30005e-06
			id	1.44033e-07
			rs	3.19789e-09
			rd	1.23572e-10
/I33/I1/I1/PM3	1.41	1.38883e-07	total	1.30801e-06
			fn	1.30005e-06
			id	1.44033e-07
			rs	3.19789e-09
			rd	1.23572e-10
/I33/I1/I1/PM4	1.41	1.38882e-07	total	1.30801e-06
			fn	1.30005e-06
			id	1.44033e-07
			rs	3.19789e-09
			rd	1.23572e-10
/I33/I1/I1/PM5	1.41	1.38883e-07	total	1.30801e-06
			fn	1.30005e-06
			id	1.44033e-07
			rs	3.19789e-09
			rd	1.23572e-10
/PORT1	54.08	8.95665e-07	total	8.10808e-06
			ext_file_noise	0
			rn	8.10808e-06
I33.I0.I28.I5.rs1	2.48	1.91783e-07	total	1.73613e-06
			fn	0
			rn	1.73613e-06
I33.I0.I28.I5.rs2	2.38	1.87883e-07	total	1.70083e-06
			fn	0
			rn	1.70083e-06
I33.I0.I28.I11.rs1	2.48	1.91783e-07	total	1.73613e-06
			fn	0
			rn	1.73613e-06
I33.I0.I28.I11.rs2	2.38	1.87883e-07	total	1.70083e-06
			fn	0
			rn	1.70083e-06

Integrated Noise Summary (in V) Sorted By Device Name
Total Summarized Noise = 1.10252e-05
Total Input Referred Noise = 1.21795e-06
The above noise summary info is for hbnoise data

47 Clear

For all of the noise summaries, here is a table of the abbreviations used in the *Param* column:

Resistor and Inductor:

Parameter	Meaning
`rn`	resistor thermal noise
`fn`	Flicker noise

Diode:

Parameter	Meaning
`id`	Diode current shot noise
`rs`	Series parasitic resistor noise

Gummel-Poon BJT:

Parameter	Meaning
`fn`	Flicker noise
`ic`	Collector current shot noise
`ib`	Base current shot noise
`rc`	Thermal noise from the collector parasitic resistor
`rb`	Thermal noise from the base parasitic resistor
`re`	Thermal noise from the emitter parasitic resistor

HiCUM:

Parameter	Meaning
`ib`	Base current shot noise
`ic`	Collector current shot noise
`rbi`	Thermal noise from the internal base parasitic resistor (See diagram below)
`rbx`	Thermal noise from the external base parasitic resistor (See diagram below)
`rcx`	Collector parasitic resistor thermal noise (See diagram below)
`re`	Emitter parasitic resistor thermal noise (See diagram below)
`fn`	Flicker noise

Figure 3-1 HiCUM Schematic Diagram

VBIC:

Parameter	Meaning
`rbi`	Thermal noise in resistor Rbi (See diagram below)
`rbx`	Thermal noise in resistor Rbx (See diagram below)
`rci`	Thermal noise in resistor Rci (See diagram below)
`re`	Thermal noise in resistor Re (See diagram below)
`rs`	Thermal noise in resistor Rs (See diagram below)
`itzf`	Collector current shot noise
`ibe`	Total base current shot noise
`ibex`	Node Bx to E base current shot noise (See diagram below)
`fn`	Ibe flicker noise (See diagram below)
`fnx`	Ibex flicker noise (See diagram below)
`iccp`	Parasitic transport collector to emitter current noise (See diagram below)
`ibep`	Parasitic transport base to emitter current noise (See diagram below)
`fnp`	Ibep flicker Noise (See diagram below)

Figure 3-2 VBIC Schematic Diagram

BSIM3:

Parameter	Meaning
`fn`	Flicker Noise

Parameter	Meaning
id	Thermal noise from the resistive drain current
rs	Thermal noise from the source parasitic resistor
rd	Thermal noise from the drain parasitic resistor

BSIM4:

Parameter	Meaning
fn	Flicker noise
id	Thermal noise from the resistive drain current noise
rs	Source parasitic resistor thermal noise
rd	Drain parasitic resistor thermal noise
igd	Gate-to-drain tunneling current noise
igs	Gate-to-source tunneling current noise
igb	Gate-to-bulk tunneling current noise
rbdb	Resistor thermal noise between dbNode and bNode (See diagram below)
rbpb	Resistor thermal noise between bNode and bNodePrime (See diagram below)
rbpd	Resistor thermal noise between Bulk and dbNode (See diagram below)
rbps	Resistor thermal noise between bulk and bNode prime
rbsb	Resistor thermal noise between sbNode and bNode (See diagram below)
rgbi	Gate Bias-Independent Resistor thermal noise

Figure 3-3 BSIM4 Schematic Diagrams

PSP-103

Parameter	Meaning
sfl	Channel current flicker noise
sth	Channel current thermal noise
sig	induced gate noise at 1 Hz
sigth	Induced gate noise
shotgd	Gate current shot noise from gate to drain
shotgs	Gate current shot noise from gate to source
sjnoise	Source to Bulk leakage current noise
djnoise	Drain to Bulk leakage and avalanche current noise
rgatenoise	Gate resistance thermal noise
rbulknoise	Bulk resistor thermal noise between node BP and BI (See diagram below)
rjnoise	Source side bulk resistor thermal noise between node BI and BS (See diagram below)
rjundnoise	Drain side bulk resistor thermal noise between node BI and BD (See diagram below)
rwellnoise	Well resistor thermal noise between node BI and B (See diagram below)
rsourcenoise	Source parasitic resistor thermal noise
rdrainnoise	Drain parasitic resistor thermal noise

Figure 3-4 PSP-103 Schematic Diagram

EKV 3.0

Parameter	Meaning
`fn_id`	Channel current flicker noise
`fn_ig`	Gate current flicker noise
`rs`	Source parasitic resistor thermal noise
`rd`	Drain parasitic resistor thermal noise
`rg`	Gate parasitic resistor thermal noise
`rsb`	Thermal noise for the substrate resistor connected between BSi and B. (See diagram below)
`rdb`	Thermal noise for the substrate resistor connected between BDi and B. (See diagram below)
`rbbs`	Thermal noise for the substrate resistor connected between BSi and Bi. (See diagram below where it is labeled R_{DSB}/2)

Parameter	Meaning
`rbbd`	Thermal noise for the substrate resistor connected between BDi and Bi. (See diagram below where it is labeled $R_{DSB}/2$)
`id`	Thermal noise of the channel resistor
`cor`	Induced gate noise
`shot_ig`	Gate current shot noise

Figure 3-5 EKV 3.0 Schematic Diagrams

For other model types:

1. In a UNIX/LINUX shell window type `which spectre`.

   This should return a path that ends in `tools/bin/spectre`.

2. Triple-click the line with the path, and then click the center mouse button to enter it as a command.

3. Use the backspace key, and delete the word `spectre`.

4. Type `cdnshelp` at the end of the expression, and press `Enter`.

   A browser window will be displayed.

5. Close the *Tip of the day* window.

6. At the top, you should see the release number of the simulator you are using.

7. Click the plus sign (+) to the left of the *MMSim* folder.

8. Click the plus sign (+) to the left of the *Virtuoso Simulator Components and Device Models Reference* folder.

9. Select the device type you are using.

10. At the bottom of that device listing, you will see *Component Statements*. Double-click on this.

11. Enter the name of the parameter in the search box in the right pane of the cdnshelp window.

12. Click the down arrow to the right of the *Find* field until you see the parameter name in the pane on the right side, and then see the definition of that parameter.

Plotting Noise Figure

On the *Direct Plot Form*:

1. Select hbNoise from the *Analysis* section.
2. Select the desired type of noise plot from the *Function* section.
3. If you want the noise figure over a specified bandwidth, check the *Integrated Over Bandwidth* checkbox.
4. Specify the frequency range you want for the noise figure calculation.

5. Click *Plot*.

   The dsb noise figure and the noise figure calculation over your bandwidth are displayed.

Viewing noise separation results

Viewing the total noise at the output from a single noise input frequency

This capability allows the noise from all sources in the circuit at different noise input frequencies to be measured. For example, all the noise that does not change frequency (the zero sideband) can be plotted, along with the other noise input frequencies (sidebands). This plotting capability allows problem frequencies to be identified.

On the *Direct Plot Form*:

1. Select *hbnoise separation* from the *Analysis* section.
2. To view the total noise at the output select *Sideband Output* from the *Function* section.
3. Select *V/sqrt(Hz)* or *v**2/Hz* from the *Signal Level* radio buttons.
4. Select the appropriate modifier from the *Modifier* section.
5. Select the sideband number form the *Output Sideband* list. 0 means no frequency translation.

6. Click *Plot*.

The output noise from sideband zero is plotted. This is the noise that is present at the output with no frequency translations included.

Plotting Noise from Sideband -1

On the *Direct Plot Form*:

1. Select the -1 sideband from the *Output Sideband* list. -1 means the noise frequency is shifted down in frequency by -1*first harmonic of hb from the frequency that is specified as the noise output frequency in the *Choosing Analyses* form.

2. Click *Plot*.

Virtuoso (R) Visualization & Analysis XL
File Edit View Graph Axis Trace Marker Measurements Tools Window Browser Help
cadence
Layout: Auto
Subwindows:
Workspace: Classic
WorkshopLib db_mixer schematic
Name
... noise; V / sqrt(
... noise; V / sqrt(
14.0
12.0
10.0
8.0
6.0
4.0
V/sqrt(Hz)(n)
10^3
10^4
10^5
10^6
10^7
10^8
freq (Hz)
mouse L:
M:
R:
3(5)
no error

As a suggestion, use the *Graph - Add Label* menu option to place labels in order to identify the traces.

Tip

If you select multiple sidebands to be plotted at the same time, the sideband number will be shown at the right side of the legend in the waveform window.

Viewing total noise at the output from each instance

Once the problem frequencies are identified in the plot of *Sideband Output*, the problem components need to be identified. From the previous plot, the zero sideband contributes the largest amount of noise to the overall output noise. Using the *Instance Output* option allows the identification of which components in the zero sideband contribute the most noise. The total noise for all the noise mechanisms within the component is plotted.

On the *Direct Plot Form*:

1. Select *Instance Output* from the *Function* section.

2. Select *V/sqrt(Hz)* or *v**2/Hz*.
3. Select the appropriate modifier.
4. Select the sideband number.
5. Select the type of component to display.

6. If you want every noise source to be considered, select *Include All Types*.
7. To exclude instances, click *Select* to the right of the *exclude inst* field, and select the device instances that you want to exclude from the plot.
8. To include specific instances, click *Select* to the right of the *include inst* field, and select the instances on the schematic.
9. Specify the number of results you want to plot.

10. Click *Plot*.

The total noise at the output for the top three BJTs is plotted.

To see which transistor goes with which trace, click the plus sign (highlighted in red on the previous graphic) to the left or the text at the upper left of the display area. The legends expand, and you can see which trace is which in the *Name* field on the left side

of the display tool. You can also either select the trace or the individual entry in the *Name* field and the trace and legend will highlight.

Viewing the exact noise mechanism within the devices

The sideband with the largest contribution was determined by selecting *Sideband Output* and plotting. The specific component was identified by selecting *Instance Output*. The specific noise mechanisms can be identified using *Source output*. This plots the largest individual noise mechanisms for the sideband you select.

On the *Direct Plot Form*:

1. Select *Source Output* from the *Function* section.

2. Select *V/sqrt(Hz)* or *v**2/Hz*.
3. Select the appropriate modifier.
4. Select the sideband number.
5. Select the type of component to display.

6. If you want every noise source to be considered, select *Include All Types*.
7. To exclude instances, click *Select* to the right of the *exclude inst* field, and select the device instances you want to exclude from the plot.
8. To include specific instances, click *Select* to the right of the *include inst* field, and select the instances on the schematic.
9. If you want to restrict to one device, put its instance name in the *include inst* field. A list of devices separated by a space is also acceptable.
10. If you want to exclude a device (or devices) enter the instance name(s) in the *exclude inst* field.
11. Specify the number of results you want to plot in the *by top* field.
12. Click *Plot.*

The exact noise mechanism within the devices is plotted.

To see which noise source goes with which trace, select the plus sign to the left or the text at the upper left of the display area. The legends will be shown individually in the *Name* field of the waveform tool. The device name and the individual noise contributor will be shown. You can select either the individual legend or the trace and both the legend

and the trace will highlight. For a list of the noise parameters for the different device types, see the list beginning just after the noise summary.

- The base current shot noise is in blue.
- The base parasitic resistance is red.
- The flicker noise is pink.

Viewing the total noise currents within the devices

This capability has limited value. All the noise currents from all the noise mechanisms within the instance are added up. No gain function is applied or are available for this measurement. It just adds up everything within that instance.

1. Select *Instance source*.
2. Select *A/sqrt(Hz)* or *A**2/Hz* in the Signal Level selection.
3. Select *Magnitude* or *dB20* in the *Modifier* section.
4. Select the sideband number.
5. If you want every noise source to be considered, select I*nclude All Types*.
6. To exclude instances, click *Select* to the right of the *exclude inst* field, and select the device instances you want to exclude from the plot.
7. To include specific instances, click *Select* to the right of the *include inst* field, and select the instances on the schematic.
8. Set the number of curves to plot by setting the Truncate function.
9. Click *Plot*.

10. The frequency response curves are displayed.

Plotting the individual noise source currents

This capability plots the individual noise currents within the device. it gives you the noise at the source instead of at the output of the simulation. This enables you to identify the largest individual noise sources in the circuit. Note that the largest noise source in the circuit may not contribute the largest noise at the output of the circuit. It depends on the relative gains from the individual sources to the output. These gain functions are avaialble, and are shown in the next section.

1. Select *Primary Source*.
2. Select *A/sqrt(Hz)* or *A**2/Hz* in the *Signal Level* section.
3. Select *Magnitude* or *dB20* in the Modifier section.
4. Select the sideband number.
5. If you want every noise source to be considered, select I*nclude All Types*.
6. To exclude instances, click *Select* to the right of the *exclude inst* field, and select the device instances you want to exclude from the plot.
7. To include specific instances, click *Select* to the right of the *include inst.* field, and select the instances on the schematic.
8. Set the number of curves to plot by setting the Truncate function.
9. Click *Plot*.

10. The frequency response curves are displayed.

For a list of the noise parameter names, please see the table under the section Noise Summary on page 372.

Plotting the gain from the individual noise sources to the output

This capability gives the gain from the individual noise currents to the output of the circuit. The noise current from the previous section times the gain that is plotted in this section gives the noise at the output of the circuit.

1. Select *Source Noise Gain*.
2. Select *A/sqrt(Hz)* or *A**2/Hz* in the *Signal Level* section.
3. Select *Magnitude* or *dB20* in the *Modifier* section.
4. Select the sideband number.
5. If you want every noise source to be considered, select *Include All Types*.
6. To exclude instances, click *Select* to the right of the *exclude inst* field, and select the device instances you want to exclude from the plot.
7. To include specific instances, click *Select* to the right of the *include inst.* field, and select the instances on the schematic.
8. Set the number of curves to plot by setting the Truncate function.
9. Click *Plot*.

The frequency response curves are displayed. For a list of the abbreviations used for the noise parameters, please see page 123 (77 of 234) of this manual.

Multiple hbnoise

Multiple hbnoise allows multiple pnoise analyses to be run after a harmonic balance simulation so that you can characterize the noise at several poinbts in your circuit. For example, in an LNA-Mixer circuit, you might want to characterize the noise at the LNA output and at the output of the mixer. Multiple hbnoise allows these analyses to be run without needing multiple simulations.

Consider an LNA-Mixer circuit shown below.

In the above circuit, there are two outputs that need to be measured. The first is at the LNA output, and the second is at the mixer output. The noise frequencies are different at the two outputs.

First, set up the harmonic balance analysis. For more information, see the Harmonic Balance section at the beginning of this chapter.

In the hbnoise *Choosing Analyses* form:

1. Select *Multiple hbnoise*, and set up the noise analysis for one of the outputs. The mixer output is shown below.

2. Select the second line in the multiple hbnoise section and set up that analysis.

3. Run the simulation. When the simulation completes, select *Results - Direct plot - Main Form*. When multiple hbnoise is run, the *Direct Plot Form* adds the ability to select which hbnoise results to plot. This is the purpose of the Multiple Out selection.

4. Click *Plot*. The single-sideband noise figure measurement for the LNA is shown below.

5. To plot the noise figure of the overall circuit, set the *Multiple Out* selection to the Mixer output at Port2. For direct conversion, usually double-sideband noise figure is desired.

6. Click *Plot*.

The waveform tool shows the noise figure curve and the integrated noise figure to 1MHz.

Noise as a Function of Blocker Power

This is a classic desensitization measurement that measures noise figure as the blocker amplitude gets large. In order to accomplish this, variables need to be added to the RF source

so that the input power of the blocker can be easily swept. The RF power in dBm is set by the variable prf in the following properties list.

- Note that the source has two large-signal tones set up.

- The first large-signal tone has the frequency set to the variable `frf1` and the amplitude set to `prf`.
- The second tone has the frequency set to the variable `frf2` and the amplitude set to `prf`. This is the blocker signal.

To measure the noise as a function of blocker power:

1. Apply the signals that cause the large-signal effects in the harmonic balance (hb) analysis, and then run an hbNoise analysis.

- The LO is set to 1.9GHz and 200mV peak. This is highlighted in red above.
- The first RF is set to 1.901G and -30 dBm. This is highlighted in blue above.
- The second RF tone is disabled by setting its frequency to zero.

  In this case, the frequency set by `frf1` is the blocker. Also in this case, the blocker frequency is set to the RF frequency so that both the gain compression plot from hb and the desensitization can be plotted in the same run. This is done so that plotting compression and plotting the noise figure wiht respect to RF input power can be done in the same simulation run. For a gain compression plot, the RF input frequency is the design RF inout frequency. .For a desensitization run, the RF

frequency would be set to the blocker frequency which is usually farther away from the LO frequency than the frequency used in this example. The process is identical. Just the frequency is changed.

On the hb *Choosing Analyses* form:

1. Set up the hb analysis appropriately.

   Because the RF power will be quite high, the number of RF harmonics is higher than normal to preserve the accuracy.

2. Sweep the RF power over an appropriate range.

   An appropriate range depends on what you are designing. Generally, you start about 15 to 25dB smaller than the compression point, and stop the sweep about 3 to 5 dB below the compression point. Addional points are defined with closer amplitude spacing so that enough points are taken to produce a smooth gain curve at high input power.

On the hbnoise *Choosing Analyses* form:

3. Usually, a single frequency is run in hbnoise, when the blocker power is swept. This is done to allow the direct plot functions to work properly after the simulatiopn.
4. Specify the desired output frequency.
5. For *Maximum sideband*, either leave the field blank which causes SpectreRF to calculate all the noise mixing frequencies from all the harmonics set in the hb *Choosing Analyses* form, or specify the same number that is used to set the highest number of harmonics in the hb *Choosing Analyses* form.
6. The rest of the properties are set up as they would be in a normal simulation. Once you have finished setting up the hb and hbnoise *Choosing Analyses* forms, run the simulation.
7. When the simulation completes, select *Results - Direct Plot - Main Form* in the ADE.

8. Select the desired noise figure measurement.

9. Click *Plot*.

The resulting plot appears.

10. To make the plot more useful, right-click one of the numbers on the X Axis and select *Swap Sweep Var* from the context menu.

The *Swap Sweep Var* window is displayed. Note that the *Sweep Variables* field is set to *prf*, which is the variable that was swept.

11. Click *OK*.

A new subwindow is displayed with the noise figure on the vertical axis, and the blocker power on the X Axis.

Noise Type = Modulated

Modulated allows the measurement of the AM and PM components of noise separately.

First, set up the hb analysis as appropriate for your circuit. For more information see the harmonic balance section of this chapter.

On the hbnoise *Choosing Analyses* form:

1. Set up an appropriate frequency range of interest.

2. Select *modulated* from the *Noise Type* drop-down list.

3. Run the simulation. Note that hbnoise modulated runs both the positive and negative frequencies. This allows both single-sideband and double-side band noise calculations to be plotted.

```
/usr2/rdavis/demos/ic61/docs_database/simulation/4_inv_tb1/spectre
File  Help                                                    cadence
*****************************************************************
    hbnoise_mod1: freq = 222.3 kHz   (6.25 %), step = 97.28 kHz    (6.
    hbnoise_mod1: freq = 395.3 kHz   (12.5 %), step = 173 kHz      (6.
    hbnoise_mod1: freq = 702.9 kHz   (18.8 %), step = 307.6 kHz    (6.
    hbnoise_mod1: freq = 1.25 MHz      (25 %), step = 547.1 kHz    (6.
    hbnoise_mod1: freq = 2.223 MHz   (31.2 %), step = 972.8 kHz    (6.
    hbnoise_mod1: freq = 3.953 MHz   (37.5 %), step = 1.73 MHz     (6.
    hbnoise_mod1: freq = 7.029 MHz   (43.8 %), step = 3.076 MHz    (6.
    hbnoise_mod1: freq = 12.5 MHz      (50 %), step = 5.471 MHz    (6.
    hbnoise_mod1: freq = 22.23 MHz   (56.2 %), step = 9.728 MHz    (6.
    hbnoise_mod1: freq = 39.53 MHz   (62.5 %), step = 17.3 MHz     (6.
    hbnoise_mod1: freq = 70.29 MHz   (68.8 %), step = 30.76 MHz    (6.
    hbnoise_mod1: freq = 125 MHz       (75 %), step = 54.71 MHz    (6.
    hbnoise_mod1: freq = 222.3 MHz   (81.2 %), step = 97.28 MHz    (6.
    hbnoise_mod1: freq = 395.3 MHz   (87.5 %), step = 173 MHz      (6.
    hbnoise_mod1: freq = 702.9 MHz   (93.8 %), step = 307.6 MHz    (6.
    hbnoise_mod1: freq = 1.25 GHz     (100 %), step = 547.1 MHz    (6.
Total time required for hbnoise analysis `hbnoise_mod1': CPU = 7.66683
Time accumulated: CPU = 11.3693 s, elapsed = 17.0253 s.
Peak resident memory used = 48.6 Mbytes.


Notice from spectre during HBNOISE analysis `hbnoise_mod2'.
    Use new Arnoldi routine


*****************************************************************
HB Noise Analysis `hbnoise_mod2': freq = (-125 kHz -> -1.25 GHz)
*****************************************************************
    hbnoise_mod2: freq = -125 kHz       (0 %), step = -125 kHz
    hbnoise_mod2: freq = -222.3 kHz  (6.25 %), step = -97.28 kHz   (6.
    hbnoise_mod2: freq = -395.3 kHz  (12.5 %), step = -173 kHz     (6.
    hbnoise_mod2: freq = -702.9 kHz  (18.8 %), step = -307.6 kHz   (6.
    hbnoise_mod2: freq = -1.25 MHz     (25 %), step = -547.1 kHz   (6.
    hbnoise_mod2: freq = -2.223 MHz  (31.2 %), step = -972.8 kHz   (6.
    hbnoise_mod2: freq = -3.953 MHz  (37.5 %), step = -1.73 MHz    (6.
    hbnoise_mod2: freq = -7.029 MHz  (43.8 %), step = -3.076 MHz   (6.
    hbnoise_mod2: freq = -12.5 MHz     (50 %), step = -5.471 MHz   (6.
    hbnoise_mod2: freq = -22.23 MHz  (56.2 %), step = -9.728 MHz   (6.
    hbnoise_mod2: freq = -39.53 MHz  (62.5 %), step = -17.3 MHz    (6.
    hbnoise_mod2: freq = -70.29 MHz  (68.8 %), step = -30.76 MHz   (6.
    hbnoise_mod2: freq = -125 MHz      (75 %), step = -54.71 MHz   (6.
    hbnoise_mod2: freq = -222.3 MHz  (81.2 %), step = -97.28 MHz   (6.
    hbnoise_mod2: freq = -395.3 MHz  (87.5 %), step = -173 MHz     (6.
    hbnoise_mod2: freq = -702.9 MHz  (93.8 %), step = -307.6 MHz   (6.
    hbnoise_mod2: freq = -1.25 GHz    (100 %), step = -547.1 MHz   (6.
6
```

Some measurements are single-sideband and some measurements are double-sideband. The next steps will show the differences.

Now plot the output noise using the *Direct Plot Form*.

On the *Direct Plot Form*:

1. Select *hbnoise* results. All these measurements are single-sideband.
2. Select *Output Noise* from the *Function* section.

3. Click Plot.
4. Select *hbnoise modulated* results. All these measurements are double-sideband.

5. Select *AM* or *PM* from the *Noise Type* radio buttons. The *AM* noise is selected below.

6. Click *Plot*.

On the plot, note that the AM noise is 3dB larger than the output noise. This is because the output noise is derived from the positive frequencies only, and the AM and PM noise is derived using the positive and negative frequencies.

Noise Type = Jitter

Driven jitter is usually used for measuring the noise in a digital circuit at a specific threshold crossing. This is usually the voltage that causes succeeding stages to be clocked. In this

case, *Decide automatically* and *auto* harmonics are suggested. These settings will automatically set the number of harmonics appropriately for the circuit.

An appropriate threshold is needed, which can be determined by plotting the time domain waveform from the hb *Direct Plot Form*. The waveform is an ifft of the harmonics that were

solved in harmonic balance. The waveform below shows an appropriate threshold value to be 1.65V. In most CMOS logic circuits, the clocking voltage is half the supply voltage.

On the hbnoise *Choosing Analyses* form, the stop frequency is half the input frequency. The start frequency is four decades below that. About five points per decade is usually sufficient. *Maximum sideband* should be left blank. The *Output* field is set to *voltage*, and the *Input*

Source field is set to *none* because usually, only the output jitter is needed. The threshold is set appropriately. The *Crossing Direction* option can be set to *all*, *rise*, or *fall*.

Choosing Analyses – Virtuoso® Analog Design Environn
Analysis
tran
dc
ac
noise
xf
sens
dcmatch
stb
pz
sp
envlp
pss
pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
hbsp
Harmonic Balance Noise Analysis
Sweeptype
relative
Relative Harmonic
0
Output Frequency Sweep Range (Hz)
Start-Stop
Start
125K
Stop
1.25G
Sweep Type
Logarithmic
Points Per Decade
Number of Steps
5
Add Specific Points
Sidebands
Maximum sideband
When using hb engine, default value is harms of 1st tone.
Output
voltage
Positive Output Node
/Out1
Select
Negative Output Node
Select
Input Source
none
Do Noise
Noise Type
jitter
jitter: jitter measurement at the output
PM jitter for driven circuit
Signal
Threshold Value
Crossing Direction
/Out1
1.65
all
Enabled
Options...
OK
Cancel
Defaults
Apply
Help

When the simulation completes, use the *Direct Plot Form* to plot the jitter. For driven circuits, select *Jee*, which is edge-to-edge jitter. The event time is the time in the ifft from the hb analysis. In the below example, 77p is the rising edge, and 275p is the falling edge. *Event Time* makes the selction of the rising and falling edge. In the *Modifier* section, *Second* gives the Jee calculation in seconds. *UI* is Unit Interval (fractions of one period), and *ppm* is parts per million.

The output noise frequency response and the edge-to-edge jitter are displayed. The jee measurement is one standard deviation for the expected jitter.

Noise Type = Timedomain

Timedomain noise is used to measure the output noise voltage at several timepoints in the ifft. This capability is simialar to `noisetype=jitter`, but instead of being limited to a single threshold value, you can specify multiple times, and instead of having Jee directly in the *Direct Plot Form*, the calculation has to be done manually.

The hb setup is the same as the setup for jitter.

First run the hb analysis and plot the time-domain waveform. Position the markers at interesting timepoints, as shown below.

On the hbnoise *Choosing Analyses* form:

1. Select the *Number of Points* check box and type 1 in the text field. This causes the first noise point at time zero to be calculated.

2. Select the *Add Specific Points* check box and enter the time values from the markers positioned in the waveform.

3. Run the simulation.

   Because pnoise runs in the frequency-domain, to calculate the noise voltage, we need to integrate the frequency-domain data to calculate the noise. The hbnoise *Choosing Analyses* form defines the frequency range, and the number of frequency translations to consider.

On the *Direct Plot Form*:

1. Select *tdnoise* from the *Analysis* section.

2. Select *HB Integ Output Noise* from the *Function* section.
3. To get the noise voltage, select *Magnitude* from the *Modifier* section.
4. The frequency range in the *Direct Plot Form* should be the same as the frequency range defined in the hbnoise *Choosing Analyses* form. This integrates the noise with all the frequency translations over the frequency range defined in the hbnoise setup.
5. Click *Plot*.

The noise voltage is plotted versus time, as shown below. As expected, the maximum noise occurs near the middle of the waveform. Markers have been placed in both subwindows at the same times.

To get the jitter time, take the expression for the integrated noise voltage and divide by the slew rate. To determine the slew rate, zoom in to an area near the voltage you want for the measurement.

Place a marker at a voltage just below the threshold voltage you desire by moving the mouse near that point, and typing m. Now move to a point just above that voltage, and type d. The slew rate is `dy` divided by `dx`.

Oscillators

Noise Type = Sources

Noise type=sources is used to measure the phase noise that is averaged over the entire cycle of the oscillator waveform.

1. First, set variables to define the tuning voltage (vt) and power supply voltages (VDD) in the ADE.

2. Set up the hb form as shown in the following figure. For more information, see the hb section at the beginning of this chapter.

3. Next set up the hbnoise form (More information is provided below).

4. The sweep defaults to relative to the frequency of the specified harmonic of the hb analysis. The frequencies specified in the *Choosing Analyses* form are added to the frequency of the specified harmonic when *relative* is selected. In this case, the first harmonic is specified because there no frequency multiplier on the output of the oscillator.

5. Relative to the oscillator frequency, specify a frequency offset.
6. Either specify the number of harmonics for the hb, or leave the *Maximum sideband* field blank, which has the same effect.
7. Set *Output* to *voltage*.
8. Select the output node.
9. Set *Input Sources* to *none*. Input-referred noise is not defined for an oscillator.
10. If you want noise separation, you must first select *sources* from the *Noise Type* drop-down list and then you can select the noise separation.
11. Run the simulation.
12. To plot the single-sideband phase noise, in the ADE, select *Results - Direct Plot - Main Form*.

On the *Direct Plot Form*:

1. Select *hbnoise* from the *Analysis* section.

2. Select *Phase Noise* from the *Function* section.

Note that the phase noise from the hbnoise result set plots the noise that is in the frequencies above the oscillator frequency only (single sideband phase noise). If the double sideband phase noise (which includes the noise from below and above the oscillator output frequency) is desired, set the *Noise Type* to *jitter*. This is documented later.

3. Click *Plot*.

Note that the phase noise continues above 0dBC/Hz, which indicates that the phase noise is larger than the amplitude of the oscillator output. This is because hbnoise is a small-signal analysis and it does not recognize the large-signal limits. The CorFreq label gives the frequency where the phase noise curve levels off at low frequency offset.

Commonly Used hbnoise Options for Oscillators

For the rest of the options, please see the Commonly Used hbnoise Options.

Augmented

augmented is the default and is required for most oscillators that have long time constants, perhaps from a Low Drop-Out (LDO) regulator. Either take the default (*yes*) or set *augmented* to *yes* for every circuit. Augmented is a new algorithm that works better than the

standard hbnoise algorithm for the measurement of close-in phase noise especially when there are low frequency poles in the circuit.

Lorentzian

There are two ways of looking at phase noise. One way is to think of measuring the spectral output of the oscillator with a network analyzer. In this case, as the frequency gets close to the oscillator frequency, the amplitude starts going up, levels off, and then drops as the oscillator frequency is passed. In this case, the noise cannot be larger than the oscillations themselves, which is 0 dBc. If this is how you view phase noise, set the *lorentzian* option to *yes*.

If you think in terms of jitter, in one cycle of the oscillator output, because of the noise, you can calculate one standard deviation of the timing. If you go two cycles, and you integrate the noise with respect to time, you get twice as much noise power, or about 1.4 times the noise voltage or jitter. When you get to an infinite number of oscillator cycles, the jitter also becomes infinite. As you increase the time, you are lowering the noise frequency, and in this case, as the time becomes large, the frequency becomes very low. At infinite time, you get infinite jitter, and this occurs at zero offset from the carrier. Jitter is just another way of looking at phase noise. In other words, the phase noise can easily exceed the carrier amplitude (0 dBc) as the offset frequency becomes small because it is heading to infinity at zero frequency offset. If this is how you view phase noise, set the *lorentzian* option to *no*.

Lorentzian=yes means calculate the leveling off. *Lorentzian=cornerfreqonly* calculates the continuously rising phase noise, but it places a marker on the phase noise curve at the frequency where the phase noise would level off. *no* causes the phase noise to continue to rise as the offset frequency approaches zero.

Comparison Between Lorentzian Yes and No

As noted above, lorentzian causes leveling off of the phase noise at low frequency.

Note that when *lorentzian=no*, the phase noise curve goes above 0 dBc. This would mean that the noise is larger than the amplitude of the oscillator. This is because hbnoise is a small-signal analysis that does not recognize the large-signal limits of the circuit like clipping. It is

similar to AC where you can specify an input magnitude of 1 megavolt, and see 10 megavolts on the output.

Noise Type = Modulated

1. First set the hb form, as shown in the following figure. For more information, see the Harmonic Balance section at the beginning of this chapter.

Because your circuit is different, you might need a different estimate of the oscillation frequency and number of harmonics.

Next, set the hbnoise form.

2. Select *modulated* from the *Noise Type* drop-down list.

3. Run the simulation.

Note that positive and negative frequencies are run in hbnoise. Since *relative* was chosen for the *Sweep Type*, the frequencies above and below the first harmonic are analyzed for noise.

```
/usr2/rdavis/demos/ic61/docs_database/simulation/4_inv_tb1/spectre
File  Help                                                    cadence
*****************************************************************
    hbnoise_mod1: freq = 222.3 kHz   (6.25 %), step = 97.28 kHz    (6.
    hbnoise_mod1: freq = 395.3 kHz   (12.5 %), step = 173 kHz      (6.
    hbnoise_mod1: freq = 702.9 kHz   (18.8 %), step = 307.6 kHz    (6.
    hbnoise_mod1: freq = 1.25 MHz      (25 %), step = 547.1 kHz    (6.
    hbnoise_mod1: freq = 2.223 MHz   (31.2 %), step = 972.8 kHz    (6.
    hbnoise_mod1: freq = 3.953 MHz   (37.5 %), step = 1.73 MHz     (6.
    hbnoise_mod1: freq = 7.029 MHz   (43.8 %), step = 3.076 MHz    (6.
    hbnoise_mod1: freq = 12.5 MHz      (50 %), step = 5.471 MHz    (6.
    hbnoise_mod1: freq = 22.23 MHz   (56.2 %), step = 9.728 MHz    (6.
    hbnoise_mod1: freq = 39.53 MHz   (62.5 %), step = 17.3 MHz     (6.
    hbnoise_mod1: freq = 70.29 MHz   (68.8 %), step = 30.76 MHz    (6.
    hbnoise_mod1: freq = 125 MHz       (75 %), step = 54.71 MHz    (6.
    hbnoise_mod1: freq = 222.3 MHz   (81.2 %), step = 97.28 MHz    (6.
    hbnoise_mod1: freq = 395.3 MHz   (87.5 %), step = 173 MHz      (6.
    hbnoise_mod1: freq = 702.9 MHz   (93.8 %), step = 307.6 MHz    (6.
    hbnoise_mod1: freq = 1.25 GHz     (100 %), step = 547.1 MHz    (6.
Total time required for hbnoise analysis `hbnoise_mod1': CPU = 7.66683
Time accumulated: CPU = 11.3693 s, elapsed = 17.0253 s.
Peak resident memory used = 48.6 Mbytes.


Notice from spectre during HBNOISE analysis `hbnoise_mod2'.
    Use new Arnoldi routine


*****************************************************************
HB Noise Analysis `hbnoise_mod2': freq = (-125 kHz -> -1.25 GHz)
*****************************************************************
    hbnoise_mod2: freq = -125 kHz       (0 %), step = -125 kHz
    hbnoise_mod2: freq = -222.3 kHz  (6.25 %), step = -97.28 kHz   (6.
    hbnoise_mod2: freq = -395.3 kHz  (12.5 %), step = -173 kHz     (6.
    hbnoise_mod2: freq = -702.9 kHz  (18.8 %), step = -307.6 kHz   (6.
    hbnoise_mod2: freq = -1.25 MHz     (25 %), step = -547.1 kHz   (6.
    hbnoise_mod2: freq = -2.223 MHz  (31.2 %), step = -972.8 kHz   (6.
    hbnoise_mod2: freq = -3.953 MHz  (37.5 %), step = -1.73 MHz    (6.
    hbnoise_mod2: freq = -7.029 MHz  (43.8 %), step = -3.076 MHz   (6.
    hbnoise_mod2: freq = -12.5 MHz     (50 %), step = -5.471 MHz   (6.
    hbnoise_mod2: freq = -22.23 MHz  (56.2 %), step = -9.728 MHz   (6.
    hbnoise_mod2: freq = -39.53 MHz  (62.5 %), step = -17.3 MHz    (6.
    hbnoise_mod2: freq = -70.29 MHz  (68.8 %), step = -30.76 MHz   (6.
    hbnoise_mod2: freq = -125 MHz      (75 %), step = -54.71 MHz   (6.
    hbnoise_mod2: freq = -222.3 MHz  (81.2 %), step = -97.28 MHz   (6.
    hbnoise_mod2: freq = -395.3 MHz  (87.5 %), step = -173 MHz     (6.
    hbnoise_mod2: freq = -702.9 MHz  (93.8 %), step = -307.6 MHz   (6.
    hbnoise_mod2: freq = -1.25 GHz    (100 %), step = -547.1 MHz   (6.
6
```

4. To plot the phas noise, in the ADE window, select *Results - Direct Plot - Main Form*.

5. On the *Direct Plot Form,* plot the phase noise from the *hbnoise* results. Note that hbnoise results are always single-sideband measurements. Only the frequencies above the oscillator poutput frequency are considered for all the available plots in the hbnoise result set.

6. Next, plot the phase noise from the *hbnoise jitter* result set. All the plottable items in the hbnoise jitter result set are always double-sideband measurements.

Note that the phase noise from the pnoise result contains the noise in the frequencies above the oscillator frequency only, and the phase noise from the hbnoise jitter results

contains the noise power from above and below the oscillator frequency. This is the reason for the 3.01 dB difference.

7. Select *hbnoise modulated* from the *Function* section.

8. Select *AM* or *PM* from the *Noise Type* section.

   Note that the AM and PM noise are also double sideband measurements.

9. Click *Plot*. Both the AM and PM plots have been added to the phase noise plots from before in the plot below. The PM component and the dsb phase noise are almost

identical. This shows that the AM and PM componets are double-sideband measurements.

To plot cycle jitter or cycle-to-cycle jitter:

1. On the *Direct Plot Form*, select *hbnoise jitter* from the *Analysis* section.

2. Select *Jc* or *Jcc* from the *Function* section.

3. The frequency range will default to the range set in the *Choose Analyses* form.

   Cycle jitter (sometimes called period jitter) and cycle-to-cycle jitter are calculated by integrating the noise in the phase noise curve. Cycle jitter fixes the first threshold in time, and measures one standard deviation of the variation of the crossing of the second edge.

Cycle-to-cycle jitter measures one standard deviation of the variation in the period of the waveform.

4. Click *Plot*.

   The Jc or Jcc calculation is shown as a label that is added to the waveform tool.

Noise Type = Jitter (FM)

Noise Type = Jitter (FM) produces the same output as Noise Type = Modulated. See Noise Type = Modulated on page 455.

Noise Type = Jitter (PM)

This analysis gives an instantaneous measurement of jitter at a threshold voltage on the timedomain waveform. This is useful for measuring the jitter of an oscillator that is driving a

digital component. This is the oscillator equivalent of *noisetype=jitter* for driven circuits. Both of these are variations of *noisetype=timedomain*. Timedomain noise calculates just the instantaneous noise. FM Jitter also calculates the noise, however, it also determines the slew rate of the output, and does the noise divided by the slew rate calculation to calculate the actual time-domain jitter.

1. First, set up the hb analysis. For more information on this, see the harmonic balance section in this chapter.

2. Next set up the hbnoise *Choosing Analyses* form.

In the *Choosing Analyses* form:

- Select *jitter* from the *Noise Type* drop-down list.

- Select the *PM* radio button.
- Specify a threshold voltage for the time-domain waveform in the *Threshold Value* field.
- Specify a crossing direction. In this case, *all* is selected so that all the threshold crossings are calculated.
- The stop frequency should be very close to half the oscillator frequency.
- The start frequency should be approximately the frequency where the phase noise reaches 0 dBc.

3. Run the simulation.

4. In the ADE window, select *Results* - *Direct Plot* - *Main Form*.

On the *Direct Plot Form*, first plot the timedomain result.

Now plot the cycle jitter (Jc) or the cycle-to-cycle jitter.

Note that there is a selection of event times if you selected *all* for *Crossing Direction*. This is the time on the waveform where the calculation was made. Note that this will not plot unless there is at least one waveform displayed in the waveform tool.

The picture below shows the time-domain waveform, and the jitter calculations for the falling edge and the rising edge.

Noise Type = Timedomain

This analysis type provides a relative measurement of the noise at each time in the ifft. Note that if you have a lot of harmonics and/or a large oversample factor, there will be many timepoints in the ifft. A method will be presented to reduce the number of hbnoise points.

1. First, set up the hb analysis for the oscillator. For more information, see the harmonic balance section at the beginning of this chapter.

2. Next, set up the hbnoise analysis.

In the hbnoise *Choosing Analyses* form:

- The stop frequency should be near one half the frequency of the oscillator.
- The start frequency should be near the point where the phase noise curve reads 0 dBC.
- Select *timedomain* from the *Noise Type* drop-down list.
- The default *Noise Skip Count* is 0 (zero).

  An hbnoise run will always be run at time = 0. A noise skip count of 2 means run the timepoint at 0 seconds, then skip 2 ifft points, then run the next one, then skip 2, and so on.

3. Run the simulation.

4. In the ADE window, select *Results* - *Direct Plot* - *Main Form*.

5. Plot the output time-domain waveform (the ifft).

6. Now plot the integrated output noise.

The waveform tool displays both results. Markers have been placed at the largest noise points, and at the times for those maximums.

The noise is highest near the zero crossing, and lowest at the peaks and valleys of the waveform.

hbSP Analysis

Overview of hbsp

Hbsp is an extension to the idea of linear S-Parameters, so that S-Parameters can be measured for circuits that translate frequency. In linear S-Parameters, a wave is launched from the input and that wave is usually partially reflected at the circuit connection, and it propagates to the output where it can be measured. Then a wave is launched at the output source, some of it is reflected back to the output, and a wave appears at the input. Using this idea, linear S-Parameters can be used effectively to characterize many linear systems.

With hbsp, a wave is launched at the input just like in S-Parameter analysis. Some of this wave is reflected back to the input, depending on the input match. Because the system is nonlinear, this wave can mix with harmonics of the periodic source that is driving the circuit to cause waves to appear at the output at many different mixing frequencies. Hbsp supports a 2-port simulation where one of the output mixing products can be measured at a time.

As an example, imagine a mixer that has a 1.1GHz RF input, and a 1GHz Local Oscillator signal. The forward wave is launched at 1.1GHz, and at the output of the mixer 1.1GHz is produced with no frequency translation. This measures the RF to IF isolation. 100MHz and 2.1GHz are the mixing products produced by mixing with the first harmonic of the LO signal. For a down conversion mixer, the 100MHz term is desired to measure the conversion gain. One of these waves can be measured in hbsp. Like linear S-Parameters, hbsp measures the power gain of the system.

For the reverse gain and match calculation, the input is at the output frequency of the mixer, and the output is at the input frequency of the mixer. The forward gain path is from 1.1GHz to 100MHz, and the reverse gain path is from 100MHz to 1.1GHz. Therefore, S11 is a measure of the input match at the RF frequency, and S22 is the output match at the IF frequency. S21 measures the forward conversion gain, and S12 measures the reverse conversion gain.

Example

Consider a double-balanced diode mixer, as shown below.

The properties of the LO source are as follows. The frequency and amplitude are set to variable names so they can be changed in the ADE without changing the schematic.

Edit Object Properties
Apply To
only current
instance
Show
system
user
CDF
Browse
Reset Instance Labels Display
Property
Value
Display
Library Name
analogLib
Cell Name
port
View Name
symbol
Instance Name
LO
off
Add
Delete
Modify
User Property
Master Value
Local Value
Display
IvsIgnore
TRUE
CDF Parameter
Value
Display
Port mode
Normal
HarmonicPort
Resistance
50 Ohms
Reactance
Port number
DC voltage
Source type
sine
Frequency name 1
LO
Frequency 1
flo Hz
Amplitude 1 (Vpk)
vlo V
Amplitude 1 (dBm)
Phase for Sinusoid 1
Sine DC level
Delay time
Display second sinusoid
Display multi sinusoid
Display modulation params
Display small signal params
Display temperature params
Display noise parameters
Multiplier
OK
Cancel
Apply
Defaults
Previous
Next
Help

The RF source is similar, but with different variable names.

The variables are set in the ADE. The *frf* variable is set to 0 (zero) in order to disable the large-signal waveform. Only the *LO* is applied at 1GHz. 1.4 volts is +13 dBm.

Next, set up the hb *Choosing Analyses* form. The hb analysis is used to characterize the nonlinearity of the system and measures the amplitude of the harmonics.

Harmonics are set to auto so that the simulator will choose the number of harmonics automatically, and oversample factor is raised above one because the currents in the diodes

are very nonsinusoidal. For more information on setting the oversample factor, refer to the harmonic balance section at the beginning of this chapter.

Down Conversion and RF to IF Isolation hbsp Setup

1. Open the hbsp *Choosing Analyses* form from the ADE window.

2. Specify the input or output frequency range in the *Start* and *Stop* fields.

   Both up conversion and the down conversion will be shown.The LO frequency is 1GHz as was shown in the harmonic balance *Choosing Analyses* form. In the figure below, the frequency range chosen is 1.05G to 1.15G. For down conversion, this is the input frequency range. The down-converted output is from 50M to 150M.

3. Enter 1 in the *Port #* column.

4. Click *Select Port* and select the input port in the circuit.

5. Click *Choose Harmonic*.

The Choose Harmonic window is displayed.

6. In the *Choose Harmonic* window, select the input frequency range.
7. Click *OK*.
8. Click *Add*.

   Port 1 is added to the list with the input frequency range.

9. Follow the steps from 2 through 8 to add a second IF output port at the downconverted frequency. The second port is added, as shown below.

10. Follow the steps from 2 through 8 to add a third IF output port at the non-converted frequency. This will measure the RF to IF isolation. The third port is added, as shown below.

The noise analysis is the same as the noise analysis in hbnoise. Refer to the hbnoise section for details. In the figure below, noise has been disabled because we will not be performing noise analysis.

11. Run the simulation.

12. When the simulation completes, select *Results - Direct Plot - Main Form* in the ADE window.

The *Direct Plot Form* is displayed.

13. In the *Direct Plot Form*, select *hbsp* from the *Analysis* section.

14. Now plot S11, S21, and S22 by clicking the buttons near the bottom of the *Direct Plot Form*.

The waveform window is displayed, as shown below.

Now plot the RF to IF isolation by selecting S31.

The RF to IF isolation in the ideal case is quite high at just over 114dB.

UP Conversion hbsp Setup

Remember that the LO is at 1GHz as it was before. Now we will reverse the input and output frequencies. The input is from 50M to 150M, and there are two outputs. One output is from 850M to 950M and the other output is from 1.05G to 1.15G. We will focus on the output at 1.p05G to 1.15G in this example.

The frequency sweep range is unchanged at 1.05G to 1.15G.

For up-conversion, reset the hbsp *Choosing Analyses* form and follow the same steps to specify the frequency for the two ports, as shown in the following figure.

Note that the input and output frequencies have changed. Now the input is from 50M to 150M, and the output is from 1.05G to 1.15G. This is done using the same *Choose Harmonic* button as before.

Also, since the gain with no frequency translation from RF to IF is not made because measuring it at 100MHz to 100MHz has little value. The 50M to 150M term at the output of an up-conversion mixer is always filtered out anyway.

The plots of the S-Parameters are shown below.

All the calculations that are available in sp are available in hbsp. Some things like the stability factor Kf are only defined when the input frequency and the output frequency are the same. When parameters become invalid because of frequency differences between the input and output, the *Direct Plot Form* displays an error, and the parameter does not plot.

The difference between sp and hbsp is that in sp, there is no signal applied to the circuit. In hbsp, a signal is applied to the circuit. The S-Parameters are measured with that signal applied. Also, many times the input and output frequencies are different in hbsp, and by definition, linear S-Parameters cannot change frequencies.

In the example above, the signal in the harmonic balance large-signal analysis was an LO signal for a mixer, but in the example below, the signal in harmonic balance is the RF input to

a power amplifier. In the next example, the input and the output are at the same frequency and the input power is swept.

Amplifier with Swept Input Power

Imagine an amplifier with a tiny amplitude RF input signal at 1GHz. In this case, the amplifier is very linear, and if an S-parameter measurement is done at a small offset frequency, for example, 1.01GHz, the same measurement is obtained as a linear S-Parameter measurement at 1.01GHz.

Now imagine that the input signal at 1GHz is near the compression point of the amplifier. Because the system is much more nonlinear, if a small-signal S-Parameter measurement is done at 1.01GHz, the S-Parameters would be different. Because the input signal is large, the instantaneous input impedance changes, and this changes the input match at 1.01GHz. Similarly, the other S-Parameter measurements would be affected.

Hbsp runs after hb, so it can be used to characterize the small-signal performance with the large signal present. This is not a large-signal S-parameter measurement, but it is close. If a true large-signal S-Parameter measurement is needed, use the LSSP wizard to run that simulation. Only S11, S22, S21, and S12 can be measured with the LSSP wizard.

1. Set the hb *Choosing Analyses* form as appropriate for your amplifier, as shown below. The delta-power at high input power is small by adding the additional points to the

Choosing Analyses form. At high power, things change quickly, so more resolution is needed.

2. In the hbsp *Choosing Analyses* form, set a single frequency point, as shown below.

Note that both *PORT1* and *PORT2* are set to the same frequency. Use a single frequency point so the gain and stability functions return a single point in the sweep. You can use multiple points if you are familiar with the calculator.

3. Run the simulation.

4. When the simulation completes, select *Results* - *Direct Plot* - *Main Form* in the ADE window.

The *Direct Plot Form* is displayed.

Plotting S21

In the *Direct Plot Form*:

1. Select the *hbsp* radio button from the *Analysis* section.
2. Select the *SP* radio button from the *Function* section.
3. Select the *dB20* radio button from the *Modifier* section.
4. Click *S21*.

   The waveform displays *S21* as a function of input power.

Plotting Kf

To plot the stability factor Kf, select the *Kf* radio button in the *Function* section and click *Plot*, as shown below. For the actual equations used for the different measurements below, refer to Chapter 10, “AnalogLib Components Used in RF Simulation,”.

The waveform window displays Kf as a function of input power.

Plotting Noise Figure

To plot the Noise Figure, select the *NF* radio button in the *Function* section and click *Plot*, as shown below.

The waveform window displays noise figure as a function of RF input power.

Note that this is a power amplifier and not an LNA. Most LNAs have a much lower noise figure in the low RF input power region.

4

Single Input Large and Small-signal Analyses

Overview of Simulation Capabilities

SpectreRF offers two different modes, Spectre and APS. Spectre uses traditional simulation techniques. APS makes the CMOS models more efficient and allows multi-threading for the device current evaluation. In addition, APS adds full parallel solving of all the matrices, thus speeding up the simulation even more. There is no loss in accuracy because there is no change in the actual equations that are simulated, or in the options that control the accuracy of the simulation. These choices are available and are implemented for periodic steady-state (pss) shooting and pss harmonic balance. Both large-signal and small-signal `pac`, `pxf`, `psp`, `pstb`, and `pnoise` analyses have the Spectre and APS implementations.

Overview of Periodic Steady-State (pss) Analysis

Pss calculates the settled time and frequency domain response of your system. Harmonic balance and shooting engines are available. Harmonic balance solves in the frequency domain and uses an ifft to calculate the time-domain response. Shooting calculates the settled time-domain waveforms and calculates the frequency-domain content using a Fourier transform.

All the harmonics of the beat frequency are calculated in the pss analysis up to the maximum harmonic specified in the *Choosing Analyses* form.

Pss needs to simulate an integer number of cycles. This limits the practical application of pss to single-input frequency circuits, or multiple-input frequency circuits that have a relatively high common multiple frequency, called the beat frequency. For example, if 1GHz and 1.1GHz frequencies are applied, the beat frequency is 100M. Solving for the fifth harmonic of the 1.1GHz input requires that 55 harmonics be calculated. This is a reasonable problem for pss. If the inputs are 1GHz and 1.001GHz, 5005 harmonics are needed to calculate through the fifth harmonic of 1.001GHz. This is not a good application for pss. If you need to simulate this kind of system, use qpss-harmonic balance or hb analysis instead.

Example

A behavioral amplifier circuit is shown below.

The properties for the amplifier are shown below.

The amplifier has input and output impedances of 50 ohms and the gain is 20dB.

The input source on the left of the schematic is called a port. The properties of the port are shown below.

The port is set to generate sinusoids, and one signal is defined. For the pss analysis, you must enter a name for the signal in the *Frequency name 1* field. Variable names are used to set the frequency and amplitude so that the frequency and amplitude can be changed in the analog design environment (ADE) without needing to change the schematic. The ADE setup is shown below.

The input frequency is 2.4GHz and the input power is -30 dBm.

Next, the pss analysis is set up, as shown below.

Choosing Analyses – Virtuoso® Analog Design Environn
Analysis
tran
dc
ac
noise
xf
sens
dcmatch
stb
pz
sp
envlp
pss
pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
hbsp
Periodic Steady State Analysis
Engine
Shooting
Harmonic Balance
Fundamental Tones
Name Expr Value Signal SrcId
1 RF frf 2.4G Large PORT0
Large
Clear/Add
Delete
Update From Hierarchy
Beat Frequency
Beat Period
2.4G
Auto Calculate
Output harmonics
Number of harmonics
6
Accuracy Defaults (errpreset)
conservative
moderate
liberal
Additional Time for Stabilization (tstab)
Save Initial Transient Results (saveinit)
no
yes
Oscillator
Sweep
New Initial Value For Each Point (restart)
no
yes
Loadpull
Enabled
Options...
OK
Cancel
Defaults
Apply
Help

The pss *Choosing* Analyses form is set to a single input at 2.4GHz with 6 harmonics and moderate accuracy.

The simulation is run, and then in the ADE, *Results - Direct Plot - Main Form* is selected. The *Direct Plot Form* is displayed, as shown below.

Here, dBm is calculated based on the voltage on a node. At the bottom of the *Direct Plot Form* are directions. In this case, the net on the schematic needs to be selected.

The waveform window is shown below.

The basic display shows the harmonics that were calculated by pss. Markers have been placed on the first harmonics of the input and output. The input is -30dBm, and the output is -10dBm.

Harmonic Balance

There are three ways to run Harmonic Balance simulations in SpectreRF. These are:

- pss-hb
- qpss-hb
- hb

Qpss with the harmonic balance engine selected and hb are the exact same analysis.

Harmonic balance in the pss analysis is slightly different than in hb and qpss harmonic balance. In pss, it is always a single-tone simulation. This means that all the harmonics of the beat frequency are calculated by pss instead of just the harmonics of the signals and the mixing products.

Also, there are some analysis options that are available directly in the ADE GUI for pss-harmonic balance that are not in the ADE GUI for hb. Although you do not require these options, you can still use an option that is not in the ADE GUI for hb by typing `option_name=value` in the *AdditionalParams* field for the option that you want to use in hb.

In the example below, there are two input frequencies at 2.4GHz and 2.5GHz.

The simulation was run with 135 harmonics so that the fifth harmonic of 2.5GHz was captured. Usually, running hb or qpss-harmonic balance is faster because fewer harmonics need to be calculated. This example is for reference only.

Choosing Analyses – Virtuoso® Analog Design Environn
Analysis
tran
dc
ac
noise
xf
sens
dcmatch
stb
pz
sp
envlp
pss
pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
hbsp
Periodic Steady State Analysis
Engine
Shooting
Harmonic Balance
Tones
Name
Expr
Value
SrcId
In2
RF_in2
2.4G
PORT1
In1
RF_in
2.5G
PORT1
Beat Frequency
100M
Auto Calculate
Oversample Factor
Number of Harmonics
135
Accuracy Defaults (errpreset)
conservative
moderate
liberal
Convergence
Additional Time for Transient-Aided HB (tstab)
Save Initial Transient Results (saveinit)
no
yes
Harmonic Balance Homotopy Method
default
Oscillator
Sweep
New Initial Value For Each Point (restart)
no
yes
Loadpull
Enabled
Options...
OK
Cancel
Defaults
Apply
Help

The output spectrum shows that all the harmonics of 100MHz are calculated.

Again note that hb or qpss-harmonic balance requires fewer harmonics to be calculated because just the significant harmonics that the circuit actually produces are calculated.

Other than this difference, pss-harmonic balance is exactly the same as hb. For more information on harmonic balance, refer to Chapter 3, “Frequency Domain Analyses: Harmonic Balance” .

DC Principles

In order to understand the pss analysis, the fundamentals of simulation need to be discussed. Since the transient starts from a DC analysis that provides the time-zero timepoint, the DC Principles are reviewed. Since shooting pss is an extension to the transient analysis, Transient Principles on page 521 are also provided as a review.

In any simulation, there has to be an equal number of simultaneous equations as unknowns in the circuit. In simulation, the KCL equations at each node are used. The fundamental assumption is that the sum of the currents at each node has to be zero.

The capacitors and inductors can be removed from the circuit because they have no effect on the DC solution. The resistors have linear current and voltage relationships and go directly into a linear algebra matrix. Sources also go directly into the matrix. These values do not change from iteration to iteration. The nonlinearities do not have constant resistance with respect to the applied input voltage.Therefore, they are replaced by resistors and current sources which go into the matrix. Now all the entries in the matrix are linear, and the matrix can be solved. An iterative technique is used to calculate the DC bias point.

Consider the circuit shown below.

On a current versus voltage plot, the system looks, as shown below.

The intercept where the battery/resistor current coming in is equal to the diode current going out is the exact solution to the problem. Since computers cannot solve algebraic equations, an iterative approach is used to calculate the solution.

A starting voltage is assumed for each nonlinearity in the circuit. The default is to assume that the device is on. For a diode, that means about 0.7 volts across the device.

In the device model, the I versus V curve is available, and the ability to calculate the tangent at any voltage is also provided. For the first iteration, a line is drawn vertically from the starting point, and at the point where the vertical line intersects the I-V curve, the tangent is calculated. The nonlinear junction is replaced with a resistor and a current source. The current source value is the negative Y-Axis intercept of the tangent, and the conductance of the resistor is the slope of the tangent. The reason for doing this is that the resistor and current sources go directly into a linear algebra matrix that can be inverted and solved for voltages

on each node. From that solution, a new tangent is calculated, that value is placed in the matrix, and it is solved again. This is shown below for the first two iterations.

Note that the iterations can continue forever, always getting closer to the correct answer, but never achieving the correct answer.

This process is used for every nonlinearity in the circuit. Note that for a more complicated device, there may also be nonlinear-controlled sources to model the output current.

Convergence

The iterations stop (and the circuit has converged) when the answer is close enough. Close enough is defined by the settings of the convergence options `reltol`, `vabstol`, and `iabstol`. Just after the linear solution is calculated, the current from the linear solution is compared with the nonlinear current from the nonlinear equation for all the nonlinearities in

the circuit. When the difference in current is less than `reltol` * nonlinear current + `iabstol` for all the devices, a second check is made. When the change in voltage from the last iteration to this one for all nodes in the circuit is less than `reltol` * (the larger of the last or this iteration voltage) + `vabstol`, then a KCL check is made for all the nodes. When the sum of the currents at each node is less than `reltol` * (largest current into the node) + `iabstol`, the solution has converged. All three criteria need to be met everywhere in the circuit to have the iterations stop. When this happens, the circuit converges and the iterations stop. Again, note that the solution is not exact. To get a more accurate solution, `reltol` and `vabstol` need to be reduced from the defaults. More accuracy in DC usually costs a few more iterations.

The default for `reltol` is `1e-3` (0.1%). The default for `vabstol` is `1e-6` (1 microvolt). The default for `iabstol` is `1e-12` (1 picoamp). By default, when the answer is within 0.1% plus a little bit, the simulator stops iterating.

With the default values, note that when the solution is 1 millivolt, the error that is allowed is 0.1% of 1 millivolt + 1 microvolt or 2 microvolts. When the solution is larger than 1 millivolt, the relative term dominates the error tolerance. Similarly, when the current reaches 1 nanoamp, the relative tolerance becomes dominant. Because most circuits have voltages and currents larger than these, the relative term (`reltol`) usually dominates the error tolerances. The absolute terms (`vabstol` and `iabstol`) apply just to a very small region near zero.

The voltage that exists from the sources in the circuit at time zero is applied, and the solution of the DC algorithm is used as the first timepoint.

Skipdc

The default for `skipdc` is `no`. Normally, a DC solution is run for the initial transient solution at time zero.

In some cases, the time zero timepoint does not converge. Using the `skipdc` option allows the simulation to proceed to the transient without a DC solution for the time zero point. In this case, an assumed solution is used as the starting point. For the nodes that have batteries connected to ground, the assumed solution is the battery voltage. For nodes that have initial conditions specified, the initial condition is used. For all other nodes, zero volts is assumed.

When `skipdc` is set to `yes`, the assumed solution is used at time zero and the transient sources start immediately. Sometimes, this causes problems for the transient, therefore, setting `skipdc` to `sigrampup` causes the time-varying to start at minus 10 percent of the stop time in the `tstab` and ramp up from zero to the final value at zero seconds. When the default for `skipdc` causes problems for the transient, easing the input signal can help the transient get started.

Transient Principles

In the transient, the input voltage is allowed to vary as a function of time, and the capacitor and inductors are added to the circuit. The capacitors are modeled using a resistor and a current source in parallel, and the inductors are modeled as a battery in series with a resistor. The values of the resistors, batteries, and current sources are adjusted at each timepoint to accurately model the correct behavior of the capacitor and inductor. Note that the matrix is still scalar (No real and imaginary part). Keeping the matrix scalar speeds up the solution at each timepoint.

The system is broken into individual timepoints, and a solution is made at each timepoint.

Convergence

At each timepoint, the system is solved for using an iterative process, just like the DC solution. The difference is that in the transient, history is used to predict the movement of the circuit, so that the number of iterations at each timepoint is reduced. The same convergence criteria is used in the transient as in the DC analysis with the exception of where the `reltol` * `voltage` or `reltol` * `current` terms come from in the circuit. This is controlled by the `relref` option.

Relref

Consider a sine wave, as shown below.

- Pointlocal criteria is `reltol` * the voltage or current at this node at this iteration or the last iteration, whichever is larger. Note that as convergence is attained, the solution is almost exactly the sinusoid, so when the voltage is small, the uncertainty is small. When the voltage is large, the uncertainty is large. This is shown below.

  Although this is the most accurate solution, more time is required for the simulation because of the very small error tolerances near zero. Usually, the local criteria, which is shown next is preferable.

- Local criteria is `reltol` * the voltage or `reltol` * `current` at this node at this or the previous iteration, or at any solution in the past. With a sinusoid starting at zero, the uncertainty is zero and gets larger as the voltage gets larger. Once the voltage reaches

the peak, the uncertainty is constant after that. This is because `reltol` * peak voltage is always larger than `reltol`* the current solution.

The error tolerance is larger after the first peak compared to `pointlocal`, but it is never larger than the value at the peak. This is a good compromise for accuracy of the simulation.

- Global criteria is `reltol` * largest voltage anywhere in the circuit now or at any time in the past. For most circuits, the largest voltage in the circuit is the power supply voltage and the largest current is the power supply current. Since the power supply voltage and

current is usually larger than the node voltages and currents, the error is larger, as shown below.

Note that when global criteria are used, the error tolerance gets much larger for low voltage nodes and low current branches. Because of this, the accuracy degrades when using global criteria.

- Note that for `pointlocal` and for local criteria at `time = zero`, when the voltage is zero, `reltol` * voltage is also zero. Since every iteration produces a different solution, an absolute term needs to be added to the relative term so that convergence can be obtained when the voltage or current signal goes through zero. For voltages, this is set by the option `vabstol`, which defaults to 1 microvolt. (`1e-6`) For currents, this is set by

the option `iabstol` which defaults to 1 picoamp (`1e-12`). This small absolute tolerance is added to the relative term at all timepoints. The absolute term is shown below.

The `relref` option affects the transient-based analysis only (tran, pss shooting, qpss shooting, and envelope shooting). It uses different combinations of criteria (`pointlocal`, `local`, `global`) for the different settings of `relref`.

	Pointlocal	**Alllocal**	**Sigglobal**	**Allglobal**
Device Current Check	Pointlocal	Local	Global	Global
Delta-V Check	Pointlocal	Local	Global	Global
KCL Check	Pointlocal	Local	Local	Global

Note that in the order of most accurate to least accurate, the `relref` settings are `pointlocal`, `alllocal`, `sigglobal`, and `allglobal`. The default for most SpectreRF simulations is `alllocal`.

Integration Methods

Trapezoidal Method

In the transient, numerical integration is used for integrating current to get voltage on capacitors and for integrating voltage to get current in inductors. One method is to assume that the current is constant at the average current that flows during the interval. Because the current is constant, the voltage is a ramp during the timestep. The waveform that might be produced is shown below.

Note that the real waveform is usually a curve. This is shown below.

Note that the area between the straight line and the curve is the charge that is in the physical system, but not in the simulation. Errors are introduced in the simulation because of a finite timestep that is in the simulation. The timestep cannot go infinitely small because it would take infinitely long to run the simulation. As a consequence, numerical integration errors are inherent in the time-domain simulation because of the finite timestep.

This error can be controlled by making the timestep smaller, as shown below.

Note that by taking twice as many timepoints, the sum of the errors is much less than the previous case. One way to accomplish this is to set a small maximum timestep. Although this produces the most accurate waveform, the runtime cost is usually quite high because the maximum timestep is used everywhere in the simulation.

When the waveform is at a constant value, there is no numerical integration error. The place where the error occurs is the curvature that is produced by the circuit. Spectre controls the timestep based on the curvature that is produced by the circuit. Because of this, the timestep

is inherently a variable, as shown in the below waveform. The X symbols on the waveform indicate the place where a timepoint exists in the transient output data.

Note that where the waveform is flat, the timestep bumps out, and where there is a high curvature, lots of points are taken. In this way, the accuracy of the waveform is preserved with the least possible number of timepoints. This is done to reduce the runtime and is the normal method of timestep control.

The trapezoidal method is generally good for normal transient simulations because the method does not exaggerate or damp any ringing that the circuit might produce. Although it is not visible in the waveform above, the trapezoidal method has ringing. At each point, there is an error term that alternates its sign at each timepoint. First a bit up, then a bit down.

Gear2 method

In the Gear2 method, the current is assumed to be a polynomial fit in the timestep. This requires the simulation history to determine the curve. Because of this, the simulation takes about 5% longer to run compared to trapezoidal. Because the current is more accurately represented, the method is a bit more accurate than trapezoidal. It also slightly numerically damps any ringing that the circuit might produce. This effect disappears completely when the timesteps are small, which is usually accomplished by setting `reltol` to `1e-4` or smaller. For very high Q circuits, small numerical integration errors cause large errors in the voltage in the resonator. For that reason, harmonic balance is usually preferable to shooting for high Q circuits. If you must simulate a high Q circuit using the transient analysis, always use the `traponly` numerical integration method.

Controlling Accuracy

The simulator works by setting a limit for the curvature. This limit is controllable by setting `lteratio` and/or `reltol`. The product of `reltol` times `lteratio` sets the actual limit. Increasing `lteratio` sets a higher amount of curvature and leads to more errors when

evaluating the numerical integration. The output with `lteratio` set to 10 times the default, is shown below.

The waveshape is substantially similar. In this case, it is OK to allow more numerical integration errors. If the details of a small portion of the waveform need to be accurate, zoom in to the time scale that is needed, as shown below.

Note the obvious piecewise linear behavior in the waveform. This is an indicator that at this time scale, the error is unacceptable. `Lteratio` and `reltol` were reduced for the following waveform.

The waveshape is substantially changed. This is because the numerical errors were significant at the time/voltage scale that we needed for the measurement.

There is still a bit of piecewise linear behavior. Reducing `reltol` and `lteratio` further will only be slightly more accurate and the runtime will be much longer. In simulation, a trade-off of accuracy versus runtime needs to be made using `reltol`, `vabstol`, and `lteratio` to get the accuracy that is needed without excessive runtimes.

Shooting PSS

Overview

Shooting is a way of calculating the steady-state waveforms of a periodic system. Any waveshape is allowed as an input with the limitation that the waveform must be periodic. First, a transient is run for the duration set by the additional time for stabilization (`tstab`) option. Next, one period of the pss beat frequency is simulated while still using the normal transient algorithm. Then, the transient continues for one period of the waveform with the addition of saving all the solution matrices at each timestep. This is called the shooting window.

At the end of simulating this period of the input, the beginning and ending states are compared. If all the voltages and currents are equal, the steady-state solution has been achieved, and the simulation stops. In general, because the transient analysis starts from a settled DC solution at time zero, there is still settling behavior going on at the end of the first time through the shooting window. This causes the solutions to be different at the beginning and ending of the shooting window. The simulator now looks through all the solution matrices which contain information about the behavior of the circuit and the nonlinear resistors, capacitors, and inductors in the circuit, Using this information, an estimate is made of the final settled voltage for every node in the circuit. This is brought back as the starting point for the shooting window, and one period is run again. At the end, the voltages and currents are again compared to see if they are equal. This process continues until the beginning and ending states are equal. This process is called the shooting Newton method.

Graphically, this is shown below.

The estimate for every node is brought back as the starting point, and the period just after 15 nsec is simulated again. This continues until the beginning and ending voltages and currents almost match.

Note that pss shooting is really an extension to the transient analysis. All the concepts and options that were discussed before in the transient review section apply to shooting pss as well.

Oscillators

The algorithm is slightly different for oscillators. First, the tstab interval is simulated. Next, a single period of the estimated frequency from the *Choosing Analyses* form is simulated. So

far, this is exactly the same as driven pss. To achieve reliable convergence, the oscillator must reach near the steady-state behavior during this phase.

Because the oscillator may also have a frequency divider at the output, additional four periods of the estimated frequency in the pss *Choosing Analyses* form are simulated. During this time, Spectre looks for a frequency divided output.

Next, one period of the lowest frequency in the circuit is determined. If the oscillator does not have a frequency divider, then one period of the oscillator output is solved for. If the oscillator has a divider, one period of the divided down frequency is solved for.

Once one period is determined, then the steady-state waveforms for all the nodes and the period are solved for so that the beginning and ending voltages and currents almost match, and the period is stable for at least two iterations. Because the period is an additional variable to be solved for compared to a driven circuit where the period is known, oscillators typically take more iterations to converge to the solution.

Delay Time

All the signal sources have a property called delay time. During the delay time, the source stays at a constant value, and then the waveform starts. The system does not become periodic until the longest delay time has elapsed. Delay times longer than one period of the output waveform of a source should be avoided. The beginning of the tstab interval is the longest delay time in the sources of the circuit.

Piecewise Linear Sources for Power Supplies

In some cases, nonperiodic piecewise linear sources are used. This is one way to start an oscillator by giving the power supply a small bump just after the transient analysis starts. In this case, set the first point at zero seconds and 80% to 90% of the full supply voltage, and specify a second point about half period of the operating frequency at the full supply voltage. Do not specify a point at one second (or some other fairly large time) and the supply voltage. Spectre will maintain the voltage of a source if the simulation time exceeds the last point from a piecewise linear source. If there is a one second point, then something that is usually at a high frequency has to simulate for one second before the circuit becomes periodic. Tstab would start at the last point in the piecewise linear waveform.

Tstab

In the pss Choosing Analyses form, there is a property called *Additional Time for Stabilization (tstab)*. In this manual, this will be called tstab. The default for tstab is zero, but

if there is a value defined for the parameter, then a normal transient will be run to a stop time equal to tstab. For driven circuits, it is seldom necessary to set tstab, but oscillators usually require it to be set.

Beat Frequency

The beat frequency is the periodicity of the system. If a single input frequency is applied, it is the frequency of the input signal. For frequencies that have inconvenient frequencies, but easy periods, a selection is available to set the period instead of the frequency.

If there are two input frequencies, the highest common multiple frequency must be set as the beat frequency. The limitation is that an integer number of cycles must be simulated for all the input frequencies. This is required because pss checks the differences between the currents and voltages at the beginning and end of the shooting window and this difference must be zero. This forces an integer number of cycles for all the sources in the circuit to be simulated.

As an example, if a 1 GHz frequency is applied to the system, the beat frequency (periodicity) of the system is 1 GHz. If 1 GHz and 1.1 GHz frequencies are applied, the beat frequency is 100 MHz. In this system, 10 cycles of 1GHz and 11 cycles of 1.1 GHz need to be simulated to get the steady-state waveform. If 1 GHz and 1.001GHz need to be simulated, the beat frequency is 1 MHz. This is not a practical pss simulation because 1000 cycles of 1 GHz and 1001 cycles of 1.001 GHz need to be simulated, and at least 3003 harmonics need to be calculated to measure the third harmonic of 1.001 GHz. The time required for the simulation and the memory requirement is excessive for this case. In this case, using qpss or hb is much faster and requires much less memory.

For oscillators, an estimate of the oscillator frequency is required in the *Beat Frequency* field. If the linear oscillator initial condition is used, the frequency range needs to be from 0.5 to 1.5 times the actual oscillation frequency. If the default oscillator initial condition is used, then the frequency must be between 0.25 and four times the actual frequency of oscillation.

Number of Harmonics

In the case of pss shooting, this is a post-processing specification that might have an effect on the minimum number of timepoints in the solution. This parameter specifies the number of harmonics to calculate in the Fourier transform that is run after the time-domain simulation to calculate the harmonics of the large-signal response. The Fourier transform is not an FFT that requires exactly spaced samples of the waveform. Rather, it is Fourier integral-based, which uses the timepoints that were actually calculated by the pss analysis as the input. Since the waveform that the Fourier-integral based algorithm sees are the actual timepoints, all the harmonics specified are accurate, even if you only specify one or two harmonics. Zero (0) is

a perfectly legal value for the number of harmonics, and it just means that no Fourier transform will be run after the time-domain solution is calculated.

If the number of harmonics is raised above 10, then more timepoints will be forced in the pss solution in order to keep the accuracy of the frequency-domain results very high. 20 timepoints will be run in the period of the highest harmonic. Therefore, if maximum harmonics is set to 20, a minimum of 400 timepoints are forced in the pss analysis. If a large number of harmonics is specified, runtime and memory requirements get large. Unless you really need a lot of harmonics, do not set a large number.

Accuracy Defaults (errpreset)

Errpreset sets the default values for six options in order to tune the simulator for your needs. Use `conservative` if you need high accuracy. `Moderate` is a general-purpose setting, and should be used for most simulations. `Liberal` is not recommended.

The following chart summarizes the options and the values for driven circuits.

errpreset	Reltol[1]	Relref	Method	Lteratio[2]	Steadyratio	Maxstep
Liberal	1e-3	Sigglobal	Traponly	3.5	0.001	Period/50
Moderate	1e-3	Alllocal	Gear2only	3.5	0.001	Period/200
Conservative	1e-4	Alllocal	Gear2only	10 or 3.5	0.01	Period/200

1. Reltol: This value is used as a maximum value in the shooting window. The value used in the tstab interval is set by the option reltol.
2. Lteratio: For conservative, 10 is used in the shooting window unless the option reltol is set to 2.85e-4 or smaller. If 2.85e-4 or smaller is used, lteratio defaults to 3.5. The value used for lteratio in the tstab interval is the same as that used in the shooting window.

For example, if `errpreset` is set to `moderate` and `reltol` is set to `3e-4`, `reltol` during the tstab interval is `3e-4`, and `reltol` in the shooting window is also `3e-4` because it is smaller than `1e-3`, which is the maximum value that can be used in the shooting window. `Lteratio` is 3.5 for the tstab interval and for the shooting window.

If `errpreset` is `conservative` and `reltol` is set to `3e-4`, then `reltol` in the tstab interval is `3e-4`, and in the shooting window it is 1`e-4`. This is because the maximum value for `reltol` in the shooting window for `conservative` is `1e-4`. `Lteratio` is `10` for both the tstab interval and the shooting window.

For oscillators, the defaults are slightly different.

errpreset	Reltol[1]	Relref	Method	Lteratio[2]	Steadyratio	Maxstep
Liberal	1e-3	Sigglobal	Traponly	3.5	0.001	Period/50
Moderate	1e-4	Alllocal	Gear2only	3.5	0.001	Period/200
Conservative	1e-5	Alllocal	Gear2only	10	0.1	Period/200

1. Reltol: This value is used as a maximum value in the shooting window. The value used in the tstab interval is set by the option reltol.
2. Lteratio: For conservative, 10 is used in the shooting window unless the option reltol is set to 2.85e-4 or smaller. If 2.85e-4 or smaller is used, lteratio defaults to 3.5. The value used in the tstab interval is the same as that used in the shooting window.

PSS Shooting Convergence

As discussed earlier, the waveform is iteratively solved for during the shooting window. This means that exact solutions are never obtained because there is another iteration that will produce a closer answer. The iterations must stop when the solution is accurate enough. This is set using the convergence options `reltol`, `vabstol`, `iabstol`, `lteratio`, `errpreset`, and an option called `steadyratio`. The iterations stop when all the voltages in all the nodes and all the currents in all the branches match at the beginning and end within the following tolerance:

```
(Convergence Criteria) * Lteratio * Steadyratio
```

The convergence criteria depend on the setting of `reltol`, `vabstol`, `iabstol`, `errpreset`, and the setting of `relref`. `Lteratio` defaults to `3.5` or `10`, depending on the setting of `reltol` and `errpreset`.

`Steadyratio` depends on the setting of `errpreset`. For more details regarding this option see Steadyratio (Shooting Only) on page 564.

ADE Implementation and Numerical Noise Floor

Next, we will discuss how SpectreRF is implemented into ADE and also provide information about how the simulation accuracy is affected by setting a number of simulator options.

A transistor-level LNA is shown below.

The input frequencies for the transistor-level LNA are set to 2.4 GHz and 2.5 GHz, as shown in the ADE window below.

Note that this setup is approaching the limit of what pss can do because 25 periods of the higher input frequency need to be simulated. If the frequencies were closer together, qpss or hb would be much more efficient.

The pss *Choosing Analyses* form is shown below.

All the sources in the circuit are read in to the *Fundamental Tones* table at the top of the *Choosing Analyses* form. When the *Auto Calculate* option is selected, ADE calculates the periodicity of the circuit and displays it in the grayed-out *Beat Frequency* field.

Note: When *Auto Calculate* is selected, manual editing of the *Beat Frequency* field is not allowed.

Accuracy and Settings and Trade-Offs

A comparison is made for several settings. Moderate and conservative settings are used.

After that, values for *reltol* and *vabstol* are specified, as shown below in the *Simulator Options* form. This form can be opened by selecting *Simulation - Options - Analog in the ADE*.

First, `reltol=1e-5` and `vabstol=3e-8` are used, and then `reltol=1e-6` and `vabstol=1e-8` are used. Finally, a lot of timepoints are forced by setting the maximum harmonics to 1000. For this test, `reltol` and `vabstol` are kept at `1e-6` and `1e-8`.

The spectral result from all the simulations is shown below.

A marker is placed at 1.1 GHz in order to measure the numerical noise floor of the simulation.

The top is with `errpreset` set to `moderate`.

The second is `conservative`.

The third is `reltol=1e-5` and `vabstol=3e-8`.

The fourth is `reltol=1e-6` and `vabstol=1e-8`.

The final one is with 1000 harmonics set. The display is scaled to show only the first 30 harmonics.

Note that the numerical noise floor drops with the steps that are taken to increase accuracy. The runtime and memory requirement also increases for each successive level of accuracy.

This is a good general series of steps to improve the accuracy of a shooting simulation. The basic process is to set up and run the simulation, make the accuracy better, and run the simulation again. If the answer does not change, then the original setting is good enough. If the answer changes, increase the accuracy again. Continue until you find the setting of the options that cause the answer to be accurate without excessive runtime.

Note that there is a practical minimum numerical noise floor at about -135dBm. Although it might be possible to lower the noise floor a bit more by taking even more timepoints in the simulation, the time and memory requirements usually are not worth the lower noise floor. Still, -135dBm is a pretty good numerical noise floor of about 3e-8 volts RMS.

Additionally, you might try using the *highorder* option to improve accuracy. This is shown in the pss options section. This option requires significantly longer to run but produces a more accurate result.

Which Engine Should be Selected?

Conceptually, shooting and harmonic balance produce the same answer as long as enough harmonics and/or a high enough oversample factor are used in hb and the accuracy of shooting is controlled well enough for your requirement. In general, if the accurate waveform is desired for a system with square waves in the solution, then shooting should be selected. If the system is very nonlinear like a sample-and-hold or a switched-capacitor filter, shooting should always be used. If the waveforms are nearly sinusoidal, then harmonic balance should be selected. If it is something in between, then try both engines and use the one that produces the fastest runtime. Note that even for square wave systems, harmonic balance can be used effectively as long as the time-domain waveform is not desired, and oversample factor is set to at least four. For more details on harmonic balance, see Chapter 3, “Frequency Domain Analyses: Harmonic Balance”.

Important Outputs in the Spectre.out File

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/D
File  Help                                                    cadence

Circuit inventory:
              nodes 250
               balun 2
            bsim3v3 68
          capacitor 58
           inductor 27
    mutual_inductor 12
            phy_res 30
               port 2
           resistor 162
            vsource 3

Notice from spectre during initial setup.
    APS Enabled.

Time for parsing: CPU = 0 s, elapsed = 615.207 ms.
Time accumulated: CPU = 209.967 ms, elapsed = 1.10148 s.
Peak resident memory used = 41.3 Mbytes.

*****************************************************
Periodic Steady-State Analysis `pss': fund = 100 MHz
*****************************************************
Trying `homotopy = gmin' for initial conditions.

Notice from spectre during IC analysis, during periodic steady state a
    GminDC = 1 pS is large enough to noticeably affect the DC solution
        dV(I0.I29.net023) = -50.214 mV
        Use the `gmin_check' option to eliminate or expand this report

============================
`pss': time = (0 s -> 10 ns)
============================
Important parameter values in tstab integration:
    start = 0 s
    outputstart = 0 s
    stop = 10 ns
    period = 10 ns
    step = 10 ps
    maxstep = 400 ps
    ic = all
    useprevic = no
    skipdc = no
    reltol = 1e-06
    abstol(V) = 10 nV
    abstol(I) = 1 pA
    temp = 27 C
    tnom = 27 C
    tempeffects = all
    method = traponly
    lteratio = 3.5
    relref = sigglobal
    cmin = 0 F
    gmin = 1 pS
    rabsshort = 1 mOhm

    pss: time = 252.7 ps    (2.53 %), step = 3.357 ps    (33.6 m%)
    pss: time = 750.9 ps    (7.51 %), step = 2.361 ps    (23.6 m%)
    pss: time = 1.25 ns     (12.5 %), step = 1.821 ps    (18.2 m%)
    pss: time = 1.751 ns    (17.5 %), step = 2.058 ps    (20.6 m%)
    pss: time = 2.251 ns    (22.5 %), step = 1.943 ps    (19.4 m%)
28
```

The circuit inventory (highlighted in red above) lists the contents of the circuit.

If APS is used, it is noted in the section highlighted in green above.

The section highlighted in blue above contains the option settings for tstab simulation.

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/D
File  Help                                                    cadence
    pss: time = 9.252 ns    (92.5 %), step = 1.75 ps     (17.5 m%)
    pss: time = 9.751 ns    (97.5 %), step = 2.276 ps    (22.8 m%)
Conv norm = 6.48e+06, max dI(I0.I28.I10.1s1:1) = 18.1916 mA, took 2.41

    pss: time = 9.252 ns    (92.5 %), step = 1.75 ps     (17.5 m%)
    pss: time = 9.751 ns    (97.5 %), step = 2.276 ps    (22.8 m%)
Conv norm = 6.48e+06, max dI(I0.I28.I10.1s1:1) = 18.1916 mA, took 2.41

Important parameter values in pss iteration:
    start = 0 s
    outputstart = 0 s
    stop = 10 ns
    period = 10 ns
    steadyratio = 10e-03
    step = 10 ps
    maxstep = 50 ps
    ic = all
    useprevic = no
    skipdc = no
    reltol = 1e-06
    abstol(V) = 10 nV
    abstol(I) = 1 pA
    temp = 27 C
    tnom = 27 C
    tempeffects = all
    errpreset = conservative
    method = gear2only
    lteratio = 3.5
    relref = alllocal
    cmin = 0 F
    gmin = 1 pS
    rabsshort = 1 mOhm

============================
`pss': time = (0 s -> 10 ns)
============================
    pss: time = 250.5 ps     (2.5 %), step = 1.086 ps     (10.9 m%)
    pss: time = 750.8 ps    (7.51 %), step = 1.406 ps     (14.1 m%)
    pss: time = 1.25 ns     (12.5 %), step = 1.077 ps     (10.8 m%)
    pss: time = 1.751 ns    (17.5 %), step = 1.232 ps     (12.3 m%)
    pss: time = 2.25 ns     (22.5 %), step = 1.251 ps     (12.5 m%)
    pss: time = 2.75 ns     (27.5 %), step = 1.22 ps      (12.2 m%)
    pss: time = 3.251 ns    (32.5 %), step = 1.553 ps     (15.5 m%)
    pss: time = 3.75 ns     (37.5 %), step = 1.269 ps     (12.7 m%)
    pss: time = 4.251 ns    (42.5 %), step = 1.765 ps     (17.7 m%)
    pss: time = 4.751 ns    (47.5 %), step = 1.486 ps     (14.9 m%)
    pss: time = 5.252 ns    (52.5 %), step = 2.291 ps     (22.9 m%)
    pss: time = 5.75 ns     (57.5 %), step = 2.466 ps     (24.7 m%)
    pss: time = 6.253 ns    (62.5 %), step = 2.74 ps      (27.4 m%)
    pss: time = 6.751 ns    (67.5 %), step = 1.919 ps     (19.2 m%)
    pss: time = 7.251 ns    (72.5 %), step = 1.632 ps     (16.3 m%)
    pss: time = 7.751 ns    (77.5 %), step = 1.611 ps     (16.1 m%)
    pss: time = 8.25 ns     (82.5 %), step = 1.291 ps     (12.9 m%)
    pss: time = 8.751 ns    (87.5 %), step = 1.548 ps     (15.5 m%)
    pss: time = 9.25 ns     (92.5 %), step = 1.14 ps      (11.4 m%)
    pss: time = 9.751 ns    (97.5 %), step = 1.444 ps     (14.4 m%)
Conv norm = 3.34e+03, max dI(I0.I28.I6.1s2:1) = -9.73351 uA, took 5.7

============================
`pss': time = (0 s -> 10 ns)
============================
    pss: time = 250.5 ps     (2.5 %), step = 1.086 ps     (10.9 m%)
    pss: time = 750.8 ps    (7.51 %), step = 1.406 ps     (14.1 m%)
    pss: time = 1.25 ns     (12.5 %), step = 1.077 ps     (10.8 m%)
28
```

The section highlighted in red above lists the options for shooting iterations.

When the `Conv_Norm` number gets less than one, the iterations stop. In this case, the error is 3.34e-3 times the maximum allowed, and the largest mismatch is I0.I28.I6.Is2 terminal 1, and the mismatch is 9.73351 microamps (highlighted in green above).

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/D
File  Help                                                    cādence
    pss: time = 3.251 ns    (32.5 %), step = 1.553 ps    (15.5 m%)
    pss: time = 3.75 ns     (37.5 %), step = 1.269 ps    (12.7 m%)
    pss: time = 4.251 ns    (42.5 %), step = 1.765 ps    (17.7 m%)
    pss: time = 4.751 ns    (47.5 %), step = 1.486 ps    (14.9 m%)
    pss: time = 5.252 ns    (52.5 %), step = 2.291 ps    (22.9 m%)
    pss: time = 5.75 ns     (57.5 %), step = 2.466 ps    (24.7 m%)
    pss: time = 6.253 ns    (62.5 %), step = 2.74 ps     (27.4 m%)
    pss: time = 6.751 ns    (67.5 %), step = 1.919 ps    (19.2 m%)
    pss: time = 7.251 ns    (72.5 %), step = 1.632 ps    (16.3 m%)
    pss: time = 7.751 ns    (77.5 %), step = 1.611 ps    (16.1 m%)
    pss: time = 8.25 ns     (82.5 %), step = 1.291 ps    (12.9 m%)
    pss: time = 8.751 ns    (87.5 %), step = 1.548 ps    (15.5 m%)
    pss: time = 9.25 ns     (92.5 %), step = 1.14 ps     (11.4 m%)
    pss: time = 9.751 ns    (97.5 %), step = 1.444 ps    (14.4 m%)
Conv norm = 3.34e+03, max dI(I0.I28.I6.Is2:1) = -9.73351 uA, took 5.7

============================
`pss': time = (0 s -> 10 ns)
============================
    pss: time = 250.5 ps     (2.5 %), step = 1.086 ps    (10.9 m%)
    pss: time = 750.8 ps    (7.51 %), step = 1.406 ps    (14.1 m%)
    pss: time = 1.25 ns     (12.5 %), step = 1.077 ps    (10.8 m%)
    pss: time = 1.751 ns    (17.5 %), step = 1.232 ps    (12.3 m%)
    pss: time = 2.25 ns     (22.5 %), step = 1.251 ps    (12.5 m%)
    pss: time = 2.75 ns     (27.5 %), step = 1.22 ps     (12.2 m%)
    pss: time = 3.251 ns    (32.5 %), step = 1.553 ps    (15.5 m%)
    pss: time = 3.75 ns     (37.5 %), step = 1.269 ps    (12.7 m%)
    pss: time = 4.251 ns    (42.5 %), step = 1.765 ps    (17.7 m%)
    pss: time = 4.751 ns    (47.5 %), step = 1.486 ps    (14.9 m%)
    pss: time = 5.252 ns    (52.5 %), step = 2.291 ps    (22.9 m%)
    pss: time = 5.75 ns     (57.5 %), step = 2.466 ps    (24.7 m%)
    pss: time = 6.253 ns    (62.5 %), step = 2.74 ps     (27.4 m%)
    pss: time = 6.751 ns    (67.5 %), step = 1.919 ps    (19.2 m%)
    pss: time = 7.251 ns    (72.5 %), step = 1.632 ps    (16.3 m%)
    pss: time = 7.751 ns    (77.5 %), step = 1.611 ps    (16.1 m%)
    pss: time = 8.25 ns     (82.5 %), step = 1.291 ps    (12.9 m%)
    pss: time = 8.751 ns    (87.5 %), step = 1.548 ps    (15.5 m%)
    pss: time = 9.25 ns     (92.5 %), step = 1.14 ps     (11.4 m%)
    pss: time = 9.751 ns    (97.5 %), step = 1.444 ps    (14.4 m%)
Conv norm = 221e-03, max dV(I0.I31.net5) = -119.863 nV, took 5.15 s.

Number of accepted pss steps =              17796
Initial condition solution time: CPU = 49.993 ms, elapsed = 61.223 ms.
pss: The steady-state solution was achieved in 3 iterations.
Number of accepted pss steps =               6778
Total time required for pss analysis `pss': CPU = 13.8679 s, elapsed =
Time accumulated: CPU = 14.0779 s, elapsed = 16.1999 s.
Peak resident memory used = 184 Mbytes.

modelParameter: writing model parameter values to rawfile.
element: writing instance parameter values to rawfile.
outputParameter: writing output parameter values to rawfile.
designParamVals: writing netlist parameters to rawfile.
primitives: writing primitives to rawfile.
subckts: writing subcircuits to rawfile.

Aggregate audit (3:26:40 PM, Wed Aug 3, 2011):
Time used: CPU = 14.1 s, elapsed = 16.9 s, util. = 83.6%.
Time spent in licensing: elapsed = 524 ms.
Peak memory used = 184 Mbytes.
Simulation started at: 3:26:23 PM, Wed Aug 3, 2011, ended at: 3:26:40
spectre completes with 0 errors, 2 warnings, and 4 notices.

spectre completes with 0 errors, 2 warnings, and 4 notices.
28
```

When the Conv_Norm is less than one, the iterations stop (highlighted in red above).

The total number of timesteps in the solution is listed (highlighted in green above). The first listing is the number of timesteps in the tstab interval. The second listing is the number of timepoints in the shooting interval.

CPU time, elapsed (clock) time, and memory statistics are listed (highlighted in blue above). When APS is used with multi-threading, the CPU time is generally larger than the elapsed time. CPU time totals the usage of all CPUs.

Guidelines for Setting up Oscillators for Simulation

Beat frequency

Enter a frequency that is between 0.25 * actual frequency to 4 * actual frequency. As long as the specified frequency is within this range, the solution should be possible.

Tstab

Tstab is the time interval that is used to start the oscillator and allows the oscillations to build up to a near steady-state amplitude.

Different ways to start the oscillator

The first thing to do when simulating oscillators is to reliably start the oscillator. Different methods are used for different types of oscillators. These are:

- **Moderate Q Feedback Oscillators**: If the oscillator is a feedback oscillator, then the linear oscillator initial condition (oscic) is highly recommended. When this is selected, the beat frequency needs to be within 0.5 * actual frequency to 1.5 * actual frequency. This performs a variation of the linear stability analysis after the DC analysis for the first timepoint in the tstab interval. It provides a very good estimate of the oscillating frequency and amplitude at every node and this estimate is used at the beginning of the tstab interval. This allows the duration of the tstab period to be about 5 to 10 periods of the oscillator frequency for normal Q circuits. In many cases, the tstab time can be set to zero. Doing this, causes the pss analysis to start with the estimated frequency and amplitude from the linear oscic.

  Another way to start the oscillator is to use a piecewise linear source as the power supply. At time zero, the voltage is about 80% to 90% of nominal. The second timepoint should

be about one half the period of the oscillator and the full power supply voltage. The bump in voltage causes a voltage transient that starts things up. For differential oscillators, do this on one side of the differential system.

A third way is to add a current pulse to a point in the circuit that causes the oscillation to start using an ipulse component from *analogLib*. Specify a single pulse that is large enough to get things started. This is accomplished by adding an ipulse component from *analogLib* and specifying the zero value to zero, the one value to be about a milliamp, the rise and fall times to about one tenth of a period, and the pulse width to about three tenths of a period. Leave the period field blank to indicate that only a single pulse should be generated. The ipulse is an ideal open when the current is zero, so it does not load the circuit.

For all the different methods other than the linear oscic, tstab must be long enough for the oscillator to reach near steady-state operation. For most oscillators, this is about 100 periods of the oscillation frequency.

- **High Q Oscillators**: For high Q oscillators, harmonic balance is recommended. The reason for this is that in harmonic balance, the motional equivalent circuit (electrical representation of the crystal) goes directly into the frequency domain representation of the circuit. In shooting, there is always a small numerical integration error in the solution. For high Q circuits, this error causes uncertainty in the pss result that can cause the pss analysis to become inaccurate.
- **Ring Oscillators**: For ring oscillators, start the oscillation by specifying initial conditions at the output of one stage of the ring. Force the output of this stage to low or high with the initial conditions. This will force the oscillator into a known state at time zero, and then at the beginning of the transient, the node(s) will begin the switch to the other state, thereby starting the oscillator. Tstab should be set to about 1 or 2 periods.

For all the different methods other than the linear oscic, tstab must be long enough for the oscillator to reach near steady-state operation. For most oscillators, this is about 100 periods of the oscillation frequency. When using linear oscic, this can usually be shortened to about 10 periods or even zero.

As usual for any pss analysis, first the tstab interval is simulated. Then one period of the frequency supplied in the pss *Choosing Analyses* form is simulated. For oscillators, an additional four periods of the specified frequency in the pss setup is simulated. In the last interval, Spectre checks for possible divided-down frequencies. If these frequencies are found, Spectre adjusts the frequency/period to correspond to the new lower frequency and tries to solve for that lower frequency.

Saveinit- A way to save the startup waveform

For many oscillators, it is desirable to see the actual startup waveform in order to verify that the startup is robust. To see the startup waveform, select *yes* for the *Save Initial Transient Results (saveinit)* property in the *Choosing Analyses* form. This will write the initial waveforms to the output data for viewing.

Oscillator Node- Used to estimate the frequency for the first pss iteration

When linear oscic is used, the oscillator node must be a node that is inside the feedback system for the oscillator. This is used to estimate the initial oscillating frequency. If the oscillator is differential, specify a pair of nodes in a symmetrical location in the oscillator.

Regardless of the setting for oscic, the oscillator node is used at the end of the tstab interval to determine an initial estimate of the actual oscillation frequency.

If the amplitude on this node drops to less than 1mV peak, an error is displayed indicating that the amplitude of the oscillator node is too small to properly estimate the frequency, and the simulation stops. If the frequency gets to 1000 * frequency specified, an error is displayed indicating that the frequency is too high to be reliable.

ADE Implementation

Below is an example ADE setup for an oscillator circuit. Let us set up a PSS analysis, run the simulation, and plot the output waveform.

Consider an oscillator shown below.

1. Select *Analyses - Choose* to open the *Choosing Analyses* form.

2. Specify an estimate of the oscillation frequency (highlighted in red above). If you use *linear* oscic, the estimate should be between 0.5 * actual oscillating frequency to 1.5 * actual oscillating frequency. Otherwise, it can be between 0.25 and 4 times the actual operating frequency.
3. Select *Conservative* (recommended) for oscillators (highlighted in green above).
4. The oscillator node should be inside the feedback system if *linear* oscic is used (highlighted in blue above). If the *default* oscic is used, the node just needs a signal on it.
5. *Select linear* oscic (recommended) for feedback oscillators (highlighted in purple above).

   default oscic starts from the time zero timepoint in the tstab interval.

Note: For a driven circuit, the process is the same except that the oscillator node and the oscillator initial condition fields are not shown.

Spectre Output File (Oscillators)

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/D
File  Help                                                    cadence

*********************************************************
Trying `homotopy = gmin' for initial conditions.

Using linear IC
Linear IC: estimated frequency is 5.12282e+09 Hz

================================
`pss': time = (0 s -> 3.05082 ns)
================================
Important parameter values in tstab integration:
    start = 0 s
    outputstart = 0 s
    stop = 3.05082 ns
    period = 195.205 ps
    step = 2.27 ps
    maxstep = 8 ps
    ic = all
    useprevic = no
    skipdc = no
    reltol = 1e-03
    abstol(V) = 1 uV
    abstol(I) = 1 pA
    temp = 27 C
    tnom = 27 C
    tempeffects = all
    method = traponly
    lteratio = 3.5
    relref = sigglobal
    cmin = 0 F
    gmin = 1 pS

    pss: time = 77.08 ps    (2.53 %), step = 2.6 ps      (85.2 m%)
    pss: time = 234.9 ps     (7.7 %), step = 8 ps         (262 m%)
    pss: time = 381.7 ps    (12.5 %), step = 7.408 ps     (243 m%)
31  HelpAction
```

The estimate of the oscillation frequency from the linear oscic (highlighted in red above) is shown first.

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/D
File  Help                                                    cadence

    pss: time = 2.364 ns     (77.5 %), step = 780.8 fs     (25.6 m%)
    pss: time = 2.518 ns     (82.5 %), step = 780.8 fs     (25.6 m%)
    pss: time = 2.67 ns      (87.5 %), step = 780.8 fs     (25.6 m%)
    pss: time = 2.822 ns     (92.5 %), step = 780.8 fs     (25.6 m%)
    pss: time = 2.975 ns     (97.5 %), step = 780.8 fs     (25.6 m%)
The Estimated oscillating frequency from Tstab Tran is = 5.16414 GHz .
Conv norm = 1.28e+03, max dI(I12.I2.I0.1s1:1) = 29.8684 mA, took 720 m

Important parameter values in pss iteration:
    start = 3.05082 ns
    outputstart = 0 s
    stop = 3.24446 ns
    period = 193.643 ps
    steadyratio = 100e-03
    step = 2.27 ps
    maxstep = 484.108 fs
    ic = all
    useprevic = no
    skipdc = no
    reltol = 10e-06
    abstol(V) = 1 uV
    abstol(I) = 1 pA
    temp = 27 C
    tnom = 27 C
    tempeffects = all
    errpreset = conservative
    method = gear2only
    lteratio = 10
    relref = alllocal
    cmin = 0 F
    gmin = 1 pS

=========================================
31  HelpAction
```

The frequency from the end of the tstab interval (highlighted in red above) is displayed next.

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/D
File  Help                                                    cadence

    pss: time = 3.182 ns     (67.6 %), step = 484.1 fs     (249 m%)
    pss: time = 3.192 ns     (72.6 %), step = 484.1 fs     (249 m%)
    pss: time = 3.201 ns     (77.6 %), step = 484.1 fs     (249 m%)
    pss: time = 3.211 ns     (82.6 %), step = 484.1 fs     (249 m%)
    pss: time = 3.221 ns     (87.5 %), step = 484.1 fs     (249 m%)
    pss: time = 3.23 ns      (92.5 %), step = 484.1 fs     (249 m%)
    pss: time = 3.24 ns      (97.5 %), step = 484.1 fs     (249 m%)
Conv norm = 142e-03, max dI(V3:p) = 31.9656 nA, took 170 ms.

Fundamental frequency is 5.14989 GHz.
Number of accepted pss steps =              2586
Initial condition solution time: CPU = 19.997 ms, elapsed = 24.801 ms.
pss: The steady-state solution was achieved in 4 iterations.
Number of accepted pss steps =              404
Total time required for pss analysis `pss': CPU = 1.11983 s, elapsed =
Time accumulated: CPU = 1.34979 s, elapsed = 5.35451 s.
Peak resident memory used = 49.6 Mbytes.

modelParameter: writing model parameter values to rawfile.
element: writing instance parameter values to rawfile.
outputParameter: writing output parameter values to rawfile.
designParamVals: writing netlist parameters to rawfile.
primitives: writing primitives to rawfile.
subckts: writing subcircuits to rawfile.

Aggregate audit (1:50:55 PM, Thur Aug 4, 2011):
Time used: CPU = 1.39 s, elapsed = 6.14 s, util. = 22.6%.
Time spent in licensing: elapsed = 269 ms.
Peak memory used = 49.7 Mbytes.
Simulation started at: 1:50:49 PM, Thur Aug 4, 2011, ended at: 1:50:55
spectre completes with 0 errors, 4 warnings, and 0 notices.

31  HelpAction
```

The solution frequency and the number of timesteps (highlighted in red above) are shown. The first listing of *accepted pss timesteps* is the number of timesteps in the tstab interval. The second number is the number in the shooting interval.

PSS Options

The options are divided into five tabs: *Convergence*, *Accuracy*, *Output*, *Reuse*, and *Misc* (miscellaneous).

The options are shown in the order that they occur in the options form in ADE.

Convergence tab

Ic (Shooting and Harmonic Balance)

Initial conditions can be specified graphically by selecting *Simulation - Convergence Aids - Initial Condition* in the ADE. Initial conditions can also be specified from a file using the *readic* property. Capacitors and inductors have the initial condition properties in the property list for the component. For capacitors, this is an initial voltage across the capacitor, and for inductors, it is an initial current in the inductor. The default is to observe all the initial conditions in the DC analysis that is used as the time-zero timepoint. The initial conditions force a voltage or current to be present in the time-zero solution. The initial conditions are released for the rest of the transient simulation in the tstab interval. The *ic* option controls the initial conditions that should be observed in the time-zero timepoint. *all* is the default. *dev* means that only the initial conditions on capacitors and inductors are observed. *node* means

that only the initial conditions on a node are observed. *dc* means that no initial conditions are observed. The example below shows the ic option set to *node*.

Skipdc (Shooting and Harmonic Balance)

In some cases, the time-zero timepoint DC analysis does not converge. Instead of stopping the simulation, *skipdc* allows the tstab simulation to continue using an assumed solution for the time-zero timepoint. The default is *no* and a DC analysis is run to get the initial timepoint. *yes* means skip the DC solution, and proceed directly to the tstab simulation. All the nodes with initial conditions specified start at the initial condition value. Nodes with batteries start at the battery voltage. Nodes with no initial conditions start at zero volts. For `skipdc=yes`, the signal sources start as specified immediately in the tstab simulation. *sigrampup* uses the same assumptions for the starting voltages as *yes* does, but the start time is set to negative one tenth of the stop time for the tstab interval. At this time, the signal source time-varying part starts at zero and linearly ramps up to the full value at time = zero. After time = zero, the sources have the full amplitude time-varying part. The example below shows the *skipdc* option set to *sigrampup*.

Readic (Shooting and Harmonic Balance)

This specifies an ascii file that contains two columns to be read as initial conditions. The first column is the node name. The second column is the voltage value. If the entry does not start with / (slash), the entry is located in the netlist directory. To find the netlist directory, select *Setup- Simulator/Directory/Host* in ADE. Look in the *Project Directory* field for the location of the simulation directory. Navigate to that directory and then to the `<Circuit Name>/spectre/<schematic or config>/netlist` directory. You can also click (...) and browse to the directory. An example is shown below.

Readns (Shooting and Harmonic Balance)

This specifies an ascii file with the same format as an ic file that is used as nodesets for the time-zero DC solution. Nodesets do not force a voltage to be held for the time-zero solution. Instead, they are a way of speeding up the time-zero calculation. As a suggestion, set the

write option and the *readns* option to the same filename. The *write* option writes the time-zero solution to a file. When this is used as a starting point, many fewer iterations are needed for the time-zero point to converge. An example is shown below.

Cmin (Shooting and Harmonic Balance)

cmin can be used to improve convergence in the tstab and shooting intervals for shooting and for the tstab interval for harmonic balance. If a value is set for *cmin*, a capacitor with this value is added to every node with the other terminal of the capacitor connected to the global ground node. If a 10f to 50f capacitor is added, this prevents instantaneous changes from occurring from timepoint to timepoint, thus improving the convergence at the cost of adding non-physical capacitors to the circuit. An example with 10 femtoFarads is shown below. Note that if 10 is entered, a 10 Farad capacitor is added from every node to ground. Remember the multiplier in the entry. An example is shown below.

Maxiters (Shooting and Harmonic Balance)

The transient analysis for tstab (for both shooting and hb) and shooting interval iterate to a solution at each timepoint. *maxiters* specifies the maximum number of iterations before the timestep is cut for another try at convergence. In some cases, model parameters can cause discontinuities in the device current or capacitance. If this occurs, the change in the circuit condition can be large enough to require more than the five iterations that are the default. Specifying *maxiters* to the 40 to 100 range usually allows the simulator to converge in spite of the discontinuity. An example with *maxiters* set to 40 is shown below.

Accuracy Tab

The *Accuracy* tab is split into two categories: INTEGRATION METHOD PARAMETERS and ACCURACY PARAMETERS. For each parameter, the most commonly used settings are discussed.

Periodic Steady State Options
Conver...
Accuracy
Output
Reuse
Misc
INTEGRATION METHOD PARAMETERS
method
euler
trap
traponly
gear2
gear2only
tstabmethod
euler
trap
traponly
gear2
gear2only
ACCURACY PARAMETERS
relref
pointlocal
alllocal
sigglobal
allglobal
lteratio
steadyratio
itres
lnsolver
gmres
qmr
bicgstab
resgmres
resgmrescycle
instant
short
long
inexactNewton
yes
no
maxacfreq
maxperiods
finitediff
yes
refine
no
highorder
yes
no
psaratio
maxorder
fullpssvec
yes
no
fdharms
OK
Cancel
Defaults
Apply
Help

Method (Shooting only)

This option controls the integration method for the shooting window. Note that harmonic balance does not have the *method* option because the solver works in the frequency domain. The default value for the shooting window is determined by the setting of *errpreset*. For *moderate* and *conservative*, the default is *gear2only*. For *liberal* it is *traponly*. Generally *gear2only* is preferred because of the absence of trapezoidal ringing inherent in the *trap* (trapezoidal) method. Note that *liberal* is not recommended for RF simulation.

In the case where you are simulating a crystal oscillator with shooting, the method option should be set to *traponly*. The ringing that is inherent in the trapezoidal method is present and may cause convergence failure because of the unpredictable amplitude of the ringing, which depends on the accumulated total numerical integration error. The benefit of having a method that neither damps nor emphasizes the oscillations is required when simulating high Q oscillators. Even though `trap` neither emphasizes nor damps the oscillations, the small numerical integration errors caused by the finite timestep can still cause some inaccuracy in the simulation result. Using *gear2only* is not recommended even though convergence is more likely because the numerical damping inherent in the *gear2only* method causes unacceptable damping in the resonator, which causes smaller than real amplitude inside the resonator. Because of this, the overall solution will result in an error and can cause inaccurate measurement of the phase noise.

Tstabmethod (Shooting and Harmonic Balance)

This option controls the integration method in the tstab interval. The default value is determined by the setting of *errpreset* and whether the circuit is driven or is an oscillator. For *moderate* and *conservative* with a driven circuit, the default is *gear2only*. For *liberal* it is *traponly*. Generally *gear2only* is preferred because of the absence of trapezoidal ringing inherent in the *trap* (trapezoidal) method. The default for *tstabmethod* is *traponly* for oscillators. *traponly* is used for oscillators because with the defaults, the tstab interval uses relatively loose convergence options in order to speed up the simulation. This causes relatively long timesteps which for the *gear2* method would cause noticeable numerical damping. *traponly* does not numerically damp, therefore, it provides a better estimate of the oscillating frequency and amplitude. An example with *tstabmethod* set to *gear2only* is shown below.

Relref (Shooting (tstab and shooting window) and Harmonic Balance (tstab only))

In the order of most to least accurate, the *relref* settings are *pointlocal*, *alllocal*, *sigglobal*, *allglobal*. The default for *moderate* and *conservative* is *alllocal* which is preferred. *relref* is used in the transient analysis in the tstab interval for both shooting and harmonic balance and in the shooting window for shooting. *relref* is used in hb only for the tstab interval because the solver works in the frequency domain. In some cases, the timestep can collapse to near zero. The symptom is that in the Spectre output window, the percent done number remains the same for many reporting intervals. The solution to this is to set *relref* to *sigglobal*, which is slightly less accurate. In the Fourier transform, the noise floor usually degrades by 3 to 6 dB. Another possibility is to use the *lteminstep* option as described in the *AdditionalParams* option of the *Misc* tab.

Refer to Relref on page 521 for a detailed discussion on *relref*.

An example with *relref* set to *sigglobal* is shown below.

Lteratio Shooting ((tstab and shooting window) and Harmonic Balance (tstab only))

lteratio is a multiplier for the allowable numerical integration error in the tstab interval and the shooting window. The default is 3.5 or 10 depending on the setting of *reltol* and *errpreset*. See Controlling Accuracy on page 529 for details. *lteratio* cannot be set smaller than 1.0 and is normally between 3.5 and 100. In some cases, the timestep collapses to near zero and setting *relref* to *sigglobal* does not fix the problem. In this case, there might be a model discontinuity. To test this, disable numerical integration timestep control by setting *lteratio* to 1e9. Note that this is an extreme measure not to be used under normal simulations. If the timestep still collapses, there is a discontinuity in one of the models in the circuit. This could be a device model or a Verilog-A model. If it is necessary to set *lteratio* very large to get the simulation to complete, you must also set *maxstep* small enough to preserve the accuracy of the simulation. An example with *lteratio* set to the smallest value of *1.0* is shown below. Note that *lteminstep* might also be used to treat this condition. See the AdditionalParams (Shooting and Harmonic Balance) on page 599 section in the *Misc* tab on how to use this option.

Steadyratio (Shooting Only)

The maximum mismatch between the beginning and ending values for the voltages and currents in the shooting window is determined by the *convergence criteria* * *Iteratio* * *steadyratio*. In some cases, with oscillators, *conv_norm* drops to a small value and then bumps up and down. However, it never gets less than one, therefore, the convergence fails. In this case, setting *steadyratio* to 1.0 will cause convergence if the *conv_norm* number gets to 10 or less. While larger settings of *steadyratio* may cause the oscillator to converge, more mismatch from beginning to end of the shooting window is allowed, which allows much more startup behavior to be in the solution. Setting *steadyratio* larger than one is not recommended. An example with *steadyratio* set to 1.0 is shown below.

Steadyratio (Harmonic Balance Only)

steadyratio in harmonic balance is used as a multiplier for the maximum delta that is allows from iteration to iteration. The default is 1.0. In some cases, *resd_norm* falls below one which indicates that the KCL check at each node is passing, but the Delta Norm is greater than one, which indicates that the motion from iteration-to-iteration is larger than the maximum allowed. Only in this case should *steadyratio* be set larger than one. Set it just larger than the value of the *resd_norm*. In this case, the smaller harmonics are changing relatively a lot, but because they are small, they do not affect the accuracy of the solution.

Itres (Shooting Only)

Use the default for this option. *itres* controls the precision (number of digits solved for) in the first iteration. The default is 1e-4 which causes 4 digits of resolution in the first iteration. Subsequent iterations are solved with more precision.

Itres (Harmonic Balance Only)

The default for *itres* for harmonic balance is 0.9, which allows up to 90% difference from the exact solution on the first iteration. Although this seems large, solving exactly for the first iteration can frequently cause non-convergence. As the iterations progress, the difference is reduced so that at the solution, full accuracy is achieved. In the case of very linear systems like an amplifier, tightening *itres* to the 1e-2 to 1e-4 range can reduce the number of frequency domain iterations and runtime. Also, when transient assist is used, the fft provides an excellent starting point for the first iteration. Therefore, setting *itres* to the 1e-2 to 1e-4 range can also reduce the number of frequency domain iterations and the runtime.

Lnsolver (Shooting Only)

Leave this option at the default. At the end of each pss iteration, a large matrix is formed and solved to calculate the new starting value in the shooting window. This matrix is solved using an iterative solver. Several different algorithms are provided for the iterative solver. *gmres* is the default because the accuracy of each iteration inherently increases. Convergence is generally good as well. The other solvers may require less memory, but they are less robust for convergence and may suffer from false convergence.

You need to have a good understanding of the linear algebra theory to understand the differences between these methods.

Gmres is the Generalized Minimum RESidual method.

Qmr is the Quasi-Minimal Residual method.

Bicgstab is the STABilized BI-Conjugate Gradient method.

Resgmres is the REStarted Generalized Minimal Residual method.

Lnsolver (Harmonic Balance Only)

Leave this option at the default. The frequency domain matrices are calculated with an iterative solver. *lnsolver* controls the algorithm that is used for the iterative solver. *gmres* is the default, and is a good general-purpose solver because it usually has higher accuracy for each iteration. The other solvers may require less memory, but they are less robust for convergence and may suffer from false convergence.

You need to have a good understanding of the linear algebra theory to understand the differences between these methods.

Gmres is the Generalized Minimum RESidual method.

Qmr is the Quasi-Minimal Residual method.

Bicgstab is the STABilized BI-Conjugate Gradient method.

Resgmres is the REStarted Generalized Minimal Residual method.

Resgmrescycle (Shooting and Harmonic Balance)

Leave this option at the default. For the resgmres linear solver, there are several different options.

Inexact Newton (Shooting Only)

In the simulation time between iterations, an estimate is made for the starting point for the next iteration (shooting) or the harmonic content for the next iteration (hb). *inexactNewton* causes an inexact solution for the matrix that calculates the next starting point to be calculated. In some cases, solving the matrix exactly on the first iteration can cause the iterative solver to need many iterations to achieve convergence. Setting i*nexactNewton* causes a different series of iterations, which may speed up the simulation. The inexactness is reduced as the iterations progress. Try this option when the time spent between iterations is excessive.

Maxacfreq (Shooting Only)

If maximum sideband in *pac*, *pnoise*, *pxf*, or *psp* is set larger than 40, then the *pss* option *maxacfreq* needs to be set to the pss beat frequency times the largest maximum sideband number in the small-signal analyses. For example, if *pac* has 20 for maximum sidebands, *pnoise* has 70, *pxf* has 10, and *psp* has 50, then *maxacfreq* needs to be set to the pss frequency times 70.

Spectre may issue a warning at the beginning of the small-signal analysis like the following one from pnoise if *maxacfreq* is not set and more than 40 sidebands are specified. If you see this warning in a small-signal analysis, set the pss option *maxacfreq* to the pss beat

frequency times the number of sidebands used in the small-signal analysis. In the example below, *maxacfreq* should be set to 150 * pss frequency.

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/DVCO_RFX_tb1/spectre/schematic/psf/spectre.out
File  Help                                                                                                  cadence

*****************************************************************************
Periodic Noise Analysis `pnoise': freq = 5.14989 GHz + (1 kHz -> 50 MHz)
*****************************************************************************
Using the operating-point information generated by PSS analysis `pss'.

Warning from spectre during PNoise analysis `pnoise'.
    WARNING (SPCRTRF-15029): The following results might not be accurate due to undersampling of the actual frequencies above
        Lower side band of 150th harmonic for all stimulus frequencies.
        Lower side band of 149th harmonic for all stimulus frequencies.
        Lower side band of 148th harmonic for all stimulus frequencies.
        Lower side band of 147th harmonic for all stimulus frequencies.
        Lower side band of 146th harmonic for all stimulus frequencies.
        Further occurrences of this warning will be suppressed.

The estimated line width of the oscillator is 1.23919 kHz.
    pnoise: freq = 1.778 kHz   (5.26 %), step = 778.3 Hz     (5.26 %)
    pnoise: freq = 3.162 kHz   (10.5 %), step = 1.384 kHz    (5.26 %)
    pnoise: freq = 5.623 kHz   (15.8 %), step = 2.461 kHz    (5.26 %)
    pnoise: freq = 10 kHz      (21.1 %), step = 4.377 kHz    (5.26 %)
    pnoise: freq = 17.78 kHz   (26.3 %), step = 7.783 kHz    (5.26 %)
    pnoise: freq = 31.62 kHz   (31.6 %), step = 13.84 kHz    (5.26 %)
    pnoise: freq = 56.23 kHz   (36.8 %), step = 24.61 kHz    (5.26 %)
    pnoise: freq = 100 kHz     (42.1 %), step = 43.77 kHz    (5.26 %)
    pnoise: freq = 177.8 kHz   (47.4 %), step = 77.83 kHz    (5.26 %)
    pnoise: freq = 316.2 kHz   (52.6 %), step = 138.4 kHz    (5.26 %)
    pnoise: freq = 562.3 kHz   (57.9 %), step = 246.1 kHz    (5.26 %)
    pnoise: freq = 1 MHz       (63.2 %), step = 437.7 kHz    (5.26 %)
    pnoise: freq = 1.778 MHz   (68.4 %), step = 778.3 kHz    (5.26 %)
    pnoise: freq = 3.162 MHz   (73.7 %), step = 1.384 MHz    (5.26 %)
    pnoise: freq = 5.623 MHz   (78.9 %), step = 2.461 MHz    (5.26 %)
    pnoise: freq = 10 MHz      (84.2 %), step = 4.377 MHz    (5.26 %)
    pnoise: freq = 17.78 MHz   (89.5 %), step = 7.783 MHz    (5.26 %)
    pnoise: freq = 31.62 MHz   (94.7 %), step = 13.84 MHz    (5.26 %)
31  HelpAction
```

An example of *maxacfreq* set to 120 GHz is shown below.

```
maxacfreq    120G
```

Maxperiods (Shooting and Harmonic Balance)

Any iterative method may fail to converge, and therefore needs an iteration limit. Maxperiods sets the maximum number of times the waveform in the shooting window can be simulated or the number of hb iterations. The default is 20 times for a driven circuit, and 50 times for an oscillator in shooting, and 100 for hb. This is a minimum allowed value for this option. Setting a number less than this default will be ignored by Spectre. This should be set higher only after watching the conv_norm during the pss iterations. If the conv_norm is trending down with no sudden large increase, setting more iterations may allow convergence. If the conv_norm

trends down, and then gets large quickly, setting more iterations is very unlikely to help. An example with 100 is shown below.

Finitediff (Shooting Only)

Leave this option at the default. This is set automatically when *readpss* is set. It allows small changes in circuit values when the *readpss* option is used.

Highorder (Shooting Only)

The normal integration method in the shooting window is *gear2only* which is a second order integration method. *highorder* allows the integration method to have up to 16th order numerical integration, thus making the numerical noise floor of the simulation much smaller. When *highorder* is set to *yes*, a normal pss is run until convergence is achieved, and then that solution is refined using a higher order integration method. In practice this has value for very linear systems like LNAs, and has smaller value for more complicated blocks like mixers. When *highorder* is set to *yes*, there can be considerable time when the highorder algorithm is running, but there is no output to the spectre output window. This is normal for highorder. An example of `highorder=yes` is shown below.

Below is the runtime for `highorder = no`.

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/DLNA1_RFT_t
File  Help                                                              cadence

    pss: time = 8.254 ns    (82.5 %), step = 8.224 ps    (82.2 m%)
    pss: time = 8.759 ns    (87.6 %), step = 10 ps        (100 m%)
    pss: time = 9.254 ns    (92.5 %), step = 7.479 ps    (74.8 m%)
    pss: time = 9.756 ns    (97.6 %), step = 9.921 ps    (99.2 m%)
Conv norm = 150e-03, max dI(Vdd:p) = -51.2279 nA, took 920 ms.

Number of accepted pss steps =              2705
Initial condition solution time: CPU = 39.994 ms, elapsed = 69.9532 ms.
pss: The steady-state solution was achieved in 3 iterations.
Number of accepted pss steps =              1120
Total time required for pss analysis `pss': CPU = 2.27965 s, elapsed = 3.73421 s.
Time accumulated: CPU = 2.46962 s, elapsed = 4.54295 s.
Peak resident memory used = 69.9 Mbytes.

modelParameter: writing model parameter values to rawfile.
element: writing instance parameter values to rawfile.
outputParameter: writing output parameter values to rawfile.
designParamVals: writing netlist parameters to rawfile.
primitives: writing primitives to rawfile.
subckts: writing subcircuits to rawfile.

Aggregate audit (4:05:19 PM, Thur Aug 11, 2011):
Time used: CPU = 2.53 s, elapsed = 5.38 s, util. = 47%.
Time spent in licensing: elapsed = 657 ms, percentage of total = 12.2%.
Peak memory used = 70 Mbytes.
Simulation started at: 4:05:14 PM, Thur Aug 11, 2011, ended at: 4:05:19 PM, Thur A
spectre completes with 0 errors, 2 warnings, and 4 notices.

13
```

Below is the runtime and the report of the order that was used in the simulation when *highorder* was set to *yes*. The runtime is much longer.

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/DLNA1_RFT_tl
File  Help                                                              cadence

    pss: time = 9.254 ns     (92.5 %), step = 7.479 ps     (74.8 m%)
    pss: time = 9.756 ns     (97.6 %), step = 9.921 ps     (99.2 m%)
Conv norm = 150e-03, max dI(Vdd:p) = -51.2279 nA, took 930 ms.

Order 14 used in      72 subintervals.
Order 16 used in      80 subintervals.

Conv residual norm = 378.
Conv solution-change norm in fdtd = 685.

Conv residual norm = 73.1.
Conv solution-change norm in fdtd = 58.

Conv residual norm = 1.67.
Conv solution-change norm in fdtd = 1.21.

Conv residual norm = 862e-06.
Number of refinements using multi-interval Chebyshev polynomial spectral algorithm

MIC-PSA finite-difference refinement finished, took 14.84 s.

Number of accepted pss steps =              2288
Initial condition solution time: CPU = 59.992 ms, elapsed = 68.563 ms.
pss: The steady-state solution was achieved in 3 iterations.
Number of accepted pss steps =              2288
Total time required for pss analysis `pss': CPU = 19.1071 s, elapsed = 20.9154 s.
Time accumulated: CPU = 19.3071 s, elapsed = 21.7495 s.
Peak resident memory used = 116 Mbytes.

modelParameter: writing model parameter values to rawfile.
element: writing instance parameter values to rawfile.
outputParameter: writing output parameter values to rawfile.
designParamVals: writing netlist parameters to rawfile.
primitives: writing primitives to rawfile.
subckts: writing subcircuits to rawfile.

Aggregate audit (4:07:18 PM, Thur Aug 11, 2011):
Time used: CPU = 19.4 s, elapsed = 22.5 s, util. = 86.1%.
Time spent in licensing: elapsed = 733 ms.
Peak memory used = 116 Mbytes.
Simulation started at: 4:06:55 PM, Thur Aug 11, 2011, ended at: 4:07:18 PM, Thur A
spectre completes with 0 errors, 2 warnings, and 4 notices.

13
```

Below is a spectral comparison between standard pss and *highorder* set to *yes*. The convergence options are at the default and *conservative* is selected. The results are for an LNA. The numerical noise floor at 1GHz dropped from -80dBm to -145 dBm.

Below is a comparison of runtimes for an LNA-Mixer circuit. The first is with *highorder* set to *no*, and the second is with *highorder* set to *yes*.

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/RxChain_LNA_M
File  Help                                                                cādence
============================
    pss: time = 257.2 ps    (2.57 %), step = 8.915 ps    (89.2 m%)
    pss: time = 758.5 ps    (7.59 %), step = 8.91 ps     (89.1 m%)
    pss: time = 1.257 ns    (12.6 %), step = 8.945 ps    (89.5 m%)
    pss: time = 1.759 ns    (17.6 %), step = 8.896 ps      (89 m%)
    pss: time = 2.259 ns    (22.6 %), step = 9.125 ps    (91.3 m%)
    pss: time = 2.759 ns    (27.6 %), step = 9.019 ps    (90.2 m%)
    pss: time = 3.251 ns    (32.5 %), step = 7.015 ps    (70.1 m%)
    pss: time = 3.75 ns     (37.5 %), step = 6.897 ps      (69 m%)
    pss: time = 4.25 ns     (42.5 %), step = 6.741 ps    (67.4 m%)
    pss: time = 4.751 ns    (47.5 %), step = 7.253 ps    (72.5 m%)
    pss: time = 5.251 ns    (52.5 %), step = 7.024 ps    (70.2 m%)
    pss: time = 5.758 ns    (57.6 %), step = 9.141 ps    (91.4 m%)
    pss: time = 6.257 ns    (62.6 %), step = 9.016 ps    (90.2 m%)
    pss: time = 6.756 ns    (67.6 %), step = 8.929 ps    (89.3 m%)
    pss: time = 7.257 ns    (72.6 %), step = 8.934 ps    (89.3 m%)
    pss: time = 7.758 ns    (77.6 %), step = 9.101 ps      (91 m%)
    pss: time = 8.258 ns    (82.6 %), step = 9.017 ps    (90.2 m%)
    pss: time = 8.753 ns    (87.5 %), step = 8.113 ps    (81.1 m%)
    pss: time = 9.256 ns    (92.6 %), step = 8.771 ps    (87.7 m%)
    pss: time = 9.755 ns    (97.6 %), step = 8.54 ps     (85.4 m%)
Conv norm = 339e-03, max dV(I33.I1.I1.I2.net18) = 16.0942 uV, took 3.52 s.

Number of accepted pss steps =              7332
Initial condition solution time: CPU = 119.982 ms, elapsed = 132.995 ms.
pss: The steady-state solution was achieved in 4 iterations.
Number of accepted pss steps =              2113
Total time required for pss analysis `pss': CPU = 16.3675 s, elapsed = 18.2945 s.
Time accumulated: CPU = 16.6275 s, elapsed = 19.1967 s.
Peak resident memory used = 188 Mbytes.

modelParameter: writing model parameter values to rawfile.
element: writing instance parameter values to rawfile.
outputParameter: writing output parameter values to rawfile.
designParamVals: writing netlist parameters to rawfile.
primitives: writing primitives to rawfile.
subckts: writing subcircuits to rawfile.

Aggregate audit (3:47:47 PM, Thur Aug 11, 2011):
Time used: CPU = 16.7 s, elapsed = 20 s, util. = 83.7%.
Time spent in licensing: elapsed = 674 ms.
Peak memory used = 188 Mbytes.
Simulation started at: 3:47:27 PM, Thur Aug 11, 2011, ended at: 3:47:47 PM, Thur Au
spectre completes with 0 errors, 3 warnings, and 4 notices.
6  HelpAction
```

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/RxChain_LNA_M
File  Help                                                                cādence

    pss: time = 6.257 ns    (62.6 %), step = 9.016 ps    (90.2 m%)
    pss: time = 6.756 ns    (67.6 %), step = 8.929 ps    (89.3 m%)
    pss: time = 7.257 ns    (72.6 %), step = 8.934 ps    (89.3 m%)
    pss: time = 7.758 ns    (77.6 %), step = 9.101 ps      (91 m%)
    pss: time = 8.258 ns    (82.6 %), step = 9.017 ps    (90.2 m%)
    pss: time = 8.753 ns    (87.5 %), step = 8.113 ps    (81.1 m%)
    pss: time = 9.256 ns    (92.6 %), step = 8.771 ps    (87.7 m%)
    pss: time = 9.755 ns    (97.6 %), step = 8.54 ps     (85.4 m%)
Conv norm = 339e-03, max dV(I33.I1.I1.I2.net18) = 16.0942 uV, took 3.71 s.

Order  2 used in       1 subintervals.
Order 14 used in      96 subintervals.
Order 16 used in     306 subintervals.

Conv residual norm = 34.1.
Conv solution-change norm in fdtd = 41.2.

Conv residual norm = 178e-03.
Conv solution-change norm in fdtd = 498e-03.
Number of refinements using multi-interval Chebyshev polynomial spectral algorithm

MIC-PSA finite-difference refinement finished, took 87.53 s.

Number of accepted pss steps =              6242
Initial condition solution time: CPU = 129.98 ms, elapsed = 147.147 ms.
pss: The steady-state solution was achieved in 4 iterations.
Number of accepted pss steps =              6242
Total time required for pss analysis `pss': CPU = 109.913 s (1m  49.9s), elapsed =
Time accumulated: CPU = 110.153 s (1m  50.2s), elapsed = 114.616 s (1m  54.6s).
Peak resident memory used = 616 Mbytes.

modelParameter: writing model parameter values to rawfile.
element: writing instance parameter values to rawfile.
outputParameter: writing output parameter values to rawfile.
designParamVals: writing netlist parameters to rawfile.
primitives: writing primitives to rawfile.
subckts: writing subcircuits to rawfile.

Aggregate audit (3:50:43 PM, Thur Aug 11, 2011):
Time used: CPU = 110 s (1m  50.3s), elapsed = 115 s (1m  55.5s), util. = 95.5%.
Time spent in licensing: elapsed = 721 ms.
Peak memory used = 617 Mbytes.
Simulation started at: 3:48:48 PM, Thur Aug 11, 2011, ended at: 3:50:43 PM, Thur Au
spectre completes with 0 errors, 3 warnings, and 4 notices.

6  HelpAction
```

The runtime is much longer for `highorder=no`, however, it allows the intermodulation products to be resolved. *highorder* set to *yes* is shown in the right subwindow below.

Psaratio (Shooting Only)

This is active when *highorder* is set to *yes*. The default is *one*. Setting smaller values produce more accurate solutions at the expense of runtime. Normally, the default should be used. In the example below, *psaratio* is set to 0.2 and a comparison of the numerical noise floor at 1GHz is made for the LNA used above. The noise floor dropped about 11 dBm when *psaratio* is lowered.

Maxorder (Shooting Only)

Leave this option at the default of 16. The order is set automatically in the highorder algorithm.

This is active when *highorder* is set to *yes*. This controls the maximum order used for the highorder method.

Fullpssvec (Shooting Only)

Leave this option at the default. At the end of each pss iteration, the settled steady-state point needs to be estimated. Because of the inherently variable timestep, not all the timepoints in

the pss solution need to be used to estimate the next starting point. Setting *fullpssvec* to *yes* causes all the timepoints to be read in to calculate the starting voltages for the next iteration.

Fdharms (Shooting Only)

fdharms controls the number of harmonics of the pss beat frequency that are exposed to the `nport`, `mtline`, and `tline` components. This should be set to the number of harmonics that are set for maximum harmonics in the pss *Choosing Analyses* form.

Output Tab

Annotate (Shooting and Harmonic Balance)

This option controls the level of detail in the output log. The default is *status*. *no* produces no output. *title* produces only the conv_norm number at each iteration. *sweep* counts down from 10 to zero at each iteration and displays conv_norm. *status* provides output at 2.5% and then at every 5% of the way to the end at each iteration and the conv_norm number. *steps* provides output at each timestep in the simulation and the conv_norm number at each iteration. An example is shown below with *title* selected.

Below is the Spectre output with *title* selected.

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/D
File  Help                                                    cadence

============================
`pss': time = (0 s -> 50 ns)
============================
Conv norm = 2.93e+03, max dI(I0.I28.I6.ls1:1) = 897.896 uA, took 4.15 s


============================
`pss': time = (0 s -> 50 ns)
============================
Conv norm = 83.4e-03, max dI(Vdd:p) = -29.646 nA, took 3.91 s.

Number of accepted pss steps =              13029
Initial condition solution time: CPU = 49.992 ms, elapsed = 49.777 ms.
pss: The steady-state solution was achieved in 3 iterations.
Number of accepted pss steps =              5463
Total time required for pss analysis `pss': CPU = 10.7784 s, elapsed =
Time accumulated: CPU = 11.1483 s, elapsed = 17.6093 s.
Peak resident memory used = 147 Mbytes.

modelParameter: writing model parameter values to rawfile.
element: writing instance parameter values to rawfile.
outputParameter: writing output parameter values to rawfile.
designParamVals: writing netlist parameters to rawfile.
primitives: writing primitives to rawfile.
subckts: writing subcircuits to rawfile.

Aggregate audit (4:57:52 PM, Tue Aug 9, 2011):
Time used: CPU = 11.2 s, elapsed = 18.4 s, util. = 61%.
Time spent in licensing: elapsed = 515 ms.
Peak memory used = 147 Mbytes.
Simulation started at: 4:57:33 PM, Tue Aug 9, 2011, ended at: 4:57:52 P
spectre completes with 0 errors, 2 warnings, and 4 notices.

39  HelpAction
```

Below is the output with *sweep* selected.

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/D
File  Help                                                    cadence

`pss': time = (0 s -> 50 ns)
============================
......9......8......7......6......5......4......3......2......1......0
Conv norm = 2.93e+03, max dI(I0.I28.I6.ls1:1) = 897.896 uA, took 4.09 s


============================
`pss': time = (0 s -> 50 ns)
============================
......9......8......7......6......5......4......3......2......1......0
Conv norm = 83.4e-03, max dI(Vdd:p) = -29.646 nA, took 4.06 s.

Number of accepted pss steps =              13029
Initial condition solution time: CPU = 29.996 ms, elapsed = 50.7491 ms
pss: The steady-state solution was achieved in 3 iterations.
Number of accepted pss steps =              5463
Total time required for pss analysis `pss': CPU = 10.9883 s, elapsed =
Time accumulated: CPU = 11.3483 s, elapsed = 17.2656 s.
Peak resident memory used = 147 Mbytes.

modelParameter: writing model parameter values to rawfile.
element: writing instance parameter values to rawfile.
outputParameter: writing output parameter values to rawfile.
designParamVals: writing netlist parameters to rawfile.
primitives: writing primitives to rawfile.
subckts: writing subcircuits to rawfile.

Aggregate audit (4:59:27 PM, Tue Aug 9, 2011):
Time used: CPU = 11.4 s, elapsed = 18 s, util. = 63.3%.
Time spent in licensing: elapsed = 470 ms.
Peak memory used = 147 Mbytes.
Simulation started at: 4:59:09 PM, Tue Aug 9, 2011, ended at: 4:59:27 P
spectre completes with 0 errors, 2 warnings, and 4 notices.

39  HelpAction
```

Below is the output with status *selected*.

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/D
File  Help                                                    cadence

    pss: time = 26.26 ns    (52.5 %), step = 10.33 ps    (20.7 m%)
    pss: time = 28.76 ns    (57.5 %), step = 10.33 ps    (20.7 m%)
    pss: time = 31.26 ns    (62.5 %), step = 10.33 ps    (20.7 m%)
    pss: time = 33.76 ns    (67.5 %), step = 10.33 ps    (20.7 m%)
    pss: time = 36.25 ns    (72.5 %), step = 10.33 ps    (20.7 m%)
    pss: time = 38.76 ns    (77.5 %), step = 10.33 ps    (20.7 m%)
    pss: time = 41.26 ns    (82.5 %), step = 10.12 ps    (20.2 m%)
    pss: time = 43.75 ns    (87.5 %), step = 9.768 ps    (19.5 m%)
    pss: time = 46.26 ns    (92.5 %), step = 9.679 ps    (19.4 m%)
    pss: time = 48.75 ns    (97.5 %), step = 9.477 ps      (19 m%)
Conv norm = 83.4e-03, max dI(Vdd:p) = -29.646 nA, took 3.93 s.

Number of accepted pss steps =              13029
Initial condition solution time: CPU = 49.993 ms, elapsed = 56.6151 ms
pss: The steady-state solution was achieved in 3 iterations.
Number of accepted pss steps =              5463
Total time required for pss analysis `pss': CPU = 10.5284 s, elapsed =
Time accumulated: CPU = 10.8583 s, elapsed = 16.8584 s.
Peak resident memory used = 147 Mbytes.

modelParameter: writing model parameter values to rawfile.
element: writing instance parameter values to rawfile.
outputParameter: writing output parameter values to rawfile.
designParamVals: writing netlist parameters to rawfile.
primitives: writing primitives to rawfile.
subckts: writing subcircuits to rawfile.

Aggregate audit (5:00:21 PM, Tue Aug 9, 2011):
Time used: CPU = 10.9 s, elapsed = 17.6 s, util. = 61.9%.
Time spent in licensing: elapsed = 891 ms, percentage of total = 5.06%
Peak memory used = 147 Mbytes.
Simulation started at: 5:00:04 PM, Tue Aug 9, 2011, ended at: 5:00:21
spectre completes with 0 errors, 2 warnings, and 4 notices.

39  HelpAction
```

Below is the output with *steps* selected.

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/D
File  Help                                                    cadence

    pss: time = 49.92 ns    (99.8 %), step = 8.185 ps    (16.4 m%)
    pss: time = 49.93 ns    (99.9 %), step = 8.561 ps    (17.1 m%)
    pss: time = 49.94 ns    (99.9 %), step = 9.267 ps    (18.5 m%)
    pss: time = 49.95 ns    (99.9 %), step = 9.638 ps    (19.3 m%)
    pss: time = 49.96 ns    (99.9 %), step = 9.541 ps    (19.1 m%)
    pss: time = 49.97 ns    (99.9 %), step = 9.461 ps    (18.9 m%)
    pss: time = 49.98 ns     (100 %), step = 9.418 ps    (18.8 m%)
    pss: time = 49.98 ns     (100 %), step = 9.435 ps    (18.9 m%)
    pss: time = 49.99 ns     (100 %), step = 7.685 ps    (15.4 m%)
    pss: time = 50 ns        (100 %), step = 7.685 ps    (15.4 m%)
Conv norm = 83.4e-03, max dI(Vdd:p) = -29.646 nA, took 4.25 s.

Number of accepted pss steps =              13029
Initial condition solution time: CPU = 39.994 ms, elapsed = 50.4529 ms
pss: The steady-state solution was achieved in 3 iterations.
Number of accepted pss steps =              5463
Total time required for pss analysis `pss': CPU = 10.9483 s, elapsed =
Time accumulated: CPU = 11.2383 s, elapsed = 33.5782 s.
Peak resident memory used = 147 Mbytes.

modelParameter: writing model parameter values to rawfile.
element: writing instance parameter values to rawfile.
outputParameter: writing output parameter values to rawfile.
designParamVals: writing netlist parameters to rawfile.
primitives: writing primitives to rawfile.
subckts: writing subcircuits to rawfile.

Aggregate audit (5:01:34 PM, Tue Aug 9, 2011):
Time used: CPU = 11.3 s, elapsed = 34.3 s, util. = 32.9%.
Time spent in licensing: elapsed = 550 ms.
Peak memory used = 147 Mbytes.
Simulation started at: 5:01:00 PM, Tue Aug 9, 2011, ended at: 5:01:34 I
spectre completes with 0 errors, 2 warnings, and 4 notices.

39  HelpAction
```

Save (Shooting and Harmonic Balance)

The default is *allpub*. This saves all the public voltages at all levels of the hierarchy in the schematic. It excludes the internal nodes of the device models. *all* adds the internal nodes of all the devices. *lvl* saves all the nodes including the internal nodes of the devices through the level of hierarchy set in the *nestlvl* option. *lvlpub* is like *lvl*, but it does not save the internal nodes. *selected* saves only the nodes that are specifically saved. In ADE, this is accomplished using the *Outputs - To Be Saved - Select On Schematic* and then selecting

the nodes and terminals specifically in the schematic. At the netlist level, this is accomplished by using a save statement with a list of names to be saved.

Nestlvl (Shooting and Harmonic Balance)

If *save* is set to *lvl* or *lvlpub*, this controls the maximum level of hierarchy to be saved. If *nestlvl* is *1*, only the top level is saved. If *nestlvl* is 2, the top level and the next level down are saved. The value for *nestlvl* can be any integer. The example below shows *lvlpub* selected that saves two levels of hierarchy.

Compression (Shooting and Harmonic Balance)

Normally, this should not be used for RF simulation. It is not digital compression. For RF simulation, where the input is sinusoidal, the size of the results file will normally double. It is useful only for circuits that are predominantly square wave.

Oppoint (Shooting and Harmonic Balance)

Normally, this should be left at the default of *rawfile*. It controls where the time-zero operating point solution should be saved to.

Skipstart (Shooting and Harmonic Balance)

Normally, this is not used for RF simulation unless there is a very long tstab. It is a way of reducing the amount of data in the output file from the tstab interval by not writing many of the timepoints to the output file. Skipstart controls the simulation time where the skipping is to start. The default is 0 (zero).

Skipstop (Shooting and Harmonic Balance)

Normally, this is not used for RF simulation unless there is a very long tstab. This specifies the simulation time in the tstab interval where the skipping is to stop. The default is the stop time in the tstab interval.

Skipcount (Shooting and Harmonic Balance)

If *skipstart* and *skipstop* are set either *skipcount* or *strobeperiod* should be set, but not both. *skipcount* saves one out of every skipcount points to the output data file. A *skipcount* of 3 saves one, then skips two points. Skipcount of 10 saves one in 10 timepoints to the output file. The default is 1, which saves every point. An example is shown below. If *tstab* is 1u, the first 5% and the last 5% has all the datapoints saved, and in between, one out of ten points is saved.

Strobeperiod (Shooting and Harmonic Balance)

If *skipstart* and *skipstop* are set either *skipcount* or *strobeperiod* should be set, but not both. *strobeperiod* forces simulation datapoints between *skipstart* and *skipstop* at the interval of *strobeperiod*. Below is an example where the first 5% and the last 5% have all the datapoints, and in between there are timepoints at 0.01usecond intervals. Note that the simulator may actually calculate many points between the *skipstart* and *skipstop* times, but only the data at the strobe interval is saved to the output file.

Strobedelay (Shooting and Harmonic Balance)

strobedelay specifies a time after the skipstart point for the beginning of the strobing. *strobedelay* must be between 0 (zero) and the time set in *strobeperiod*.

Outputtype (Shooting Only)

This controls what type of information is to be saved in the output file. The default of *all* saves both the time and frequency data, but the data can be restricted to either time or frequency domain data.

Reuse Tab

Write (Shooting Only)

write specifies a filename to which the DC solution used as the first timepoint be written to. If the name does not start with slash (/), the file will be written in the *netlist* directory. To determine the netlist directory location, select *Setup - Simulator/Directory/Host* in ADE. The *Project Directory* field lists the location of the simulation directory. From this directory, the netlist directory is in `<Circuit Name>`/`<simulator name>`/*<schematic or config>/netlist*.

An example is shown below.

Writefinal (Shooting Only)

writefinal writes the final pss timepoint from the shooting window (not the final timepoint of the tstab interval) to the filename specified. See the description for *write* for the location. An example is shown below.

Swapfile (Shooting Only)

During the shooting interval, the solution matrices need to be saved in order to calculate the settled voltages of every node for the next iteration. The solution matrices are the internal solution for all the simultaneous equations for the circuit. The solution vector is also saved, and this is the solution voltages and currents. Normally, this data is kept in memory which makes the Spectre process get larger, starting with the second iteration of pss. On 32-bit systems, a maximum process-size of slightly less than 4GB limits the size of the circuit that can be simulated. Even on 64-bit systems, there is a limit set by the amount of memory that is installed on the system that is running Spectre.

Swapping that is done by the operating system is inefficient because there is no logical relationship for the swapped memory pages. Setting swapfile to a name causes Spectre to write all the solution matrices to the disk in sequential cylinders so that the read time from the disk is faster than using swapping by the operating system. Disk accesses are inherently much slower than memory accesses, however, using swapfile is considerably faster than swapping in the OS. Using swapfile reduces the size of the Spectre process so that it fits in the memory installed in the machine, or allows larger circuits to be simulated. There is a large amount of data that typically is written to the disk. Because of this, the best choice is to write the data in a disk that is local to the machine. If the data is written over a high-latency network, it might actually be better to use the local swap space. Only a trial can determine which is better. Below is an example of setting *swapfile*.

swapfile writes to the netlist directory by default. To locate the simulation directory, select *Setup - Simulator/Directory/Host* and look in the *Project Directory* field. Navigate to that directory, and then navigate to the *<`Circuit_Name`>/spectre/<schematic_or_config>/*

netlist directory. If the size of the swapfile gets larger than 1GB, it splits into multiple files, as shown below.

```
lnx-rdavis3> ls -al swap_file*
-rw-------  1 rdavis mail 1073736200 Aug  9 17:33 swap_file_0
-rw-------  1 rdavis mail  639143800 Aug  9 17:34 swap_file_1
lnx-rdavis3>
```

Writepss (Shooting and Harmonic Balance)

This option specifies that the full internal state of the pss analysis be written out to the file specified in the option. An example is shown below.

writepss `pss_state_file` ...

For large circuits, the file can be quite large.

```
lnx-rdavis3> ls -l pss_state_file
-rw-------  1 rdavis mail 35527866 Aug  9 17:49 pss_state_file
lnx-rdavis3>
```

Readpss (Shooting and Harmonic Balance)

This option specifies the file to be read in as a starting point for the next pss analysis. If nothing is changed in the circuit, then only one iteration is performed and small-signal analyses can be run with a much faster time for the pss analysis. Changes can be made to the circuit or analysis options as long as the topology of the circuit stays the same. Changes introduce a discontinuity at the beginning of the pss analysis, which might be large enough to cause the pss to not converge.

An example of *readpss* is shown below.

When a circuit value is changed, a refinement is done automatically. Basically, the pss analysis is re-run. Instead of starting from scratch, it starts from the solution that is in the *writepss* file. This is shown below.

Checkpss (Shooting Only)

Leave this option at the default of *yes*. This option is used in conjunction with the *readpss* option and the *highorder* option. If *readpss* is set to a filename, (The pss is read from a file), and if *highorder* is set to *yes,* only then is *checkpss* read. If *checkpss* is set to *yes* (default), then any changes in the circuit are observed, and the *readpss* file serves as the starting point for the pss iterations. If the *checkpss* option is set to *no*, then none of the changes to the circuit elements are used. The readpss file is used directly with no changes.

Saveclock (Shooting and Harmonic Balance)

saveclock saves the state of the simulation in the tstab interval at the time interval in seconds specified by *saveclock*. When the clock time has passed in the tstab interval, the file specified by the option *savefile* is created. When subsequent clock time intervals have passed, the file is overwritten. Use only one of the save options at a time. The example below shows *saveclock* set to three minutes.

Saveperiod (Shooting and Harmonic Balance)

saveperiod saves the state of the simulation in the tstab interval at the simulation time interval in seconds specified by *saveperiod*. When the simulation time has passed in the tstab interval, the file specified by the option savefile is created. When subsequent simulation time intervals have passed, the file is overwritten. Use only one of the save options at a time. The example below shows *saveperiod* set to 25 nanoseconds.

Savetime (Shooting and Harmonic Balance)

savetime is a list of times in the tstab interval where the state of the simulation is written out. The list should be specified with spaces between the entries. The information is written out with the filename specified in the *savefile* option with extensions for the time after that. Use only one of the save options at a time.

The example below shows four times in the tstab interval to save.

Savefile (Shooting and Harmonic Balance)

This is the file name to write out the tstab state information to. If you do not specify a filename that starts with the slash (/) character, the file is stored in the *netlist* directory. To locate the *netlist* directory, first select *Setup - Simulator/Directory/Host* menu in ADE, and read the path to the project directory. This is the path to the *simulation* directory. In the simulation directory, navigate to the `<circuit_name>`*/spectre/<schematic or config>/netlist* directory.

Note: Do not specify a relative pathname like *./save_file/run1*.

Recover (Shooting and Harmonic Balance)

recover specifies the file name to recover the tstab simulation from. If *saveperiod* or *saveclock* are used to make the savefile, then just the same name as specified in *savefile* is used. If *savetimes* has a list of times specified, then several files are created at the times specified in the list. Click (...) and browse to the netlist directory where the savefiles are shown.

Select one from the list, and click *Open*. This adds the full path to the *recover* option, as shown below. Also note that in the example below, the *savefile* option is still set, and times after the restart times are specified. This is specifically allowed. As long as tstab is 200n or larger, the files at 175n and 200n will be added and can be reused later.

Now change tstab to a later time. In this case 200n was set.

When the simulation runs, the savefiles are created, as shown below.

The waveform from the original tstab and the restarted tstab overlay is shown below.

Even in the zoomed-in view, the waveforms overlay, as shown below.

Misc Tab

Step (Shooting Only)

Leave this option at the default. All the nodes are checked for curvature in the local truncation error (Numerical integration error) check. If that node has a capacitor or inductor connected to it, the timestep should be controlled to reduce the numerical integration error. If the node does not have a capacitor or inductor connected to it and it has a lot of curvature, the timestep can only be reduced to the value specified in the *step* option. The default is the simulation interval divided by 1000.

Maxstep (Shooting and Harmonic Balance)

In shooting, *maxstep* is observed in the tstab interval and in the shooting window. In harmonic balance, it is observed in the tstab interval only. Normally, *maxstep* should be left at the default and either setting *errpreset* or changing *reltol* is used to control the accuracy. In some cases, extreme accuracy is needed, and this is accomplished by setting a small maxstep value. A small constant timestep produces maximum accuracy at the cost of runtime and memory consumption.

An example with *maxstep* set to 1 picosecond set is shown below.

Tstart (Shooting and Harmonic Balance)

Normally, this is not used. *tstart* specifies the starting time for the tstab interval and defaults to 0 (zero). A negative, 0, or positive *tstart* is allowed. An example with *tstart* set to 100n is shown below.

Use envelope for tstab (Shooting and Harmonic Balance)

This option is not recommended because we are considering ending support for this feature. Normally, a transient analysis is used for the tstab interval. In the case of circuits that have high frequencies and long time constants like oscillator amplitude control or crystal oscillators, the time required for the transient analysis can be excessive. In this case, select *yes* for *Use Envelope for tstab*, and set up a shooting envelope analysis. In the pss *Choosing Analyses* form, set tstab slightly larger than the stop time for the envelope analysis. The example below shows the *Use Envelope for tstab* option set to *yes*. For more information on the envelope analysis, see Chapter 6, “Envelope (ENVLP) Analysis”.

The envlp and pss *Choosing Analyses* forms are shown below.

Choosing Analyses -- Virtuoso® Analog Design Environn
Analysis
tran
dc
ac
noise
xf
sens
dcmatch
stb
pz
sp
envlp
pss
pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
hbsp
Envelope Following Analysis
Engine
Shooting
Harmonic Balance
Stop Time
1u
Start ACPR Wizard
Clock Info
Period
Fund Frequency
4.98G
Clock Name
Number of Harmonics
6
Time Step Control
fixed
adaptive
Oscillator
Oscillator node
/I12/outn
Select
Reference node
Select
Osc initial condition
default
linear
Save Initial Transient Results (saveinit)
no
yes
Accuracy Defaults (errpreset)
conservative
moderate
liberal
Enabled
Options...
OK
Cancel
Defaults
Apply
Help

Choosing Analyses – Virtuoso® Analog Design Environn
Analysis
tran
dc
ac
noise
xf
sens
dcmatch
stb
pz
sp
envlp
pss
pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
hbsp
Periodic Steady State Analysis
Engine
Shooting
Harmonic Balance
Fundamental Tones
Name Expr Value Signal SrcId
Large
Clear/Add
Delete
Update From Hierarchy
Beat Frequency
Beat Period
5G
Auto Calculate
Output harmonics
Number of harmonics
10
Accuracy Defaults (errpreset)
conservative
moderate
liberal
Additional Time for Stabilization (tstab)
1.05u
Save Initial Transient Results (saveinit)
no
yes
Oscillator
Oscillator node
/net011
Select
Reference node
/I12/outn
Select
Osc initial condition
default
linear
Sweep
New Initial Value For Each Point (restart)
no
yes
Loadpull
Enabled
Options...
OK
Cancel
Defaults
Apply
Help

AdditionalParams (Shooting and Harmonic Balance)

This is a field for `<option_name>=value` statements. Multiple `<option_name>=value` statements are allowed with a space between them. Generally, this is used to unlock the beta features, however, there is one case where this field might be useful for an option that is not in the GUI.

In some cases, either in the tstab interval or in the shooting window, the timestep collapses to near zero. The symptom is that the percent complete stays at the same value for many reporting intervals. In this case, there might be a discontinuity in one of the models.

In transient simulation, two things can reduce the timestep; having too much numerical integration error or taking too many iterations at a single timestep can cause the timestep to be cut. Spectre does not report what is causing the timestep to become small. One way to eliminate the possibility of numerical integration error (which is called local truncation error in the simulator) as the cause of reducing the timestep is to set *lteratio* to 1e9. The disadvantage of this is that it disables the normal method of timestep control, therefore, *maxstep* must be set to maintain accuracy.

Another way to accomplish this is to set *lteminstep* to a value of about the stop time divided by 1e5. This allows the normal method of timestep control, but if for some reason the numerical integration error wants to cut the timestep too small, the error is ignored. *lteminstep* is the way to specify this. To do this in the ADE, type `lteminstep=<Stop_Time>/1e5`. If you use `lteratio=1e9`, or if you set *lteminstep* and you still have very small timesteps, try increasing *maxiters* in the *Convergence* tab to between 40 and 100. When *lteratio* or *lteminstep* is set and you still have timestep problems, there is likely a discontinuity in either a device model or in a Verilog-A model. An example is shown below.

Help for Convergence Issues

Spectre can have difficulties in three areas:

- **Trouble with the time-zero timepoint**

  Occasionally, Spectre begins the simulation process, reports the inventory, and then reports that it is starting the tstab interval, and then appears to hang. If you have access to a CPU monitor, you will find that the CPU is at nearly 100% utilization. The likely cause is that the voltages from the signal sources at time=zero are non-zero, and the DC algorithm for calculating the time-zero timepoint is stuck either in `homotopy=source` or `gmin`. In this case, use the global options form (*Simulation - Options - Analog* in the

ADE) and set the *homotopy* option in the *Algorithm* tab to *dptran*. This is highly likely to resolve the problem. If this is ineffective, try setting *try_fast_op* to *no* in the global options *Algorithm* tab. If the problem persists, set `skipdc=sigrampup` and skip the time-zero calculation entirely.

- **Trouble in the tstab interval**

  The typical problem here is that in some cases, the timestep collapses to near zero, and as a result, the simulation appears to be stuck at a certain point. If this is the case, first try setting *relref* to *sigglobal*. This causes the accuracy to degrade a bit, but it allows the simulation to get past the troublesome point. If this fails, in the pss options form set the *lteminstep* option to about the tstab time divided by 1e5. If tstab is 100n, then type `lteminstep=100n/1e5` in the *additionalParams* field, and try again. If this does not fix the problem, there is likely a discontinuous model in the circuit. To be sure, set `maxiters=100`, `lteratio=1e9`, and `maxstep` about equal to `tstab/1e4`. Now run the simulation. If the problem persists, there is very likely a discontinuous model somewhere in the circuit. Use the Spectre output on which the nodes do not converge to determine which model has the problem.

  If you see the same problem in the shooting window, use the same series of steps.

- **Trouble in the shooting interval**

  Driven Circuits

  - Check that the option *iabstol* is not set smaller than 1e-12.
  - If there is a frequency divider in the circuit, remember that the input frequency will be divided down. Specify the divided-down frequency as the pss frequency. Remember that pss must simulate one cycle of the periodicity of the circuit in order to converge.
  - Set a longer tstab. A longer tstab allows the circuit to get closer to steady-state and thus converge better.
  - Check the tstab stopping point. For a sine wave, you want to avoid having the tstab waveform stop at a peak or valley of the waveform. For a square wave, the best point to stop at in the tstab waveform is just before the next transition where most of the voltages have relatively settled.
  - Try changing the tstab time by a fraction of a period. Just moving to a different point can help convergence. Generally, the longer the tstab, the higher the probability of convergence.
  - Do not over-tighten *reltol* and *vabstol*. A practical minimum for *reltol* is 1e-6 and for *vabstol*, 1e-8. Although some circuits may allow smaller values, many would not.

  Oscillators

- Make sure that the oscillator is starting by selecting *yes* for *saveinit* and then viewing the startup waveform. If the waveform looks like a DC level for a while and then starts near the end of the tstab interval, try one of the methods listed in Different ways to start the oscillator on page 548 to start the oscillator more robustly.
- In some cases, Spectre will find a node that looks like it is a frequency divider output even though there is no divider in the circuit. When this happens frequently, convergence is negatively affected. Look at the starting and stopping times during the shooting iterations to see if this is happening and if you are sure that there is no frequency divider in the circuit, set the estimated frequency in the *Choosing Analyses* form to about 3.5 times the actual frequency of oscillation. The reason this works is that at the end of tstab, one period of the estimate is simulated, and then four periods of the estimated frequency is simulated, and during this interval, Spectre looks for a frequency divided output. Setting the estimated frequency to 3.5 times the real oscillation frequency causes just over one cycle of the real oscillation frequency to be simulated, and so Spectre cannot find a frequency-divided output.
- If there is a frequency divider in the circuit, make sure that the oscillator node is on the output of the frequency divider and the pss estimated frequency is the divided down frequency.
- Change tstab by a fraction of a period so that the starting and ending point are different. For sinusoidal oscillators, avoid peaks and valleys. For square wave oscillators, try to end tstab just before a transition. When using this strategy, note that the save and recover options can prevent the need to resimulate from time zero each time a different tstab is used.
- Try a longer tstab. Sometimes there are long time constants in the bias circuitry. When using this strategy, note that the *save* and *recover* options can prevent the need to resimulate from time zero each time a different tstab is used.
- Try setting `reltol=1e-5` and `vabstol=3e-8`. Setting tighter tolerances makes the amplitude and frequency estimate from the tstab interval more accurate.
- Try setting *maxacfreq* to about the frequency times 1000. This forces more timepoints, which makes the simulation more accurate. More time and more memory are required when this is set.
- If the *conv_norm* number is trending down with no large increases from iteration to iteration, try setting *maxperiods* to the 200 range. If *conv_norm* generally trends down during the additional iterations, set maxperiods to a higher value. If the *conv_norm* trends down, and then suddenly has a large increase, it is unlikely that setting *maxperiods* to a higher value will succeed.

Examples

Generally speaking, this section shows examples using shooting. The reason is that harmonic balance has Chapter 3, which is dedicated to the hb *Choosing Analyses* form. All the capabilities shown are available for shooting and harmonic balance unless otherwise noted.

Pss shooting with a single input to an amplifier or the LO applied in a mixer

The example below shows an amplifier.

1. In the schematic, for the RF port assign names for the sources in the *Frequency name 1* and *Frequency name 2* fields.

   This is just a name for you to use to remember the function of that signal. In this case, the name and the variable that sets the frequency are the same, but they do not need to agree. It is perfectly acceptable to have a name like RF1 and a frequency variable freq1. Additionally, the amplitude of both RF signals is set to the same variable name. This is not required either. In the case of measuring a large-signal IP3 where you deliberately want to match the amplitudes of both inputs, it is convenient to use the same name in both amplitudes.

In the case where you want the flexibility to introduce a blocking signal in addition to an RF signal, it makes sense to use two different variables because the blocker usually has a very high amplitude compared to the RF signal.

2. In ADE, assign values for the variables that are used in the input port. In this example, the first RF frequency is set to 2.4GHz, and the second RF frequency is set to 0, which disables that frequency from being generated.

3. In the ADE *Choosing Analyses* form, select pss in the *Analysis* section.

4. Select an engine that you want to use. *Shooting* and *Harmonic Balance* are both available in pss.

5. All the sources in the top-level circuit are read in to the list in the *Fundamental Tones* section. If you have signal sources lower in the level of the hierarchy, select *Update From Hierarchy*.

6. For the beat frequency, either select *Auto Calculate*, or type the frequency for the periodicity of the circuit.

If you are simulating a frequency divider, selecting *Auto Calculate* is inappropriate because it bases the calculation on the frequency of the source in the circuit. In this case, specify the frequency of the divider output.

7. Specify the number of harmonics that you want to calculate from the waveform in the *Number of harmonics* field.

If you do not want harmonics to be calculated, type 0 (zero) in the *Number of harmonics* field. When the number of harmonics is raised above 10, more timepoints are forced in the pss which raise the simulation time and the memory required for the simulation.

8. In most cases, moderate accuracy is enough. If you want more accuracy, select *conservative*. If this is not enough, refer to the section Accuracy and Settings and Trade-Offs on page 542.

9. Run the simulation.

10. When the simulation completes, select *Results - Direct Plot - Main Form* in the ADE. The *Direct Plot Form* is displayed, as shown below.

Plotting the output spectrum in dBm

In the Direct Plot form:

1. Select *pss* results.

2. Select *Power*.

3. To plot the power in dBm from a voltage on a net with an assumed resistor on that net, select *Net (specify R)* from the *Select* drop-down list.

4. Select *dBm*.

5. Select the net in the schematic you want to plot.

The spectral plot is shown in the waveform tool.

Same Example Using Harmonic Balance

The hb *Choosing Analyses* form is only slightly different from the shooting form. The list of sources in the *Tones* section lists all the sources including the ones that have the frequency set to zero. There is an *Oversample Factor* field. There is also an hbhomotopy choice. For more information on harmonic balance, see Chapter 3, “Frequency Domain Analyses: Harmonic Balance”.

Choosing Analyses – Virtuoso® Analog Design Environn
Analysis
tran
dc
ac
noise
xf
sens
dcmatch
stb
pz
sp
envlp
pss
pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
hbsp
Periodic Steady State Analysis
Engine
Shooting
Harmonic Balance
Tones
Name
Expr
Value
SrcId
frf2
frf2
0
PORT1
frf
frf
2.4G
PORT1
Beat Frequency
2.4G
Auto Calculate
Oversample Factor
1
Number of Harmonics
6
Accuracy Defaults (errpreset)
conservative
moderate
liberal
Convergence
Additional Time for Transient-Aided HB (tstab)
Save Initial Transient Results (saveinit)
no
yes
Harmonic Balance Homotopy Method
default
Oscillator
Sweep
New Initial Value For Each Point (restart)
no
yes
Loadpull
Enabled
Options...
OK
Cancel
Defaults
Apply
Help

The waveform results agree for shooting and harmonic balance, as shown in the waveform tool.

Loadpull

Generally, loadpull is performed using harmonic balance. There are several reasons, but the biggest reason is that harmonic balance is almost always faster than shooting for a loadpull simulation.

PSS Harmonic Balance

For a loadpull simulation, add a triplexer (for setting the reflection and angle for up to three harmonics) or the ten_plexer (for up to 10 harmonics) as the load to the circuit. These are present in the `rfLib` library.

Normally, properties are used to set the reflection coefficient magnitude and phase for the harmonics that are present in the triplexer or ten_plexer. The example below shows the ten_plexer, and has 0 (zero) set for all the reflection coefficients and angles.

Descend-read into the triplexer or ten_plexer. Triplexer is shown. The top bandpass filter is for the first harmonic. The middle is the second harmonic. The bottom is the third harmonic.

1. Add two variables in ADE.

The example above shows `mag` and `theta` variables added. These variables should not be used to set any component values in the circuit.

Triplexer from `rfLib` directory contains three filters for the first, second, and third harmonics and portAdapters with port loads for each of the three filters.

2. Open the PSS or HB *Choose Analyses* form.

3. Select the *Loadpull* checkbox.

4. Click *Select* on the right of the greyed out *Load Instance* field.

5. In the triplexer schematic, select the topmost port on the right side of the schematic if you want a loadpull for the first harmonic. Select the next port down for a loadpull of the second harmonic. Select the third port down for the third harmonic.

   The instance name is entered in the *Load Instance* field. Direct entry by typing is not allowed for this field.

6. Click the *Variable* button on the right of the *rho (Z)* field. Select the variable you entered in ADE.

   This variable will be used to set the value of the reflection coefficient in the port.

7. Specify a sweep range, and set the spacing.

8. When completed, click *OK*.

   The values are displayed in the *Choose Analyses* form.

9. Click the *Variable* button on the right of the *phi (Z)* field. This sets the angle of the reflection coefficient.

10. In the *Select Design Variable* window, select the second variable you entered and specify the sweep limits.

11. When completed, click *OK*.

The *Choose Analyses* form is updated.

12. Specify the system resistance in the *Z0* field.

The *Choosing Analyses* form should appear similar to the one below.

pz
sp
envlp
pss
pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
hbsp
Periodic Steady State Analysis
Engine
Shooting
Harmonic Balance
Tones
Name
Expr
Value
SrcId
RF
RF_in
2.45G
PORT3
Beat Frequency
2.45G
Auto Calculate
Oversample Factor
Number of Harmonics
7
Accuracy Defaults (errpreset)
conservative
moderate
liberal
Convergence
Additional Time for Transient-Aided HB (tstab)
3n
Save Initial Transient Results (saveinit)
no
yes
Harmonic Balance Homotopy Method
default
Oscillator
Sweep
New Initial Value For Each Point (restart)
no
yes
Loadpull
Load Instance
/I8/PORT14
Select
rho (Z)
mag
{1m:991m}step:110m
Variable ...
phi (Z)
theta
{-180:160}step:20
Variable ...
Z0
50
Enabled
Options...
OK
Cancel
Defaults
Apply
Help

13. When completed, click *OK* and run the simulation in ADE.
14. In ADE, select *Results - Direct Plot - Main Form*.

There are two methods to plot the loadpull curves.

First method

1. Select *pss or hb results*.

2. Select *Power Contours*.

3. Select *Magnitude*, *dB10*, or *dBm*.

4. Specify 2 more than the number of contour lines you want.

   The smallest and largest power occurs at a single point which is plotted, but are very difficult to see in the waveform tool.

5. Select the *Close Contours* check box. This causes the contours from the loadpull to be displayed as circles or ellipses, depending on the distortion of the circuit.

6. Select the first harmonic.

7. Select the top terminal of the top port in the triplexer.

The power contours appear.

Second method

1. Select *pss* or *hb results*.

2. Select *Power*.

3. Select *variable*.

4. Select *Magnitude*, *dB10*, or *dBm*.

5. Select the *harmonic number*.

6. Select *Loadpull Contour*.

7. Specify 2 more than the number of curves you want.
8. Select the *Closed Contours* check box. This causes the contours from the loadpull to be displayed as circles or ellipses, depending on the distortion of the circuit.
9. Select the top terminal of the top port of the triplexer circuit.

   The loadpull curves appear in the waveform tool.

PSS Shooting Loadpull

1. For a loadpull simulation, copy a a triplexer (for setting the reflection and angle for up to three harmonics) or the ten_plexer (for up to 10 harmonics) into a local writable library, and rename the cell. These components are located in the `rfLib` library.

2. Open the triplexer_local schematic, and select the portAdapter for the desired harmonic. The top portAdapter is for the first harmonic, and the harmonic numbers increase as you move down in the schematic. The top portAdapter is highlighted below.

The properties list is set as follows:

3. Change *Phase of Gamma (degrees)* and *Mag of Gamma (linear scale)* to variable names. Note that the phase is above the magnitude in the properties list.

4. Save the local copy.

5. Add the local copy of the triplexer or ten_plexer as the load in the schematic. Save the schematic.

Virtuoso® Analog Design Environment L Editing: RF_APS DLNA1_RFT_tb1_loadpull schematic
Launch File Edit View Create Check Options Migrate Window Help
cadence
Workspace: ADEL
Search
Navigator
Default
Search
Name
DLNA1_RF...loadpull
I0 (G1LNA)
I14 (triplexer)
C0 (cap)
C2 (cap)
I7 (balun)
I9 (balun)
L0 (ind)
L1 (ind)
L4 (ind)
PORT1 (port)
Vc (vdc)
Vdd (vdc)
Vpd (vdc)
gc
Property Editor
G1LNA
I0
Jul 21 15:42:56 2010
Cadence Design
Diff. LNA Testbench
A.
1.0
OF
mouse L: schSingleSelectPt()
M: sevSaveState('sevSession2)
R: schHiMousePopUp()
5(6)
Cmd: Sel: 0 Status: ...successful. T=27 C Simulator: spectre

6. Add the two variables you used in the local triplexer or ten_plexer into ADE.

The example above shows `mag` and `theta` variables added. These variables should not be used to set any component values in the circuit.

Triplexer from your local library directory contains three filters for the first, second, and third harmonics and portAdapters with port loads for each of the three filters.

7. Open the PSS *Choose Analyses* form.

8. Set up the top of the form as usual.
9. Select the *Loadpull* checkbox.

10. Click *Select* on the right of the greyed out *Load Instance* field.
11. In the triplexer_local schematic, select the topmost portAdapter on the right side of the schematic if you want a loadpull for the first harmonic. Select the next portAdapter down

for a loadpull of the second harmonic. Select the third portAdapter down for the third harmonic. The first harmonic portAdapter is highlighted below.

The instance name is entered in the *Load Instance* field. Direct entry by typing is not allowed for this field.

12. Click the *Variable* button on the right of the *rho (Z)* field. Select the variable you entered in ADE.

This variable is used to set the value of the reflection coefficient in the port.

13. Specify a sweep range, and set the spacing.

14. When completed, select *OK*.

The values are displayed in the *Choose Analyses* form.

15. Click the *Variable* button on the right of the *phi (Z)* field. This sets the angle of the reflection coefficient.

16. In the *Select Design Variable* window, select the second variable you entered and specify the sweep limits.

17. When completed, click *OK*.

The *Choose Analyses* form is updated.

xf
sens
dcmatch
stb
pz
sp
envlp
pss
pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
hbsp
Periodic Steady State Analysis
Engine
Shooting
Harmonic Balance
Tones
Name
Expr
Value
SrcId
RF
RF_in
2.45G
PORT3
Beat Frequency
2.45G
Auto Calculate
Oversample Factor
Number of Harmonics
7
Accuracy Defaults (errpreset)
conservative
moderate
liberal
Convergence
Additional Time for Transient-Aided HB (tstab)
3n
Save Initial Transient Results (saveinit)
no
yes
Harmonic Balance Homotopy Method
default
Oscillator
Sweep
New Initial Value For Each Point (restart)
no
yes
Loadpull
Load Instance
/I8/PORT14
Select
rho (Z)
mag
{1m:991m}step:110m
Variable ...
phi (Z)
theta
{-180:160}step:20
Variable ...
Z0
50
Enabled
Options...
OK
Cancel
Defaults
Apply
Help

18. When completed, click *OK* and run the simulation in ADE.

19. In ADE, select *Results - Direct Plot - Main Form*.

There are two methods of plotting loadpull curves.

First method

1. Select *pss results*.

2. Select *Power Contours*.

3. Select *Magnitude*, *dB10*, or *dBm*.

4. Specify 2 more than the number of contour lines you want.

   The smallest and largest power occurs at a single point which is plotted, but are very difficult to see in the waveform tool.

5. Select the *Close Contours* check box. This causes the contours from the loadpull to be displayed as circles or ellipses, depending on the distortion of the circuit.

6. Select the first harmonic.

7. Select the top terminal of the top port in the triplexer.

The power contours appear.

Second method

1. Select *pss results*.

2. Select *Power*.

3. Select *variable*.

4. Select *Magnitude*, *dB10*, or *dBm*.

5. Select the *harmonic number*.

6. Select *Loadpull Contour*.

7. Specify 2 more than the number of curves you want.
8. Select the *Closed Contours* check box. This causes the contours from the loadpull to be displayed as circles or ellipses, depending on the distortion of the circuit.
9. Select the top terminal of the top port of the triplexer circuit.

   The loadpull curves appear in the waveform tool.

Two Input Frequencies

Two input frequencies can be simulated as long as the beat frequency is high compared to the frequencies that are applied to the circuit. In general, if the pss beat frequency is smaller than about 1/25th the highest input frequency, the simulation is faster, if you use qpss or hb.

In the example below, 30 periods of 900MHz and 31 periods of 960 MHz need to be calculated. This is on the edge of acceptability for pss. For this simulation, it might be faster

to simulate using qpss or hb because fewer harmonics need to be solved, which requires less memory and time.

Note that when the frequencies are moved slightly to 910M and 970M, the beat frequency drops to 10MHz. In this case, it would definitely be faster to use hb or qpss. These analyses are much faster than pss. Also, since the pss frequency is lower, more harmonics need to be calculated in order to see the desired mixing products.

Sweeps

For the next plots, a sweep is set up in the pss *Choosing Analyses* form. A sweep is needed in order to measure compression.

Sweeps can be made either in the parametric plot tool in ADE where the number of swept things is not limited, or in the pss *Choosing Analyses* form, where the number of swept things is limited to three. This example shows using the pss *Choosing Analyses* form shown below. For documentation on the parametric plot tool, please refer to the ADE documentation.

Choosing Analyses - Virtuoso® Analog Design Environmer
xf
sens
dcmatch
stb
pz
sp
envlp
pss
pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
hbsp
Periodic Steady State Analysis
Engine
Shooting
Harmonic Balance
Fundamental Tones
Name Expr Value Signal SrcId
1 frf1 frf1 900M Large RF
Large
Clear/Add
Delete
Update From Hierarchy
Beat Frequency
Beat Period
900M
Auto Calculate
Output harmonics
Number of harmonics
5
Accuracy Defaults (errpreset)
conservative
moderate
liberal
Additional Time for Stabilization (tstab)
Save Initial Transient Results (saveinit)
no
yes
Oscillator
Sweep
1
Variable
1
2
3
Frequency Variable?
no
yes
Variable Name
Select Design Variable
Sweep Range
Start-Stop
Center-Span
Start
Stop
Sweep Type
Linear
Logarithmic
Step Size
Number of Steps
Add Specific Points
New Initial Value For Each Point (restart)
no
yes
Loadpull
Enabled
Options...
OK
Cancel
Defaults
Apply
Help

1. Select a sweep number from the *Sweep* drop-down list.

2. Select the type of parameter that is being varied from the *Variable* drop-down list.

3. Type the variable name in the *Variable Name* field. You can also select the variable from a list. To do this, click *Select Design Variable*. and the list is displayed.

Select the desired variable, and click *OK* to enter the variable.

4. Define the sweep range. Set up a sweep range, and if you desire values that are in between the sweep values, or above or below the sweep values, select *Add Specific Points*. Type a list of values separated by spaces. This is shown below.

For many power measurements, currents need to be saved. If you are using Shooting, these currents can be saved, To save all the currents from the ADE environment. Select *Outputs - Save All*, and then select *all* for *Select Device Currents (currents)*. This takes the currents that are calculated from every component in the circuit and adds them into the simulator result file.

For hb, port, and voltage source, only currents are saved, and this save is automatic. The ADE *Save Options* form shown above has no effect in hb. If you need to measure a current other than in a port or voltage source, add an iprobe in series with the branch.

If all the currents need to be saved when harmonic balance is used as the engine, select *yes* for the *Select AC terminal currents (useprobes)* option above. This adds a zero volt battery in series with every branch in the circuit. Although this results in saving all the currents, it also adds entries in the harmonic balance matrices for each voltage source. As a result, the hb matrices get much larger, therefore, more time and memory are required to solve the equations.

In addition, there is a very small chance of causing unresolvable numerical issues and the simulator issues a singular matrix error message. If you see this error, turn off the *Select AC terminal currents (useprobes)* option and run the simulation again.

Note: The *Select AC terminal currents (useprobes)* option is not recommended if you run a small-signal analysis after the large-signal shooting pss analysis.

Compression Point

This requires a swept input power setup as shown in the section Loadpull on page 611.

Make sure that the power at the first point in the sweep is low enough so that there is no compression in the output at that power level. The first point in the sweep is used to establish the uncompressed power for the compression measurement.

To plot a compression point measurement:

1. Select *Results - Direct Plot - Main Form* in the ADE.

The Direct Plot Form is displayed, as shown below.

2. Select *Compression Point* from the *Function* section.
3. Select the type of thing that needs to be probed in the circuit from the *Select* drop-down list.
4. Specify the compression level.
5. Specify input or output-referred compression amount.

6. Select the output harmonic to be plotted.
7. Make the selection that is shown at the bottom of the *Direct Plot Form*. In this example, a port in the schematic that is used as a load should be selected.

   The compression curve is displayed.

Voltage Gain

This example shows a swept measurement. To set this up, see Loadpull on page 611. A sweep is not required for this measurement.

The voltage gain direct plot function provides the output spectrum divided by the amplitude of the input at the specified input frequency. Note that the input amplitude at the input

frequency has a single value. All the output harmonics at all the calculated output frequencies are scaled by the single voltage at the input.

To plot the voltage gain:

1. Select *Voltage Gain* from *Function* section.
2. Select the appropriate modifier.
3. Select the power level of the sweep.

4. Select the input harmonic frequency.
5. Select the output node in the schematic.
6. Select the input node in the schematic.

Below is the output spectrum of the circuit, divided by the amplitude of the input at the input frequency.

Power Gain

This example shows a swept measurement. To set this up, see Loadpull on page 611. A sweep is not required for this measurement.

The power gain direct plot function provides the output spectral power divided by the power of the input at the specified input frequency. Note that the input power at the input frequency has a single value. All the output harmonics at all the calculated output frequencies are scaled by the single voltage at the input.

To plot the power gain of a circuit:

1. Select *Power Gain* from the *Function* section.
2. Select the appropriate modifier.
3. Select the input power.
4. Select the frequency of the input.
5. Select the terminal of the load.

6. Select the terminal of the input source.

Below is the output power spectrum, divided by the magnitude of the input power at the input frequency.

Power-Added Efficiency

This example shows a swept measurement. To set this up, see Loadpull on page 611. A sweep is not required for this measurement.

To plot power-added efficiency:

1. Select *Power Added Eff* from the *Function* section.
2. Select the frequency of the input.
3. Select the number of DC sources that power the circuit.
4. Select the terminal of the output load.

5. Select the terminal of the input source.

6. Select the terminal(s) of the DC sources.

The swept PAE curve is displayed.

Compression Versus Output Power

This example shows a swept measurement. To set this up, see Loadpull on page 611.

To plot compression versus output power:

1. Select *Comp Vs Pout* from the *Function* section.
2. Select the output frequency.
3. Select the load port in the circuit.

The compression curve is displayed. The assumption is that the first point in the sweep has no compression.

Power Gain Versus Output Power

This example shows a swept measurement. To set this up, see Loadpull on page 611.

To see the power gain versus the output power:

1. Select *Power Gain vs Pout* from the *Function* section.
2. Select the appropriate modifier.
3. Select the output frequency.
4. Select the input frequency.
5. Select the terminal of the load.

6. Select the terminal of the input source.

The plot is displayed. The vertical axis is the power gain. The horizontal axis is the swept input power.

Plotting THD

This example shows a swept measurement. To set this up, see Loadpull on page 611. A sweep is not required for this measurement.

To plot the total harmonic distortion:

1. Select *THD* from the *Function* section.
2. Select the output frequency.
3. Select the net in the schematic.

The THD versus input power plot is displayed.

Large-signal IP3

This example shows a swept measurement. To set this up, see Loadpull on page 611.

In pss, all the harmonics of the input waveform are calculated. If the input frequencies get close together, the pss beat frequency becomes small compared to the highest input frequency. If you need to simulate more than 25 periods of the highest input frequency, switch to hb or qpss which run faster than pss.

To plot a large-signal IP3, both inputs are enabled in the ADE window by specifying the values in the *Variables* section of the ADE for the frequency and amplitude of both RF sources. Note

that in this case, the highest frequency (950MHz) divided by the beat frequency (50MHz) is 19. This is a reasonable problem for pss.

The input is swept over a reasonable range of power levels. If the lower boundary of the sweep is too small, the noise floor of the simulator will be present in the result. If the input power gets too high, the inter-modulation product gets unpredictable. In this example, the

power is deliberately started at a very small value and the stopping power is in a very nonlinear region. This is done so the proper way to plot an IP3 can be shown later,

1. When the setup is complete, run the simulation.
2. When the simulation completes, select *Results - Direct Plot - Main Form* in ADE.

3. Select *IPN Curves* from the *Function* section.
4. Select the type of output signal to plot (or Port) The default is to select a port in the schematic.
5. Select *Variable Sweep*.
6. Set the order of the curves to be plotted. Usually, the order is 2nd or 3rd.
7. Select whether you want input or output-referred IP3 or IP2.
8. Leave the extrapolation point blank for the first plot (explained later).
9. Select the third or second-order frequency from the 2nd or *3rd Order Harmonic* field.
10. Usually, you select the frequency that has the highest amplitude, which is often the lower frequency term.
11. Select the first-order frequency from the *1st Order Harmonic* field. Usually you want the smallest amplitude, which is often the higher frequency term.
12. Select the output port in the schematic.

The swept large-signal IP3 is plotted, as shown below.

- Plotting IP3 always involves setting an extrapolation point. First plot the curve using the default for the extrapolation point.
- At high power, the third-order curve becomes unpredictable. At low power, the simulation has a noise floor. The data from the intermodulation product starts to level off.
- Set the extrapolation point above the noise floor, but below the large-signal region.
- The actual data for the intermodulation product should follow the ideal curve for at least one data point in the sweep above and below the extrapolation point.

- When -30 is set as the extrapolation point, the data for the third-order frequency follows the ideal curve above and below the extrapolation point, as shown below.

- Note that the IP3 calculation has increased by about 7dB. Make sure that the extrapolation point is set correctly to get the correct measurement.

- The above plot is actually a small-signal IP3 measurement. If this is your objective, using rapid IP3 or IP2 in the AC *Choosing Analyses* form for amplifiers and in the PAC *Choosing Analyses* form for circuits that translate frequencies will run much faster and will provide a solution that is just as accurate.
- Note that `bsim3` and `bsim4` models have a limitation that causes the IP3 measurement to become inaccurate for switched-FET mixers whether it is done in rapid IP3, pss, qpss, or hb. In this case, the psp model is highly recommended. If this model is not available from the foundry, use pss, qpss, or hb at as high an input amplitude as possible without causing large-signal behavior in the third-order product.

Oscillator Simulation

There are no input signals for an oscillator. In the pss analysis, the waveform needs to be calculated. To do this, both the frequency and the amplitudes need to be solved. This requires more iterations to calculate the solution.

In the ADE, variables are used to set the tuning voltage and the power supply voltage.

In the pss *Choosing Analyses* form:

1. Select *pss*.

2. Select the engine that you want to use.

3. Specify a frequency estimate that is between 0.5 and 1.5 times the actual oscillation frequency if you are using the linear oscillator initial condition, or from 0.25 to four times the actual oscillating frequency if you use the default oscic.

4. Set the number of harmonics you want for the spectral calculation.

5. Select the accuracy. *conservative* accuracy is recommended for oscillators.

6. Specify about zero to five periods for tstab when the linear oscic is used and about 100 periods if the default oscic is used on a feedback oscillator.

If you have a ring oscillator, set about three periods for tstab.

7. If you want to see the startup waveform, select yes for *saveinit*.

8. Select *Oscillator*. If you have a single-ended oscillator, specify one node inside the feedback system for the *Oscillator node*. If you have a differential oscillator, specify two nodes at mirror locations in the *Oscillator node* and the *Reference node*.

9. If you have a feedback oscillator, linear oscic is recommended because tstab can be drastically shortened, thus saving simulation time.

10. If you have a ring oscillator, set one stage high or low using initial conditions, and use the default oscic.

11. Now run the simulation. When the simulation completes, select *Results - Direct Plot - Main form* in ADE.

Plotting the Oscillator Startup Waveform

To plot the startup waveform, select tstab results and then select a node in the circuit. If you do not see the tstab results, go back to the *Choosing Analyses* form and select *yes* for *saveinit*.

When the linear oscic is used, the startup characteristics of the waveform are not very evident. If you want to see the startup waveform to evaluate how robustly your oscillator starts up, use the default oscic and provide a very small starting kick at time zero plus so that the waveform starts small in amplitude and builds up. In this case, it is likely that tstab needs to

be longer to allow the oscillator to reach a near steady-state operation before going into the pss solver.

The result is displayed in the waveform tool. Note that with the linear oscic, the oscillator starts at time = zero with the estimate of the frequency and amplitude from the very beginning. This

allows a very short tstab. Also note that the simulation continues after the setting of tstab. During this time, Spectre searches the oscillator for a divided-down output frequency.

Plotting the Oscillator Output Spectrum

To plot the output spectrum, select *pss*, and then select either *Voltage* or *Power*.

When a node is selected in the schematic, the power in dBm is plotted in the waveform tool, as shown below.

Additions for Oscillator Swept Tuning Voltage

In the *Choosing Analyses* form:

1. Select *Sweep*.
2. Type the variable name that sets the tuning voltage in the *Variable Name* field.

3. Set a reasonable sweep range for your oscillator.

Choosing Analyses - Virtuoso® Analog Design Environmer
pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
hbsp
Periodic Steady State Analysis
Engine
Shooting
Harmonic Balance
Fundamental Tones
Name Expr Value Signal SrcId
Large
Clear/Add
Delete
Update From Hierarchy
Beat Frequency
Beat Period
5G
Auto Calculate
Output harmonics
Number of harmonics
10
Accuracy Defaults (errpreset)
conservative
moderate
liberal
Additional Time for Stabilization (tstab)
2n
Save Initial Transient Results (saveinit)
no
yes
Oscillator
Oscillator node
/net011
Select
Reference node
/I12/outn
Select
Osc initial condition
default
linear
Sweep
1
Variable
Frequency Variable?
no
yes
Variable Name
Vt
Select Design Variable
Sweep Range
Start-Stop
Center-Span
Start
0.15
Stop
3.15
Sweep Type
Linear
Logarithmic
Step Size
Number of Steps
0.2
Add Specific Points
New Initial Value For Each Point (restart)
no
yes
Loadpull
Enabled
Options...
OK
Cancel
Defaults
Apply
Help

4. Run the simulation.

5. When the simulation completes, select Results - *Direct Plot* - *Main Form* in ADE. The *Direct Plot Form* is displayed.

Plotting Output Power Versus Tuning Voltage

To plot the power as a function of the tuning voltage:

1. Select *pss* results.

2. Select *Power*.

3. Select *variable* sweep.

4. Select *dBm*.

5. Select the harmonic number. The frequency range is also shown.

6. Select the node in the schematic you want to plot.

The output power versus the tuning voltage is plotted, as shown below.

Plotting Output Power Versus Output Frequency

Now plot the harmonic frequency on the same plot.

The frequency is plotted on the same subwindow. A second vertical axis is used.

To plot the output power versus the frequency, right click on one of the numbers on the X Axis, and select Y vs. Y. The *Y vs Y* window appears. Select the frequency for the X Axis from the *Select a Trace* drop-down list and click *OK*.

The plot of the output power versus the output frequency is plotted in a new subwindow.

Plotting the Output Frequency Versus the Tuning Voltage

Now quit the waveform window, and select *Add To Outputs* in the *Direct Plot Form*, and plot the harmonic frequency again.

The tuning curve is plotted again, as shown below.

In the ADE, a new expression is added to the *Outputs* section.

Plotting the Modulation Sensitivity

1. Double-click the expression in the *Outputs* section.

   The *Setting Outputs* window is displayed. The expression is different for every oscillator.

2. Click *Open*.

The calculator opens with the expression in it.

3. Select the derivative function. The expression changes.

4. Set the plotting mode to *New Subwindow*.
5. Click the *Plot* icon.

The modulation sensitivity in MHz/Volt is plotted in the subwindow on the right.

Periodic AC Analysis (PAC)

Small-signal Versus Large-Signal Analysis for Conversion Gain

One way to measure conversion gain is to apply the LO and the RF signals in harmonic balance (hb) or quasi-periodic steady-state (qpss) and calculate the full large-signal response at each RF input frequency, and then plot the output as a function of the sweep frequency. While this works, because all the large-signal effects and many harmonics are calculated, the required time for the simulation can be significant.

The alternative is to use a small-signal approach similar to the AC analysis. The AC analysis runs considerably faster than running the transient at multiple frequencies because it ignores the large-signal effects. In circuits with frequency translation, the AC analysis is unusable because the calculation is based on the DC bias point. In hb Periodic AC (PAC) analysis, instead of using the DC bias point, the harmonic balance periodic steady-state (pss) solution with just the LO is used as the basis of the calculation. The large-signal analysis only calculates the harmonics of a single input, and no mixing products, and then hb pac is run after the hb pss analysis and calculates the outputs that are produced by mixing the pac input with the harmonics that are in the pss analysis. The overall time savings can be very significant.

Example: Conversion Gain (Down Conversion)

Consider the double-balanced diode mixer shown below.

The RF input source has the frequency and amplitude set to variable names in order to allow changing the test setup without needing to change the circuit. This is suggested for all the sources used in the circuit that might need to be changed. Also note that the pac magnitude term is set to 1 volt. This makes it easy for a conversion gain measurement because the input signal is 0dB when the *dB20* modifier is used in the *Direct Plot Form* for the pac result.

The property list is shown below.

The easiest way to get the variables list into the ADE is to select *Variables - Copy From Cellview*. To set the values, select the value field to the right of the variable, type the desired value, and press *Enter*.

The LO frequency is 1GHz and its amplitude is 1.4 volts peak. This is +13 dBm. (+13 dBm is a common drive level for diode mixers.) The RF frequency is 1.1GHz, but it is disabled because the amplitude is set to 0 (zero). Setting either the frequency or the amplitude to zero disables the production of that waveform from the port. This provides an easy way to enable or disable signals without needing to change the schematic.

The pss analysis needs to be set up first. Note that both shooting and harmonic balance are available in pss for the engine. The example below shows a single input at 1GHz with 10 harmonics using the shooting engine. For more information, see Overview of Periodic Steady-State (pss) Analysis on page 505. Diode mixers have rapid changes in the diode

currents. Because of this rapid change, shooting is likely to be faster and more accurate than harmonic balance.

Next, the PAC analysis needs to be set up. Like AC, the frequency range specified in the form is the input frequency range. When shooting is used for the pss engine, *Maximum sideband* should be set somewhere less than four times the number of harmonics set in the pss analysis. Setting 10 sidebands calculates the mixing products from the input frequency in PAC with the harmonics through the tenth harmonic in the pss analysis. The actual number chosen depends on the mixing products that are important in your application. When harmonic balance is chosen for the engine, leave the *Maximum sideband* field blank.

Once the analyses are set up, the simulation can be run by selecting *Simulation - Netlist and Run* in the ADE. When the simulation finishes, select *Results - Direct Plot - Main Form*.

The example below shows a typical *Direct Plot Form* setup to measure the conversion gain.

When the net in the schematic is selected, the waveform is displayed.

Note that all the short line segments are the mixing products that are produced when mixed with the harmonics in the periodic steady-state (pss) large-signal analysis.

In the down-conversion case, the frequencies that are produced for the down-converted output product are desired. In the *Direct Plot Form*, select *sideband* instead of *spectrum*,

and then select the desired frequency range from the list at the bottom of the form. Only that mixing product will be displayed, as shown below.

The conversion loss is about 6dB.

Example: Conversion Gain (Up Conversion)

The process is similar for an up-conversion mixer. In this case, the input frequency is low, as shown below. The LO remains at 1GHz, as it was in the previous section.

The RF outputs are 50MHz through 150MHz above and below the 1 GHz LO.

In the *Direct Plot Form*, both the outputs above and below the LO frequency might be desired.

Select the first frequency, and click the second while holding the *Ctrl* key. Now select the output net in the circuit and the results will plot.

Note that the conversion loss is almost identical for both mixing products.

PAC General Principles

The pss solution with LO calculates the amplitude of the harmonics and characterizes the nonlinearity of the system. Therefore, in pac, the output mixing products of the input signal mixing with the LO harmonics can be calculated. Like ac analysis, pac allows the input to be swept in frequency to calculate a frequency response of the output mixing products. Also, like ac, when pac is used for conversion gain measurements, the conversion gain is the same regardless of the input amplitude. The small-signal conversion gain is calculated and is a good representation of the gain until compression is approached or exceeded in the real circuit. If a measurement of compression is required, use a large-signal analysis like pss, hb, or qpss.

PAC used with Harmonic Balance PSS

When harmonic balance is used for the pss analysis, the harmonics of the circuit are calculated directly by the harmonic balance engine.

Think of a perfectly linear circuit with a single input applied. Because the circuit is perfectly linear, in the pss analysis when harmonic balance is used, the frequency domain solution only has the harmonic at the input frequency at every net in the circuit. No other harmonics are created because the system is perfectly linear. Because the circuit is linear, if a second input is applied, no frequency translation can occur.

Now imagine a real circuit with a single small input amplitude signal at 1GHz. In the frequency domain with harmonic balance selected in pss, the first harmonic is present along with the higher harmonics. Because the input signal is small, the signals at the harmonics of 1GHz are very small. If a second input is applied at 1.1 GHz, that signal is weakly converted when it is mixed with the harmonics of the circuit. Certainly, 1.1GHz would appear at the output with no mixing. 100MHz and 2.1GHz are produced by mixing with the 1GHz fundamental frequency. 900MHz and 3.1GHz are produced when 1.1GHz mixes with the second harmonic of the 1GHz input. 1.1GHz mixes with the other harmonics as well, producing a wide range of output frequencies from a single input frequency. The harmonic output might look like the figure below. Only the first three harmonics are shown. The black harmonics are the pss harmonics, and the red mixing products are pac outputs, which are called sidebands.

Note that there is an output mixing product that is created at 100MHz. Because the real circuit is nonlinear, harmonics of 100MHz are created in the circuit. In pac, the assumption is that all the signals are small, and with this assumption, the harmonics of 100MHz do not need to be calculated. This assumption also applies to all the other outputs in pac. Therefore, the PAC output is a partial output from the circuit with only the main mixing products displayed. Since in most cases, these are the desired outputs, conversion gain can still be obtained by using pac. In other words, pac calculates the main mixing products that are created by mixing with the pss harmonics, but the harmonics of the pac outputs are not calculated.

Now imagine that the 1GHz signal is larger. If the input goes up 10dB, the first harmonic goes up 10dB, In the frequency domain, if harmonic balance is selected in pss, the second harmonic would come up 20dB, the third harmonic would come up 30dB, and so on. If a second input at 1.1GHz is applied to the circuit, this mixes with the fundamental and the harmonics of 1GHz and more strongly mixes up-and-down in frequency. This is illustrated below. The black harmonics are the pss harmonics, and the red mixing products are pac outputs, which are called sidebands.

PAC used with Shooting PSS

Shooting is a variation of the transient analysis where the individual timepoints are calculated, and the entire waveform is iterated to calculate the steady-state waveform. The transient algorithm works by using all the equations at all the nodes in the circuit where the assumption is made that the sum of the currents is zero at every node and at every timepoint. Ohm's law is observed, and the equations look like `I=V*G` where `G` is the conductance. In general, the matrix form of the equations is constructed, so the system of simultaneous equations becomes `[G]*[V]=[I]` where `[G]` is a square matrix, and `[V]` and `[I]` are columns. To solve for [`V]`, the `[G]` matrix is inverted and this matrix multiplies the `[I]` matrix to solve for all the voltages on all the nodes. At the end of the pss analysis, all the solution matrices are saved as data to be passed to the small-signal analyses. The result is that the conductance is calculated and saved for every nonlinear component in the circuit at every timepoint.

For the following circuit:

Below is the I-versus-V curve with 3 different values for the input voltage. The center, positive peak of the input, and the negative peak of the input are shown. The input amplitude is small.

In the above figure, the amplitude of the 1 GHz signal causes small variations in the slope of the tangent of the nonlinear curve as the input signal is varied in amplitude. The slope is the conductance. Because the circuit has a nonlinear device, the conductance of the device varies slightly from timepoint to timepoint in the pss analysis. Because the input signal at 1.0GHz is small, the variation in conductance from timepoint to timepoint is small. The slope changes just a little bit as the input signal is moved over its range.

Now imagine that the 1GHz signal is 10dB larger. A 10dB change is a linear factor of about 3.16. The larger amplitude of the input causes a larger portion of the nonlinear device curves to be traversed, and thus the variation in conductance becomes larger. If a second input at

1.1GHz were applied to the circuit, this mixes with the fundamental and the harmonics of 1GHz and more strongly mixes up and down in frequency. This is illustrated below.

Therefore, the variation in the conductance of the devices from timepoint to timepoint when shooting is selected in pss, or the harmonic content if harmonic balance is selected in pss contains information about how strongly a circuit translates frequencies, and this is used by the pac small-signal analysis to calculate the small-signal conversion gain.

Overview of Simulation Capabilities

Pac has a choice called specialized analysis which is located near the bottom of the *Choosing Analyses* form. This sets the mode of the measurement to be either averaged over the whole pss waveform or at a specific timepoint in the pss waveform. Many different types of analyses can be performed, like conversion gain, distortion contribution, and instantaneous gain measurements. Below is a description of the capabilities for each specialized analysis setting.

Also note that in this section, just the concepts are discussed without any mention of how to do the setup in the Analog Design Environment (ADE). To see the ADE setup, see ADE Implementation on page 724.

Specialized Analysis=None

Down Conversion: Input Near the First harmonic

At the bottom of the pac *Choosing Analyses* form, there is a selection called *Specialized Analyses*. The default is *None*, and this is for measuring the conversion gain that is the average conversion gain with the large signal applied to the circuit (usually the LO or the sample clock) applied to the circuit. The mixing products that are produced are called sidebands. The sideband number is the harmonic number that is being mixed with to provide the output. A positive number means that the output mixing product is higher in frequency than the input frequency specified in the *Choosing Analyses* form. A negative number means that the output is lower in frequency than the input frequency. A diagram is shown below.

We tend to think of positive frequencies. The default is to reflect negative frequencies to positive frequencies.

Again, in pac, a single input frequency is applied at a time, and then the output mixing products are calculated through the highest harmonic of the periodic steady-state (pss) simulation specified by the *Maximum sideband* parameter. *Maximum sideband* defaults to seven for shooting and the total number of harmonics specified in the pss analysis for hb. Usually *Maximum sideband* is set to calculate the desired mixing products. This can be

accomplished by setting *Maximum sideband* or by using the select from range capability in the pac *Choosing Analyses* form.

Up-conversion: Input Near Zero Frequency

In a similar manner, up-conversion is shown below.

Frequency Sweep

This analysis is like AC, where the input frequency is swept. The input frequency range is specified in the pac *Choosing Analyses* form.

In addition to the frequency, there is a selection called *Sweep Type*. The choices are *relative* and *absolute*. When *absolute* is selected, the frequency range is used directly with no frequency conversion. When *relative* is selected, a *Relative Harmonic* field appears. The frequency sweep is shifted by the specified frequency of the harmonic in the pss analysis. For example, assume that the pss is at 1GHz, and a log sweep is desired above the first harmonic of the pss analysis. In this case, you need to select *relative* sweep, and specify 1 in the *Relative Harmonic* field. Next, type in 1K to 100M for the frequency range with a log sweep and 3 to 5 points per decade.

The sweep will start at 1G + 1K, and sweep to 1G + 100M using log spacing from 1K to 100M. This is illustrated below.

To use a log sweep below the first harmonic, use the same frequency range, and type in -1 as the relative harmonic. This is illustrated below.

Maximum Sideband

Sideband is the name of the PAC output mixing product. The sideband number is the pss harmonic number that is mixed with. If the sideband number is negative, the output frequency is mixed down in frequency from the input. If the sideband is positive, the output is mixed up in frequency.

Conceptually, an infinite number of mixing products are produced when a single input is applied to a nonlinear system. From the practical point of view, usually only a small number of mixing products need to be measured.

There is a property in the *Choosing Analyses* form called *Maximum sideband* that is used to define how many mixing products should be calculated. If *Maximum sideband* is zero, the pac analysis will only contain the frequency that appears without mixing or aliasing of any type. This is not equivalent to a linear AC analysis because pac takes into account the instantaneously varying nature of the LO or sample clock signal because of the large signal being present in the pss analysis.

When *Maximum sideband* is one, the outputs without frequency translation and the outputs that mix or alias with the first harmonic of the pss analysis are present in the pac analysis. When *Maximum sideband* is 10, the pac analysis includes mixing through the 10th harmonic of the pss analysis. When shooting is used for the pss analysis, the number of sidebands can be any number up to four times the number of harmonics specified in the pss analysis. When harmonic balance is selected, up to the number of harmonics in the pss analysis can be selected.

Setting Harmonics and Sidebands

Shooting

The pss analysis, when shooting is selected, has a minimum of 200 timepoints, so it inherently has frequency domain content through the 100th harmonic of the pss beat frequency. Maximum harmonic is a pss post-processing property that specifies how many harmonics to calculate in the Fourier transform of the waveform in pss. If the number of harmonics is zero, it just means that no frequency domain information should be calculated in the simulation. If the number of harmonics is 10 or less, there is no effect on the PSS waveform. When maximum harmonics gets larger than 10, the minimum number of timepoints in the pss analysis is increased. 20 timepoints at the frequency of the highest harmonic are forced. Thus, if maximum harmonics is raised, the pss waveform gets more accurate at the cost of longer runtimes and more memory required to run the pss simulation because more timepoints are forced in the pss simulation.

In order for the pac analysis to correctly measure conversion gain, the wave shape at the highest mixing harmonic needs to be accurate, so with the default settings in shooting pss, up to 40 sidebands can be accurately calculated in the pac analysis. In most cases, only a small number of sidebands needs to be calculated, and so with shooting, you can count on the pac result to be accurate.

If you need more than 40 sidebands when you are using shooting pss, you need to set the *maxacfreq* option in the pss analysis to the pss beat frequency times the number of sidebands requested in the pac analysis, or alternatively, set the number of pss harmonics to the number of sidebands divided by four. Either method works to set the maximum timestep in the pss analysis as required for the pac analysis to be accurate.

Harmonic Balance

When *Harmonic Balance* is selected in the pss analysis, only the harmonics specified in the pss *Choosing Analyses* form exist in the solution. In the pac analysis, because only the harmonics specified in the pss-hb *Choosing Analyses* form exist, the maximum sideband term can only be the number of harmonics specified in the pss analysis or less. Usually when *Harmonic Balance* is selected in the pss analysis, the *Maximum sideband* field in pac usually should be left blank which calculates all the output mixing products.

Note that with harmonic balance, the number of harmonics (and oversample factor if you have non-sinusoidal waveforms) in the pss analysis needs to be determined. Start with a number of harmonics in the pss based on an estimate of how many harmonics the circuit actually produces with your input waveform, and run the pss and pac analyses. Remember that for non-sinusoidal waveforms, oversample factor also needs to be set greater than one. Run the analysis and do the desired measurement. Now increase the number of harmonics by roughly 50% and run the simulation again. If the answer does not change, then you had enough harmonics originally, and you might actually be able to decrease the number of harmonics from the original setting. If you have non-sinusoidal waveforms, increase the oversample factor before increasing the number of harmonics to find the minimum value for both oversample factor and harmonics.

Modulated

Modulated pac is typically used for measuring AM to PM measurements in driven circuits and oscillators. In this case, the input is usually the power supply ripple at low frequency, and the desired output is near the first harmonic of the output. Modulated is an averaged measurement over the whole pss waveform (shooting) or all the harmonics (harmonic balance). Pac apples a modulated input signal to the circuit, and is able to measure a modulated output signal from the circuit. The types of modulation are Single Side Band (SSB), AM, and PM.

A diagram is shown below for SSB input.

Note that although the input occurs at one frequency, the AM output can be measured directly from the single input.

When AM is applied at the input, the following figure is shown.

The difference is that in addition to the input at low positive frequency, another input (to form an AM signal) is applied at a low negative input. This second input adds to the output signal near the first harmonic.

For PM, the negative input amplitude is negative, as shown below.

By running all three inputs, the SSB/AM/PM input to SSB/AM/PM output can be calculated. This is a setup choice in the *Choosing Analyses* form.

In the *Choosing Analyses* form, the input frequency range, and the output harmonic must also be specified.

All the plot functions are implemented in the *Direct Plot Form*.

Rapid IP2/IP3 (AC and PAC)

Rapid IP2 and Rapid IP3 have been added to the ac and pac analyses. First, a provision has been made to allow two input frequencies. Second, the ac and pac analyses have been enhanced so that harmonics of those signals and the intermodulation distortion can be calculated. Doing this in a small-signal analysis is much faster than doing this in a large-signal analysis because the large-signal effects are ignored.

The Rapid IP2 and Rapid IP3 that is available in the ac *Choosing Analyses* form allows the calculation of IP2 and IP3 for amplifiers. In pac, Rapid IP3 and Rapid IP2 are provided for circuits that translate frequencies, whether that translation is an up or down conversion. In this case, the signal that is causing the frequency translation (the LO or the sample clock) is applied in the pss analysis first in order to characterize the nonlinearity of the system so that

the correct IP2/IP3 measurement can be made. Note that pac and ac are small-signal analyses, so the limitation is that the measurement is a small-signal IP3 or IP2. If you need a large-signal IP2 or IP3 measurement, use a large-signal analysis like hb or qpss to make the measurement. Note that when you are using Rapid IP2 or Rapid IP3 in the ac analysis, either a SpectreRF license is required, or an additional mmsim token is required to run the analysis.

Make sure that the input level for the IP3 or IP2 measurement is in the small-signal range. Rapid IP2 and IP3 have exceptionally small numerical noise floors, therefore, it is recommended that you specify a level that is too small, rather than too large. Rapid IP2 and Rapid IP3 work by evaluating the device equations for nonlinearity. Do not specify one volt as the input amplitude because this is generally in a large-signal range, and usually causes the devices to reach points where the device currents are deliberately linearized, thus making the IP2/3 measurement more linear than it actually is.

The plot functions for Rapid IP2 and Rapid IP3 are implemented in the *Direct Plot Form*.

1. Select *Results - Direct Plot - Main Form*. In the Direct Plot Form.
2. Select *ac* or *pac* from the *Analysis* section.
3. Select *Rapid IP2* or *Rapid IP3* (depending on which was set up) from the *Function* section, and click *Plot*.

The X and Y points are shown for the intercept.

The X value is input-referred IP2/IP3, and the Y is output-referred IP2/IP3.

Note that if you have a switched-FET mixer and you are using the BSIM3 or BSIM4 models, due to a model limitation, the IP3 calculation from rapid IP2/IP3 will be incorrect. In this case, the PSP model is recommended. If a PSP model is not available, use hb or qpss with an input amplitude that is as large as possible while still being in the small-signal range for your circuit or switch to a 1dB gain compression measurement, The compression measurement is much more accurate than the IP3 measurement for switched FET mixers.

Compression and IP2 distortion summary

The distortion summaries measure distortion for an amplifier in ac or in circuits that have frequency translation in pac. The idea is to find which devices in a signal path contribute relatively more or less to the distortion. Since IP3 and compression are related mathematically, the compression distortion summary provides information about which devices contribute the most amount of third order intermodulation distortion. The IP2 distortion summary measures which devices contribute to the second order distortion. Note that when you are using the distortion summaries in the ac analysis for amplifiers, either a SpectreRF license is required, or an additional mmsim token is required to run the analysis.

Both distortion summaries use the *PAC Magnitude* property in the input port. Make sure that the *PAC Magnitude* (in volts or dBm) is actually in the small-signal range, (At least 10 dB smaller than the 1dB gain compression point) or the results will not be accurate. This is because the pac results do not take into account the large-signal limits like slew-rate or power supply voltage and the device models deviate from the ideal equations at very large amplitude by just continuing a linear slope above a certain amplitude. That amplitude depends on the device model parameters and is likely to be reached with a 1 volt input amplitude. The compression distortion summaries have an extremely low numerical noise floor, so it is better to set an input amplitude that is too small, than to set it too large.

The distortion summaries provide a list output, not a plot output. Accordingly, instead of selecting *Results - Direct Plot - Main Form*, select *Results - Print - PAC Distortion Summary*.

For the compression distortion, you get the actual gain with the distortion of each device by itself divided by the ideal small-signal gain provided by the pac analysis. In addition, the amplitude of the first three harmonics of the linear output frequency is calculated.

For the IM2 distortion summary, you get the total IM2 output amplitude that results from having only a single device in the circuit.

Sampled

Sampled is provided in order to calculate the instantaneous conversion gain at a voltage threshold crossing in the pss waveform or at specific timepoints in the pss simulation. Note that this analysis is fundamentally different than the analyses discussed earlier. All the previous analyses measured the average conversion gain that is produced when all the harmonics are taken into account (hb) or all the timepoints are taken into account (shooting). In sampled, an instantaneous measurement at a single point in time (or a list of times) of the LO or clock waveform is made.

Note that when harmonic balance is selected, sampled pac is the same as in shooting, but the time-domain waveform is the ifft of the harmonics calculated by the harmonic balance analysis. This is useful for measuring the conversion gain from the power supply to the oscillator output at a threshold crossing, or at specific times in the LO waveform to the output of a mixer.

When shooting is selected, a minimum of 200 timepoints exist in the solution waveform, and shooting automatically makes the timestep smaller in areas of rapid change, thus, there is likely a datapoint in the pss analysis that is very near the voltage level you specify as a threshold. In harmonic balance, the timepoints in the ifft depend on the number of harmonics and the oversample factor. Set a large enough number of harmonics and/or set a large

enough oversample factor to make sure that there is a timepoint in the ifft that is very near the actual threshold value you specify in the *Choosing Analyses* form.

ADE Implementation

Note: In general, the examples in this section show shooting pss. The examples work for harmonic balance pss unless otherwise noted. This is done mostly to show shooting. Harmonic balance has its own section in Chapter 3, “Frequency Domain Analyses: Harmonic Balance.”.

The settings in the PAC form can be slightly different for shooting and harmonic balance, and the method for both is covered in each section.

Circuit, Input port Setting, and ADE Setup for all the Examples in This Section

Consider the following circuit:

This above circuit is an LNA and a mixer for a 2.4GHz application that is implemented in CMOS. The input source (a port in Cadence terminology) has the values for its frequency and amplitude set to variable names. The PAC magnitude property is set to 1 volt in order to allow easy conversion gain measurements. The reason for the variables is that the frequency and amplitude can be changed in the ADE environment without changing the schematic.

The *Edit Object Properties* form for the input port is shown below.

The ADE has the variables set to appropriate values, as shown below.

Note that the variables fb1 and fb2 (which set the input frequency of the input port) are set to zero, which effectively disables the RF input. Only the LO signal is applied to the circuit for the pss analysis.

Using PAC Analysis for Conversion Gain Measurement

Set up the pss analysis with just the large signal that causes the frequency translation to occur, which is generally the LO signal or the sample clock. For more information, refer to Overview of Periodic Steady-State (pss) Analysis on page 505. Regarding whether to use shooting or harmonic balance, pick the analysis that produces the smallest runtime. In

general, shooting is recommended for systems that have sharp edges, and harmonic balance is recommended for systems that are more sinusoidal.

Set up the pac *Choosing Analyses* form as follows:

1. Specify the input frequency range. In this case, the *Sweeptype* is set to *absolute*, which means that there is no frequency shift applied to the input frequency. If a log sweep above the first harmonic is desired, set the *Sweeptype* to *relative*, and specify one as the relative harmonic number. The input frequency range would be 1MHz to 100MHz to get

the same frequency range as in the example above. Relative shifts the frequency specified the harmonic number times the pss frequency.

2. For pac, the *Maximum sideband* field should be set to the highest harmonic of the pss analysis that mixing products are desired to be calculated. This is generally determined by the application. If you only need one or two output mixing product, choose *Select from range*, and highlight the sidebands you want to calculate. Setting *Maximum sideband* for pac is usually not critical as long as enough sidebands are calculated to measure the desired output mixing product (called a sideband). The *Direct Plot Form* allows the plotting of individual output mixing products as will be shown later. As a general recommendation, set the number of sidebands equal to the number of pss harmonics.

3. For a conversion gain measurement, set the *PAC Magnitude* to 1 volt on the *Edit Object Properties* form, as shown earlier. This is always volts peak.

4. Now run the simulation. In the Analog Design Environment (ADE), select *Simulation - Netlist and Run*.

The *Direct Plot Form* is displayed.

Mixing products are calculated for mixing the pac input frequency range with the harmonics calculated in the pss analysis. Harmonics of the pac input and harmonics of the output mixing products are not calculated because pac is a small-signal analysis. Because it is small-signal, the harmonics of the input and the harmonics of the mixing products are so small that they do not matter.

To measure the outputs you desire, place a marker at the desired output frequency of interest and read the conversion gain, as shown below.

To plot a single output sideband (mixing product), in the *Direct Plot Form*:

1. Select *sideband* instead of *spectrum*, and select the desired output mixing product form the list as shown below.

2. Select the output net in the schematic, or click the *Replot* button in the *Direct Plot Form*. The waveform tool displays the output, as shown below.

3. Position a marker by moving the cursor to the desired output frequency and type `m`. The marker appears, as shown in the figure above.

4. To position the marker at an exact frequency, place it as above, and then move your cursor over the intercept point on the trace. Click and hold the right mouse button, move to *Marker Properties*, and release the mouse button. The *Point Marker Properties* window appears, as shown below.

5. Type the frequency in the *Position* field, and click *OK* or *Apply*.

Sampled Circuits and Switched-capacitor Filters

For sampled circuits or for switched-capacitor circuits, always use shooting pss because the clocking waveforms have rapid transitions. Shooting captures these transitions in a natural and accurate way.

In pac, generally only the sideband with zero frequency translation is desired, so *Maximum sideband* can be set to zero. A warning is produced in the Spectre output window stating that the number of sidebands is lower than anticipated. This warning can be ignored.

Note that the average gain is determined by pac. In the sampling circuit, this is generally very near unity gain, but for the switched-capacitor circuit without a sample and hold, the gain is generally a bit less than half amplitude which is about -6dB. This is because the output waveform follows the input waveform for a little less than half the time, and the capacitors are reset by the switches the other half the time.

If you desire to see the aliasing at different harmonics of the clock, use pxf analysis. For example, imagine a sampling frequency of 1MHz. In pac, with the zero sideband only, the input and output frequencies are always the same. Therefore, if the input is at 1.1MHz in pac, the output is also at 1.1MHz. Generally, the term that is interesting is the term that aliases down from 1.1MHz to 100KHz which is not measured by the pac analysis (The input and output frequencies are the same in the zero sideband.) To see the aliasing at the different harmonics, use pxf with an output frequency of near zero to 500KHz, and specify *Maximum sideband* to the harmonic number desired. If the *Maximum sideband* is greater than 40,

either set the pss option *maxacfreq* to the number of sidebands times the pss beat frequency, or set the number of harmonics to the number of sidebands divided by four.

Commonly Used PAC Options

Freqaxis

freqaxis specifies whether you want to see the negative frequency axis or not.

The analysis calculates the output frequencies and amplitudes, so *out* and *absout* are reasonable choices. *out* displays the negative frequency axis. *absout* (absolute value of the output) displays positive output frequencies. This is the default.

If you select *in*, all the outputs at the different frequencies are plotted on the same input frequency range scale. This is not recommended. The default is *absout*.

Below is an example of *freqaxis* set to *out*. Note that the data is the same as before. It is just displayed with the negative frequency axis present. The negative frequencies have not been reflected up to the positive frequency domain.

Below is an example with the *freqaxis* option set to *in*. All the outputs at different frequencies are plotted on the input frequency scale. The frequency can be calculated by selecting the appropriate sideband from the list on the left side of the waveform tool, and then adding the sideband number times the pss beat frequency to the frequency displayed on the X axis.

Tolerance

Leave this option at the default value.

Pac uses an iterative solver to calculate the output amplitudes. Any iterative solver needs an error tolerance to specify when to stop iterating because the solution is accurate enough, and the tolerance option specifies that accuracy for pac. For shooting, the default tolerance is 1e-9. For driven circuits where HB is the pss engine, the default is 1e-6. For oscillators where hb is selected for the pss engine, the default is 1e-4.

Solver

The solver option is used only for driven circuits. The default solver is the turbo solver.

When hb is used as the engine in pss, leave this option at the default.

When shooting is used, sometimes when the pac input frequency is very close to the frequency of one of the harmonics in the pss, warning messages will appear in the pac output warning that the accuracy might not be good enough. If you see these messages, select the std solver, which has better ability to handle frequencies that are very close to a harmonic in the pss, but which takes longer to run than the turbo solver.

Hbprecond_solver

This option is only available only when harmonic balance is selected in PSS.

The basic solver is the only solver available in standard Spectre. Autoset is the default solver when APS is used. This solver is faster, but occasionally stagnates. When stagnation is detected, APS automatically switches to the basic solver, and prints a message in the Spectre output window.

Oscsolver

The oscsolver option is used only for oscillators. The default solver is the *turbo* solver.

When hb is used as the engine in pss, leave this option at the default.

When shooting is used, sometimes when the pac input frequency is very close to the frequency of one of the harmonics in the pss, warning messages appear in the pac output warning that the accuracy might not be good enough. If you see these messages, select the *std* solver, which has better ability to handle frequencies that are very close to a harmonic in the pss, but takes longer to run than the *turbo* solver.

For very large oscillator designs, the ira oscsolver may take less memory and run faster with no loss in accuracy, however, it is less robust for convergence compared to *turbo* or *std*.

Gear_order

Do not change this option.

Lnsolver

Leave this option at the default. Each pac sideband is internally calculated from a matrix. This matrix is solved using an iterative solver. Several different algorithms are provided for the iterative solver. *gmres* is the default because the accuracy of each iteration inherently increases. Convergence is generally good as well. The other solvers may require less memory, but they are less robust for convergence and may suffer from false convergence.

Considerable knowledge is needed to understand the differences between these methods. For more information, refer to the books on linear algebra theory.

gmres is the Generalized Minimum RESidual method.

qmr is the Quasi-Minimal Residual method.

bicgstab is the STABilized BI-Conjugate Gradient method.

resgmres is the REStarted Generalized Minimal Residual method.

Resgmrescycle

Leave this option at the default. For the resgmres linear solver, there are several different options.

Outputperiod

Do not use this option.

AdditionalParams

additionalParams is typically used for new features that are being beta tested.

For more information about the other options, type `spectre -h pac` at the command prompt in a Unix shell window

PAC With Multiple PSS Inputs

This is generally used when the conversion gain is desired with a large blocker signal present at the RF input. The basic strategy is to apply the signals that cause the nonlinearity to occur (the LO and the blocking signal) in the pss, and then use pac to measure the conversion gain when the blocker is present. In many cases, the blocker power is swept in order to measure the degradation in conversion gain when the blocker gets large.

Multiple input frequencies might be able to be run in the PSS, depending on the ratio between the highest input frequency divided by the pss beat frequency. If this ratio is about 25 or less, then pss can be used for the simulation. If the ratio is greater than 25, consider using harmonic balance (hb) or qpss to run the simulation because it is very likely to take less time to run.

To set up the second input, the variable fb1 was set to 2.5G in the ADE window, as shown

below. This variable sets the frequency of the first source in the port component that supplies the RF signal. In this case, the LO is at 2.4GHz and the RF is at 2.5GHz.

The pss was set up with 175 total harmonics of 100MHz to allow harmonics through the seventh harmonic of the LO signal at 2.5GHz to be calculated. For more information on pss, Overview of Periodic Steady-State (pss) Analysis on page 505.

Choosing Analyses – Virtuoso® Analog Design Environn
Analysis
tran
dc
ac
noise
xf
sens
dcmatch
stb
pz
sp
envlp
pss
pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
hbsp
Periodic Steady State Analysis
Engine
Shooting
Harmonic Balance
Fundamental Tones
Name Expr Value Signal SrcId
1 fb1 fb1 2.5G Large PORT1
3 flo flo 2.4G Large
4 flo flo 2.4G Large
Large
Clear/Add
Delete
Update From Hierarchy
Beat Frequency
Beat Period
100M
Auto Calculate
Output harmonics
Number of harmonics
175
Accuracy Defaults (errpreset)
conservative
moderate
liberal
Additional Time for Stabilization (tstab)
Save Initial Transient Results (saveinit)
no
yes
Oscillator
Sweep
New Initial Value For Each Point (restart)
no
yes
Loadpull
Enabled
Options...
OK
Cancel
Defaults
Apply
Help

Remember that the frequency range is the input frequency range. In this example, the frequencies just above the first harmonic of the LO are swept.

The pac form is set up to calculate the mixing products for all the harmonics that are calculated in the pss analysis. When shooting is used in the pss analysis, up to four times as many sidebands can be specified in pac as harmonics are specified in pss. When harmonic balance is used in pss, *Maximum sideband* can be set up to the number of harmonics in the pss analysis. Sidebands are the mixing products that are produced when the input signal mixes with the harmonics in the pss analysis.

Now run the analysis. The spectral plot below from the pss analysis shows that because there were more input frequencies in the circuit, many more harmonics are produced by the circuit, which is as expected.

As shown below, because there are many more pss harmonics to mix with, many more mixing products (sidebands) are produced by pac.

Virtuoso (R) Visualization & Analysis XL
File Edit View Graph Axis Trace Marker Measurements Tools Window Browser Help
cadence
Layout Auto
Workspace Classic
RF_APS RxChain_LNA_Mixer sch...
RF_APS RxChain_LNA_Mixer sch...
v/BB_Out; pac dB20(V)
Name
Vis harmo
.../BB_Out; pac dB20(V) 151
.../BB_Out; pac dB20(V) 152
.../BB_Out; pac dB20(V) 153
.../BB_Out; pac dB20(V) 154
.../BB_Out; pac dB20(V) 155
.../BB_Out; pac dB20(V) 156
.../BB_Out; pac dB20(V) 157
.../BB_Out; pac dB20(V) 158
.../BB_Out; pac dB20(V) 159
.../BB_Out; pac dB20(V) 160
.../BB_Out; pac dB20(V) 161
.../BB_Out; pac dB20(V) 162
.../BB_Out; pac dB20(V) 163
.../BB_Out; pac dB20(V) 164
.../BB_Out; pac dB20(V) 165
.../BB_Out; pac dB20(V) 166
.../BB_Out; pac dB20(V) 167
.../BB_Out; pac dB20(V) 168
.../BB_Out; pac dB20(V) 169
.../BB_Out; pac dB20(V) 170
.../BB_Out; pac dB20(V) 171
.../BB_Out; pac dB20(V) 172
.../BB_Out; pac dB20(V) 173
.../BB_Out; pac dB20(V) 174
.../BB_Out; pac dB20(V) 175
25.0
0.0
-25.0
-50.0
-75.0
-100.0
-125.0
-150.0
V(dB)
0.0
5.0
10.0
15.0
20.0
freq (GHz)
mouse L:
M:
R:
12(26)
no error

The next figure shows a zoomed-in area near zero frequency. Only the mixing products larger than -100 dB are shown.

Note that the index in the pac analysis is the frequency shift from the input frequency to the output frequency in multiples of the pss beat frequency. The output highlighted above mixes with the 24th harmonic of 100MHz, which is the LO signal in the circuit. This is the main output mixing product when RF mixes with LO.

Note that the RF input in the pac analysis at 2.4001G to 2.45G can also mix with 2.5G (The additional RF signal.) When the RF input is at 2.4001G the IF output is 99.9MHz when it mixes with the other RF tone at 2.5GHz. As the input frequency in the pac sweeps up, the output frequency sweeps down to 50MHz. Since it is mixing with 2.5GHz, the output frequency is downshifted by -25*100MHz, thus the sideband number shown in the legend at the left is -25. This is shown below.

Rapid IP3/IP2

Rapid IP3 and Rapid IP2 are available in the ac and the pac *Choosing Analyses* forms. Rapid IP3 in the ac form is for measuring IP3 of amplifiers where there is no frequency translation, and Rapid IP3 in the pac form allows the measurement of IP3 where frequency translations are present. Rapid IP3 and IP2 measure small-signal IP3 and IP2.

The example below is for an LNA-mixer circuit.

1. First set up the pss form with just the LO applied, as shown below. For more information, see Overview of Periodic Steady-State (pss) Analysis on page 505. Both the shooting and harmonic balance pss engines are supported in pac.

Choosing Analyses – Virtuoso® Analog Design Environn
Analysis
tran
dc
ac
noise
xf
sens
dcmatch
stb
pz
sp
envlp
pss
pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
hbsp
Periodic Steady State Analysis
Engine
Shooting
Harmonic Balance
Fundamental Tones
Name Expr Value Signal SrcId
3 flo flo 2.4G Large
4 flo flo 2.4G Large
Large
Clear/Add
Delete
Update From Hierarchy
Beat Frequency
Beat Period
2.4G
Auto Calculate
Output harmonics
Number of harmonics
10
Accuracy Defaults (errpreset)
conservative
moderate
liberal
Additional Time for Stabilization (tstab)
Save Initial Transient Results (saveinit)
no
yes
Oscillator
Sweep
New Initial Value For Each Point (restart)
no
yes
Loadpull
Enabled
Options...
OK
Cancel
Defaults
Apply
Help

In the example below, the LO is at 2.4GHz, and the RF inputs are at 2.401GHz and 2.4012GHz. This makes the main mixing products at the output of the mixer at 1MHz and 1.2MHz. The third order intermods are at 800KHz and 1.4MHz. Because the frequencies are similar, either the 1 or 1.2 MHz products can be set as the linear frequency, and either the 800KHz or 1.4MHz term can be used for the third order frequency. Generally, the smallest amplitude linear term and the largest amplitude third order term are used for the IP3 measurement. If IP2 was desired, the second order product is at the difference frequency between the 2 RF inputs (200KHz in this example). The first order terms are the same as IP3.

2. Select the *pac* analysis. At the bottom of the form in the *Specialized Analysis* section, select *Rapid IP3* or *Rapid IP2*, depending on what you want to measure. The entries for the forms are identical, except for the frequency of the intermodulation product. For switched FET mixers, make sure that the psp model is used. Incorrect results are produced for BSIM3 and BSIM4 models due to a limitation in the model. If you need to measure IP3 with a BSIM3 or 4 model, use hb analysis with as large an amplitude as possible while maintaining the amplitude in the small-signal region.

3. Define the input port, input frequencies, input power, the frequencies of the linear and second or third order products, and the output node. An example for Rapid IP3 is shown below. Make sure that the input power is in the small-signal range. (At least 10dB smaller than the compression point)

4. Select the output net in the circuit.

   Note that for amplifiers, the pss analysis does not need to be set up. Just use the ac *Choosing Analyses* form which has the same appearance as the pac *Choosing Analyses* form shown below. The linear output frequency can be either of the two input

frequencies. The third-order frequency can be the frequency spacing above or below the RF input frequencies.

5. Leave the *Maximum Non-linear Harmonics* field blank. The default value is appropriate for the Rapid IP2/IP3 calculation.

6. Run the simulation. When the simulation is complete, select *Results - Direct Plot - Main Form*.

7. Select *pac* results.

8. Select *Rapid IP3*.

   Rapid IP2 is shown if an IP2 analysis is run.

9. Click *Plot*.

   The waveform is displayed. The marker shown at the intercept point shows the input and output-referred IP3 or IP2.

Compression Distortion Summary

The compression distortion summary is provided in order to get a relative measure of which components in a gain path contribute more or less compression. Because compression and IP3 are related mathematically, the compression distortion provides information about which components contribute to IP3.

Note that the Compression Distortion Summary is available in the ac and the pac *Choosing Analyses* forms. Compression Distortion Summary in the ac form is for measuring which components add compression in amplifiers where there is no frequency translation, and Compression Distortion Summary in the pac form allows the measuring of components that contribute to compression where frequency translations are present in the circuit.

Note that for sampling circuits or for switched-capacitor filters, the sample clock(s) are applied in the pss, and the small-signal input Is applied in the pac analysis.

The *Contributor Instances* field is provided so that you can select which components in the circuit should be considered in the analysis. If the *Contributor Instances* field is left blank, all the nonlinearities in your circuit are run.

The following example is for an LNA-Mixer circuit.

1. First set up the pss analysis with just the LO or sample clock applied, as shown below. For more information, see Overview of Periodic Steady-State (pss) Analysis on page 505.

2. Now select pac analysis.

3. Select *Compression Distortion Summary* from the *Specialized Analyses* drop-down list.

   The following notice appears to remind you to set the PAC amplitude parameter in the source in the schematic to a value somewhere in the small-signal range. This is generally at least 10dB smaller than the 1dB gain compression point.

4. If you leave the *Contributor Instances* field blank, all the nonlinear devices in your circuit will be run. If you want a subset, click *Select*, and select the instances you want on the schematic.
5. Specify the input frequency in the *Frequency of Input Source* field.
6. Specify the output frequency in the *Frequency of Linear Output Signal* field.
7. Leave the *Maximum Non-Linear Harmonics* field blank. The default value is *optimum* for the compression distortion measurement.
8. Select the output node in the circuit. When the *Out-* field is left blank, ADE automatically assigns this to the global ground node.

The completed form appears similar to the one shown below.

9. In the *Edit Object Properties* form, make sure that the *PAC magnitude (dBm)* is set in the small-signal range. In this example, it is set to -30Bm.

10. If you make changes to your schematic that you want for the distortion summary, but you do not want to save them to the circuit file, make the desired changes and then select *Check - Current Cellview* in the schematic window. Later, if you want to save them, just

select *Check* and *Save*. If you do not want them, quit from the schematic window, and do not save the changes.

ADE will netlist from the checked view, so it will get the changes.

11. When you are done, run the simulation.

12. When the simulation completes, select *Results - Print - PAC Distortion Summary*.

The summary is displayed below.

Results Display Window

Window Expressions Info Help cadence

PAC Compression Distortion Summary

Results in pac_distortion

Instance	Distortion(dB)	Nonlinear Mag at 1st 2nd & 3rd harm of linear freq (V) freq=1e+06	freq=2e+06	freq=3e+06
Total	-395.525m	4.20876m	8.78522n	1.03786m
I33.I1.I0.NM1	-49.3204m	869.753u	1.79933m	166.338u
I33.I1.I0.NM0	-49.3204m	869.753u	1.79934m	166.338u
I33.I1.I1.NM0	-54.8309m	775.099u	1.86679m	173.351u
I33.I1.I1.NM1	-54.8309m	775.099u	1.86679m	173.352u
I33.I1.I0.PM4	-14.2716m	155.121u	265.572u	45.5523u
I33.I1.I0.PM3	-14.2708m	155.113u	265.576u	45.5468u
I33.I1.I0.PM2	-14.2681m	155.086u	265.53u	45.9909u
I33.I1.I0.PM5	-14.2673m	155.077u	265.533u	45.9919u
I33.I1.I1.PM5	3.63289m	146.6u	252.135u	44.5893u
I33.I1.I1.PM2	3.63251m	146.598u	252.135u	44.5895u
I33.I1.I1.PM3	3.62865m	146.588u	252.106u	44.5306u
I33.I1.I1.PM4	3.62825m	146.585u	252.106u	44.5307u
I33.I0.NM9	-6.98721m	125.694u	32.7988u	1.8931u
I33.I0.NM11	-6.98721m	125.694u	32.7988u	1.89321u
I33.I1.I0.PM0	-3.59119m	91.8659u	472.13u	28.1052u
I33.I1.I0.PM1	-3.59119m	91.8659u	472.13u	28.1051u
I33.I1.I1.PM0	-6.85581m	84.1694u	477.265u	28.411u
I33.I1.I1.PM1	-6.85581m	84.1693u	477.266u	28.4116u
I33.I0.NM10	-1.24714m	15.7121u	6.89581u	67.0635n
I33.I0.NM12	-1.24714m	15.712u	6.89581u	67.0525n
I33.I1.I0.I2.NM3	-2.12409u	36.5601n	2.41982p	66.4574n
I33.I1.I0.I2.NM33	-2.12396u	36.5592n	2.35605p	66.2714n
I33.I1.I1.I2.NM3	-2.67336u	35.7247n	1.05165p	66.3585n
I33.I1.I1.I2.NM33	-2.67324u	35.7238n	1.03473p	66.4086n
I33.I1.I0.I2.PM11	179.98n	2.06775n	24.347f	4.47264n
I33.I1.I0.I2.PM12	179.979n	2.06774n	23.0481f	4.47092n
I33.I1.I1.I2.PM11	-77.5038n	2.06497n	84.7601f	4.4728n
I33.I1.I1.I2.PM12	-77.5039n	2.06496n	80.5783f	4.47234n
I33.I1.I0.I2.NM1	160.321p	44.4538p	755.051f	7.44491n
I33.I1.I0.I2.NM2	159.498p	44.4433p	827.882f	7.4449n
I33.I1.I1.I2.NM2	3.12365n	34.3734p	544.994f	7.44659n
I33.I1.I1.I2.NM1	3.12358n	34.3727p	537.549f	7.4466n
I33.I1.I0.I2.NM32	-272.7p	2.96525p	86.9421f	173.115p
I33.I1.I0.I2.NM0	-272.7p	2.96525p	87.0916f	173.116p
I33.I1.I1.I2.NM32	32.7312p	1.93102p	116.033f	173.067p
I33.I1.I1.I2.NM0	32.7312p	1.93101p	116.003f	173.067p

50

- The first column is the instance name in the schematic.

- The second column is the gain with the nonlinearity of that single device divided by the ideal gain from the ac or pac analysis.

  Gain compression is shown as a negative number. Gain expansion is shown as a positive number.

- The third, fourth, and fifth columns are the amplitudes of the first, second, and third harmonic of the linear output frequency with just the nonlinearity of that single device taken into account.

Sampled Circuits and Switched-capacitor Filters

For sampled circuits or for switched-capacitor circuits, always use shooting pss because the clocking waveforms have rapid transitions. Shooting captures these transitions in a natural and accurate way.

In pac, set up the compression distortion form just as you would for the mixer example above. Note that the linear output frequency is the input frequency. Otherwise, the setup is exactly the same as the setup shown above.

IM2 Distortion Summary

The steps for the IM2 distortion summary are very similar to the Compression Distortion Summary in the previous section. The differences are noted below.

- The IM2 distortion summary provides the output voltage (magnitude, real, and imaginary parts) of the second order distortion signal for each component in the schematic by itself.
- The IM2 distortion summary does not provide a selection mechanism for the devices in the circuit. All the devices are run in the IM2 distortion summary.
- The output format is slightly different. The first column is the instance name in the schematic. The second column is the magnitude of the IM2 product that is produced by

that single component in volts. The third column has the real and imaginary parts of the voltage. A sample is shown below.

```
PAC IM2 Distortion Summary

Results in        pac_im2distortion

Instance          Distortion(V)[Magnitude Complex]
Total             56.8138n    complex(56.8132n, 279.356p)
/I33/I1/I0/PM4    1.45004m    complex(1.45003m, -2.96617u)
/I33/I1/I0/PM3    1.45003m    complex(1.45003m, -2.96618u)
/I33/I1/I0/PM2    1.45002m    complex(-1.45001m, 2.96629u)
/I33/I1/I0/PM5    1.45001m    complex(-1.45001m, 2.96628u)
/I33/I1/I1/PM4    1.43768m    complex(1.43768m, -2.95021u)
/I33/I1/I1/PM3    1.43768m    complex(1.43767m, -2.95019u)
/I33/I1/I1/PM2    1.43767m    complex(-1.43767m, 2.95025u)
/I33/I1/I1/PM5    1.43767m    complex(-1.43767m, 2.95024u)
/I33/I1/I1/I2/NM3    144.198n    complex(-97.8239n, -105.941n)
/I33/I1/I1/I2/NM33    144.197n    complex(97.8234n, 105.94n)
/I33/I1/I0/I2/NM3    140.951n    complex(-96.7777n, -102.476n)
/I33/I1/I0/I2/NM33    140.951n    complex(96.7775n, 102.476n)
/I33/I1/I1/I2/PM11    10.0589n    complex(-7.44012n, -6.7695n)
/I33/I1/I1/I2/PM12    10.0589n    complex(7.44012n, 6.76949n)
/I33/I1/I0/I2/PM11    10.0337n    complex(-7.6139n, -6.53482n)
/I33/I1/I0/I2/PM12    10.0337n    complex(7.6139n, 6.53481n)
/I33/I1/I1/PM1    5.07326n    complex(-5.07326n, 4.89652p)
/I33/I1/I0/PM0    4.36043n    complex(4.36039n, 18.022p)
/I33/I1/I1/PM0    3.67355n    complex(3.67349n, -21.7191p)
/I33/I1/I1/NM0    3.00179n    complex(-3.00172n, 20.0364p)
/I33/I1/I1/NM1    2.61043n    complex(2.61009n, -41.8165p)
/I33/I1/I0/NM0    1.95443n    complex(-1.95432n, -20.0203p)
/I33/I1/I0/NM1    1.64469n    complex(1.64469n, 387.776f)
/I33/I1/I0/PM1    753.377p    complex(753.304p, 10.4897p)
/I33/I0/NM9       266.846p    complex(-266.8p, -4.96938p)
/I33/I1/I1/I2/NM2    226.574p    complex(-212.697p, -78.0755p)
/I33/I1/I1/I2/NM1    226.573p    complex(212.694p, 78.079p)
/I33/I0/NM12      165.221p    complex(165.216p, 1.39871p)
/I33/I0/NM10      126.074p    complex(125.993p, -4.53295p)
```

Sampled Circuits and Switched-capacitor Filters

For sampled circuits or for switched-capacitor circuits, always use shooting pss because the clocking waveforms have rapid transitions. Shooting captures these transitions in a natural and accurate way.

In pac, set up the IP2 distortion form just as you would for the mixer example above. Note that the linear output frequency is one of the input frequencies and the intermodulation frequency is the frequency spacing. Otherwise, the setup is exactly the same as the setup shown above.

Modulated PAC Analysis

Modulated pac analysis is usually applied to oscillators to measure AM-to-PM conversion from low-frequencies on the power supply (ripple) to the output near the first harmonic. It can also be applied to digital circuits for the same purpose. Although the shape of the transfer function is different, the steps are the same for both applications.

In the circuit, make sure that any PAC magnitude terms are removed on the input source (if there is one), and set the PAC Magnitude to 1 volt in the source that is used as the power supply. If you do not want to save the change permanently, just select *Check - Current View* in the schematic.

This example is for an oscillator. The circuit is shown below:

In the ADE, set values for the power supply voltage and for the tuning voltage.

Set up the pss analysis. For more information on pss, see Overview of Periodic Steady-State (pss) Analysis on page 505.

In the pac *Choosing Analyses* form, first select *Modulated* in the *Specialized Analyses* section, as shown below.

Note that selecting *Modulated* causes the *Sweep type* to be set to *relative*. Since the gain from power supply ripple to the output is desired, the relative harmonic number is zero, and then the desired frequency range is entered. If the shooting engine is used, maximum sideband can be up to 4 times the number of harmonics in the pss analysis harmonics as shown above. Setting the *Sweeptype* to *relative* allows a frequency shift in multiples of the pss beat frequency to be applied to the input frequency. In this example, when the relative harmonic is zero, the input frequency range is not shifted at all in frequency.

Maximum sideband needs to be set larger than is needed for measuring the output frequency. In this case, the input signal is the ripple at low frequency, and the output is near the first harmonic. Because of this, *Maximum sideband* needs to be one or larger. In this case, the setting of *Maximum sideband* is set to the same number as the maximum harmonics setting in pss.

Input Type is set to *SSB/AM/PM* so that all the combinations of SSB, AM, and PM to SSB, AM, or PM can be calculated.

Next, the output frequency ranges to be calculated can be set. Note that a list of harmonics can be entered. The easiest way is to click the *Choose* button to the right of the field, and select the desired frequencies from a list in the *Choose Harmonic* window, as shown below.

The same process is used to set the input frequency, but only one frequency range is allowed.

Once the analysis has been set up, run the simulation. When the simulation completes, select *Results - Direct Plot - Main Form*.

The input defaults to *AM*. Select *PM* for the output. Since many output frequencies were calculated, you need to select the one you want to plot. In this example, the output just above the first harmonic is selected. Once this is done, select a net in the schematic.

The output is displayed. To make a measurement at a specific frequency, position a marker. The measurement is in radians per volt of input on the supply that has PAC magnitude set.

Sampled PAC Analysis

Pac sampled is for measuring the conversion gain at a specific instantaneous time in the pss waveform. The first step is to determine an appropriate threshold for the measurement by setting up the pss analysis, as shown below. Shooting is shown below, but this is also

available in harmonic balance. For harmonic balance, the waveform is an ifft of the harmonics.

Now run the analysis. When the analysis completes, select *Results - Direct Plot - Main Form* in the ADE and plot the time-domain waveform on the output net, as shown below.

The waveform is shown below.

Note that 2.5 volts is a value near the middle of the waveform. This is chosen as the threshold for the calculation. Pick a threshold that is appropriate for your system.

Now set up the pac *Choosing Analyses* form.

1. At the top of the form, set up the input frequency range like normal.
2. At the bottom of the form, select *Sampled*. Usually the gain to a node is desired, so select voltage.

3. Click *Select* to the right of the *Net+* field and select the node in the schematic. Note that if the *Net-* field is left blank, the global ground node is used.
4. Specify the desired threshold voltage based on the time-domain waveform from the pss analysis
5. Choose an option from the *Crossing Direction* drop-down box. In the example, *all* is selected.
6. If you need more than 16 crossing points, type the number you need in the *Maximum Samples* field.

7. If you want to sample at specific times from the pss waveform, you can type in the times in the *Additional Timepoints* field.

8. Now run the simulation. When the simulation completes, open the *Direct Plot Form* and set it up, as shown below.

9. Select *pac* results.
10. Select *Voltage*.
11. Select *spectrum*.
12. Select *dB20*.
13. Select the time you want from the list.
14. Select the output node in the schematic.

The waveform is plotted as shown below

To zoom in to the frequency of interest, double-click one of the numbers on the X Axis, and type in the output frequency range of interest. This is shown below.

To measure the conversion gain to the output, move the tracking cursor to the output frequency and read the gain or place a point marker by typing `m`.

Virtuoso (R) Visualization & Analysis XL
File Edit View Graph Axis Trace Marker Measurements Tools Window Browser Help
cadence
Layout: Auto
Subwindows: AC Response
Workspace: Classic
Data Point
50.0000M
WorkshopLib db_mixer schematic
Periodic AC Response
Name
Vis harmo
...time=51.3719p; pac dB20(V)
...1.3719p; pac dB20(V) -10
...1.3719p; pac dB20(V) -9
...1.3719p; pac dB20(V) -8
...1.3719p; pac dB20(V) -7
...1.3719p; pac dB20(V) -6
...1.3719p; pac dB20(V) -5
...1.3719p; pac dB20(V) -4
...1.3719p; pac dB20(V) -3
...1.3719p; pac dB20(V) -2
...1.3719p; pac dB20(V) -1
...1.3719p; pac dB20(V) 0
...1.3719p; pac dB20(V) 1
...1.3719p; pac dB20(V) 2
...1.3719p; pac dB20(V) 3
...1.3719p; pac dB20(V) 4
...1.3719p; pac dB20(V) 5
...1.3719p; pac dB20(V) 6
...1.3719p; pac dB20(V) 7
...1.3719p; pac dB20(V) 8
...1.3719p; pac dB20(V) 9
...1.3719p; pac dB20(V) 10
M1: 50.0(MHz) 14.151665(B)
V(dB)
14.4
14.3
14.2
14.1
14.0
13.9
13.8
13.7
13.6
13.5
1.0
20.8
40.6
60.4
80.2
100
freq(MHz)
mouse L:
M:
R:
7(11)
RMB zoom reset to xy mode.

Periodic Noise Analysis (Pnoise)

Small-Signal Versus Large-Signal Analysis for Noise

In most cases, noise is a small-signal problem in the real world. Consider an amplifier. One approach is to calculate the small-signal gain from the noise sources to the output, then calculate the frequency response of the noise source, and then add up the noise power at the output. This is done in the linear noise analysis. This runs quickly because the gain calculation just takes the tangent at the DC operating point for everything in the circuit. This ignores large-signal effects like slew-rate limits and clipping. Because noise in most circuits is actually very small compared to the output levels, this approach works well. Because it is a linear approach, frequency translations cannot be calculated.

When the input to the amplifier gets large enough that it starts to create nonlinearity, the frequency translations in the circuit need to be taken into account. Noise near the harmonics can mix up or down to the output frequency. In this case, the large signal needs to be applied in a large-signal analysis like periodic steady-state (pss) where the nonlinearity and harmonic levels can be calculated. After the pss runs, pnoise runs and is similar to linear noise except the frequency translations of the circuit are taken into account. Instead of calculating a single gain from the noise source to the output, both the amplitude of the noise and the gain for the noise sources at the significant harmonics needs to be calculated, and each noise source now has several contributions at the output from several different noise frequencies that mix or alias to the output frequency. The basic strategy is to apply the signals that cause nonlinearity in the pss analysis, and follow that with a pnoise calculation.

This same approach can be taken for mixers where the LO signal deliberately introduces nonlinearity for the purpose of frequency translation. The LO signal is applied to the mixer in the pss analysis, and pnoise follows that. This works with the mixer by itself, or in combination with other circuits, like an LNA and a baseband amplifier.

Many designs need to tolerate a large amplitude blocking signal. The basic strategy is to apply the LO and the blocking signal (and not the RF signal) in the pss analysis and follow with pnoise. Note that depending on the frequencies of the LO and the blocker, pss may not be appropriate. If the highest frequency divided by the periodicity of the system (the pss beat frequency) is about 25 or less, pss-pnoise is a reasonable approach. If the highest input frequency is 2.5GHz and the periodicity is 1MHz, use hb-hbnoise or qpss-qpnoise instead because they will require much less time and memory to run. This is because pss calculates all the harmonics of the pss beat frequency, and 2500 harmonics would be needed just to calculate the fundamental frequency of the 2.5GHz input. Both hb and qpss calculate just the mixing frequencies that are actually produced by the circuit, which is a much smaller number of harmonics. The blocker power can be swept in the pss analysis and pnoise is run after each

sweep value for the blocker power to measure the increase in noise with higher blocker power (desensitization).

For large-signal noise problems, or for systems that are not periodic, transient noise is provided. For more information on this, refer to the *Virtuoso® Spectre® Circuit Simulator and Accelerated Parallel Simulator User Guide*. This approach requires specifying a maximum noise frequency to be analyzed, which in RF systems is usually quite high. Two timepoints are created in the period of noisefmax. The transient needs to be run over many noise points in order to characterize the noise present in the circuit and needs post-processing to see the results in the frequency domain. This usually requires much more time than using small-signal noise analyses.

Overview of Simulation Capabilities

The fundamental quantity calculated in pnoise is the output noise of the circuit with a large signal applied and including the noise translations caused by the nonlinearity of the circuit. In order to run pnoise, a periodic steady-state (pss) analysis needs to be run first in order to calculate the nonlinearity of the circuit and the amplitude of the harmonics that are present to mix with. Pnoise takes this information from the pss analysis and then for each noise source, it calculates the transfer functions from all the frequencies specified by the maximum sideband field in the *Choosing Analyses* form. Then it calculates the frequency response of the noise source itself. Next, it calculates the total noise power at the output for that source at all the noise frequencies that contribute at the output. Finally, it adds up all the noise power from all the sources at all the frequencies. Input-referred calculations require the selection of the input frequency range by setting the reference sideband. This selection is necessary because there are many transfer functions from the input to the output caused by mixing or aliasing from the different harmonics of the system. The passband frequency is also needed for the noise figure calculations.

Note that pnoise is a small-signal calculation that works for periodic systems. If the noise is large-signal, or the system is not periodic, transient noise should be used.

Example

Consider the double-balanced diode mixer below where a single-sideband noise figure measurement is desired.

The input source has variables set to define the frequency and amplitude of the two large-signal outputs. This is done so that the frequency and the amplitude can be set in the ADE window without changing the schematic.

The property list is shown below.

The variables have values set in the ADE variables section. The easiest way to get the variables list is to generate a netlist from the ADE or use *Variables - Copy From CellView*. This causes all the variable names to be entered in the variables section. Just click the value field for each variable and enter a value. Press <Enter> when you are done. In the figure below, the LO frequency set by the variable flo is 1GHz. The variables *fb1* and *fb2* which set the frequency of the large-signal RF inputs is zero. This disables the large-signal waveform. This circuit just has the signal that causes the frequency translations to occur (the LO signal) applied. Note that 1.4 volts peak is +13 dBm. This is a common drive level for diode mixers.

The pss analysis needs to be run first in order to measure the amplitude of the harmonics and to characterize the nonlinearity of the system. From this, the noise folding from all the noise sources can be calculated in the pnoise analysis. For more information on the setup of the pss form, please refer to Overview of Periodic Steady-State (pss) Analysis on page 505.

Pss can be set to use either the shooting engine or the harmonic balance engine. The example below uses the shooting engine. Because shooting is a variation of the transient which automatically reduces the timestep in areas or rapid transition (where the diodes switch) and because the pss includes at least 200 timepoints in the solution, the pss shooting solution itself contains information to the 100th harmonic of 1GHz. For the pnoise to be

accurate, at least 5 timepoints are needed at the highest frequency that is being calculated in the pnoise. Thus, with the defaults, frequency translations through the 40th harmonic of 1GHz can be accurately calculated when shooting is selected as the engine.

When harmonic balance is selected as the engine, the number of harmonics and the oversample factor need to be set appropriately. For more information, see Chapter 3,

"Frequency Domain Analyses: Harmonic Balance," Below is an example of harmonic balance in the pss *Choosing Analyses* form.

Regardless of the engine that is used in the pss *Choosing Analyses* form, the noise output frequency and the number of noise translations to be calculated are set up in the pnoise form. When harmonic balance is selected for the engine in pss, leave the maximum sideband field blank or set it to the number of harmonics in the pss-hb *Choosing Analyses* form, which calculates the noise translations for all the harmonics that are calculated in the pss analysis. When shooting is selected, you must specify the number of noise translations you want to calculate in the maximum sidebands field. Up to 40 noise translations can be calculated with

the default settings in pss. The example below shows 40 sidebands, which is the number of harmonics used in the pss-hb analysis that is shown in the preceding figure.

Choosing Analyses -- Virtuoso® Analog Design Environm
Analysis
tran
dc
ac
noise
xf
sens
dcmatch
stb
pz
sp
envlp
pss
pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
hbsp
Periodic Noise Analysis
PSS Beat Frequency (Hz)
1G
Multiple pnoise
Sweeptype
default
Sweep is currently absolute
Output Frequency Sweep Range (Hz)
Start-Stop
Start
50M
Stop
150M
Sweep Type
Linear
Step Size
Number of Steps
20
Add Specific Points
Sidebands
Method
default
fullspectrum
Maximum sideband
40
When using shooting engine, default value is 7.
Output
probe
Output Probe Instance
/if
Select
Input Source
port
Input Port Source
/RF
Select
Reference Side-Band
|f(in)| = |f(out) + refsideband * fund|
Select from list
From (Hz)
0
To (Hz)
10G
Max. Order
1
side Frequencies 1G
u 50M 150M 0
l 850M 950M -1
u 1.05G 1.15G 1
Noise Type
sources
sources: single sideband (SSB) noise analysis
Noise Separation
yes
no
separate noise into source and gain
Enabled
Options...
OK
Cancel
Defaults
Apply
Help

The reference sideband setting is used to specify the passband frequency for the noise figure calculation.

In the example above, the output frequency is near baseband.

The reference sideband selects the passband frequency for the noise figure calculation. Once both forms are set up, the simulation is run. To plot outputs, select *Results - Direct Plot - Main Form* in the ADE window.

The top of the form lists all the analyses that were run. Select *pnoise* where the noise figure is calculated. Next, select *NF* because that is what is desired to be plotted. If the noise figure over a bandwidth is desired, select *Integrated Over Bandwidth*, and specify the frequency range you desire for the measurement.

When you are done, click *Plot*.

The waveform window plots the noise figure curve and the integrated noise figure calculation over the frequency range you specified in the direct plot form. This is shown below.

Principles of Pnoise

Pnoise has three basic types of noise calculation. The first mode calculates the average noise power that occurs with the variations of the LO or clock signal taken into account. The noise is averaged over the LO or clock cycle. This is calculated when the noisetype parameter is set to sources. The AM and PM components of the noise can be calculated when the noisetype is set to modulated. When the noisetype is set to jitter for driven circuits, an instantaneous measurement of the noise at the threshold crossing in the pss analysis is calculated. This allows jitter measurements for digital outputs on either the rising, falling, or both edges of the clock. For oscillators, when jitter is selected, either an averaged

measurement can be made for phase noise, or an instantaneous measurement can be made at one threshold voltage of the oscillator output voltage or current. There is a fourth setting in the *Choosing Analyses* form which is called timedomain. Conceptually, this does the same noise measurement as noisetype=jitter, but it requires you to do the jitter calculation manually. It also allows multiple places in the rising and/or falling edges to be measured.

Noise Type=sources

Noise Output Near Zero Frequency

First, the pss analysis is run to calculate the nonlinearity and the harmonics. Then pnoise is run to calculate the noise. In all systems, there is a maximum number of noise translations that need to be taken into account. Either the system harmonics become so small that the amount of noise translation becomes minimal or the devices themselves have finite noise bandwidths.

These noise calculations from mixing or aliasing with the different harmonics are called sidebands.

In the diagram above the X axis is the frequency. The top represents the harmonics that are calculated by the pss analysis. The bottom represents the noise output frequency. The arrows show the various mixing paths to the output frequency. Pnoise calculates the noise contribution at the output with all these mixing paths in place. Black is the noise that occurs without mixing or aliasing. Purple is the noise that mixes with the first harmonic. Blue is the noise that mixes with the second harmonic. Red is the noise that mixes with the third harmonic.

In pnoise, the output frequency range is specified along with the highest harmonic to calculate the mixing products.

Noise Output Near The First Harmonic

Here, the noise output frequency is just above the first harmonic frequency. Black is the noise that occurs without mixing. Purple is the noise that mixes with the first harmonic. Blue is the noise that mixes with the second harmonic. Note that the noise frequency just below the first harmonic mixes with the second harmonic to provide the output just above the first harmonic. Red is the noise that mixes with the third harmonic. Note that the second term that mixes with the third harmonic is just above the fourth harmonic of the pss analysis which is not shown. The noise term shown with a dotted line is mixing with the fourth harmonic of the system.

Frequency Sweep

Since the fundamental quantity that is calculated in pnoise is the output noise, the frequency range defined in the *Choosing Analyses* form defines an output frequency range to calculate the noise for. For driven circuits, the default is to sweep an absolute frequency range. Oscillators will be covered later in this chapter.

For an up-conversion system, sometimes, a log frequency range is desired near one of the harmonics of the system. In that case, select relative sweep, and specify a harmonic number. In the output frequency range, type the desired offset frequency from that harmonic. Below is

a diagram that shows relative to the first harmonic, which calculates the frequencies just above the first harmonic.

When a log sweep is desired below the first harmonic, set the relative harmonic to -1 as shown in the diagram below.

When automatic is selected for sweep type, 100 total points will be calculated. In most cases, fewer points are needed, so it is usually better to select linear or log and then specify the sweep you desire. Typically, 3 to 5 points per decade or about 20 total points are needed.

Maximum Sideband

There is a property in the *Choosing Analyses* form called *Maximum sideband* that is used to define how many mixing products should be calculated. If *Maximum sideband* is zero, the noise analysis will only contain the noise that appears without mixing or aliasing of any type.

This is not equivalent to a linear noise analysis because pnoise takes into account the instantaneously varying nature of the noise because of the large signal being present in the pss analysis.

When *Maximum sideband* is one, then the noise without frequency translation and the noise that mixes or aliases with the first harmonic of the pss is present in the noise analysis. When *Maximum sideband* is 10, the noise analysis includes noise mixing through the 10th harmonic of the pss analysis. When harmonic balance is selected as the pss engine, the number of harmonics in the pss analysis and the maximum number of sidebands should be the same. This is easily accomplished by leaving the *Maximum sideband* field blank in the *Choosing Analyses* form in ADE. When shooting is selected for the pss engine, up to 40 sidebands can be set in the maximum sideband field by default. If you need more than 40 sidebands in pnoise, consider using full-spectrum pnoise.

Full-Spectrum Pnoise

Full-Spectrum pnoise is useful for circuits like switched-capacitor filters or sampling circuits where aliasing occurs through very high harmonics of the clock. The runtime advantages are large with no loss in accuracy of the result. Full-Spectrum pnoise is available when shooting is selected for the pss engine and APS is selected in the *Setup - High Performance Simulation* menu in ADE. Selecting *fullspectrum* in the pnoise form forces APS to be selected in ADE. If you are running from the command line without using aps, fullspectrum will not be used. In normal pnoise when shooting is selected, you need to set the maximum sideband term. In full-spectrum pnoise you do not except in cases where the pss beat frequency is 100KHz or less. In this case, set maximum sidebands to the 1/F noise corner frequency divided by the pss beat frequency. Pnoise calculates all the noise translations it can, based on the maximum timestep in the pss analysis. The easiest way to change the maximum timestep in pss is to increase the number of harmonics above 10 in the pss *Choosing Analyses* form. Full-spectrum pnoise runs faster than normal pnoise when the maximum number of sidebands in pnoise is about 50 or greater. The larger the number of sidebands, the larger the speedup for fullspectrum.

Fullspectrum pnoise is available for all available noisetypes. Noise separation is not supported when fullspectrum is selected.

Setting Harmonics and Sidebands

Harmonic Balance as the PSS engine

The process of setting harmonics and sidebands is similar to setting the maximum number of harmonics for the harmonic balance simulation. Start with an estimate based on the harmonic content of the input signal(s) and the nonlinearity of the circuit. Then run a simulation. Raise harmonics and sidebands and run the simulation again. If the noise measurement did not change, then the original number of harmonics and sidebands may be reduced. If it did change, then more are needed. For more details on this, refer to the section Harmonic Balance Noise Analysis (hbnoise) on page 301.

Shooting as the PSS Engine

For full-spectrum pnoise, start with 10 pss harmonics, Select full-spectrum, and select APS. Run the simulation, and plot the noise result. Now increase the number of harmonics in the pss analysis, which forces more pss timepoints, and run the simulation again. If the pnoise result didnot change, then you had enough harmonics to begin with. If the noise result did change, then increase the number of pss harmonics and run the simulation again.

For regular pnoise, estimate the number of sidebands based on the pss harmonic plot. As a coarse estimate, find the harmonic where the LO or clock signal drops to about 40 dB below the amplitude of the LO or clock first harmonic. For example, if this is the 10th harmonic, set maximum sideband to 10. Run the simulation and plot the noise measurement. Increase the number of sidebands until the noise result does not change anymore. Use the smallest number of sidebands that produces a stable result. If the number of sidebands is greater than about 50, consider using full-spectrum pnoise because it is likely to run faster than normal pnoise.

Noise Separation

In addition to the total output noise, the individual noise contributions can be plotted if noise separation is selected in the *Choosing Analyses* form. More information about the noise separation will be provided later. Noise separation is available when the noise type is set to sources only and is not available for fullspectrum pnoise. The things that can be plotted are:

1. Total noise at the output from each individual noise input frequency. This allows the identification of which noise frequencies are causing the noise problem.
2. The noise at the output from each instance name with all noise mechanisms included for mixing from an individual noise input frequency. Once the noise frequency is identified in point 1 above, this capability plots the total noise from each component at the

troublesome noise frequency. This allows the identification of which component in the circuit is causing the problem.

3. The noise at the output from each instance name with all noise mechanisms broken out separately for mixing from an individual noise input frequency. This capability splits out the individual noise sources at the troublesome noise frequency. For example, if a parasitic resistor was found to be the problem, the component might be resized to reduce the noise.

4. The current noise at the noise instance for each instance name with all noise mechanisms included for mixing from an individual noise input frequency. This capability has limited value. For this plot, all the noise source currents for all the noise mechanisms within the component are added together.

5. The noise current in the instance from each instance name with all noise mechanisms broken out separately for mixing from an individual noise input frequency. This capability allows the identification of which noise sources are the largest in the circuit. The measurement is at the source, not at the output.

6. The gain to the output from each individual instance with all the individual noise sources broken out separately for mixing from an individual input frequency. This capability plots the gain from the noise sources in point 5 above to the output of the circuit.

Multiple Pnoise

Some systems contain multiple points in the circuit where the noise measurement is desired. When the *Noise Type* is *sources*, multiple pnoise analyses are allowed. At the top of the pnoise *Choosing Analyses* form, an option called *Multiple pnoise* is provided. When this option is selected, up to six pnoise analyses can be made after a single pss analysis.

Noisetype = Modulated for Driven Circuits and Oscillators (AM and PM Noise)

Modulated is provided so that the AM and PM components of noise can be separated. When the *Noise Type* is *sources*, the total noise (AM + PM) is calculated. To illustrate AM noise, see the phasor diagram below.

This diagram illustrates a periodic signal with AM noise. The black phasor is the output signal, and AM noise is a random vector that varies the amplitude of the signal.

PM noise is shown in the phasor diagram below.

In this case, the amplitude is constant, but the noise is a random vector perpendicular to the signal phasor. The timing of the signal varies a bit from cycle to cycle.

Most circuits produce a combination of AM and PM noise. This concept is shown below.

In this case, a random vector at any angle and amplitude is produced that varies at each instant of time. If the uncertainty creates a round shape, the AM component and the PM component are the same size.

In many circuits, AM or PM noise may dominate. If the PM noise dominates, the circle turns into an ellipse that is larger in the perpendicular direction than in the parallel direction. This is shown below.

Modulated also provides the total noise in the frequencies above the output frequency (the USB in the *Direct Plot Form*) and the noise below the output frequency (the LSB in the *Direct Plot Form*).

Jitter for Driven Circuits

Jitter is usually applied to digital circuits where the average noise over the entire waveform is not appropriate. The noise in the high state and in the low state does not matter because the output voltage is so far from the voltage that triggers the load that it does not matter. The only time the noise matters is at the exact time where the trigger event is generated. Also, the noise in the low and high state is quite small because one device is completely off, and the other device is driven on. It is basically a resistor. The device is not very good as an amplifier because the output is pulled hard into a rail. As the state changes, both devices are on, and they are relatively better amplifiers, so the noise is likely to be larger in the switching interval. In jitter for driven circuits, the voltage for the noise measurement, and the selection of the rising edge, the falling edge, or both is made. At the threshold voltage specified, a noise measurement is made at that instant in time. Once the noise voltage is known, the timing

uncertainty can be determined from the slew rate. Since the RMS noise voltage is calculated, the timing uncertainty is one standard deviation from the ideal crossing time. This is illustrated below.

Timedomain for Driven Circuits

This is a variation of the jitter analysis explained in the previous section. In this analysis, instead of specifying a threshold voltage, a list of times from the time-domain waveform is supplied. A noise analysis is performed at those instants in time, and the noise voltage is calculated at each timepoint. The conversion from noise voltage to time-domain jitter is a manual step. Divide the noise voltage by the slew rate to get the jitter.

Oscillators

Noise Type = Sources

For oscillators, the pnoise frequency range defaults to *relative*. Specify the harmonic number as appropriate for the system you are simulating. If you are simulating an oscillator by itself, then the harmonic number is likely to be 1. If you have an oscillator and a diode frequency doubler, then the harmonic number is likely to be 2. If you have an oscillator with a frequency divider, in the pss form, you should specify the approximate frequency of oscillation for the frequency-divided signal. In the pnoise form, if the noise is desired on the frequency divided output, then the relative harmonic is 1. If the noise is desired at the output of the oscillator, the relative harmonic number is the divide ratio. The meaning of relative is to take the frequency of the harmonic number specified and add to it the frequencies specified in the *Choosing Analyses* form. If the oscillator had a 1GHz output, and the pnoise had 1M relative to the first harmonic specified, the actual output frequency is 1G + 1M, or 1001M.

Oscillators do not need noise figure calculations and are not typically driven, so the input source should be set to *none*. The output is usually a voltage, and an output node is typically selected.

Pnoise is a small-signal analysis and is not limited by large-signal effects, such as clipping or slew-rate limits. As a result, at low offset frequency, the phase noise might be significantly greater than 0dBc/Hz. This indicates that the noise is larger than the oscillations, which is not physically possible. If you want to see the phase noise curve level off at low frequency, set the *lorentzian* option to *yes*. More information is provided in the lorentzian option discussion.

Phase noise is the total output noise (the AM plus the PM component of noise) divided by the harmonic specified as the relative harmonic for the node specified for the noise measurement. If you want to see the AM and PM components of noise, or you want to measure jitter, or you want to make an instantaneous point in time noise measurement, set the *Noise Type* to *modulated* or *jitter*.

Modulated

Modulated is provided so that the AM and PM components of noise can be separated out. When the *Noise Type* is *sources*, the total noise (AM + PM) is calculated. For a diagram to understand the concepts, see Noisetype = Modulated for Driven Circuits and Oscillators (AM and PM Noise) on page 797

Timedomain

This is usually applied to oscillators that drive digital dividers where the average noise over the entire waveform is not appropriate. The noise above and below the threshold does not matter because the output voltage is so far from the voltage that triggers the load that it can never cross the threshold. The only time the noise matters is at the exact time where the trigger event is generated. In timedomain for oscillators, a list of times from the waveform is supplied and a noise measurement is made at that list of times. Once the noise voltage is known, the timing uncertainty can be determined from the slew rate. The noise calculated from this analysis provides a relative measure of the noise at different times. For a quantitative measurement, use PM jitter. For a diagram that illustrates this, see Jitter for Driven Circuits on page 800.

FM Jitter

FM jitter calculates a standard averaged phase noise measurement, and also the AM and PM components from the modulated analysis, and adds the ability to integrate the phase noise curve to calculate the cycle jitter or the cycle-to-cycle jitter. The jitter calculations are

integrated into the *Direct Plot Form* in ADE. All the measurements are averaged over the oscillator cycle.

PM Jitter

This is like the timedomain jitter measurement. You specify a threshold voltage and the timing jitter is calculated. In PM Jitter, a quantitative jitter measurement can be made from the direct plot form. For a diagram on how this works, see Jitter for Driven Circuits on page 800.

Noise Calculations

Phase Noise (Driven Circuit)

For a driven circuit with a single input frequency, phase noise is the total noise at the output divided by the amplitude of the first harmonic at the output node specified in the pnoise analysis. Note that phase noise from the pnoise results selection is a single-sideband phase noise. Only the frequencies above the oscillator frequency are included in the measurement. To get a double-sideband phase noise measurement, set the noise type to jitter, and select the pnoise jitter result set.

Noise Factor and Noise Figure

When the pnoise analysis is set up with voltage selected instead of probe for the input, noise figure and noise factor will not have the noise of the load subtracted automatically from the measurement. Using a port as the input probe is required to obtain noise figure and noise factor calculations.

In the *Direct Plot Form*, noise factor and noise figure are both single-sideband measurements.

Noise Factor

Noise Factor is:

```
((All the noise power at the output from all the components with all the frequency
translations) - (Noise power at the output from the load))/Noise power at the output
from the resistance of the input port from the passband frequency only.
```

Note that in order to calculate the noise factor, a port component must be used as the input. In the *Choosing Analyses* form, the input port is identified by the instance name. The port component is a voltage source in series with a resistor and is considered as a single element.

No access is provided to the results for the internal node where the resistor and voltage source join.

In order to determine the passband frequency, the reference sideband must be supplied. This specifies the passband frequency for the noise figure calculation. The easiest way to select the reference sideband is to use the *Select from list* feature in the pnoise *Choosing Analyses* form.

The noise from the load is normally defined by selecting the port or resistor that is serving as the load for the circuit in the *Choosing Analyses* form.

Noise Figure

Noise figure is:

```
10*log(10) (Noise Factor)
```

Noise factor (IEEE) is similar to the noise figure measurement above. If you subtract from the numerator all the noise from the input port at all frequencies except the passband, you have the IEEE noise factor.

Note that in order to calculate the IEEE noise factor, a port component must be used as the input. In the *Choosing Analyses* form, the input port is identified by the instance name. The port component is a voltage source in series with a resistor and is considered as a single element. No access is provided to the results for the internal node where the resistor and voltage source join.

In order to determine the passband frequency, the reference sideband must be supplied. This specifies the passband frequency for the noise figure calculation. The easiest way to select the reference sideband is to use the *Select from list* feature in the *Choosing Analyses* form.

The noise from the load is normally defined by selecting the port or resistor that is serving as the load for the circuit in the *Choosing Analyses* form.

Noise Factor IEEE

Noise Factor IEEE is:

```
((All the noise power at the output from all the components with all the frequency
translations) - (Noise power at the output from the resistor in the input port at
all frequencies except for the passband) - (Noise power at the output from the
load))/Noise power at the output from the resistance of the input port from the
passband frequency only.
```

Noise Figure IEEE

Noise figure IEEE is:

```
10*log(10) (IEEE Noise Factor)
```

Noise factor DSB is similar to the single sideband noise factor above, except in the denominator, the noise power from the input port at the image frequency is added.

Note that Noise Figure and NFiee are both single-sideband measurements. The difference between the measurements is that in noise figure, all the noise frequencies from the input port that contribute noise at the output are included, and in NFieee, only the noise at the passband frequency is included. As a result, NFieee will produce a lower noise figure than Noise Figure will The difference between the measurements depends on the circuit.

.

Noise Factor DSB

Noise factor (DSB) is:

```
((All the noise power at the output from all the components with all the frequency
translations) - (Noise power at the output from the load))/Noise power at the output
from the resistance of the input port from the passband and image frequencies only.
```

Note that in order to calculate the DSB noise factor, a port component must be used as the input. In the *Choosing Analyses* form, the input port is identified by the instance name. The port component is a voltage source in series with a resistor and is considered as a single element. No access is provided to the results for the internal node where the resistor and voltage source join.

In order to determine the passband frequency, the reference sideband must be supplied. This specifies the passband frequency and by inference the image frequency for the noise figure calculation. The easiest way to select the reference sideband is to use the *Select from list* feature in the *Choosing Analyses* form.

The noise from the load is normally defined by selecting the port or resistor that is serving as the load for the circuit in the *Choosing Analyses* form.

Noise Figure DSB

Noise figure (DSB) is:

```
10*log(10) (Noise Factor (DSB))
```

Summary:

`No` = total output noise
`Ns` = noise at the output due to the input probe at the passband frequency (the source)
`Nsi` = noise at the output due to the image harmonic at the source
`Nso` = noise at the output due to harmonics other than input at the source
`Nl` = noise at the output due to the output probe (the load)
`IRN` = input referred noise
`G` = gain of the circuit
`F` = noise factor
`NF` = noise figure
`Fdsb` = double sideband noise factor
`NFdsb` = double sideband noise figure
`Fieee` = IEEE single sideband noise factor
`NFieee` = IEEE single sideband noise figure

then,

```
IRN = sqrt(No^2/G^2)
F = (No^2 - Nl^2)/Ns^2
NF = 10*log10(F)
Fdsb = (No^2 - Nl^2)/(Ns^2+Nsi^2)
NFdsb = 10*log10(Fdsb)
Fieee = (No^2 - Nl^2 - Nso^2)/Ns^2
NFieee = 10*log10(Fieee)
```

When the pnoise analysis is set up with voltage selected instead of probe, noise figure cannot be calculated.

Input-Referred Noise

Important

When the pnoise analysis is set up with the input source being a port, input-referred noise may not be exact. In this case, the input-referred noise is referred to the voltage source that is inside the port component. The amplitude of the noise is divided by two, which assumes a perfect match. If an accurate input-referred noise is desired, set the *Input Source* to *voltage*, and select the input node. This will disable the noise figure calculation.

Noise transfer function

Pnoise calculates the forward gain from the input passband to the output and saves it. In the *Direct Plot Form*, this is called *Transfer Function*, and is in linear volts at the output divided by volts at the input.

Commonly Used Pnoise Options

Cyclo2txtfile

Select the `cyclo2txtfile` option if you want the simulator to create a file that has all the noise information in it. When this option is set, pnoise runs a noise analysis at all the sideband frequencies using the sweep information from the pnoise form. The file is created in the `psf`

directory and has the filename `pnoise.pnoise.data`. If you intend to keep this file for use in a voltage source with the noise file set to a name, make sure you copy this file to another directory. The `psf` directory contents are deleted at the start of each simulation.

The `psf` directory is located either in the `simulation/<circuit_name>/spectre/schematic` directory or in the `<circuit_name>/spectre/config` directory. The simulation directory defaults to your home directory and can be set by your system administrator. Contact your system administrator to find the simulation directory if you are unfamiliar with the location.

The output file from cyclo2txtfile has two columns. The left column is the frequency, and the right column is the noise voltage squared, normalized to a 1Hz noise bandwidth. Below is an example of the file.

```
Terminal
File  Edit  View  Terminal  Tabs  Help
1.0000000002384186e+03     4.1976179202311482e-20
1.5848931927680969e+03     1.6713526184450628e-20
2.5118864316940308e+03     6.6541397418897690e-21
3.9810717058181763e+03     2.6490872073399725e-21
6.3095734450817108e+03     1.0545957949468125e-21
1.0000000000238419e+04     4.1981471199230932e-22
1.5848931924819946e+04     1.6711642040673237e-22
2.5118864315271378e+04     6.6548863771520372e-23
3.9810717055559158e+04     2.6594618704367568e-23
6.3095734448194504e+04     1.0893235490275314e-23
1.0000000000023842e+05     5.1584490781642718e-24
1.5848931924629211e+05     4.2122011260108464e-24
2.5118864315128326e+05     7.3817813521682905e-24
3.9810717055368423e+05     1.8376642299234070e-23
6.3095734448051453e+05     7.2002344421267964e-23
1.0000000000002384e+06     1.3844360510239424e-21
1.5848931924612522e+06     8.2447867935178601e-25
2.5118864315097332e+06     1.8906195812299195e-23
3.9810717055351734e+06     8.4013044161565058e-23
6.3095734448022842e+06     2.4460759370357001e-22
1.0000000000000238e+07     6.0508315175955095e-22
1.5848931924611330e+07     1.3563078334928723e-21
2.5118864315096140e+07     2.6606482609752016e-21
"hbnoise.pnoise_hbnoise.data" [readonly] 651L, 32550C
```

Interpolation in Noise Files

All independent sources have a property called Noise file name. This is an ascii file with two columns. The left column is the frequency, and the right column is the noise voltage squared

normalized to a 1 Hz bandwidth. This file is read for all the small-signal noise analyses. Between the points in the file, an interpolation is made.

Both curves in the above figure use the `cyclo2txt` output file for a circuit at about 1.85GHz. Just the noise in the file is plotted. The top curve has a noise sweep to just less than one half the oscillation frequency. The bottom curve has a sweep to 10MHz relative to the first harmonic. There were 10 harmonics in the pss simulation and 10 sidebands in the pnoise analysis.

Because the noise analysis was referenced to the first harmonic, the +10 sideband starts at +10*1.85GHz + 1.85GHz, or 20.35GHz, and goes to about 20.36GHz for the 10MHz sweep, and 21.25GHz for the 900MHz sweep. Note that the data near 20GHz in both starts at about -120dBV and drops. The minus 10 sideband starts at -10*1.85GHz + 1.85GHz, or -16.65GHz to -16.64GHz for the 10MHz sweep and -15.75GHz for the 900MHz sweep. Note that there is no data on the bottom side of the 10th and 11th harmonic. The data in between the frequency points in the file is interpolated. That is the reason for the large rise for the last 2 noise spikes.

The spikes from the 900MHz sweep is narrower because the data goes to a higher frequency near 17GHz and 19GHz.

The figure below shows the curves overlaid on the same strip.

Note that the interpolation above and below the actual data in the 10MHz sweep (the blue trace) causes much more noise than the real system produces. Make sure that your noise sweep is sufficient for your requirements.

Solver

This option is used for driven circuits only.

The default is *turbo*.

For harmonic balance pnoise, leave this option at the default value. For shooting pnoise, sometimes you will see warning messages that the residual is larger than the tolerance specified. This usually occurs at low offset frequencies. If you see this message, then select the *std* solver, which is more able to handle low offset frequencies. The runtime will increase when the std solver is selected.

Oscsolver

This option is used for oscillators only.

The default is *turbo*.

For harmonic balance pnoise, leave this option at the default value. For shooting pnoise, sometimes you will see warning messages that the residual is larger than the tolerance specified. This usually occurs at low offset frequencies. If you see this message, then select the std solver, which is more able to handle low offset frequencies. The runtime will increase when the std solver is selected.

The ira solver may use less memory than turbo or std, but it is less robust for convergence.

Augmented

Leave this option at the default. (augmented=yes)

If your circuit has low frequency poles, (usually from a biasing circuit) the default augmented solver is much more accurate than the standard solver. Pmonly and amonly should not be selected.

hbprecond_solver

This option is only available when harmonic balance is set for the pss solver. When APS is not used, only the basic solver is available. When APS is used, first a much faster preconditioner is used. Occasionally, this preconditioner will stagnate, and when it does, it issues a warning that it is switching to the basic solver, and it reverts back to the basic solver. A preconditioner is a mathematical algorithm that makes the iterative matrix solving process faster.

Lorentzian

There are two ways of looking at phase noise. One way is to think of measuring the spectral output of the oscillator with a network analyzer. In this case, as the frequency gets close to the oscillator frequency, the amplitude starts going up, levels off, and then drops as the oscillator frequency is passed. In this case, the noise cannot be larger than the oscillations themselves, which is 0 dBc. If this is how you view phase noise, set the *lorentzian* option to *yes*.

If you think in terms of jitter, in one cycle of the oscillator output, because of the noise, you can calculate one standard deviation of the timing. If you go two cycles, and you integrate the noise with respect to time, you get twice as much noise power, or about 1.4 times the noise voltage or jitter. When you get to an infinite number of oscillator cycles, the jitter also becomes infinite. As you increase the time, you are lowering the noise frequency, and in this case, as

the time becomes large, the frequency becomes very low. At infinite time, you get infinite jitter, and this occurs at zero offset from the carrier. Jitter is just another way of looking at phase noise. In other words, the phase noise can easily exceed the carrier amplitude (0 dBc) as the offset frequency becomes small because it is heading to infinity at zero frequency offset. If this is how you view phase noise, set the *lorentzian* option to *no*.

Lorentzian=yes means calculate the leveling off. Lorentzian=cornerfreqonly calculates the continuously rising phase noise, but it places a marker on the phase noise curve at the frequency where the phase noise would level off. No causes the phase noise to continue to rise as the offset frequency approaches zero.

Tolerance

Leave this option at the default.

Pnoise uses an iterative solver to calculate the noise, and any iterative solver needs a tolerance term to specify when the system has been solved accurately enough. The default for shooting pnoise is 1e-9. Harmonic balance driven circuit pnoise tolerance is 1e-6 and 1e-4 for oscillators.

Gear_Order

Do not set this option.

Lnsolver

Leave this option at the default, which is *gmres*. To calculate the conversion gain from the noise sources to the output, many matrices are formed and solved to calculate the noise at the output of the system. These matrices are solved using an iterative solver. Several different algorithms are provided for the iterative solver. Gmres is the default because the accuracy of each iteration inherently increases. Convergence is generally good as well. The other solvers may require less memory, but they are less robust for convergence and may suffer from false convergence.

Considerable knowledge is needed to understand the differences between these methods. For more information, seek graduate-level textbooks that describe linear algebra theory.

Gmres is the Generalized Minimum RESidual method.

Qmr is the Quasi-Minimal Residual method.

Bicgstab is the STABilized BI-Conjugate Gradient method.

Resgmres is the REStarted Generalized Minimal Residual method.

Resgmrescycle

Leave this option at the default, which is *short*. For the resgmres solver, there are several different options.

Saveallsidebands

Leave this option at the default value of *no*. If you want to see the noise contributors from the different noise input frequencies, select *yes* for noise separation in the pnoise main *Choosing Analyses* form.

Enable Osc PPV

This option is used for the PLL noise methodology only, and does not need to be set as long as you use the PLL Noise Wizard from ADE.

AdditionalParams

This is typically used for customers who are evaluating new features. If you know the option name and the value, you can type it here. For example, you can type solver=std in this field instead of using the check box in the options form.

ADE Implementation

General Notes

All pnoise types are supported whether the pss engine is shooting or harmonic balance. Because Chapter 3 in this manual talks about harmonic balance exclusively, in this Chapter most of the examples will reference shooting pss. Where there are differences between shooting and harmonic balance, those differences will be noted. Otherwise, shooting and harmonic balance are completely interchangeable, and will provide the same answer as long as enough harmonics are used in harmonic balance.

Because shooting takes a minimum of 200 timepoints in the pss, the pss data inherently contains data through the 100th harmonic of the pss. For this reason, unless there is a requirement to calculate noise translations through more than the 40th harmonic of the pss, it is seldom required to force more harmonics in shooting pss. Also with shooting pss, fullspectrum is available, which does not require setting sidebands at all. In harmonic balance, only the harmonics that are solved in the pss exist. When harmonic balance is selected, if you set maximum sideband larger than the number of pss harmonics, a warning message will be entered in the Spectre output log, and maximum sideband will be reset to the number of pss harmonics.

Driven Circuits

Noise Type = Sources

On the *Choosing Analyses* form:

1. First, set up the pss analysis. For more information on the pss analysis, see the beginning of this chapter.

2. In the pnoise Analysis selection, specify the output frequency range of your circuit.

Note: Since the fundamental quantity that is calculated by the noise analysis is the noise at the output, the frequency range is always the output frequency range.

3. Set about 3 to 5 points per decade in the *Points Per Decade* field.

4. When hb is selected as the engine in the pss form, leave the *Maximum sideband* field blank or set it to the same number as the number of harmonics in the HB analysis. This specifies that the noise that mixes with all the hb harmonics is calculated. When the pss engine is set to shooting, first decide whether or not to use full-spectrum pnoise, and if the default method is used, set a number of sidebands that is appropriate for your circuit. APS is selected automatically and is required when using full-spectrum pnoise.

5. Specify the output of the circuit. When output is set to *probe*, a resistor or a port can be selected and this noise will be subtracted from the noise figure calculation. This is an easy way to get the correct answer for a noise figure calculation.

   The nodes that the probe is connected to also define the output nodes for the circuit. If you connect the probe across a differential circuit, the noise measurement is for that

differential circuit. Also note that when you select a probe, it is only used in the noise figure calculation. The output noise includes the noise of the selected element.

6. Select *port* from the *Input Source* drop-down list for noise figure calculation.
7. Choose *Select from list* from the drop-down list for reference sideband, which specifies the input frequency range (also called the passband frequency range) for the noise figure calculation.

8. Select the passband frequency range from the list.
9. Select *sources* from the *Noise Type* drop-down list to calculate the average noise power at the output averaged over the entire time of one cycle of the input waveform.

10. Select *yes* for *Noise Separation* to enable plotting of the individual noise contributors in the circuit. Noise separation is only available when *Noise Type* is set to *sources*.

Multiple pnoise

When *Noise type* is set to *sources*, multiple pnoise analyses can be run after a single pss analysis. This is useful for LNA-Mixer combinations, or in any circuit where there are multiple points in the circuit where noise is important.

To run multiple analyses after a single pss:

1. First, set up the pss analysis as appropriate for your circuit. For more information, lease see the pss section at the beginning of this chapter.

2. Select the *Multiple pnoise* check box located at the top of the pnoise *Choosing Analyses* form.

3. Set up all the noise analyses you want to run, as shown below.

Noise Type = Modulated

Modulated allows the measurement of the AM and PM components of noise separately.

First, set up the pss analysis as appropriate for your circuit. For more information, see the periodic steady-state section at the beginning of this chapter.

On the pnoise *Choosing Analyses* form:

1. Set up an appropriate frequency range of interest.

2. Set an appropriate number of sidebands. When harmonic balance is selected for the pss analysis, this field should be left blank. Fullspectrum is available for shooting only, and will be faster than the default pnoise when the number of noise sidebands is about 50 or greater with no loss in accuracy.

3. Select *modulated* from the *Noise Type* drop-down list.

4. Run the simulation. Note that modulated runs both the positive and negative frequencies. This is required to calculate the AM and PM components.

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/4_
File  Help                                                    cadence
    mod1: freq = 1.981 MHz      (30 %), step = 731.1 kHz       (5 %)
    mod1: freq = 3.14 MHz       (35 %), step = 1.159 MHz       (5 %)
    mod1: freq = 4.976 MHz      (40 %), step = 1.836 MHz       (5 %)
    mod1: freq = 1.981 MHz      (30 %), step = 731.1 kHz       (5 %)
    mod1: freq = 3.14 MHz       (35 %), step = 1.159 MHz       (5 %)
    mod1: freq = 4.976 MHz      (40 %), step = 1.836 MHz       (5 %)
    mod1: freq = 7.887 MHz      (45 %), step = 2.911 MHz       (5 %)
    mod1: freq = 12.5 MHz       (50 %), step = 4.613 MHz       (5 %)
    mod1: freq = 19.81 MHz      (55 %), step = 7.311 MHz       (5 %)
    mod1: freq = 31.4 MHz       (60 %), step = 11.59 MHz       (5 %)
    mod1: freq = 49.76 MHz      (65 %), step = 18.36 MHz       (5 %)
    mod1: freq = 78.87 MHz      (70 %), step = 29.11 MHz       (5 %)
    mod1: freq = 125 MHz        (75 %), step = 46.13 MHz       (5 %)
    mod1: freq = 198.1 MHz      (80 %), step = 73.11 MHz       (5 %)
    mod1: freq = 314 MHz        (85 %), step = 115.9 MHz       (5 %)
    mod1: freq = 497.6 MHz      (90 %), step = 183.6 MHz       (5 %)
    mod1: freq = 788.7 MHz      (95 %), step = 291.1 MHz       (5 %)
    mod1: freq = 1.25 GHz      (100 %), step = 461.3 MHz       (5 %)
Total time required for pnoise analysis `mod1': CPU = 1.26981 s, elaps
Time accumulated: CPU = 1.75973 s, elapsed = 3.44038 s.
Peak resident memory used = 31.5 Mbytes.


*************************************************************
Periodic Noise Analysis `mod2': freq = (-125 kHz -> -1.25 GHz)
*************************************************************
Using the operating-point information generated by PSS analysis `pss'.
    mod2: freq = -198.1 kHz      (5 %), step = -73.11 kHz       (5 %)
    mod2: freq = -314 kHz       (10 %), step = -115.9 kHz       (5 %)
    mod2: freq = -497.6 kHz     (15 %), step = -183.6 kHz       (5 %)
    mod2: freq = -788.7 kHz     (20 %), step = -291.1 kHz       (5 %)
    mod2: freq = -1.25 MHz      (25 %), step = -461.3 kHz       (5 %)
    mod2: freq = -1.981 MHz     (30 %), step = -731.1 kHz       (5 %)
    mod2: freq = -3.14 MHz      (35 %), step = -1.159 MHz       (5 %)
    mod2: freq = -4.976 MHz     (40 %), step = -1.836 MHz       (5 %)
    mod2: freq = -7.887 MHz     (45 %), step = -2.911 MHz       (5 %)
    mod2: freq = -12.5 MHz      (50 %), step = -4.613 MHz       (5 %)
    mod2: freq = -19.81 MHz     (55 %), step = -7.311 MHz       (5 %)
    mod2: freq = -31.4 MHz      (60 %), step = -11.59 MHz       (5 %)
    mod2: freq = -49.76 MHz     (65 %), step = -18.36 MHz       (5 %)
    mod2: freq = -78.87 MHz     (70 %), step = -29.11 MHz       (5 %)
    mod2: freq = -125 MHz       (75 %), step = -46.13 MHz       (5 %)
    mod2: freq = -198.1 MHz     (80 %), step = -73.11 MHz       (5 %)
    mod2: freq = -314 MHz       (85 %), step = -115.9 MHz       (5 %)
    mod2: freq = -497.6 MHz     (90 %), step = -183.6 MHz       (5 %)
    mod2: freq = -788.7 MHz     (95 %), step = -291.1 MHz       (5 %)
    mod2: freq = -1.25 GHz     (100 %), step = -461.3 MHz       (5 %)
Total time required for pnoise analysis `mod2': CPU = 2.71959 s, elaps
19
```

Now plot the Output noise using the Direct Plot Form.

On the *Direct Plot Form*:

1. Select *pnoise modulated* from the *Analysis* section.
2. Select *AM* or *PM* from the *Noise Type* section. *AM* is selected here.
3. Select *Magnitude* or *dBc* in the *Modifier* section.

4. Click *Plot*.

On the plot, note that the AM noise is 3dB larger than the output noise. This is because the output noise is derived from the positive frequencies only (single-sideband), and the AM and PM noise is derived using the positive and negative frequencies (double-sideband).

Noise Type = Jitter

Driven jitter is usually used for measuring the noise in a digital circuit at a specific threshold crossing. Usually, shooting pss is used to capture the waveform, but this capability is also implemented in harmonic balance. This shows the harmonic balance example. The pss *Choosing Analyses* form has a lot of harmonics and the oversample factor is set to 4, as shown below. Because the digital output is usually a square wave, a high number of

harmonics and a large oversample factor need to be set. For more information, please see the harmonic balance section at the beginning of Chapter 3.

An appropriate threshold is needed, which can be determined by plotting the time domain waveform from the pss *Direct Plot Form*. The waveform below shows an appropriate threshold value to be 1.65V.

On the pnoise *Choosing Analyses* form, the stop frequency is half the input frequency. The start frequency is 4 decades below that. Log sweep with about 5 points per decade is usually sufficient. *Maximum sideband* should be left blank when harmonic balance is selected for the pss engine. When the pss engine is set to shooting, the number of sidebands should be about two to four times the number of pss harmonics. The *Output* field is set to *voltage*, and

the *Input Source* field is set to *none* because usually, only the output jitter is needed. The threshold is set appropriately. The *Crossing Direction* option can be set to *all*, *rise,* or *fall*.

When the simulation completes, use the *Direct Plot Form* to plot the jitter. For driven circuits, select *Jee*, which is an edge-to-edge jitter. The event time is the time in the time-domain waveform for the large-signal pss simulation. In the below example, in the *Event Time*

selection, 77p is the rising edge, and 275p is the falling edge. If you are in doubt, plot the time-domain waveform from pss and go to the time listed in the *Event Time* field.

In the Modifier section, *UI* is unit interval, and *ppm* is parts per million.

The output noise frequency response and the edge-to-edge jitter are displayed. The jee measurement is one standard deviation for the expected jitter.

Noise Type = Timedomain

1. First, set up the pss analysis as appropriate for your circuit. For more information, see the periodic steady-state section at the beginning of this chapter.

Choosing Analyses -- Virtuoso® Analog Design Environn
Analysis
tran
dc
ac
noise
xf
sens
dcmatch
stb
pz
sp
envlp
pss
pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
hbsp
Periodic Steady State Analysis
Engine
Shooting
Harmonic Balance
Fundamental Tones
Name Expr Value Signal SrcId
3 flo flo 2.4G Large
4 flo flo 2.4G Large
Large
Clear/Add
Delete
Update From Hierarchy
Beat Frequency
Beat Period
2.4G
Auto Calculate
Output harmonics
Number of harmonics
10
Accuracy Defaults (errpreset)
conservative
moderate
liberal
Additional Time for Stabilization (tstab)
Save Initial Transient Results (saveinit)
no
yes
Oscillator
Sweep
New Initial Value For Each Point (restart)
no
yes
Loadpull
Enabled
Options...
OK
Cancel
Defaults
Apply
Help

2. For timedomain noise, first plot the time-domain waveform from the pss analysis, and place markers at the different voltages of interest in the waveform as shown below.

On the pnoise *Choosing Analyses* form:

1. Set the stop frequency to half the pss frequency.
2. Set the start frequency to 4 decades smaller than the stop frequency.

3. Leave the *Maximum sideband* field blank when using harmonic balance as the pss engine. When shooting is used, specify two to four times the number of sidebands as harmonics in the pss analysis. Shooting is shown below.

4. Select *voltage* from the *Output* drop-down list.

5. Specify the output node in the *Positive Output Node* field. When the negative node is left blank, the global ground node is used as the reference node.

6. Select *none* from the *Input Source* drop-down list.

7. Select *timedomain* from the *Noise Type* drop-down list.

8. Select the *Number of Points* check box and specify 1 in the text field. This tells time-domain noise to run only at the first timepoint.

9. Select the *Add Specific Points* check box and specify the time values for the desired instantaneous noise measurements. Separate the entries with a space. This list of times will be analyzed as well as the time=zero point.

Oscillators

Noise Type = Sources

The settings are the same as driven `noise type = sources` with the exception of being a relative sweep by default. Since the phase noise of the oscillator is near one of the harmonics, *Sweeptype* is set to *relative* by default. A harmonic number is provided for the case where there is a frequency multiplier or a frequency divider on the output of the oscillator. Noise type = sources also has the ability to plot the noise contributors from mixing with each harmonic when noise separation is selected. There is an example of noise separation later in this chapter.

The pnoise setup is similar to driven except that the relative harmonic is specified. The frequency of the relative harmonic number will be added (if the harmonic number is positive) or subtracted (if the harmonic number is negative) to the frequencies in the Frequency Sweep Range field.

1. First, set up the pss analysis as appropriate for your circuit. For more information on the pss analysis, see the periodic steady-state analysis section at the beginning of this chapter.

On the pnoise *Choosing Analyses* form:

2. Specify the appropriate harmonic number for your design. In this example, the first harmonic is specified in the *Relative Harmonic* field. If you are designing an oscillator by itself, the harmonic number is one. If you have an oscillator with a frequency doubler on the output, the harmonic number is two.

3. The frequency range is added to the frequency of the harmonic number that is specified.

4. If the pss engine is harmonic balance, leave the *Maximum sideband* field blank. If the pss engine is shooting, select whether the default method or the full-spectrum method should be used. Full-spectrum is faster than the default method when more than about 50 sidebands are required for your circuit and it also does not require the maximum sideband field to be set. When the default method is selected, specify the number of desired frequency translations in the *Maximum sideband* field. This will always be specific to the design. For a rough estimate, plot the frequency domain content, and find the harmonic number where the amplitude has dropped about 40dB. Try this number in maximum sidebands. To verify that you have enough sidebands, raise the number by about 50% and run again. If the noise result did not change, then you had enough sidebands to begin with. Up to 40 sidebands can be specified by default. If you need more

than 40 sidebands, set the number of pss harmonics to the number of sidebands divided by two to four.

5. Select *voltage* from the *Output* drop-down list.

6. Specify the output node in the *Positive Output Node* field. When the negative node is left blank, the global ground node is used.
7. Select *none* from the *Input Source* drop-down list.
8. Select *sources* from the *Noise Type* drop-down list. This gives a noise measurement that is the average noise power at the output with the large signal applied in the pss

analysis and the noise folding specified by the maximum sideband parameter in the *Choosing Analyses* form.

9. Select *yes* for *Noise Separation,* if you want to plot the individual noise contributors. Fullspectrum does not support noise separation.

Noise Type = Modulated

This is the same as driven noise type = modulated (see Noise Type = Modulated on page 827). The AM and PM components of phase noise are available for plotting. Generally jitter is the preferred setting because it calculates the modulated result and it allows the calculation of jitter for the oscillator output. For examples, see Oscillators on page 944.

Noise Type = Timedomain

This provides a relative measurement of the noise at the individual points of the time-domain waveform. This is supported in both harmonic balance and in shooting. For a quantitative measurement, use the PM Jitter measurement.

1. First, set up the pss analysis as appropriate for your circuit. For more information, see the periodic steady-state analysis at the beginning of this chapter.

2. Run the pss analysis, and note the number of timepoints in the shooting window. Note that the first timepoint report in the spectre output window is for the tstab interval, and the

second is for the shooting window. The default behavior of timedomain noise is to run a pnoise analysis at each pss timepoint, which is usually prohibitive in terms of runtime. Determine a number of desired points where you want a pnoise analysis to be run. The larger the noiseskipcount, the larger the number of pss timepoints are skipped by the pnoise analysis. The number of timepoints in the tstab interval is shown in the green box below. The number of timepoints in the shooting window is shown in the red box below.

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/o:
File  Help                                                    cadence
    pss: time = 7.977 ns    (32.5 %), step = 1.364 ps     (250 m%)
    pss: time = 8.004 ns    (37.5 %), step = 1.364 ps     (250 m%)
    pss: time = 8.033 ns    (42.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.059 ns    (47.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.087 ns    (52.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.114 ns    (57.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.141 ns    (62.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.169 ns    (67.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.196 ns    (72.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.223 ns    (77.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.25 ns     (82.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.278 ns    (87.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.305 ns    (92.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.332 ns    (97.6 %), step = 1.364 ps     (250 m%)
Conv norm = 498e-03, max dI(Q0:i_excess) = -155.32 nA, took 40 ms.

Fundamental frequency is 1.83312 GHz.
Number of accepted pss steps =              4211

Notice from spectre during periodic steady state analysis `pss'.
    Trapezoidal ringing is detected during pss analysis.
        Please use method=trap for better results and performance.

Initial condition solution time: CPU = 0 s, elapsed = 613.928 us.
pss: The steady-state solution was achieved in 6 iterations.
Number of accepted pss steps =              410
Total time required for pss analysis `pss': CPU = 359.946 ms, elapsed
Time accumulated: CPU = 529.919 ms, elapsed = 3.91507 s.
Peak resident memory used = 43.2 Mbytes.


Notice from spectre.
    12 notices suppressed.

modelParameter: writing model parameter values to rawfile.
element: writing instance parameter values to rawfile.
outputParameter: writing output parameter values to rawfile.
designParamVals: writing netlist parameters to rawfile.
primitives: writing primitives to rawfile.
subckts: writing subcircuits to rawfile.

Aggregate audit (11:26:37 AM, Tue Apr 10, 2012):
Time used: CPU = 570 ms, elapsed = 4.76 s, util. = 12%.
Time spent in licensing: elapsed = 329 ms, percentage of total = 6.93%
Peak memory used = 43.3 Mbytes.
Simulation started at: 11:26:32 AM, Tue Apr 10, 2012, ended at: 11:26:
spectre completes with 0 errors, 34 warnings, and 34 notices.
23  HelpAction
```

3. On the pnoise *Choosing Analyses* form, Select *timedomain* from the *Noise Type* drop-down list.

4. Select the *Noise Skip Count* check box and type the number determined above in the text box. This example shows 5, which means run pnoise at the first timepoint, then skip 6, and run again.

5. Run the analysis.

6. In ADE, select *Results - Direct Plot - Main Form*.
7. Select *tdnoise* results.
8. Select *Integrated Output Noise*.
9. Specify the start frequency from the pnoise *Choosing Analyses* form.
10. Click *Plot.*

.

11. The integrated noise curve is plotted.

12. To determine which points on the oscillator output waveform have the most noise, plot the pss output waveform and place markers at the appropriate points.

Noise Type = Jitter

FM Jitter

This selection allows for measurement of the cycle jitter, the cycle-to-cycle jitter, and the AM and PM components of phase noise. This is an averaged noise measurement over one cycle of the oscillator.

1. First, set up the pss form as appropriate for your circuit. For more information on the pss analysis, please see the periodic steady-state section at the beginning of this chapter.

2. On the pnoise *Choosing Analyses* form, select *Jitter* from the *Noise Type* drop-down list.

3. Select the *FM* radio button.

4. Fill out the form as it would be when noisetype is set to sources. When Jitter is selected, the noise analysis will be run above and below the harmonic number selected in the

pnoise form. This is done so that the noise from above and below the oscillator output frequency can be included in the noise calculation.

5. In ADE, select *Results - Direct Plot - Main Form*.
6. Select *pnoise* in the *Analysis* section.

7. Select *Phase Noise*.
8. Select *Plot*.
9. Select *Pnoise Jitter* in the *Analysis* section.
10. Click *Plot*. Note that Phase Noise from the pnoise result set is a single sideband measurement. Phase Noise from the pnoise jitter result set is a double-sideband measurement. As such, it is usually 3dB larger than Phase Noise from the pnoise result set.

11. Select Jc or Jcc. In the modifier section, UI is Unit interval, and ppm is parts per mission.

12. Click *Plot*. The Jc or Jcc calculation is plotted as text in the waveform window.

PM Jitter

This allows for instantaneous measurement of jitter at individual voltages in the pss waveform. This is supported in both shooting and harmonic balance.

1. Set up the pss analysis as appropriate for your circuit. For more information, please see the periodic steady-state analysis at the beginning of this chapter.

2. On the pnoise *Choosing Analyses* form, select *Jitter* from the *Noise Type* drop-down list.

3. Select *PM* as the selection for *jitter: jitter measurement at the output*.

4. Specify the voltage for instantaneous noise measurement in the *Threshold Value* field.
5. Select a value from the *Crossing Direction* drop-down list.
6. Run the simulation.
7. In ADE, select *Results - Direct Plot - Main Form*.
8. Select *pnoise jitter*.

9. Select Jc or Jcc.

10. Click *Plot*. The jitter calculation appears as a label in the waveform tool.

Examples

Driven Circuit Noise Setup

On the *Choosing Analyses* form:

1. First, set up the pss analysis for your circuit. For more information see the pss section at the beginning of this chapter. Shooting is shown in the example below.

2. In the pnoise *Choosing Analyses* form, specify the output frequency range of your circuit.

3. If you use log spacing, set about 3 to 5 points per decade in the *Points Per Decade* field.

4. When Harmonic Balance is set for the pss engine, you can leave the *Maximum sideband* field blank. All the mixing products from the large-signal analysis will be considered. If the pss engine is shooting, select whether the default method or the full-spectrum method should be used. Full-spectrum is faster than the default method when many sidebands are required for your circuit and it does not require the sidebands to be set. When the default method is selected, specify the number of the desired frequency

translations in the *Maximum sideband* field. Up to 40 sidebands can be specified by default.

5. Select the resistor or port component used for the load resistance. This excludes the noise from this component for the noise figure calculation, but keeps it for the total noise output calculation.

6. If you want a noise figure calculation, you must use a port as the input for the circuit. Set the instance name.

7. The easiest way to select the reference sideband is to set it to *Select from list*.

8. Select the passband frequency range you want for the noise figure and input-referred noise calculations.

9. If you want to see the noise contributions from the individual frequencies and the different device noise calculations individually on the output of the noise analysis, enable noise separation. This is not supported for the full-spectrum method.

10. When you are done with the setup, run the simulation.

Noise Summary

The noise summary provides information about the devices that contribute noise to the output. This information is available any time a pnoise simulation completes.

In the ADE, select *Results - Print - Noise Summary*. If noise separation is enabled, there will be two choices at the very top. `Pnoise_src` has information about the noise currents at the individual noise sources in the circuit. Pnoise has information about noise at the output of the circuit.

If you want the noise contributors at a single frequency, select *spot noise*. If you want to have the noise integrated over a frequency range, select *integrated noise*.

If you want noise in volts, set the *noise unit* to *V.*

Generally, *Include All Types* should be selected. If you just want the noise from specific types of noise generators, you can select them from the list.

If you want to specifically include or exclude instances in the list, use the select button to the right side of the include instances and exclude instances fields and then select the components in the schematic.

Truncate just applies to the list. The total input and output-referred noise at the bottom of the noise summary output always includes all the noise from everything.

The figure below is the noise summary window with the sort function being set to noise contributors. Then there are three windows that show the three different sorted outputs.

An example of the noise summary output sorted by noise contributors.

Results Display Window

Window Expressions Info Help cadence

Device	Param	Noise Contribution	% Of Total
/PORT1	rn	5.74134e-06	52.27
/I33/I0/NM10	id	1.93329e-06	5.93
/I33/I0/NM12	id	1.93329e-06	5.93
I33.I0.I28.I11.rs1	rn	1.22871e-06	2.39
I33.I0.I28.I5.rs1	rn	1.22871e-06	2.39
I33.I0.I28.I11.rs2	rn	1.20374e-06	2.30
I33.I0.I28.I5.rs2	rn	1.20374e-06	2.30
/I33/I0/R0/R3	rs	1.15842e-06	2.13
/I33/I0/R7/R3	rs	1.15842e-06	2.13
/I33/I1/I1/PM5	fn	1.07333e-06	1.83
/I33/I1/I1/PM3	fn	1.07333e-06	1.83
/I33/I1/I1/PM4	fn	1.07327e-06	1.83
/I33/I1/I1/PM2	fn	1.07327e-06	1.83
/I33/I1/I0/PM5	fn	9.74707e-07	1.51
/I33/I1/I0/PM3	fn	9.74707e-07	1.51
/I33/I1/I0/PM4	fn	9.74647e-07	1.51
/I33/I1/I0/PM2	fn	9.74647e-07	1.51
/PORT2	rn	8.17354e-07	1.06
I33.I0.I28.I11.rs11	rn	5.98174e-07	0.57
I33.I0.I28.I5.rs11	rn	5.98174e-07	0.57

Integrated Noise Summary (in V) Sorted By Noise Contributors
Total Summarized Noise = 7.94156e-06
Total Input Referred Noise = 8.75341e-07
The above noise summary info is for pnoise data

37

An example of the noise summary output sorted by composite noise.

The below output is sorted by device name.

Results Display Window

Window Expressions Info Help cadence

Device	% Of Total	Inp Ref Noise Param		Noise Contribution
/PORT1	52.27	6.32625e-07	rn	5.74134e-06
			total	5.74134e-06
			ext_file_noise	0
/I33/I0/NM10	6.16	2.17227e-07	total	1.97143e-06
			id	1.93329e-06
			fn	3.80059e-07
			rs	6.65244e-08
			rd	4.89872e-09
/I33/I0/NM12	6.16	2.17227e-07	total	1.97143e-06
			id	1.93329e-06
			fn	3.80059e-07
			rs	6.65244e-08
			rd	4.89872e-09
I33.I0.I28.I11.rs1	2.39	1.35389e-07	rn	1.22871e-06
			total	1.22871e-06
			fn	0
I33.I0.I28.I5.rs1	2.39	1.35389e-07	rn	1.22871e-06
			total	1.22871e-06
			fn	0
I33.I0.I28.I11.rs2	2.30	1.32637e-07	rn	1.20374e-06
			total	1.20374e-06
			fn	0
I33.I0.I28.I5.rs2	2.30	1.32637e-07	rn	1.20374e-06
			total	1.20374e-06
			fn	0
/I33/I0/R0/R3	2.13	1.27644e-07	rs	1.15842e-06
			total	1.15842e-06
			id1	0
			id2	0
/I33/I0/R7/R3	2.13	1.27644e-07	rs	1.15842e-06
			total	1.15842e-06
			id1	0
			id2	0
/I33/I1/I1/PM5	1.84	1.12659e-07	total	1.07807e-06
			fn	1.07333e-06
			id	1.00931e-07
			rs	2.22398e-09
			rd	8.72856e-11
/I33/I1/I1/PM3	1.84	1.12659e-07	total	1.07807e-06
			fn	1.07333e-06
			id	1.00931e-07
			rs	2.22398e-09
			rd	8.72856e-11
/I33/I1/I1/PM4	1.84	1.12653e-07	total	1.07801e-06
			fn	1.07327e-06
			id	1.00927e-07
			rs	2.22392e-09
			rd	8.73227e-11
/I33/I1/I1/PM2	1.84	1.12653e-07	total	1.07801e-06
			fn	1.07327e-06
			id	1.00927e-07
			rs	2.22392e-09
			rd	8.73227e-11
/I33/I1/I0/PM5	1.52	1.02408e-07	total	9.79874e-07
			fn	9.74707e-07
			id	1.00469e-07
			rs	2.16655e-09
			rd	8.1428e-11
/I33/I1/I0/PM3	1.52	1.02408e-07	total	9.79874e-07
			fn	9.74707e-07
			id	1.00469e-07
			rs	2.16655e-09

38

For all of the noise summaries, here is a table of the abbreviations used in the Param column:

Resistor and Inductor:

Parameter	Meaning
`rn`	resistor thermal noise
`fn`	Flicker noise

Diode:

Parameter	Meaning
`id`	Diode current shot noise
`rs`	Series parasitic resistor noise

Gummel-Poon BJT:

Parameter	Meaning
`fn`	Flicker noise
`ic`	Collector current shot noise
`ib`	Base current shot noise
`rc`	Thermal noise from the collector parasitic resistor
`rb`	Thermal noise from the base parasitic resistor
`re`	Thermal noise from the emitter parasitic resistor

HiCUM:

Parameter	Meaning
`ib`	Base current shot noise
`ic`	Collector current shot noise
`rbi`	Thermal noise from the internal base parasitic resistor (See diagram below)
`rbx`	Thermal noise from the external base parasitic resistor (See diagram below)
`rcx`	Collector parasitic resistor thermal noise (See diagram below)
`re`	Emitter parasitic resistor thermal noise (See diagram below)
`fn`	Flicker noise

Figure 4-1 HiCUM Schematic Diagram

VBIC:

Parameter	Meaning
rbi	Thermal noise in resistor Rbi (See diagram below)
rbx	Thermal noise in resistor Rbx (See diagram below)
rci	Thermal noise in resistor Rci (See diagram below)
re	Thermal noise in resistor Re (See diagram below)
rs	Thermal noise in resistor Rs (See diagram below)
itzf	Collector current shot noise
ibe	Total base current shot noise
ibex	Node Bx to E base current shot noise (See diagram below)
fn	Ibe flicker noise (See diagram below)
fnx	Ibex flicker noise (See diagram below)
iccp	Parasitic transport collector to emitter current noise (See diagram below)
ibep	Parasitic transport base to emitter current noise (See diagram below)
fnp	Ibep flicker Noise (See diagram below)

Figure 4-2 VBIC Schematic Diagram

BSIM3:

Parameter	Meaning
`fn`	Flicker Noise

Parameter	Meaning
id	Thermal noise from the resistive drain current
rs	Thermal noise from the source parasitic resistor
rd	Thermal noise from the drain parasitic resistor

BSIM4:

Parameter	Meaning
fn	Flicker noise
id	Thermal noise from the resistive drain current noise
rs	Source parasitic resistor thermal noise
rd	Drain parasitic resistor thermal noise
igd	Gate-to-drain tunneling current noise
igs	Gate-to-source tunneling current noise
igb	Gate-to-bulk tunneling current noise
rbdb	Resistor thermal noise between dbNode and bNode (See diagram below)
rbpb	Resistor thermal noise between bNode and bNodePrime (See diagram below)
rbpd	Resistor thermal noise between Bulk and dbNode (See diagram below)
rbps	Resistor thermal noise between bulk and bNode prime
rbsb	Resistor thermal noise between sbNode and bNode (See diagram below)
rgbi	Gate Bias-Independent Resistor thermal noise

Figure 4-3 BSIM4 Schematic Diagrams

PSP-103

Parameter	Meaning
sfl	Channel current flicker noise
sth	Channel current thermal noise
sig	induced gate noise at 1 Hz
sigth	Induced gate noise
shotgd	Gate current shot noise from gate to drain
shotgs	Gate current shot noise from gate to source
sjnoise	Source to Bulk leakage current noise
djnoise	Drain to Bulk leakage and avalanche current noise
rgatenoise	Gate resistance thermal noise
rbulknoise	Bulk resistor thermal noise between node BP and BI (See diagram below)
rjnoise	Source side bulk resistor thermal noise between node BI and BS (See diagram below)
rjundnoise	Drain side bulk resistor thermal noise between node BI and BD (See diagram below)
rwellnoise	Well resistor thermal noise between node BI and B (See diagram below)
rsourcenoise	Source parasitic resistor thermal noise
rdrainnoise	Drain parasitic resistor thermal noise

Figure 4-4 PSP-103 Schematic Diagram

EKV 3.0

Parameter	Meaning
`fn_id`	Channel current flicker noise
`fn_ig`	Gate current flicker noise
`rs`	Source parasitic resistor thermal noise
`rd`	Drain parasitic resistor thermal noise
`rg`	Gate parasitic resistor thermal noise
`rsb`	Thermal noise for the substrate resistor connected between BSi and B. (See diagram below)
`rdb`	Thermal noise for the substrate resistor connected between BDi and B. (See diagram below)
`rbbs`	Thermal noise for the substrate resistor connected between BSi and Bi. (See diagram below where it is labeled $R_{DSB}/2$)
`rbbd`	Thermal noise for the substrate resistor connected between BDi and Bi. (See diagram below where it is labeled $R_{DSB}/2$)
`id`	Thermal noise of the channel resistor
`cor`	Induced gate noise
`shot_ig`	Gate current shot noise

Figure 4-5 EKV3.0 Schematic Diagrams

For other model types:

1. In a unix/linux shell window type `which spectre`.

   This should return a path that ends in `tools/bin/spectre`.

2. Triple-click the line with the path, and then click the center mouse button to enter it as a command.

3. Use the backspace key, and delete the word `spectre`.

4. Type `cdnshelp` at the end of the expression, and press `Enter`.

   A browser window will be displayed.

5. Close the *Tip of the day* window.

6. At the top, you should see the release number of the simulator you are using.

7. Click the plus sign (+) to the left of the MMSim folder.

8. Click the plus sign (+) to the left of the *Virtuoso Simulator Components and Device Models Reference* folder.

9. Select the device type you are using.

10. At the bottom of that device listing, you will see *Component Statements*. Double-click on this.

11. Enter the name of the parameter in the search box in the right pane of the cdnshelp window.

12. Click the down arrow to the right of the *Find* field until you see the parameter name in the pane on the right side, and then see the definition of that parameter.

Noise Figure

On the *Direct Plot Form*:

1. Select *pnoise* from the *Analysis* section.
2. Select the desired type of noise plot from the *Function* section. *Noise Figure* is single-sideband noise figure.
3. If you want the noise figure over a specified bandwidth, check the *Integrated Over Bandwidth* checkbox.
4. Specify the frequency range you want for the noise figure calculation.

5. Click *Plot*.

   The dsb noise figure and the noise figure calculation over your bandwidth are displayed.

Viewing Noise Separation Results

Viewing the Total Noise at the Output from a Single Noise Input Frequency

This capability allows the noise from all sources in the circuit at different noise input frequencies to be measured. For example, all the noise that does not change frequency (the zero sideband) can be plotted, along with the other noise input frequencies (sidebands) This plotting capability allows problem frequencies to be identified.

On the Direct Plot Form:

1. Select *pnoise separation* from the *Analysis* section.
2. To view the total noise at the output select *Sideband Output* from the *Function* section.
3. Select *V/sqrt(Hz)* or *v**2/Hz* from the *Signal Level* radio buttons.
4. Select the appropriate modifier from the *Modifier* section.

5. Select the sideband number form the *Output Sideband* list. 0 means no frequency translation.

6. Click *Plot*.

The output noise from sideband zero is plotted. This is the noise that is present at the output with no frequency translations included.

Plotting Noise from Sideband -1

On the *Direct Plot Form*:

1. Select the -1 sideband from the *Output Sideband* list. Selecting the -1 sideband means the noise frequency is shifted down in frequency by -1* first harmonic of pss from the frequency that is specified as the noise output frequency in the *Choosing Analyses*

form. For example, if the pss frequency is 1.9G, and the noise output frequency is 1M, the noise input frequency is -1.899GHz.

2. Click *Plot*.

As a suggestion, use the *Graph - Add Label* menu option to place labels to identify the traces, as shown in the figure above.

If you select multiple sidebands to be plotted at the same time, the sideband number will be shown at the right side of the legend in the waveform window.

Viewing Total Noise at the Output from each Instance

Once the problem frequencies are identified in the plot of Sideband Output, the problem components need to be identified. From the previous plot, the zero sideband contributes the largest amount of noise to the overall output noise. Using the Instance Output allows the identification of which components in the zero sideband contribute the most noise. The total noise for all the noise mechanisms within the component is plotted.

On the *Direct Plot Form*:

1. Select *Instance Output* from the *Function* section.
2. Select *V/sqrt(Hz)* or *V**2/Hz*.
3. Select the appropriate modifier.
4. Select the sideband number. In this example, the zero sideband was determined to contribute the most noise at the output of the circuit.

5. Select the type of component to display. If you want all of the types, choose *Include All Types*.

6. Specify the number of results you want to plot.

7. Click *Plot*.

The total noise at the output for the top three components is plotted.

To see which component goes with which trace, select the plus sign (highlighted in red on the previous graphic) to the left or the text at the upper left of the display area. The legends expand, and you can see which trace is which in the Name field on the left side of the display tool. You can also either select the trace or the individual entry in the Name field and the trace and legend will highlight. The instance name is shown on the right side of the legend. You may need to change the size of the legend area by moving your mouse cursor over the line that separates the two areas in the waveform tool. The cursor will

change shape. Now click-and-hold the mouse button, move to a new location, and release the mouse button.

Viewing the Exact Noise Mechanism Within the devices

The sideband with the largest contribution was determined by selecting Sideband Output and plotting. The specific component was identified by selecting Instance Output. The specific noise mechanisms can be identified using Source output. This plots the largest individual noise mechanisms for the sideband you select.

On the *Direct Plot Form*:

1. Select *Source Output* from the *Function* section.
2. Select *V/sqrt(Hz)* or *v**2/Hz*.
3. Select the appropriate modifier.

4. Select the sideband number.
5. Select the type of component to display. If you want all of the types, choose *Include All Types*.

6. If you want to restrict to one device, put its instance name in the *include inst* field. A list of devices separated by a space is also acceptable.
7. If you want to exclude a device (or devices) enter the instance name(s) in the *exclude inst* field.
8. Specify the number of results you want to plot in the *by top* field.
9. Click *Plot.*

The exact noise mechanism within the devices is plotted.

To see which noise source goes with which trace, select the plus sign to the left or the text at the upper left of the display area. The legends will be shown individually in the Name field of the waveform tool. The device name and the individual noise contributor will be shown. You can select either the individual legend or the trace and both the legend and the trace will be highlighted. The first part is the instance name, and the second part is the specific noise mechanism within that instance. For a list of the abbreviations used

for the noise parameter, see the list that starts on page 123 (77 of 234) In the example below, fn is flicker noise, and id is the channel resistance thermal noise.

Viewing the total noise currents within the devices

This capability has limited value. All the noise currents from all the noise mechanisms within the instance are added up. No gain function is applied or are available for this measurement. It just adds up everything within that instance.

1. Select *Instance source*.
2. Select *A/sqrt(Hz)* or *A**2/Hz* in the Signal Level selection.
3. Select *Magnitude* or *dB20* in the *Modifier* section.
4. Select the sideband number.
5. If you want every noise source to be considered, select I*nclude All Types*.
6. To exclude instances, click *Select* to the right of the *exclude inst* field, and select the device instances you want to exclude from the plot.
7. To include specific instances, click *Select* to the right of the *include inst* field, and select the instances on the schematic.
8. Set the number of curves to plot by setting the Truncate function.
9. Click *Plot*.

10. The frequency response curves are displayed.

Plotting the individual noise source currents

This capability plots the individual noise currents within the device. This capability gives you the noise at the source instead of at the output of the simulation.

1. Select *Primary Source*.
2. Select *A/sqrt(Hz)* or *A**2/Hz* in the *Signal Level* selection.
3. Select *Magnitude* or *dB20* in the Modifier section.
4. Select the sideband number.
5. If you want every noise source to be considered, select I*nclude All Types*.
6. To exclude instances, click *Select* to the right of the *exclude inst* field, and select the device instances you want to exclude from the plot.
7. To include specific instances, click *Select* to the right of the *include inst.* field, and select the instances on the schematic.
8. Set the number of curves to plot by setting the Truncate function.
9. Click *Plot*.

10. The frequency response curves are displayed.

For a list of the noise parameter names, please see page 123 (77 of 234) of this chapter.

Plotting the gain from the individual noise sources to the output

This capability gives the gain from the individual noise currents to the output of the circuit.

1. Select *Source Noise Gain*.
2. Select *A/sqrt(Hz)* or *A**2/Hz* in the *Signal Level* selection.
3. Select *Magnitude* or *dB20* in the *Modifier* section.
4. Select the sideband number.
5. If you want every noise source to be considered, select *Include All Types*.
6. To exclude instances, click *Select* to the right of the *exclude inst* field, and select the device instances you want to exclude from the plot.
7. To include specific instances, click *Select* to the right of the *include inst.* field, and select the instances on the schematic.
8. Set the number of curves to plot by setting the Truncate function.
9. Click *Plot*.

The frequency response curves are displayed. For a list of the abbreviations used for the noise parameters, please see page 123 (77 of 234) of this manual.

Multiple pnoise

Consider an LNA-Mixer circuit shown below.

In the above circuit, there are two outputs that need to be measured. The first is at the LNA output, and the second is at the mixer output. The noise frequencies are different at the two outputs.

1. First, set up the pss form as appropriate for your design. For more information on the pss analysis, please see the periodic steady-state analysis at the beginning of this chapter.

2. In the pnoise *Choosing Analyses* form, select *Multiple pnoise*, and set up the noise analysis for one of the outputs. The mixer output is shown below.

3. Select the second line in the *Multiple pnoise* section and set up that analysis.

4. Run the simulation. When the simulation completes, select *Results - Direct plot - Main Form*. The single-sideband noise figure measurement for the LNA is shown below. The selection mechanism for the noise result is called Multiple Out and is highlighted below.

5. Click *Plot*.

Virtuoso (R) Visualization & Analysis XL
File Edit View Graph Axis Trace Marker Measurements Tools Window Browser Help
cadence
Layout: Auto
Subwindows: Hz -> 2.5 GHz)
Workspace: Classic
RF_APS RxChain_LNA_Mixer sch...
Periodic Noise Analysis `pnoiseOut2': freq = (2.4001 GHz -> 2.5 GHz)
Name
...A_Outp I33.LN
Integrated Noise Figure (dB, 2.4001GHz - 2.4102GHz) = 2.07253
2.1
2.075
2.05
2.025
2.0
1.975
1.95
dB (dB)
2.4
2.425
2.45
2.475
2.5
freq (GHz)
mouse L:
M:
R:
5(11)
plot new graph subwindow

6. To plot the noise figure of the overall circuit, set the *Multiple Out* selection to the Mixer output at Port2. For direct conversion, usually double-sideband noise figure is desired.

7. Click *Plot*.

The waveform tool shows the noise figure curve and the integrated noise figure to 1MHz.

Noise as a Function of Blocker Power

- Note that the input source has 2 large-signal tones set up.

- The first large-signal tone has the frequency set to the variable *fb1* and the amplitude set to *pb*.
- The second tone has the frequency set to the variable *fb2* and the amplitude set to *pb*. This is the blocker signal.

To measure the noise as a function of blocker power:

1. Apply the signals that cause the large-signal effects in the pss analysis, and then run a pnoise analysis. In this case, the LO and the RF signal should be applied in the pss analysis. Note that if the highest input frequency divided by the PSS beat frequency is above about 25, it is very likely that hb or qpss would run faster than pss because hb and qpss only calculate the actual frequencies produced by the circuit, while pss calculates all the harmonics of the beat frequency.

   - The LO is set to 2.4GHz.

   - The first RF is set to 2.5G and -24 dBm.

   - The second RF tone is disabled by setting its frequency to zero.

On the pss *Choosing Analyses* form:

1. Set up the pss analysis appropriately. For more information, see the pss section at the beginning of the chapter.
2. Sweep the RF power over an appropriate range. Generally speaking, the power should start about 20 to 40 dB below the below the 1dB gain compression point, and sweep through the power level that is needed to meet the specification for the communication system you are working on. In general, this higher level would be near the compression point, or perhaps slightly above it.

On the pnoise *Choosing Analyses* form:

1. Usually, a single frequency is run in the pnoise, when the blocker power is swept.

2. Specify the desired output frequency.

   Shooting: Because to go through the fifth harmonic of 2.5GHz requires that sidebands be set to 125, (5*25th harmonic of 100MHz), fullspectrum pnoise is chosen so the simulation time is reduced. When fullspectrum is chosen, the number of frequency translations depends on the maximum step in the pss analysis. The maximum step of the pss analysis depends on how the maximum harmonic field is set. In this case, to verify that enough sidebands are being used, first set the maximum harmonics term to a bit less than the number anticipated for pnoise. Run the simulation, and plot the noise figure. Roughly double the number of pss harmonics, and run the simulation again. If the noise figure plot did not change, you had enough to start with, and try reducing the number of harmonics in the pss analysis. Reducing the number of harmonics below 10 will have no effect on the simulation. If the noise figure increased, then double the number of harmonics again and re-run the simulation. Use the smallest number of harmonics that produces a stable pnoise result. If the default pnoise method is used, start with an estimate of the number of frequency translations you need based on the nonlinearity of the system. Set that as the number of sidebands, Divide this number by 4, and set that as the number of pss harmonics. Run the simulation, and plot the noise result. Now double the number of pss harmonics and the number of pnoise sidebands. Rerun the simulation and replot the noise result. If the noise result did not change, you had enough harmonics and sidebands to begin with. If it did change, double again, and replot. Use the smallest number of harmonics and sidebands that give a stable result.

   Harmonic Balance: Leave the *Maximum sideband* field blank, and estimate the number of required harmonics in the pss analysis. Run the simulation and plot the noise. Now increase the number of harmonics in pss by about 50%, and run the simulation again. If the noise changed, increase the pss harmonics and run again. If the noise did not change, reduce the number of pss harmonics an run again. Use the smallest number of harmonics you can in pss that maintains the stable noise output in pnoise.

3. The rest are set as the preceding section.

4. Run the simulation when you are done.

5. When the simulation completes, select *Results - Direct Plot - Main Form* in the ADE.

6. Select *pnoise*.
7. Select the desired noise figure measurement.
8. Click *Plot*.

The resulting plot appears.

9. To make the plot more useful, right-click one of the numbers on the X Axis and select *Swap Sweep Var* from the context menu.

The *Swap Sweep Var* window is displayed. Note that the *Sweep Variables* field is set to *prf*, which is the variable that was swept.

10. Click *OK*.

A new subwindow is displayed with the noise figure on the Y axis, and the blocker power on the X Axis.

Noise Type = Modulated

Modulated allows the measurement of the AM and PM components of noise separately. This can be applied to any driven circuit or oscillator. This section is for driven circuits. See the oscillator section if you have an oscillator. Note that this is an averaged measurement over the full cycle of the pss waveform, not an instantaneous measurement at a single threshold voltage.

First, set up the pss analysis as appropriate for your circuit. For more information, refer to the section Overview of Periodic Steady-State (pss) Analysis on page 505.

On the pnoise *Choosing Analyses* form:

1. Set up an appropriate frequency range of interest.

Shooting: Set maximum sideband up to 4 times the number of pss harmonics. Up to 40 sidebands can be set using the defaults. Fullspectrum is not available for harmonic balance.

2. **Harmonic balance**: Leave the *Maximum sideband* field blank. This calculates noise translations for all the harmonics in the pss-harmonic balance analysis.

3. Select the output net or nets in the schematic. If the *Negative Output Node* field is left blank, the global ground net is used as the entry.

4. Select *modulated* from the *Noise Type* drop-down list.

5. Run the simulation. Note that modulated runs both the positive and negative frequencies. These frequencies are above and below the harmonic specified in the pnoise *Choosing Analyses* form.

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/4_
File  Help                                                    cādence

    mod1: freq = 314 kHz        (10 %), step = 115.9 kHz        (5 %)
    mod1: freq = 314 kHz        (10 %), step = 115.9 kHz        (5 %)
    mod1: freq = 497.6 kHz      (15 %), step = 183.6 kHz        (5 %)
    mod1: freq = 788.7 kHz      (20 %), step = 291.1 kHz        (5 %)
    mod1: freq = 1.25 MHz       (25 %), step = 461.3 kHz        (5 %)
    mod1: freq = 1.981 MHz      (30 %), step = 731.1 kHz        (5 %)
    mod1: freq = 3.14 MHz       (35 %), step = 1.159 MHz        (5 %)
    mod1: freq = 4.976 MHz      (40 %), step = 1.836 MHz        (5 %)
    mod1: freq = 7.887 MHz      (45 %), step = 2.911 MHz        (5 %)
    mod1: freq = 12.5 MHz       (50 %), step = 4.613 MHz        (5 %)
    mod1: freq = 19.81 MHz      (55 %), step = 7.311 MHz        (5 %)
    mod1: freq = 31.4 MHz       (60 %), step = 11.59 MHz        (5 %)
    mod1: freq = 49.76 MHz      (65 %), step = 18.36 MHz        (5 %)
    mod1: freq = 78.87 MHz      (70 %), step = 29.11 MHz        (5 %)
    mod1: freq = 125 MHz        (75 %), step = 46.13 MHz        (5 %)
    mod1: freq = 198.1 MHz      (80 %), step = 73.11 MHz        (5 %)
    mod1: freq = 314 MHz        (85 %), step = 115.9 MHz        (5 %)
    mod1: freq = 497.6 MHz      (90 %), step = 183.6 MHz        (5 %)
    mod1: freq = 788.7 MHz      (95 %), step = 291.1 MHz        (5 %)
    mod1: freq = 1.25 GHz      (100 %), step = 461.3 MHz        (5 %)
Total time required for pnoise analysis `mod1': CPU = 319.951 ms, elap
Time accumulated: CPU = 759.883 ms, elapsed = 2.28462 s.
Peak resident memory used = 37.5 Mbytes.


*************************************************************
Periodic Noise Analysis `mod2': freq = (-125 kHz -> -1.25 GHz)
*************************************************************
Using the operating-point information generated by PSS analysis `pss'.
    mod2: freq = -198.1 kHz      (5 %), step = -73.11 kHz        (5 %)
    mod2: freq = -314 kHz       (10 %), step = -115.9 kHz        (5 %)
    mod2: freq = -497.6 kHz     (15 %), step = -183.6 kHz        (5 %)
    mod2: freq = -788.7 kHz     (20 %), step = -291.1 kHz        (5 %)
    mod2: freq = -1.25 MHz      (25 %), step = -461.3 kHz        (5 %)
    mod2: freq = -1.981 MHz     (30 %), step = -731.1 kHz        (5 %)
    mod2: freq = -3.14 MHz      (35 %), step = -1.159 MHz        (5 %)
    mod2: freq = -4.976 MHz     (40 %), step = -1.836 MHz        (5 %)
    mod2: freq = -7.887 MHz     (45 %), step = -2.911 MHz        (5 %)
    mod2: freq = -12.5 MHz      (50 %), step = -4.613 MHz        (5 %)
    mod2: freq = -19.81 MHz     (55 %), step = -7.311 MHz        (5 %)
    mod2: freq = -31.4 MHz      (60 %), step = -11.59 MHz        (5 %)
    mod2: freq = -49.76 MHz     (65 %), step = -18.36 MHz        (5 %)
    mod2: freq = -78.87 MHz     (70 %), step = -29.11 MHz        (5 %)
    mod2: freq = -125 MHz       (75 %), step = -46.13 MHz        (5 %)
    mod2: freq = -198.1 MHz     (80 %), step = -73.11 MHz        (5 %)
    mod2: freq = -314 MHz       (85 %), step = -115.9 MHz        (5 %)
    mod2: freq = -497.6 MHz     (90 %), step = -183.6 MHz        (5 %)
13
```

Now plot the output noise using the *Direct Plot Form*.

On the *Direct Plot Form*:

1. Select pnoise.
2. Select *Output Noise* from the *Function* section.

3. Click *Plot*.
4. Select *pnoise modulated*.

5. Select *AM* or *PM* from the *Noise Type* radio buttons.

6. Click *Plot*.

On the plot, note that the AM noise is 3dB larger than the output noise. 3dB is a linear factor of 1.414. This is because the output noise is derived from the positive frequencies only (single sideband), and the AM and PM noise is derived using the positive and negative frequencies (double sideband).

Noise Type = Jitter

Driven jitter is usually used for measuring the noise in a digital circuit at a specific threshold crossing.This is an instantaneous measurement at one or more points in the time-domain waveform at a single threshold voltage. Accordingly, the pss *Choose Analyses* form uses shooting, as shown below. For more information on the pss analysis, please see Overview of Periodic Steady-State (pss) Analysis on page 505. This is also supported for harmonic

balance. If harmonic balance is used, make sure you set oversample factor to 4 or 8, and use enough harmonics to capture the waveform accurately.

An appropriate threshold is needed, which can be determined by plotting the time domain waveform from the pss *Direct Plot Form*. The waveform below shows an appropriate threshold value to be 1.65V for this circuit, which is a 3.3 Volt logic circuit.

On the pnoise *Choosing Analyses* form:

1. The stop frequency is half the input frequency.
2. The start frequency is 4 decades below that.
3. About 5 points per decade is usually sufficient.
4. *Maximum sideband* should be set to 40 for shooting, and left blank for harmonic balance. Some systems need more than 40 sidebands. To verify that 40 is enough, first run 40 sidebands and plot the noise. Next, double the number of PSS harmonics and the number of pnoise sidebands and run the simulation again. If the results are the same, then 40 sidebands is enough. If the results change, you need more harmonics and sidebands. If *fullspectrum* is chosen, start with 10 harmonics and plot the noise. Then double to 20 harmonics, re-run the simulation, and re-plot the noise. If the noise did not

change, go back to 10 harmonics. If it did change, continue to increase the number of pss harmonics until the noise measurement no longer changes.

5. The *Output* field is set to *voltage*, and the *Input Source* field is set to *none* because usually, only the output jitter is needed.

6. The *Noise Type* is set to *Jitter*.

7. The threshold is set appropriately. The *Crossing Direction* option can be set to *all*, *rise* or *fall*. In the example below, it is set to *all*, which allows jitter measurements on the rising and falling edges of the digital output.

Choosing Analyses – Virtuoso® Analog Design Environn
Analysis
tran
dc
ac
noise
xf
sens
dcmatch
stb
pz
sp
envlp
pss
pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
hbsp
Periodic Noise Analysis
PSS Beat Frequency (Hz)
2.5G
Sweeptype
relative
Relative Harmonic
0
Output Frequency Sweep Range (Hz)
Start-Stop
Start
125K
Stop
1.25G
Sweep Type
Logarithmic
Points Per Decade
Number of Steps
5
Add Specific Points
Sidebands
Method
default
fullspectrum
Maximum sideband
40
When using shooting engine, default value is 7.
Output
voltage
Positive Output Node
/Out1
Select
Negative Output Node
Select
Input Source
none
Noise Type
jitter
jitter: jitter measurement at the output
PM jitter for driven circuit
Signal
Threshold Value
Crossing Direction
/Out1
1.65
all
Enabled
Options...
OK
Cancel
Defaults
Apply
Help

Now run the simulation. When the simulation completes, use the *Direct Plot Form* to plot the jitter. For driven circuits, select *Jee*, which is edge-to-edge jitter. The event time is the time in the pss waveform. In the below example, 77p is the rising edge, and 275p is the falling edge.

The output noise frequency response and the edge-to-edge jitter are displayed. The jee measurement is one standard deviation for the expected jitter.

Noise Type = Timedomain

This is used to measure the output noise voltage at several timepoints in the pss waveform. The circuit and the pss setup is the same as the setup for jitter in the section just before this one.

First set up and run the pss analysis and plot the time-domain waveform.

Position the markers at interesting timepoints, as shown below.

To plot this waveform,

1. In ADE, select *Results - Direct Plot - Main Form*.

2. Select pss.
3. In the *Sweep* section, select *time*.
4. Select the desired net in the schematic.
5. Move your mouse cursor to the point in the waveform where you want a marker, and type `m`.
6. You can move the markers by moving to the dot that highlights the position of the marker on the trace, click and hold the mouse button, move to the desired location, and release the mouse button.

On the pnoise *Choosing Analyses* form:

1. Set the *Sweeptype* to *absolute*.
2. Set the *Stop* frequency to one half the PSS frequency.
3. Set the *Start* frequency 4 decades below the *Stop* frequency.
4. Select log sweep, and set 5 points per decade.
5. For shooting, select *default* or *fullspectrum*. If you select *default*, set the number of sidebands to 40.

Some systems require more frequency translations to be used. To check for this if you used fullspectrum, double the number of pss harmonics to 20, and run the simulation again. If the noise measurement did not change, you can go back to 10 pss harmonics. If it did change, double the number of harmonics, and rerun the simulation. Use the smallest number of harmonics that produce a stable result.

When you are using the default method, double the number of sidebands to 80, and set pss harmonics to 20. Rerun the simulation, and use the above method, except both the number of sidebands and the pss harmonics need to be changed for each run.

For harmonic balance, leave the number of pnoise sidebands blank. Start with an estimate of the number of harmonics you need based on the nonlinearity of the system and the input waveform. For square wave input, you should also set oversample factor to 2 or 4. Run the simulation. and plot the result. Now double the number of pss harmonics and run the simulation again. If the noise changed, double the number of harmonics again. Use the smallest number of harmonics that produce a stable noise measurement.

1. Select the *Number of Points* check box and type 1 in the text field. The default in timedomain noise is to run a pnoise analysis at every timepoint in the pss analysis. Because noise is usually only required at several timepoints, setting the number of points to 1 causes a single point at the first timepoint in the pss analysis to be run in pnoise.
2. Select the *Add Specific Points* check box and enter the time values from the markers positioned in the waveform. Use the space character to separate the entries.
3. Run the simulation.

On the Direct Plot Form:

1. Select *tdnoise* from the *Analysis* section.

2. Select *Integ Output Noise* from the *Function* section.
3. Select *Add to Outputs*.
4. Click *Plot*.

The noise voltage is plotted as shown below. As expected, the maximum noise occurs near the 1.65 volt level of the waveform. This is shown in the right subwindow below.

To get the jitter in seconds, take the integrated noise voltage and divide by the slew rate. To determine the slew rate, place marker A and marker B near the threshold crossing at 77.3 picoseconds. The slew rate is calculated for you in the display. The slew rate is

shown in the left subwindow below, and the integrated noise is taken from the subwindow on the right.

The result is shown in the calculator below.

This compares well with the Jee calculation of 57.79 femtoseconds from the section just before this one. Note that this is one standard deviation for the jitter.

Oscillators

Noise Type = Sources

Noise type=sources is used to measure the single sideband phase noise that is averaged over the entire cycle of the oscillator waveform.

1. First, set up the pss form as shown in the following figure. For more information see the pss section at the beginning of this chapter of the user guide.

2. Next, set up the pnoise form as shown below. For the oscillator, you may need to change the start and stop frequencies and the output node. Select *Sources* from the *Noise Type* drop-down list to measure the average noise of the oscillator.

The sweep defaults to relative to the frequency of the specified harmonic of the pss analysis. The frequencies specified in the *Choosing Analyses* form are added to the frequency of the specified harmonic when *relative* is selected. In this case, the first harmonic is specified because there is no frequency multiplier on the output of the oscillator.

1. Set the relative harmonic number. For an oscillator by itself, the relative harmonic is one. For an oscillator with a prescaler and you desire the output of the prescaler, the harmonic number is one. If you want the noise of the VCO without the prescaler, then the harmonic number is the divide ratio. If you have an oscillator with a frequency multiplier and you want the noise at the output of the frequency multiplier, enter the frequency multiplication ratio. If you want the noise at the output of the VCO by itself, enter one.
2. Relative to the output frequency, specify a frequency offset.
3. Generally, three to five points per decade in a log sweep are sufficient.
4. Setting sidebands depends on whether shooting or harmonic balance has been chosen as the pss engine.
   a. When harmonic balance is selected, leave the maximum sideband field blank, which causes the noise translations for all the pss harmonics to be calculated. Fullspectrum is not available for harmonic balance. Experiment with the number of harmonics and oversamplefactor to find the most efficient pss-pnoise runtime. Overview of Periodic Steady-State (pss) Analysis on page 505 for more information on setting harmonics.
   b. When shooting is the pss engine, fullspectrum is available and recommended. Fullspectrum calculates the number of noise translations to include based on the maximum timestep in the pss analysis. By default, frequency translations through the 20th harmonic are taken into account by fullspectrum pnoise. In the unusual case where the noise calculation needs to go through a very high harmonic of the pss analysis, increase the number of harmonics in the pss form to the minimum number of harmonics that causes the noise result to not increase with a larger number of pss harmonics. Generally speaking, these will be switched-capacitor circuits or sampling circuits where aliasing can go through a very high harmonic of the clock. Fullspectrum on this type of circuit will run much faster than the default pnoise simulation. The larger the number of sidebands, the larger the speed differential.
5. Note that when *fullspectrum* pnoise is chosen, APS will be enabled in ADE. Fullspectrum will only be run in APS. It is not available in standard Spectre. APS requires

additional tokens or different licensing to run compared to standard Spectre. This will be shown with a pop-up window when the ADE environment is used.

6. Set *Output* to *voltage*.
7. Select the output node.
8. Set *Input Sources* to *none*. Input-referred noise is not defined for an oscillator.
9. If you want noise separation, you must first select *sources* from the *Noise Type* drop-down list and then you can select the noise separation. Noise separation allows plotting of where specifically the noise is coming from.
10. Run the simulation.
11. In the ADE, select *Results* - *Direct Plot* - *Main Form*.

On the *Direct Plot Form*:

1. Select *pnoise* from the *Analysis* section.

2. Select *Phase Noise* from the *Function* section.

   Note that the phase noise from the pnoise result set plots the noise that is in the frequencies above the oscillator frequency only. If the double sideband phase noise, (which includes the noise from below and above the oscillator output frequency) is desired, set the *Noise Type* to *jitter*. This is documented later in this chapter.

3. Click *Plot*.

Note that the phase noise continues above 0dBc/Hz, which indicates that the phase noise is larger than the amplitude of the oscillator output. This is because pnoise is a small-signal analysis and it does not recognize the large-signal limits. The *CorFreq* label gives the frequency where the phase noise curve levels off at low frequency offset.

Noise Separation

One of the benefits of simulation is that since the noise terms are all calculated individually, we can store the data in a way to make these individual calculations available for viewing. To see the different noise separations, select *yes* for *Noise Separation* in the pnoise *Choosing Analyses* form.

The default in pnoise is to calculate all the noise translations that are specified in the *Maximum sideband* property in the *Choosing Analyses* form, add all these contributions

at the output, and display that total noise term. Enabling noise separation allows you to see which components and/or which frequencies contribute the most noise at the output. Once you know where the noise is coming from, you can design a solution to the problem.

Noise separation is available for shooting and harmonic balance, but not for full spectrum shooting pnoise.

When the choice is made and the simulation is run, select *Results - Direct Plot - Main Form* in ADE.

Viewing the total noise at the output from each sideband

This allows the measurement of the noise at the output from the individual noise input frequencies in the circuit.

When noise separation is set to *yes* in the pnoise *Choosing Analyses* form, you can also plot subsets of the total noise in order to identify where the noise is coming from. In the *Direct Plot Form*, select the *pnoise separation* result set.

To view the total noise from mixing with different pss harmonics:

1. Select *Sideband Output*.

   Remember that the zero sideband means that the noise input is the same frequency as the noise output. Selecting the -1 sideband means that the noise that is below the output frequency that mixes with the first harmonic is plotted. -2 means that the noise frequency is below the noise output frequency, and it mixes with the second harmonic of the pss analysis. +1 means that the noise frequency is above the noise output frequency, and mixes with the first harmonic of the pss analysis.

   Example:

   In this case, the PSS frequency is 1.9G. In the pnoise form, *relative* is selected, and the relative harmonic number is 1. This means that the output noise just above the first harmonic frequency is being analyzed. When the noise output frequency is 1MHz, the zero sideband is 1MHz above the first harmonic of the pss analysis. The -1 sideband is (-1 * 1.9GHz) + 1MHz + 1.9GHz), or 1MHz. The +1 sideband is at (+1 * 1.9G) + 1MHz + 1.9GHz, or 3.801GHz.

2. Select *V / sqrt(Hz)* or *V** 2 / Hz*.

3. Select *Magnitude* or *dB20* in the *Modifier* section.

4. Select one or more sidebands from the *Direct Plot Form*.

5. Click *Plot*. The results display in the waveform tool. A vertical marker has been placed so the individual results at 1MHz can be read out.

6. To see the individual contributions, select the legend at the upper left for the sideband number you want to see. To see the noise at the output with no frequency translation, select the zero sideband as shown below. The legend and the curve will highlight.

7. To see the noise that is upconverted from near zero frequency, select the -1 sideband. Selecting the -1 sideband means that the noise frequency is one times the pss frequency

(the oscillator frequency) below the noise output frequency. This includes the 1/F noise that upconverts to the output.

8. To see the noise that appears by mixing with the second harmonic, select the -2 sideband.

9. To see the noise converted by mixing with the third harmonic, select the -3 sideband.

Viewing the total noise at the output for all the noise mechanisms in each noise device for one noise sideband

This capability allows you to plot the total noise from each device at each separate input frequency. This gives you an idea of not only the frequencies that are troublesome, but also the specific devices that are troublesome.

1. Select Instance Output.

2. Select *V / sqrt(Hz)* or *V**2 / Hz*.

3. Select linear *Magnitude*, or *dB20*.
4. Select only one noise sideband.
5. To plot the noise from all the different device types, select *Include All Types*.
6. To make individual selections, choose each individual device type in the list. To select more than one device type at a time, press and hold the <Ctrl> key while you choose an additional category.
7. To include only the devices you select from the schematic included, select I*nclude None*, and click the *Select* button to the right of *Include Inst*. Now select the instances you want to include from the schematic. As you select the instances, the instance name will appear in the list. Select to the right of *Include inst*. Now select the instances you want in the schematic. As you click the instances, the instance Name will appear in the list.
8. In the Truncate field, type in the number of traces you want to view. The instances with the largest noise contributions will be plotted.

The highlighted trace above is a plot of all the noise contributors in *Q0* at the output of the circuit.

To plot the individual noise contributors form each instance from a single noise sideband

This plot splits the noise into each specific noise generator at each individual noise input frequency.

1. Select Source Output.

2. Select *V / sqrt(Hz)* or *V**2 / Hz*.

3. Select linear *Magnitude*, or *dB20*.
4. Select only one noise sideband.
5. To plot the noise from all the different device types, select *Include All Types*.
6. To make individual selections, choose each individual device type in the list. To select more than one device type at a time, press and hold the <Ctrl> key while you choose an additional category.
7. To include only the devices you select from the schematic included, select *Include None*, and click the *Select* button to the right of *Include Inst*. Now select the instances you want to include from the schematic. As you select the instances, the instance name will appear in the list. Select to the right of *Include inst*. Now select the instances you want in the schematic. As you click the instances, the instance Name will appear in the list.
8. In the Truncate field, type in the number of traces you want to view. The instances with the largest noise contributions will be plotted.

The instance name and the noise parameter are displayed in the waveform tool. See the red square in the above figure. For a table of the noise parameters, see page 123 (77 of 234) of this document.

Viewing the total noise current from each instance at the source of the noise

Instance source changes the focus from the output to the noise source itself, and plots the actual noise currents at the source for each individual noise frequency. All the noise from all the individual noise sources combined for each device is available.

1. In the *Direct Plot Form*, select *Instance Source*.

2. Select *V / sqrt(Hz)* or *V**2 / Hz*.

3. Select linear *Magnitude*, or *dB20*.
4. Select only one noise sideband.
5. To plot the noise from all the different device types, select *Include All Types*.
6. To make individual selections, choose each individual device type in the list. To select more than one device type at a time, press and hold the Control key while you choose an additional category.
7. To include only the devices you select from the schematic included, select *Include None*, and click the *Select* button to the right of *Include Inst*. Now select the instances you want to include from the schematic. As you select the instances, the instance name will appear in the list. Select to the right of *Include inst*. Now select the instances you want in the schematic. As you click the instances, the instance Name will appear in the list.
8. In the *Truncate* field, type in the number of traces you want to view. The instances with the largest noise contributions will be plotted. The total noise from all sources in Q0 at 1MHz offset is about 84 picoAmps per root hertz.

Viewing the individual noise current from each noise source within each instance at the source of the noise

This splits out the individual noise currents at the source for each individual noise source in the circuit.

1. In the *Direct Plot Form*, select *Primary Source*.

2. Select *V / sqrt(Hz)* or *V**2 / Hz*.

3. Select linear *Magnitude*, or *dB20*.
4. Select only one noise sideband.
5. To plot the noise from all the different device types, select *Include All Types*.
6. To make individual selections, choose each individual device type in the list. To select more than one device type at a time, press and hold the Control key while you choose an additional category.
7. To include only the devices you select from the schematic included, select *Include None*, and click the *Select* button to the right of Include Inst. Now select the instances you want to include from the schematic. As you select the instances, the instance name will appear in the list. Select to the right of *Include inst*. Now select the instances you want in the schematic. As you click the instances, the instance Name will appear in the list.
8. In the *Truncate* field, type in the number of traces you want to view. The instances with the largest noise contributions will be plotted. The individual noise from all the different sources in Q0 at 1MHz offset is shown below. To compare the noise with the total noise

shown in the previous section, add the noise power (not the voltage) from the individual noise sources by taking the square root of the sum of the squares.

Viewing the transfer function from the individual sources of the noise to the output of the circuit

1. In the *Direct Plot Form*, select *Src Noise Gain*.

2. Select *V / sqrt(Hz)* or *V**2 / Hz*.

3. Select linear *Magnitude*, or *dB20*.
4. Select only one noise sideband.
5. To plot the noise from all the different device types, select *Include All Types*.
6. To make individual selections, choose each individual device type in the list. To select more than one device type at a time, press and hold the Control key while you choose an additional category.
7. To include only the devices you select from the schematic included, select *Include None*, and click the *Select* button to the right of *Include Inst*. Now select the instances you want to include from the schematic. As you select the instances, the instance name will appear in the list. Select to the right of *Include inst*. Now select the instances you want in the schematic. As you click the instances, the instance Name will appear in the list.
8. In the *Truncate* field, type in the number of traces you want to view. The instances with the largest noise contributions will be plotted. The gain from all the noise sources to the output is plotted. Only the noise sources from Q0 are visible. For a table that lists the noise parameter for each device type, please see page 123 (77 of 234) of this document.

Commonly Used Pnoise Options

augmented=yes: is the default and is required for most oscillators that have long time constants, perhaps from a Low Drop-Out (LDO) regulator. Either take the default or set *augmented* to *yes* for every circuit.

Lorentzian

There are two ways of looking at phase noise. One way is to think of measuring the spectral output of the oscillator with a network analyzer. In this case, as the frequency gets close to the oscillator frequency, the amplitude starts going up, levels off, and then drops as the oscillator frequency is passed. In this case, the noise cannot be larger than the oscillations themselves, which is 0 dBc. If this is how you view phase noise, set the *lorentzian* option to *yes*.

If you think in terms of jitter, in one cycle of the oscillator output, because of the noise, you can calculate one standard deviation of the timing. If you go two cycles, and you integrate the noise with respect to time, you get twice as much noise power, or about 1.4 times the noise voltage or jitter. When you get to an infinite number of oscillator cycles, the jitter also becomes infinite. As you increase the time, you are lowering the noise frequency, and in this case, as the time becomes large, the frequency becomes very low. At infinite time, you get infinite jitter, and this occurs at zero offset from the carrier. Jitter is just another way of looking at phase noise. In other words, the phase noise can easily exceed the carrier amplitude (0 dBc) as the offset frequency becomes small because it is heading to infinity at zero frequency offset. If this is how you view phase noise, set the *lorentzian* option to *no*.

lorentzian=yes means calculate the leveling off. **lorentzian=cornerfreqonly** calculates the continuously rising phase noise, but it places a marker on the phase noise curve at the frequency where the phase noise would level off. *no* causes the phase noise to continue to rise as the offset frequency approaches zero.

Comparison Between Lorentzian Yes and No

As noted above, lorentzian causes leveling off of the phase noise at low frequency.

Note that when *lorentzian=no*, the phase noise curve goes above 0 dBc. This would mean that the noise is larger than the amplitude of the oscillator, which is not physical. This is because pnoise is a small-signal analysis that does not recognize the large-signal limits of

the circuit like clipping. It is similar to AC where you can specify an input magnitude of 1 megavolt, and see 10 megavolts on the output.

Cyclo2txtfile=yes: Setting this option causes pnoise to be run with all the noise frequencies and sidebands specified in the *Choosing Analyses* form. It creates a text file with the sorted frequencies and the volts-squared per hertz values that result for your circuit. This file is in the psf directory, and is called *pnoise.pnoise.data*.

To locate the `psf` directory, first select *Setup - Simulator/Directory/Host* in ADE. The path to the simulation directory is shown in the `Project Directory` field. In that directory,

browse to the `simulation/<Circuit_Name>/spectre/<View_Name>/psf` directory. View_Name is typically schematic or config. Do a list, and look for the file pnoise.pnoise.data.

```
Terminal
File  Edit  View  Terminal  Tabs  Help
artistLogFile                pnoise.-7.xfersrcs.pnoise
designParamVals.info         pnoise.7.xfersrcs.pnoise
element.info                 pnoise.-8.xfersrcs.pnoise
logFile                      pnoise.8.xfersrcs.pnoise
modelParameter.info          pnoise.-9.xfersrcs.pnoise
outputParameter.info         pnoise.9.xfersrcs.pnoise
pnoise.0.xfersrcs.pnoise     pnoise.pnoise
pnoise.-10.xfersrcs.pnoise   pnoise.pnoise.data
pnoise.10.xfersrcs.pnoise    pnoise.sources.pnoise
pnoise.-1.xfersrcs.pnoise    pnoise.xfersrcs.pnoise
pnoise.1.xfersrcs.pnoise     primitives.info.primitives
pnoise.-2.xfersrcs.pnoise    pss.fd.pss
pnoise.2.xfersrcs.pnoise     pss.td.pss
pnoise.-3.xfersrcs.pnoise    pss.tran.pss.dsn
pnoise.3.xfersrcs.pnoise     pss.tran.pss.trn
pnoise.-4.xfersrcs.pnoise    runObjFile
pnoise.4.xfersrcs.pnoise     simRunData
pnoise.-5.xfersrcs.pnoise    spectre.out
pnoise.5.xfersrcs.pnoise     subckts.info.subckts
pnoise.-6.xfersrcs.pnoise    variables_file
pnoise.6.xfersrcs.pnoise
lnx-rdavis3>
lnx-rdavis3> cp pnoise.pnoise.data ~
```

If you intend to keep this file for use in a voltage source with the noise file set to a name, make sure to copy that file to another directory to save it. The psf directory contents are deleted at the beginning of each simulation.

For more information regarding the other pnoise options, refer to the commonly used pnoise options section starting on page 81 (35 of 242) (of this document.

Noise Type = Modulated

Modulated allows the measurement of the AM and PM components of noise, along with the noise response above and below the harmonic number specified in the pnoise *Choosing Analyses* form.

1. First set the pss form, as shown in the following figure.

You need an estimate of the oscillation frequency and the number of harmonics for your circuit. For more information regarding the pss setup, see Overview of Periodic Steady-State (pss) Analysis on page 505.

2. Next set up the pnoise form.

3. Select *modulated* from the *Noise Type* drop-down list.

Note that the rest of the fields are the same as for when noise=sources as shown in the preceding section.

4. Run the simulation.

Note that positive and negative frequencies are run in pnoise. Since *relative* was chosen for the *Sweep Type*, the frequencies above and below the first harmonic are analyzed for noise.

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/o
File  Help                                                    cadence
    mod1: freq = 3.162 kHz    (43.8 %), step = 1.384 kHz    (3.12 %)
    mod1: freq = 5.623 kHz    (46.9 %), step = 2.461 kHz    (3.12 %)
    mod1: freq = 10 kHz         (50 %), step = 4.377 kHz    (3.12 %)
    mod1: freq = 17.78 kHz    (53.1 %), step = 7.783 kHz    (3.12 %)
    mod1: freq = 31.62 kHz    (56.2 %), step = 13.84 kHz    (3.12 %)
    mod1: freq = 56.23 kHz    (59.4 %), step = 24.61 kHz    (3.12 %)
    mod1: freq = 100 kHz      (62.5 %), step = 43.77 kHz    (3.12 %)
    mod1: freq = 177.8 kHz    (65.6 %), step = 77.83 kHz    (3.12 %)
    mod1: freq = 316.2 kHz    (68.8 %), step = 138.4 kHz    (3.12 %)
    mod1: freq = 562.3 kHz    (71.9 %), step = 246.1 kHz    (3.12 %)
    mod1: freq = 1 MHz          (75 %), step = 437.7 kHz    (3.12 %)
    mod1: freq = 1.778 MHz    (78.1 %), step = 778.3 kHz    (3.12 %)
    mod1: freq = 3.162 MHz    (81.2 %), step = 1.384 MHz    (3.12 %)
    mod1: freq = 5.623 MHz    (84.4 %), step = 2.461 MHz    (3.12 %)
    mod1: freq = 10 MHz       (87.5 %), step = 4.377 MHz    (3.12 %)
    mod1: freq = 17.78 MHz    (90.6 %), step = 7.783 MHz    (3.12 %)
    mod1: freq = 31.62 MHz    (93.8 %), step = 13.84 MHz    (3.12 %)
    mod1: freq = 56.23 MHz    (96.9 %), step = 24.61 MHz    (3.12 %)
    mod1: freq = 100 MHz       (100 %), step = 43.77 MHz    (3.12 %)
Total time required for pnoise analysis `mod1': CPU = 329.95 ms, elaps
Time accumulated: CPU = 829.873 ms, elapsed = 2.24678 s.
Peak resident memory used = 40 Mbytes.


*************************************************************************
Periodic Noise Analysis `mod2': freq = 1.83312 GHz + (-1 Hz -> -100 MH
*************************************************************************
Using the operating-point information generated by PSS analysis `pss'.
    mod2: freq = -1.778 Hz    (3.12 %), step = -778.3 mHz   (3.12 %)
    mod2: freq = -3.162 Hz    (6.25 %), step = -1.384 Hz    (3.12 %)
    mod2: freq = -5.623 Hz    (9.38 %), step = -2.461 Hz    (3.12 %)
    mod2: freq = -10 Hz       (12.5 %), step = -4.377 Hz    (3.12 %)
    mod2: freq = -17.78 Hz    (15.6 %), step = -7.783 Hz    (3.12 %)
    mod2: freq = -31.62 Hz    (18.8 %), step = -13.84 Hz    (3.12 %)
    mod2: freq = -56.23 Hz    (21.9 %), step = -24.61 Hz    (3.12 %)
    mod2: freq = -100 Hz        (25 %), step = -43.77 Hz    (3.12 %)
    mod2: freq = -177.8 Hz    (28.1 %), step = -77.83 Hz    (3.12 %)
    mod2: freq = -316.2 Hz    (31.2 %), step = -138.4 Hz    (3.12 %)
    mod2: freq = -562.3 Hz    (34.4 %), step = -246.1 Hz    (3.12 %)
    mod2: freq = -1 kHz       (37.5 %), step = -437.7 Hz    (3.12 %)
    mod2: freq = -1.778 kHz   (40.6 %), step = -778.3 Hz    (3.12 %)
    mod2: freq = -3.162 kHz   (43.8 %), step = -1.384 kHz   (3.12 %)
    mod2: freq = -5.623 kHz   (46.9 %), step = -2.461 kHz   (3.12 %)
    mod2: freq = -10 kHz        (50 %), step = -4.377 kHz   (3.12 %)
    mod2: freq = -17.78 kHz   (53.1 %), step = -7.783 kHz   (3.12 %)
    mod2: freq = -31.62 kHz   (56.2 %), step = -13.84 kHz   (3.12 %)
    mod2: freq = -56.23 kHz   (59.4 %), step = -24.61 kHz   (3.12 %)
14
```

5. In the ADE window, select *Results - Direct Plot - Main Form*.
6. On the *Direct Plot Form,* plot the phase noise from the *pnoise* results.

7. Next, plot the phase noise from the *pnoise jitter* result set.

Note that the phase noise from the pnoise result contains the noise in the frequencies above the oscillator frequency only, and phase noise from the pnoise jitter results contain

the noise power from above and below the oscillator frequency. This is the reason for the 3.01 dB difference.

8. Select *pnoise modulated* from the *Analysis* section.

9. Select *AM* or *PM* from the *Noise Type* section.

   Note that the AM and PM noise are also double sideband measurements. This can be seen in the result by noting that the phase noise from the pnoise jitter result set is equal to the PM component of phase noise.

10. Click *Plot*.

To plot cycle jitter or cycle-to-cycle jitter

1. On the *Direct Plot Form*, select *pnoise jitter* from the *Analysis* section.

2. Select *Jc* or *Jcc* from the *Function* section.

   The frequency range will default to the range set in the *Choose Analyses* form.

3. Click *Plot*.

The jc or jcc calculation is shown as a label that is added to the waveform tool.

Noise Type = Jitter (FM)

Noise Type = Jitter (FM) produces the same output as Noise Type = Modulated. See Noise Type = Modulated on page 978.

Noise Type = Jitter (PM)

This analysis gives an instantaneous measurement of jitter at a threshold voltage on the timedomain waveform. This is useful for measuring the jitter of an oscillator that is driving a digital component.

1. First, set up the pss analysis. For more information on this, see Overview of Periodic Steady-State (pss) Analysis on page 505.

2. Next set up the pnoise *Choosing Analyses* form, as shown below.

In the *Choosing Analyses* form:

- Select *jitter* from the *Noise Type* drop-down list.
- Select the *PM* radio button.
- Specify a threshold voltage for the time-domain waveform in the *Threshold Value* field.
- Specify a crossing direction.
- The stop frequency should be very close to half the oscillator frequency.
- The start frequency should be approximately the frequency where the phase noise reaches 0 dBc.

3. Run the simulation.

4. In the ADE, select *Results - Direct Plot - Main Form*.

On the *Direct Plot Form*, first plot the pss time-domain result.

Now plot the cycle jitter (Jc) or the cycle-to-cycle jitter.

Note that there is a selection of event times if you selected *all* for *Crossing Direction*. This is the time on the waveform where the calculation was made.

Following are the calculations for the falling edge and the rising edge.

Noise Summary

The noise summary provides information about the devices that contribute noise to the output. This information is available any time an pnoise simulation completes.

In the ADE, select *Results - Print - Noise Summary*. If noise separation is enabled, there will be two choices at the very top. *pnoise_src* has information about the noise currents at the individual noise sources at the noise source. *pnoise(pmjitter)* has information about noise at the output of the circuit.

If you want the noise contributors at a single frequency, select *spot noise*. If you want to have the noise integrated over a frequency range, select *integrated noise*.

If you want noise in volts, set the *noise unit* to *V.*

Generally, *Include All Types* should be selected. If you just want the noise from specific types of noise generators, you can select them from the list.

Truncate just applies to the list. The total input and output-referred noise at the bottom of the noise summary output always includes all the noise from everything in the circuit.

Jittereventtime - This is a list of the times in the PSS waveform where the noise calculations were made. To see whether the times are in the PSS waveform, plot the time-domain waveform from PSS. In many cases, only a single time is in the list.

The figure below is the noise summary window with the sort being noise contributors.

The output below shows the nosie summary sorted by noise contributors. The largest noise contributor is on the top of the list.

Results Display Window

Window Expressions Info Help cadence

Device	Param	Noise Contribution	% Of Total
/I77/I11/M2	fn	5.33	12.18
/I77/I11/M1	fn	3.83091	6.29
/I77/I10/M2	fn	3.60685	5.58
/I77/I2/M2	fn	3.37504	4.88
/I77/I8/M2	fn	3.37319	4.88
/I77/I6/M2	fn	3.37301	4.88
/I77/I4/M2	fn	3.37261	4.88
/I77/I7/M2	fn	3.37105	4.87
/I77/I5/M2	fn	3.37076	4.87
/I77/I3/M2	fn	3.36834	4.86
/I77/I9/M2	fn	3.35622	4.83
/I77/I1/M2	fn	3.29032	4.64
/I77/I1/M1	fn	3.18047	4.34
/I77/I6/M1	fn	2.6974	3.12
/I77/I7/M1	fn	2.69718	3.12
/I77/I8/M1	fn	2.69717	3.12
/I77/I9/M1	fn	2.69714	3.12
/I77/I5/M1	fn	2.69685	3.12
/I77/I4/M1	fn	2.6906	3.10
/I77/I3/M1	fn	2.6883	3.10

```
Spot Noise Summary (in V/sqrt(Hz)) at 1K Hz Sorted By Noise Contributors
Total Summarized Noise = 15.2717
No input referred noise available
The above noise summary info is for pnoise(pmjitter) data with  jittereventtime = 1.39119e-11
```

7

For the abbreviations used in the Param column, refer to the table under the section Noise Summary on page 866.

Noise Type = Timedomain

This analysis type provides a relative measurement of the noise at each time in the pss output waveform. Note that if you have a lot of harmonics and/or a large oversample factor, there will

be many timepoints in the ifft in harmonic balance. In the shooting output, there are at least 200 timepoints. A method will be presented to reduce the number of pnoise points.

1. First, set up the pss analysis for the oscillator. For more information, see Overview of Periodic Steady-State (pss) Analysis on page 505.

2. Next, set up the pnoise analysis, as shown below.

3. In the pnoise *Choosing Analyses* form:

- The stop frequency should be near one half the frequency of the oscillator.
- The start frequency should be near the point where the phase noise curve reads 0dBc.
- Select *timedomain* from the *Noise Type* drop-down list.
- The default *Noise Skip Count* is 0 (zero).

A pnoise run will always be run at time = 0. A noise skip count of 2 means run the timepoint at 0 seconds, then skip 2 pss timepoints, then run the next one, then skip 2, and so on. Pick an appropriate number for noise skip count so that the pnoise runtime is not excessive. In harmonic balance, there will be 2*number of harmonics*oversample

timepoints. In shooting, first run pss by itself and then you can read from the Spectre output file how many timepoints there are.

```
/net/lnx-rdavis3/usr2/rdavis/demos/ic61/docs_database/simulation/o
File  Help                                                    cadence

=========================================
`pss': time = (7.7999 ns -> 8.34542 ns)
=========================================
    pss: time = 7.814 ns    (2.54 %), step = 1.364 ps     (250 m%)
    pss: time = 7.841 ns    (7.54 %), step = 1.364 ps     (250 m%)
    pss: time = 7.868 ns    (12.5 %), step = 1.364 ps     (250 m%)
    pss: time = 7.896 ns    (17.5 %), step = 1.364 ps     (250 m%)
    pss: time = 7.923 ns    (22.5 %), step = 1.364 ps     (250 m%)
    pss: time = 7.95 ns     (27.5 %), step = 1.364 ps     (250 m%)
    pss: time = 7.977 ns    (32.5 %), step = 1.364 ps     (250 m%)
    pss: time = 8.004 ns    (37.5 %), step = 1.364 ps     (250 m%)
    pss: time = 8.033 ns    (42.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.059 ns    (47.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.087 ns    (52.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.114 ns    (57.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.141 ns    (62.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.169 ns    (67.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.196 ns    (72.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.223 ns    (77.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.25 ns     (82.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.278 ns    (87.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.305 ns    (92.6 %), step = 1.364 ps     (250 m%)
    pss: time = 8.332 ns    (97.6 %), step = 1.364 ps     (250 m%)
Conv norm = 498e-03, max dI(Q0:i_excess) = -155.32 nA, took 20 ms.

Fundamental frequency is 1.83312 GHz.
Number of accepted pss steps =              4211

Notice from spectre during periodic steady state analysis `pss'.
    Trapezoidal ringing is detected during pss analysis.
        Please use method=trap for better results and performance.

Initial condition solution time: CPU = 0 s, elapsed = 770.807 us.
pss: The steady-state solution was achieved in 6 iterations.
Number of accepted pss steps =              410
Total time required for pss analysis `pss': CPU = 339.948 ms, elapsed
Time accumulated: CPU = 449.93 ms, elapsed = 1.83729 s.
Peak resident memory used = 39.3 Mbytes.


Notice from spectre.
    12 notices suppressed.


*********************************************************************
Periodic Noise Analysis `pnoise': freq = 1.83312 GHz + (10 Hz -> 900 M
*********************************************************************
14
```

4. Run the simulation.

5. In ADE, select *Results - Direct Plot - Main Form*.
6. Plot the output waveform.

7. Now plot the integrated output noise.

The waveform tool displays both results. Markers have been placed at the largest noise points, and at the times for those maximums. Remember that this is a relative measurement

for comparing the noise voltage at different points in time only. The noise is near zero at the valleys, and near the maximum at the peaks.

The noise is highest near the zero crossing, and lowest at the peaks and valleys of the waveform.

Periodic Stability Analysis (PSTB)

The periodic stability (PSTB) analysis evaluates the local stability of a periodically time-varying feedback circuit. It is a small-signal analysis, like STB analysis, except that the circuit is first linearized about a periodically varying operating point as opposed to a simple DC operating point. Linearizing about a periodically time-varying operating point allows the stability evaluation to include the effect of the time-varying operating point.

The stability evaluation of a periodically varying circuit is a two step process.

- First, the small stimulus is ignored and the periodic steady-state response of the circuit to a possibly large periodic stimulus is computed using PSS analysis. As a normal part of the PSS analysis, the periodically time-varying representation of the circuit is computed and saved for later use.
- Finally, the small stimulus is applied to compute the loop gain of the zero sideband with a `probe` component. The local stability can be evaluated using the gain margin, phase margin, or a Nyquist plot of the loop gain. To perform PSTB analysis, you must use a `probe` instance and specify it with the `probe` parameter.

The loop-based algorithm requires that you place the `probe` on the feedback loop to identify and characterize the particular loop of interest. The introduction of the `probe` component should not change any of the circuit characteristics. Because of the time-varying properties of the circuit, the loop gain at different places might be different but you can use the loop gain at any point to evaluate stability.

The loop-based algorithm provides stability information for both single loop circuits and for multi-loop circuits in which you can place a `probe` component on a critical wire to break all loops. For a general multi-loop circuit, such a critical wire might not be available. The loop-based algorithm can only be performed on individual feedback loops to ensure they are stable.

Unlike other analyses in SpectreRF, this analysis can only sweep frequency.

Parameters for PSTB Analysis

Sweep Interval Parameters

`start=0`	Start sweep limit value.
`stop`	Stop sweep limit value.
`center`	Center value of sweep.

`span=0`	Sweep limit span of values.
`step`	Step size for a linear sweep.
`lin=50`	Number of steps for a linear sweep.
`dec`	Points per decade for a logarithmic sweep.
`log=50`	Number of steps for a logarithmic sweep.
`values=[...]`	Array of sweep values.

Probe Parameters

`probe`	Probe instance around which the loop gain is calculated.

Output Parameters

`save`	Signals to output. Possible values are `all`, `lvl`, `allpub`, `lvlpub`, `selected`, `none` or `nooutput`.
`nestlvl`	Levels of subcircuits to output.

Convergence Parameters

`tolerance`	Relative tolerance for the linear solver. The default value is $1.0e^{-9}$ for the Shooting solver and $1.0e^{-6}$ for the Harmonic Balance solver.
`gear_order=2`	Gear order used for small-signal integration.
`solver=turbo`	Solver type. Possible values are `std` or `turbo`.
`oscsolver=turbo`	Oscillator solver type. Use `ira` for large circuits. Possible values are `std`, `turbo` or `ira`.
`lnsolver=gmres`	Linear solver. Possible values are `bicgstab`, `gmres`, `qmr`, or `resgmres`.
`resgmrescycle=short`	Controls recycling and the length of the computation cycle. Possible values for large signal analysis are `instant`, `short`, and `long`. Possible values for small signal analyses are `instant`, `short`, `long`, `recycleinstant`, `recycleshort`, and `recyclelong`.

Annotation Parameters

`annotate=sweep`	Degree of annotation. Possible values are `no`, `title`, `sweep`, `status` or `steps`.
`stats=no`	The `stats` parameter is not supported. Use annotate.
`title`	Analysis title.

Sweep

You can specify sweep limits by providing:

- The end points of the sweep
- The center value and the span of the sweep
- An array of specific values to sweep

Steps can be linear or logarithmic and you can specify either the number of steps or the size of each step. You can specify a step size parameter (`step`, `lin`, `log`, `dec`) to determine whether the sweep is linear or logarithmic. If you do not provide a step size parameter, the sweep is linear when the ratio of stop to start values is less than 10:1 and logarithmic when this ratio is equal to or greater than 10:1. 100 total points will be run.

Alternatively, you may provide specific values for the `sweep` parameter using the `values` parameter. If you give both a specific set of values and a set specified using a sweep range, the two sets are merged and collated before being used. All frequencies are in Hertz.

Understanding Loop-Based and Device-Based Algorithms

Both loop-based and device-based algorithms are available for periodic small-signal stability analysis. When the `probe` parameter points to a current probe or voltage source instance, the loop-based algorithm is used; when it points to a supported active device instance, the device-based algorithm is used.

About the PSTB Loop-Based Algorithm

The PSTB loop-based algorithm is based on a subset of Nyquist criteria. The analysis outputs the loop gain waveform.

The PSTB loop-based algorithm calculates the true loop gain, which consists of both normal loop gain and reverse loop gain. The loop-based algorithm requires that you place a `probe`

component in the feedback loop to identify and characterize the particular loop of interest. Introducing the `probe` component should not change circuit characteristics. The `probe` component is an `iprobe` from *analogLib* for a single-ended circuit and a `cmdmprobe` from *analogLib* for a differential circuit.

The loop-based algorithm provides accurate stability information for single loop circuits. It also provides accurate stability information for multi-loop circuits in which you can place a `probe` component on a critical wire to break all loops. For general multi-loop circuits, such a critical wire may not be available. The loop-based algorithm can only be performed on individual feedback loops to ensure they are stable. Although the stability of all feedback loops is a necessary condition for the whole circuit to be stable, the multi-loop circuit tends to be stable if all individual loops are associated with reasonable stability margins.

Device-Based Algorithm

The device-based algorithm calculates the loop gain around a particular active device, which must be a gain instance such as a bjt or mos transistor. This algorithm is often applied to assess the stability of circuit designs in which local feedback loops cannot be neglected. The loop-based algorithm cannot be used for such designs because the local feedback loops are inside the devices where a `probe` component cannot be inserted.

When the `probe` parameter points to a particular active device, the dominant controlled source in the device is nulled during the analysis. The device-based algorithm produces accurate stability information for a circuit in which a critical active device can be identified such that nulling the dominant gain source of this device renders the whole network to be passive.

PSTB Analysis

1. In the schematic, add an *iprobe* from analogLib in series with the feedback path.
2. If it is a differential circuit, add a *cmdmprobe* from analogLib.

3. Set up a PSS analysis. For more information, please refer to Overview of Periodic Steady-State (pss) Analysis on page 505.

4. On the Choosing Analyses form, select *pstb*.

5. Type in a frequency range of interest for your application.

For slowly changing loop gain, you can set 10 points per decade. If the loop gain has quick changes (usually in oscillators) you will need more points per decade.

6. Select *iprobe* (or cmdmprobe) in the circuit.

For many applications, you might want to plot the PSS waveform.

7. You can select multiple values of the sweep by holding the <*Ctrl*> key and clicking the entries you want, or by clicking a starting member then pressing the <*Shift*> key and clicking the ending member.

8. Select the net in the schematic.

9. To see details of the waveform, position your cursor near the trace, and press and hold the <*Alt*> key. The cursor does not show in this screenshot. It is at the upper left of the box on the purple trace.

10. Note that the purple trace is approaching the clipping level on the positive side, and is just starting to clip on the negative side.

11. Now on the Direct Plot form, select *pstb*.

12. Select *Loop Gain*.

13. Select *Magnitude*.

Note: When *Mag&Phase* is selected, there is no dB20 modifier for the loop gain.

14. Select *dB20*.

15. Click *Plot*.

16. To read out the trace attributes, move your cursor near a trace, and press and hold the <*Alt*> key. The cursor is not shown in this screen shot. It is at the upper left of the readout box. The 700mV input trace is purple.

- Below 700mV input, the open loop gain is largely unaffected.
- At 700mV, the DC gain drops.
- As the clipping and slew-rate limiting get worse, the gain drops even more.

17. To plot the phase, select *Phase*.

18. Click *Plot*.

19. To view the attributes of a trace position your cursor as desired, then hold down the alt key. The cursor is not shown in this screenshot. It is at the upper left corner of the text box. The 700mV trace is in purple.

- In traditional feedback theory, generally only the forward and reverse transfer functions are considered. The assumption is that there is a summing junction with a gain of -1. Thus the oscillatory point is when the loop gain is unity and the phase is 180 degrees.
- In the circuit, we can not remove the summing junction, thus the phase that causes oscillation is zero or multiples of 360 degrees.
- This circuit has almost 90 degrees of phase margin.

20. For oscillators, add the same *iprobe* or *cmdmprobe* to the circuit.

21. Set up the PSS. For more information see Overview of Periodic Steady-State (pss) Analysis on page 505.

22. On the *Choosing Analyses* form, select *pstb*.

23. Type in a range of interest for your application.

24. Type in a reasonable number of points for your application. The more rapidly the loop gain changes, the more points you need.

25. Select the *iprobe* (or *cmdmprobe*) instance in the schematic.

26. When the simulation completes, select *Results - Direct Plot - Main Form* in the ADE.

27. Select *pstb results*.

28. Select *Loop Gain*.

29. Select *Magnitude*.

30. Select *dB20*.

31. Click *Plot*.

32. Select *Phase*.

33. Click *Plot*.

- Position the markers by moving the mouse cursor near the trace and then typing `m`.

- The markers can be repositioned by moving the cursor to the intercept point, clicking and holding the mouse button, moving the cursor, then releasing the mouse button.
- Note that at the zero degree phase point, the loop gain is almost exactly 0dB - Unity gain.
- The frequency of this point is 1.83GHz.

When the PSS result is plotted, the actual oscillating frequency is 1.833GHz. This is very close to the prediction from pstb.

Periodic Transfer Function Analysis (PXF)

A conventional transfer function (XF) analysis computes the transfer function from every source in the circuit to a single output. An XF analysis differs from a conventional AC analysis in that the AC analysis computes the response from a single stimulus to every node in the circuit.

The difference between the PXF and PAC analyses is similar. The PXF analysis computes the transfer functions from any source at any frequency to a single output at a single frequency. Like PAC analysis, PXF analysis models frequency conversion effects. This is illustrated in Figure 4-6 on page 1029.

The PXF analysis directly computes useful quantities such as:

- Conversion efficiency (the transfer function from input to output at a desired frequency)
- Image and sideband rejection (input to output at an undesired frequency)
- LO feed-through and power supply rejection (undesired input to output at all frequencies)

PXF analysis measures conversion gains, especially those from the input source to the output. It also computes the conversion gain of the specified sideband as well as various unwanted images including the baseband feed through. PXF analysis also computes the coupling from other inputs, such as the LO and the power supplies. These computations model frequency translation. PXF analysis determines the sensitivity of the output to either up-converted or down-converted noise from either the power supplies or the LO.

The output is sensitive to signals at many frequencies at the input of the mixer. The input signals are replicated and translated by each harmonic of the LO. The signals shown in Figure 4-6 on page 1029 are those that end up at the output frequency.

Computing transfer functions for a periodically varying circuit is a two step process.

- First, the small stimulus is ignored and the periodic steady-state response of the circuit to possibly large periodic stimulus is computed using PSS analysis.

  As a normal part of the PSS analysis, the periodically time-varying representation of the circuit is computed and saved for later use.

- Finally, using the PXF analysis, small-signal excitations are applied to compute the transfer functions.

Figure 4-6 Mixer Output Signals Shown by PXF Analysis

A PXF analysis cannot be used alone and must follow a PSS analysis. However, any number of periodic small-signal analyses, such as PAC, PSP, and Pnoise can follow a single PSS analysis.

Important

Unlike other SpectreRF small-signal analyses, the PXF analysis can sweep only frequency.

Parameters for PXF Analysis

Sweep Interval Parameters

`start=0`	Start sweep limit.
`stop`	Stop sweep limit.
`center`	Center of sweep.
`span=0`	Sweep limit span.
`step`	Step size for a linear sweep.
`lin=50`	Number of steps for a linear sweep.
`dec`	Points per decade.
`log=50`	Number of steps for a logarithmic sweep.
`values=[...]`	Array of sweep values.
`sweeptype=unspecified`	Specifies whether the sweep frequency range is the absolute frequency of input or whether it is relative to the port harmonics. When an unspecified value is used, SpectreRF sweeps the absolute input source for non-PSP-driven cases; for other cases, SpectreRF sweeps relative to the port harmonics. Possible values are `absolute`, `relative` or `unspecified`.
`relharmnum=1`	Harmonic to which relative frequency sweep should be referenced.

Probe Parameter

`probe`	Compute every transfer function to this probe component.

Sampled Analysis Parameters

For *Specialized Analyses* is *Sampled*

`ptvtype=timeaveraged`	Specifies whether the PTV analysis is traditional or sampled under certain conditions. Possible values are `timeaveraged` or `sampled`.
`sampleprobe`	The crossing event at this port triggers the sampled small signal computation.
`thresholdvalue=0`	Sampled measurement is made when the signal crosses this value.

`crossingdirection=all`	Specifies which transitions trigger sampling. Possible values are `all`, `rise` or `fall`.
`maxsamples=16`	Maximum number of sampled events to be processed during the sampled analysis.
`sampledelay=[...]`	Sampled measurement is made with delay from the time of the crossing event on the control signal or probe. ■ The first value corresponds to the rising edge. ■ The second value corresponds to the falling transition.
`extrasampletimepoints=[...]`	Additional time points for sampled PTV analysis.
`sampleratio=1`	The multiple times of fund frequency that sample frequency divides into.

Output Parameters

`stimuli=sources`	Stimuli used for PXF analysis. Possible values are `sources` or `nodes_and_terminals`.
`sidebands=[...]`	Array of relevant sidebands for the analysis.
`maxsideband=0`	An alternative to the `sidebands` array specification. `maxsideband=0` automatically generates the array. `[-maxsideband... 0... +maxsideband]`
`freqaxis`	Specifies whether the results should be output versus the input frequency, `in`, the output frequency, `out`, or the absolute value of the input frequency, `absin`. The default for logarithmic frequency sweeps is `out`. The default otherwise is `absin`. Possible values are `absin`, `in` or `out`.
`save`	Signals to output. Possible values are `all`, `lvl`, `allpub`, `lvlpub`, `selected`, or `none`.
`nestlvl`	Levels of subcircuits to output.

`oscout=total`	The type of output for oscillator simulation. Default value is `total` for the output of total modulation response from oscillator simulation. Other values are `pm` for the output of phase-modulation response; `am` for the output of amplitude-modulation response. Possible values are `total`, `pm` or `am`.

Convergence Parameters

`tolerance`	Relative tolerance for the linear solver. For the Shooting solver, the default value is $1.0e^{-9}$. For the Harmonic Balance solver, the default value is $1.0e^{-6}$.
`gear_order=2`	Gear order used for small-signal integration.
`solver=turbo`	Solver type. Possible values are `std` or `turbo`.
`oscsolver=turbo`	Oscillator solver type. Possible values are `std`, `turbo` or `ira`. The `ira` solver is recommended for very large circuits.
`lnsolver=gmres`	Linear solver. Possible values are `bicgstab`, `gmres`, `qmr`, or `resgmres`.
`resgmrescycle=short`	Controls recycling and the length of the computation cycle. Possible values for large signal analysis are `instant`, `short`, and `long`. Possible values for small signal analyses are `instant`, `short`, `long`, `recycleinstant`, `recycleshort`, and `recyclelong`.

Annotation Parameters

`annotate=sweep`	Degree of annotation. Possible values are `no`, `title`, `sweep`, `status`, or `steps`.
`stats=no`	Analysis statistics. Possible values are `no` or `yes`.
`title`	Analysis title.

Modulation Conversion Parameters

For *Specialized Analyses* is *Modulated*

`modulated=no`	Compute transfer functions and conversions between modulated sources and outputs. Possible values are `single`, `first`, `second`, or `no`.
`outmodharmnum=1`	Harmonic for the PXF output modulation.
`inmodharmvec=[...]`	Harmonic list for the PXF modulated sources.
`moduppersideband=1`	Index of the upper sideband included in the modulation of an input for PXF analysis or an output for PAC analysis.

Output Parameters

The output variable you measure can be voltage or current, and the variable frequency is not limited by the period of the large-periodic solution. When you sweep a selected output frequency, you can select the periodic small-signal input frequencies by specifying either one of the `maxsideband` or `sideband` parameters.

For a set of *n* integer numbers representing the sidebands

$$K_1, K_2, .. K_n$$

the input signal frequency at each sideband is computed as

$$f(in) = f(out) + k_i \times fund(pss)$$

where

- $f(out)$ represents the (possibly swept) output signal frequency
- $fund(pss)$ represents the fundamental frequency used in the corresponding PSS analysis

When you analyze a down-converting mixer and sweep the IF output frequency, $k_i = +1$ for the RF input represents the first upper sideband, and $k_i = -1$ for the RF input represents the first lower sideband. If you specify the maximum sideband value by setting the `maxsideband` value to k_{max}, you are selecting all $2 \times k_{max} + 1$ sidebands from $-k_{max}$ to $+k_{max}$.

For PXF analysis, the number of sidebands you select with `maxsideband` does not substantially increase the simulation time. However, to ensure accurate PXF analysis results, set the `maxacfreq` parameter for the corresponding PSS analysis to guarantee that

$$|max\{f(in)\}| < maxacfreq$$

otherwise, the computed solution might be contaminated by aliasing effects. The PXF simulation does not run when

$$|f(out)| > maxacfreq$$

For these extreme cases, diagnostic messages indicate how you should change the `maxacfreq` parameter value in the PSS analysis. In a majority of simulations, however, this is not an issue because the `maxacfreq` value is never allowed to be smaller than 40 times the PSS fundamental.

With PXF analysis the frequency of the stimulus and response are usually different (this is an important way in which the PXF analysis differs from the XF analysis). Use the `freqaxis` parameter to specify whether the results should be output versus the input frequency, `in`, the output frequency, `out`, or the absolute value of the input frequency, `absin`.

Probe Parameters

You can specify the output with a pair of nodes or a probe component. Any component with two or more terminals can be a voltage probe. When there are more than two terminals, they are grouped in pairs. Use the `portv` parameter to select the appropriate pair of terminals. Alternatively, you can simply specify a voltage to be the output by giving a pair of nodes on the PXF analysis statement.

Any component that naturally computes current as an internal variable, such as a voltage source, can be a current probe. If the probe component computes more than one current, you use the `porti` parameter to select the appropriate current. Do not specify both the `portv` and `porti` parameters. If you do not specify either parameter, the probe component provides a reasonable default.

The Analog Circuit Design Environment (ADE) provides the following two ways to set probes on the Choosing Analyses form:

- When you specify input or output sources
- When you specify positive or negative output nodes

Output Parameters

The `stimuli` parameter (on the PXF Options form) specifies the transfer function inputs. You select one of two choices:

- Use `stimuli` = `sources` to use the sources present in the circuit. To compensate for gains or losses in the test fixture, use the `xfmag` source component parameter to adjust the computed gain to compensate for gains or losses in a test fixture. In hierarchical netlists, limit the number of sources using the `save` and `nestlvl` parameters.
- Use `stimuli` = `nodes_and_terminals` to compute all possible transfer functions. Use this option when you cannot anticipate which transfer functions you might need to examine. This is useful when you do not know in advance which transfer functions are interesting. Note that in the ADE environment, the data can be directly plotted from sources in the circuit only. To plot the extra data in the ADE, use the results browser and calculator.

Transfer functions for nodes are computed assuming that a unit magnitude flow (current) source is connected from the node to ground. Transfer functions for terminals are computed assuming that a unit magnitude value (voltage) source is connected in series with the terminal. By default, the PXF analysis computes the transfer functions from a small set of terminals.

For transfer functions from specific terminals, specify the terminals in the `save` statement. Use the `:probe` modifier (for example, `Rout:1:probe`) or specify `useprobes=yes` on the options statement. For transfer functions from all terminals, specify `currents=all` and `useprobes=yes` on the `options` statement.

Modulation Parameters

You can make modulated small-signal measurements from the Analog Circuit Design Environment (ADE). The `modulated` option for PXF analysis and other modulated parameters are set by ADE. The PXF analysis with the `modulated` option produces results which might have limited use outside the ADE environment. The Direct Plot Form is configured to analyze modulated small-signal measurement results. Direct Plot can combine several wave forms to measure AM and PM transfer functions from single sideband or modulated stimuli to the specified output.

Sampled Small-Signal Analysis

Sampled small signal PXF and PAC analyses use the Analog Design Environment (ADE) environment. The sampled options are set by ADE. The Sampled option produces results which might have limited use outside ADE. Direct Plot is configured to analyze the results and

make sampled measurements due to single sideband or sampled stimuli. A sampled analysis is a small-signal analysis with a use model similar to the sampled (timedomain) pnoise analysis. Specifically, you first create a circuit (schematic or netlist description) and place port or source components to specify the key elements where the transfer function to the output is of interest.

The ability to sample the noise, signal slope and transfer function at a particular time point is a valuable investigative tool for design and verification. For example it will be used in a design of switched-capacitor filters, logic circuits and in the evaluation of the power supply rejection. A Sampled analysis computes the transfer function from different parts of the circuit to the output at a particular time point. The output signal is sampled at the clock rate (large signal period which is used in PSS).

A new option in the *Specialized Analysis* cyclic menu, *Sampled*, opens the *Sampled* analysis fields in the *Choosing Analyses* form. Several fields are available to specify the sampling event. First, you select a control signal to observe in search for the triggering event(s). It could be a single or differential voltage signal or a probe, either voltage or current. The threshold value and the crossing direction are the parameters of a triggering timing event. In case of special need, you can specify a delayed measurement from the time of the crossing event.

You can also specify actual time points for sampling the output. The same form will be used and either one or the combination of both approaches is used to do the sampled small signal analysis.

Swept PXF Analysis

Specify sweep limits by providing either the end points (`start` and `stop`) or by providing the center value and the span (center and `span`) of the sweep.

Specify sweep steps as `linear` or `logarithmic`. Either specify the number of steps or the size of each step. You can give a step-size parameter (`step`, `lin`, `log`, `dec`) to determine whether the sweep is `linear` or `logarithmic`. If you do not give a step-size parameter, the sweep is `linear` when the ratio of `stop` to `start` values is less than 10, and `logarithmic` when this ratio is 10 or greater.

Alternatively, use the `values` parameter to specify the particular values that the sweep parameter should take. If you provide both a specific set of values and a set of values specified using a sweep range, the two sets are merged and collated before being used. All frequencies are in Hertz.

Conversion Gain Measurement

1. Set the frequencies of the RF inputs to zero by setting the variable values that set the frequency to zero in the variables section of the ADE.

2. Set an output frequency range of interest.

3. Do not specify too many points.
4. Set the highest harmonic of the PSS for which you want to calculate the mixing products.
5. Set the output node(s).

   This is an averaged measurement of the conversion gain.

Commonly Used Options

Freqaxis: Display positive and negative input frequencies. *Absin* (absolute value) displays the inputs on the positive frequency axis. *Out* is not recommended. With PXF, you specify the output, and PXF calculates the input frequencies and the forward gain from the sources in the

circuit. If you select *out*, all the inputs are displayed on the same frequency axis, which can be very confusing.

PXF calculates the forward transfer functions from the sources in the circuit to the output. When the direct plot is selected, you click the source from which you want to transfer the functions. If you select the input port, you get the transfer functions from the input to the output

that was specified in the *Choosing Analyses* form. If you select the power supply, you can measure the PSRR.

The gain on the upper and lower bands near the first harmonic are equal.

The gain near the second harmonic is quite low.

The gain near the third harmonic is only slightly less than unity.

These are the transfer functions from the RF source to the output with the LO at 1.9GHz.

To measure the conversion gain, move the mouse pointer to the frequency you want and type m.

For an instantaneous measurement at a specific threshold, first determine the threshold by plotting the time-domain waveform from PSS.

The threshold for the example is 2.5 Volts.

On the *Choosing Analyses* form:

1. Set an output frequency range of interest.

   Note: Do not take too many points.

2. Set the highest harmonic of the PSS for which you want to measure the transfer functions.

3. Specify the output node(s) in the schematic.

4. Select *Sampled*.

5. Select either a voltage across an element (probe) or the voltage on a pair of nodes.
6. Specify either the element or the nodes.
7. Specify the voltage threshold for which you want the measurement.
8. Specify the crossing direction. `all`, `rising`, or `falling` edges can be selected.
9. If you need more than 16 points to be calculated, set maximum samples at least as high as you need.
10. If you want a delay after the threshold, specify the time as desired.
11. If you want measurements at more points, type the time value of the points you want. The easiest way to get the times is to place markers on the time-domain results.

Now, on the Direct Plot form:

12. Select *pxf* from the *Analyses* section.
13. Select the type of measurement you want form the *Function* section.
14. Select what you want to display from the *Modifier* section.
15. Select the time of the threshold crossing.
16. Select the source in the schematic.

Note: The gain is slightly different at this point compared to the averaged gain calculated earlier in this section.

- The gain is slightly different at different times because the transconductance of the devices in the circuit varies on an instantaneous basis.
- The time can be identified by using the slider at the right of the window, and selecting one of the legends. The legend and the trace highlight.

Measuring AM to PM Conversion

This example shows how to measure AM to PM conversion for an oscillator.

On the *Choosing Analyses* form:

1. Set the PSS harmonic number for which the output is desired.
2. Set the desired output frequency range.
3. Select Linear or Log sweep.
4. Do not specify too many points.
5. Set the maximum sideband larger than the relative harmonic.

6. Specify the output node(s).

7. Select *Modulated*.
8. Select *SSB/AM/PM*.
9. Click in the *Input Modulated Harmonic* list field.
10. Click *Choose*.

    For this example, it is desired to measure the AM to PM conversion for power supply ripple at low frequency to the output near the first harmonic.

11. Select the desired input frequency range.

In the *Direct Plot Form*:

12. Click in the *Output Modulated Harmonic* field.

13. Click *Choose*.

14. Select the desired output frequency range.

When the frequencies are selected, entries are made in the appropriate field.

15. When the simulation completes, select *Results - Direct Plot - Main Form* in the ADE window.

16. Select *pxf modulated* results.

17. Set the *input* to *AM*.

18. Set the *output* to *PM*.

19. Select *dB20*.

20. Select the power supply in the schematic.

The AM to PM conversion is shown in radians per volt that then has dB applied.

.

Periodic S-parameter Analysis (PSP)

The Periodic S-parameter (PSP) analysis is used to compute scattering and noise parameters for n-port circuits that exhibit frequency translation. Such circuits include mixers, switched-capacitor filters, samplers and other similar circuits.

PSP is a small-signal analysis similar to the conventional SP analysis, except, the circuit is first linearized about a periodically time-varying operating point as opposed to a simple DC operating point. Linearizing about a periodically time-varying operating point allows the computation of S-parameters between circuit ports that convert signals from one frequency band to another.

PSP analysis also calculates noise parameters in frequency-converting circuits. PSP computes noise figure (both single-sideband and double-sideband), input referred noise, equivalent noise parameters, and noise correlation matrices. The noise features of PSP analysis, as for Pnoise analysis but unlike SP analysis, include noise folding effects due to the periodically time-varying nature of the circuit.

Computing the n-port S-parameters and noise parameters of a periodically varying circuit is a two step process.

- First, the small stimulus is ignored and the periodic steady-state response of the circuit to possibly large periodic stimulus is computed using PSS analysis.

  As a normal part of the PSS analysis, the periodically time-varying representation of the circuit is computed and saved for later use.

- Next, using the PSP analysis, small-signal excitations are applied to compute the n-port S-parameters and noise parameters.

  A PSP analysis cannot be used alone and must follow a PSS analysis. However, any number of periodic small-signal analyses, such as PAC, PSP, PXF, and Pnoise can follow a single PSS analysis.

Like other SpectreRF small-signal analyses, the PSP analysis can sweep only frequency.

PSP Synopsis

For a PSP analysis, you need to specify the port and port harmonic relations. Select the ports of interest by setting the *port* parameter. Set the periodic small-signal output frequencies of interest by setting the *portharmsvec* or the *harmsvec* parameters.

For a given set of *n* integer numbers representing the harmonics K_1, K_2, ... K_n, the scattering parameters at each port are computed at the frequencies

$f(scattered) = f(rel) + K_i \times fund(pss)$

where *f(rel)* represents the relative frequency of a signal incident on a port, *f(scattered)* represents the frequency to which the relevant scattering parameter represents the conversion, and *fund(pss)* represents the fundamental frequency used in the corresponding PSS analysis.

Therefore, when analyzing a down-converting mixer with signal in the upper sideband and sweeping the RF input frequency, the most relevant harmonic for RF input is $K_i = 1$ and for IF output $K_i = 0$. Hence we can associate $K_2=0$ with the IF port and $K_1=1$ with the RF port. *S21* represents the transmission of signal from the RF to IF, and *S11* the reflection of signal back to the RF port. If the signal was in the lower sideband, then a choice of $K_1=-1$ would be more appropriate.

You can use either the *portharmsvec* or the *harmsvec* parameters to specify the harmonics of interest. If you give *portharmsvec*, the harmonics must be in one-to-one correspondence with the ports, with each harmonic associated with a single port. If you specify harmonics with the optional *harmsvec* parameter, then all possible frequency-translating scattering parameters associated with the specified harmonics are computed.

For PSP analysis, the frequencies of the input and of the response are usually different (this is an important way in which PSP differs from SP). Because the PSP computation involves inputs and outputs at frequencies that are relative to multiple harmonics, the *freqaxis* and *sweeptype* parameters behave somewhat differently in PSP than they do in PAC and PXF.

The *sweeptype* parameter controls the way the frequencies are swept in PSP analysis. Specifying *relative* sweep, sweeps relative to the analysis harmonics (not the PSS fundamental). Specifying *absolute* sweep, sweeps the absolute input source frequency. For example, with a PSS fundamental of 100MHz, the *portharmsvec* set to [9 1] to examine a down-converting mixer, *sweeptype=relative*, and a sweep *range* of *f(rel)=0->50MHz*, then S21 would represent the strength of signal transmitted from the input port in the range 900-950MHz to the output port at frequencies 100->150MHz.

Using `sweeptype=absolute` and sweeping the frequency from *900->950MHz* would calculate the same quantities, since *f (abs)=900->950MHz*, and *f (rel) = f (abs) - K1 * fund(pss) = 0->50MHz*, because *K1=9* and *fund(pss) = 100MHz*.

The *freqaxis* parameter is used to specify whether the results should be output versus the scattered frequency at the input *port(in)*, the scattered frequency at the output *port(out)* or the absolute value of the frequency swept at the input *port(absin)*.

Note: Unlike in PAC, PXF, and Pnoise analyses, increasing the number of requested ports and harmonics *increases* the simulation time substantially.

If you need PSP results at frequencies that are beyond the 40th harmonic of the PSS fundamental frequency, you need to set `maxacfreq`, which is a PSS option. Set `maxacfreq` to `harmonic number x PSS beat frequency`.

PSP analysis also computes noise figures, equivalent noise sources, and noise parameters. The noise computation, which is skipped only when the *donoise* parameter is set to `no`, requires additional simulation time.

Noise Calculations Performed by PSP

Name	Description	Output Label
N_o	Total output noise at frequency f	
N_s	Noise at the output due to the input probe (the source)	
N_{si}	Noise at the output due to the image harmonic at the source	
N_{so}	Noise at the output due to harmonics other than input at the source	
N_l	Noise at the output due to the output probe (the load)	
IRN	Input referred noise	**In**
G	Gain of the circuit (See **Note:**)	**Gain**
F	Single sideband noise factor	**F**
NF	Single sideband noise figure	**NF**
F_{dsb}	Double sideband noise factor	F_{dsb}
NF_{dsb}	Double sideband noise figure	NF_{dsb}
F_{ieee}	IEEE single sideband noise factor	F_{ieee}
NF_{ieee}	IEEE single sideband noise figure	NF_{ieee}

Note: The gain computed by PSP is the voltage gain from the actual circuit input to the circuit output, not the gain from the internal port voltage source to the output.

PSP analysis performs the following noise calculations.

Input referred noise

$$IRN = \sqrt{\frac{N_o^2}{G^2}}$$

Single sideband noise factor

$$F = \frac{(N_o^2 - N_l^2)}{N_s^2}$$

Single sideband noise figure

$$NF = 10 \times \log 10(F)$$

Double sideband noise factor

$$F_{dsb} = \frac{N_o^2 - N_l^2}{N_s^2 + N_{si}^2}$$

Double sideband noise figure

$$NF_{dsb} = 10 \times \log 10(F_{dsb})$$

IEEE single sideband noise factor

$$F_{ieee} = \frac{N_o^2 - N_l^2 - N_{so}^2}{N_s^2}$$

IEEE single sideband noise figure

$$NF_{ieee} = 10 \times \log 10(F_{ieee})$$

To insure accurate noise calculations, you need to set the *maxsideband* or *sidebands* parameters to include the relevant noise folding effects. The *maxsideband* parameter is only relevant to the noise computation features of PSP.

Parameters for PSP Analysis

Unlike other analyses in SpectreRF, the PSP analysis can only sweep frequency.

Sweep Interval Parameters

`start=0`	Start sweep limit.
`stop`	Stop sweep limit.
`center`	Center of sweep.
`span=0`	Sweep limit span.
`step`	Step size for a linear sweep.
`lin=50`	Number of steps for a linear sweep.
`dec`	Points per decade.
`log=50`	Number of steps for a logarithmic sweep.
`values=[...]`	Array of sweep values.
`sweeptype=relative`	Specifies whether the sweep frequency range is absolute frequency of input or whether it is relative to the port harmonics. When the unspecified value is used, SpectreRF sweeps the absolute input source for non-PSP-driven cases; for other cases, SpectreRF sweeps relative to the port harmonics. Possible values are `absolute`, `relative` or `unspecified`.

Port Parameters

`ports=[...]`	List of active ports. Ports are numbered in the order given. For purposes of noise figure computation, the input is considered port 1 and the output is considered port 2.

`portharmsvec=[...]`	List of harmonics active on the specified list of active ports. The `portharmsvec` vector must have a one-to-one correspondence with the `ports` vector.
`harmsvec=[...]`	List of active harmonics in addition to the harmonics associated with specific ports by the `portharmsvec` vector.

Output Parameter

`freqaxis`	Specifies whether the results should be output versus the input frequency, the output frequency, or the absolute value of the input frequency. Default is `in`. Possible values are `absin`, `in` or `out`.

Noise Parameter

`donoise=yes`	Perform noise analysis. If *oprobe* is specified as a valid port, `donoise` is set to *`yes`*, and a detailed noise output is generated. Possible values are `no` or `yes`.

Probe Parameter

`maxsideband=7`	Maximum sideband included when computing noise either up-converted or down-converted to the output by the periodic drive signal.

Convergence Parameters

`tolerance`	Relative tolerance for the linear solver.
`gear_order=2`	Gear order used for small-signal integration. Possible values are 1 or 2.
`solver=turbo`	Solver type. Possible values are `std` or `turbo`.
`oscsolver=turbo`	Oscillator solver type. Suggest to use ira for huge circuit. Possible values are `std`, `turbo`, `ira`, or `direct`.
`lnsolver=gmres`	Linear solver. Possible values are `bicgstab`, `gmres`, `qmr`, or `resgmres`.

`resgmrescycle=short`	Controls recycling and the length of the computation cycle. Possible values for large signal analysis are `instant`, `short`, and `long`. Possible values for small signal analyses are `instant`, `short`, `long`, `recycleinstant`, `recycleshort`, and `recyclelong`.

Annotation Parameters

`annotate=sweep`	Degree of annotation. Possible values are `no`, `title`, `sweep`, `status`, or `steps`.
`stats=no`	Analysis statistics. Possible values are `no` or `yes`.
`title`	Analysis title.

Setting up a PSP simulation - Input Frequency Range

In PSP, you can specify either an input frequency range, or an output frequency range. This is the input frequency range. On the *Choosing Analyses* form:

1. Specify the input frequency range.
2. Specify log or linear spacing.
3. Do not take too many points for *Sweep Type* and *Step Size*.
4. Type in the number 1 for the port number.
5. Click the *Name* field.
6. Click *Select Port*.
7. Select the input port in the schematic.
8. Click *Choose Harmonic*.

   The Choose Harmonic window appears.

9. Select the input frequency range.
10. Click *OK*.

The *Choosing Analyses* form updates with the harmonic number.

11. Click *Add*. If this step is skipped, the port and frequency definition will not be run.

The line appears in the upper section.

12. Type the number 2 in the *Port Number* field.

13. Delete the entries in the edit fields.

14. Click *Select Port*.

15. Select the output port in the schematic.

16. Click *Choose Harmonic*.

17. Choose the desired output frequency range.

18. Click *OK*.

19. Click *Add*. If this step is skipped, the simulation will not include this port definition.

The port definition appears in the list.

20. Add as many port definitions as required.

Setting up PSP - Output Frequency Range

On the *Choosing Analyses* form:

1. Type the output frequency range.
2. Select Log or linear spacing.
3. Do not take too many points.
4. Type the number 1 in the Port Number field.
5. Place the mouse cursor in the *Name* field.
6. Click *Select Port*.
7. Select the input port in the schematic.
8. Click *Choose Harmonic*.

9. Select the input frequency range in the *Choose Harmonic* window.

10. Click *OK*.

11. Click *Add*. If this step is skipped, this port definition will not be in the simulation.

The port definition appears in the list.

12. Type the number 2 in the Port Number field.

13. Place the cursor in the *Name* field.

14. Click *Select Port*.

15. Choose the output port in the schematic.

16. Click in the *Harm* field.

17. Click *Choose Harmonic*.

18. Select the output frequency in the *Choose Harmonic* window.

19. Click *OK*.

20. Click *Add*. If this step is skipped, the port definition will not be in the simulation.

The port definition appears in the list.

21. Repeat these steps as necessary for additional ports in the circuit.

22. Specify the number of sidebands.

If you need more than 40 sidebands, set the PSS option *maxacfreq* to the max sideband number times the PSS frequency.

Commonly Used Options

All the plots from PSP will be plotted on the same frequency axis. Select the desired frequency range in the *freqaxis* option.

1. When the simulation completes, select *Results - Direct Plot - Main Form* in the ADE window.

2. To plot S-Parameters, select *SP*.
3. Select Z or Y Smith (usually).
4. Select S11or the desired S-parameter measurement.

5. The S11 plot appears.

To position a marker, move the mouse pointer to the desired point on the curve and type m.

Note: This is a zoomed-in view.

Note: When plotting on a Smith chart, always check the frequency range for the desired plot in the Spectre output window. If the frequency is negative, the Smith chart plot will be upside down.

Note that Stability Factor is defined only for amplifiers. This example is a down-converting mixer.

6. Noise Figure is single-sideband noise figure.
7. If you want IEEE single-sideband noise figure, select *Nfieee*.
8. If you want double-sideband noise figure, select *NFdsb*.

9. To plot noise circles, select *NC*.

10. At first, type a wide range for the *Start* and *Stop*. Then refine the entries. There is no autoscale for this plot.

11. The noise circles are plotted.

12. You can identify the noise figure of the individual curve by matching the color in the legend, or by selecting the curve.

13. The curve will highlight, and the 3 legends will highlight.

14. To plot gain circles select GAC.

15. Set reasonable *range* and *step*.

16. The gain circles are plotted.

5

Multiple Input Large and Small-signal Analyses

Quasi-Periodic Steady-State Analysis (QPSS)

The quasi-periodic steady-state (QPSS) analysis computes the quasi-periodic steady-state response of a circuit that operates on multiple time scales. A quasi-periodic signal has dynamics in multiple fundamental frequencies. Closely spaced or incommensurate fundamentals cannot be efficiently resolved by PSS analysis. (Incommensurate frequencies are those for which there is no period that is an integer multiple of the period of each frequency.) QPSS analysis allows you to compute circuit responses to several moderately large input signals in addition to a strongly nonlinear tone which represents the LO or clock signal. A typical example is the intermodulation distortion measurements of a mixer with two closely spaced moderate input signals. QPSS treats one particular input signal (usually the one that causes the most nonlinearity or the largest response) as the large signal, and the others as moderate signals.

When you perform a QPSS analysis

1. An initial transient analysis runs with all moderate input signals suppressed.
2. A number of stabilizing iterations run (always at least 2) with all signals activated.
3. The Shooting Newton method runs.

The QPSS analysis using the Shooting engine employs the Mixed Frequency Time (MFT) algorithm extended to multiple fundamental frequencies. For details about the MFT algorithm, see *Steady-State Methods for Simulating Analog and Microwave Circuits*, by K. S. Kundert, J. K. White, and A. Sangiovanni-Vincentelli, Kluwer, Boston, 1990. Basically, instead of simulating the continuous waveform, enough periods of the large signal are simulated with all the signals applied in order to calculate the mixing products specified by the number of harmonics for all the tones. the larger the number of tones and harmonics on each tone, the larger the number of periods of the large signal need to be simulated, thus the runtime and memory requirements go up.

Except for switched-capacitor filters and sampling circuits or other very nonlinear circuits, Harmonic Balance should be used for QPSS because the runtime and memory requirement are much less.

As is true for PSS analysis, QPSS analysis uses the Shooting Newton method as its backbone. However, unlike PSS analysis (where each Newton iteration performs a single transient integration), for each Newton iteration the QPSS analysis performs a number of transient integrations of one large signal period. Each integration differs by a phase-shift in each moderate input signal. When you set up a QPSS analysis, you determine the number of integrations performed by the number of harmonics of moderate fundamentals you select.

You select the moderate signals to model with the `maxharms` parameter as follows

$$\text{maxharms} = [k_1, k_2, \ldots, k_n]$$

The total number of integrations for the simulation is calculated as

$$(2k_1 + 1) \times (2k_2 + 1) \times \ldots \times (2k_n + 1)$$

One consequence is that the efficiency of the algorithm depends significantly on the number of harmonics required to model the responses of moderate fundamentals. Another consequence is that the number of harmonics of the large fundamental does not significantly affect the efficiency of the Shooting algorithm. The boundary conditions of a Shooting interval are such that the time domain integrations are consistent with a frequency domain transformation with a shift of one large signal period.

Comparing QPSS Analysis with PSS and PAC Analyses

A QPSS analysis is similar to a PSS analysis followed by a PAC analysis in that if you treat one of the input signals as a small signal, a signal whose harmonics do not contribute significantly to the output, you can use the PSS/PAC analyses to model intermodulation distortion effectively.

The QPSS analysis has the following advantages that make it the analysis of choice in many situations. For example, with a QPSS analysis you can measure

- Harmonic distortion and frequency translation effects created by multiple moderate-signal inputs, including all third order products. These measurements are impossible to obtain using PSS/PAC analysis because they do not model the effects of small-signal harmonics.

- The effects of multiple moderate signals. With PSS/PAC analysis, you are restricted to modeling the effects of a single small signal on the fundamental you compute with the PSS analysis. A QPSS analysis lets you model all moderate-signal inputs, including the sums of sinusoids that are not periodic.

To see the difference between the information available with QPSS and PSS/PAC analyses, compare how the two approaches determine the output signals produced by input signals at 900 MHz and 905 MHz. The discussion below assumes you are interested in data only for fundamentals and their first harmonics.

If you perform a PSS analysis for the 900 MHz signal followed by a PAC analysis that applies 905 MHz as a small-signal with `maxsideband = k`$_1$, you get

$$i \times 900M + 905M$$

Where

$$i = \pm 1, \pm 2, .., \pm k_1$$

If you perform a QPSS analysis with input signals at 900 MHz (for the large tone) and 905 MHz (for the moderate tone), you get information about the following additional frequency translation signals created by moderate tones.

The output signals are centered at the following frequencies:

$$i \times 900M + j \times 905M$$

where

$$i = \pm 1, \pm 2, .., \pm k_1$$

and

$$j = \pm 1, \pm 2, .., \pm k_2$$

Again, the number of harmonics of the moderate tone (k_2) effects the simulation time.

Figure 5-1 shows that more information is available from a QPSS analysis than from a PSS analysis followed by a PAC analysis.

Figure 5-1 Comparison of Information From QPSS and PSS/PAC Analyses

QPSS Parameters

Distortion Fundamental Parameters

`funds=[ ... ]`	Array of fundamental frequency names to use in analysis.
`maxharms=[ ... ]`	Array of number of harmonics of each fundamental to consider for each fundamental.
`selectharm`	Name of harmonics selection methods. Possible values are `box`, `diamond`, `funnel` or `axis`. Default is `diamond` when `maximorder` or `boundary` is set; otherwise, default is `box`.
`evenodd=[...]`	Array of `even`/`odd`/`all` strings for moderate tones to select harmonics.
`boundary`	Harmonic selection boundary.
`maximorder`	Maximum intermodulation order (same as `boundary` parameter).

`harmlist=[...]`	Array of harmonics indexes.
`freqdivide`	Large signal frequency division. (Harmonic Balance engine only)

Simulation Interval Parameters

`tstab=0.0 s`	Extra stabilization time for independent sources after the onset of periodicity.
`stabcycles=2`	Stabilization cycles run with both large and moderate sources enabled.
`tstart=0.0 s`	Initial transient analysis start time.

Time-Step Parameters

`maxstep (s)`	Maximum time step. Default derived from `errpreset`. See "The errpreset Parameter in QPSS Analysis" on page 1085.
`maxacfreq`	Maximum frequency requested in a subsequent periodic small-signal analysis. Default derived from `errpreset` and `harms`. This parameter is valid only for Shooting. See "The errpreset Parameter in QPSS Analysis" on page 1085.
`step=0.001 period (s)`	Minimum time step. Used only to maintain the aesthetics of the results. This parameter is valid only for Shooting.

Initial-Condition Parameters

`ic=all`	Sets the initial condition. Possible values are `dc`, `node`, `dev`, or `all`.
`skipdc=no`	If `yes`, there will be no dc analysis before transient analysis. Possible values are `no`, `yes`, and `sigrampup`.
`readic`	File that contains initial condition.

Convergence Parameters

`readns`	File that contains an estimate of the initial transient solution.
`cmin=0 F`	Minimum capacitance from each node to ground.

Output Parameters

`save`	Signals to output. Possible values are `all`, `lvl`, `allpub`, `lvlpub`, `selected` or `none`.
`nestlvl`	Levels of subcircuits to output.
`oppoint=no`	Should operating point information be computed for the initial timestep. If so, where should it be sent? Possible values are `no`, `screen`, `logfile`, or `rawfile`.
`skipstart=starttime s`	The time to start skipping output data.
`skipstop=stoptime s`	The time to stop skipping output data.
`skipcount`	Save only one of every `skipcount` points.
`strobeperiod (s)`	The output strobe interval (in seconds of transient time).
`strobedelay=0 s`	The delay (phase shift) between the `skipstart` time and the first strobe point.
`compression=no`	Do data compression on output. Possible values are `no`, `alllocal`, `pointlocal`, `sigglobal`, `abstol`, or `yes`.
`saveinit=no`	If set, save the waveforms for the initial transient before steady-state. Possible values are `no` or `yes`.

State-File Parameters

`write`	File to which initial transient solution (before steady-state) is to be written.
`writefinal`	File to which final transient solution in steady-state is to be written. This parameter is supported for only the Shooting engine.
`swapfile`	Temporary file that holds steady-state information. Tells SpectreRF to use a regular file rather than virtual memory to hold the periodic operating point. Use this option if the simulator warns about not having enough memory to complete the analysis. This parameter is supported for only the Shooting engine.

Integration Method Parameter

`method`	Integration method. The default is derived from `errpreset`. This parameter is supported for only the Shooting engine. Possible values are `euler`, `trap`, `traponly`, `gear2`, or `gear2only`. See "The errpreset Parameter in QPSS Analysis" on page 1085.

Accuracy Parameters

`errpreset`	Selects a reasonable collection of parameter settings. Possible values are `liberal`, `moderate` or `conservative`. See "The errpreset Parameter in QPSS Analysis" on page 1085.
`relref`	Reference used for the relative convergence criteria. Default derived from `errpreset`. Possible values are `pointlocal`, `alllocal`, `sigglobal`, or `allglobal`.
`lteratio`	Ratio used to compute LTE tolerances from Newton tolerance. Default derived from `errpreset`.
`lteminstep=0.0 s`	Local truncation error will be ignored if the step size is less than `lteminstep`.
`steadyratio`	Ratio used to compute steady state tolerances from LTE tolerance. Default derived from `errpreset`.
`maxperiods`	Maximum number of simulated periods to reach steady-state.
`lnsolver=gmres`	Linear solver. Possible values are `bicgstab`, `gmres`, `qmr`, or `resgmres`.
`resgmrescycle=short`	Controls recycling and the length of the computation cycle. Possible values for large signal analysis are `instant`, `short`, and `long`. Possible values for small signal analyses are `instant`, `short`, `long`, `recycleinstant`, `recycleshort`, and `recyclelong`.
`itres=1e`$^{-4}$ `for shooting, 0.9 for HB`	The `itres` parameter controls the residual for iterative solution of linearized matrix equation at each Newton iteration. Tightening the parameter can help with the Newton convergence, but won't affect the result accuracy. The value should be between [0, 1].
`inexactNewton=no`	Inexact Newton method. Possible values are `no` or `yes`.

`finitediff`	Options for finite difference refinement method after quasi-periodic Shooting method. `finitediff` is changed from `no` to `samegrid` automatically when `readqpss` and `writeqpss` are used to re-use QPSS results. Possible values are `no`, `yes` or `refine`.

Harmonic Balance Parameters

`harmonicbalance=no`	Use the Harmonic Balance simulation engine instead of the time-domain Shooting method engine. Possible values are `no` or `yes`.
`flexbalance=no`	An alternative name for the `harmonicbalance` parameter.
`hbpartition_defs=[...]`	Specify the instances included in each Harmonic Balance partition.

`hbpartition_fundratios=[...]`	Specify the fundamental frequency ratio of each tone in each Harmonic Balance partition. For example, when
	`funds=["LO" "RF1" "RF2"]`
	`hbpartition_fundratios=["2;1;1" "1;1;1"]`
	then the fundamental frequency of the first tone in the first partition is `2*LO` and the fundamental frequencies of the remaining tones in the two partitions are equal to `1*RF1` or `1*RF2`.
	All the values specified for `hbpartition_fundratios` must be integers.

`hbpartition_harms=[...]`	Specify the maximum number of positive harmonics of each tone for every Harmonic Balance partition. For example, `hbpartition_harms=["10;0;0" "5;3;3"]` indicates that the maximum numbers of positive harmonics for the first partition are 10, 0, and 0; and the maximum numbers of positive harmonics for the second partition are 5, 3, and 3.
`oversamplefactor=1`	Oversample device evaluations.
`oversample=[...]`	An array of oversample factors for each tone. This `oversample` vector overrides `oversamplefactor`.
`hbhomotopy=tstab`	Specify the Harmonic Balance homotopy selection method. Possible values are `source`, `tone`, `tstab`, or `gsweep`. The `source` value is not applicable for autonomous circuits.

`sweepic`	IC extroplation method in sweep HB analysis. Possible values are `linear`, `log`, or `none`. Default value is `none`.
`gstart=1.e-7`	The beginning conductance value when `hbhomotopy` is set to `gsweep`.
`gstop=1.e-12`	The ending conductance value when `hbhomotopy` is set to `gsweep`.
`glog=5`	Number of steps for the log sweep when `hbhomotopy` is set to `gsweep`.
`backtracking=no`	Activate the backtracking utility of Newton's Method. Turn this utility on when slow convergence occurs in PSS or QPSS. Possible values are `no` or `yes`.

Annotation Parameters

`stats=no`	Analysis statistics. Possible values are `no` or `yes`.
`annotate=sweep`	Degree of annotation. Possible values are `no`, `title`, `sweep`, `status`, or `steps`.
`annotateic=no`	Degree of annotation for initial condition. Possible values are `no`, `title`, `sweep`, `status`, `steps`, `iters`, `detailed` or `rejects`.
`title`	Analysis title.

Newton Parameters

`maxiters=5`	Maximum number of iterations per time step.
`restart=no`	Restart the DC/PSS/QPSS solution from scratch if set to `yes`. If set to `no`, reuse the previous solution as initial guess. If set to `firstonly`, restart from scratch when it is first point of sweep, it is only supported in HB. The default value is `no` for HB and `yes` for Shooting. Possible values are `no`, `yes` or `firstonly`.

Circuit Age

`circuitage (Years)`	Stress Time. Age of the circuit used to simulate hot-electron degradation of MOSFET and BSIM circuits.

`writeqpss`	File to which final quasi-periodic steady-state solution is to be written. Small signal analyses such as QPAC, QPXF and QPnoise can read in the steady-state solution from this file directly instead of running the QPSS analysis again. The file of Shooting and HB can't mutually reuse.
`readqpss`	File from which final quasi-periodic steady-state solution is to be read. Small signal analyses such as QPAC, QPXF and QPnoise can read in the steady-state solution from this file directly instead of running the QPSS analysis again. The file of Shooting and HB can't mutually reuse.

Tstab Save and Restart Parameters

`ckptperiod`	Checkpoint the analysis periodically using the specified period.
`saveperiod`	Save the transient analysis periodically on the simulation time.
`saveclock (s)`	Save the tran analysis periodically on the wall clock time. Default is 1800s for spectre.
`savetime=[...]`	Save the analysis states into files on the specified time points.
`savefile`	Save the analysis states into the specified file.
`recover`	Specify the file to be restored.
`oscic=default`	Oscillator IC method. It determines how the starting values for the oscillator are calculated. `oscic=lin` gives you an accurate initial value, but it takes some time. `fastic` is very fast, but it is less accurate. `oscic=skip` directly uses the user provided frequency as the initial guess frequency. It is for two tier method only. Possible values are `default`, `lin`, `fastic`, or `skip`.

The QPSS analysis uses the same parameters as the PSS analysis with a couple of parameters added and a few parameters extended. The most important parameters for QPSS analysis are the `funds` and `maxharms` parameters, which replace and extend the `fund` (or `period`) and `harms` parameters that are used in PSS analysis.

The `funds` parameter accepts a list of names of fundamentals that are present in the sources. (These fundamental names are specified by the source parameter `fundname`.) The simulator figures out the frequencies associated with the fundamental names. An important feature of the funds parameter is that each input signal can be composed of more than one source, provided that the sources all have the same fundamental name. In ADE, this is accomplished by setting the frequency name 1 parameter for all the sources at the same or integer multiple frequencies to the same name. The fundamental frequency for each

fundamental name is the greatest common factor of all the frequencies associated with the name. Simulation terminates if you do not list all the fundamental names on the `funds` parameter.

If you do not specify `maxharms`, a warning message displays, and the number of harmonics defaults to 1 for each fundamental.

The first fundamental is considered the large signal. You can use a few heuristics to pick the large fundamental. This can be selected in ADE. Large and moderate are terminology used by the simulator. The simulation result is a full large-signal solution that results with all the signals applied to the circuit.

- Pick the fundamental that is not sinusoidal.
- Pick the fundamental that causes the most nonlinearity.
- Pick the fundamental that causes the largest response.

An initial transient analysis can be run in order to help improve the convergence of the QPSS analysis. This can be enabled by setting a stop time for the initial transient in the `Additional Time For Transient-Aided HB` (`tstab`) parameter. Only the tone that is defined by selecting *Large* for the associated source can be run in the initial transient. Thus, the largest amplitude tone should be set to the large tone. Spectre does not automatically choose the largest amplitude tone to use for the initial transient analysis.

The `maxharms` parameter accepts a list of numbers of harmonics that are required to sufficiently model responses due to different fundamentals.

The role of some PSS parameters is extended for QPSS analysis.

The `maxperiods` parameter that controls the maximum number of Shooting iterations for PSS analysis also controls the maximum number of Shooting iterations for QPSS analysis. Its default value is 50. The `tstab` parameter controls both the length of the initial transient integration, with only the large tone activated, and the number of stabilizing iterations, with the moderate tones activated. The stable iterations are run before Newton iterations begin.

The errpreset Parameter in QPSS Analysis

The `errpreset` parameter quickly adjusts several simulator accuracy parameters to fit your needs.

Important

In most cases, `errpreset` should be the only accuracy parameter you need to adjust.

- For a fast simulation with reasonable accuracy, set `errpreset` to *`liberal`*.
- For greater accuracy, set `errpreset` to *`moderate`*.
- If accuracy is your primary concern, set `errpreset` to *`conservative`*.

If you do not specify a value for `steadyratio`, it is always 1.0, and it is not affected by `errpreset`.

Table 5-1 shows the effect of `errpreset` settings (`liberal`, `moderate`, and `conservative`) on the default values of a set of accuracy parameters.

Table 5-1 Parameter Default Values for `errpreset` Settings

errpreset	reltol	relref	method	lteratio	maxstep
`liberal`	$1e^{-3}$	sigglobal	gear2only	3.5	clock period/80
`moderate`	$1e^{-4}$	sigglobal	gear2only	3.5	clock period/100
`conservative`[1]	$1e^{-5}$	sigglobal	gear2only	10	clock period/200

1. For `errpreset=conservative`, `lteratio=10.0` by default. `lteratio=3.5` *only* when you set `reltol<=1e`$^{-4}$`*` (`10.0/3.5`).

These `errpreset` settings include a default `reltol` value which is an enforced upper limit for `reltol`. The only way to *decrease* the `reltol` value is in the `options` statement. The only way to *increase* the `reltol` value is to relax the `errpreset` setting. SpectreRF sets the `maxstep` parameter value so that it is no larger than the value given in Table 5-1.

With the exception of the `reltol` and `maxstep` values, the `errpreset` setting does not change any parameter values you have explicitly set. The actual values used for the QPSS analysis are given in the log file. If `errpreset` is not specified in the netlist, `liberal` settings are used.

The Compression Parameter in QPSS Analysis

The default value for the `compression` parameter is `no`. Note that compression only applies to the waveform in the `tstab` interval - the startup waveform. The output file stores data for every signal at every time point for which SpectreRF calculates a solution. SpectreRF saves the x-axis data only once, because every signal has the same x value. If `compression=yes`, SpectreRF writes data to the output file only if the signal value changes by at least twice the convergence criteria. In order to save data for each signal independently, x-axis information corresponding to each signal must be saved. If the signals stay at constant values for large periods of simulation time, setting `compression=yes` results in a smaller output data file. If

the signals in your circuit move around a lot, setting `compression=yes` results in a larger output data file.

Input source settings for QPSS

Different wave shapes can be generated by the port component. This is set to generate sinusoids. When sine is selected, one or two frequencies can be generated from the same source.

1. Specify a name for frequency 1.

2. Set the frequency to a variable name.
3. Set the input power to a variable name.

4. Click *Display Second Sinusoid*.

Note: This is a display function only. If there are entries for the second sinusoid in the fields and Display Second Sinusoid is not checked, the hidden values will still be netlisted.

5. Specify a name for the second input from this source.

6. Set the frequency to a different name than the one used for the first frequency.

7. Specify a variable name for the second frequency input power. In this case, the same variable is used as would be the case for an IP3 measurement.

   PAC amplitude will be used for PAC and QPAC, not for the QPSS analysis.

The variables are added in the ADE, where the values are assigned.

- The RF amplitude is -30 dBm.
- The LO amplitude is 200mV Peak.
- The LO frequency is 1.9GHz.
- The RF frequencies are 1.901GHz and 1.9012GHz.

8. Select the engine. QPSS is faster and requires less memory when Harmonic Balance is selected. Only use Shooting when you have very nonlinear circuits like switched capacitor filters or sample and holds.

The sources in the circuit are read into the form and appear in the list.

9. To change a value, select the source line.
10. Then edit it in this line.
11. When you are done, click *Change*.

Important

Set tstab to yes for the signal that causes the largest amount of distortion.

In many cases, the simulation can run faster if harmonics are trimmed from the simulation. Set Harmonics to Select.

12. Select *Diamond* or *Funnel* cut. Diamond is useful for 2-input IP3 simulations for an amplifier. Funnel cut is useful for circuits with a large amplitude signal like an LO, and then 1 or more smaller amplitude signals.

13. Set the maximum order for the cut. Set it at least 1 order higher than the order you are measuring.

14. Select *moderate* for most simulations. Select *conservative* for simulations where you want more accuracy.

15. If you want transient-assisted Harmonic Balance, specify a *tstab* value. The signal with tstab enabled will be applied in the transient for the time specified. A Fourier transform is made of the last cycle, and this is the starting point for the frequency domain iterations. Many simulations can run faster if tstab is set to about 5 periods of the large amplitude signal.

16. If you want to see the startup waveform from the tstab interval, select *yes* tor *saveinit*.

17. If you have convergence difficulties when using Harmonic Balance, you might want to set a *hbhomotopy* method. Try tstab (or set a tstab manually) if you have a frequency divider or a very large amplitude LO. Try source if you have very large amplitude LO signals or if you have a narrow bandwidth filter. Try tone is your circuit has one very large amplitude signal. Try gsweep if you have a circuit with very high impedance nodes.

18. If you want a sweep, select *Sweep*. Up to three variables can be swept in the QPSS Choosing Analyses form. Any number of variables can be swept in the parametric plot tool. This will be single analyses executed one after the other on one core. If you want to have parallel runs, sweep the value with a global variable in ADE-XL or GXL, and specify the number of parallel runs.

19. Enter the variable you want to sweep.

20. Specify an appropriate sweep range.

21. For the Harmonic Balance engine, select no for *restart*. For *Shooting*, select yes.

Commonly Used QPSS Options

Accuracy options

➡ Set *maxperiods* larger than 100 to allow more iterations in Harmonic balance. For a Shooting, the default is 20.

Reuse options

1. *Writeqpss* is a filename that saves the contents of the QPSS to a file. This can be a very large file.
2. *Readqpss* reads the file from above as the starting point for this analysis. This can save time as long as the circuit has not changed drastically from the previous run.
3. *Saveperiod* saves the internal state of the simulator in the tstab interval at spacingsof the time specified in saveperiod. Later times overwrite the file. Do not specify both *saveperiod* and *savetime*.
4. *Savetime* can be a list of times. The internal state of the simulator in the tstab interval at the times specified is written out to a file. Do not specify both *saveperiod* and *savetime*.
5. *Savefile* is the filename to use for the save information.
6. *Recover* is the filename to read in the tstab interval. The simulation picks up from this time as if the simulation were continuous.

Direct Plot

Plotting a spectral output

1. Select *qpss.*
2. Select *voltage*.
3. Select the type of signal to be plotted, such as net or differential nets.

4. Select *spectrum*.
5. Select *peak* or *rms*.
6. Select an appropriate *Modifier*.
7. If you have swept results, a table will appear with all the sweep values. If you want one value to be plotted, select one value from the list. If you want multiple values, hold the control key down and click, or select a starting value and then shift-click the ending value.

Below is the output in strip mode with markers placed for measurements. To place a marker, move the mouse cursor near the measurement you want, and type m.

Plotting the power in dBm

1. Select Power. Currents do not need to be saved for this measurement.

2. Select *Net* (Specify R)
3. Type in a resistance if it is not 50 ohms.
4. Select *Spectrum*.
5. Select *dBm*.

6. If you have a sweep, select the input power levels you want to plot.
7. Select the net in the schematic.

   The power in dBm for the selected net is plotted.

Markers have been positioned by moving the mouse cursor near the measurement you want to make, and then type m.

Plotting voltage gain

1. Select *Voltage Gain*.

2. Select *Spectrum*.

3. Select the appropriate *modifier*.

4. If you have a sweep, select one input power level.

5. Select one input frequency.

6. Select the output net in the schematic.

7. Now select the input net.

The output spectrum appears. All the spectral lines are scaled by the selected input harmonic on the input net.

- The voltage gain is about 14.12 dB.

Plotting large-signal IP2 or IP3

1. Select *IPN Curves*.

2. Select *Net (Specify R)*.

3. Select *Variable Sweep*.
4. Type in a reasonable extrapolation point. This is cover in the steps on page 1104.
5. Select *Input* or *Output Referred IP3*.
6. Select the order.

7. Select the intermodulation frequency.

8. Select the linear frequency.

9. Select the net on the schematic.

The IP3 curve appears.

- The IP3 calculation looks like it is measuring the level at the right side for the third order intercept point. In fact it measures the real intercept point.
- The output node is displayed.
- The third order frequency is displayed.
- The linear output frequency is displayed.
- Select an extrapolation point where the third order data follows the ideal curve for a range of values on either side of the extrapolation point. At high power, the third order curve is unpredictable. At low power (especially with Shooting) there can be a numerical noise floor. The data tends to level out.

If you have switched-capacitor filter or a sample and hold

1. Select the *Shooting engine*.

2. If you sweep with Shooting, select *yes* for *restart*.

3. Disable one input source by setting *its frequency* to zero by setting the variable name in the ADE variables section to zero.

With the LO and one RF signal applied, the compression can be measured.

The Choosing Analysis form redraws and shows only one input.

4. When the simulation completes, select *Results - Direct Plot - Main Form* in the ADE window.

Plotting compression point

1. Select *Compression Point*.

2. Select which type of signal you want to select in the schematic.

3. Enter the compression level in dB.
4. Select *Input* or *Output-referred Compression Point*.
5. Select the output frequency.
6. Select the net on the schematic.

The compression curve appears.

- The compression calculation is displayed.
- The node name is display.
- The output frequency is displayed.

Viewing compression versus output power

1. Select *Comp vs. Pout*.

2. Select *Net (Specify R).*

3. Select the output frequency.
4. Select the net in the schematic.

The plot appears.

Make sure that the input power at the start of the sweep is low enough. The assumption is that the compression at the lowest input power is zero.

5. The next measurement (power gain versus power out) requires that you save the currents for the output and input ports. The way to save all the currents is to select *Outputs - Save All* in the ADE. Harmonic Balance will only save currents in the sources and ports in the circuit. To save a branch current when using HB, add an `iprobe` in the circuit.

The following form appears.

6. Select *all*.
7. Select *OK*.

This measurement requires that the currents be saved. See the previous steps.

8. To measure the power gain versus the output power, select *Power Gain Vs Pout*.

9. Select *Magnitude* or *dB10*.

10. Select the output frequency.

11. Select the input frequency.

12. Select the terminal of the output load in the schematic.

13. Select the terminal of the input source in the schematic.

The power gain curve displays.

QPSS Small Signal Analyses

Quasi-Periodic Noise Analysis (QPnoise)

The Quasi-Periodic Noise (QPnoise) analysis is a quasi-periodic small-signal analysis similar to the conventional noise analysis, except that with QPnoise the circuit is first linearized about a quasi-periodically time-varying operating point as opposed to a simple DC operating point. Linearizing about a quasi-periodically time-varying operating point includes frequency conversion and intermodulation effects. Simply linearizing about a DC operating point cannot include frequency translation because linear time-invariant circuits do not exhibit frequency translation. QPnoise also includes the effect of a quasi-periodically time-varying bias point on the noise generated by the various components in the circuit. Hence QPnoise is useful for predicting the noise behavior of mixers, switched-capacitor filters, and other periodically or quasi-periodically driven circuits. You cannot use a QPnoise analysis alone. It must follow a QPSS analysis. However, any number of quasi-periodic small signal analyses (QPAC, QPSP, QPXF, and QPnoise) can follow a single QPSS analysis.

Computing the small-signal response of a quasi-periodically varying circuit is a two step process.

1. First, the small stimulus is ignored and the quasi-periodic steady-state response of the circuit to possibly large periodic stimuli is computed with a QPSS analysis. As a normal part of the QPSS analysis, the quasi-periodically time-varying representation of the circuit is computed and saved for later use.
2. Then, the small stimuli representing both individual noise sources in the circuit as well as the input noise are applied to the periodically-varying linear representation to compute the small signal response. This is done using the QPnoise analysis.

QPnoise Output

The time-average of the noise at the output of the circuit is computed in the form of a spectral density versus frequency. The output of the circuit is specified with either a pair of nodes or a probe component. To specify the output of a circuit with a probe, specify it using the `oprobe` parameter. If the output is voltage (or potential), choose a *resistor* or a *port* as the output probe. If the output is current (or flow), choose a `vsource` or `iprobe` as the output probe.

If you want the input-referred noise, specify the input source using the `iprobe` parameter. Currently, only a `vsource`, an `isource`, or a `port` can be used as an input probe. If the input source is noisy, as is a `port`, the noise analysis will compute the noise factor (F) and noise figure (NF). To match the IEEE definition of noise figure, the input probe must be a `port` with no excess noise and you must set its `noisetemp` parameter to 16.85 C (290 K).

In addition, the output load must be a `resistor` or `port` and must be identified as the `oprobe`.

If `port` is specified as the input probe, then both input-referred noise and gain are referred back to the equivalent voltage source inside the port. S-parameter analysis calculates those values in traditional sense.

Use the `refsideband` parameter to specify which conversion gain to use when computing input-referred noise, noise factor, and noise figure. The reference sideband satisfies:

$$|f(input)| = |f(out) + \text{refsideband frequency shift}|$$

The reference sideband option (`refsidebandoption`) specifies whether to consider the input at the frequency or the input at the individual quasi-periodic sideband specified. Note that different sidebands can lead to the same frequency.

Sidebands are vectors in QPnoise analysis. Assume you have one large tone and one moderate tone in a QPSS analysis. A sideband K_i will be a vector $[K_{i_}1\ K_{i_}2]$. It gives the frequency at

$K_{i_}1$ * *fund(large tone of QPSS)* + $K_{i_}2$ * *fund(moderate tone of QPSS)*

Use `refsideband=[0 0...]` when the input and output of the circuit are at the same frequency (such as with amplifiers and filters). When the `refsideband` parameter value differs from the 0 vector, QPnoise computes the single side-band noise figure.

This is much easier in ADE because select from list is available as a choice for the reference sideband. When this is chosen, you just select the desired input frequency for the noise figure calculation from the list.

The noise analysis always computes

- Total noise at the output. This includes contributions from the input source and the output load.
- The amount of output noise that is attributable to each noise source in the circuit. These are computed and output individually.

If you identify the input source with `iprobe` and it is a `vsource` or an `isource`, the input-referred noise is computed. This includes the noise from the input source itself.

If you identify the input source with `iprobe` and it is noisy, as is the case with ports, the noise factor and noise figure are computed.

Noise Calculations Performed by QPnoise

Name	Description	Output Label
N_o	Total output noise	**Out**
N_s	Noise at the output due to the input probe (the source)	
N_{si}	Noise at the output due to the image harmonic at the source	
N_{so}	Noise at the output due to harmonics other than input at the source	
N_l	Noise at the output due to the output probe (the load)	
IRN	Input referred noise (See **Note:**)	**In**
G	Gain of the circuit (See **Note:**)	**Gain**
F	Single sideband noise factor	**F**
NF	Single sideband noise figure	**NF**
F_{dsb}	Double sideband noise factor	F_{dsb}
NF_{dsb}	Double sideband noise figure	NF_{dsb}
F_{ieee}	IEEE single sideband noise factor	F_{ieee}
NF_{ieee}	IEEE single sideband noise figure	NF_{ieee}

Note: For the QPnoise analysis, the computation of *gain* and *IRN* assumes that the circuit under test is impedance-matched to the input source. This can introduce inaccuracy into the *gain* and *IRN* computation.

SpectreRF performs the following noise calculations.

Input referred noise

$$IRN = \sqrt{\frac{N_o^2}{G^2}}$$

Single sideband noise factor

$$F = \frac{(N_o^2 - N_l^2)}{N_s^2}$$

Single sideband noise figure

$$NF = 10 \times \log 10(F)$$

Double sideband noise factor

$$F_{dsb} = \frac{N_o^2 - N_l^2}{N_s^2 + N_{si}^2}$$

Double sideband noise figure

$$NF_{dsb} = 10 \times \log 10(F_{dsb})$$

IEEE single sideband noise factor

$$F_{ieee} = \frac{N_o^2 - N_l^2 - N_{so}}{N_s^2}$$

IEEE single sideband noise figure

$$NF_{ieee} = 10 \times \log 10(F_{ieee})$$

QPnoise Synopsis

At the UNIX command line use the optional terminals (`p` and `n`) to specify the output of the circuit. If you do not give the terminals, then you must specify the output with a `probe` component.

In practice, noise can mix with each of the harmonics and intermodulation products of the periodic drive signals applied in the QPSS analysis and end up at the output frequency. However, the QPnoise analysis includes only the noise that mixes with a finite set of harmonics. The noise contributions that are folded in are controlled by several parameters: `clockmaxharm` and `sidevec` parameters. Generally speaking, the `clockmaxharm` is used. This is the maximum clock order field in the ADE Choosing Analyses form.

The `clockmaxharm` parameter affects only the first tone, which is typically the LO or a clock frequency. If Harmonic Balance is set for the engine in QPSS, this is the tone that has `tstab` enabled. It limits the maximum harmonic order of the large tone that will be considered. The order of the harmonics to be considered for the moderate tones is limited by the corresponding entries in the `maxharms` list in QPSS. In the case of the two tones in QPSS, the harmonics of the first tone are limited by `clockmaxharm`. For the second tone, the maximum harmonic order is `maxharms`[2].

If you select sidebands using the `sidevec` parameter, then these selected sidebands are the only sidebands included in the calculation. Take care when specifying the QPnoise `sidevec` and `clockmaxharm` parameters and the QPSS `maxharms` parameter. Your noise results will be in error if you fail to include a sideband that contributes significant noise to the output.

The number of requested sidebands will not substantially change the simulation time.

In quasi-periodic analyses sidebands are vectors, or, in other words, harmonic combinations. One way to specify them is using the `sidevec` parameter. When a QPSS analysis has one large tone and one moderate tone, the sideband is represented by a vector K^1 as $[K^1_1\ K^1_2]$. The corresponding frequency translation is

$$K^1_1 \times fund(qpss - large - tone) + K^1_2 \times fund(qpss - moderate - tone)$$

When there are `L` tones total in the QPSS analysis (`1` large tone and `L-1` moderate tones), there is also a given set of `n` integer vectors representing the sidebands

$K^1 = \{ K^1_1, \ldots, K^1_j, \ldots, K^1_L\}$
$K^2 = \{ K^2_1, \ldots, K^2_j, \ldots, K^2_L\}$
…
$K^n = \{ K^n_1, \ldots, K^n_j, \ldots, K^n_L\}$

The QPnoise analysis computes the output frequency corresponding to each sideband as follows.

$$f^{i}(out) = f(in) + \sum_{i=1}^{L} \left(K_{j}^{i} \cdot f_{j} \right)$$

where

- $f(in)$ represents the (possibly swept) input frequency
- f_j represents the `fundamental frequency` used in the corresponding QPSS analysis.

Enter the `sidevec` parameter as a sequence of integer numbers separated by spaces. For example, you would enter the set of vectors `{1 1 0} {1 -1 0} {1 1 1}` as follows

```
sidevec=[ 1 1 0 1 -1 0 1 1 1]
```

The other way to specify the sidebands is the `clockmaxharm` parameter and the `maxharms` parameter in the preceding QPSS analysis. Only the large tone, the first fundamental in QPSS, is affected by the QPnoise `clockmaxharm` parameter value. It limits the maximum harmonic order of the large tone that will be considered. All the remaining tones, the moderate tones, are limited by the QPSS `maxharms` parameter value.

Given the following parameters

- In the QPSS analysis input, `maxharms=[`k^0_{max} k^2_{max}`...` k^n_{max}`]`
- In the QPnoise analysis input, `clockmaxharm=`K_{max}

The QPnoise analysis output generates the following number of sidebands

$(2*K_{max} + 1)*(2*k^2_{max}+1)*(2*k^3_{max}+1)*...*(2*k^n_{max}+1)$

Swept QPnoise Analysis

Specify sweep limits by providing either the end points (`start` and `stop`) or by providing the center value and the span of the sweep (`center` and `span`).

Specify sweep steps as linear or logarithmic as well as the number of steps or the size of each step. You can give a step-size parameter (`step`, `lin`, `log`, `dec`) to determine whether the sweep is linear or logarithmic. If you do not give a step-size parameter, the sweep is linear when the ratio of `stop` to `start` values is less than 10:1, and logarithmic when this ratio is equal to or greater than 10:1.

Alternatively, you can specify the particular values for the sweep parameter using the `values` parameter. If you give both a specific set of values and a set specified using a sweep range, the two sets are merged and collated before simulation. All frequencies are in Hertz.

Noise Separation

The noise from the individual sidebands can be calculated and are direct plottable in ADE besetting the `separatenoise` parameter to `yes`. This allows the noise from the individual `nmixing` frequencies to be plotted, thus giving a more complete picture of where the noise is coming from.

QPnoise Parameters

Sweep Interval Parameters

`start=0`	Start sweep limit.
`stop`	Stop sweep limit.
`center`	Center of sweep.
`span=0`	Sweep limit span.
`step`	Step size for a linear sweep.
`lin=50`	Number of steps for a linear sweep.
`dec`	Points per decade.
`log=50`	Number of steps for a logarithmic sweep.
`values=[...]`	Array of sweep values.
`sweeptype`	Specifies whether the sweep frequency range is absolute (actual) frequency of input or if it is relative to the `relharmvec` sideband frequency. Possible values are `absolute` or `relative`.
`relharmvec=[...]`	Sideband vector of QPSS harmonics. Relative frequency sweeps are referenced to the `relharmvec`.

Probe Parameters

`oprobe`	Compute total noise at the output defined by this component.

`iprobe`	Refer the output noise to this component.
`refsideband=[...]`	Use the conversion gain associated with this sideband when computing input-referred noise or noise figure.
`refsidebandoption=individual`	Whether to view the sideband as a specified frequency or as an individual sideband. Possible values are `freq` or `individual`.

Output Parameters

`clockmaxharm=7`	In Shooting pnoise, the parameter determines the maximum sideband included when computing noise either up-converted or down-converted to the output by the periodic drive signal. The default value for the Shooting pnoise is 7. In HB pnoise, this parameter determines the size of the small signal system when the HB pnoise is performed. This parameter is critical for the accuracy of the HB pnoise analysis: using small maxsideband may cause accuracy loss. The default value for the HB pnoise is the `harms`/`maxharms` setting in the HB large signal analysis.
`sidevec=[...]`	Array of relevant sidebands for the analysis.
`save`	Signals to output. Possible values are `all`, `lvl`, `allpub`, `lvlpub`, `selected`, or `none`.
`nestlvl`	Levels of subcircuits to output.
`saveallsidebands=no`	Save noise contributors by sideband. Possible values are `no` or `yes`.
`separatenoise=no`	Separate Noise into sources and transfer functions. Possible values are `no` or `yes`.

Convergence Parameters

`tolerance`	Relative tolerance for the linear solver. For the Shooting solver, the default value is $1.0e^{-9}$. For the Harmonic Balance solver, the default value is $1.0e^{-6}$.
`gear_order=2`	Gear order used for small-signal integration.

`solver=turbo`	Solver type. Possible values are `std` or `turbo`.
`lnsolver=gmres`	Linear solver. Possible values are `bicgstab`, `gmres`, `qmr`, or `resgmres`.
`resgmrescycle=short`	Controls recycling and the length of the computation cycle. Possible values for large signal analysis are `instant`, `short`, and `long`. Possible values for small signal analyses are `instant`, `short`, `long`, `recycleinstant`, `recycleshort`, and `recyclelong`.

Annotation Parameters

`annotate=sweep`	Degree of annotation. Possible values are `no`, `title`, `sweep`, `status`, or `steps`.
`stats=no`	Analysis statistics. Possible values are no or yes.
`title`	Analysis title.

Generally, QPnoise is used when there is an LO and a blocker in the circuit.

- The default blocker power is set in the ADE.
- The LO Frequency is set with a variable name.
- The blocker frequency is set with a variable name.
- The second RF signal is disabled by setting its frequency to zero.

Setting up the QPnoise analysis

1. First, set up the qpss analysis. For more information, see page 1090 of this document.

2. Now on the Choosing Analyses form, select *qpnoise*.

3. Type in an output frequency range for your application.

4. Do not specify too many points.

5. For HB, you can leave *maximum sidebands* blank. For Shooting, specify the same number of sidebands as you specify for the large tone.

6. Use a resistor or a port as a load, then select it. This will exclude the noise from the load for the noise figure calculations, but retain it for the total noise.

7. If you want a noise figure calculation, you must use a port as the input. Specify the instance name in the *Input Port Source* field.

8. Set *Reference Sideband* to select from list.

9. Type in a reasonable range of input frequencies.

10. Select the input frequency you want for input-referred noise and noise figure.

11. If you want the individual components of noise to be available for plotting, select *yes*.

12. When the simulation completes, select *Results - Print - Noise Summary*.

13. Select the noise unit from the *noise unit* drop-down list box.

14. Select a spot noise at a single frequency or integrated noise over a range of frequencies measurement.

15. Select *Include All Types.*

16. Type in how many results you want to see in the *truncate* field.

17. Select the sort type by clicking the desired checkbox from *sort by* field.
18. Select *Apply* or *OK*.

    The noise summary appears.

Device	Param	Noise Contribution	% Of Total
/Q1	re	3.23152e-09	35.87
/Q5	re	3.23152e-09	35.87
/Q1	rc	1.14252e-09	4.48
/Q5	rc	1.14252e-09	4.48
/Q4	re	1.06831e-09	3.92
/Q8	re	1.06831e-09	3.92
/Q2	re	5.46227e-10	1.02
/Q3	re	5.46227e-10	1.02
/Q6	re	5.46227e-10	1.02
/Q7	re	5.46227e-10	1.02
/Q4	rc	4.17188e-10	0.60
/Q8	rc	4.17188e-10	0.60
/Q1	rb	3.86241e-10	0.51
/Q5	rb	3.86241e-10	0.51
/Q8	ic	3.83978e-10	0.51
/Q4	ic	3.83978e-10	0.51
/Q5	ic	3.78941e-10	0.49
/Q1	ic	3.78939e-10	0.49
/Q4	rb	2.83069e-10	0.28
/Q8	rb	2.83068e-10	0.28

Spot Noise Summary (in A/sqrt(Hz)) at 1K Hz Sorted By Noise Contributors
Total Summarized Noise = 5.39579e-09
No input referred noise available
The above noise summary info is for qpnoise_src data

16

- Percent of total is the percentage of noise power this component adds.
- Noise contribution is the amplitude in the noise unit you select.
- Param is the specific noise mechanism in the device.
- Device is the instance name in the schematic.

Getting the direct plot form

1. Select *Results - Direct Plot - Main Form* in the ADE window.

2. Select *qpnoise*.

3. Select the measurement you want to make.

4. If you want a noise figure calculation over a specified bandwidth, select *Integrated Over Bandwidth*.

5. Type in the frequency range for the noise figure calculation.

The noise figure curve (normalized to a 1 Hz noise bandwidth) is plotted.

If you specified a frequency range, the noise figure will be displayed.

Viewing individual noise contributors

1. Select *qpnoise separation*.

2. Select *sideband output*. This gives the noise at the output from each mixing product separately.

3. Select Volts per root hertz or volts-squared per hertz.

4. Select *magnitude* or *dB20*.

5. Select the *sideband number*. There are as many numbers as there are inputs to the circuit. In this example, there is the LO and the blocker signal, so there are two numbers.

The labels at the top are the frequency name properties that are associated with that input. Here the first column is the LO and the second column is the blocker.

The first number represents the frequency shift (the harmonic the signal is mixing with) in multiples of the LO frequency. The second number is the frequency shift from,mixing with the blocker.

0 0 means that the output frequency specified in the Choosing Analyses form is shifted by `zero` x `Lo frequency` plus `zero` x `blocker frequency`. This is the noise that is inherent with no frequency translation at all from mixing with either input frequency.

6. Select *Plot*.

The noise from the 0 0 sideband is plotted. The curve is normalized to a 1 Hertz bandwidth.

Plotting a different sideband

-1 0 means that the noise frequency being plotted is shifted down by the LO frequency. This is the noise that is below the first harmonic frequency that mixes down to the output frequency.

➡ Select *Plot*.

- The first noise curve that is plotted has the zero suffix.
- The second curve plotted has the one suffix.
- The noise with no frequency translation is larger than the noise translated down from the low side of the first harmonic.

Plotting noise from all internal noise sources from each instance with a given frequency translation

1. Select *Instance Output*.

2. Select volts per root Hertz or volts-squared per hertz.

3. Select the *magnitude* or *dB*.

4. Select the *sideband number*. The first number is the harmonic of the LO tone that is being mixed with. The second number is the harmonic number of the blocker that is being mixed with. If the numbers are negative, it means the noise below the harmonic. If the number is positive, it means the noise above the harmonic. 0 0 means the output with no mixing of either harmonic.

-1 0 means the noise below the LO harmonic that mixes down to the output.

5. Select either *Include All Types*, or select the type of device you want.

6. If you want to exclude an instance, enter its instance name in this field. A list is allowed.
7. If you want to exclude everything except what is in the list, enter the instance names of the components you want to see in this field.
8. Enter how may curves you want to see.

9. Select *Plot*.

The 4 curves with the most noise are plotted. The noise bandwidth is one Hertz.

Viewing transfer functions for each noise source at a given frequency translation

1. Select *Source Output*.

2. Select volts per root Hertz, or volts-squared per hertz.

3. Select *magnitude* or *dB20*.

4. Select the *sideband number*.

5. Select *Include all types*, or select a class of devices.

6. If you want to exclude an instance, enter its instance name in this field. A list is allowed.

7. If you want to exclude everything except what is in the list, enter the instance names of the components you want to see in this field.

8. Enter how may curves you want to see.

9. Select *Plot*.

To see which trace is which, select the legend. The legend and the trace will be highlighted.

Desensitization measurement (noise figure versus blocker power)

This section has a swept measurement. For details on how to set up the swept QPSS, see page 6.

1. When you sweep in QPSS, the ADE default is to select a single frequency.

2. Type in the desired output frequency.

3. Put the number of harmonics you specified for the large tone (Shooting) or the tone with tstab enabled (Harmonic Balance). If you use HB, you can leave this blank.

4. Use a resistor or a point for a load. Specify the instaqncename of thatloadhere in order to exclude the noise of the load from the noise figure calculation.

5. The rest of the noise calculations will have that noise present.

6. If you want a noise figure measurement, you must use a port as the input source. Specify its instance name as desired.

7. Set *Reference Sideband* to *Select from list*.

8. Select the desired input frequency from the list.

9. If you want to be able to plot the individual noise contributors, set *Noise Separation* to *yes*.

10. When the simulation completes, select *Results - Direct Plot - Main Form* from the ADE window.

11. Select *qpnoise*.

12. Select the desired measurement.

13. Select *plot*.

Depending on your release, you may see results displayed like this.

To get a more useful display, double-click one of the numbers on the X Axis. The Axis Attributes window appears.

14. Set the *Sweep Variable* to the variable name that was swept.

15. Select *OK*.

The curve redraws to display the noise figure as a function of the blocker power.

Quasi-Periodic AC Analysis (QPAC)

The quasi-periodic AC (QPAC) analysis computes transfer functions for circuits that exhibit multi-tone frequency translation. Such circuits include mixers, switched-capacitor filters, samplers, phase-locked loops, and similar circuits.

QPAC is a quasi-periodic small-signal analysis like the conventional AC analysis, except that with QPAC the circuit is first linearized about a quasi-periodically time-varying operating point as opposed to a simple DC operating point. Linearizing about a quasi-periodically time-varying operating point produces transfer-functions that include frequency translation. Simply linearizing about a DC operating point cannot include frequency translation because linear time-invariant circuits do not exhibit frequency translation.

Computing the small-signal response of a quasi-periodically varying circuit is a two step process.

1. First, the small stimulus is ignored and the quasi-periodic steady-state response of the circuit to possibly large periodic stimuli is computed with a QPSS analysis. As a normal part of the QPSS analysis, the quasi-periodically time-varying representation of the circuit is computed and saved for later use.
2. Second, the small stimuli are applied to the periodically-varying linear representation to compute the small signal response. This is done using the QPAC analysis.

QPAC Output Frequency and Sideband Vectors

You select the set of quasi-periodic small-signal output frequencies you are interested in by setting either the `clockmaxharm` or the `sidevec` output parameters.

In quasi-periodic analyses sidebands are vectors, or, in other words, harmonic combinations. One way to specify them is using the `sidevec` parameter. When a QPSS analysis has one large tone and one moderate tone, the sideband is represented by the vector K^1 as $[K^1_1\ K^1_2]$. The corresponding frequency translation is

$$K^1_1 \times fund(qpss-large-tone) + K^1_2 \times fund(qpss-moderate-tone)$$

When there are `L` tones total in the QPSS analysis (`1` large tone and `L-1` moderate tones), there is also a given set of `n` integer vectors representing the sidebands

$$K^1 = \{ K^1_1, \ldots, K^1_j, \ldots, K^1_L \}$$
$$K^2 = \{ K^2_1, \ldots, K^2_j, \ldots, K^2_L \}$$

…
$K^n = \{ K^n_1, \ldots, K^n_j, \ldots, K^n_L \}$

The QPAC analysis computes the output frequency corresponding to each sideband as follows.

$$f^i(out) = f(in) + \sum_{i=1}^{L} \left(K^i_j \cdot f_j \right)$$

Where

- $f(in)$ represents the (possibly swept) input frequency
- f_j represents the `fundamental frequency` used in the corresponding QPSS analysis.

In modeling a down-converting mixer with *low-side* LO and with swept RF input frequency, the most relevant sideband for the IF output is `{-1,0}`. For an up-converting mixer with swept IF input frequency, the most relevant sideband for the RF output is `{1,0}`. In a typical IP3 measurement, IM1 will correspond to the {-1, 0} sideband and IM3 to the {-1, 2} sideband.

Enter the `sidevec` parameter as a sequence of integer numbers separated by spaces. For example, you would enter the set of vectors `{1 1 0} {1 -1 0} {1 1 1}` as follows

```
sidevec=[ 1 1 0 1 -1 0 1 1 1]
```

The easiest way to do this is using *Select from list* in the ADE environment. In the sidebands selection, choose *Select from list*, and then highlight the desired frequencies using the shift key and the left mouse button.

The other way to specify the sidebands is the `clockmaxharm` parameter and the `maxharms` parameter in the preceding QPSS analysis. Only the large tone, the first fundamental in QPSS, is affected by the QPAC `clockmaxharm` parameter value. It limits the maximum harmonic order of the large tone that will be considered. All the remaining tones, the moderate tones, are limited by the QPSS `maxharms` parameter value.

Given the following parameters

- In the QPSS analysis, $\texttt{maxharms=[}k^0_{max}\ k^2_{max} \ldots\ k^n_{max}\texttt{]}$
- In the QPAC analysis, $\texttt{clockmaxharm=}K_{max}$

The QPAC analysis output generates the following number of sidebands

$$(2*K_{max} + 1)*(2*k^2_{max}+1)*(2*k^3_{max}+1)*\ldots*(2*k^n_{max}+1)$$

Note: The number of sidebands you request does not substantially increase the simulation time.

For a QPAC analysis, the stimulus and response frequencies are usually different (this is an important way in which QPAC analysis differs from AC analysis). QPAC can calculate the mixing products that occur by mixing the input frequency specified in the QPAC analysis with the harmonics that are calculated by the QPSS analysis.

Use the QPAC `freqaxis` parameter to specify how to output the QPAC simulation results. For

- `freqaxis=`*in* the results are output versus the input frequency. This is not advised because the frequency of the output signal is lost.
- `freqaxis=`*out* the results are output versus the output frequency
- `freqaxis=`*absout* the results are output versus the absolute value of the output frequency

Swept QPAC Analysis

Specify sweep limits by providing either the end points (`start` and `stop`) or by providing the center value and the span of the sweep (`center` and `span`).

Specify sweep steps as linear or logarithmic as well as the number of steps or the size of each step. You can give a step-size parameter (`step`, `lin`, `log`, `dec`) to determine whether the sweep is linear or logarithmic. If you do not give a step-size parameter, the sweep is linear when the ratio of `stop` to `start` values is less than 10, and logarithmic when this ratio is 10 or greater.

Alternatively, you can specify the particular values that the sweep parameter should take using the `values` parameter. If you give both a specific set of values and a set specified using a sweep range, the two sets are merged and collated before simulation. All frequencies are in Hertz.

QPAC Parameters

Sweep Interval Parameters

`start=0`	Start sweep limit.
`stop`	Stop sweep limit.
`center`	Center of sweep.

`span=0`	Sweep limit span.
`step`	Step size, linear sweep.
`lin=50`	Number of steps, linear sweep.
`dec`	Points per decade.
`log=50`	Number of steps, log sweep.
`values=[...]`	Array of sweep values.
`sweeptype`	Specifies if the sweep frequency range is absolute (actual) frequency of input or if it is relative to the `relharmvec` sideband frequency. Here `relative` means relative to the input port frequency. Possible values are `absolute` or `relative`.
`relharmvec=[...]`	Sideband vector of QPSS harmonics. Relative frequency sweeps are referenced to the `relharmvec`.

Output Parameters

`sidevec=[...]`	Array of relevant sidebands for the analysis.
`clockmaxharm=0`	An alternative to the `sidevec` array specification, which automatically generates the array: `[-clockmaxharm ... 0 ... +clockmaxharm]` `[-maxharms(qpss)[2]...0...+maxharms(qpss)[2]]` `[...]`.
`freqaxis`	Specifies whether the results should be output versus the input frequency, the output frequency, or the absolute value of the output frequency. the default is `in` for logarithmic frequency sweeps and `absout` otherwise. Possible values are `absout`, `out` or `in`.
`save`	Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, or none.
`nestlvl`	Levels of subcircuits to output.

Convergence Parameters

`tolerance`	Relative tolerance for linear solver, default value is 1.0e-9 for Shooting-based solver; 1.0e-6 for harmonicbalance-based solver.
`gear_order=2`	Gear order used for small-signal integration. Possible values are 1 or 2.

`solver=turbo`	Solver type. Possible values are `std` or `turbo`.
`lnsolver=gmres`	Linear solver. Possible values are `bicgstab`, `gmres`, `qmr`, or `resgmres`.
`resgmrescycle=short`	Controls recycling and the length of the computation cycle. Possible values are `instant`, `short`, `long`, `recycleinstant`, `recycleshort`, and `recyclelong`.

Annotation Parameters

`annotate=sweep`	Degree of annotation. Possible values are no, title, sweep, status, or steps.
`stats=no`	Analysis statistics. Possible values are no or yes.
`title`	Analysis title.

Input source settings for QPAC

- PAC amplitude will be used for PAC and QPAC and HBAC analyses. When you set 1 volt, you are setting 1 volt peak.
- Set PAC amplitude to 1 volt on the RF source to measure conversion gain. Set PAC magnitude dBm in the small-signal range in the compression distortion summary.
- If you want to measure the AM to PM conversion from a power supply to the output of a circuit, set the PAC amplitude to 1 volt in the poser supply, and remove any PAC amplitude terms from all the other sources. Check the schematic so you can netlist without saving to the circuit file.

Measuring small-signal conversion gain as a function of blocker power

1. Set the LO and blocker frequencies in the variables section of ADE.

2. First, set up the QPSS analysis. For more information see page 1090.

3. Choose *qpac*.

4. Set the input frequency.

5. Set the highest harmonic of the large signal (Shooting) or the signal with tstab enabled (HB) you want to calculate results for. This is entered in the *Maximum Clock LOrder* field.

6. When the simulation completes, select *Results - Direct Plot - Main Form* in the ADE window.

7. Select *qpac* results.

8. Select what type of signal you want to plot. (For example, net or differential net.)

9. Select *variable sweep*.

10. Select *peak*. Remember that the PAC amplitude is always set to volts peak when the amplitude is in volts. (dBm is rms)

11. Select the appropriate *modifier*.

12. Select the output frequency you want to plot.

13. Select the net on the schematic.

The conversion gain goes down as the blocker amplitude goes up.

The next plot shows a non-swept qpss-qpac conversion gain measurement. To set up, set the following in ADE.

14. Set the blocker power by setting the variable values in the variables section of ADE.
15. Set the LO and blocker frequencies by setting the variable values in the variables section of ADE.
16. Disable the second RF input by setting its frequency to zero by setting the variable values in the variables section of ADE.

Setting up swept QPAC analysis

1. First, set up a non-swept QPSS. For more information, see page 1090.

2. Select *qpac*.

3. Select *Linear* or *Log*.

4. Set a desired input frequency range. In this example, the input is just above the LO frequency.

5. Do not specify too many points.

6. Select the highest harmonic number of the large signal (Shooting) or the signal with tstab enabled (HB) you want output for.

7. When the simulation completes, select *Results - Direct Plot - Main Form* in the ADE window.

8. Select *qpss results* to plot the output spectrum.

9. Select *spectrum*.

10. Select *rms*.
11. Select *dB20*.
12. Select the net in the schematic.

The plot for this will be shown in 2 pages.

13. Select *qpacresults*.

14. Select what you want to plot. A node voltage is shown below.

15. Select the type of signal you want to plot. One of the choices is differential nets.

16. Select *spectrum*. This shows all the output mixing products. If you only want to see a small number, select sideband where you can plot one at a time.

17. Select peak.

18. Select dB20.

19. Select the net in the schematic.

QPAC calculated the mixing products that are produced by mixing the input frequency range specified in the QPAC Choosing Analyses form with all the harmonics that are

calculated in the QPSS. Because there are a lot of harmonics in the QPSS, there are a lot of sidebands in the QPAC. The QPSS output is in the left pane of the waveform tool.

The little specs you see in the QPAC results in the right pane of the waveform tool are the output mixing products calculated by QPAC.

Given below is a zoomed in view. The QPSS result is on the left, the QPAC result is on the right. (See area of the plot encircled in red.)

20. To identify the harmonics that this signal is mixing with, select the trace in the QPAC display.

21. Scroll through the results until you find the one that is highlighted. Use the slider on the right.

22. Harmonic -1 0 means that the signal is mixed down with the first harmonic of the LO, and is not mixed with the blocker at all.

This is the 1 -2 sideband. It means the input is mixed up with the first harmonic of the LO, and mixed down with the second harmonic of the blocker. (See area of the plot encircled in red.)

This is the -3, 2 sideband. The frequency is mixed down with the third harmonic of the LO, and then mixed up with the second harmonic of the blocker. (See area of the plot encircled in red.)

Quasi-Periodic S-parameter Analysis (QPSP)

The quasi-periodic SP (QPSP) analysis computes scattering and noise parameters for n-port circuits that exhibit frequency translation. Such circuits include mixers, switched-capacitor filters, samplers, phase-locked loops, and the like.

QPSP is a quasi-periodic small-signal analysis similar to the SP analysis except that with QPSP the circuit is first linearized about a quasi-periodically time-varying operating point as opposed to either a simple periodically time-varying operating point or a DC operating point. Linearizing about a quasi-periodically time-varying operating point allows the computation of S-parameters between circuit ports that convert signals from one frequency band to another.

The QPSP analysis also calculates noise parameters in frequency-converting circuits. QPSP computes noise figure (single-sideband, double-sideband, and IEEE single-sideband), input referred noise, equivalent noise parameters, and noise correlation matrices. As is also true for QPnoise analysis, but unlike SP analysis, the noise features of the QPSP analysis include noise folding effects due to the quasi-periodically time-varying nature of the circuit.

Computing the n-port S-parameters and noise parameters of a quasi-periodically varying circuit is a two step process.

1. First, the small stimuli are ignored and the quasi-periodic steady-state response of the circuit to possibly large periodic stimuli is computed with a QPSS analysis. As a normal part of the QPSS analysis, the quasi-periodically time-varying representation of the circuit is computed and saved for later use.
2. Second, using the QPSP analysis, the small-signal excitations are applied to compute the n-port S-parameters and noise parameters.

QPSP Output Frequencies and Sideband Vectors

To specify the QPSP analysis, you must specify the physical ports and the port harmonics combinations that form the *virtual ports* of interest. In QPSP as in PSP, port sidebands are used to assign the frequency translation between ports.

- Set the `port` parameter for the physical ports of interest
- Set either the `portharmsvec` or the `harmsvec` parameter to select the quasi-periodic small-signal output frequencies, or harmonics, of interest
  - When you use the `portharmsvec` parameter, the harmonic vectors must be in one-to-one correspondence with the ports, with one harmonic combination associated with each physical port listed in the `port` list. That is in the presence of two tones

in QPSS, each physical port needs a combination of two harmonics assigned to it, so there are two integer numbers in `portharmsvec` for each entry in `port`.

- When you specify harmonic combinations with the optional `harmsvec` parameter, the QPSP analysis computes all possible frequency-translating scattering parameters associated with the specified harmonics.

- The easiest way to define harmonics in QPSP is to use the ADE GUI to select the desired ports and frequencies.

In quasi-periodic analyses sidebands are vectors, or, in other words, harmonic combinations. One way to specify them is using the `sidevec` parameter. When a QPSS analysis has one large tone and one moderate tone, the sideband is represented by a vector K^1 as $[K^1_1\ K^1_2]$. The corresponding frequency translation is

$$K^1_1 \times fund(qpss - large - tone) + K^1_2 \times fund(qpss - moderate - tone)$$

When there are `L` tones total in the QPSS analysis (`1` large tone and `L-1` moderate tones), there is also a given set of `n` integer vectors representing the sidebands

$$K^1 = \{ K^1_1, \ldots, K^1_j, \ldots, K^1_L \}$$
$$K^2 = \{ K^2_1, \ldots, K^2_j, \ldots, K^2_L \}$$
$$\ldots$$
$$K^n = \{ K^n_1, \ldots, K^n_j, \ldots, K^n_L \}$$

QPSP computes the S-parameters from each virtual port to all the others. The frequency translation from one virtual port to the other is calculated using the following.

$$f^i(scattered) = f^m(incident) + \sum_{i=1}^{L} \left(\left(K^i_j - K^m_j \right) \cdot f_j \right)$$

Where

- *f(scattered)* is the frequency to which the relevant scattering parameter represents the conversion
- *f(incident)* represents the relative frequency of a signal incident on a port
- f represents the `fundamental frequency` used in the corresponding QPSS analysis.

Input and Output Frequencies in QPSP

For the QPSP analysis, the frequency of the input and the frequency of the response are usually different (this is an important way in which QPSP analysis differs from SP analysis).

When you analyze a down-converting mixer with a signal in the upper sideband and you sweep the RF input frequency

- The most relevant harmonic for RF input is K^i={1,0}
- The most relevant harmonic for IF output is K^i={0,0}

Hence, you can associate

- K^1={1,0} with the RF port
- K^2={0,0} with the IF port.

The frequency translation will be the following

$$\Delta f(RFtoIF) = f(IF) - f(RF) = (0 \cdot f_1 + 0 \cdot f_2) - ((1 \cdot f_1 + 0 \cdot f_2)) = -f(LO)$$

- S_{21} represents the transmission of the signal from the RF port with $f(incident) = f(RF)$ to the IF port with $f(scattering) = f(IF) = f(RF) + \Delta f(RF\ to\ IF) = f(incident) - f(LO)$.
- S_{11} represents the reflection of the signal back to the RF port.
- S_{12} represents the transmission from the IF port with $f(incident) = f(IF)$ to the RF port with $f(scattering) = f(RF) = f(IF) - \Delta f(RF\ to\ IF) = f(incident) + f(LO)$

If the signal is in the lower sideband, then a choice of K_1 = {-1,0} is more appropriate.

Important

Because the QPSP computation involves inputs and outputs at frequencies that are relative to multiple harmonics, the `freqaxis` and `sweeptype` parameters behave somewhat differently in QPSP analysis than they do in both QPAC and QPXF analyses.

The `sweeptype` parameter controls the way the frequencies are swept.

- A *`relative`* `sweeptype` indicates a sweep relative to the first virtual port harmonics vector
- An *`absolute`* `sweeptype` indicates a sweep of the absolute input source frequency

For the relative sweep with the frequency f_{rel} and the first virtual port with harmonic vector

$$\begin{bmatrix} K_1^1 K_2^1 \end{bmatrix}$$

the incident and scatter frequencies for S_{21} are

$$f_{incident} = f_{rel} + K_1^1 \cdot f_1 + K_2^1 \cdot f_2$$

$$f_{scatter} = f_{rel} + K_1^1 \cdot f_1 + K_2^1 \cdot f_2 + \Delta f(RFtoIF) = f_{rel} + K_1^2 \cdot f_1 + K_2^2 \cdot f_2$$

while for S_{12} they are

$$f_{incident} = f_{rel} + K_1^2 \cdot f_1 + K_2^2 \cdot f_2$$

$$f_{scatter} = f_{rel} + K_1^2 \cdot f_1 + K_2^2 \cdot f_2 - \Delta f(RFtoIF) = f_{rel} + K_1^1 \cdot f_1 + K_2^1 \cdot f_2$$

For the *absolute sweeptype* the sweep frequency will be used as an incident frequency on the first virtual port in S_{21}

$$f_{incident} = f_{abs}$$

$$f_{scatter} = f_{abs} + \Delta f(RFtoIF) = f_{abs} + K_1^2 \cdot f_1 + K_2^2 \cdot f_2 - \left(K_1^1 \cdot f_1 + K_2^1 \cdot f_2\right)$$

or scatter frequency for S_{12}

$$f_{incident} = f_{abs} + \Delta f(RFtoIF) = f_{abs} + K_1^2 \cdot f_1 + K_2^2 \cdot f_2 - \left(K_1^1 \cdot f_1 + K_2^1 \cdot f_2\right)$$

$$f_{scatter} = f_{abs}$$

For example, with the following parameter values including `relative sweeptype`

- QPSS fundamentals of 1000 MHz and 1010 MHz

- `portharmsvec = [0 1 -1 1]` (to examine a down converting mixer)
- `sweeptype=relative`
- sweep range of *f(rel)*=`0 -> 5` MHz

The frequency translation

$$\Delta f = (-1 \cdot f_1 + 1 \cdot f_2) - (0 \cdot f_1 + 1 \cdot f_2) = -f(LO)$$

The resulting S_{21} represents the strength of the signal transmitted from the input at the first virtual port in the range 1010->1015 MHz to the output at the second virtual port at frequencies of 10->15 MHz. Accordingly, S_{12} represents the signal transmitted from the second virtual port with $f_{incident}$=10->15 MHz to the first virtual port at $f_{scatter}$ = 1010->1015 MHz.

Using the following changed parameter values including `absolute sweeptype` calculates the same quantities.

- `sweeptype=absolute`
- sweep range of *f(abs)*=`1010->1015` MHz

Both configuration calculate the same quantities

f(abs) = `1010->1015` MHz

$f(rel) = f(abs) - (K^1{}_1 * f_1 + K^1{}_2 * f_2)$= 0 -> 5 MHz

because

- $K^1{}_1$=`0`
- $K^1{}_2$=`1`
- f_1=`1000` MHz
- f_2=`1010` MHz.

Use the `freqaxis` parameter to specify whether the results should be output versus the frequency at the first virtual port, the frequency at the second virtual port (out), or the absolute value of the frequency swept at the first virtual port (`absin`).

Note: Requesting additional ports and harmonics increases the simulation time substantially.

Noise Analysis with QPSP

The QPSP analysis performs noise analysis which includes noise figures, equivalent noise sources, and noise parameters. The noise computation, which is performed by default and skipped only when you set `donoise=no`, requires additional simulation time beyond that required for S-parameter calculation.

Noise Calculations Performed by QPSP

Name	Description	Output Label
N_o	Total output noise at frequency f	
N_s	Noise at the output due to the input probe (the source)	
N_{si}	Noise at the output due to the image harmonic at the source	
N_{so}	Noise at the output due to harmonics other than input at the source	
N_l	Noise at the output due to the output probe (the load)	
IRN	Input referred noise	**In**
G	Gain of the circuit (See **Note:**)	**Gain**
F	Single sideband noise factor	**F**
NF	Single sideband noise figure	**NF**
F_{dsb}	Double sideband noise factor	F_{dsb}
NF_{dsb}	Double sideband noise figure	NF_{dsb}
F_{ieee}	IEEE single sideband noise factor	F_{ieee}
NF_{ieee}	IEEE single sideband noise figure	NF_{ieee}

Note: For the QPSP analysis, the gain computed is the voltage gain from the actual circuit input to the circuit output, not the gain from the internal port voltage source to the output. For the noise characterization the first virtual port is dedicated as the input, the second virtual port serves as the output.

SpectreRF performs the following noise calculations.

Input referred noise

$$IRN = \sqrt{\frac{N_o^2}{G^2}}$$

Single sideband noise factor

$$F = \frac{(N_o^2 - N_l^2)}{N_s^2}$$

Single sideband noise figure

$$NF = 10 \times \log 10(F)$$

Double sideband noise factor

$$F_{dsb} = \frac{N_o^2 - N_l^2}{N_s^2 + N_{si}^2}$$

Double sideband noise figure

$$NF_{dsb} = 10 \times \log 10(F_{dsb})$$

IEEE single sideband noise factor

$$F_{ieee} = \frac{N_o^2 - N_l^2 - N_{so}}{N_s^2}$$

IEEE single sideband noise figure

$$NF_{ieee} = 10 \times \log 10(F_{ieee})$$

Noise Folding Effects

To insure accurate noise calculations, set the `clockmaxharm` parameter to include the relevant noise folding effects.The `clockmaxharm` parameter is only relevant to the noise computation features of QPSP.

Swept QPSP Analysis

Specify sweep limits by providing either the end points (`start` and `stop`) or by providing the center value and the span of the sweep (`center` and `span`).

Specify sweep steps as linear or logarithmic. Either specify the number of steps or the size of each step. You can give a step-size parameter (`step`, `lin`, `log`, `dec`) to determine whether the sweep is linear or logarithmic. If you do not give a step-size parameter, the sweep is linear when the ratio of `stop` to `start` values is less than 10, and logarithmic when this ratio is 10 or greater.

Alternatively, you may specify the particular values that the sweep parameter should take using the `values` parameter. If you give both a specific set of values and a set of values specified using a sweep range, the two sets are merged and collated before being used. All frequencies are in Hertz.

QPSP Parameters

Sweep Interval Parameters

`start=0`	Start sweep limit.
`stop`	Stop sweep limit.
`center`	Center of sweep.
`span=0`	Sweep limit span.
`step`	Step size, linear sweep.
`lin=50`	Number of steps, linear sweep.

`dec`	Points per decade.
`log=50`	Number of steps, log sweep.
`values=[...]`	Array of sweep values.
`sweeptype=relative`	Specifies if the sweep frequency range is an absolute frequency, i.e. actual frequency; or if it is relative to the "relharmvec" sideband frequency. In QPSP, relative means relative to the input port frequency. Possible values are `absolute` or `relative`.

Port Parameters

`ports=[...]`	List of active ports. Ports are numbered in the order given. For purposes of noise figure computation, the input is considered port 1 and the output is considered port 2.
`portharmsvec=[...]`	List of the reference sidebands for the specified list of ports. The `portharmsvec` vector must have a one-to-one correspondence with the `ports` vector.
`harmsvec=[...]`	Secondary list of additional, active sidebands. Sidebands included in the `harmsvec` vector are in addition to those associated with specific ports by `portharmsvec`.

Output Parameter

`freqaxis`	Specifies whether the results should be output versus the input port frequency, the output port frequency, or the absolute value of the input frequency. Default is `in`. Possible values are `absin`, `in` or `out`.

Noise Parameter

`donoise=yes`	Perform noise analysis. If *oprobe* is specified as a valid port, `donoise` is set to *`yes`* and a detailed noise output is generated. Possible values are `no` or `yes`.

Probe Parameter

`clockmaxharm=7`	In Shooting pnoise, the parameter determines the maximum sideband included when computing noise either up-converted or down-converted to the output by the periodic drive signal. The default value for the Shooting pnoise is 7. In HB pnoise, this parameter determines the size of the small signal system when the HB pnoise is performed. This parameter is critical for the accuracy of the HB pnoise analysis: using small maxsideband may cause accuracy loss. The default value for the HB pnoise is the `harms`/`maxharms` setting in the HB large signal analysis.

Convergence Parameters

`tolerance`	Relative tolerance for linear solver. Default value is $1.0e^{-9}$ for Shooting-base solver. Default value is $1.0e^{-6}$ for harmonic-balance-based solver.
`gear_order=2`	Gear order used for small-signal integration, 1 or 2.
`solver=turbo`	Solver type. Possible values are `std` or `turbo`.
`lnsolver=gmres`	Linear solver. Possible values are `bicgstab`, `gmres`, `qmr`, or `resgmres`.
`resgmrescycle=short`	Controls recycling and the length of the computation cycle. Possible values for large signal analysis are `instant`, `short`, and `long`. Possible values for small signal analyses are `instant`, `short`, `long`, `recycleinstant`, `recycleshort`, and `recyclelong`.

Annotation Parameters

`annotate=sweep`	Degree of annotation. Possible values are no, title, sweep, status, or steps.
`stats=no`	Analysis statistics. Possible values are no or yes.
`title`	Analysis title.

QPSP Setup

1. First set up the LO and blocker frequencies and amplitudes in ADE by setting the variable values.

2. Set up the QPSS analysis. For more information, please see page 1090.

3. Select *qpsp*.

4. Relative means relative to the frequency specified in the Choosing Analyses form. If absolute is selected, port 1 will receive this frequency.

5. If relative is selected, then the frequency range can be either input or output frequency range. This example shows an output of a down-conversion mixer.

6. Do not take too may points.

7. Type 1 in the *Port Number* field.

8. Place the cursor in the *Name* field.

9. Click *Select Port*.

10. Select the input port in the circuit.

11. Click in the *Harm* field.

12. Select *Choose Harmonic*.

13. Choose the input frequency range you want from the list.

14. Select *OK*.

There will be an entry with the same number of number as there are tones in the circuit. This example has 2 input tones in QPSS so there are two numbers in the list.

1 0 means that compared to the frequency specified in the form, the input frequency is shifted by +1*frequency of the LO plus 0*frequency of the blocker.

15. Select *Add*. If you forget this step, the port definition will not be simulated.

When you select *Add*, a line appears in the list box.

16. Type 2 in the *Port Number* field.

17. Click in the *Name* field.

18. Click *Select Port*.

19. Select the output port in the schematic.

20. Click in the *Harm* field.

21. Select *Choose Harmonic*.

22. Select the output frequency range.

23. Select *OK*.

0 0 means that the frequency in the Choosing Analyses form is not shifted at all by either input.

24. Select *Add*. If you forget this step, the port definition will not be simulated.

When you select *Add*, the port definition appears in the list box.

Plotting S-parameter results.

1. When the simulation completes, select *Results - Direct Plot - Main Form* in the ADE window.
2. Select *qpsp results*.

3. Select *SP*.

4. Select the type of plot you would like to see.

5. Select the modifier.

6. Select the desired S-parameter measurement.

The forward power gain (S21) is about 5.2 dB close in to the LO.

Plotting noise figure

1. Select *qpsp results*.

2. Select the type of noise figure you want to plot. `NF` is single-sideband noise figure. `F` is noise factor single sideband. `Nfminis` the minimum achievable noise figure. `Fieee` is the IEEE single-sideband noise factor. `Nfieeis` the IEEE single-sideband noise figure. `Fdsbis` double-sideband noise factor. `NFdsbis` the double-sideband noise figure.

3. Select Plot.

The noise figure plots.

Quasi-Periodic Transfer Function Analysis (QPXF)

A conventional transfer function (QPXF) analysis computes the transfer function from every source in the circuit to a single output. It differs from a conventional AC analysis in that the AC analysis computes the response from a single stimulus to every node in the circuit. The differences between the QPAC and QPXF analyses are similar. The Quasi Periodic Transfer Function or QPXF analysis computes the transfer functions from all the sources at any frequency to a single output at a single frequency. Thus, like QPAC analysis, QPXF analysis includes frequency conversion effects.

The QPXF analysis directly computes

- Conversion Efficiency --The transfer function from the input to the output at a specified frequency.
- Image Rejection and Sideband Rejection -- The transfer function from the input to output at an undesired frequency.
- LO Feed-Through and Power Supply Rejection -- The transfer function from an undesired input to output at all frequencies.

Computing the small-signal response of a quasi-periodically varying circuit is a two step process.

1. First, the small stimulus is ignored and the quasi-periodic steady-state response of the circuit to possibly large periodic stimuli is computed with a QPSS analysis. As a normal part of the QPSS analysis, the quasi-periodically time-varying representation of the circuit is computed and saved for later use.
2. Second, the small stimulus is applied to the quasi-periodically time-varying linear representation and the small signal response is computed. This is done using the QPXF analysis.

QPXF Output Frequencies and Sideband Vectors

Either voltage or current can be the QPXF output variable of interest. The variable's frequency is not constrained by the period of the large signal quasi-periodic solution. When you sweep a selected output frequency, you select the set of quasi-periodic small-signal input frequencies you are interested in by setting either the `clockmaxharm` or the `sidevec` output parameter.

In quasi-periodic analyses sidebands are vectors, or, in other words, harmonic combinations. One way to specify them is using the `sidevec` parameter. When a QPSS analysis has one

large tone and one moderate tone, the sideband is represented by a vector K^1 as $[K^1_1\ K^1_2]$. The corresponding frequency translation is

$$K^1_1 \times fund(qpss - large - tone) + K^1_2 \times fund(qpss - moderate - tone)$$

When there are `L` tones total in the QPSS analysis (`1` large tone and `L-1` moderate tones), there is also a given set of `n` integer vectors representing the sidebands

$K^1 = \{ K^1_1, \ldots, K^1_j, \ldots, K^1_L\}$
$K^2 = \{ K^2_1, \ldots, K^2_j, \ldots, K^2_L\}$
…
$K^n = \{ K^n_1, \ldots, K^n_j, \ldots, K^n_L\}$

The QPXF analysis computes the input frequency corresponding to each sideband as follows.

$$f^i(in) = f(out) + \sum_{i=1}^{L} \left(K^i_j \cdot f_j \right)$$

where

- *f(out)* represents the (possibly swept) output signal frequency
- f_j represents the `fundamental frequency` used in the corresponding QPSS analysis.
- The easiest way to do this is to use the ADE GUI where the sidebands selection is set to *Select from list*, and then the individual input frequencies to calculate are selected using the control key and the left mouse button.

When you analyze a down-converting mixer with an LO and a single RF input frequency, while sweeping the IF output frequency

- K_i=`{1,0}` for the RF input represents the first upper-sideband
- K_i=`{-1,0}` for the RF input represents the first lower-sideband.

Enter the `sidevec` parameter as a sequence of integer numbers separated by spaces. For example, you would enter the set of vectors `{1 1 0} {1 -1 0} {1 1 1}` as follows

```
sidevec=[ 1 1 0 1 -1 0 1 1 1]
```

The easiest way to accomplish this is using the ADE Choosing Analyses form.

The other way to specify the sidebands is the `clockmaxharm` parameter and the `maxharms` parameter in preceding QPSS analysis. Only the large tone, the first fundamental in QPSS, is affected by the QPAC `clockmaxharm` parameter value. It limits the maximum harmonic order of the large tone that will be considered. All the remaining tones, the moderate tones, are limited by the QPSS `maxharms` parameter value.

Given the following parameters

- In the QPSS analysis, `maxharms=[`k^0_{max} k^2_{max} ... k^n_{max}]
- In the QPXF analysis, `clockmaxharm=`K_{max}

The QPXF analysis output generates the following number of sidebands

$(2*K_{max}+1)*(2*k^2_{max}+1)*(2*k^3_{max}+1)*...*(2*k^n_{max}+1)$

Note: The number of sidebands you request does not substantially increase the simulation time.

For a QPXF analysis, the stimulus and response frequencies are usually different (this is an important way in which QPXF differs from XF analysis).

Use the QPXF `freqaxis` parameter to specify how to output the QPXF simulation results. For

- `freqaxis=`*in* the results are output versus the input frequency
- `freqaxis=`*out* the results are output versus the output frequency. This is not recommended because the frequencies of the input signals are lost.
- `freqaxis=`*absin* the results are output versus the absolute value of the input frequency

You can specify the output for the QPXF analysis with either a probe component or with a pair of nodes. Any component with two or more terminals can be a voltage probe. When there are more than two terminals, the terminals are grouped in pairs and you use the `portv` parameter to select the appropriate pair of terminals. Alternatively, you can simply give a pair of nodes to specify a voltage as the output.

Any component that naturally computes current as an internal variable can be a current probe. If the probe component computes more than one current, use the `porti` parameter to select the appropriate current. Do not specify both `portv` and `porti` parameters for a simulation. If you specify neither a `portv` or `porti` parameter, the probe component provides a reasonable default.

Transfer Function Inputs

The QPXF `stimuli` parameter specifies the transfer function inputs. You have two choices.

- The `stimuli=sources` parameter value uses the sources present in the circuit as inputs. You can adjust the computed gain to compensate for gains or losses in a test fixture with the `xfmag` parameters provided by the sources. You can use the `save` and `nestlvl` parameters to limit the number of sources in hierarchical netlists.

The `stimuli=nodes_and_terminals` parameter value computes all possible transfer functions. This is useful when you do not know in advance which transfer functions might be interesting. In the ADE environment, these data deom the nodes and terminals don't have direct plot functions. The data must be plotted from the results browser and the calculator.

Transfer functions for nodes are computed assuming that a unit magnitude flow (current) source is connected from the node to ground. Transfer functions for terminals are computed assuming that a unit magnitude value (voltage) source is connected in series with the terminal.

By default, the transfer functions are computed from a small set of terminals.

- If you want transfer functions from specific terminals, specify the terminals in the `save` statement. Use the `:probe` modifier (ex. `Rout:1:probe`) or specify `useprobes=yes` on the `options` statement.
- If you want transfer functions from all terminals, specify `currents=all` and `useprobes=yes` on the `options` statement.

Swept QPXF Analysis

Specify sweep limits by providing either the end points (`start` and `stop`) or by providing the center value and the span of the sweep (`center` and `span`).

Specify sweep steps as linear or logarithmic as well as the number of steps or the size of each step. You can give a step-size parameter (`step`, `lin`, `log`, `dec`) to determine whether the sweep is linear or logarithmic. If you do not give a step-size parameter, the sweep is linear when the ratio of `stop` to `start` values is less than 10, and logarithmic when this ratio is 10 or greater.

Alternatively, you may specify the particular values that the sweep parameter should take using the `values` parameter. If you give both a specific set of values and a set specified using a sweep range, the two sets are merged and collated before being used. All frequencies are in Hertz.

QPXF Parameters

Sweep Interval Parameters

`start=0`	Start sweep limit.
`stop`	Stop sweep limit.
`center`	Center of sweep.
`span=0`	Sweep limit span.
`step`	Step size, linear sweep.
`lin=50`	Number of steps, linear sweep.
`dec`	Points per decade.
`log=50`	Number of steps, log sweep.
`values=[...]`	Array of sweep values.
`sweeptype`	Specifies if the sweep frequency range is absolute (actual) frequency or if it is relative to the `relharmvec` sideband frequency. Here `relative` means relative to the input port frequency. Possible values are `absolute` and `relative`.
`relharmvec=[...]`	Sideband vector of QPSS harmonics. Relative frequency sweep is referenced to `relharmvec`.

Probe Parameters

`probe`	Compute every transfer function to this probe component.

Output Parameters

`stimuli=sources`	Stimuli used for PXF analysis. Possible values are `sources` or `nodes_and_terminals`.
`sidevec=[...]`	Array of relevant sidebands.
`clockmaxharm=0`	An alternative to the `sidevec` array specification, which automatically generates the array: `[ - clockmaxharm ... 0 ... +clockmaxharm ]` `[- maxharms(QPSS)[2]...0...maxharms(QPSS)[2]]` `[...]`.

freqaxis	Specifies whether the results should be output versus the input frequency, the output frequency, or the absolute value of the input frequency. The default is out for logarithmic frequency sweeps and absin otherwise. Possible values are absin, in or out.
save	Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, or none.
nestlvl	Levels of subcircuits to output.

Convergence Parameters

tolerance	Relative tolerance for linear solver, default value is 1.0e-9 for Shooting-based solver; 1.0e-6 for harmonicbalance-based solver.
gear_order=2	Gear order used for small-signal integration. Possible values are1 or 2.
solver=turbo	Solver type. Possible values are std or turbo.
lnsolver=gmres	Linear solver. Possible values are bicgstab, gmres, qmr, or resgmres.
resgmrescycle=short	Controls recycling and the length of the computation cycle. Possible values for large signal analysis are instant, short, and long. Possible values for small signal analyses are instant, short, long, recycleinstant, recycleshort, and recyclelong.

Annotation Parameters

annotate=sweep	Degree of annotation. Possible values are no, title, sweep, status, or steps.
stats=no	Analysis statistics. Possible values are no or yes.
title	Analysis title.

Setting up QPXF Analyses

1. Set up the LO and blocker amplitudes and frequencies.

2. Set up the QPSS analysis. For more information, see page 1090.

3. Select *qpxf*.

4. Type in an appropriate output frequency range for your application.

5. Do not specify too many points.

6. Type in the highest harmonic of the LO you want transfer functions for.

7. Click in the *Positive Output Node* field.

8. Click *Select*.

9. Select the output node in the schematic.

   If you have a differential circuit, use the same technique for selecting the other output. Leaving a Negative Node blank causes the global ground node to be selected automatically.

10. When the simulation completes, select *Results - Direct Plot - Main Form* in the ADE window.

11. Select *qpxf* results.

12. Select *Voltage Gain*.

13. Select *dB20*.

14. Select the input port in the schematic for the transfer functions from the input to the output. Select the power supply in the schematic to measure the PSRR.

There will be a transfer function on both sides of all the harmonics in the QPSS. This is a zoomed in view.

The rest of the harmonics are the mixing products from these 2 inputs.

There is some overlap in the curves. The blocker is 70MHz above the LO. The output frequency goes up to 50MHz. Thus the transfer functions above the LO extend to 50MHz above the LO frequency, and the transfer function below the blocker extends to 50MHz down from the blocker frequency. The overlap is 30MHz wide. If you imagined a vertical marker placed at 1.940GHz, it would intersect two curves. One is for the gain near the LO and one is the gain near the blocker.

These are the transfer functions from the power supply to the output.

- If there is a curve above unity gain, then there is the potential for oscillations.
- Note there is a curve near zero Hertz and zero dB that goes above 0dB.

This is the zoomed in view. This circuit is a potential oscillator.

6

Envelope (ENVLP) Analysis

Envelope analysis is used instead of a transient analysis when there are fast things and slow things in the same circuit. The simulation term for this condition is called multi-rate simulation. In the transient, a large number of cycles of the fast signal must be simulated in order to capture the slow behavior in the circuit. Instead of simulating every fast cycle as in the transient, envelope skips many cycles of the fast signal because the change is very slow. Although many cycles are skipped, the result is almost as accurate as transient analysis.

The primary application for envelope analysis is for digital modulation of a high frequency carrier, or calculating the startup waveform of a high-Q oscillator.

Envelope can solve for one cycle of the RF carrier either by solving for the waveform when shooting is selected, or by using harmonic balance when harmonic balance is selected. For very nonlinear systems like polar modulation, use shooting. For relatively more linear circuits like and I/Q modulator and or power amplifier, use harmonic balance. Harmonic balance has the advantage of having two fast envelope modes that can dramatically speed up simulation.

Harmonic balance envelope has three different algorithms. The first is to simulate everything at the transistor level. Level1 characterizes the circuit at the start of the simulation, and builds a behavioral model that is then run instead of the circuit. Level1 is useful for circuits that donot have memory effects. Level2 characterizes the circuit and builds a behavioral model at the beginning, and then runs that behavioral model, however, level 2 does take into account mild to moderate memory effects. Because the characterization is more complete and the behavioral model is more complex, level2 takes longer to run than level1, but it is still much faster than running at the transistor level.

Example

Consider a circuit with an RF signal source, a behavioral PA, and a 50 ohm resistor, as shown below.

The rfVsource from analogLib is set to produce a 2.45GHz signal at +10dBm, and is using the 20MHz legacy 802.11n signal.

The behavioral PA has 20dB of gain, and is very linear.

Now start ADE, and open the envelope *Choosing Analyses* form. The form is blank.

Now click *Start ACPR Wizard*. This is the easiest way to set up a simulation where a power spectral density (PSD) plot is needed.

To set up this form, complete the following steps:

1. Type a dummy name in in the *Clock Name* field.
2. Select *Net* or *Differential Nets* from the *How to measure* drop-down list.
3. Click *Select* located to the right of the *Net* field, and then click the output net in the schematic.
4. Select the standard you want from the *Channel Definitions* pull-down menu.
5. Set the symbol start time to a time just larger than zero. After you entera value and move out of that field, the number will change to the next time that is in the modulation I and Q file.
6. All the fields below are populated with appropriate entries for the standard you selected.
7. Select the appropriate window function. A good, general-purpose window function is the *Hanning* window. Communicate with the system designer to determine the window function choice.
8. Click *OK*.

This will set the step period in the envelope *Choosing Analyses* form, and it will enter expressions in the output section of the ADE window to plot the PSD (power spectral density, or frequency response) and ACPR measurements for the upper and lower channel.

In the Envelope *Choosing Analyses* form:

1. Set the carrier frequency, and number of harmonics. Usually, oversample should be set to 1 when harmonic balance is used. Also, if you are making an ACPR measurement, select conservative accuracy.

2. Click *OK*, and start the simulation. At the end of the simulation, the PSD will plot. The spectral mask will also be plotted.

3. The ACPR measurement will be shown in the outputs section of the ADE, as shown below.

4. To plot a constellation diagram, select *Results - Direct Plot - Main Form* in ADE.

5. Select *EVM*.
6. Select the first harmonic.
7. Select *OFDM*.
8. Select the Signal Standard.
9. Type 0 (zero) in the *Symbol Start* field for all the standards except for *802.11a*, which should be *4u*.
10. The rest of the properties are entered based on the signal standard.
11. Select the net in the schematic.
12. After a few seconds, the constellation diagram is displayed.

13. The EVM appears for the random data that is 64-QAM modulated, and for the signal in the control channels which is BPSK modulated.

Envelope Principles

Harmonic Balance

Use harmonic balance if the circuit has sinusoidal waveforms and is fairly linear. This is true for circuits with sinusoidal waveforms operating below compression. If polar modulation is used where the circuit is operated in a very nonlinear mode, then shooting is suggested.

Basic Principles

Consider an amplifier that is being driven by an unmodulated source. Because of the finite bandwidth of the system, there is some startup behavior, and then the amplifier reaches steady-state.

The time-zero timepoint is the initial transient solution, which is a DC analysis that observes all the time-zero values for all the waveforms in the system.

The waveform above is an actual starting waveform for harmonic balance. Vertical markers have been placed at 400nsec intervals, which correspond to one cycle of 2.5GHz.

When the system becomes predictable with a low order polynomial, harmonic balance envelope begins to skip cycles. The number of cycles it skips depends on whether fixed or adaptive timestep control is selected. For fixed timestep control, the next strobing point is simulated. For adaptive timestep control, the number of cycles skipped depends on the curvature of the envelope. For an unmodulated sinusoid, conceptually, the cycle skipping could continue to grow to the end of the simulation because the same waveform repeats over and over again.

Note that for an accurate PSD and ACPR measurement, the data needs to be strobed at the time interval in the original I and Q modulation file. In harmonic balance, there does not need to be an integer number of skipped cycles.

If that delta-time was 8.3333333333nsec, to get an integer number of skipped cycles, the carrier frequency needs to be an integer multiple of 120MHz. Note that when the carrier is 2.4GHz, this is a multiple of 20. Because of the integer multiple, the envelope time-domain waveform shows a horizontal line between the cycles of simulation.

When the carrier is 2.41GHz, there is a small offset from the actual period of the carrier frequency. This causes a sinusoidal line to appear between the cycles that are simulated. This is because the periods do not exactly align.

Note that harmonic balance envelope is not a series of steady-state harmonic balance simulations. It is a hybrid of transient and harmonic balance.

Consider an amplifier followed by a diode peak detector shown below.

At time zero, there is no signal, so the filtered output at the net *out2* starts at zero. The resistors and capacitors cause the signal to ramp up in an exponential fashion.

If the envelope simulation had been steady-state at each timepoint, the voltage at the net out2 would have been constant and non-zero from the start.

For systems that can have complex behavior, make sure that adaptive time step control is selected. This allows the number of skipped cycles to be dynamic along with the variations in the envelope signal. If this system had been an analog AGC circuit, the feedback might have resulted in overshooting and ringing. This behavior could accurately be determined by using adaptive timestep control.

Circuit-Level and Fast Envelope

Circuit-level simulation is the default and is used when the *Fast envlp mode* checkbox is either not selected, or set to *off*.

When *level1* is selected, the I and Q modulating sources in the circuit need to be identified in the *Choosing Analyses* form. The amplitude of the I source is varied over a range with 30 datapoints being the default number of voltages selected for the I amplitude with large-signal harmonic balance analysis being run at each amplitude of the I source. In rare instances, 30 points is not enough to accurately characterize the system, and in that case, set the *sweepmethod* option to *fine*, which will set 50 points for the characterization rather than 30.

Then a behavioral model is created. The assumption here is that the frequency response of the circuit is constant over the bandwidth of the channel. The behavioral model only has compression effects that are identified by the harmonic balance measurements that are run at the start of the analysis. Another assumption is that the I and Q responses are identical and have ideal quadrature behavior. If a transmit mixer is included in the simulation, level1 and level 2 cannot be used to measure what happens when the amplitude of the I and Q LO signals are mismatched and/or the phase of the I and Q LO signals are not in perfect quadrature.

This behavioral model is then simulated instead of the transistor-level circuit. When the behavioral circuit is simulated, all the detail of what happens inside the circuit is lost. Only the data for the list of nodes specified in the *Output Nodes* field is available for plotting. Usually, only the input node and the output node is the list.

Level2 is similar to level1, only an hbac analysis is run in addition to the hb analysis. This allows the measurement of the frequency response of the system. The bandwidth of the hbac analysis is defined by the *Modulation Bandwidth* property in the *level2 Choosing Analyses* form. Including the frequency response effects allows mild to moderate memory effects to be included in the simulation. Because the characterization is more complete, and the behavioral model is more complex, the runtime is longer than level1, but is still considerably faster than running at the transistor level.

Shooting

Shooting is suggested if polar modulation is used where the circuit is operated in a nonlinear mode. Use harmonic balance if the circuit has sinusoidal waveforms and is fairly linear. This is true for circuits with sinusoidal waveforms operating below compression.

Consider an amplifier that is being driven by an unmodulated source. Because of the finite bandwidth of the system, there is some startup behavior, and then the amplifier reaches steady-state.

The time-zero timepoint is the initial transient solution, which is a DC analysis that observes all the time-zero values for all the waveforms in the system. At this point, the only thing that

is known is the time-zero value, so the projected ending point of the first period is the starting point.

The waveform above is an actual starting waveform for shooting. Vertical markers have been placed at 400nsec intervals, which correspond to one cycle of 2.5GHz.

Note that at the end of the first period, the ending point misses the starting point by a lot. This error between the predicted and actual ending point is used to control the simulation. In this case, the error is large, so the transient continues for another period. The beginning and ending points of the first cycle allow a prediction of a ramp as the ending point for the second cycle.

When the system becomes predictable with a low order polynomial, shooting envelope begins to skip cycles. First, it will skip one period, which is done twice in a row in the above example. Then it skips 2 periods, and then it skips 4 periods. For an unmodulated sinusoid, conceptually, the cycle doubling could continue to the end of the simulation because the same waveform repeats over and over again.

Shooting has the limitation that only an integer number of cycles can be skipped. This is because the starting and ending voltages need to be predictable, which happens only when an integer number of cycles is skipped.

When digital modulation is present, the number of skipped cycles depends on the curvature of the envelope. When the envelope has little curvature, many cycles can be skipped. When the envelope has a large curvature, not many cycles can be skipped.

In order to maintain accuracy, adaptive timestep needs to be used with shooting to allow the envelope algorithm to maintain the accuracy of the simulation.

Note that for an accurate PSD and ACPR measurement, the data needs to be strobed at the time interval in the original I and Q modulation file. For maximum accuracy, the carrier frequency needs to be an integer multiple of the reciprocal of the delta-T in the I and Q modulation file.

If that delta-time was 8.3333333333nsec, the carrier needs to be an integer multiple of 120MHz. Note that when the carrier is 2.4GHz, this is a multiple of 20. The sampled carrier cycles are exactly aligned with the data in the I and Q modulation file.

When the carrier is 2.45GHz, because of the limitation of skipping an integer number of cycles, the timing of the actual points in the envelope is not an integer multiple of 8.3333nsec. This causes small interpolation errors in the result due to the time mismatch.

Shooting envelope is a general-purpose extension of the transient and is useful for calculating the startup behavior of many circuits that have very long time constants and very short time constants in the same circuit. An example is analog AGC (Automatic Gain Control).

As an example, consider an amplifier that has a diode peak detector after the amplifier.

The capacitor voltage starts at zero because there is no signal at time zero, and then has an exponential charging curve set by the resistors and capacitor in the peak detector.

ADE Implementation and Detailed Envelope Settings for a Driven Circuit

First a modulation source is needed. Cadence provides a component in *rfLib* called *rfVsource*, which has modulated signals for the *802.11n* standards.

If you are using a different standard, you will need to get a piecewise linear I and Q modulation file from your system design group. These files will have a constant delta-time between the points. This delta-time should be 1/(channel bandwidth*6) or smaller if you need an ACPR measurement, and 1/(channel bandwidth*10) or smaller if you need a second adjacent channel power ratio calculation. This is required to meet the Nyquist criteria for the modulation.

To create the modulated RF signal from the I and Q modulation files, either a port component from *analogLib* or *ACPR_Source* from *rfLib* can be used. If the port is used, you need to scale the amplitude of the RF signal to achieve the correct power at the output of the port. If

you use the *rfVsource*, it will read the I and Q data files, and the power in dBm can be set directly on the source. Note that this power will be achieved when you simulate to exactly the end time in the I and Q modulation files. If you simulate a shorter time, due to the random nature of the data, the power will only be approximate. The properties list for the port is shown below.

The properties list for the ACPR source is shown below.

The raised cosine filter should not be needed for most simulations.

Setting up the Envelope Analysis in ADE

1. Examine the I and Q modulating file. If you are using *rfVsource*, all the values you need are set when you use the ACPR wizard.
2. Calculate 1/time difference in these files. If you are using shooting, consider making the carrier frequency in the design an integer multiple of this frequency. This will reduce the numerical noise floor of the simulation. For harmonic balance, it is not needed unless you

are trying to determine the phase of the signal based on the time-domain result. This frequency has been set in the ADE variable *frf* shown below.

3. Now select *Analysis - Choose* or the *Choosing Analyses* icon on the right side of the ADE window.

4. Select *envlp*. The ENVLP *Choosing Analyses* form is displayed.

5. Now click the *Start ACPR Wizard* button.

The ACPR wizard is displayed.

6. From the *Clock Name* drop-down list, select the name that is in the *Frequency name* 1 or 2 property that sets the carrier frequency in the circuit. If you are using *rfVsource* or *ACPR_source*, this name is *fc*.
7. Select *Net* or *Differential net*, depending on your design.
8. Click *Select* on the right of the *Net* field and then select the output net(s) in the schematic.
9. If you are working on a Qualcomm standard, select *Power Density*. Otherwise, select *Power*.

10. If you are working on an 802.11n standard, select the standard from theChannel Definitions drop-down list. The rest of the fields in the form will have starting values. Please see the discussion for Symbol Start, Window Size, and Repetitions below.

11. If you are working on another standard, select Custom.

12. Type the channel bandwidth in the *Main Channel Width* field.

13. In the Adjacent frequencies section select *low*.

The edit fields below this section become active.

14. Type the frequency limits for the first low adjacent channel for the ACPR calculation. Note that these frequencies are referenced to the center of the main channel. For frequencies on the low side of the main channel, the numbers will be negative.

15. Select *Change*.

This moves the contents of the edit field up into the list.

16. In a similar manner, select *high*, enter the frequency range, and click *Change*.

17. If you have second adjacent specifications, add the name and frequency limits in the edit field above, and then click *Add*.

18. Set *Symbol Start(Sec)* field to the time when you want the Fourier transform to begin. In many I and Q files, there is a time delay caused by digital filtering. The start time for the frequency response calculation needs to be after that delay time. If you are unsure,

simulate the I and Q file in a piecewise-linear source, and watch for the time that the output becomes non-zero, and pick a time slightly later than that.

19. Set the strobe period to the delta-time in the I and Q modulation file. If you are unsure, view the file in a text editor, and look at the left entry of the second line. The first line should have a left entry of zero (seconds) The second entry left entry (in seconds) should be entered in the strobe period field. The right entry is the voltage at the time in the left column.

Note: When you enter a strobe period, the ACPR wizard will update the Symbol Start field so that it is an integer multiple of the strobe period.

The Fourier transform used to calculate the frequency response of the circuit is inherently noisy. To make the spectrum less noisy, the *psd* function is used. The *psd* function splits the time that is simulated into smaller pieces, which allows multiple shorter time sequences to be considered individually. A fourier transform is performed on all the pieces individually, and then the power of all the first harmonics is averaged. This is done for the rest of the harmonics also. Because of the averaging, the noise is reduced. Because a shorter time exists in each piece, the frequency of the first harmonic goes up. In the *psd*, this reduces the resolution (the number of frequency points analyzed in the

spectrum). Generally, a trade-off is needed between the number of pieces, the length of the simulation, and the noise in the psd calculation.

20. Window size is the number of delta-times in the I and Q input files you want to include for the spectral calculation. This needs to be a power of two. Generally, this is in the range of 256 to 8192.

21. Specify repetitions as a power of 2. This will run as many windows sizes as you specify.

22. The *Symbol Stop(Sec)* is calculated based on the symbol start, window size, and repetitions fields. Make sure that the data in your I and Q modulation file has entries to at least this time. View the I and Q modulation file, and look at the left entry (the time) in the last line of the file to make sure that ending time is larger than the *Symbol Stop(Sec)* time in the ACPR wizard.

The resolution bandwidth is calculated based on the entries for strobe period and window size. It is the frequency resolution of the Fourier transform of the output signal.

23. Because the data bits in the modulation are random, a Fourier transform window function needs to be selected. There are advantages and disadvantages to all window functions. The *Hanning* window is a reasonable window if the window function is not specified by the standard. Check with your system designer to find out his or her preference.

24. When you are done, click *OK*.

In the *Choosing Analyses* form for ENVLP, the *Stop Time* field is populated. The *Time Step Contro*l is also set to *fixed* and the *Step Period* field has the entry that you specified for the strobing interval in the ACPR wizard. A number of envelope options are

also set to provide the output from the simulator at the times needed for the FFT-based frequency response calculation.

25. Either type the frequency of the carrier, the number of harms, and the oversample factor, or select names, and set the desired number of harmonics.

26. If you have square waves in your circuit, use shooting. If you decide to use harmonic balance, you need relatively more harmonics. In this case, also set oversample factor to two or four.

27. If you have a frequency divider in your circuit, specify the divider ratio in the *Freqdivide Ratio* field.

28. For the *Time Step Control* field, select *fixed* or *adaptive* for Harmonic Balance and *adaptive* for Shooting.

29. For most ACPR measurements, select *conservative*.

30. Select the fast envelope level you desire. This is available only when harmonic balance is chosen for the engine. When the fast envelope mode is set to off (default), normal envelope is performed at the transistor level. Harmonic Balance or Shooting is used to solve for a subset of the RF cycles. Level 1 is available for circuits where acceleration is desired, and the circuit has little memory effect. Level2 can model slight to moderate memory effect. Both of these require defining the I and Q sources in the circuit, and for level2, the channel bandwidth needs to be specified.

31. Select whether you want to have all the harmonics on the output along with the time-domain waveform or not. The default in fast envelope is to not calculate the time-domain output waveform and higher harmonics in order to save time. If you take the default, only the harmonic number you specify in the *Output harmonic* field is calculated. If you want the all the harmonics and/or the time-domain waveform, select the *yes* check box to the right of *Output time-domain waveform and all harmonics*.

If you are using the port component with an I and Q modulation file, that port should be selected for both the I and Q source. If you are using the *ACPR_source* or *rfVsource*,

descend-read into the *ACPR_source* schematic. The I source is highlighted in red below, and the Q source is highlighted in blue below.

In rare cases, when fast envelope is used, questionable results are produced. There is an envelope option called *sweepmethod* that can improve the accuracy. *coarse* is the default, and *fine* produces a more accurate model at the expense of runtime at the beginning of the simulation. Finer detail is run in the harmonic balance analyses at the beginning of the simulation when *fine* is selected.

The mathematical model can be saved by setting the `writenv` option to a file name. After this, the model file can be read in and used directly by setting the `readenv` option. You cannot set both options at the same time.

Now run the simulation. When the simulation completes, the psd (frequency response) plots in the waveform window.

The unit for the psd is dBW (dB with respect to 1 Watt).

The ACPR is shown in ADE outputs section.

Plotting EVM

EVM is integrated into the Direct Plot Form.

1. In ADE, select *Results - Direct Plot - Main Form*.
2. Select *envlp* results.
3. Select *EVM*.
4. Select the first harmonic.
5. Select *OFDM*.
6. If you are using rfVsource, select the standard you are using in rfVsource.
7. Set the symbol start to 0 (zero) except for 802.11a, which is 4u.

8. Select the output net in the circuit.

The EVM for both the data channels and the control channels is displayed.

Plotting Main Channel Power

Plotting main channel power is a bit less straightforward than plotting EVM.

1. In ADE, select *Results - Direct Plot - Main Form*.
2. Select *envlp* results.
3. Select *Main Channel Power*.
4. Select the first harmonic.
5. Type the bandwidth of the main channel.

6. Open the envelope *Choosing Analyses* form.
7. Open the ACPR wizard.
8. In the ACPR wizard, triple-click the entry in the *Symbol Start* field.
9. In the *Direct Plot Form*, click the middle mouse button in the *From* field. The entry from the ACPR wizard should transfer.
10. In a similar manner, transfer the Strobe Period, Window Size, and Repetitions entries.

11. Select the same window function that was used in the ACPR wizard.

12. Select the output net in the schematic. The power calculation is displayed with a marker in the display tool.

Plotting the Envelope Voltage Versus Time

1. In the ADE, select *Results - Direct Plot - Main Form*.
2. Select *envlp* results.
3. Select *Voltage*.
4. Select harmonic time.
5. Select the appropriate modifier.

6. Select the first harmonic.

7. Select the net in the schematic.

8. Zoom in if you would like. The symbols are turned on in the display below so the strobe points are visible.

Plotting the Time-Domain Waveform

This is available in fast envelope *level1* and *level2* only when *yes* is selected for Output time-domain waveform and all harmonics.

1. In ADE, select *Results - Direct Plot - Main Form*.
2. Select *envlp* results.
3. Select *Voltage*.

4. Select *Time*.

5. Select the net in the schematic.

6. Zoom in, if desired.

The line that is visible between the spikes can be used to determine the phase of the signal as time increases only if the carrier frequency is an integer multiple of 1/delta-T in the I and Q modulation file. When the line is near the middle, the phase is zero or 180 degrees. When the line is near the top, the phase is 90 degrees. When the line is near the bottom, the phase is 270 degrees.

When you zoom in further, you can see the individual carrier cycles that envelope is calculating.

To see the full transient output, set the *transtrobeperiod* option to the period of the carrier divided by an integer. The choppiness of the waveform and the runtime is controlled by the

number of timepoints in the waveform that gets calculated. This is available in harmonic balance transistor-level simulation only. An example is shown below.

When zoomed in, the waveform is continuous.

Frequency Modulated Input Signals

Frequency modulated signals can be simulated in transistor-level envelope simulations only. When frequency modulation is used, the period of the waveform varies as a function of the modulation that is applied. Conventional envelope simulation cannot be used because the fundamental assumption is that the period is constant and defined by the carrier signal.

To allow frequency modulation, a new option has been added that allows the period to be solved at each carrier cycle that is simulated.

To simulate a frequency modulated signal, follow these steps:

1. Set up the frequency modulated source. In this example, a simple sinusoidal modulated port is used. If you use this element, remember that the modulation index is in percent.

2. Set up for a simulation using the envlp *Choosing Analyses* form and the ACPR wizard as shown before. Choose a standard with a bandwidth that is slightly larger than the FM modulation frequency. This will set the simulation to produce data at the time interval needed for the psd calculation. See the previous section for the steps that are needed.

3. Click *Options* in the envelope *Choosing Analyses* form.

4. Select the *Misc* tab.

5. Select the check box to the left of the number 1 for the *fmspeedup* option.
6. Type the modulation frequency in the *modulationbw* field.
7. Click *OK*.

8. In the envelope *Choosing Analyses* form, type in the center frequency in the *Fundamental Frequency* field.

9. Set harmonics and oversample as required. For more information, see the harmonic balance section at the beginning of Chapter 3.

10. Choose an accuracy level.

11. Set the *Fast envlp* mode to *off*.

12. Click *OK* and run the simulation.

13. When the simulation completes, the psd will plot.

14. If you want to see a spectral plot, set the line style to spectral. To do this, select the legend at the upper left. Right-click and select *Type* from the context menu. In the popup window, select *Spectral*. The line style will change to a more familiar plot.

15. When you zoom in to the individual spikes, you will see multiple spikes. There will be either one or two phantom spikes on each side of the output sideband because of the DFT sample window used in the psd function. The number of spikes and the relative

amplitude depend on the actual window function you selected. Remember that these spikes are not real. They are produced because of the window function itself.

Commonly Used Options

Options are provided to control the specific behavior of the simulation. This next section discusses the effect of the options in the options form for envelope. They are listed as they appear on the *Options* form.

Convergence Tab

ic (Shooting and Harmonic Balance)

Initial conditions can be specified graphically by selecting *Simulation - Convergence Aids - Initial Condition* in the ADE. Initial conditions can also be specified from a file using the *readic* property. Capacitors and inductors have the initial condition properties in the property list for the component. For capacitors, this is an initial voltage across the capacitor, and for inductors, it is an initial current in the inductor. The default is to observe all the initial conditions in the DC analysis that is used as the time-zero timepoint. The initial conditions force a voltage or current to be present in the time-zero solution. The initial conditions are released for the rest of the transient simulation in the tstab interval. The *ic* option controls the initial conditions that should be observed in the time-zero timepoint. *all* is the default. *dev* means that only the initial conditions on capacitors and inductors are observed. *node* means that only the initial conditions on a node are observed. *dc* means that no initial conditions are observed. The example below shows *node*.

Skipdc (Shooting and Harmonic Balance)

In some cases, the time-zero timepoint DC analysis does not converge. Instead of stopping the simulation, *skipdc* allows the envelope simulation to continue using an assumed solution for the time-zero timepoint. The default is *no* and a DC analysis is run to get the initial timepoint. *yes* means skip the DC solution, and proceed directly to the envelope simulation. All the nodes with initial conditions specified start at the initial condition value. Nodes with batteries start at the battery voltage. Nodes with no initial conditions start at zero volts. For `skipdc=yes`, the signal sources start as specified immediately in the envelope simulation. The example below shows the *skipdc* option set to *no*. The default is *yes*.

Readic (Shooting and Harmonic Balance)

This specifies an ascii file that contains two columns to be read as initial conditions. The left column is the node name. The right column is the voltage value. If the entry does not start with / (slash), the entry is located in the netlist directory. To find the netlist directory, select *Setup- Simulator/Directory/Host* in ADE. Look in the *Project Directory* field for the location of the simulation directory. Navigate to that directory and then to the `<Circuit Name>/spectre/<schematic or config>/netlist` directory. You can also click ( ) and browse to the directory. An example is shown below. There is no default.

Readns (Shooting and Harmonic Balance)

This specifies an ascii file with the same format as an ic file that is used as nodesets for the time-zero DC solution. Nodesets do not force a voltage to be held for the time-zero solution. Instead, they are a way of speeding up the time-zero calculation. As a suggestion, set the *write* option and the *readns* option to the same filename. The *write* option writes the time-zero solution to a file. When this is used as a starting point, many fewer iterations are needed for the time-zero point to converge. An example is shown below. There is no default.

Cmin (Shooting and Harmonic Balance)

cmin adds a capacitor to ground from all nodes in the circuit to ground with the value specified in the *cmin* field. Note that if 1 is specified, a 1 farad capacitor is added from every node to ground in the circuit. Do not forget the multiplier suffix. Values from 1f to 100f are typically used to allow convergence. An example with *cmin* set to 10f is shown below. The default is 0.

Maxiters (Shooting and Harmonic Balance)

The transient analysis for tstab (for both shooting and hb) and shooting interval iterate to a solution at each timepoint. *maxiters* specifies the maximum number of iterations before the timestep is cut for another try at convergence. In some cases, model parameters can cause discontinuities in the device current or capacitance. If this occurs, the change in the circuit condition can be large enough to require more than the five iterations that are the default. Specifying *maxiters* in the 40 to 100 range usually allows the simulator to converge inspite of the discontinuity. An example with *maxiters* set to 40 is shown below.

restart (Shooting and Harmonic Balance)

This option is used when a sweep is run. If restart is *no*, the time-zero timepoint does not restart from scratch. It restarts from the time-zero solution of the previous sweep value. *yes* forces a new time-zero solution to be re-run from scratch every time. The default is *no*.

envmaxiters (Shooting and Harmonic Balance)

The solution for each period of the carrier is an iterative solution. *envmaxiters* sets the maximum number of iterations that are allowed before the number of periods skipped is reduced. The default is three iterations for shooting and 40 for harmonic balance.

Accuracy Tab

method (tstab and shooting window for shooting, tstab only for Harmonic Balance)

This option controls the integration method for the shooting window. Note that in harmonic balance, the *method* option only applies in the tstab interval because the harmonic balance envelope solver works in the frequency domain. The default value for the shooting window is determined by the setting of *errpreset*. For *moderate* and *conservative*, the default is *gear2only*. For *liberal* it is *traponly*. Generally, *gear2only* is preferred because of the absence of trapezoidal ringing inherent in the *trap* (trapezoidal) method. Note that *liberal* is not recommended for RF simulation. When you simulate high Q oscillators, set *method* to *traponly*. This is needed because *gear2only* slightly numerically damps the oscillations.

envmethod (Shooting and Harmonic Balance)

This option controls the integration method for the part of the envelope analysis during the skipped cycles. In this part of the simulation, numerical integration still needs to be performed to get correct answers. For example, if there is a long time constant, the behavior on the node with the long time constant needs to be integrated even during the skipped cycles. *envmethod* controls this. The default is *gear2only* except for autonomous envelope (oscillators) where the numerical damping of *gear2only* is unacceptable. In this case, the default is *traponly*, which neither damps nor exaggerates the oscillations.

envmethod ☐ euler ☐ trap ☐ traponly
☐ gear2 ☐ gear2only ☑ trapgear2

relref (Shooting (tstab and shooting window) and Harmonic Balance (tstab only))

In the order of most to least accurate, the *relref* settings are *pointlocal*, *alllocal*, *sigglobal*, *allglobal*. The default for *moderate* is *sigglobal* and for *conservative* it is *alllocal. relref* is used in the transient analysis in the tstab interval for both shooting and harmonic balance and in the shooting window for shooting. *relref* is used in hb only for the tstab interval because the solver works in the frequency domain. In some cases, the timestep can collapse to near zero. The symptom is that in the Spectre output window, the percent done number remains the same for many reporting intervals. The solution to this is to set *relref* to *sigglobal*, which is slightly less accurate. Another possibility is to use the *Iteminstep* option as described in the *AdditionalParams* option of the *Misc* tab.

An example with *relref* set to *sigglobal* is shown below.

relref ☐ pointlocal ☐ alllocal ☑ sigglobal ☐ allglobal

Iteratio ((tstab and shooting window for shooting) and Harmonic Balance (tstab only))

Iteratio is a multiplier for the allowable numerical integration error in the tstab interval and the shooting window. The default is 3.5 or 10 depending on the setting of *reltol* and *errpreset*. See Controlling Accuracy on page 529 for details. *Iteratio* cannot be set smaller than 1.0 and is normally between 3.5 and 100. In some cases, the timestep collapses to near zero and setting *relref* to *sigglobal* does not fix the problem. In this case, there might be a model discontinuity. To test this, disable numerical integration timestep control by setting *Iteratio* to 1e9. Note that this is an extreme measure not to be used under normal simulations. If the timestep still collapses, there is a discontinuity in one of the models in the circuit. This could be a device model or a Verilog-A model. If it is necessary to set *Iteratio* very large to get the simulation to complete, you must also set *maxstep* small enough to preserve the accuracy

of the simulation. An example with *Iteratio* set to the smallest value of *1.0* is shown below. Note that *Iteminstep* might also be used to treat this condition. See *AddionalParams* in the *Misc* tab to see how to use this option.

envIteratio (Shooting and Harmonic Balance)

envIteratio is a multiplier for the allowable numerical integration error in the numerical integration that occurs in the part of the waveform where cycles are skipped. The default is 0.35 for conservative and 3.5 for moderate. Liberal is not recommended.

steadyratio (shooting Only)

The maximum mismatch between the beginning and ending values for the voltages and currents in the shooting window is determined by the *convergence criteria* * *Iteratio* * *steadyratio*. An example with *steadyratio* set to 1.0 is shown below. The default is 0.1 for moderate, and 1.0 for conservative.

Output Tab

annotate

Annotate (Shooting and Harmonic Balance)

This option controls the level of detail in the output log. The default is *status.* The level of detail increases as you move to the right.

Save (Shooting and Harmonic Balance)

The default is *allpub*. This saves all the public voltages at all levels of the hierarchy in the schematic. It excludes the internal nodes of the device models. *all* adds the internal nodes of all the devices. *lvl* saves all the nodes including the internal nodes of the devices through the level of hierarchy set in the *nestlvl* option. *lvlpub* is like *lvl*, but it does not save the internal nodes of the devices. *selected* saves only the nodes that are specifically saved. In ADE, this is accomplished using the *Outputs - To Be Saved - Select On Schematic* menu option

and then selecting the nodes and terminals specifically in the schematic. At the netlist level, this is accomplished by using a save statement with a list of names to be saved.

Nestlvl (Shooting and Harmonic Balance)

If *save* is set to *lvl* or *lvlpub*, this controls the maximum level of hierarchy to be saved. If *nestlvl* is *1*, only the top level is saved. If *nestlvl* is 2, the top level and the next level down are saved. The value for *nestlvl* can be any integer. The example below shows *lvlpub* selected that saves two levels of hierarchy. There is no default for *nestlvl*.

Compression (Shooting and Harmonic Balance)

Do not use this option.

outputtype (Shooting and Harmonic Balance)

This option is rarely used. It controls which simulation results to save. Normally, both time and frequency domain data is saved. Envelope saves the time-domain data only. Spectrum saves just the frequency-domain data. The default is to save both time and frequency-domain data. Note that in fast envelope, you need to select the *yes* check box to the right of *Output time-domain waveform and all harmonics* in order to calculate time-domain output.

strobeperiod (Shooting and Harmonic Balance)

This option specifies the envelope time in seconds to force cycles of the carrier to be simulated. Shooting will round this time to an integer number of periods of the carrier. This is used to force data at the times needed for the psd calculation. There is no default value.

transtrobeperiod (Shooting and Harmonic Balance)

Normally, the time-domain waveform is a cycle of carrier followed by many skipped cycles, followed by another cycle of simulation. Since there is no waveform between the last point for the first cycle and the first point of the second cycle, the waveform tool just connects those points together with a straight line. The waveform can be made into a continuous waveform using the *transtrobeperiod* option. This should be set to an integer fraction of one period of the carrier frequency. When this is set, Spectre still skips cycles, but it also creates an interpolated, continuous waveform with timepoints at the transtrobeperiod interval. In the example below, 20 points are set for one period of the carrier at 2.5G.

Spectre does take expressions, as shown above. There is no default value for *transtrobeperiod*.

Here is a comparison of the 2 waveforms.

The red waveform is the normal waveform. The vertical spikes in the red waveform are the individual periods of the carrier that are normally calculated. When *transtrobeperiod* is set, the waveform becomes continuous, as shown in blue. A closer view is shown below.

Reuse Tab

Write (Shooting and Harmonic Balance)

write specifies a filename to which the DC solution used as the first timepoint be written to. If the name does not start with slash (/), the file is written in the *netlist* directory. To determine the netlist directory location, select *Setup - Simulator/Directory/Host* in ADE. The *Project Directory* field lists the location of the simulation directory. From this directory, the netlist directory is in `<Circuit Name>`/`<simulator name>`/*<schematic or config>/netlist*. There is no default value.

Writefinal (Shooting Only)

writefinal writes the final envelope timepoint from the last period of the envelope analysis (not the final timepoint of the tstab interval) to the filename specified. See the description for *write* for the location. There is no default value.

Swapfile (Shooting Only)

This option is not currently implemented.

Saveclock (Shooting and Harmonic Balance)

This option is not currently implemented.

Saveperiod (Shooting and Harmonic Balance)

This option is not currently implemented.

Savetime (Shooting and Harmonic Balance)

This option is not currently implemented.

Savefile (Shooting and Harmonic Balance)

This option is not currently implemented.

Recover (Shooting and Harmonic Balance)

This option is not currently implemented.

Misc Tab

start (Shooting and Harmonic Balance)

This option is rarely used. It defines the starting time for the envelope simulation. The default is 0.

outputstart (Shooting and Harmonic Balance)

This option is rarely used. This defines the simulation time to begin to save data to the output file. The default is to start writing data at time 0.

tstab (Shooting and Harmonic Balance)

This is seldom used in envelope simulation. This specifies a simulation time for the transient analysis to run before starting the envelope simulation. The default is to run 3 periods of the carrier signal.

modulationbw (Shooting and Harmonic Balance)

modulationbw and *envmaxstep* accomplish the same thing, which is to define a maximum time interval to skip between carrier cycles that are simulated. When *modulationbw* is set, at least eight periods of the carrier are simulated in the time period of one cycle of the frequency *modulationbw*. *modulationbw* does not force data at equally spaced intervals, like *envstrobeperiod* does. Instead, it sets a a maximum delta-time for skipping cycles. There is no default.

maxstep (Shooting: tstab and waveform calculation. Harmonic balance: tstab interval only)

This option is rarely used. This sets the maximum delta-time for the shooting waveform in each cycle of the carrier. The default forces at least 20 timepoints for moderate, and 50 timepoints for conservative. It also is used in the transient analysis in the tstab interval.

envmaxstep (Shooting and Harmonic Balance)

envmaxstep and *modulationbw* accomplish the same thing, which is to define a maximum time interval to skip between carrier cycles that are simulated. *envmaxstep* sets the maximum time between periods of the carrier that can be skipped. *envmaxstep* does not

force datapoints at regularly spaced intervals. Instead, it sets a a maximum delta-time for skipping cycles. The default forces at least 25 periods to be simulated when *moderate* is selected, and 50 when *conservative* is selected.

pactimes (Shooting and Harmonic Balance)

During the transistor-level envelope simulation, one or more of the carrier periods that are simulated can be used as the nonlinear operating point for the small-signal pac, pxf, pnoise, or psp analyses. *pactimes* is a list of times separated by a space for the large-signal envelope solution that should be used for the small-signal analyses. The default is to not use any points for the small-signal analyses.

pacnames (Shooting and Harmonic Balance)

This option is used in combination with *pactimes*. Specify a list of the small-signal analyses pac, pxf, pnoise, and/or psp in this option field, and these analyses will be run at the times specified in *pactimes*. The default is to not run any small-signal analyses.

When this option has a list of small-signal analyses, the corresponding pac, pxf, pnoise, and psp analyses must also be set up and enabled.

fmspeedup (Shooting and Harmonic Balance)

This option allows the period of the carrier cycle to vary because of frequency modulation. When *fmspeedup* is set to the default of 0, the period of the carrier is not allowed to vary. When it is set to 1, the period is allowed to vary.

AdditionalParams (Shooting and Harmonic Balance)

This is a field for `<option_name>=value` statements. Multiple `<option_name>=value` statements are allowed with a space between them. Generally, this is used to unlock the beta features, however, there is one case where this field might be useful for an option that is not available in the GUI.

In some cases, in the tstab interval, the timestep collapses to near zero. The symptom is that the percent complete stays at the same value for many reporting intervals. In this case, there might be a discontinuity in one of the models.

In transient simulation, two things can reduce the timestep; having too much numerical integration error or taking too many iterations at a single timestep can cause the timestep to be cut. Spectre does not report what is causing the timestep to become small. One way to eliminate the possibility of numerical integration error (which is called local truncation error in the simulator) as the cause of reducing the timestep is to set *lteratio* to 1e9. The disadvantage of this is that it disables the normal method of timestep control, therefore, *maxstep* must be set to maintain accuracy.

Another way to accomplish this is to set *lteminstep* to a value of about the stop time divided by 1e5. This allows the normal method of timestep control, but if for some reason the numerical integration error wants to cut the timestep too small, the error is ignored. *lteminstep* is the way to specify this. To do this in the ADE, type `lteminstep=<Stop_Time>/1e5`. If you use `lteratio=1e9`, or if you set *lteminstep* and you still have very small timesteps, try increasing *maxiters* in the *Convergence* tab to between 40 and 100. When *lteratio* or *lteminstep* is set and you still have timestep problems, there is likely a discontinuity in either a device model or in a Verilog-A model. An example is shown below.

Autonomous ENVLP Analysis (Oscillators)

Autonomous envelope is useful for simulating the startup behavior of high Q oscillators, and for simulating the frequency pulling and/or pushing effects caused by random data being applied to a system. In autonomous envelope, the period and amplitude of each cycle of the oscillator that is simulated needs to be solved.

In general, harmonic balance is suggested for LC oscillators, and shooting should be used for ring oscillators. Crystal oscillators should use harmonic balance.

The only additional requirement in the *Choosing Analyses* form for oscillators is the requirement to select the check box for oscillators, and specify a node or pair of nodes that

has the oscillator signal on it. If the *osc initial condition* checkbox is set to *linear*, the nodes must be in the resonator of the oscillator. Although not a requirement, it is recommended that you use adaptive timestep control, and set the *modulationbw* (modulation bandwidth) option to the frequency of the oscillator divided by the Q of the oscillator.

Example: Startup behavior of a crystal oscillator

Here is an example circuit.

First, add an isource, set to *Source type* to *pulse*, and connect it to the resonator as highlighted above. The properties list is shown below.

Next, set the properties.

1. Set the *Zero value* to 0 (zero). The pulse starts at the zero value. With zero amps, the isource is a perfect open, so it does not load the circuit.

2. Set the *One value* to a current that starts the oscillator, but doesnot drive it too hard at the start. You will need to experiment with this value using the transient. In the waveform window below, the junction of the inductor and capacitor in the crystal motional equivalent circuit is plotted. This voltage will get quite large because of the high Q. Change the *One value* so that one to ten volts peak is generated at this node at the start of the transient simulation, and no huge transients are generated.

3. Set the *Rise time* and *Fall time* to about 1/10th to 1/100th of the period. This is a 10MHz crystal oscillator. In this case, 1/100th of the period was used.

4. Set the *Pulse width* so that half a period is generated by the pulse. Half the period of 10M is 50nsec, so the total of *Rise time*, *Fall time*, and *Pulse width* is 50nsec.

5. Do not make an entry in the *Period of waveform* property.

The effect of these entries is to set up a single current pulse to start the oscillator.

Now, set up the envelope *Choosing Analyses* form.

1. Select the appropriate engine. For LC and crystal oscillators, select *Harmonic Balance*. For ring oscillators select *Shooting*.

2. Set an appropriate *Stop Time* for your simulation. The time chosen will depend on the Q of the oscillator. About five times the Q periods need to be simulated.
3. Set the *Fundamental frequency*,*Number of Harmonics*, and *Oversample Factor* as needed for your simulation. See the discussion about setting harmonics and oversample factor in *Chapter 3: Harmonic Balance*.
4. Select*adaptive Time Step Control*. This allows the simulator to place the periods of oscillator simulation at the appropriate place.
5. Enable the check box next to the *Oscillator* Property, and specify a node inside the resonator for the oscillator node(s).
6. Set *Osc initial condition* to *default*. This ensures that the oscillator starts up as a result of the current pulse applied to the resonator.
7. Select *moderate* accuracy. In virtually every case, this is accurate enough.
8. *Fast envlp mode* must be set to *off*. Fast envelope is not available for oscillators.

Options are not required, but are suggested. Click *Options* at the bottom of the *Choosing Analyses* form.

Set the *modulationbw* (modulation bandwidth) property to the period divided by the Q of the oscillator. Expressions are allowed by Spectre.

Now run the simulation. Below is a comparison of the runtimes for the envelope analysis and for the transient analysis. Note that the runtime is much shorter for the envelope analysis.

```
/lan/csv/rfrd_v2/user/rdavis/rdavis/demos/ic61/docs_database/simulation/10M_Crystal_Oscillator_ipulse/spectre/scl
File  Help                                                                                                cadence
Current Frequency = 10010667.2032111026
EnvStepIndex = 516, envTime = 9.80662 ms, completed = (98.1 %)
Current Frequency = 10010667.1750947013
EnvStepIndex = 517, envTime = 9.8264 ms, completed = (98.3 %)
Current Frequency = 10010667.1481821686
EnvStepIndex = 518, envTime = 9.84617 ms, completed = (98.5 %)
Current Frequency = 10010667.1224628966
EnvStepIndex = 519, envTime = 9.86595 ms, completed = (98.7 %)
Current Frequency = 10010667.0979062133
EnvStepIndex = 520, envTime = 9.88573 ms, completed = (98.9 %)
Current Frequency = 10010667.0743909441
EnvStepIndex = 521, envTime = 9.90551 ms, completed = (99.1 %)
Current Frequency = 10010667.0519390963
EnvStepIndex = 522, envTime = 9.92529 ms, completed = (99.3 %)
Current Frequency = 10010667.0304751713
EnvStepIndex = 523, envTime = 9.94507 ms, completed = (99.5 %)
Current Frequency = 10010667.0099518057
EnvStepIndex = 524, envTime = 9.96485 ms, completed = (99.6 %)
Current Frequency = 10010666.9903005771
EnvStepIndex = 525, envTime = 9.98463 ms, completed = (99.8 %)
Current Frequency = 10010666.9716027323
Frequency = 1.001067e+07 (Hz)

Total clock cycles: 100109, skipped cycles: 99583, speed-up factor: 189
Done with the envelope-following analysis.
Clean up envelope following analysis.
Total time required for envlp analysis `envlp': CPU = 19.717 s, elapsed = 16.387 s.
Time accumulated: CPU = 19.917 s, elapsed = 18.6783 s.
Peak resident memory used = 47.1 Mbytes.
10  HelpAction
```

```
/lan/csv/rfrd_v2/user/rdavis/rdavis/demos/ic61/docs_database/simulation/10M_Crystal_Oscillator_ipulse/spectre/scl
File  Help                                                                                                cadence
    tran: time = 9.461 ms    (94.6 %), step = 67.72 ps     (677 n%)
    tran: time = 9.491 ms    (94.9 %), step = 67.79 ps     (678 n%)
    tran: time = 9.521 ms    (95.2 %), step = 167.8 ps    (1.68 u%)
    tran: time = 9.552 ms    (95.5 %), step = 334.2 ps    (3.34 u%)
    tran: time = 9.582 ms    (95.8 %), step = 664.1 ps    (6.64 u%)
    tran: time = 9.612 ms    (96.1 %), step = 13.8 ps      (138 n%)
    tran: time = 9.642 ms    (96.4 %), step = 18.85 ps     (189 n%)
    tran: time = 9.672 ms    (96.7 %), step = 848.8 ps    (8.49 u%)
    tran: time = 9.702 ms      (97 %), step = 877.8 ps    (8.78 u%)
    tran: time = 9.733 ms    (97.3 %), step = 11.78 ps     (118 n%)
    tran: time = 9.75 ms     (97.5 %), step = 742.7 ps    (7.43 u%)
    tran: time = 9.78 ms     (97.8 %), step = 145.8 ps    (1.46 u%)
    tran: time = 9.811 ms    (98.1 %), step = 7.884 ps    (78.8 n%)
    tran: time = 9.841 ms    (98.4 %), step = 60.96 ps     (610 n%)
    tran: time = 9.871 ms    (98.7 %), step = 110.5 ps    (1.11 u%)
    tran: time = 9.902 ms      (99 %), step = 25.34 ps     (253 n%)
    tran: time = 9.932 ms    (99.3 %), step = 722.8 ps    (7.23 u%)
    tran: time = 9.962 ms    (99.6 %), step = 881.9 ps    (8.82 u%)
    tran: time = 9.992 ms    (99.9 %), step = 544.3 ps    (5.44 u%)
Number of accepted tran steps =            22275386

Notice from spectre during transient analysis `tran'.
    Trapezoidal ringing is detected during tran analysis.
        Please use method=trap for better results and performance.

Initial condition solution time: CPU = 0 s, elapsed = 622.988 us.
Intrinsic tran analysis time:    CPU = 2.09802 ks, elapsed = 2.12532 ks.
Total time required for tran analysis `tran': CPU = 2.09802 ks (34m  58.0s), elapsed = 2.12536 ks (35m  25.4s).
Time accumulated: CPU = 2.11798 ks (35m  18.0s), elapsed = 2.14414 ks (35m  44.1s).
Peak resident memory used = 47.3 Mbytes.
10  HelpAction
```

Now use the *Direct Plot Form* to plot the waveform. In ADE, select *Results - Direct Pot - Main Form*.

1. Select *envlp* results.
2. Select *Voltage*.
3. Select *Net* or *Differential Nets*.
4. Select *time*.

5. Select a net in the schematic. The waveform is displayed.

The waveforms above are the envelope results on the left, and the transient results on the right. The results are almost identical for both, and envelope takes 16.4 seconds versus 2125.4 seconds for the transient on the same machine.

Common Problems With Oscillator Startup

In some cases, the oscillator will have a hard time starting up. You can recognize this by looking at the waveform in the resonator of the oscillator. Normally, the junction of the capacitor and the inductor in the crystal equivalent circuit will have voltages in the order of

KiloVolts peak at steady-state. When you see the ragged startup waveforms, as shown below, set a fixed timestep with about 15 to 100 periods as the step interval.

Spectre takes expressions in the entries, so it makes the period calculation easy.

The example above sets 20 periods of 10MHz as the step interval.

Also, set the following analysis options:

Set the *method* and *envmethod* options to *traponly*. *trap* does not damp or emphasize the oscillations of the oscillator. The default is *gear2only*, which slightly damps the oscillations. Also, set *relref* to *alllocal*. This is needed because of the mix of very large amplitude signals on the resonator with much smaller oscillations in the rest of the circuit.

Also, in some circuits, the initial pulse that is used to start the oscillator can cause large transients at the beginning of the envelope analysis and cause problems with the

determination of the frequency at the beginning of the simulation. To determine if this is the case, look at the startup waveform at the beginning of the simulation.

Note that the waveform is still settling at the beginning of the simulation. In this case, four periods are being simulated instead of one. You can see the frequency that is being used in the output log.

```
/lan/csv/rfrd_v2/user/rdavis/rdavis/demos/ic61/docs_database/simulation/10M_Crystal_Oscillator_ipulse/spectre/schematic/psf/s
File  Help                                                                                                                cādence
EnvStepIndex = 5, envTime = 2.75506 us, completed = (27.6 m%)
Current Frequency = 2506036.68218424218
EnvStepIndex = 6, envTime = 4.75211 us, completed = (47.5 m%)
Current Frequency = 2502134.45324389217
EnvStepIndex = 7, envTime = 6.75103 us, completed = (67.5 m%)
Current Frequency = 2500827.12081682542
EnvStepIndex = 8, envTime = 8.75053 us, completed = (87.5 m%)
Current Frequency = 2500493.17668261472
EnvStepIndex = 9, envTime = 10.7502 us, completed = (108 m%)
Current Frequency = 2500388.11727654515
EnvStepIndex = 10, envTime = 12.7499 us, completed = (128 m%)
Current Frequency = 2500350.27277679415
EnvStepIndex = 11, envTime = 14.7496 us, completed = (147 m%)
Current Frequency = 2500335.9484101506
EnvStepIndex = 12, envTime = 16.7494 us, completed = (167 m%)
Current Frequency = 2500330.40835776972
EnvStepIndex = 13, envTime = 18.7491 us, completed = (187 m%)
Current Frequency = 2500328.2433063793
EnvStepIndex = 14, envTime = 20.7488 us, completed = (207 m%)
Current Frequency = 2500327.38979573501
EnvStepIndex = 15, envTime = 22.7486 us, completed = (227 m%)
Current Frequency = 2500327.04948608158
EnvStepIndex = 16, envTime = 24.7483 us, completed = (247 m%)
Current Frequency = 2500326.905961012
EnvStepIndex = 17, envTime = 26.748 us, completed = (267 m%)
Current Frequency = 2500326.8438337245
EnvStepIndex = 18, envTime = 28.7478 us, completed = (287 m%)
Current Frequency = 2500326.81160016358
EnvStepIndex = 19, envTime = 30.7475 us, completed = (307 m%)
Current Frequency = 2500326.7915088064
EnvStepIndex = 20, envTime = 32.7473 us, completed = (327 m%)
Current Frequency = 2500326.77601764817
EnvStepIndex = 21, envTime = 34.747 us, completed = (347 m%)
Current Frequency = 2500326.7623735969
EnvStepIndex = 22, envTime = 36.7467 us, completed = (367 m%)
Current Frequency = 2500326.74946824415
EnvStepIndex = 23, envTime = 38.7465 us, completed = (387 m%)
Current Frequency = 2500326.73686284898
EnvStepIndex = 24, envTime = 40.7462 us, completed = (407 m%)
Current Frequency = 2500326.72438499564
EnvStepIndex = 25, envTime = 42.746 us, completed = (427 m%)
Current Frequency = 2500326.71196702309
EnvStepIndex = 26, envTime = 44.7457 us, completed = (447 m%)
Current Frequency = 2500326.6995823523
EnvStepIndex = 27, envTime = 46.7454 us, completed = (467 m%)
Current Frequency = 2500326.68722051196
EnvStepIndex = 28, envTime = 48.7452 us, completed = (487 m%)
Current Frequency = 2500326.67487733951
EnvStepIndex = 29, envTime = 50.7449 us, completed = (507 m%)
Current Frequency = 2500326.66255115345
EnvStepIndex = 30, envTime = 52.7446 us, completed = (527 m%)
Current Frequency = 2500326.65024124179
49  HelpAction
```

Setting *tstab* in the *Misc* tab can cause the simulation to start robustly.

The waveform is much smoother with tstab set.

7

Large Signal S-parameter Simulation (LSSP) Wizard

Large Signal S-Parameters

Characterizing RF circuits with small-signal S-parameters is a well established practice. However, small-signal S-parameters are not sufficient to describe both strongly nonlinear circuits and circuits that exhibit frequency translation. This is especially true for designs that use power amplifiers and mixers.

As a natural extension of small-signal S-parameters, large-signal S-parameters (LSSPs) are defined as the ratio of reflected (or transmitted) waves to incident waves. Since small-signal S-parameters are based on the simulation of a linearized circuit, they are independent of input power. This description illustrates how to measure LSSPs using Spectre and SpectreRF in the Analog Design Environment (ADE).

LSSPs are based on large-signal steady state simulation techniques such as SpectreRF's PSS analysis with both its time domain Shooting Newton (TD) engine and it's Harmonic Balance (HB) engine. LSSPs are sensitive to input power levels.

For multi-port circuits, you can represent

- The reflected wave, B_i at port i
- The incident wave, A_i at port i
- The frequency $k\omega$(harmonic k)

by

- The port voltage, V_i
- The current, I_i
- The termination impedance, R_i

The multi-port circuit representation is described as follows.

The reflected wave, B_i, at port i, and the frequency $k\omega$ (harmonic k) are characterized by Equation 7-1.

(7-1) $$B_i(k\omega) = \frac{V_i(k\omega) + R_i I_i(k\omega)}{2\sqrt{R_i}}$$

The incident wave, A_i, at port i, and the frequency $k\omega$ (harmonic k) are characterized by Equation 7-2.

(7-2) $$A_i(k\omega) = \frac{V_i(k\omega) - R_i I_i(k\omega)}{2\sqrt{R_i}}$$

The LSSPs between port i with harmonic m and port j with harmonic n, are defined as in Equation 7-3.

(7-3) $$S_{ij}(m\omega, n\omega) = \frac{B_i(m\omega)}{A_j(n\omega)}$$

Where $A_i(k\omega) = 0$, for all ports other than j and for harmonics other than n.

Large-Signal S-parameters for a Two-Port Circuit

Consider calculating LSSPs for a two-port circuit. When you apply a source at port 1 and terminate port 2, as shown in Figure 7-1, you can measure S_{11} and S_{12} at a given input power P_1.

Figure 7-1 Test Setup to Measure the Large-Signal S-Parameters S11 and S21

Similarly, when you apply a source at port 2 and terminate port 1, as shown in the Figure 7-2, you can measure two more LSSPs, S_{12} and S_{22}.

Figure 7-2 Test Setup to Measure the Large-Signal S-Parameters S12 and S22

To be consistent with the input power level measured for S_{11} and S_{21}, you can compute the power applied to port 2 in this stage using the relationship in Equation 7-4.

(7-4) $$P_2 = |S_{21}|^2 P_1$$

Measuring Large Signal S-parameters using SpectreRF

Following the test bench setup methods shown in Figure 7-1 and Figure 7-2, you can measure the large-signal S-parameters in terms of input power with a swept PSS analysis in SpectreRF.

The following example explains the procedures using the circuit shown in Figure 7-3, which is the *EF_example_LSSP* circuit from the RFworkshop library. Since this is an amplifier, you will only measure the large-signal S-parameters for the fundamental frequency. In principle, you might use a similar method to measure the parameters across different frequencies.

The steps below indicate only a few steps for setting up Large Signal S-parameters. See the LSSP Appnote along with the example database present in the installation for more information. You can access the application note from the following location:

$MMSIMHOME/tools/spectre/examples/SpectreRF_workshop/SpectreRF_AN

Figure 7-3 Power Amplifier for measuring Large-Signal S-parameters

Simulation 1: Calculating LSSP in Terms of Input Power

1. Open the schematic view of the *EF_example_LSSP design* from the library.

Note:

- LSSP analysis requires ports to be used for the input and output, just like SP simulation.
- In this schematic view, port is used as the source for both input and output. The PORT1 and PORT2 ports are set to normal sine port with variable frequency and amplitude.
- Frequency and Amplitude for both the ports are set to variable because input and output frequencies, and the power need to be swept in this case.
- LSSP analysis requires that both the ports in the circuit must have large-signal definitions in the port. This is because the LSSP analysis does not work if the large-signal definition is provided only for one port.

2. Load the state *sweep_power* in workshop.

Note: Alternatively, you can directly load the saved state from the *View* list box.

LSSP Analysis Setup for sweep_power state

ADE L Window

Choosing Analyses form (top portion) for HB Analysis

Choosing Analyses form (bottom portion) for HB Analysis

Choosing Analyses form (top portion) for PSS (Shooting) Analysis

Choosing Analyses form (bottom portion) for PSS (Shooting) Analysis

Note: You only need to setup either of the HB or PSS (Shooting) analysis for running LSSP analysis.

Steps

1. Select *Tools* - *RF* - *LSSP* from the top menu

   The Large Signal S-parameter Wizard form for `sweep_power` state is displayed.

   The input and output port should be specified. The Sweep setting is only for input port, the output port is adjusted by Spectre after input sweep is done.

2. Run the simulation.

   When the SImulation completes, the results are plotted as below:

Simulation 2: Calculating LSSP in Terms of Beat Frequency

With the results of Simulation 1, you can select a fit input/output power. This power can be used to sweep beat frequency values for the PSS.

1. Set the following:

   `pin=-10, pout=13.66. pout=|S21|*|S21|*p1`

   or

   `pout(dbm)=p1(dbm)+db20(mag(S21)).`

   You can read out these two values from previous simulation's result.

Tip

Setting the number of steps in power sweep to 2 will help you find matching output power easily.If you know the match input and output power, you can do this directly.

LSSP Analysis Setup for sweep_freq state

ADE L Window

Choosing Analyses form (top portion) for HB Analysis

Choosing Analyses form (bottom portion) for HB Analysis

Choosing Analyses form (top portion) for PSS (Shooting) Analysis

Choosing Analyses form (bottom portion) for PSS (Shooting) Analysis

Note: You only need to setup either of the HB or PSS (Shooting) analysis for running LSSP analysis.

Steps

1. Select *Tools* - *RF* - *LSSP* from the top menu

   The Large Signal S-parameter Wizard form for `sweep_freq` state is displayed.

2. Set The Large Signal Parameter Wizard sweep frequency.
3. Run the simulation.

When the SImulation completes, the results are plotted as below:

References

[1] V. Rizzoli, A. Lipparini, F. Mastri, “Computation of Large-Signal S-parameters by Harmonic-Balance Techniques,” in *Electronics Letters*, Vol. 24, Issue 6, pp. 329-330, 17 Mar 1988.

[2] Jan Verspecht, Frans Verbeyst, Marc Vanden Bossche, “Network Analysis Beyond S-parameters: Characterizing and Modeling Component Behavior under Modulated Large-Signal Operating Conditions,” in *56th ARFTG Conference Proceedings*, Broomfield, Colorado, USA, December 2000.

8

Simulating Noise and Jitter

Analyzing Time-varying Noise

RF circuits are usually driven by periodic inputs. The noise in RF circuits is generated by sources that can therefore typically be modeled as periodically time-varying. Noise that has periodically time-varying properties is said to be cyclostationary.

Characterizing Time-Domain Noise

Noise in a circuit that is periodically driven, say with period T, exhibits statistical properties that also vary periodically.To understand time-domain characterization of noise, consider the simple circuit shown in Figure 8-1.

Figure 8-1 Very Simple Mixer Schematic

The amplitude of the noise measured at the RF output shown in Figure 8-1 periodically varies depending on the magnitude of the modulating signal $p(t)$, as shown by the sample points in Figure 8-2.

Figure 8-2 Time-Varying Noise Process Analyzed at ξ_1 and ξ_2

In Figure 8-2

- The solid line shows the envelope *p(t)* that modulates the noise process.
- The circles show possible phase points on the envelope where you might calculate the time-varying noise power.
- The circles marked ξ_1 and ξ_2 indicate the two phase points on the envelope where time-varying noise power is calculated.
- Noise in circuits that are periodically driven, say with period *T*, exhibits statistical properties that also vary periodically. To understand time-domain characterization of noise, consider the simple circuit shown in Figure 8-1 on page 1305. The amplitude of the noise measured at the RF output periodically varies depending on the magnitude of the modulating signal *p(t)*, as shown by the sample points (or circles on the signal envelope) in Figure 8-2 on page 1306.

- Figure 8-2 is a representation of periodically-modulated noise. It shows noise processes for two different phases in the periodic interval. Each process is stationary.

Virtuoso® Spectre® circuit simulator RF analysis (SpectreRF) can calculate the time-varying noise power at any point in the fundamental period. In fact, SpectreRF can calculate the full auto correlation function

$$R^{\xi}(p,q) = < x^{\xi}(p)x^{\xi}(p+q) > = R^{\xi}(q)$$

and its spectrum for the discrete-time processes x^{ξ} obtained by periodically sampling the time-domain noise process at the same point in phase.

Figures 8-3 and 8-4 show two such noise processes for two different phases in the periodic interval. Each process is stationary. Figure 8-3 shows the noise process for the phase marked ξ_1 in Figure 8-2 on page 1306.

Figure 8-3 Noise Process for Phase ξ_1

Figure 8-4 shows the noise process for the phase marked ξ_2 in Figure 8-2 on page 1306.

Figure 8-4 Noise Process for Phase ξ_2

See the "Reference Information on Time-Varying Noise" on page 1320 for a more detailed introduction to noise in periodically time-varying systems.

Calculating Time Domain Noise

The following steps tell you how to calculate time-domain noise using SpectreRF.

1. In a terminal window, type `icms` (in IC5141) and virtuoso (in IC 6.1.4) to start the environment.
2. In the ADE window, select *Analyses – Choose.*

   The Choosing Analyses form appears.
3. In the Choosing Analyses form, highlight pss and perform the PSS analysis setup.

4. In the Choosing Analyses form, highlight *pnoise.*
5. The Choosing Analyses form changes to let you specify information for a Pnoise analysis.
6. In the Choosing Analyses form, perform the following:
   a. Choose Noise Type *timedomain*.
   b. Specify an appropriate frequency range and sweep for the analysis.

      You might, for example, perform a linear sweep up to the fundamental frequency. Because each time point in the calculation is a separate frequency sweep, use the minimum number of frequency points possible to resolve the spectrum. This step minimizes computation time.
   c. Specify a *noiseskipcount* value or specify additional explicit time points with *noisetimepoints.*
   d. Specify an appropriate set of time points for the time-domain noise analysis.
   e. Use *noiseskipcount* to calculate time-domain noise for one of every *noiseskipcount* time points.

      If you set *noiseskipcount* to a value greater than or equal to zero, the simulator uses the *noiseskipcount* parameter value and ignores any *numberofpoints* parameter value. When *noiseskipcount* is less than zero, the simulator ignores the *noiseskipcount* parameter. The default is *noiseskipcount = -1.*

      You can add specific points by specifying a time relative to the start of the PSS simulation interval. *noiseskipcount = 5* performs noise calculations for about 30 time points in the PSS interval.

      If you only need a few time points, add them explicitly with the *noisetimepoints* parameter and set *noiseskipcount* to a large value like 1000.
7. In the ADE window, choose *Simulation – Netlist and Run.*
8. The simulation runs.
9. In the ADE window, choose *Results – Direct Plot – PSS*.

   The PSS Results form appears.
10. To calculate time-varying noise power, perform the following steps in the PSS Results form:
    a. Click *tdnoise* and then select *Integrated noise power*.

b. Type `0` as the start frequency and the PSS fundamental frequency as the stop period.

For example, type `1G` if the PSS period is 1ns.

A periodic waveform appears that represents the expected noise power at each point in the fundamental period.

11. To display the spectrum of the sampled processes, perform the following steps in the PSS Results form:

a. Highlight *Output Noise*.

b. Highlight *Spectrum* for the type of sweep.

c. Clicking on *Plot*.

A set of curves appears, one for each sample phase in the fundamental period.

12. To calculate the autocorrelation function for one of the sampled processes, perform the following steps:

a. Display the spectrum using instructions from step 11.

b. In the ADE window, choose *Tools – Calculator*.

c. The calculator appears.

d. Click *wave* in the calculator and the select the appropriate frequency-domain spectrum.

One of the sample waveforms is brought into the calculator

e. Choose *DFT* from the list of special functions in the calculator. Then set 0 as the *From* and the PSS fundamental as the *To* value.

f. Choose an appropriate window (e.g., Cosine2) and number of samples (around the number of frequency points in the interval [0,1/T]),

g. Apply the *DFT* and plot the results.

Harmonic *q* of the DFT results gives the value of the discrete autocorrelation for this sample phase, *R(q)*.

Be sure the noise is in the correct units of power (e.g., V^2/Hz), not V/square root of Hz) before performing the DFT to obtain the autocorrelation.

Calculating Noise Correlation Coefficients

To characterize the noise in multi-input/multi-output systems, it is necessary to calculate both the noise power at each port and the correlation between the noise at various ports. The situation is complicated in RF systems because the ports may be at different frequencies. For example, in a mixer, the input port may be at the RF frequency and the output port at the IF frequency.

Denote the power spectrum of a signal x by $S_{XX}(\omega)$, that is

$$S_{XX}(\omega) = X^*(\omega)\, X(\omega)$$

where $X(\omega)$ is the Fourier transform of the signal $x(t)$. For random signals like noise, calculate the expected value of the power spectrum $S_{XX}(\omega)$. To characterize the relationship between two separate signals $x(t)$ and $y(t)$, you also need the cross-power spectrum

$$S_{XY}(\omega) = X^*(\omega)\, Y(\omega)$$

For random signals, the degree to which x and y are related is given by the cross-power spectrum. You can define a correlation coefficient $\rho xy(\omega)$ by

$$\rho XY(\omega) = \frac{S_{XY}(\omega)}{\sqrt{S_{XX}(\omega) S_{YY}(\omega)}}$$

- A correlation coefficient ($\rho xy(\omega)$) of 0 indicates the signals are completely uncorrelated.
- A correlation coefficient of 1 indicates the signals are perfectly correlated. For example, a signal is always perfectly correlated with itself.
- You might also want to consider correlations between noise at different frequencies. The following quantity

$$S^{\alpha}_{XY}(\omega)$$

expresses the correlation of a signal x at frequency ω with the signal y at frequency $\omega + \alpha$. For example, white Gaussian noise is completely uncorrelated with itself for $a \neq 0$. Noise in an RF system generally has $S^{\alpha}(\omega)$ non-zero when α is the fundamental frequency, for example, the LO frequency in a mixer.

After you have measured the noise properties of a circuit, you can represent the circuit as a noiseless multiport with equivalent noise sources. For example, in Figure 8-5, first you

measure the noise voltage appearing at the excitation ports of the circuit on the left in Figure 8-5. Then, you can express the noise properties of the circuit as two equivalent frequency-dependent noise voltages V_{N1} and V_{N2}, and a complex correlation coefficient $\rho 12$.

Figure 8-5 Calculating Noise Correlations and Equivalent Noise Parameters

When you know the noise at each port and its correlation, you can obtain any of various sets of equivalent noise parameters. For example, you can express noise in an impedance representation as the equivalent correlated noise voltage sources shown in Figure 8-5, as equivalent noise resistances and the correlation parameters, and as F_{min}, R_N, G_{opt}, and B_{opt}.

Cyclostationary Noise Example

As an example which illustrates the various aspects of cyclostationary noise, consider the simple mixer circuit shown in Figure 8-6.

Figure 8-6 Simple Mixer Circuit

In this simple mixer circuit, white Gaussian noise passes through a high-order band-pass filter with center frequency ω_0. Then it is multiplied by two square-waves which have a phase shift a with respect to each other. Finally the output of the ideal multipliers is put through a one-pole low-pass filter to produce I and Q outputs.

The time-domain behavior of the noise is examined first. The most dramatic effect can be seen by looking directly at the mixer outputs in Figure 8-7 on page 1313. This figure shows the contributions to the time-varying noise power made by three separate source frequencies. Two of the source frequencies were selected around ω_0, the third source frequency was selected away from ω_0, slightly into the stop band of the band-pass filter. The sharp change in noise power over the simulation interval occurs because the mixers were driven with square-wave LO signals.

Figure 8-7 Time-Varying Noise Power Before Low-Pass Filter

Figure 8-8 shows the spectrum of a sampled noise process. Note the periodically replicated spectrum.

Figure 8-8 Spectrum of a Sampled Noise Process

The noise behavior at the output ports is examined next. The output spectra at the I and Q outputs are shown in Figures 8-9 and 8-10. The noise density at I is concentrated around zero

because the noise at the RF input to the mixers (band-limited around ω_0) is shifted down to zero and up to $2\omega_0$, but components not around zero are eliminated by the low-pass filter.

Figure 8-9 Power Spectra With LO Tones 90^{deg} Out of Phase

More interesting is the cross-correlation spectrum of the I and Q outputs, shown as the dashed line in Figures 8-9 and 8-10. When the signals applied to the mixers are 90 degrees out of phase (as in Figures 8-9), the cross-power spectral density of the noise at the separate I and Q outputs is small, indicating little noise correlation. If the tones are not quite out of phase (as in Figures 8-10), the correlation is much more pronounced, though in neither case is it completely zero.

In Figures 8-9 and 8-10, the solid and dashed lines represent the following

- The solid line represents the power spectrum for the I output with the function

$$S_{II}(\omega)^0$$

- The dashed line represents the cross-spectral density for I and Q with the function

$$S_{IQ}^{0}(\omega)$$

Figure 8-10 Power Spectra With LO Tones 72^{deg} Out of Phase

A more interesting example comes from examining the correlation between the noise at the I output and the noise at the RF input. The density function as given by

$$S_{IR}^{(1)}(\omega)$$

is significant because it represents the correlation between the noise at the I output around the baseband frequency with the noise at the RF input, ω_0 higher in frequency. The correlation

is high because the noise at the RF input is centered around ω_0 and converted to zero-centered noise by the mixer.

Figure 8-11 Noise Spectrum at the RF Input

In Figure 8-11, the noise spectrum at the RF input is given by the following function

$$S_{RR}^{0}(\omega)$$

Figure 8-12 Noise Spectrum at Various Power Densities

In Figure 8-12, the solid, dashed, and dashed-dot lines represent the following

- The solid line represents the cross power spectrum which indicates correlation between output noise power at the I output versus noise at the RF input that is one harmonic higher in frequency. This is represented by the following function

$$S_{IR}^{(1)}(\omega)$$

- The dashed line represents the noise spectrum at the I output with the following function

$$S_{II}^{(0)}(\omega)$$

- The dashed-dot line represents the noise spectrum at the RF input with the following function

$$S_{RR}^{(0)}$$

Finally a detailed circuit example was considered. A transistor-level image-reject receiver with I and Q outputs was analyzed. The noise spectra at the I and Q outputs were found to be very similar, as shown in Figure 8-13.

Figure 8-13 Power Spectral Densities of an Image-Reject Receiver

In the image-reject receiver example shown in Figure 8-13, the power spectral densities are represented as follows

- I output is a solid line
- Q output is a dashed line
- IQ cross-power density is a dash-dot line

The IQ cross-power density was smaller, but not negligible, indicating that the noise at the two outputs is partially correlated. The correlation coefficient between noise at the I and Q outputs of the image-reject receiver is shown in Figure **8-14**.

Figure 8-14 Correlation Coefficient for Output Noise of Image-Reject Receiver

Reference Information on Time-Varying Noise

The following sections provide background and reference information on the following noise-related topics:

- Thermal Noise
- Linear Systems and Noise
- Time-Varying Systems and the Autocorrelation Function
- Time-Varying Systems and Frequency Correlations
- Time-Varying Noise Power and Sampled Systems

Thermal Noise

The term *noise* is commonly used to refer to any unwanted signal. In the context of analog circuit simulation, noise is distinguished from such phenomena as distortion in the sense that it is non-deterministic, being generated from *random* events at a microscopic scale. For example, suppose a time-dependent current *i(t)* is driven through a linear resistor, as shown in Figure 8-15.

Figure 8-15 Deterministic Current Source Driving a Noisy Linear Resistor

The voltage that appears across the resistor is

$$v(t) = i(t)R + n(t)$$

The desired signal $i(t)R$, shown in Figure8-16, is corrupted by an added noise voltage $n(t)$ that is due to resistive thermal noise. The thermal noise of the resistor is modelled by a current source in parallel with the resistor.

Figure 8-16 The Desired Signal i(t)R

The total measured voltage is shown in Figure 8-17.

Figure 8-17 The Total Measured Voltage

The added noise process alone, $n(t)$, is a random process and so it must be characterized in ways that are different than for deterministic signals. That is, at a time t_0 the voltage produced by the driven current can be exactly specified—it is $i_0 sin t_0 R$. Just by inspecting Figure 8-16 we can predict this part of the measured signal.

On the other hand, the exact value of the noise signal cannot be predicted in advance, although it can be measured to be a particular value $n(t_0)$. However, if another measurement is performed, the noise signal $n(t)$ we obtain is different and Figure 8-17 changes. Due to its innate randomness, we must use a statistical means to characterize $n(t)$.

Now consider the circuit in Figure 8-18, where we restrict attention to the noise source/ resistor pair alone.

Figure 8-18 Resistor Modeled as a Noiseless Resistance with an Equivalent Noise Current Source

A typical measured noise current/voltage is shown in Figure 8-19.

Figure 8-19 Typical Measured Noise Current/Voltage

Because we cannot predict the specific value of $n(t)$ at any point, we might instead try to predict what its value would be on average, or what we might *expect* the noise to be. For example, if we measure many noise voltage curves in the time domain, $n(t)$, and average over

many different curves, we obtain an approximation to the expected value of $n(t)$ which we denote by $E\{n(t)\}$. For thermal noise, we find that $E\{n(t)\} = 0$. Therefore, instead of computing $E\{n(t)\}$, let us instead compute $E\{n(t)^2\}$, the expected noise power. An example of this sort of measurement is shown in Figure 8-20. 250 measurements were needed to compute this curve.

Figure 8-20 Expected Noise Power

Now suppose that we wish to tap the circuit at multiple points. Each point has its own noise characteristics, but they are not necessarily independent. Consider the circuit shown in Figure 8-21.

Figure 8-21 Circuit Illustrating Correlated Noise

The signals $n_1(t)$ and $n_2(t)$ are obtained by measuring the voltage across a single resistor ($n_1(t)$), and across both resistors ($n_2(t)$), respectively. Just measuring $E\{n_1(t)^2\}$ and $E\{n_2(t)^2\}$ is not enough to predict the behavior of this system, because $n_1(t)$ and $n_2(t)$ are not independent.

To see $n_1(t)$ and $n_2(t)$ are not independent, consider Figures 8-22 and 8-23. Samples of each of the processes are taken and plotted on an X-Y graph.

Figure 8-22 Samples of $n_1(t)$ Plotted Versus $n_2(t)$

Because $n_1(t)$ composes part of $n_2(t)$, $n_1(t)$ and $n_2(t)$ are correlated so in Figure 8-22, the X-Y plot has a characteristic skew along the $X=Y$ line, relative to the $n_1(t)$, $n_3(t)$ plot in Figure 8-23,

Figure 8-23 Samples of $n_1(t)$ Plotted Versus $n_3(t)$

The signals $n_1(t)$ and $n_3(t)$ are uncorrelated because they represent thermal noise from different sources. The additional measurement needed to describe the random processes is the measurement of the correlation between the two processes, $E\{n_1(t)n_2(t)\}$. We can also define a time-varying correlation coefficient ρ, with $\rho \in [0,1]$, as

$$\rho(t) = \frac{E\{n_1(t)n_2(t)\}}{\sqrt{E\left\{n_1(t)^2\right\}E\left\{n_2(t)^2\right\}}}$$

A value of $\rho=0$ indicated completely uncorrelated signals, and a value near one indicates a high degree of correlation. In this example we would find that $\rho(t) = 1/2$, representing the fact that each of the two noise sources contributes half of the process $n_2(t)$.

When there are multiple variables of interest in the system, it is convenient to use matrix notation. We write all the random processes of interest in a vector, for example

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

and then we can write the correlations as the expected value of a vector outer product, $E\{x(t)x^H(t)\}$, where the H superscript indicates Hermitian transpose.

For example, we might write a time-varying correlation matrix as

$$R_{xx}(t,\ t) \equiv E\{x(t)x^H(t)\} = \begin{bmatrix} E\{x_1(t)x_1(t)\} & E\{x_1(t)x_2(t)\} \\ E\{x_2(t)x_1(t)\} & E\{x_2(t)x_2(t)\} \end{bmatrix}$$

Linear Systems and Noise

The examples in the preceding sections describe how to characterize purely static systems. Now we need to add some elements with memory, such as inductors and capacitors.

As a first example, consider adding a capacitor in parallel to the simple resistor, as shown in Figure 8-24.

Figure 8-24 A Simple RC Circuit

A sample of the noise process in shown in Figure 8-25.

Figure 8-25 Noise Process for a Simple RC Circuit

The noise looks different than the noise of the resistor alone, because the low-pass filter action of the RC circuit eliminates very high frequencies in the noise. However, we cannot see this effect simply by measuring $E\{n(t)^2\}$ as shown in Figure 8-26.

Figure 8-26 Expected Noise Power for an RC Circuit

The measurement of $E\{n(t)^2\}$ is independent of time for an RC circuit, just as it was for the for the resistor circuit.

Spectral Densities in Two Simple Circuits

Instead of expected noise power, let us look at the expected power density in the frequency domain. Let $n(\omega)$ denote the Fourier transform of one sample of $n(t)$. Then, $E\{n(\omega)n(\omega)^*\}$ is the expected power spectral density, which we denote by $S_n(\omega)$.

In the present case, the capacitor has a pronounced effect on the spectral density. Figure 8-26 shows a computed power spectral density for the resistor thermal noise previously

considered. The spectrum is essentially flat (some deviations occur because a finite number of samples was taken to perform the calculation). The flat spectrum represents the fact that in the resistor's noise, all frequencies are, in some statistical sense, equally present. We call such a process *white noise*.

Power Spectral Density for Resistor Thermal Noise

Figure 8-27 shows the spectrum of the noise process after filtering by the resistor-capacitor system.

Figure 8-27 Resistor-Capacitor Filtered Spectral Noise Process

It is easy to rigorously account for the effect of the RC-filter on the power spectrum of the noise signal. Suppose a random signal x is passed through a time-invariant linear filter with frequency-domain transfer function $h(\omega)$. Then the output is $y(\omega)=h(\omega)x(\omega)$.

Because expectation is a linear operator, we can easily relate the power spectral density of y, $S_y(\omega)$ to $S_x(\omega)$, the power spectral density of x, by using the definitions of y and power density. Specifically,

$$S_y(\omega) = E\{y(\omega)y(\omega)^*\} = E\{h(\omega)x(\omega)x(\omega)^* h(\omega)^*\} = |h(\omega)|^2 S_x(\omega)$$

The noise from the resistor can be considered to be generated by a noise current source i, with power density

$$S_i(\omega) = \frac{4k_B T}{R}$$

placed in parallel with the resistor. With the capacitor in parallel, the transfer function from the current source to the resistor voltage is just the impedance $Z(\omega)$,

$$h(\omega) = Z(\omega) = \frac{(1/C)}{j\omega + \frac{1}{RC}}$$

and so the noise voltage power density is

$$S_n(\omega) = \frac{\frac{4k_B T}{RC}}{\omega^2 + \left(\frac{1}{RC}\right)^2}$$

Clearly the spectrum is attenuated at high frequencies and reaches a maximum near zero.

For a vector process, we may define a matrix of power-spectral densities,

$$S_{xx}(\omega) \equiv E\left\{x(\omega)x^H(\omega)\right\}$$

The diagonal terms are simple real-valued power densities, and the off-diagonal terms are generally complex-valued cross-power densities between two variables. The cross-power density gives a measure of the correlation between the noise in two separate signals at a specific frequency. We may define a correlation coefficient as

$$\rho_{ij}(\omega) \equiv \frac{S_{x_i x_j}(\omega)}{[S_{x_i}(\omega) S_{x_j}(\omega)]^{1/2}}$$

It is often more useful to examine the correlation coefficient because the cross-power density may be small. As an example, consider a noiseless amplifier. The noise at the input is simply a scaled version of the noise at the output leading to a $\rho=1$, but the cross-power density is much smaller than the output total noise power density if the amplifier has small gain.

In a numerical simulation it is important to compute *only* the correlation coefficient when the diagonal spectral densities are sufficiently large. If one of the power densities in the denominator of the correlation-coefficient definition is very small, then a small numerical error could lead to large errors in the computed coefficient, because of division by a number close to zero.

In the vector case, the transfer function is also a matrix $H(\omega)$, such that $y(\omega)=H(\omega)x(\omega)$ and so the spectral densities at the input and output are related by

$$S_{yy}(\omega) = E\left\{ H(\omega) x(\omega) x^H(\omega) H^H(\omega) \right\} = H(\omega) S_{xx}(\omega) H^H(\omega)$$

Time-Varying Systems and the Autocorrelation Function

If all the sources of noise in a system are resistors, and the circuit consists strictly of linear time-invariant elements, then the matrix of spectral densities $S_{xx}(\omega)$ is sufficient to describe the noise. However, most interesting RF circuits contain nonlinear elements driven by time-varying signals. This introduces time-varying noise sources as well as time-varying filtering. Because most noise sources are small, and generate small perturbations to the circuit behavior, for purposed of noise analysis, most RF circuits can be effectively modeled as linear time-varying systems. The simple matrix of power spectra is not sufficient to describe these systems.

To see this, return to the simple resistor example. Suppose that a switch is connected between the resistor and the voltage measuring device, as shown in Figure 8-28.

Figure 8-28 SimpleTime-Varying Circuit with Switch

Further suppose that the switch is periodically opened and closed. When the switch is open, there is no noise measured. When the switch is closed, the thermal noise is seen at the voltage output. A typical noise waveform is shown on the bottom left in Figure 8-29.

Figure 8-29 Typical Waveforms for Noise, Noise Power and White Noise

The time-varying noise power $E\{n(t)^2\}$ can be computed and is shown in Figure 8-29 on the top left, above the time-varying noise waveform. The expected power periodically switches between zero and the value expected from the resistor noise. This is different than the resistor-only and resistor-capacitor systems considered previously. Indeed, no linear time-invariant system could create this behavior. However, if we examine the power spectrum on the right in Figure 8-29, we again find that it is flat, corresponding to *white* noise.

The Autocorrelation Function

At this point it is clear that $E\{n(t)\}$ and $E\{n(t)^2\}$ do not completely specify the random process $n(t)$, nor does the power spectral density. To obtain a complete characterization, consider measuring $n(t)$ at two different timepoints, t_1 and t_2. $n(t_1)$ and $n(t_2)$ are two separate random variables. They may be independent of each other, but in general they have some correlation. Therefore, to completely specify the statistical characteristics of $n(t_1)$ and $n(t_2)$ together, we must specify not only the variances $E\{n(t_1)^2\}$ and $E\{n(t_2)^2\}$, but also the covariance $E\{n(t_1)n(t_2)\}$. In fact because $n(t)$ has infinite dimension, an infinite number of these correlations must be specified to characterize the entire random process. The usual way of doing this is by defining the autocorrelation function $R_n(t,t+\tau) = E\{n(t)n(t+\tau)\}$.

If $x(t)$ is a vector process,

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

then we define the autocorrelation matrix as

$$R_{xx}(t,\ t+\tau) \equiv E\left\{x(t)x^H(t+\tau)\right\} = \begin{bmatrix} E\{x_1(t)x_1(t+\tau)\} & E\{x_1(t)x_2(t+\tau)\} \\ E\{x_2(t)x_1(t+\tau)\} & E\{x_2(t)x_2(t+\tau)\} \end{bmatrix}$$

where superscript H indicates Hermitian transpose.

The diagonal term gives the autocorrelation function for a single entry of the vector, e.g, $E\{x_1(t)x_t(t+\tau)\}$. For $\tau=0$, this is the time-varying power in the single process, e.g. $E\{x_1(t)^2\}$. If the process $x(t)$ is Gaussian, it is completely characterized by its autocorrelation function $R_x(t, t+\tau)$ because all the variances and co-variances are now specified.

We can also precisely define what it means for a process to be *time-independent*, or *stationary*—A stationary process is one whose autocorrelation function is a function of τ

only, not of t. This means that not only is the *noise power* $E\{n(t)^2\}$ independent of t, but the correlation of the signal at a time point with the signal at another timepoint is only dependent on the difference between the timepoints, τ. The white noise generated by the resistor, and the RC-filtered noise, are both stationary processes.

Connecting Autocorrelation and Spectral Densities

At different points in the discussion above it was claimed that the expected time-varying power $E\{n(t)^2\}$ of the resistor voltage is constant in time, and also the power density $S_n(\omega)$ is constant in frequency. At first this seems odd because a quantity that is *broad* in time should be *concentrated* in frequency, and vice versa.

The answer comes in the precise relation of the spectral density to the autocorrelation function. Indeed, it turns out that the spectral density is the Fourier transform of the autocorrelation function, but with respect to the variable τ, not with respect to t. In other words, the measured spectral density is related to the correlation of a random process with time-shifted versions of itself. Formally, for a stationary process $R_n(t,t+\tau) = R_n(t))$ we write

$$S_n(f) = \int_{-\infty}^{\infty} e^{i\omega\tau} R_n(\tau) d\tau$$

For example, in the resistor-capacitor system considered above, we can calculate the autocorrelation function $R_n(\tau)$ by an inverse Fourier transform of the power spectral density, with the result

$$R_n(\tau) = \left(\frac{4k_B T}{C}\right) e^{-|\tau|/(RC)}$$

From inspecting this expression we can see that what is happening is that adding a capacitor to the system creates memory. The random current process generated by the thermal noise of the resistor has no memory of itself so the currents at separate time-instants are not correlated. However, if the current source adds a small amount of charge to the capacitor, the charge takes a finite amount of time to discharge through the resistor creating voltage. Thus voltage at a time-instant is correlated with the voltage at some time later, because part of the voltage at the two separated time instants is due to the same bit of added charge. From inspecting the autocorrelation function it is clear that the correlation effects last only as long as the time it takes any particular bit of charge to decay, in other words, a few times the RC time constant of the resistor-capacitor system.

Note that the process is still stationary because this memory effect depends only on how long has elapsed since the bit of charge has been added, or rather how much time the bit of charge has had to dissipate, not the absolute time at which the charge is added. Charge added at separate times is not correlated because arbitrary independent amounts can be added at a given instant. In particular, the time-varying noise power,

$$E\left\{n(t)^2\right\} = \int_{-\infty}^{\infty} S_n(\omega)d\omega$$

Time-Varying Systems and Frequency Correlations

Now we have seen that the variation of the spectrum in frequency is related to the correlations of the process, in time. We might logically expect that, conversely, variation of the process in time (that is, non-stationarity) might have something to due with correlations of the process in frequency. To see why this might be the case, suppose we could write a random process x as a sum of complex exponentials with random coefficients,

$$x = \sum_{k=-K}^{K} c_k e^{i\omega t}$$

Noting that $c_{-k}=c_k^*$, the time-varying power in the process is

$$E\left\{x^2(t)\right\} = \sum_{k=-K}^{K} \sum_{l=-K}^{K} E\{c_k c^*{}_l\} e^{i(w_k - w_l)t}$$

and it is clear that $E\{x(t)^2\}$ is constant in time if and only if

$$E\{c_k c^*{}_L\} = \left|c_k\right|^2 \varsigma_{kl}$$

In other words, the coefficients of expansion of sinusoids of different frequencies must be uncorrelated. In general, a stationary process is one whose frequency-domain representation contains no correlations across different frequencies.

To see how frequency correlations might come about, let us return to the resistor-switch example. Let $n(t)$ denote the voltage noise on the resistor, and $h(t)$ the action of the switch, so

that the measure voltage is given by $v(t) = h(t)n(t)$, where $h(t)$ is periodic with period T and frequency

$$\omega_0 = \frac{2\pi}{T}$$

The time-domain multiplication of the switch becomes a convolution in the frequency domain,

$$v(\omega) = h(\omega) \otimes n(\omega)$$

where $\otimes$ denotes convolution.

Because $h(t)$ is periodic, its frequency-domain representation is a series of Dirac deltas,

$$h(\omega) = \sum_k h_k \varsigma(\omega - k\omega_0)$$

and so

$$v(\omega) = \sum_k h_k n(\omega - k\omega_0)$$

and the spectral power density is simply

$$S_v(\omega) = E\{\langle v(\omega)v(\omega)^* \rangle\} = \sum_k \sum_l h_k h_l^* E\{n(\omega - k\omega_0)n(\omega - l\omega_0)^*\}$$

Because the process n is stationary, this reduces to

$$S_v(\omega) = \sum_k |h_k|^2 S_n(\omega - k\omega_0)$$

Because $S_n(\omega)$ is constant in frequency, $S_v(\omega)$ is also.

However, the process v is no longer stationary because frequencies separated by multiples of ω_0 have been correlated by the action of the time-varying switch. We may see this effect in

the time-variation of the noise power, as in Figure 8-29 on page 1332, or we may examine the correlations directly in the frequency domain.

To do this, we introduce the *cycle spectra*

$$S_{xx}^{\alpha}(\omega)$$

that are defined by

$$S_{xx}^{\alpha}(\omega) = E\left\{x(\omega)x^{H}(\omega+\alpha)\right\}$$

and are a sort of cross-spectral density, taken between two separate frequencies. $S_0(\omega)$ is just the power spectral density we have previously discussed. In fact we can define a frequency-correlation coefficient as

$$\rho_{n}^{\alpha}(\omega) \equiv \frac{S_{n}(\omega)^{\alpha}}{\sqrt{S_{n}(\omega)S_{n}(\omega+\alpha)}}$$

and if

$$\rho_{n}^{\alpha}(\omega) = 1$$

then the process n has frequency content at ω and $\omega+\alpha$ that is perfectly correlated.

Consider separating out a single frequency component of a random process and multiplying by a sinusoidal waveform of frequency α, as shown in Figure 8-30. The component at ω is shifted to re-appear at $\omega+\alpha$ and $\omega-\alpha$. The new process' frequency components at $\omega-\alpha$ and $\omega+\alpha$ are deterministically related to the components of the old process located at ω Therefore, they are correlated, and $S^{2a}(\omega)$ is non-zero.

Figure 8-30 Time-Variation Introduces Frequency Correlation

Physically, what happens is that to form a waveform with a defined *shape* in time, the different frequency components of the signal must add in a coherent, or correlated fashion. In a process like thermal noise, the Fourier coefficients at different frequencies have phase that is randomly distributed with respect to each other, and the Fourier components can only add incoherently. Their powers add, rather than their amplitudes. Frequency correlation and time-variation of statistics are thus seen to be equivalent concepts.

Another way of viewing the cycle spectra is that they represent, in a sense, the two-dimensional Fourier transform of the autocorrelation function, and are therefore just another way of expressing the statistics of the process.

Time-Varying Noise Power and Sampled Systems

Again supposing the signal n to be cyclostationary with period T, for each sample phase $\xi \in [0,T)$, we may define the discrete-time autocorrelation function

$$R_n^{\xi}(p,\ q)$$

to be

$$R_n^{\xi}(p,\ p+q) = R_n(\xi + pT,\ \xi + (p+q)T)$$

Because the cyclostationary process R_n is periodic, by inspection

$$R_n^{\xi}(p,\ p+q)$$

is independent of p and thus stationary, that is

$$R_n^{\xi}(p,\ p+q) = R_n^{\xi}(q)$$

Note that

$$R_n^{\xi}(p,\ p) = R^{\xi}(0)$$

gives the expected noise power, $R_n(\xi,\xi)$, for the signal at phase ξ. Plotting $R^{\xi}(0)$ versus ξ shows how the noise power varies periodically with time.

The discrete-time process

$$R_n^{\xi}(p,\ p+q) = R_n^{\xi}(q)$$

can be described in the frequency-domain by its discrete Fourier transform,

$$R_n^{\xi}(\phi) = \sum_{q=-\infty}^{\infty} R^{\xi}(q) e^{iq2\pi\phi T}$$

Note that the spectrum of the discrete (sampled) process

$$R_n^{\xi}(\phi)$$

is periodic in frequency with period $1/T$.

All noise power is aliased into the Nyquist interval $[-1/2T, 1/2T]$ (or, equivalently, the interval $[0,1/T]$). Generally it is the noise spectrum which is available from the circuit simulator. To obtain the autocorrelation function or time-varying noise power, an inverse Fourier integral must be calculated by

$$R_n^{\xi}(q) = \int_0^{1/T} R_n^{\xi}(\phi) e^{iq2\pi\phi} d\phi$$

Summary

- All useful noise metrics can be interpreted in terms of correlations. Physically these can be interpreted as the expected value of two-term products. In the case of random vectors these are expected values of vector outer products.
- The power spectral density of a variable indexed i is

  $$S_{x_i x_i}(\omega) = E\{x_i(\omega) x_i(\omega)^*\}$$

  This is what the current SpectreRF pnoise analysis computes.

  $S_{xx}(\omega)$ is constant if and only if x is a white noise process. In that case $R_{xx}(\tau) = R\delta(\tau)$ if there are no correlations in time for the process.

  The cross-power densities of two variables x_i and x_j are

  $$S_{x_i x_j}(\omega) = E\left\{x_i(\omega) x_j(\omega)^H\right\}$$

  If and only if the two variables have zero correlation at that frequency, then

  $$S_{x_i x_j} = 0$$

A correlation coefficient may be defined as

$$\rho_{ij}(\omega) \equiv \frac{S_{x_i x_j}(\omega)}{\sqrt{S_{x_i}(\omega) S_{x_j}(\omega)}}$$

and $\rho_{ij}(f) \in [0,1]$.

The cycle-spectra

$$S_{xx}^{\alpha}(f)$$

represent correlations between frequencies separated by the cycle-frequency α

$$S_{xx}^{\alpha}(f) = E\left\{ x(\omega) x^{H}(\omega + \alpha) \right\}$$

For a single process x_i, a correlation coefficient may be defined as

$$\rho_{x_i}^{\alpha}(\omega) \equiv \frac{S_{x_i x_i}(\omega)^{\alpha}}{\sqrt{S_{x_i}(\omega) S_{x_i}(\omega + \alpha)}}$$

and

$$\rho_{x_i}^{\alpha}(f) \in [0,\ 1]$$

- A process is stationary if and only if

$$S_{xx}^{\alpha}(\omega) = 0$$

for all ω and all $\alpha \neq 0$, that is, if there are no correlations in frequency for the process.

In other words,

$$S_{xx}^{\alpha}(\omega) = S_{xx}(\omega)\delta(\alpha)$$

- A process is cyclostationary if

$$S_{xx}^{\alpha} = 0$$

for all $\alpha \neq m\omega_0$ for some ω_0 and integer m. Frequencies separated by $m\omega_0$ are correlated. A stationary process passed through a periodically linear-time varying filter in general is cyclostationary with ω_0 the fundamental harmonic of the filter.

We might also compute correlations between different nodes at different frequencies, with the obvious interpretation and generalization of the correlation coefficients.

Oscillator Noise Analysis

In RF systems, local oscillator phase noise can limit the final system performance. Virtuoso® Spectre® circuit simulator RF analysis (SpectreRF) lets you rigorously characterize the noise performance of oscillator elements. This appendix explains phase noise, tells how it occurs, and shows how to calculate phase noise using the SpectreRF simulator.

"Phase Noise Primer" on page 1343 discusses how phase noise occurs and provides a simple illustrative example.

"Models for Phase Noise" on page 1346 contains mathematical details about how the SpectreRF simulator calculates noise and how these calculations are related to other possible phase noise models. You can skip this section without any loss of continuity, but this section can help you better understand how SpectreRF calculates phase noise and better appreciate the drawbacks and pitfalls of other simple phase noise models. This section can also help in debugging difficult circuit simulations.

"Calculating Phase Noise" on page 1357 provides some suggestions for successful and efficient analysis of oscillators and discusses the limitations of the simulator.

"Troubleshooting Phase Noise Calculations" on page 1359 explains troubleshooting methods for difficult simulations.

"Frequently Asked Questions" on page 1364 answers some commonly asked questions about phase noise and the SpectreRF simulator.

"Further Reading" on page 1369 and "References" on page 1369 list additional sources of information on oscillator noise analysis.

The procedures included in this appendix are intended for SpectreRF users who analyze oscillator noise. You must have a working familiarity with SpectreRF simulation and its operating principles. In particular, you must understand the SpectreRF PSS and Pnoise analyses. For information, see *Virtuoso Spectre Circuit Simulator RF Analysis Theory.*

Phase Noise Primer

Consider the simple resonant circuit with a feedback amplifier shown in Figure 8-31, a parallel LC circuit with nonlinear transconductance. At small capacitor voltages, the transconductance is negative, and the amplifier is an active device that creates positive feedback to increase the voltage on the capacitor. At larger voltages, where the transconductance term goes into compression, the amplifier effectively acts as a positive resistor (with negative feedback) and limits the capacitor voltage.

Figure 8-31 A Simple Resonant Oscillator

A simple model for the nonlinear transconductance is a cubic polynomial. We hypothesize a nonlinear resistor with a current-voltage relation given by

$$i(v) = -\left(\frac{v}{R}\right)(1 - \alpha v^2)$$

The effect of the resistor in parallel with the inductor and the capacitor can be lumped into this transconductance term. The parameter is a measure of the strength of the nonlinearity in the transconductance relative to the linear part of the total transconductance. Because the signal amplitude grows until the nonlinearity becomes significant, the value of this parameter does not affect the qualitative operation of the circuit.

For simplicity, for the remainder of this appendix

$$\alpha = 1/3$$

After some renormalization of variables, where time is scaled by

$$1/\omega_0$$

with

$$\omega_0 = \frac{1}{\sqrt{LC}}$$

and current is scaled by

$$\sqrt{C/L}$$

You can write the differential equations describing the oscillator in the following form

$$\frac{dv}{dt} = -i + \frac{1}{Q}(1 - \alpha v^2)v + \xi(t)$$

and

$$\frac{di}{dt} = v$$

In these equations, v and i represent the normalized capacitor voltage and inductor current, respectively, and $\xi(t)$ is a small-signal excitation such as white Gaussian noise, $Q = R/\omega_0 L$ is the quality factor of an RLC circuit made by replacing the nonlinear transconductance by a positive resistance R.

The equations just discussed describe the familiar Van der Pol oscillator system. This model includes many of the qualitative aspects of oscillator dynamics, yet it is simple enough to analyze in detail. Many more complicated oscillators that operate in a weakly nonlinear mode can be approximated with this model by using the first few terms in the Taylor series expansion of the relevant transconductances.

As a brute-force method of calculating the noise properties of this circuit, the nonlinear stochastic differential equations that describe the current and voltage processes were numerically integrated [1], and the noise power was obtained using a standard FFT-periodiagram technique. This technique requires several hundred simulations of the oscillator over many thousands of periods. Consequently, it is not a feasible approach for practical circuits, but it is rigorously correct in its statistical description even though it requires no knowledge of the properties of oscillators, noise, periodicity, or signal amplitudes. Figure 8-32 shows the total time-averaged noise in the voltage variable.

Figure 8-32 Noise in a Simple Van der Pol System

By plotting *Power Spectral Density* against *Normalized Noise Frequency Offset* for a Q = 5 system, Figure 8-32 shows noise in a simple Van der Pol system.

The left half of Figure 8-32 shows noise as a function of absolute frequency.

The right half of Figure 8-32 shows noise as a function of frequency offset from the oscillator fundamental frequency.

The dashed line is LC-filtered white noise, the dash-dot line is RLC-filtered white noise, the solid line is SpectreRF phase noise, and (x) marks are noise power from a full nonlinear stochastic differential equation solution.

The resulting noise power spectral density looks much like the voltage versus current response of a parallel LC circuit. The oscillator in steady-state, however, does not look like an

LC circuit. As you see in the following paragraphs, this noise characteristic similarity occurs because both systems have an infinite number of steady-state solutions.

The characteristic shape of the small-signal response of an LC circuit results because an excitation at the precise resonant frequency can introduce a drift in the amplitude or phase of the oscillation. The magnitude of this drift grows with time and is potentially unbounded. In the frequency domain, this drift appears as a pole on the imaginary axis at the resonant frequency. The response is unbounded because no restoring force acts to return the amplitude or phase of the oscillation to any previous value, and perturbations can therefore accumulate indefinitely.

Similarly, phase noise exists in a nonlinear oscillator because an autonomous oscillator has no time reference. A solution to the oscillator equations that is shifted in time is still a solution. Noise can induce a time shift in the solution, and this time shift looks like a phase change in the signal (hence the term *phase noise*). Because there is no *resistance* to change in phase, applying a constant white noise source to the signal causes the phase to become increasingly uncertain relative to the original phase. In the frequency domain, this corresponds to the increase of the noise power around the fundamental frequency.

If the noise perturbs the signal in a direction that does not correspond to a time shift, the nonlinear transconductance works to put the oscillator back on the original trajectory. This is similar to AM noise. The signal uncertainty created by the amplitude noise remains bounded and small because of the action of the nonlinear amplifier that created the oscillation. The LC circuit operates differently. It lacks both a time (or phase) reference and an amplitude reference and therefore can exhibit large AM noise.

Another explanation of the similarity between the oscillator and the LC circuit is that both are linear systems that have poles on the imaginary axis at the fundamental frequency, ω_0. That is, at the complex frequencies $s = i\omega_0$. However, the associated transfer functions are not the same. In fact, because of the time-varying nature of the oscillator circuit, multiple transfer functions must be considered in the linear time-varying analysis.

Understanding the qualitative behavior of linear and nonlinear oscillators is the first step towards a complete understanding of oscillator noise behavior. Further understanding requires more quantitative comparisons that are presented in Models for Phase Noise. If you are not interested in these mathematical details, you might skip ahead to “Calculating Phase Noise” on page 1357.

Models for Phase Noise

This section considers several possible models for noise in oscillators. In the engineering literature, the most widespread model for phase noise is the Leeson model [2]. This heuristic model is based on qualitative arguments about the nature of noise processes in oscillators. It

shares some properties with the LC circuit models presented in the previous section. These models fit well with an intuitive understanding of oscillators as resonant RLC circuits with a feedback amplifier. In the simplest treatment, the amplifier is considered to be a negative conductance whose value is chosen to cancel any positive real impedance in the resonant tank circuit. The resulting linear time-invariant noise model is easy to analyze.

Linear Time-Invariant (LTI) Models

To calculate the noise in a parallel RLC configuration, the noise of the resistor is modeled as a parallel current source of power density

$$S(\omega) = \frac{4k_B T}{R}$$

Where k_B is Boltzman's constant. In general, if current noise excites a linear time-invariant system, then the noise power density produced in a voltage variable is given by [3] as follows

$$S_v(\omega) = |H(\omega)|^2 S_i(\omega)$$

where $H(\omega)$ is the transfer function of the LTI transformation from the noise current source *input* to the voltage *output*. The transfer function is defined in the standard way to be

$$H(\omega) = \frac{v_0(\omega)}{i_s(\omega)}$$

where i_s is a (deterministic) current source and v_0 is the measured voltage between the nodes of interest.

It follows that the noise power spectral density of the capacitor voltage in the RLC circuit is, at noise frequency $\omega = \omega_0 + \omega'$ with $\omega' << \omega_0$.

$$S_v(\omega') = \frac{4k_B TR}{1 + 4(\omega' / \omega_0)^2 Q^2}$$

where the quality factor of the circuit is

$$Q = \frac{R}{\omega_0 L}$$

The parallel resistance is R (the source of the thermal noise), and ω_0 is the resonant frequency.

If a noiseless negative conductance is added to precisely cancel the resistor loss, the noise power for small ω' / ω_0 becomes

$$S_v(\omega') = \frac{k_B T R}{(\omega' / \omega_0)^2 Q^2}$$

This linear time-invariant viewpoint explains some qualitative aspects of phase noise, especially the $(\omega_0 / Q\omega')^2$ dependencies. However, even for this simple system, a set of complicating arguments is needed to extract approximately correct noise from the LTI model. In particular, we must explain the 3 dB of excess amplitude noise inside the resonant bandwidth generated by an LC model but not by an oscillator (see “Amplitude Noise and Phase Noise in the Linear Model” on page 1352). Furthermore, many oscillators, such as relaxation and ring oscillators, do not naturally fit this linear time-invariant model. Most oscillators are better described as time-varying (LTV) circuits because many phenomena, such as upconversion of $1/f$ noise, can only be explained by time-varying models.

Linear Time-Varying (LTV) Models

For linear time-invariant systems, the noise at a frequency ω is directly due to noise sources at that frequency. The relative amplitudes of the noise at the system outputs and the source noise are given by the transfer functions from noise sources to the observation point. Time-varying systems exhibit frequency conversion, however, and each harmonic $k\omega_0$ in the oscillation can transfer noise from a frequency $\omega \pm k\omega_0$ to the observation frequency ω In general, for a stationary noise source $\xi(t)$, the total observed noise voltage is [3]

$$S_v(\omega) = \sum_k |H_k(\omega)|^2 S_\xi(\omega + k\omega_0)$$

Each term in the series represents conversion of current power density at frequency $\omega + k\omega_0$ to voltage power density at frequency ω with gain $|H_k(\omega)|^2$. As an example, return again to the Van der Pol oscillator with $\alpha = 1/3$ and notice how a simple time-varying linear analysis of noise proceeds.

The first analysis step for the Van der Pol oscillator is to obtain a large-signal solution, so you set $\xi(t) = 0$. In the large-Q limit, the oscillation is nearly sinusoidal and so it is a good approximation to assume the following

$$v(t) = a \sin \omega_0 t$$

The amplitude, a, and oscillation frequency can be determined from the differential equations that describe the oscillator. Recognizing that

$$i(t) = \left(\frac{a}{\omega_0}\right)\cos(\omega_0 t)$$

and substituting into the equation for dv/dt, a and ω_0 are determined by the following

$$a\omega_0 \cos(\omega_0 t) - \left(\frac{1}{Q}\right)\left(a \sin(\omega_0 t) - \frac{a^3}{3}\sin^3(\omega_0 t)\right) - \left(\frac{a}{\omega_0}\right)\cos(\omega_0 t) = 0$$

Substituting

$$\sin^3(\omega_0 t) = \frac{3\sin(\omega_0 t) - \sin(3\omega_0 t)}{4}$$

and using the orthogonality of the sine and cosine functions, it follows that

$$a - \left(\frac{a^3}{4}\right) = 0$$

and

$$\omega_0 - \left(\frac{1}{\omega_0}\right) = 0$$

(The $\sin(3\omega_0 t)$ term is relevant only when we consider higher-order harmonics of the oscillation.) Therefore, to the lowest order of approximation, $a = 2$ and $\omega_0 = 1$.

The only nonlinear term in the Van der Pol equations is the current-voltage term, $v^3/3$. This term differentiates the Van der Pol oscillator from the LC circuit. The small-signal conductance is the derivative with respect to voltage of the nonlinear current

$$\frac{-(1-v^2)}{Q}$$

With

$$v(t) = 2\sin t$$

the small-signal conductance as a function of time is

$$(1/\ Q)(1-\cos 2t)$$

Because there is a nonzero, time-varying, small-signal conductance, the Periodic Time Varying Linear (PTVL) model is different from the LTI LC circuit model. In fact, the time-average conductance is not even zero. However, the time-average power dissipated by the nonlinear current source is zero, a necessary condition for stable, sustained oscillation.

Oscillators are intrinsically time-varying elements because they trade off excessive gain during the low-amplitude part of the cycle with compressive effects during the remainder of the cycle. This effect is therefore a generic property not unique to this example.

To complete the noise analysis, write the differential equations that the small-signal solution $i_{s(t)}$, $v_{s(t)}$ must satisfy,

$$\frac{dv_s}{dt} = -i_s + \frac{1}{Q}(1-3\alpha v^2(t))v_s + \xi(t)$$

and

$$\frac{di_s}{dt} = v_s$$

From the large signal analysis, $v(t) = 2\sin t$, and so

$$\frac{dv_s}{dt} = -i_s + \frac{1}{Q}(2\cos 2t - 1)v_s + \xi(t)$$

and

$$\frac{di_s}{dt} = v_s$$

The time-varying conductance can mix voltages from a frequency ωto ω- 2. For small ώ, if an excitation is applied at a frequency $\omega = 1 + \omega'$, i_s and v_s are expected to have components at $1 + \omega'$ and $-1 + \omega'$ for the equations to balance. (Higher-order terms are again presumed to be small.) Writing

$$i_{s(t)} = i_{+} e^{(i(1+\omega')t)} + i_{-} e^{(i(-1+\omega')t)}$$

and substituting into the small-signal equations with

$$\xi(t) = c_{+} e^{i(1+\omega')t}$$

leads to the following system of equations for i_+ and i_-

$$\begin{bmatrix} 1-(1+\omega')^2 + \left(\frac{i}{Q}\right)(1+\omega') & -\left(\frac{i}{Q}\right)(-1+\omega') \\ -\left(\frac{i}{Q}\right)(1+\omega') & 1-(-1+\omega')^2 + \left(\frac{i}{Q}\right)(-1+\omega') \end{bmatrix} \begin{bmatrix} i_{+} \\ i_{-} \end{bmatrix} = \begin{bmatrix} c_{+} \\ 0 \end{bmatrix}$$

Solving these equations gives the transfer function from an excitation at frequency $1 + \omega'$ to the small-signal at frequency $1 + \omega'$ that we call $H_0(\omega')$. A similar analysis gives the other significant transfer function, from noise at frequency $-1 + \omega'$ of amplitude C_ to the small-signal response at frequency $1 + \omega'$, that we call $H_{-2}(\omega')$. In the present case, for small ώ,

$$H_0^2 \cong H_{-2}^2 \cong \frac{R^2}{16Q^2\left(\frac{\omega'}{\omega_0}\right)^2}$$

For a general Van der Pol circuit with a parallel resistor R that generates white current noise, $\xi(t)$, with $S_\xi(\omega) = 4\, k_B\, T/R$,

$$S_v(\omega') = \frac{k_B TR}{2\left(\frac{\omega'}{\omega}\right)^2 Q^2}$$

Note that this is precisely one-half the noise predicted by the LC model.

You can gain additional insight about phase noise by analyzing the time-domain small-signal response. The small-signal current response is.

$$i_s(t) = \frac{ie^{i\omega' t}(c_- + c_+)}{2\omega'} \sin t$$

Notice that c_+ and c_- are complex random variables that represent the relative contribution of white noise at separate frequencies. As white noise has no frequency correlations, they have uncorrelated random phase, and thus zero amplitude expectation, and unit variance in amplitude. Because the large-signal current is $i(t) = 2\cos t$, and the sine and cosine functions are orthogonal, the total noise for small ω' that we computed is essentially all phase noise.

Amplitude Noise and Phase Noise in the Linear Model

Occasional claims are made that in oscillators, “Half the noise is phase noise and half the noise is amplitude noise.” However, as the simple time-varying analysis in the previous section shows, in a physical oscillator the noise process is mostly phase noise for frequencies near the fundamental. It is true that in an LC-circuit half the total noise power corresponds to AM-like modulation and the other half to phase modulation. In the literature, the AM part of the noise is sometimes disregarded when quoting the oscillator noise although this is not always the case. (The SpectreRF simulator computes the total noise generated by the circuit; see “Details of the SpectreRF Calculation” on page 1353).

However, a *linear* oscillator does not really exist. Physical oscillators operate with a tradeoff of gain that causes growing signal strength and nonlinear compressive effects that act to limit the signal amplitude. For noise calculation, the oscillator cannot be considered a linear time-invariant system because there are intrinsic nonlinear effects that produce large phase noise but limited amplitude noise. Oscillators are time-varying, and they therefore require a time-varying small-signal analysis.

Arguments that start with stationary white noise and pass it through a linear model in a forward-analysis fashion produce incorrect answers. This is true because they neglect the time-variation of the conductances (and possibly the capacitances) in the circuit. In the simple cases considered here, the conductances vary in time in a special way so as to produce no amplitude noise, only phase noise.

They have that special variation because they result from linearization about an oscillator limit cycle. An oscillator in a limit cycle has a large response to phase perturbations, but not to amplitude perturbations. The amplitude perturbations are limited by the properties of the nonlinear amplifier, but the phase perturbations can persist. The SpectreRF simulator calculates the correct phase noise because it *knows* about the oscillator properties.

Similarly, arguments [13] that start with noise power and derive phase noise in a backwards fashion also usually produce incorrect results because they cannot correctly account for frequency correlations in the noise of the oscillator. These frequency correlations are introduced by the time-varying nature of the circuit.

Occasionally, a netlist appears in which a negative resistance precisely cancels a positive resistance to create a pure LC circuit. Because such a circuit has an infinite number of oscillation modes, the SpectreRF simulator cannot correctly calculate the noise because it assumes a unique oscillation. Such a circuit is not physically realizable because adding or subtracting a microscopically small amount of conductance makes the circuit either go into nonlinear operation (amplifier saturation) or become a damped LC circuit that has a unique final equilibrium point. This equilibrium point is the zero-state solution. Trying to create the negative resistance oscillator is like trying to bias a circuit on a metastable point. Any amplitude oscillation can exist, depending on the initial conditions, as long as the amplitude is less than the amplifier saturation point.

Details of the SpectreRF Calculation

This section contains the mathematical details of how the SpectreRF simulator computes noise in oscillators. Understanding the material in this section can help you troubleshoot and understand difficult oscillator problems.

The analysis the SpectreRF simulator performs is similar to the simple analysis in the section “Linear Time-Varying (LTV) Models” on page 1348. During analysis, the SpectreRF simulator

1. First finds the periodic steady state of the oscillator using the PSS analysis.
2. Then linearizes around this trajectory.
3. The resulting time-varying linear system is used to calculate the noise power density. The primary difference between the SpectreRF calculation and the previous analysis is that the basis functions used for the SpectreRF calculation are not just a few sinusoids, but rather a collection of many piecewise polynomials. The use of piecewise polynomials allows the SpectreRF simulator to solve circuits with arbitrary waveforms, including circuits with highly nonlinear behavior.

Noise computations are usually performed with a small-signal assumption, but a rigorous small-signal characterization of phase noise is complicated because the variance in the phase of the oscillation grows unbounded over time. From a mathematical viewpoint, an oscillator is an autonomous system of differential equations with a stable limit cycle. An oscillator has phase noise because it is neutrally stable with respect to noise perturbations that move the oscillator in the direction of the limit cycle. Such *phase* perturbations persist with time, whereas transverse fluctuations are damped with a characteristic time inversely proportional to the quality factor of the oscillator.

Further care is necessary because, in general, the two types of excitations (those that create phase slippage and those responsible for time-damped fluctuations) are not strictly those that are parallel or perpendicular, respectively, to the oscillator trajectory, as is sometimes claimed (for example, in [4]).

However, one must realize that the noise powers at frequencies near the fundamental frequency correspond to correlations between points that are widely separated on the oscillator envelope. In other words, they are long-time signal effects. In fact, asymptotically (at long times), the ratio of the variance of any state variable to its power at the fundamental frequency is unity for any magnitude of the noise excitation. Therefore, in practical cases, you can consider only small deviations in the state variables when describing the phase noise.

The first step in the noise analysis is to determine the oscillator steady-state solution. This is done in the time domain using Shooting methods [5]. After the periodic steady-state is obtained, the circuit equations are linearized around that waveform in order to perform the small-signal analysis.

The time-varying linear system describing the small-signal response $vs(t)$ of the oscillator to a signal $w(t)$ can be written in general form as [6, 7]

$$\left[C(t)\frac{d}{dt} + G(t)\right]v_s \equiv L(t)v_s(t) \;=\; w(t)$$

where $C(t)$ and $G(t)$ represent the linear, small-signal, time-varying capacitance and conductance matrixes, respectively. These matrixes are obtained by linearization about the periodic steady-state solution (the limit cycle). To understand the nature of time-varying linear analysis, the concept of Floquet multipliers is introduced.

Suppose $x(t)$ is a solution to the oscillator circuit equations that is periodic with period T. If $x(0)$ is a point on the periodic solution $x_L(t)$, then $x(T) = x(0)$. If $x(0)$ is perturbed slightly off the periodic trajectory, $x(0) = x_L(0) + \delta x$, then $x(T)$ is also perturbed, and in general for small δx,

$$x(T) - x_L(T) \approx \frac{\partial x(T)}{\partial x(0)} \delta x$$

The Jacobian matrix

$$\frac{\partial x(T)}{\partial x(0)}$$

is called the sensitivity matrix. The SpectreRF simulator uses an implicit representation of this matrix both in the Shooting method that calculates the steady-state and in the small-signal analyses. To see how the sensitivity matrix relates to oscillator noise analysis, consider the effect of a perturbation at time $t = 0$ several periods later, at $t = nT$. From the above equation,

$$x(nT) - x_L(nT) \approx \left[\frac{\partial x(T)}{\partial x(0)}\right]^n \delta x$$

so

$$x(nT) - x_L(nT) \approx \sum_i C_i \lambda_i^n \phi_i$$

where ϕ_i is an eigenvector of the sensitivity matrix. The C_i are the expansion coefficients of δx in the basis of ϕ_i. If ψ_i is a left eigenvector (an eigenvector of its transpose) of the sensitivity matrix, then

$$C_i = \psi_i^T \delta x$$

Let λ be an eigenvalue of the sensitivity matrix. In the context of linear time-varying systems, the eigenvalues λ are called *Floquet multipliers*. If all the λ have magnitude less than one (corresponding to left-half-plane poles), the perturbation decays with time and the periodic trajectory is stable. If any λ has a magnitude greater than one, the oscillation cannot be linearly stable because small perturbations soon force the system away from the periodic trajectory $x_L(t)$.

A stable nonlinear physical oscillator, however, must be neutrally stable with respect to perturbations that move it in the direction of the orbit. These are not necessarily perturbation *in* the direction of the orbit because, in general,

$$\Psi \neq \phi_t$$

This is true because a time-shifted version of the oscillator periodic trajectory still satisfies the oscillator equations. In other words, one of the Floquet multipliers must be equal to unity. This Floquet multiplier is responsible for phase noise in the oscillator. The associated eigenvector determines the nature of the noise.

If $\lambda = e^{\eta}$ is a Floquet multiplier, then $\eta + ik\omega_0$ is a pole of the time-varying linear system for any integer k. Therefore, because of the unity Floquet multiplier, the time-varying linear system has poles on the imaginary axis at $k\omega_0$. This is very similar to what occurs in a pure LC resonator, and it explains the identical shape of the noise profiles.

Because operator $L(t)$ has poles at the harmonics of the oscillation frequency, numerical calculations of the noise at nearby frequencies become inaccurate if treated in a naive manner [8, 9]. To correctly account for the phase noise, the SpectreRF simulator finds and extracts the eigenvector that corresponds to the unity Floquet multiplier. To correctly extract the phase noise component, both the right and left eigenvectors are required. After these vectors are obtained, the singular (phase noise) contribution to the noise can be extracted. The remaining part of the noise can be obtained using the usual iterative solution techniques [6] in a numerically well-conditioned operation.

In Figure 8-32 on page 1345, you can see that the SpectreRF PTVL analysis correctly predicts the total noise, including the onset of 3 dB amplitude noise outside the bandwidth of the resonator. Note that this simulation was conducted at

$$E\left\{\xi^2(t)\right\} = 10^{-3}$$

which represents a very high noise level that is several orders of magnitude higher than in actual circuits. The good match of the PTVL models to the full nonlinear simulation shows the validity of the PTVL approximation.

Calculating Phase Noise

The following sections suggest simulation parameters, give you tips for using these parameters, and advise you about checking for accuracy.

Setting Simulator Options

The SpectreRF time-varying small-signal analyses are more powerful than the standard large-signal analyses (DC, TRAN) but, like any precision instrument, they also have greater sensitivity to numerical errors. For many circuits, particularly oscillators, more simulator precision is needed to get good results from the PAC, PXF, and Pnoise calculations than is needed to get good DC or TRAN results.

The small-signal analyses operate by linearizing around the periodic steady state solution. Consequently, the oscillator noise analysis, and the periodic small-signal analyses in general, inherit most of their accuracy properties from the previous PSS simulation. You must be sure the PSS simulation generates a sufficiently accurate linearization. See "What Can Go Wrong" on page 1360 for a discussion.

Table 8-1 recommends simulator options for various classes of circuits.

Table 8-1 Recommended SpectreRF Parameter Values

Circuit	errpreset	reltol	vabstol	iabstol
Easy	moderate	$1.0e^{-4}$	default	default
Hard-I	conservative	$1.0e^{-5}$	10n	1p
Hard-II		$1.0e^{-6}$	1n	0.1p
Hard-III		$1.0e^{-7}$	0.1n	0.1p

- *Easy* circuits are low-Q (about $Q < 10$) resonant oscillators, ring oscillators, and weakly-nonlinear relaxation oscillators. Most textbook circuits are in this category. This is the default setting when you set `errpreset=moderate`.
- *Hard-I* circuits are most high-Q resonant oscillators; circuits with complicated AGC, load, or bias circuitry; and relaxation or ring oscillators that exhibit moderate to strong

nonlinear or *stiff* effects. This is the best general-purpose set of options and it is the default when you set `errpreset=conservative`.

- *Hard-II* and *Hard-III* circuits include a few particularly difficult circuits.

  Usually these options are used only in a convergence study or for circuits that previously failed a conversion study using less strict options. Circuits in this category often exhibit some form of unusual behavior (see "What Can Go Wrong" on page 1360). Sometimes this behavior results from circuit properties, for example, some very high-*Q* crystal oscillators and some very stiff relaxation oscillator circuits. Occasionally, the behavior reflects a design flaw.

Usually setting `method=gear2only` is recommended for the PSS simulation (but see "What Can Go Wrong" on page 1360).

The parameters in Table A-1 are used in error control at two places.

At the local truncation error (LTE) at each transients integration. The LTE control formula is

$$v_n(t) - v_{n,\ pred}(t) < lteratio \times [reltol \times v_{n,\ max} + vabstol]$$

- To control the periodicity error over the period. The periodicity error control formula is

$$v_n(0) - v_n(T) < lteratio \times steadyratio \times [reltol \times v_{n,\ max} + vabstol]$$

The default value for `vabstol` is `1.0e`$^{-6}$, The default value for `iabstol` is `1.0e`$^{-12}$ which is accurate enough for most cases. Tighten `iabstol` when necessary. Refer to section 4.3 in [17] for detailed discussion about transient integration.

To speed up transients integration during the `tstab` stage, the default values at the `tstab` stage are `reltol=1.0e`-3, `lteratio=3.5`, `relref=sigglobal`, `maxstep=T`/25. `method=traponly`. After the `tstab` stage, those parameters are set back according to `errpreset`.

For circuits, such as extremely high Q oscillators, that need very high accuracy, you can gain further accuracy by turning on the highorder refinement `highorder=yes` and `errpreset=conservative`. This runs multi-interval Chebyshev PSS refinement after the Shooting phase of the PSS analysis.

An effective and fast method to start the oscillator is to run the new autonomous envelope analysis and to save the simulation results. Use the results of the autonomous envelope analysis as the initial condition for the PSS analysis.

A longer `tstab` stage helps with PSS convergence. However a longer `tstab` stage can slow the simulation. Using autonomous envelope analysis to establish `tstab` is considerably faster than using the transient analysis for `tstab`. See *Virtuoso Spectre Circuit Simulator RF Analysis Theory.* for information on the autonomous envelope analysis.

For releases before MMSIM60 USR1, an effective method to start the oscillation is to add a kicker to the circuit. The kicker can be either a voltage or current source. To effectively start the oscillation, the kicker has to be placed at the most sensitive place which usually is close to the oscillating transistor. The kicker can be either a PWL source or a damped sinusoidal source with frequency set to the oscillation frequency. The kicker must die down and remain stable after oscillation is established to avoid affecting the PSS analysis.

Troubleshooting Phase Noise Calculations

The SpectreRF simulator calculates noise effectively for most oscillators. However, circuits that are very stiff, very nonlinear, or just poorly designed can occasionally cause problems for the simulator. Stiff circuits exhibit dynamics with two or more very different time scales; for example, a relaxation oscillator with a square-wave-like periodic oscillation. Over most of the cycle, the voltages change very slowly, but occasional rapid transitions are present. This section describes some of the reasons for the problems, what goes wrong, how to identify problems, and how to fix them.

See "Details of the SpectreRF Calculation" on page 1353 for help troubleshooting particularly difficult circuits.

Known Limitations of the Simulator

Any circuit that does not have a stable periodic steady-state cannot be analyzed by the SpectreRF simulator because oscillator noise analysis is performed by linearizing around a waveform that is assumed to be strictly periodic.

For example, oscillators based on IMPATT diodes generate strong sub-harmonic responses and cannot be properly analyzed with the SpectreRF simulator. As another example, Colpitts oscillators, properly constructed, can be made to exhibit chaotic as well as sub-harmonic behavior.

Similarly, any circuit with significant large-signal response at tones other than the fundamental and its harmonics might create problems for the simulator. Some types of varactor-diode circuits might fit this category. In addition, some types of AGC circuitry and bias circuitry can create these effects.

The SpectreRF simulator cannot simulate these circuits because simulation of an autonomous circuit with sub-harmonic or other aperiodic components in the large signal response essentially requires foreknowledge of which frequency components are important. Such foreknowledge requires Fourier analysis of very long transient simulations and cannot be easily automated. Such simulations can be very expensive.

What Can Go Wrong

The SpectreRF simulator can have problems in the following situations.

Generic PSS Simulation Problems

Any difficulties in the underlying PSS analysis affect the phase noise computation. For example, underestimating the oscillator period or failing to start the oscillator properly can cause PSS convergence problems that make running a subsequent Pnoise analysis impossible.

Hypersensitive Circuits

Occasionally, you might see circuits that are extremely sensitive to small parameter changes. Such a circuit was a varactor-tuned VCO that had the varactor bias current, and therefore the oscillation frequency, set by a $1\ \mathrm{T\Omega}$ resistor. Changing to a $2\ \mathrm{T\Omega}$ resistor, which is a $1e^{-12}$ relative perturbation in the circuit matrixes, changed the oscillation frequency from 125 MHz to 101 Mhz. Such extreme circuit sensitivity results in very imprecise PSS simulations. In particular, the calculated periods have relatively large variations. If precise PSS simulations are impossible, precise noise calculations are also impossible. In such a case, you must fix the circuit.

Subharmonics or Parametric Oscillator Modulation

Sometimes bias and AGC circuitry might create small-amplitude parasitic oscillations in the large signal waveform. You can identify these oscillations by performing a transient simulation to steady-state and then looking for modulation of the envelope of the oscillation waveform. For high-Q circuits and/or low-frequency parasitics, this transient simulation might be very long.

In this case, because the oscillator waveform is not actually periodic, the PSS simulation can only converge to within approximately the amplitude of the parasitic oscillation. If the waveform possesses a parasitic oscillation that changes amplitude, over one period, around 10^{-5} relative to the oscillator envelope, then convergence with `reltol` < 10^{-5} is probably not possible (assuming `steadyratio` is one or less).

These effects might also appear as a parametric sideband amplification phenomenon.

See "Frequently Asked Questions" on page 1364 for more information.

Small-Signal Frequency is Much Higher than the Fundamental Frequency

The same timesteps are used for both the small-signal analysis and the PSS analysis. If the small-signal frequency is much higher than the fundamental frequency, much smaller timesteps might be required to accurately resolve the small-signal than are needed for the large signal. To force the SpectreRF simulator to take sufficiently small timesteps in the PSS simulation, be sure the `maxacfreq` parameter is set correctly.

Wide Timestep Variation

Occasionally, in simulations that generate PSS waveforms with timesteps that vary over several orders of magnitude, the linear systems of equations that determine the small-signal response become ill-conditioned. As a result, the noise analysis is inaccurate. Usually this occurs because you have requested excessive simulator precision; for example, nine-digit precision. You can sometimes eliminate this problem using `method = traponly` in the PSS solution. You might also set `maxstep` to a very small value in the PSS analysis or you might specify a very large `maxacfreq` value.

Problems with Device Models

When the device models leave their physically meaningful operating range during the large-signal PSS solution, the noise calculations are usually inaccurate. Similarly, when the models are discontinuous, or have discontinuous derivatives, the small-signal analysis might be inaccurate.

Problems Resolving Floquet Multipliers in Stiff Relaxation Oscillators

Sometimes in very stiff relaxation oscillators, the PSS solution rapidly and easily converges; but the numerically calculated Floquet multiplier associated with the PSS solution is far from unity. Typically, this multiplier is real and has a magnitude much larger than unity. The SpectreRF simulator prints a warning (see "Message III" on page 1363). It is interesting that sometimes the phase noise is quite accurate even with low simulation tolerances. If you have this problem, perform a convergence study.

Problems Resolving Floquet Multipliers in High-Q Resonant Circuits

In a physical oscillator, there is one Floquet multiplier equal to unity. In an infinite-Q linear resonator, however, the multipliers occur in complex conjugate pairs. A very high-Q nonlinear oscillator has another Floquet multiplier on the real axis nearly equal to, but slightly less than, one. In this presence of numerical error, however, these two real Floquet multipliers can appear to the simulator as a complex-conjugate pair. The phase noise is computed using the Floquet vector associated with the unity Floquet multiplier. When the two multipliers appear as a complex pair, the relevant vector is undefined. When the SpectreRF simulator correctly identifies this situation, it prints a warning (see "Message III" on page 1363). The solution is usually to simulate using the next higher accuracy step (see Table 8-1 on page 1357). Sometimes varying `tstab` can also help with this problem.

If the circuit is really an infinite-Q resonator (for example, a pure parallel LC circuit), the multipliers always appear as complex conjugate pairs and the noise computations are not accurate close to the fundamental frequency. Such circuits are not physical oscillators, and the SpectreRF simulator is not designed to deal with them; see "Amplitude Noise and Phase Noise in the Linear Model" on page 1352 and "Frequently Asked Questions" on page 1364.

Phase Noise Error Messages

SpectreRF displays error messages when it encounters several types of known numerical difficulty. To interpret the error messages produced by the phase noise analysis, you must know the material in "Details of the SpectreRF Calculation" on page 1353.

Message I

The Floquet eigenspace computed by spectre PSS analysis appears to be inaccurate. PNOISE computations may be inaccurate. Consider re-running the simulation with smaller reltol and method=gear2only.

The eigenvector responsible for phase noise was inaccurately computed and the PSS simulation tolerances might be too loose. Try simulating the circuit at the next higher accuracy setting (see Table 8-1 on page 1357) and then compare the calculated noise in the two simulations.

Message II

The Floquet eigenspace computed by spectre PSS analysis appears to be ill-defined. PNOISE computations may be inaccurate. Consider re-running the simulation with smaller reltol, different tstab(s), and method=gear2only. Check the circuit for unusual components.

This can be an accuracy problem, or it can result from an unusual circuit topology or sensitivity. Tighten the accuracy requirements as much as possible (see Table 8-1 on page 1357). If this message appears in all simulations, the noise might be incorrect even if the simulations agree.

Message III

The Floquet eigenspace computed by spectre PSS analysis appears to be inaccurate and/or the oscillator possesses more than one stable mode of oscillation. PNOISE computations may be inaccurate. Consider re-running the simulation with smaller reltol, different tstab(s), and method=gear2only.

All the real Floquet multipliers were well-separated from unity, suggesting that the PSS simulation tolerances might be too loose. Simulate the circuit at the next higher accuracy setting (see Table 8-1 on page 1357) and then compare the calculated noise in the two simulations. If the calculated noise does not change, it is probably correct even if this message appears in both simulations.

The tstab Parameter

Because SpectreRF performs the PSS calculation in the time domain by using a *Shooting method*, an infinite number of possible PSS solutions exist, depending on where the first timepoint of the PSS solution is placed relative to the oscillator phase.

The placement of the first timepoint is determined by the length of the initial transient simulation, which you can control using the `tstab` parameter. If the `tstab` value causes the edges of the periodic window to fall on a point where the periodic oscillator waveform is making very rapid transitions, it is very difficult for PSS to converge. Similarly, the results of the small-signal analyses are probably not very accurate. Avoid such situations. If the start of the PSS waveform falls on a very fast signal transition, you usually need to view the results of further small-signal analyses with some skepticism.

Although a poor choice of the `tstab` parameter value can degrade convergence and accuracy, appropriate use of `tstab` can help to identify problem circuits and to estimate the reliability of their noise computations.

If you perform several PSS and Pnoise computations that differ only in their `tstab` parameter values, the results should be fairly similar, within a relative deviation of the same order of magnitude as the simulator parameter `reltol`. If this is not the case, you might not have set the simulator accuracy parameters sufficiently tight to achieve an accurate solution; and you need to reset one or more of the parameters `reltol`, `vabstol`, or `iabstol`. The circuit might also be poorly designed and very sensitive to perturbations in its parameters.

If the calculated fundamental period of the oscillator varies with `tstab` even when you set `reltol`, `iabstol`, and `vabstol` to very small (but not vanishingly small) values, the circuit is probably poorly designed, exhibiting anomalous behavior, or both. (see "Known Limitations of the Simulator" on page 1359).

Frequently Asked Questions

The following questions are similar to those commonly asked about oscillator noise analysis with the SpectreRF simulator.

Does SpectreRF simulation calculate phase noise, amplitude noise, or both?

SpectreRF simulation computes the total noise of the circuit, both amplitude and phase noise. What the analog circuit design environment plots as *phase noise* is really the total noise scaled by the power in the fundamental oscillation mode. Close enough to the fundamental frequency, the noise is all phase noise, so what the analog circuit design environment plots of *phase noise* is really the phase noise as long as it is a good ways above the noise floor.

Some discussions of oscillator noise based on a simple resonator/amplifier description describe the total noise, at small frequency offsets from the fundamental, as being half amplitude noise and half phase noise. In reality, for physical oscillators, near the fundamental nearly all the noise is phase noise. Therefore, these simple models overestimate the total noise by 3 dB. For a detailed explanation, see the phase noise theory described in "Details of the SpectreRF Calculation" on page 1353 and the detailed discussion of the Van der Pol oscillator "Linear Time-Varying (LTV) Models" on page 1348.

I have a circuit that contains an oscillator. Can I simulate the oscillator separately and use the phase noise SpectreRF calculates as input for a second PSS/PNOISE simulation?

No. Oscillators generate noise with correlated spectral sidebands. Currently, SpectreRF simulation output represents only the time-average noise power, not the correlation information, so the noise cannot be input to a simulation that contains time-varying elements that might mix together noise from separate frequencies.

If the second circuit is a linear filter (purely lumped linear time-invariant elements, such as resistors, capacitors, inductors, or a linearization of a nonlinear circuit around a DC operating point) that generates no frequency mixing, then you can use the output of the SpectreRF Pnoise analysis as a *noisefile* for a subsequent NOISE (not Pnoise) analysis.

How accurate are the phase noise calculations? What affects the errors?

Initially, it is important to distinguish between modeling error and simulation (numerical) error. If the device models are only good to 10% the simulation is only good to 10% (or worse). So, for the rest of this appendix, we discuss numerical error introduced by the approximations in the algorithms.

You must also distinguish between absolute and relative signal frequencies in the noise analysis. When the noise frequency is plotted on an absolute scale, the error is primarily a function of the variance in the calculated fundamental period. This is true because of the singular behavior, in these regions, of the phase noise near a harmonic of the fundamental. To see this behavior, note that for the simple oscillator driven by white noise, the noise power is proportional to the offset from the fundamental frequency,

$$S_v(\omega) \propto \frac{1}{(\omega - \omega_0)^2}$$

If you make a small error in the calculation of ω_0, the error ΔS_v in the noise is proportional to $^1S / {}^1\omega_0$

$$\Delta S_v(\omega) \propto \frac{\Delta\omega_0}{(\omega - \omega_0)^3}$$

This error can be very large even if $\Delta\omega_0$, the error in ω_0, is small. However, because of the way SpectreRF simulation extracts out the phase noise, the calculated phase noise, as a function of offset from the fundamental frequency, can be quite accurate even for very small offsets.

Now consider how much error is present in the calculated fundamental frequency. Because the numerical error is related to many simulation variables, it is difficult to quantify, without examination, how much is present. However, as a rough approximation, if we define the quantity

$$r = min\left\{ reltol, \frac{iabstol}{max(i)}, \frac{vabstol}{max(v)} \right\}$$

where $max(i)$ and $max(v)$ are the maximum values of current and voltage over the PSS period, then, under some assumptions, $\Delta\omega_0$, the error in the fundamental ω_0, probably satisfies

$$r\omega_0 < \delta\omega_0 < Mr\omega_0$$

where M is the number of timesteps taken for the PSS solution. This analysis assumes that `steadyratio` is sufficiently tight, not much more than one, and also that `iabstol` and `vabstol` are sufficiently small.

If you require a good estimate of the accuracy in the fundamental, run the PSS simulation with many different accuracy settings, initial conditions and `tstab` values (See "The tstab Parameter" on page 1363). For example, to estimate how much numerical error remains in the calculated fundamental frequency for a given simulation, run the simulation; reduce `reltol`, `iabstol`, and `vabstol` by a factor of 10 to100; rerun the simulation; and then compare the calculated fundamental frequencies. For the sorts of parameters we recommend for oscillator simulations, four to five digits of precision seems typical. Past that point, round off error and anomalous effects introduced by vastly varying timesteps offset any gains from tightening the various accuracy parameters.

For phase noise calculations, again it is unrealistic to expect relative precision of better than the order of `reltol`. That is, if `reltol` is 10^{-5} and the oscillator fundamental is about 1 GHz, the SpectreRF numerical fuzz for the calculated period is probably about 10 KHz. Therefore, when plotted on an absolute frequency scale, the phase noise calculation exhibits substantial variance within about 10 KHz of the fundamental.

However, when plotted on a frequency scale *relative* to the fundamental, the phase noise calculation might be more precise for many oscillators. If the circuit is strongly dissipative (that is, low-Q, such as ring oscillators and relaxation oscillators), the phase noise calculation is probably fairly accurate up to very close to the fundamental frequency even with loose simulation tolerance settings. High-Q circuits are more demanding of the simulator and require more stringent simulation tolerances to produce good results. In particular, circuits that use varactor diodes as tuning elements in a high-Q tank circuit appear to cause occasional problems. Small modifications to the netlist (runs with different `tstab` values and minor topology changes) can usually tell you whether (and where) the simulator results are reliable.

Simulation accuracy is determined by how precisely SpectreRF simulation can solve the augmented nonlinear boundary value problem that determines the periodic steady-state. The accuracy of the BVP solution is controlled primarily by the simulation variables `reltol`, `iabstol`, `vabstol`, `steadyratio`, and `lteratio`. Typically, `steadyratio` and `lteratio` are fixed, so `reltol` is usually the variable of interest.

Occasionally accuracy might be somewhat affected by other variables such as `relref`, `method`, the number of timesteps, and `tstab`. Again, the physical properties of the circuit might limit the accuracy.

I have a circuit with an oscillator and a sinusoidal source. Can I simulate this circuit with SpectreRF simulation?

In general, SpectreRF simulation is not intended to analyze circuits that contain autonomous oscillators and independent periodic sources.

If the circuit contains components that could potentially oscillate autonomously and also independent large-signal sinusoidal sources, SpectreRF simulation works properly only if two conditions are fulfilled. The system must be treated as a driven system, and the coupling from the sinusoidal sources to the oscillator components must be strong enough to lock the oscillator to the independent source frequency. (In different contexts, this is known as *oscillator entrainment* or *phase-locking*) The normal (nonautonomous) PSS and small-signal analyses function normally in these conditions.

If the autonomous and driven portions of the circuit are weakly coupled, the circuit waveform might be more complicated; for example, a two-tone (quasi-periodic) signal with incommensurate frequencies. (Incommensurate frequencies are those for which there is no period that is an integer multiple of the period of each frequency.) Even if PSS converges, further small-signal analyses (PAC, PXF, Pnoise) almost certainly give the wrong answers.

What is the significance of total noise power?

First, you must understand that SpectreRF simulation calculates and measures noise in voltages and currents. The total power in the phase process is unbounded, but the power in the actual state variables is bounded.

Oscillator phase noise is usually characterized by the quantity

$$d(f) = \frac{Sv(f)}{P_1}$$

where P_1 is the power in the fundamental component of the steady state solution and $Sv(f)$ is the power spectral density of a state variable V.

For an oscillator with only white-noise sources, $L(f)$ has a Lorentzian line shape,

$$L(f) = \frac{1}{\pi}\frac{a}{a^2 + f^2}$$

where a is dependent on the circuit and noise sources, and thus the total phase noise power

$$\int L(f)df = 1$$

Because

$$varv(t) = R_v(t,t) = \int_{-\infty}^{\infty} S_v(f)df$$

we are led to the uncomfortable, but correct, conclusion that the variance in any variable is 100 percent of the RMS value of the variable, *irrespective of circuit properties or the amplitude of the noise sources*.

Physically, this means that if a noise source has been active, because $t = -\infty$, then the voltage variable in question is randomly distributed over its whole trajectory. Therefore, the relative variance is one. Clearly, the variance is not a physically useful characterization of the noise, and the total noise power must be interpreted carefully. What is actually needed is the variance as a function of time, given a fixed reference for the signal in question; or, more often, the rate at which the variance increases from a zero point; or, sometimes, the increment in the variance from cycle to cycle. That is, we want to specify the phase of the oscillator signal at a given time point and to find a statistical characterization of the variances relative to that time. But because of the non-causal nature of the Fourier integral, quantities like the total noise power give us information about the statistical properties of the signal over all time.

What's the story with pure linear oscillators (LC circuits)?

Oddly enough, SpectreRF simulation is not set up to do Pnoise analysis on pure LC circuits.

Pure LC circuits are not physically realizable oscillators, and the mathematics that describes them is different from the mathematics that describes physical oscillators. A special option must be added to the code in order for Pnoise to handle *linear oscillators*. See "Models for Phase Noise" on page 1346, and, in particular, "Amplitude Noise and Phase Noise in the Linear Model" on page 1352. Because the normal NOISE analysis is satisfactory for these circuits and also much faster, it is unlikely that Pnoise is modified.

Why doesn't the SpectreRF model match my linear model?

As is discussed in "Amplitude Noise and Phase Noise in the Linear Model" on page 1352, the difference between the SpectreRF model (the correct answer) and the linear oscillator model is that in the linear oscillator, both the amplitude and the phase fluctuations can become large. However, in a nonlinear oscillator, the amplitude fluctuations are always bounded, so the noise is half as much, asymptotically.

We emphasize that computing the correct total noise power requires using the time-varying small signal analysis. An oscillator is, after all, a time-varying circuit by definition. Time-invariant analyses, like the *linear oscillator model*, can sometimes be useful, but they can also be misleading and should be avoided.

There are funny sidebands/spikes in the oscillator noise analysis. Is this a bug?

Very possibly this is parametric small-signal amplification, a real effect. This sometimes occurs when there is an AGC circuit with a very long time constant modulating the parameters of circuit elements in the oscillator loop. Sidebands in the noise power appear at frequencies offset from the oscillator fundamental by the AGC characteristic frequency.

Similarly, any elements that can create a low-frequency parasitic oscillation, such as a bias inductor resonating with a capacitor in the oscillator loop, can create these sorts of sidebands.

Further Reading

The best references on the subject of phase noise are by Alper Demir and Franz Kaertner. Alper Demir's thesis [10], now a Kluwer book, is a collection of useful thinking about noise. Kaertner's papers [11, 12, 9] contain a reasonably rigorous and fairly mathematical treatment of phase noise calculations.

The book by W. P. Robins [13] has a lot of engineering-oriented thinking. However, it makes heavy use of LTI models, and much of the discussion about noise cannot be strictly applied to oscillators. As a consequence, you must interpret the results in this book with care.

Hajimiri and Lee's paper [4] is worth reading, but their analysis is superseded by Kaertner's.

Other references include [8, 14, 15, 16].

References

[1] P. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*. Springer-Verlag, 1995.

[2] D. Leeson, "A simple model of feedback oscillator noise spectrum," *Proc. IEEE*, vol. 54, pp. 329–330, 1966.

[3] W. A. Gardner, *Introduction to random processes*. McGraw Hill, 1990.

[4] A. Hajimiri and T. Lee, "A general theory of phase noise in electrical oscillators," *IEEE Journal of Solid. State Circuits*, vol. 33, pp. 179–193, 1998.

[5] R. Telichevesky, J. White, and K. Kundert, “Efficient steady-state analysis based on matrix-free krylov-subspace methods,” in *Proceedings of 32rd Design Automation Conference*, June 1995.

[6] R. Telichevesky, J. White, and K. Kundert, “Efficient AC and noise analysis of two-tone RF circuits,” in *Proceedings of 33rd Design Automation Conference*, June 1996.

[7] M. Okumura, T. Sugaware, and H. Tanimoto, “An efficient small-signal frequency analysis method of nonlinear circuits with two frequency excitations,” *IEEE Transactions on Computer-Aided Design*, vol. 9, pp. 225–235, 1990.

[8] W. Anzill and P. Russer, “A general method to simulate noise in oscillators based on frequency domain techniques,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 41, pp. 2256–2263, 1993.

[9] F. X. Kärtner, “Noise in oscillating systems,” in *Proceedings of the Integrated Nonlinear Microwave and Millimeter Wave Circuits Conference*, 1992.

[10] A. Demir, *Analysis and simulation of noise in nonlinear electronic circuits and systems*. PhD thesis, University of California, Berkeley, 1997.

[11] F. X. Kaertner, “Determination of the correlation spectrum of oscillators with low noise,” *IEEE Trans. Microwave Theory and Techniques*, vol. 37, pp. 90–101, 1989.

[12] F. X. Kaertner, “Analysis of white and f-a noise in oscillators,” *Int. J. Circuit Theory and Applications*, vol. 18, pp. 485–519, 1990.

[13] W. P. Robins, *Phase Noise in Signal Sources*. Institution of Electrical Engineers, 1982.

[14] A. A. Abidi and R. G. Meyer, “Noise in relaxation oscillators,” *IEEE J. Sol. State Circuits*, vol. 18, pp. 794–802, 1983.

[15] B. Razavi, “A study of phase noise in cmos oscillators,” *IEEE J. Sol. State Circuits*, vol. 31, pp. 331–343, 1996.

[16] K. Kurokawa, “Noise in synchronized oscillators,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 16, pp. 234–240, 1968.

[17] K. Kundert, “The Designer’s Guide to Spice & Spectre,” *Kluwer Academic Publishers*, 1995.

Measuring AM, PM and FM Conversion

Derivation

Consider a sinusoid that is simultaneously both amplitude and phase modulated as in Equation 8-1.

(8-1) $$v_m(t) = A_c(1+\alpha(t))\cos(\omega_c t + \phi_c + \phi(t))$$

In Equation 8-1, A_c, ϕ_c, ω_c, are the amplitude, phase and angular frequency of the carrier, while $\alpha(t)$, and $\phi(t)$ are the amplitude and phase modulation.

When you assume that $\phi(t)$ is small for all t, this allows the narrowband angle modulation approximation as in Equation 8-2. See [ziemer76].

(8-2) $$v_m(t) = A_c(1+\alpha(t))[\cos(\omega_c t + \phi_c) - \phi(t)\sin(\omega_c t + \phi_c)]$$

Converting to complex exponentials gives Equation 8-3.

(8-3) $$v_m(t) = \frac{A_c}{2}(1+\alpha(t)) \left[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)} + j\phi(t)\left(e^{j(\omega_c t + \phi_c)} - e^{-j(\omega_c t + \phi_c)}\right)\right]$$

Letting both the amplitude and phase modulation be complex exponentials with the same frequency, ω_m, gives Equations 8-4, 8-5, 8-6, 8-7 and 8-8.

(8-4) $$\alpha(t) = Ae^{j\omega_m t}$$

(8-5)
$$\phi(t) = \Phi \frac{e^{j\omega_m t} + e^{-j\omega_m t}}{2}$$

Where

(8-6)
$$A = A_{A e^{j\phi_A t}}$$

(8-7)
$$\Phi = A_{\Phi e^{j\phi_\Phi t}}$$

(8-8)
$$v_m(t) = \frac{A_c}{2}\left(1 + A e^{j\omega_m t}\right) \left[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)} + j\frac{1}{2}\Phi\left(e^{j\omega_m t} + e^{-j\omega_m t}\right)\left(e^{j(\omega_c t + \phi_c)} - e^{-j(\omega_c t + \phi_c)}\right)\right]$$

Assuming that both A and Φ are small and neglecting cross modulation terms gives Equation 8-9.

(8-9)
$$v_m(t) = \frac{A_c}{2}\left[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)} + A e^{j\omega_m t} e^{j(\omega_c t + \phi_c)} + A e^{j\omega_m t} e^{-j(\omega_c t + \phi_c)} + j\frac{\Phi}{2}\left(e^{j\omega_m t} + e^{-j\omega_m t}\right) e^{j(\omega_c t + \phi_c)} - j\frac{\Phi}{2}\left(e^{j\omega_m t} + e^{-j\omega_m t}\right) e^{-j(\omega_c t + \phi_c)}\right]$$

Simplifying gives Equation 8-10.

(8-10)

$$v_m(t) = \frac{A_c}{2}\Big[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)} + Ae^{j((\omega_m + \omega_c)\,t + \phi_c)} + Ae^{j((\omega_m - \omega_c)\,t - \phi_c)} + \frac{1}{2}j\Phi\Big[e^{j((\omega_m + \omega_c)\,t + \phi_c)} - e^{j((\omega_m - \omega_c)\,t - \phi_c)} + e^{j((-\omega_m + \omega_c)t + \phi_c)} - e^{j((-\omega_m - \omega_c)t - \phi_c)}\Big]\Big]$$

In Equation 8-10,

- The AM terms are

$$Ae^{j((\omega_m + \omega_c)\,t + \phi_c)} + Ae^{j((\omega_m - \omega_c)\,t - \phi_c)}$$

- The PM terms are

$$\frac{1}{2}\Big[j\Phi e^{j((\omega_m + \omega_c)\,t + \phi_c)} - j\Phi e^{j((\omega_m - \omega_c)\,t - \phi_c)} + j\Phi e^{j((-\omega_m + \omega_c)\,t + \phi_c)} - j\Phi e^{j((-\omega_m - \omega_c)\,t - \phi_c)}\Big]$$

Because the left-side term $v_m(t)$ represents a real-time signal, only the real parts of the complex terms are of interest. Dropping the imaginary parts and rearranging Equation 8-10 produces Equation 8-11,

(8-11)

$$v_m(t) = \frac{A_c}{2}\Big[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)} + Ae^{j(\omega_m - \omega_c)\,t}e^{-j\phi_c} - j\Phi e^{j(\omega_m - \omega_c)\,t}e^{-j\phi_c} + Ae^{j(\omega_m + \omega_c)\,t}e^{j\phi_c} + j\Phi e^{j(\omega_m + \omega_c)\,t}e^{j\phi_c}\Big]$$

When you ignore the negative ω_m term in Equation 8-11, you get

- The LSB (lower sidebands) terms are

$$Ae^{j(\omega_m - \omega_c)t}e^{-j\phi_c} - j\Phi e^{j(\omega_m - \omega_c)t}e^{-j\phi_c}$$

- The USB (upper sidebands) terms are

$$Ae^{j(\omega_m + \omega_c)t}e^{j\phi_c} + j\Phi e^{j(\omega_m + \omega_c)t}e^{j\phi_c}$$

Assume that you perform a PAC analysis, which applies a single complex exponential signal that generates responses at the upper and lower sidebands of the ω_c signal. Assume the transfer functions are L and U, so the lower and upper sideband signals are given by Equations 8-12 and 8-13.

(8-12) $$l(t) = Le^{j(\omega_m - \omega_c)t}$$

(8-13) $$u(t) = Ue^{j(\omega_m + \omega_c)t}$$

Where

$$L = A_L e^{j\phi_L}$$

$$U = A_U e^{j\phi_U}$$

Matching common frequency terms between Equations 8-11, 8-12, and 8-13 gives Equations 8-14, 8-15, 8-16 and 8-17.

(8-14) $$L = \frac{A_c}{2}\left(Ae^{-j\phi_c} - j\Phi e^{-j\phi_c}\right)$$

(8-15) $$U = \frac{A_c}{2}\left(Ae^{j\phi_c} + j\Phi e^{j\phi_c}\right)$$

(8-16) $$\frac{2}{A_c}Le^{j\phi_c} = A - j\Phi$$

(8-17) $$\frac{2}{A_c}Ue^{-j\phi_c} = A + j\Phi$$

Solving for the modulation coefficients gives Equations 8-18 and 8-19.

(8-18) $$A = \frac{1}{A_c}\left(Le^{j\phi_c} + Ue^{-j\phi_c}\right)$$

(8-19) $$\Phi = \frac{j}{A_c}\left(Le^{j\phi_c} - Ue^{-j\phi_c}\right)$$

Thus, Equation 8-18 gives the transfer function for amplitude modulation and Equation 8-19 gives the transfer function for phase modulation.

Positive Frequencies

Notice that `L` is defined in Equation 8-12 on page 1374 to be the transfer function from the input to the sideband at $\omega_m - \omega_c$, which is a negative frequency. This is usually a natural definition for use with the Virtuoso® Spectre® circuit simulator RF analysis (SpectreRF) small signal analyses (depending on the setting of the `freqaxis` parameter). It can be cumbersome though when the only data available is at positive frequencies. Thus, the transfer function to $\omega_c - \omega_m$ is defined as

$$\tilde{L}$$

Then, as in Equation 8-20,

(8-20) $$\tilde{l}(t) = \tilde{L}e^{j(\omega_c - \omega_m)t}$$

Because the signals are real, L is a complex conjugate of

$$\tilde{L}$$

And the reverse is also true, as in Equation 8-21,

(8-21) $$L = \tilde{L}^*$$

Equations 8-22 and 8-23 are produced by rewriting Equation 8-18 on page 1375 and Equation 8-19 on page 1375 in terms of

$$\tilde{L}$$

(8-22) $$A = \frac{1}{A_c}\left(\tilde{L}^* e^{j\phi_c} + Ue^{-j\phi_c}\right)$$

(8-23) $$\Phi = \frac{j}{A_c}\left(\tilde{L}^* e^{j\phi_c} - Ue^{-j\phi_c}\right)$$

FM Modulation

For FM modulation, the phase modulation $\phi(t)$ becomes the integral of the FM modulation signal, $\omega(t)$ as shown in Equation 8-24.

(8-24) $$v_m(t) = A_c \cos(\omega_c t + \phi(t))$$

Where

(8-25) $$\phi(t) = \int \omega(t)dt$$

Recall from Equation 8-5 on page 1372 and Equation 8-19 on page 1375 that

(8-26) $$\phi(t) = \frac{j}{A_c}\left(Le^{j\phi_c} - Ue^{-j\phi_c}\right)e^{j\omega_m t}$$

Combining Equation 8-25 and Equation 8-26 and the differentiating both sides results in Equations 8-27, 8-28, and 8-29.

(8-27) $$\omega(t) = \frac{\omega_m}{A_c}\left(Ue^{-j\phi_c} - Le^{j\phi_c}\right)e^{j\omega_m t}$$

(8-28) $$\Omega = A_\Omega e^{j\phi_\Omega} = \frac{\omega_m}{A_c}\left(Ue^{-j\phi_c} - Le^{j\phi_c}\right)$$

Or

(8-29) $$\Omega = j\omega_m \Phi$$

Simulation

The test circuit, represented by the two netlists shown in Example on page 1378 and Example on page 1378, was run with SpectreRF. The test circuit consists of three, linear, periodically-varying modulators that are driven with the same input. The input is constant valued in the large signal `PSS` analysis, and generates a single complex exponential analysis during the `PAC` analysis. The idea is to compute the transfer functions from this input to the upper and lower sidebands at the output of the modulators and then use the derivation just

described to convert these transfer functions into transfer functions to the AM, PM, and FM modulations and then check the simulation results against the expected results.

Notice that `freqaxis=out`. This is necessary to match the derivation. If you would rather use `freqaxis=absout`, you would have to use the complex conjugate of `L` as in Equation 8-22 on page 1376 and Equation 8-23 on page 1376.

Netlist for the AM, PM, and FM Conversion Test Circuit

```
// AM, PM, and FM modulation test circuit

simulator lang=spectre
ahdl_include "modulators.va"

parameters MOD_FREQ=10MHz
parameters CARRIER_FREQ=1GHz

Vin (in 0) vsource pacmag=1 pacphase=0
Mod0 (unmod in) AMmodulator freq=CARRIER_FREQ mod_index=0
Mod1 (am in) AMmodulator freq=CARRIER_FREQ mod_index=1
Mod2 (pm in) PMmodulator freq=CARRIER_FREQ kp=1
Mod3 (fm in) FMmodulator freq=CARRIER_FREQ fd=MOD_FREQ

waves pss fund=CARRIER_FREQ outputtype=all tstab=2ns harms=1
xfer pac start=MOD_FREQ maxsideband=4 freqaxis=out
```

The netlist for the modulator models shown in Example 8-29, has the filename `modulators.va`.

Netlist for the Modulator Models Written in Verilog-A

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

module AMmodulator (out, in);
    input in;
    output out;
    electrical out, in;
    parameter real freq = 1 from (0:inf);
    parameter real mod_index = 1;

    analog begin
        V(out) <+ (1+mod_index*V(in)) * cos(2*`M_PI*freq*$abstime);
        $bound_step( 0.05 / freq );
    end
endmodule

module PMmodulator (out, in);
    input in;
    output out;
    electrical out, in;
    parameter real freq = 1 from (0:inf);
```

```
    parameter real kp = 1 from (0:inf);

    analog begin
        V(out) <+ cos(2*`M_PI*freq*$abstime + kp*V(in));
        $bound_step( 0.05 / freq );
    end
endmodule

module FMmodulator (out, in);
    input in;
    output out;
    electrical out, in;
    parameter real freq = 1 from (0:inf);
    parameter real fd = 1 from (0:inf);
    real phi;

    analog begin
        V(out) <+ cos(2*`M_PI*(freq*$abstime + idtmod(fd*V(in),0,1, -0.5)));
        $bound_step( 0.05 / freq );
    end
endmodule
```

Results

The simulations were run with various values for `pacphase` on `Vin`.

Table 8-2 shows results for the output of the AM modulator with $v_{LO} = cos(\omega_c t)$.

Table 8-3 shows results for the output of the PM modulator with $v_{LO} = cos(\omega_c t)$.

Table 8-2 Results for the AM Modulator Output

pacphase	L	U	A	Φ
0	1/2	1/2	1	0
45	$\frac{1+j}{2\sqrt{2}}$	$\frac{1+j}{2\sqrt{2}}$	$\frac{1+j}{2}$	0
90	$j/2$	$j/2$	j	0
180	$-1/2$	$-1/2$	-1	0

Table 8-3 Results for the PM Modulator Output

pacphase	L	U	A	Φ
0	$-1/2$	1/2	0	1
45	$\frac{1-j}{2\sqrt{2}}$	$\frac{j-1}{2\sqrt{2}}$	0	$\frac{1+j}{\sqrt{2}}$
90	1/2	$-1/2$	0	j
180	$j/2$	$-j/2$	0	-1

If you repeat the simulations but replace the *cos* function in the modulators with the *sin* function, which is equivalent to changing the `LO` to $v_{LO} = sin(\omega_c t)$ or setting $\phi_c = -90$, you achieve the following results.

- Table 8-4 shows results for the output of the AM modulator with $v_{LO} = sin(\omega_c t)$.

- Table 8-5 shows results for the output of the PM modulator with $v_{LO} = sin(\omega_c t)$.

Table 8-4 Results for the AM Modulator Output

pacphase	L	U	A	Φ
0	$j/2$	$-1/2$	1	0
45	$\frac{j-1}{2\sqrt{2}}$	$\frac{1-j}{2\sqrt{2}}$	$\frac{1+j}{\sqrt{2}}$	0
90	$-1/2$	$1/2$	j	0
180	$-j/2$	$j/2$	-1	0

Table 8-5 Results for the PM Modulator Output

pacphase	L	U	A	Φ
0	1/2	1/2	0	1
45	$\frac{1+j}{2\sqrt{2}}$	$\frac{1+j}{2\sqrt{2}}$	0	$\frac{1+j}{\sqrt{2}}$
90	$j/2$	$j/2$	0	j
180	$-1/2$	$-1/2$	0	-1

Finally, Table 8-6 shows the results for the FM modulator with $v_{LO} = cos(\omega_c t)$. The FM modulator has a modulation coefficient of ω_m built-in, which renormalizes the results.

Table 8-6 Results for the FM Modulator Output

pacphase	L	U	Ω
0	$-1/2$	1/2	1
45	$-\frac{1+j}{2\sqrt{2}}$	$\frac{1+j}{2\sqrt{2}}$	$\frac{1+j}{\sqrt{2}}$
90	$-j/2$	$j/2$	j
180	1/2	$-1/2$	-1

Conclusion

This appendix shows that the `PAC` analysis can be used to determine the level of AM or PM modulation that appears on a carrier. This is done by applying a small signal and using the phase of the carrier along with the transfer function to the upper and lower sidebands of the carrier to compute an AM or PM transfer function.

References

[Ziemer 76] R. Ziemer and W. Tranter. *Principles of Communications: Systems, Modulation, and Noise.* Houghton Miffin, 1976.

[Robins 96] W. Robins. *Phase Noise in Signal Sources (Theory and Application)*. IEE Telecommunications Series, 1996.

9

Noise-Aware PLL Flow

Noise-aware PLL Flow

Phase-locked loops (PLL) are at the core of some important RF designs such as frequency synthesis, clock and data recovery, and clock de-skew. Some of the PLL characteristics, such as phase noise, jitter, power supply and substrate noise interference, step response, acquisition time, and static phase offset are of interest to most designers, but to measure these characteristics, the simulator must overcome the following obstacles:

- The large difference in the time-constant between the VCO and other PLL blocks

  The VCO can operate in the range of GHz; the PFD and CP in the range of MHz; the output of LPF and control of VCO in the range of KHz. Transient analysis forces all the PLL blocks to use the same small time-steps required by the VCO frequency, which makes the analysis very time-consuming.

- The VCO is an autonomous circuit while the other blocks are driven circuits

  PSS and QPSS analyses cannot simulate this kind of circuit.

- The PLL generates repetitive switching events as an essential part of its operation, and the noise performance must be evaluated in the presence of this large-signal behavior

  Transient noise analysis is very time-consuming and neither PNOISE nor QPNOISE analyses can be used.

The SpectreRF noise-aware PLL flow provides a solution to these challenges. This flow uses a macro-model based simulation methodology, in which the PFD-CP, VCO, and divider in the PLL are replaced by automatically generated macro-models that characterize the behavior of the original blocks. The VCO and the divider are integrated into one macro-model, greatly improving the simulation efficiency. In addition, both integer N and fractional-N PLLs can be simulated.

Preparation

When a PLL is used as a frequency synthesizer, the implementation usually consists of four main blocks: PFD-CP, LPF, VCO, and divider. Figure 9-1 on page 1384 is an example of a synthesizer schematic that includes these blocks.

Figure 9-1 Block diagram of a frequency synthesizer

The primary steps in setting up and using the noise-aware PLL flow are:

1. Preparing the PFD-CP, VCO, divider, and LPF subcircuits.
2. Inserting the PFD-CP into the PFD-CP test bench and running PSS and PNOISE with the PFD-CP extraction plugin, which automatically generates a Verilog-A model.
3. Inserting the VCO into the VCO test bench and running PSS and PNOISE with the VCO extraction plugin, which automatically generates a `.mat` file defining a VCO CMI macro-model.
4. Simulating the divider with PSS and PNOISE, which generates a Verilog-A model for the divider jitter.
5. Inserting the VCO CMI model, the PFD-CP Verilog-A module, the divider jitter Verilog-A model, and the LPF subcircuit into the PLL macro-model test bench and running a TRAN analysis with the phase noise PSD plugin.

Note: Be aware of the following information.

- The test benches mentioned above are included in `pllMMLib`.
- The divider's large signal behavior is merged into the VCO.
- To use the plugin, ADE needs to run in batch mode. This option can be set in the `.cdsinit` file by using

```
envSetVal("spectre.envOpts" "controlMode" 'string "batch")
```

- The command for running the Spectre simulator with a plugin is

```
spectre -plugin *.so
```

The following plugin files (`.so`) and CMI lib file are needed:

- `libpllPPVoscModel_sh.so` (for VCO macro-model extraction)
- `libpllTTpfd_cpModel_sh.so` (for PFD-CP macro-model extraction)
- `libpllDivider_sh.so` (for divider macro-model extraction)
- `libpllMMpsd_sh.so` (for phase noise PSD calculation)
- `libpll_sh.so` (VCO macro-model CMI lib)

Using the Noise-Aware PLL flow in ADE

The above mentioned step 2 (PFD-CP extraction), step 3 (VCO extraction), and step 5 (PLL simulation with macro-models) are integrated into ADE as a wizard. To open the wizard, in the Analog Design Environment (ADE), choose *Tools – RF – Wizard – PLL*.

Figure 9-2 PLL Macro Model Wizard

To use the flow, the *Enable PLL Macro Model* option must be selected. For PFD-CP extraction, divider extraction or VCO extraction, a library name and cell name must be specified to indicate where the corresponding macro-model is placed after it is generated. For PLL simulation, *Bench Type* should be *PLL Bench*.

VCO Extraction

Character of the Large Signal

A VCO is an oscillator whose output frequency (f_{out}) can be tuned by an input control voltage (V_{ctrl}). The relationship between the output frequency and the input voltage is known as the tuning curve. A simple VCO schematic (a) and its tuning curve (b) are shown in Figure 9-3 on page 1387. In the tuning curve beyond V_{min} and V_{max}, the VCO output frequency changes little with V_{ctrl} change. V_{DD} and V_{SS} variation can also have an impact on the output frequency.

Figure 9-3 Simple VCO and tuning curve

Based on the VCO tuning curve, a linear table VCO model can be built, which is

(9-1) $$f_{out} = K_i \cdot v(t) + f_i$$

where K_i is the VCO gain around the control voltage V_i and $v(t)$ is the control voltage deviation from V_i. From Equation 9-1 on page 1387, the VCO output phase can be obtained as

(9-2) $$\phi_{out}(t) = 2\pi f_i t + \int K_i v(t) dt_i$$

In the SpectreRF noise-aware PLL flow, the VCO macro-model is a nonlinear table model based on perturbation projection vector (PPV) theory. In this model, the VCO output phase is determined by

(9-3) $$\phi_{out}(t) = 2\pi f_i t + 2\pi f_i a(t)$$

(9-4) $$\frac{d\alpha(t)}{dt} = ppv^T \cdot n(t)$$

where $n(t)$ represents noise sources at every node and branch. The PPV can be output using SpectreRF PSS or PNOISE analysis. The nonlinear model can be used to characterize the following VCO effects that are difficult challenges for the linear model.

- Injection locking
- Phase noise due to supply and substrate noise

VCO Internal Noise

Generally the internal noise sources consist of white and flicker noise and have the form

(9-5) $$S_u(\Delta f) \sim a + \frac{b}{\Delta f}$$

where Δf is the offset frequency, a is white noise and another item represents flicker noise. Under noise perturbation, the VCO output phase noise takes the form

(9-6) $$S_\phi(\Delta f) \sim \frac{a}{\Delta f^2} + \frac{b}{\Delta f^3}$$

VCO Macro-Model

The VCO macro-model is a table model that is obtained by sweeping the input control voltage V_{ctrl}, V_{DD} and V_{SS}. Its main content is shown in Table 9-1 on page 1389. V_{out} is the VCO output waveform. The VCO macro-model looks up values in the table according to the input

control voltage, V_{DD} and V_{SS} and calculates the output based on Equation 9-3 on page 1388 and Equation 9-6 on page 1388.

Table 9-1

$V_{ctrl}[1]$	$f_{out}[1]$	$V_{out}[1]$	PPV[1]	$S_{\phi}[1]$
...	...	...	...	...
$V_{ctrl}[i]$	$f_{out}[i]$	$V_{out}[i]$	PPV[i]	$S_{\phi}[i]$
...	...	...	...	...
$V_{ctrl}[n]$	$f_{out}[n]$	$V_{out}[n]$	PPV[n]	$S_{\phi}[n]$

Extracting the VCO Macro-Model

To extract the VCO macro-model,

1. Insert the VCO transistor-level circuit into a VCO extraction test bench from the `pllMMLib` library.
   - For a VCO with one output terminal, use `vco_bench`.
   - For a VCO with two outputs, use `vco_twin_bench`.
   - For a VCO with four outputs, use `vco_tet_bench`.

   Figure 9-3 on page 1390 is the schematic of the test bench for a VCO with two outputs.

Figure 9-4 VCO test bench for macro-model extraction

2. In the PLL Macro Model Wizard, select *VCO Bench*.
3. Select the kind of VCO template to be used.

   There are four VCO templates that can be selected: *Normal*, *Fast*, *fractional-N*, and *Fast Frac-N*, which correspond to four symbols: `oscmm`, `oscmm_fast`, `oscmm_frac`, and `oscmm_frac_fast`. (See "Using VCO Macro-Models" on page 1393 for information about the four symbols).
4. In the *VCO Multiphase* field, select the number of outputs used by the VCO.

Figure 9-5 PLL Macro Model Wizard

5. Specify a library name and a cell name to contain the generated VCO macro-model.
6. Select the kind of VCO sweep.
   - If the *VTUNE* value is chosen for VCO Sweep, the three parameters `vtunestart`, `vtunestop`, and `vtunesteps` are set in the netlist
   - If the *VDD* value is chosen for *VCO Sweep*, the three parameters `vddstart`, `vddstop`, and `vddsteps` are set in the netlist. For example,
   - parameters vddstart=3.2 vddstop=3.4 vddsteps=3
   - If the *VSS* value is chosen for *VCO Sweep*, the three parameters `vssstart`, `vssstop`, and `vsssteps` are set in the netlist.
7. Set up a PSS + PNOISE analysis. In the PNOISE analysis, set `oprobe=vco_vout` rather than setting two output nets.
8. Run the simulation with the VCO extraction plugin.

Using the plugin for VCO extraction, the command line is

```
spectre -plugin libpllPPVoscModel_sh.so
```

After extraction, a VCO macro-model is generated in the specified library and an associated file `oscPPVmodel.mat` is generated. The content of the file is similar to that illustrated in Table 9-1 on page 1389.

Note: Be aware of the following information.

- In PLL designs, there are often buffers and prescalers after the VCO that could have a loading effect. These circuits can be included in the VCO extraction. However, including too many circuits or circuits that are large can affect the accuracy of the VCO macro-model.
- The plugin searches the source instances `vco_vtune`, `vco_vdd`, `vco_vss`, and `vco_vout` to get the related nets and terminals measurement results. These four names must not be changed.
- The V_{ctrl} sweep range should be between V_{min} and V_{max} (see Figure 9-3 on page 1387). A sweep grid that uses small steps increases the accuracy of the macro-model at the cost of increased extraction time. A good strategy is to use small steps in the grid near the locked V_{ctrl} value and a coarse grid farther away.
- The V_{DD} and V_{SS} sweeps are optional. They simulate the impact of the large supply variation impact on the PLL. If the supply noise can be seen as a small signal, the V_{DD} and V_{SS} sweeps are not necessary. If both are selected, the total number of PSS + PNOISE runs are `N1*N2*N3`, where `N1` is the number of sweep points for V_{ctrl} , `N2` is the number for V_{DD} , and `N3` is the number for V_{SS}. So, the simulation can be very time-consuming.
- If no PNOISE analysis is set, the generated VCO macro-model is noiseless.
- VCO noise includes both white noise and flicker noise. The frequency sweep range should be larger than a decade. The noise below the start frequency (and beyond the stop frequency) point is treated as white noise with power the same as that of the start frequency (or stop frequency). So, if a one frequency point sweep is set, the VCO noise is white.
- From Equation 9-4 on page 1388, `vco_vdd` and `vco_vss` are equivalent for VCO extraction. From this point, `vco_vdd` and `vco_vss` can be any voltage source whose noise can affect the VCO output, not only the drain or source supply for the transistor. For example, if the analog supply AVDD noise and the digital supply DVDD noise are considered in the VCO at the same time, you can extract the VCO by connecting `vco_vdd` to AVDD and `vco_vss` to DVDD.

Using VCO Macro-Models

There are 12 symbols (4 templates times 3 multiphase choices) that are supported for the oscmm model.

- oscmm Symbol on page 1394
- oscmm_fast Symbol on page 1395
- oscmm_frac Symbol on page 1397
- oscmm_frac_fast Symbol on page 1400

The oscmm symbols with `twin` in their names have two output terminals so that the symbols can be used to simulate differential VCOs.

- oscmm_twin Symbol on page 1401
- oscmm_fast_twin Symbol on page 1403
- oscmm_frac_twin Symbol on page 1404
- oscmm_frac_fast_twin Symbol on page 1406

The oscmm symbols that have `tet` in their names have four output terminals so that the symbols can be used to simulate VCOs with four output phases.

- oscmm_tet Symbol on page 1407
- oscmm_fast_tet Symbol on page 1409
- oscmm_frac_tet Symbol on page 1410
- oscmm_frac_fast_tet Symbol on page 1412

oscmm Symbol

Pins:

pin	use
`div_p`	Divider output.
`div_gnd`	Divider ground.
`vtune`	VCO control voltage input.
`vdd`	VCO power voltage input.
`vss`	VCO substrate voltage input.
`osc_p`	VCO output.
`osc_gnd`	VCO ground.

Parameters:

parameter	use
`model_file`	Name of the file containing the VCO macro-model. Default value is `oscPPVmodel.mat`.
`divider_ratio`	The VCO output divided by this ratio is the divider output.
`vlo`	VCO and divider output low level voltage.
`vhi`	VCO and divider output high level voltage.

parameter	use
`bound`	Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement.
`vcojitter`	This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from `oscPPVmodel.mat`. If specified as zero, the VCO is noiseless.
`jitterstart`	Determines when the VCO noise (or jitter) takes effect in the transient.
`noiseseed`	A positive integer that serves as the seed for the random number generator. Specifying the same seed allows you to reproduce a previous experiment.
osctrtf	Risetime and falltime of VCO output waveform.
divtrtf	Risetime and falltime of divider output waveform.

oscmm_fast Symbol

Pins:

pin	use
`div_p`	Divider output.
`div_gnd`	Divider ground.

pin	use
`vtune`	VCO control voltage input.
`vdd`	VCO power voltage input.
`vss`	VCO substrate voltage input.

Parameters:

parameter	use
`model_file`	Name of the file containing the VCO macro-model. Default value is `oscPPVmodel.mat`.
`divider_ratio`	The VCO output divided by this ratio is the divider output.
`vlo`	VCO and divider output low level voltage.
`vhi`	VCO and divider output high level voltage.
`bound`	Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement.
`vcojitter`	This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from `oscPPVmodel.mat`. If specified as zero, the VCO is noiseless.
`jitterstart`	Determines when the VCO noise (or jitter) takes effect in the transient.
`noiseseed`	A positive integer that serves as the seed for the random number generator. Specifying the same seed allows you to reproduce a previous experiment.

Note: The `oscmm_fast` symbol does not include VCO output, which makes the simulation very fast. However, this lack of VCO output disables phase noise measurement at the VCO output.

oscmm_frac Symbol

Pins:

pin	use
`div_p`	Divider output.
`div_gnd`	Divider ground.
`vtune`	VCO control voltage input.
`vdd`	VCO power voltage input.
`vss`	VCO substrate voltage input.
`osc_p`	VCO output.
`osc_gnd`	VCO ground.
`div_ctl<0:11>`	Control bus. Controls the divider ratio instantly according to the signal on it.

Parameters:

parameter	use
`model_file`	Name of file containing the VCO macromodel. Default value is `oscPPVmodel.mat`.
`vlo`	VCO and divider output low level voltage.
`vhi`	VCO and divider output high level voltage.

parameter	use
bound	Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement.
vcojitter	This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from oscPPVmodel.mat. If specified as zero, the VCO is noiseless.
jitterstart	Determines when the VCO noise (or jitter) takes effect in the transient.
div_ctl_vlo	Low level of the control signal on the control bus.
div_ctl_vhi	High level of the control signal on the control bus.
div_ratios	Array of divide ratios. For example, [2477 2478 2479 2480].
noiseseed	A positive integer that serves as the seed for the random number generator. Specifying the same seed allows you to reproduce a previous experiment.
mappingmode	Mapping mode between divider control bus signal and divider ratio. Possible values are parammapping, filemapping and signalmapping. Default value is parammapping.
mappingfile	The file which contains the mapping between divider control bus signal and divider ratio.
osctrtf	Risetime and falltime of VCO output waveform.
divtrtf	Risetime and falltime of divider output waveform.

Note: Be aware of the following information.

- The oscmm_frac symbol allows you to change the divider ratio instantly, which can be used to simulate a fractional-N PLL or PLL channel selection.
- The signals on the div_ctl<0:11> bus are seen as digital signals. The high level is specified by parameter div_ctl_vhi and the low level by parameter div_ctl_vlo. The threshold voltage is determined by (div_ctl_vhi+div_ctl_vlo)/2. At any time, the signals form a control byte CTLBYTE with a value range of 0~4095. The divider's instant divide ratio is determined by CTLBYTE and the mapping mode. If mapping mode is "parammapping", the parameter div_ratios. div_ratios is an array and its maximum length is 4096. Suppose div_ratios(n) represents the nth value in the

`div_ratios` array, the mapping between `CTLBYTE` and the divider's instant divide ratio is shown in Table 9-2 on page 1399.

Table 9-2

CTLBYTE	divider ratio
0x000	div_ratios(0)
…	…
0x064	div_ratios(100)
…	…
0xFFF	div_ratios(4095)

If mapping mode is filemapping, the parameter mappingfile gives the mapping between CTLBYTE and the divider ratio. The file content is two columns ASCII, the first column is CTLBYTE (which must be binary) and the second column is divider ratio. If an instant CTLBYTE is not found in the file, the previous divider ratio will be used.

For example, if a file has the following columns:

010110101111 120
000101010011 121
000111010001 122

which shows that the divider ratio has changed from 120 to 122, according to CTLBYTE.

If mapping mode is signalmapping, CTLBYTE is directly interpreted as the divider ratio which is in the range of 1 ~ 4095, such as 0x064 representing a divider ratio of 100.

oscmm_frac_fast Symbol

Pins:

pin	use
`div_p`	Divider output.
`div_gnd`	Divider ground.
`vtune`	VCO control voltage input.
`vdd`	VCO power voltage input.
`vss`	VCO substrate voltage input.
`div_ctl<0:11>`	Control bus.

Parameters:

parameter	use
`model_file`	Name of file containing the VCO macromodel. Default value is `oscPPVmodel.mat`.
`vlo`	VCO and divider output low level voltage.
`vhi`	VCO and divider output high level voltage.
`bound`	Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement.

parameter	use
`vcojitter`	This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from `oscPPVmodel.mat`. If specified as zero, the VCO is noiseless.
`jitterstart`	Determines when the VCO noise (or jitter) takes effect in the transient.
`div_ctl_vlo`	Low level of the control signal on the control bus.
`div_ctl_vhi`	High level of the control signal on the control bus.
`div_ratios`	Array of divide ratios. Such as [2477 2478 2479 2480].

Note: Be aware of the following information.

- The `oscmm_frac_fast` symbol does not include VCO output, which makes simulation very fast. However, this lack of VCO output disables phase noise measurement at the VCO output.

oscmm_twin Symbol

Pins:

pin	use
`div_p`	Divider output 0.
`div_p1`	Divider output 1.

pin	use
osc_p	VCO output 0.
osc_p1	VCO output 1.
div_gnd	Divider ground.
vtune	VCO control voltage input.
vdd	VCO power voltage input.
vss	VCO substrate voltage input.
osc_gnd	VCO ground.

Parameters:

parameter	use
model_file	Name of the file containing the VCO macro-model. Default value is oscPPVmodel.mat.
divider_ratio	The VCO output divided by this ratio is the divider output.
vlo	VCO and divider output low level voltage.
vhi	VCO and divider output high level voltage.
bound	Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement.
vcojitter	This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from oscPPVmodel.mat. If specified as zero, the VCO is noiseless.
jitterstart	Determines when the VCO noise (or jitter) takes effect in the transient.
noiseseed	A positive integer that serves as the seed for the random number generator. Specifying the same seed allows you to reproduce a previous experiment.

oscmm_fast_twin Symbol

Pins:

pin	use
`div_p`	Divider output 0.
`div_p1`	Divider output 1.
`div_gnd`	Divider ground.
`vtune`	VCO control voltage input.
`vdd`	VCO power voltage input.
`vss`	VCO substrate voltage input.

Parameters:

parameter	use
`model_file`	Name of the file containing the VCO macro-model. Default value is `oscPPVmodel.mat`.
`divider_ratio`	The VCO output divided by this ratio is the divider output.
`vlo`	VCO and divider output low level voltage.
`vhi`	VCO and divider output high level voltage.
`bound`	Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement.

parameter	use
`vcojitter`	This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from `oscPPVmodel.mat`. If specified as zero, the VCO is noiseless.
`jitterstart`	Determines when the VCO noise (or jitter) takes effect in the transient.
`noiseseed`	A positive integer that serves as the seed for the random number generator. Specifying the same seed allows you to reproduce a previous experiment.

oscmm_frac_twin Symbol

Pins:

pin	use
`div_p`	Divider output 0.
`div_p1`	Divider output 1.
`osc_p`	VCO output 0.
`osc_p1`	VCO output 1.
`div_gnd`	Divider ground.
`vtune`	VCO control voltage input.
`vdd`	VCO power voltage input.

pin	use
vss	VCO substrate voltage input.
osc_gnd	VCO ground.
div_ctl<0:11>	Control bus.

Parameters:

parameter	use
model_file	Name of file containing the VCO macromodel. Default value is oscPPVmodel.mat.
vlo	VCO and divider output low level voltage.
vhi	VCO and divider output high level voltage.
bound	Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement.
vcojitter	This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from oscPPVmodel.mat. If specified as zero, the VCO is noiseless.
jitterstart	Determines when the VCO noise (or jitter) takes effect in the transient.
div_ctl_vlo	Low level of the control signal on the control bus.
div_ctl_vhi	High level of the control signal on the control bus.
div_ratios	Array of divide ratios. For example, [2477 2478 2479 2480].
noiseseed	A positive integer that serves as the seed for the random number generator. Specifying the same seed allows you to reproduce a previous experiment.

oscmm_frac_fast_twin Symbol

Pins:

pin	use
`div_p`	Divider output 0.
`div_p1`	Divider output 1.
`div_gnd`	Divider ground.
`vtune`	VCO control voltage input.
`vdd`	VCO power voltage input.
`vss`	VCO substrate voltage input.
`div_ctl<0:11>`	Control bus.

Parameters:

parameter	use
`vhi`	VCO and divider output high level voltage.
`bound`	Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement.
`vcojitter`	This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from `oscPPVmodel.mat`. If specified as zero, the VCO is noiseless.

parameter	use
`jitterstart`	Determines when the VCO noise (or jitter) takes effect in the transient.
`div_ctl_vlo`	Low level of the control signal on the control bus.
`div_ctl_vhi`	High level of the control signal on the control bus.
`div_ratios`	Array of divide ratios. Such as, [2477 2478 2479 2480].
`vhi`	VCO and divider output high level voltage.
`bound`	Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement.

oscmm_tet Symbol

Pins:

pin	use
`div_p`	Divider output 0.
`div_p1`	Divider output 1.
`div_p2`	Divider output 2.
`div_p3`	Divider output 3.
`osc_p`	VCO output 0.
`osc_p1`	VCO output 1.

pin	use
osc_p2	VCO output 2.
osc_p3	VCO output 3.
div_gnd	Divider ground.
vtune	VCO control voltage input.
vdd	VCO power voltage input.
vss	VCO substrate voltage input.
osc_gnd	VCO ground.

Parameters:

parameter	use
model_file	Name of file containing the VCO macromodel. Default value is oscPPVmodel.mat.
vlo	VCO and divider output low level voltage.
vhi	VCO and divider output high level voltage.
bound	Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement.
vcojitter	This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from oscPPVmodel.mat. If specified as zero, the VCO is noiseless.
jitterstart	Determines when the VCO noise (or jitter) takes effect in the transient.
div_ctl_vlo	Low level of the control signal on the control bus.
div_ctl_vhi	High level of the control signal on the control bus.

oscmm_fast_tet Symbol

Pins:

pin	use
`div_p`	Divider output 0.
`div_p1`	Divider output 1.
`div_p2`	Divider output 2.
`div_p3`	Divider output 3.
`div_gnd`	Divider ground.
`vtune`	VCO control voltage input.
`vdd`	VCO power voltage input.
`vss`	VCO substrate voltage input.

Parameters:

parameter	use
`model_file`	Name of the file containing the VCO macro-model. Default value is `oscPPVmodel.mat`.
`divider_ratio`	The VCO output divided by this ratio is the divider output.
`vlo`	VCO and divider output low level voltage.
`vhi`	VCO and divider output high level voltage.

parameter	use
`bound`	Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement.
`vcojitter`	This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from `oscPPVmodel.mat`. If specified as zero, the VCO is noiseless.
`jitterstart`	Determines when the VCO noise (or jitter) takes effect in the transient.
`noiseseed`	A positive integer that serves as the seed for the random number generator. Specifying the same seed allows you to reproduce a previous experiment.

oscmm_frac_tet Symbol

Pins:

pin	use
`div_p`	Divider output 0.
`div_p1`	Divider output 1.
`div_p2`	Divider output 2.
`div_p3`	Divider output 3.
`osc_p`	VCO output 0.

pin	use
osc_p1	VCO output 1.
osc_p2	VCO output 2.
osc_p3	VCO output 3.
div_gnd	Divider ground.
vtune	VCO control voltage input.
vdd	VCO power voltage input.
vss	VCO substrate voltage input.
osc_gnd	VCO ground.
div_ctl<0:11>	Control bus.

Parameters:

parameter	use
model_file	Name of file containing the VCO macromodel. Default value is oscPPVmodel.mat.
vlo	VCO and divider output low level voltage.
vhi	VCO and divider output high level voltage.
bound	Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement.
vcojitter	This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from oscPPVmodel.mat. If specified as zero, the VCO is noiseless.
jitterstart	Determines when the VCO noise (or jitter) takes effect in the transient.
div_ctl_vlo	Low level of the control signal on the control bus.
div_ctl_vhi	High level of the control signal on the control bus.
div_ratios	Array of divide ratios. For example, [2477 2478 2479 2480].

parameter	use
noiseseed	A positive integer that serves as the seed for the random number generator. Specifying the same seed allows you to reproduce a previous experiment.

oscmm_frac_fast_tet Symbol

Pins:

pin	use
div_p	Divider output 0.
div_p1	Divider output 1.
div_p2	Divider output 2.
div_p3	Divider output 3.
div_gnd	Divider ground.
vtune	VCO control voltage input.
vdd	VCO power voltage input.
vss	VCO substrate voltage input.
div_ctl<0:11>	Control bus.

Parameters:

parameter	use
model_file	Name of file containing the VCO macromodel. Default value is oscPPVmodel.mat.
vlo	VCO and divider output low level voltage.
vhi	VCO and divider output high level voltage.
bound	Minimum number of time steps per period. A value of 1 speeds up simulation but a large number, such as 10, produces more accurate noise measurement.
vcojitter	This parameter assigns VCO period jitter due to white noise. If this value is not specified, oscmm calculates the period jitter from oscPPVmodel.mat. If specified as zero, the VCO is noiseless.
jitterstart	Determines when the VCO noise (or jitter) takes effect in the transient.
div_ctl_vlo	Low level of the control signal on the control bus.
div_ctl_vhi	High level of the control signal on the control bus.
div_ratios	Array of divide ratios. Such as, [2477 2478 2479 2480].

oscmm CMI Command Line

Using oscmm CMI, the command line is

```
spectre -cmiconfig myconfig.cfg
```

myconfig.cfg is a config file, its content is as follows.

```
append   ... /tools.lnx86/spectre/lib
load libpll_sh.so
```

Extracting the PFD-CP Macro-Model

A simple three state PFD-CP schematic (a) and its input and output relationships (b) are shown in Figure 9-6 on page 1414.

Figure 9-6 PFD-CP input and output

It receives two input signals, the reference clock (V_{ref}) and the feedback clock (V_{fd}). The CP output current signal (I_{out}) changes according to the phase difference ($\Delta\tau$ or $\Delta\phi$) of the two PFD input signals. If only the average output current $<I_{out}>$ is considered, then

(9-7) $$\langle I_{out}\rangle = I_{max} \cdot (\Delta t) / T$$

where T is the period of the two input signals and I_{max} is the CP maximum output current. Generally, $<I_{out}>$ is noisy for PFD-CP internal noise. Assuming that the output current noise variance is $(\sigma_{<Iout>})^2$, the variance can be equivalent to the input phase noise (or input edge jitter), as follows,

(9-8) $$J_{ee} = \sigma_{\langle Iout\rangle} \cdot T / I_{max}$$

where J_{ee} is the input signal edge jitter. $\sigma_{<Iout>}$ can be calculated as follows,

(9-9) $$\sigma_{\langle Iout\rangle} = \sqrt{\frac{1}{2}\int_0^{\infty} S_n(f)\,df}$$

where $S_n(f)$ is the PFD-CP total output noise power spectral density, which can be obtained using PNOISE analysis.

There are several nonideal effects that impact PFD-CP performance.

- PFD-CP current mismatch

Figure 9-7 Ideal PFD_CP output when the PLL is locked

The `UP` current and `DOWN` current open at the same time, so the net output should be zero. Generally, for a real circuit, the `UP` current and `DOWN` current are mismatched, which leads to a nonzero net output, although the input reference and feedback clock phases are the same (see (a) in Figure 9-6 on page 1416). To remain locked, the reference and feedback clock must have a small offset so that the net output current in one cycle is zero (see (b) in Figure 9-6 on page 1416). This output has periodic ripples that can modulate `Vtune` and lead to a closed-loop VCO output that has spurs with offset frequency `N*`f_{ref}, where N is a positive integer and f_{ref} is a reference frequency.

Figure 9-8 Effect of UP and DOWN current mismatch on PFD-CP

- PFD-CP delay mismatch

  Figure 9-6 on page 1417 shows the delay mismatch. Suppose the PLL is locked and the `UP` and `DOWN` signals arrive at the same time, which will open the `UP` and `DOWN` current for a short interval. If the delay from the `UP` signal to the `UP` current is different from that from the `DOWN` signal to the `DOWN` current, the net output current in a cycle is not constant but instead has ripples or spurs.

Figure 9-9 Effect of UP and DOWN current delay mismatch on PFD-CP

These irregularities are injected into the `Vtune` and impact the PLL output.

Other nonideal effects, such as rising time mismatch, falling time mismatch, and output current offset, can have a similar effect on the performance of the PFD-CP.

- PFD-CP jitter variation with phase difference

  The jitter and output current variation of the PFD-CP is dependent on the phase difference between the input reference clock, feedback clock and the load voltage. Simulation performed on a sample design shows the jitter dependency on phase difference in Figure 9-6 on page 1418.

Figure 9-10 PFD-CP jitter related to phase difference

PFD-CP Macro-model Extraction

For PFD-CP macro-model extraction, the PFD-CP transistor level circuit is instantiated in the PFD-CP extraction test bench `pfd_cp_bench`, which is provided in the `pllMMLib` library. Figure 9-11 on page 1419 is the schematic of the test bench.

Figure 9-11 PFD-CP test bench for macro-model extraction

To run the extraction,

1. Set up a PSS + PNOISE analysis. Set the reference clock and feedback clock to square waves with the same reference frequency. Set the correct DC value for *Load source* so that the CP can work normally under the bias.
2. In the PLL Macro Model Wizard, select *PFDCP Bench*.
3. Specify a library name and a cell name to contain the PFD-CP macro-model.

Figure 9-12 PLL Macro Model Wizard

4. The optional *PFDCP Sweep* item is for simulating the jitter and output current dependency on phase difference. The three supported kinds of sweep are *Disable, fine*, and *coarse*.

   - If *Disable* is selected, jitter and the current is measured at 0.1*T and -0.1*T
   - Selecting a *fine* sweep requires the most extraction time but the created macro-model is the most accurate. If *fine* is selected, the parameter `jittersweep=fine` is set in the netlist. This selection measures jitter and output current values at phase difference equals 0.1*T, 0.01*T, 0.001*T, 0.000*T, 0, -0.0001*T, -0.001*T, -0.01*T, -0.1*T respectively, where T is the reference clock period.
   - If *coarse* is selected, jitter and the current is measured at 0.1*T, 0 and -0.1*T.

5. The optional Load Voltage Sweep option can be used for simulating the load voltage effect on the jitter and output current.
6. The static locking angle of the PFD-CP can be accommodated by adding a variable in ADE with the variable name `phase_offset`, with the static locking angle as the value for `phase_offset`.
7. Run the simulation with the PFD-CP extraction plugin. After PFD-CP extraction finishes, the PFD-CP Verilog-A macro-model is generated.

Note: Be aware of the following information.

- The Spectre simulator command for using the plugin for PFD-CP extraction is

  ```
  spectre -plugin libpllTTpfd_cpModel_sh.so
  ```

- The plugin searches the `Vref`, `Vfb`, and `Load` source instances to get the related nets and terminals measurement results. These three names must not be changed.
- Usually, a tri-state PFD-CP is a strongly nonlinear circuit, so it is best to use Shooting PSS + PNOISE to extract the PFD-CP model. Also, Shooting is usually more accurate than Harmonic Balance for obtaining the rising time and delay, which are time domain parameters.
- Set the two parameters `dir` and `vth`. The first parameter specifies the PFD input trigger edge (a `dir` value >= 0 sets a rise edge trigger; a `dir` value <0 sets a fall edge trigger); the second parameter specifies the plugin trigger threshold voltage.
- Use the vsource `delay` parameter to set the phase difference between V_{ref} and V_{fb}. The delay difference of V_{fb} and V_{ref} can be arbitrary.
- Choose the upper limit of the sweep frequency range of the PNOISE analysis so that the total noise at frequencies outside the range is negligible. The noise should be at least 40 dB down and dropping at the highest frequency simulated. Choose the lower limit of integration by considering what value is below the system bandwidth and the amount of validity in the flicker noise model, which can grow unbounded as the frequency decreases.

Using PFD-CP macro-model

pllTTpfd_cp Symbol

pins:

pin	use
`out`	CP current output.
`ref`	Reference clock input.
`fb`	Feedback clock input.

Parameters:

parameter	use
`Imax`	CP maximum output current.
`Vtrans`	Input trigger threshold.
`vth`	pfd_cp trigger threshold voltage.
`uptr`	Up current rise or fall time.
`downtr`	Down current rise or fall time.
`refdelay`	Delay time from reference clock input to up current output.
`fbdelay`	Delay time from feedback clock input to down current output.

parameter	use
`Imis`	Mismatch between up current and down current. It is defined by $$I_{mis} = \frac{I_{up} - I_{down}}{I_{max}}$$ $$I_{max} = \frac{I_{up} - I_{down}}{2}$$
`Ioffset`	Output current offset.
`dir`	Set `dir`=1 for a rising edge trigger and `dir`=-1 for a falling edge trigger.
`jitter`	Equivalent input edge jitter from PFD-CP internal noise.
`modelfile`	The file contains PFD-CP macro-model jitter information.
`noisestart`	The time when jitter noise starts.
`noiseseed`	Jitter noise seed.

Note: Be aware of the following information.

- If the `dir` parameter is not set during extraction, its default value is 1.
- If the `vth` parameter is not set during extraction, the default value of `Vtrans` is 1.0.

Divider Extraction

Divide Macro-Model Basis

Divider's large signal behavior +N is integrated into VCO macro-model, so extracted divider s macro-model will only include its small signal (jitter) behavior, risetime, falltime and delay.

Suppose v*(t)* represents the divider's ideal periodical output, a falling edge threshold cross event happens at (t0, v1) (see figure G-13 (a)). However, when noise *n(t)* is added to the signal, $v_n(t) = v(t) + n(t)$, the cross event is displaced slightly and happens at (t0+j(t_0), v1) (see G-13 (b)), j(t_0) is instant jitter. The jitter is determined by n(t) and the slew rate of the signal dV(t)/dt.

Generally *n* is not stationary, but cyclostationary. It is only important to know when the noisy periodic signal *v (t)* crosses the threshold, so the statistics of *n* are only significant at the time when $v_n(t)$ crosses the threshold. In this case, the edge-edge jitter can be calculated:

$$J_{ee} = \frac{\sqrt{\mathrm{var}(n(t_0))}}{dv(t_0)/dt}$$

Figure 9-13 PDF-CP input and output

Divider Macro-Model Extraction

For divider macro-model extraction, the divider transistor level circuit is instantiated in the divider extraction test bench `divider_bench`, which is provided in the `pllMMLib` library. Figure G-14 is the schematic of the test bench.

Figure 9-14 Divider test bench for macro-model extraction

To run the extraction:

1. Set up a PSS + PNOISE analysis.

   The input signal should be set to square waves. PNOISE sweep upper frequency should be set to $f_0/2$, where f_0 is input signal s frequency. The noise type should be set to “jitter” in PNOISE.

2. In the ADE PLL wizard, select *Divider Bench*.

3. Specify a *Library Name* and a *Cell Name* to contain the divider macro-model.

Figure 9-15

4. Run the simulation with divider extraction plugin. After divider extraction finishes, a Verilog-A module file is generated, which is the divider Verilog-A macro-model.

Note:

- Using the plugin for divider extraction, the command line is:

  ```
  spectre plugin libpllDivider_sh.so
  ```

- The plugin searches the Vin and Vout to get the related nets and terminals measurement results. These names cannot be changed.
- At present, only Shooting PSS + PNOISE can be used to extract divider macro-model.
- When noise type is jitter, SpectreRF uses strobed pnoise analysis to compute $S_n(f)$. The sample point should be set to coincide with the point where the output signal crosses the threshold of the subsequent stage (the phase detector) in the appropriate direction.

Using Divider Macro-Model

Model name: jee_gen

Symbol: pllDivider

pins: (out in)

out	Divider output.
In	Divider input.

Parameters:

`Vth`	Threshold voltage of input cross event.
`High`	High level voltage of divider output.
`Low`	Low level voltage of divider output.
`Frequency_Divider`	Output signal's frequency.
`modelFile`	File which contains divider model information. It is generated automatically during extraction. Main content in this file is jitter power spectrum.
`Tr`	Risetime of divider output signal.
`Tf`	Falltime of divider output signal.
`Td`	Delay time from input to output.
`Jitter`	Divider jitter (J_{ee}). The jitter has Gaussian distributions. If this parameter is provided, the macro-model s internal jitter is disabled. By default, the macro-model s internal jitter is frequency dependent.

`noiseSeed`	Noise seed of jitter random process.
`noiseStart`	Time when jitter is added to output signal.

Note: This jee_gen model can be also used to simulate reference clock noise. That is, if you can provide reference clock's jitter power spectrum according modelFile format, the model will reproduce the jitter profile.

PLL Simulation with Macro-Models

After PFD-CP and VCO extraction, the whole PLL simulation with these two macro-models can be performed. Figure 9-16 on page 1430 and Figure 9-17 on page 1431 show the PLL test bench in `pllMMLib`.

If the PLL noise performance is of interest, add a `freq_meter` instance to the test bench. This instance measures the periods of VCO output (`vco_p` and `vco_n`) or divider output (`div_p div_n`) in response to rise cross events and writes the periods into a file. After a transient analysis, a plugin searches this instance and the file and calculates the phase noise power spectrum density (PSD) from the periods in the file.

To prepare for and run the simulation, do the following:

1. Attach the LPF circuit into `lpf_ckt`.
2. Insert the extracted `pllTTpfd_cp` view and pllDivider view into the PLL bench. Set parameters if needed.
3. Insert the extracted oscmm view into the PLL bench. Set the oscmm parameters.
4. Set the `freq_meter` parameters. There are four parameters used in `freq_meter`:

parameter	use
`Vthup`	Assigns the rise cross threshold voltage.
`ttol`	Time tolerance for cross event.
`outfile`	Assigns the output file name (default is `periods.txt`).
`outStart`	Gives the time when the measured periods will be output to the outfile.

5. In the PLL Macro Model Wizard, select *PLL Bench*.
6. Set the Window size. The default window size is 1024. This specifies the number of transient points in each overlapping section to run an FFT on. Small numbers (short sequences) produce more averaging and a higher minimum frequency in the psd and larger delta frequencies in the psd itself. This should be a power of two. This can result in losing the ability to see the spurious output of the PLL itself. If this happens, increase the window size and run a longer transient analysis in the PLL Bench.
7. Set up the Tran analysis and run the simulation.

If the PSD plugin is used, a file `pllmmnoise.vcsv` is generated in the result directory. It can be opened directly by using the Result Browser or from the Direct Plot Form by clicking *PLL Noise PSD*.

Figure 9-16 PLL macro-model test bench (integer-N)

Figure 9-17 PLL macro-model test bench (fractional-N)

Note: Be aware of the following information.

- Using the PSD plugin for phase noise PSD calculation, the command line is
  `spectre -plugin libpllMMpsd_sh.so`
- To obtain accurate locked PLL noise behavior, ensure that the `freq_meter` `outStart` parameter is large enough so that the PLL is well locked.
- The PSD plugin searches the `vco_freq` instance to get the related period measurement results. This name must not be changed.
- The single-sided PSD output offset frequency range is Δf_{min}~Δf_{max} with respect to f_0 which is the VCO locked output frequency, where $\Delta f_{max}=f_0/2$ and $\Delta f_{min}= 4/(t_{start}t_{stop})$. The t_{stop} is the transient analysis stop time and the t_{start} equals to 'outstart' value of the freq_meter. To obtain low offset frequency PLL noise behavior, the transient analysis stop time must be set to a large value.
- To simulate a fractional-N PLL, use either the `oscmm_frac` or `oscmm_frac_fast` symbols in the PLL test bench. The `div_ctrl<0:11>` can be connected to the delta-sigma modulator.

Sigma-Delta Modulator Macro-Model

There are three demo sigma-delta modulators macro-models, `SDM_MASH_3rd_111_macro`, `SDM_3rd_MASH_111` and `SDM_3rd_MASH_2-1` in `pllMMLib`. The latter two are constructed using veriloga-A modules, such as `z_inv_digi` (z^{-1}), `z_integrator_digi`($z^{-1}/(1-z^{-1})$), `one-z_inv`($1-z^{-1}$) and `quantizer_midtread_digi (quantizer)`. You can construct other type of sigma-delta modulators with these modules.)

The function and parameters are:

Module name: `z_inv_digi_`(z^{-1})

Function: one bit sample

pins (in out clk):

`in`	Signal input
`out`	Signal output
`clk`	Clock signal input

parameters:

`Vtrans`	Clock trigger threshold voltage.

Module name: `z_integrator_digi_`($z^{-1}/(1-z^{-1})$)

Function: one bit z integrator

pins (in out clk):

`in_Signal`	input
`out_Signal`	output
`clk`	Clock signal input

Module name: `quantizer_midtread_digi_(quantizer)`

Function: `quantizer`. The input signal will be quantized between `N*level_hi` and `N*level_lo`, the steps will be `N`. `[-N/2, N/2]` will be quantized as `0`.

pins (in out):

`in`	input
`out`	output

parameters:

`N`	The quantizer resolution is `1/N`. `N` must be a power of 2.
`level_hi`	Highest quantized value is `N*level_hi`.
`level_lo`	Lowest quantized value is `N*level_lo`.
`tr`	Risetime and falltime of the output.
`td`	Delay time from input to output.
`tt`	Time tolerance of transition event.

Module name: `SDM_MASH_3rd_111_macro`

Function: Mash 111 type 3rd SDM

pins (fset cp f f av):

`fset`	Fraction part is set by this pin's voltage which should be 0~1.
`cp`	Clock signal input.
`f`	Output of the sigma-delta. It's default value is in the range of -3~4.
`f av`	Average value of pin f.

parameters:

`BIT`	Set the ADC bit in the module.
`Vtrans`	CP trigger threshold.

`outOffset`	Output value of the sigma-delta is in the range of -3+outOffSet ~ 4+outOffset.

10

AnalogLib Components Used in RF Simulation

The PORT Element

You can use the *port* component, located in the *analogLib* library, in RF circuits for Virtuoso® Spectre® circuit simulator RF Analysis (SpectreRF) and Spectre S-parameter simulations.

The *port* component, located in the *analogLib* library, is a superset of the existing *psin* component. The *port* component supports all the source types of the Spectre *port* primitive: *pwl*, *pulse*, *sine*, *dc*, and *exp.* In addition it includes the *bit* and *prbs* source types. If you are using the `psin` component, it is highly recommended that you switch to the port component because the properties are organized in a better way, and the levels in volts peak and in dBm both cannot be set at the same time. This is easily accomplished by obtaining the properties form for `psin`, and changing the cell name to `port`. All the properties that are set in the `psin` component will retain their values when you make the change. The existing psin is retained for legacy designs and will not be enhanced in any way moving forward.

The *port* component is an independent resistive source tied between positive and negative terminals. It is equivalent to a voltage source in series with a resistor, where the reference resistance of the *port* is the value of the resistor.

Capabilities of the port Component

While the *port* component is most useful as a stimulus in high-frequency circuits, it also has the following unique capabilities.

- It can define the ports of a circuit to the S-parameter analysis.
- It has an intrinsic noise source that lets the noise analysis directly compute the noise figure of the circuit.
- Is the only source for which you can specify the amplitude in terms of power.
- It generates sinusoidal, exponential, piecewise-linear, periodic piecewise-linear, pulse, bit, and prbs waveforms for all the large-signal analyses.
- It can have different impedances for each harmonic number in the harmonic balance large-signal analysis.

Terminating the Port

Except for the `bit` and `prbs` waveforms, when you specify the voltage on a *port*, all Spectre simulations assume that the port is properly terminated in it's reference resistance. The specified voltage value is not the voltage on the internal voltage source, which is actually set to twice the value specified on the `port`. If you use a `port` source to drive an open circuit, the voltage (for DC, transient, AC, and PAC signals) is double its specified value. However, you can alternatively specify the amplitude of the sine wave in the transient and PAC analyses as the power in dBm delivered by the *port* when terminated with the reference resistance. For the `bit` and `prbs` source, the voltages specified for the zero and one states are used directly in the internal source, without multiplying by two.

The *port* component *Edit Object Properties* form is shown in Figure 10-1 on page 1437 and Figure 10-2 on page 1438.

Figure 10-1 Top of the port Component Edit Object Properties Form

Figure 10-2 Bottom of the port Component Edit Object Properties Form

Parameters for the Port Component

Port Parameters

Port parameters include *Resistance*, *Reactance*, *Port Number*, and *Multiplier*.

Resistance

The reference resistance of the system. The value must be a real number, but not 0. The default value is 50 ohms. This applies for all analyses. Note that the resistance and reactance can be set individually for each harmonic in harmonic balance large-signal analyses only. For more information, refer to Using the Harmonic Port with Harmonic Balance on page 1439.

Reactance

The imaginary part of impedance is used for harmonic balance large and small-signal analyses, and for the ac, noise, xf, sens, stb, pz, and sp.analyses. The reactance is not used for the dc, transient, and shooting-based large and small-signal analyses. The value must be a real number. The default is zero ohms. Note that the resistance and reactance can be set individually for each harmonic in harmonic balance large-signal analyses only. For more information, refer to Using the Harmonic Port with Harmonic Balance on page 1439.

Port number

The number associated with the *port*. The value must be a non-zero integer and must be unique for each *port* in a schematic. The *Port number* is not automatically indexed when you place a new *port* on your schematic. There is no need to assign this number because in the sp analysis, you just select the port instances you want from the schematic.

If you do assign port numbers, start with the number one, and number the ports sequentially. A gap in the port number sequence causes a simulation error if the port list is not defined in the S-parameter analyses.

Multiplier

Leave this property at the default. The multiplicity factor specifies the number of *ports* in parallel. The value must be a non-zero real number and the default is 1. For example, if you set *Resistance* to 50 and *Multiplier* to 2, you specify two *ports* in parallel, with an effective reference resistance of 25 ohms.

Using the Harmonic Port with Harmonic Balance

The port component has a feature called harmonic port that allows the specification of different impedances at different harmonic numbers. This is available for harmonic balance large-signal analyses only and does not work with any other analysis.

For this example, a 2.4GHz LNA is the example circuit.

The input and output ports are set up normally as shown in the properties forms below.

PORT1 is the input and PORT2 is the output.

One Input Frequency

The harmonic balance *Choosing Analyses* form has a single input applied.

To use the harmonic port, select *Harmonic Port* on the properties form. In this case, the port being used as a load in the circuit is selected. The *Edit harmonic port* window is displayed.

For this setup, a single input frequency is applied to the circuit. In this case, harmonic numbers 1, 2, 3, 4, and 5 exist in the simulation result. In the *Harmonic* column, you enter the harmonic number. In the *Resistance* column, you enter the resistance. In the *Reactance* column, you enter the series reactance in ohms. Positive reactance adds inductance, and negative reactance adds capacitance. Multiple entries are allowed, so

different impedances can be set for each harmonic. The resistance or reactance can be left blank, which is interpreted as zero resistance or reactance.

Note: When a variable name is used to set the value of resistance or reactance, that variable must be added to ADE manually. The *Variables - Copy From Cellview* menu option will not enter the variable into the variable table in ADE. To add a variable manually:

1. Double-click on the last variable in the variables list. The *Editing Design Variables* window is displayed.
2. In the *Editing Design Variables* window, click *Next*. The *Name* and *Value (expr)* fields become editable.
3. Enter the variable name in the *Name* field, and the starting value in the *Value (expr)* field.
4. Click *OK* or *Apply*. The variable and value are entered into the table.

Note: Whenever you have an expression for one of the values, you must enclose the expression in parentheses. If this is not done, the simulation error is generated:

```
ERROR: Length of hvec is not equal to length of rvec multilies tones or length of
rvec is not equal to length of xvec.
```

Here is a comparison where the third harmonic is set normally in the top trace, and to 1 ohm in series with +j1mohm in the bottom trace. The lower impedance for the third harmonic in the bottom trace lowers the amplitude of the third harmonic.

Two Input Frequencies

Now the harmonic balance *Choosing Analyses* form has two frequencies.

Also, the diamond cut is set with a very large *MaximOrder*. The reason for doing this is to see the harmonic numbers that correspond to the frequencies produced by the circuit. Each frequency has two harmonic numbers that indicate which harmonics are mixing with which to produce a specific output frequency. For example, take the 2.44GHz term. The harmonic number is 1 0. The frequency that is produced is 1*2.44G + 0*2.46G. Now look at the intermod at 2.42GHz. This index is 2 -1. The frequency is 2*2.44G - 1*2.46G. These are the harmonic numbers that can be set for the harmonic port. Wild cards (*) are not allowed in the harmonic number specification.

Note that if there were three input frequencies, there would be three numbers that specify every output frequency.

In this case, if the harmonic was specified as 3, there would be an error produced by spectre because there is a single index in the harmonic, but there are 2 indices for all the harmonics of the existing simulation. The number of entries in the harmonic field must match the number of input frequencies in the circuit.

The above example sets the impedance of the third order intermods to be much lower than 50 ohms. If the source is in series, setting the resistance and/or reactance to a large number can block harmonics you don't want. If the port is in parallel like this one, setting small resistance and reactance values can short out an undesired harmonic. This can be used for debugging the circuit, or for setting values to maximize power-added efficiency. The

resistance or reactance can be left blank, which is interpreted as zero resistance or reactance.

The top trace above is from the normal port, and the bottom trace is from the harmonic port. The third order harmonics are much smaller because they are essentially shorted out.

Source type

The *Source type* parameter lets you select a wave shape for the port from the *Source type* cyclic field: *dc*, *pulse*, *exp*, *pwl*, *sine, bit*, prbs, or blank. Each *Source type* has different parameter settings associated with it. You can define several different wave shapes and quickly switch between them without losing the wave shape settings. The wave shape settings are described in detail in the *DC*, *Pulse*, *Piecewise Linear*, *Sinusoidal*, *Exponential, bit,* and *prbs*Waveform Parameters sections.

The typical *Source types* used in SpectreRF analyses are *dc*, *piecewise linear, pulse*,and *sine*. For example, you can quickly switch from a sinusoid (for PSS analysis) to a DC level (for PAC analysis) by changing the *Source type* from *sine* to *dc*. The sinusoidal properties remain with the port, and can be changed when the *Source type* is returned to *sine*. This is true when switching between any of the sourcetypes. All the properties are retained for everything that you define.

When you set *Source type* to a blank value, the *port* acts as a resistive load.

DC Parameters

To generate a dc level from the *port* component, select *dc* in the *Source type* cyclic field. When the *Source type* is set to *dc*, the *dc* and *temperature effect* parameters are active.

The *dc* setting sets the DC level for all analyses.

DC voltage

The *DC voltage* parameter sets the port's DC level for all analysis. The value must be a real number. If you do not specify the DC value, the default value is 0 volts.

The *DC voltage* parameter specifies the DC voltage across the *port* when it is terminated in its reference resistance. In other words, the *DC voltage* of the internal voltage source is double the specified DC value, *dc*. Because all small signal analyses (AC, SP, XF, and Noise) use DC analysis results, the *DC voltage* level also affects the small-signal analyses.

Figure 10-3 Source type=dc in the Edit Object Properties form

The *Display small signal params, Display temperature params,* and *Display noise parameters* fields are discussed in Small-Signal Parameters on page 1539, "Temperature Effect Parameters" on page 1541, and "PRBS Waveform Parameters" on page 1526.

Pulse Waveform Parameters

The pulse waveform can generate a step, a single pulse, or a periodic pulse waveform.

To generate a pulse waveform from the *port* component, select *pulse* in the *Source type* cyclic field.

When you specify the voltage on a *port*, you are specifying the voltage when the port is properly terminated, and not the voltage on the internal voltage source. Therefore, the voltage on the internal source is set to twice the value specified on the *port*.

Figure 10-4 Source type=pulse in the Edit Object Properties form

Frequency name 1

The *Frequency name 1* parameter is listed in the *Fundamental Tones* section of the *Choosing Analyses* form for pss, qpss, and hb when *Names* is selected.

Important

If you have two different sources at the same frequency or at integer multiple of the same frequency, make sure that you use the same frequency name 1 or 2 property for both sources. An example might be RF1.

Delay time

The *Delay time* parameter is the simulation time that the port stays at the zero value without becoming periodic. Generally, this should be set less than the period of the waveform. Longer times are allowed, however, this forces a transient analysis until the longest delay time before the large-signal analyses pss, qpss, and hb can start. If a 1GHz signal and a delay time of 1 second is in your circuit, a transient is forced for 1 billion cycles of the input before the pss, qpss, or hb can start. This effectively prevents them from starting because of the long simulation time in the transient analysis. This must be a real number.

Zero value

The *Zero value* property is the time-zero value of the pulse in Volts. The default is zero volts.

One value

The *One value* property specifies the voltage that the signal transitions to at time equals zero plus. The default is one volt, which is unlikely to be what you want. Always set this property.

Period of waveform

The *period* property specifies the period of the pulse. If this property is left blank, a single pulse is generated at the beginning of the transient simulation.

Rise time

The *Rise time* property specifies the zero to 100% risetime for the pulse. This is the time for the complete transition from the zero value to the one value.

Fall time

The *Fall time* property specifies the zero to 100% falltime for the pulse. This is the time for the complete transition from the one value to the zero value.

Pulse width

The *Pulse width* property specifies the time to remain at the one value. The default is infinity.

Pulse Waveform Examples

Generating a step from zero to 1.2 Volts

1. Set the zero value to 0 volts.
2. Set the one value to 1.2 volts.

3. Set the risetime to a time that is appropriate for your circuit. A suggested starting value is about $\frac{1}{10 \times CircuitBandwidth}$.

   The default is the falltime. If the falltime is not given, the default is $\frac{Period}{100}$. If the period is not given, then the default is $\frac{SimulationTime}{100}$.

   The waveform is shown below.

4. Setting a delay time to 1n shifts the waveform to the right.

The waveform is shown below.

Specifying a startup bump in a supply voltage

This is commonly used to start oscillators, and is usually done using a vsource.

1. Set the zero value to about 80% of the supply voltage.
2. Set the one value to the supply voltage.

3. Set the risetime as appropriate for the system you are simulating.

4. The waveform is shown below.

Specifying a single pulse at startup, then zero after that.

This is usually used with the ipulse or isource components to start an oscillator.

1. Set the zero value to zero.
2. Set the one value large enough to get things started.
3. Set the rise and fall times to about one tenth of the period of the oscillator.

4. Set the pulse width to about three tenths of the period.
5. Leave the period field blank.
6. The waveform is shown below.

To generate a pulse with an arbitrary duty cycle and differing rise and fall times

1. Set the zero and one values to the levels needed for your system.
2. Set the rise and fall times to approximate the real input source. The values can be different.
3. Set the period to the value needed for your frequency.

4. Set the pulse width between 0.05 to about 0.95 times the period.

5. The waveform is shown below.

To generate a 50% duty cycle square wave

1. Set the zero and one values as appropriate for your system.
2. Set the rise and fall times as appropriate for your system and make them equal.
3. Set the period to that needed to generate your input frequency.

4. Set the pulse width to half the period minus the risetime.

5. The waveform is shown below.

PWL Waveform Parameters

Piecewise linear waveforms allow an arbitrary input waveform to be generated. The input can either be a file that contains time and voltage pairs, or you can enter the time-voltage pairs directly in the PWL source properties form. Remember that the voltages you enter in the piecewise linear file assume that the port is properly terminated. The internal voltage source gets set to double the value specified in the piecewise linear voltage specifications.

Important

If the period property is set, the source repeats the waveform between zero and period over and over again. If the period property is not set and the transient analysis goes past the last point in the PWL file, the last voltage value in the file is held until the end of the simulation. If the period property is not set, the signal does not become periodic until after the time of the last point in the PWL file, where the source output becomes a DC level. If the file has a stopping point that has a large time, then

the periodic analyses pss, qpss, and hb run a transient until the longest time in any input PWL file has been reached before they go into the steady-state analyses, and these analyses use the DC level that is present at the end of the PWL file as the input signal.

Important

If you want a periodic piecewise linear waveform for pss, qpss, and hb, make sure that the period property is set.

To generate a piecewise linear waveform from the *port* component, select *pwl* in the *Source type* cyclic field. This sets the *Source type* CDF parameter to *pwl* and displays additional fields for the PWL CDF parameter settings.

Important

The advantage of entering a square or pulse wave using the pwl source is that the starting value can be at the center of the waveform. In the pulse waveform, the signal starts at the zero value, and because of this, in some cases, the tstab time must be extended to allow the waveform to settle. Starting the waveform in the middle can drastically reduce tstab and/or dramatically improve convergence for systems that have square or pulse waves and relatively long settling times for the nodes with the square or pulse wave signal.

When you specify the voltage on a *port*, you are specifying the voltage when the port is properly terminated, and not the voltage on the internal voltage source. Therefore, the voltage on the internal source is set to twice the value specified on the *port*.

Figure 10-5 Source type=pwl in the Edit Object Properties form

Waveform Entry Method

With the *Waveform Entry Method* buttons, select how you enter piecewise-linear data,

- By specifying a *File name*.
- By entering a series of *Voltage/Time points* directly in the properties list for the port.

File name

When you select *File* as the *Waveform Entry Method*, type the name of the file containing your piece wise-linear data in the *File name* field. The *File* name must be a string. There is no default. In ADE, define the directory that contains the file by selecting *Setup - Simulation Files*, and entering either the rooted path (starts with slash (/)) or relative path (starts with dot (.)) in the *Include Paths* field. Multiple directories can be specified in the *Include Paths* fields.

In your file, list the piecewise-linear data in the form of time-value pairs. Enter one pair per line with a space or tab between the time and voltage values. The numbers in the file must use exponential notation, for example, 1.04e-3. You cannot use SI scale factors (p, n, u, m, k, M, G, etc.).

Number of PWL/Time pairs

When you select *Voltage/Time points* as the *Waveform Entry Method*, the *Number of PWL-Time pairs* field (tvpairs) opens. Enter the number of time-value pairs you plan to enter. The form expands to let you enter the designated number of *Time* and *Voltage* values. Units are seconds and volts. The default is 0 and the maximum value is 50.

In this example, the number of voltage-time pairs is 6.

Figure 10-6 Waveform Entry Method=Voltage/Time points

The waveform that is produced is shown below.

The *DC offset, Amplitude scale factor,* and *Time scale factor* parameter fields let you quickly adjust the amplitude, frequency, and offset of your piecewise-linear data pairs without editing each individual time-value pair in the *PWL* waveform.

DC offset

DC offset for the *PWL* waveform. The default is zero volts.

Important

Note that for PWL waveforms, the DC Voltage term in the port only applies for the DC based analyses and not for the PWL waveform itself. To shift the DC level of the PWL waveform in the large-signal analyses (tran, envlp, pss, qpss, and hb) use the

DC offset property. When the DC voltage property and the DC offset property are both set, the actual offset value used in the large signal analyses is defined by the DC offset property by itself, and not the DC voltage plus the DC offset property.

To show the effect of setting both the DC voltage and DC offset properties, in the following example, the DC Voltage and the DC offset are both set to 0.1.

Note that the signal only observes the DC offset in the transient.

Amplitude scale factor

Amplitude scale factor (*scale*) for the *PWL* waveform. The default value is one. The example below shows a scale factor of 3.

Note that the waveform is multiplied by three in amplitude.

Time scale factor

Time scale factor is a multiplier for the times given for the *PWL* waveform. The default is 1. Setting the time scale factor greater than 1 increases the times in the file, and reduces the frequency.

Note that setting the time scale factor to 0.5 multiplies all the times including the time in the period property.

Cosine Filter

The raised cosine filter is used when the PWL source is used to define digital modulation. Values are `none` (default), and `nrc`. `nrc` specifies a raised cosine filter to be used on the output waveform of the PWL source. This can be used to lower the numerical noise floor of

the modulation specified in the I and Q modulating file. When `nrc` is selected, the PWL file must have evenly spaced points with timepoints at 1/(2*Bandwidth of the filter).

Rolloff factor

When `nrc` is selected, the *Rolloff factor* (*rolloff*) of the filter can be specified. The default value is `0.2`, which is reasonable for simulations. If you consider a continuous waveform, changing the rolloff factor has an effect on the ringing that is produced in the time domain. Smaller rolloff factors produce more ringing in the continuous waveform. Envelope does not simulate the continuous waveform. Instead, envelope samples the waveform at the time interval in the I and Q modulation file. This delta-time in the I and Q file is always related to the modulation frequency, and it also determines the bandwidth of the cosine filter as described below. The effect of this is that the sampling occurs at the times where the ringing of the filter is zero. Therefore, the cosine filter does not change the spectral content calculated in envelope. It just reduces the numerical noise floor of the input signal.

Bandwidth

The *Bandwidth* (*pwlbandwidth*) of the Cosine Filter. If you want to only allow in-band modulation, this is the channel bandwidth. If you want to allow an accurate ACPR measurement, set the bandwidth to three times the channel bandwidth. If you want to measure two adjacent channels for ACPR, set the bandwidth to five times the channel bandwidth. Note that the I and Q data must have properly spaced time intervals of $\frac{1}{2 \times Bandwidth}$ for this capability to be used.

Note: The *Rolloff factor* and *Bandwidth* options are available only when `nrc` is selected in the *Cosine Filter* drop-down list.

Desired rms value

Desired rms voltage for the PWL waveform. Note that this will be set based on the datapoints in the PWL file or in the manually defined points entered directly into the properties form only. This has implications for periodic piecewise linear or if you go beyond the last point in the file in the transient analysis.

If the signal is defined for only part of the period or if you go beyond the last point in the pwl description, the rms calculation will not take into account the fact that the signal remains at the last voltage after the last timepoint in the pwl file. If you want this to be taken into account, define a point in the pwl description at the stop time of the transient or at the end of the period.

Consider an exmple in which a 1GHz signal is being generated, with points for the first period of 1GHz defined. No points after 1nsec exist in the PWL description. Also note that the desired RMS value is set to 1.5 (volts).

Now the transient result is plotted.

Vertical markers have been placed at the midpoint of the high and low states. The voltage is slightly greater than 1.5 because of the finite rise and fall times in the PWL description.

Now, the period is set to 2nsec. This will make the waveform periodic at 500MHz.

Now the transient result is plotted. Note that the high and low values are the same as before.

Now the transient waveform is sent to the calculator, and the actual rms value is calculated.

Instead of producing 1.5 volts as specified, only 1.061 volts is produced.

Now add another datapoint in the PWL description at 2nsec, and zero volts. Now the period and the time interval that defines the PWL points exactly agree.

Now the transient result is plotted.

Note that the positive and negative voltages have changed. This is because the PWL description completely defines the waveform during the entire first period of the waveform.

Now the signal is set to the calculator, and the rms is calculated. It is appended to the last table.

Now the RMS value is exactly 1.5 volts.

Now leave the last timepoint at 2nsec, and reduce the period to 1nsec.

Now, the transient result is plotted. The positive and negative voltages still reflect the rms value for the 2nsec interval defined by the timepoints themselves. It has not truncated the

timepoints to just the first period. It is also not periodic. When the period is smaller than the time of the last timepoint in the PWL description, Spectre assumes that the period is in error, and ignores the property.

Tip

If you need to set an rms voltage for a periodic piecewise linear signal, make sure that the timepoints in the piecewise linear description define timepoints over exactly the entire first period of the waveform.

Do not set *Desired rms value* and *Power of PWL waveform* at the same time. If both are specified, first the amplitude is calculated based on the Power of PWL waveform property, and then that amplitude is multiplied by the *Desired rms value* setting.

Delay time

The *Delay time* parameter sets the time that the signal remains at the first point in the file before the pwl points begin to take effect. Generally, this should be set less than the period of the waveform. Longer times are allowed, however, this forces a transient analysis until the longest delay time before the periodic large-signal analyses pss, qpss, and hb can start. If a 1GHz signal and a delay time of 1 second is in your circuit, a transient is forced for 1 billion cycles of the input before the pss, qpss, or hb can start. This effectively prevents them from starting because of the long simulation time in the transient analysis.

Breakpoints

Possible values are *no*, *yes*, or blank. If you set *Breakpoints* to *yes*, you force SpectreRF to place time points at each point specified in a *PWL* waveform during a transient analysis. This can be very expensive for waveforms with many points. If you set *Breakpoints* to *no*, SpectreRF inspects the waveform, looking for abrupt changes, and forces time points only at those changes. If you set *Source type = pwl* and set *Breakpoints* to blank, the default is *yes* if the number of points you specify is less than 20.

Period

The *PWL* waveform is periodic if you specify *Period* in seconds.

If the value of the waveform you specify is not exactly the same at both its beginning and its end, then you must provide a non-zero value for *Transition Width*.

Power of PWL waveform

This is the power in dBm for a 50 ohm resistor for the output of the PWL source. See Desired rms value on page 1478 for important information about how to define timepoints in the PWL input points. Do not set *Desired rms value* and *Power of PWL waveform* at the same time. If both are specified, first the amplitude is calculated based on the *Power of PWL waveform* property, and then that amplitude is multiplied by the *Desired rms value* setting.

Transition Width

Transition width (*twidth*) is used when making *PWL* waveforms periodic and the ending value of the PWL file does not equal the beginning value. The default is the `PWL period/1000`.

Before repeating, the waveform changes linearly in an interval of *Transition Width* from its value at *(Period – Transition Width)* to its value at the beginning of the waveform. Thus, the *Transition Width* must always be much less than the *Period*.

Sinusoidal Waveform Parameters

The *port* component can generate up to two sinusoids simultaneously. They are denoted as 1 and 2. You can set the amplitude, frequency, and phase for both individually. The amplitude can be set to either a voltage or a power level. When you set a power level, the assumption is that the port is perfectly matched. The source that is internal to the port gets double the amplitude specified by the power in dBm.You can also specify sinusoidal AM or FM modulation of sinusoid 1. Sinusoid 2 cannot be modulated.

To generate sinusoidal waveforms, set the *Source type* in the properties list to *sine*.

The first sinusoid is described by the parameters *Frequency name 1, Frequency 1, Amplitude 1 (Vpk)*, *Amplitude 1 (dBm)*, *Phase for Sinusoid 1*, *Sine DC level*, and by AM or FM modulation terms.

Figure 10-7 Source type=sine in the Edit Object Properties form

Frequency name 1

This property names the fundamental tones of sinusoid 1. After you save the schematic, the names you assign appear in the *Fundamental Tones* list box on the *Choosing Analyses* form.

Important

When you have multiple ports that are connected to the circuit at different places, but have the same frequency, or integer multiples of the same frequency, make sure that the *Frequency name 1* property is the same for every instance. An example might be RF1. For harmonic balance, when names is selected in the *Choosing Analyses* form, and for qpss, the frequency name property specifies that all the signals with the same name should be treated as one tone in the simulation. This can save a lot of time when you run those analyses.

Frequency 1 is the frequency of the first sinusoidal waveform (carrier frequency). You typically use unmodulated signals in SpectreRF analyses. The value must be a real number. The default frequency is zero, which causes no signal to be generated. Always set this property.

Amplitude 1 (Vpk)

The peak amplitude of the first sinusoidal waveform that you generate. The value specified is the voltage delivered into a matched load. You can select either *Amplitude 1 (Vpk)* or *Amplitude 1 (dBm)*, but not both. If *Amplitude 1 (Vpk)* has a value, the *Amplitude 1 (dBm)* field is grayed out. The value must be a real number. The default is 1 volt.

When you specify the voltage on a *port*, you are specifying the voltage when the port is properly terminated, and not the voltage on the internal voltage source. Thus, the voltage on the internal source is set to twice the value specified on the *port*.

Amplitude 1 (dBm)

Amplitude 1 (dBm) is the amplitude of the first sinusoidal waveform, in dBm.The resistance for the dBm calculation is taken from the resistance of the port. The value specified is the power delivered into a matched load. You can select either *Amplitude 1 (Vpk)* or *Amplitude 1 (dBm)*, but not both. If *Amplitude 1 (dBm)* has a value, the *Amplitude 1 (Vpk)* field is grayed out. The value must be a real number.

Phase for Sinusoid 1

The phase at the specified *Delay time*. To achieve a specified phase and still remain continuous, the sinusoidal waveform might start before the given *Delay time*. For example, if you want to generate a cosine wave, set this parameter to 90°. The value must be a real number. The default is 0 degrees.

Important

For circuits that have long settling times for the nodes connected to the sinusoidal signal, it can save a lot of time and/or dramatically improve convergence if you set phases other than zero or 180 degrees by using the delay time parameter instead of the *Phase* parameter. Imagine a need to start at a 90 degree phase shift. If the *Phase for Sinusoid 1* property is set to 90, the waveform that is produced starts at the positive peak, If instead, the delay time was set to 3/4ths of the period and the *Phase for Sinusoid 1* property was set to zero, instead of starting at the peak and needing to settle to near the middle value in the tstab interval, the signal itself starts at the center value, so the circuit has to settle a much smaller amount.

Sine DC level

Sets the DC level in volts for sinusoidal waveforms in large-signal analyses. This parameter is used when the sinusoid has a different average level than the one specified for the DC analyses. If not specified, the average value of the sinusoid is the same as that of the DC level of the source.

Important

When this property is specified, it is used directly. The DC level of the sinusoid is not the DC voltage plus the Sine DC level. It is just the Sine DC level.

Modulation Parameters

Display Modulation Parameters

Important

The check box is a display feature in the form only. If there are properties that are set and the display check box is not selected, although you will not see the parameters in the form, these parameters will still be netlisted, and will still have an effect on the simulation.

When selected, the form expands and the following modulation parameters are displayed: *FM modulation index, FM modulation freq, AM modulation index, AM modulation freq, AM modulation phase,* and *Damping factor*.

Important

These parameters only affect the sinusoidal waveform.

Important

Only the first `sinusoid` can be modulated.

Figure 10-8 Display modulation params

AM Modulation (Background Information)

The amplitude modulation for the sinusoidal case is defined as

$$V(t) = (A \bullet \sin(\omega_C t)) + \frac{AM}{2}[\sin((\omega_C + \omega_{Mod})t + \phi) + \sin((\omega_C - \omega_{Mod})t - \phi)]$$

where

- $V(t)$ is the amplitude modulation waveform
- ω_C is the carrier frequency
- AM is the AM modulation index
- ω_{Mod} is the modulation frequency
- ϕ is the *AM modulation phase*

The amplitude modulation parameters affect only the first sinusoid generated by *port*. They have no effect on the second sinusoid.

AM modulation Index

This property specifies the amplitude of the modulation signal. When the AM modulation index is 1.0, the signal is 100% modulated. When the AM modulation index is 0.5, the signal is 50% modulated.

AM Modulation Frequency

This property specifies the frequency of the sine wave that is used to modulate the signal that is specified in the *Frequency 1* property. The AM modulation frequency should be much lower than the *Frequency1* property.

AM Modulation Phase

This specifies the initial phase of the AM modulation sinusoid. The default is zero degrees.

The *Edit Object Properties* form for a 100% modulated signal is shown below.

Cell Name
port
View Name
symbol
Instance Name
PORT0qq
Add
Delete
Modify
User Property
Master Value
Local Value
Display
IvsIgnore
TRUE
CDF Parameter
Value
Display
Resistance
50 Ohms
Reactance
Port number
DC voltage
Source type
sine
Frequency name 1
RF1
Frequency 1
5M Hz
Amplitude 1 (Vpk)
1 V
Amplitude 1 (dBm)
Phase for Sinusoid 1
Sine DC level
Delay time
Damping factor
Display second sinusoid
Display multi sinusoid
Display modulation params
AM modulation index
1
AM modulation freq
50K
AM modulation phase
FM modulation index
FM modulation freq
Number of FM Files
none
one
two
IQ modulation I File
IQ modulation Q File
Amplitude scale factor
Power of PWL waveform
Cosine Filter
Display small signal params
Display temperature params
off

The output waveform is shown below.

FM Modulation (Background Information)

The frequency modulation for the sinusoidal case is defined as

$V_{FM}(t) = A \sin(2\pi f_c t + \beta \sin(2\pi f_m t) + \phi)$

where

- `A` is the amplitude of sinusoid 1
- β is the FM modulation index
- $\sin(2\pi f_m t)$ is the modulation signal
- f_c is the carrier frequency
- ϕ is the *phase for sinusoid 1*

The frequency modulation parameters affect only the first sinusoid generated by *port*. They have no effect on the second sinusoid.

FM modulation frequency

FM modulation frequency for the sinusoidal waveform (`fm` in the previous equation).

FM modulation index

FM index of modulation for the sinusoidal waveform, the ratio of peak frequency deviation divided by the center frequency (β in the above equations).

$\beta = \Delta f / f_m$

Important

The value is in percent, with a default of zero percent.

The property list for an FM signal with a modulation index of 100% is show below.

Browse
Reset Instance Labels Display
Property
Value
Display
Library Name
analogLib
Cell Name
port
View Name
symbol
Instance Name
PORT0qq
off
Add
Delete
Modify
User Property
Master Value
Local Value
Display
IvsIgnore
TRUE
CDF Parameter
Value
Display
Resistance
50 Ohms
Reactance
Port number
DC voltage
1 V
Source type
sine
Frequency name 1
RF1
Frequency 1
1M Hz
Amplitude 1 (Vpk)
1 V
Amplitude 1 (dBm)
Phase for Sinusoid 1
Sine DC level
Delay time
Damping factor
Display second sinusoid
Display multi sinusoid
Display modulation params
AM modulation index
AM modulation freq
AM modulation phase
FM modulation index
100
FM modulation freq
50K
Number of FM Files
none
one
two
IQ modulation I File
IQ modulation Q File
Amplitude scale factor

The transient waveform goes from zero frequency to twice the center frequency. This is an extremely large modulation index.

Number of FM Files

The first sinusoid can also be phase modulated with a pair of PWL files that define the desired modulation.

- *none* specifies that the FM signal should be modulated with a sinusoid.

  In this case, the FM signal from the source will be represented as:

  $V_{FM}(t) = A \sin(2\pi f_c t + \beta \sin(2\pi f_m t))$

- When *one* is selected, a *Name of FM File1* field appears. Provide the filename of the PWL file that is to be used for the frequency modulation.

In this case, the FM signal from the source will be represented as:

$$V_{FM}(t) = A \sin(2\pi f_c t + \beta \int_0^t \delta\omega(\tau) d\tau)$$

This is shown in the Edit Object Properties form below.

- When *two* is selected, the *Name of FM File1 and Name of FM file2* fields appear. Provide the filename of the PWL file that is to be used for the phase modulation.

  In this case, the FM signal from the source will be represented as:

  $$V_{FM}(t) = A(t)\sin(2\pi f_c t + \beta\theta(t))$$

  Here,

  β is the frequency modulation index

  $$A(t) = \sqrt{(I(t))^2 + (Q(t))^2}$$

  where I(t) is the waveform specified in the Name of FM File1 pwl file, and Q(t) is

  $$\theta(t) = \arctan\left(\frac{Q(t)}{I(t)}\right)$$

waveform specified in the Name of FM File2 pwl file. This is shown below.

Browse
Reset Instance Labels Display
Property
Value
Display
Library Name
analogLib
Cell Name
port
View Name
symbol
Instance Name
PORT0qq
Add
Delete
Modify
User Property
Master Value
Local Value
Display
IvsIgnore
TRUE
CDF Parameter
Value
Display
Resistance
50 Ohms
Reactance
Port number
DC voltage
1 V
Source type
sine
Frequency name 1
RF1
Frequency 1
5M Hz
Amplitude 1 (Vpk)
1 V
Amplitude 1 (dBm)
Phase for Sinusoid 1
Sine DC level
Delay time
Damping factor
Display second sinusoid
Display multi sinusoid
Display modulation params
AM modulation index
AM modulation freq
AM modulation phase
FM modulation index
100
FM modulation freq
Number of FM Files
none
one
two
Name of FM File1
Zero_Offset.in.spectre
Name of FM File2
3u_Offset.in.spectre
IQ modulation I File
off

Modulating a carrier with a digital modulation I and Q file

This capability provides exactly the same capability as the *ACPR_Source* in rfLib. The I and Q PWL files provide the information to be modulated, and the source acts as an ideal modulator.

IQ modulation I File

This parameter should be set to the filename to the PWL file that contains the digital modulation 'I' information. This is a file with 2 columns. The left column is the time, and the right column is the voltage. Usually, the file is generated with a constant delta-T from entry to entry in the file. Generally, this file is provided by the system designer.

IQ modulation Q File

This parameter should be set to the filename to the PWL file that contains the digital modulation 'Q' information.This is a file with 2 columns. The left column is the time, and the right column is the voltage. Usually, the file is generated with a constant delta-T from entry to entry in the file. Generally, this file is provided by the system designer.

Cosine Filter

The raised cosine filter is used when the PWL source is used to define digital modulation. Values are `none` (default), and `nrc`. `nrc` specifies a raised cosine filter to be used on the output waveform of the PWL source. This can be used to lower the numerical noise floor of the modulation specified in the I and Q modulating file. When `nrc` is selected, the PWL file must have evenly spaced points with timepoints at 1/(2*Bandwidth of the filter).

Rolloff factor

When `nrc` is selected, the *Rolloff factor* (*rolloff*) of the filter can be specified. The default value is `0.2`, which is reasonable for simulations. If you consider a continuous waveform, changing the rolloff factor has an effect on the ringing that is produced in the time domain. Smaller rolloff factors produce more ringing in the continuous waveform. Envelope does not simulate the continuous waveform. Instead, envelope samples the waveform at the time interval in the I and Q modulation file. This delta-Time in the I and Q file is always related to the modulation frequency, and it also determines the bandwidth of the cosine filter as described below. The effect of this is that the sampling occurs at the times where the ringing of the filter is zero, thus the cosine filter does not change the spectral content calculated in envelope. It just reduces the numerical noise floor of the input signal.

Bandwidth

The *Bandwidth* (*pwlbandwidth*) of the Cosine Filter. If you want to only allow in-band modulation, this is the channel bandwidth. If you want to allow an accurate ACPR measurement, set the bandwidth to three times the channel bandwidth. If you want to measure two adjacent channels for ACPR, set the bandwidth to five times the channel bandwidth. Note that the I and Q data must have properly spaced time intervals of $\frac{1}{2 \times Bandwidth}$ for this capability to be used.

Note: The *Rolloff factor* and *Bandwidth* options are available only when `nrc` is selected in the *Cosine Filter* drop-down list box.

The *Rolloff factor* and *Bandwidth* options are available only when `nrc` is selected in the Cosine Filter.

Damping factor

Damping factor for the sinusoidal waveform. *Damping factor* specifies the time it takes to go from the envelope (full amplitude) at *time*=0 to 63 percent of the full amplitude. For example, consider the following damped sinusoid:

$$v(t) = A\, e^{-\sigma t} \sin(2\pi f t + \phi)$$

where

σ = *Damping factor*, `A` is the amplitude of sinusoid 1, and ϕ is the *Phase for sinusoid 1*

- If $\sigma = 0$, the waveform is a pure sinusoid (steady state).
- If $\sigma < 0$, the waveform exhibits decaying oscillations.
- If $\sigma > 0$, the waveform exhibits growing oscillations.
- It takes 5σ to diminish to 1 percent of the peak amplitude. The value must be a real number. Default: 0. Units: 1/seconds

The following figure shows the effect of *Damping factor* on the first sinusoid for three values of σ.

Figure 10-9 Effect of Damping Factor 1 on the First Sinusoid

Display second sinusoid

Important

The check box is a display feature in the form only. If there are properties that are set and the display check box is not selected, although you will not see the parameters in the form, these parameters will still be netlisted, and will still have an effect on the simulation.

Displays the CDF parameters for the second sinusoid in the *Edit Object Properties* and *Add Instance* forms. When selected, the form expands to show the following CDF parameters: *Frequency name 2, Frequency 2, Amplitude 2 (Vpk), Amplitude 2 (dBm),* and *Phase for Sinusoid 2*.

Important

The second sinusoid cannot be modulated.

Figure 10-10 Display second sinusoid

Frequency name 2

Name for the second sinusoid. After you save the schematic, the name you assign is listed in the *Fundamental Tones* section of the *Choosing Analyses* form for pss, qpss, and hb when names is selected.

Important

If you have two different sources at the same frequency or at integer multiple of the same frequency, make sure that you use the same frequency name 1 or 2 property for both sources. An example might be RF1.

Frequency 2 is the frequency of the second sinusoidal waveform in Hertz. There is no default value.

Amplitude 2 (Vpk)

Peak amplitude of the second sinusoidal waveform. The value specified is the voltage delivered into a matched load. You can select either *Amplitude 2 (Vpk)* or *Amplitude 2 (dBm)*, but not both. If *Amplitude 2 (Vpk)* has a value, the *Amplitude 2 (dBm)* field is grayed out. The value must be a real number. The default is 1 volt.

When you specify the voltage on a *port*, you are specifying the voltage when the port is properly terminated, and not the voltage on the internal voltage source. Thus, the voltage on the internal source is set to twice the value specified on the *port*.

Amplitude 2 (dBm)

Amplitude 2 (dBm) is the amplitude of the second sinusoidal waveform, in dBm. The resistance for the dBm calculation is taken from the resistance parameter of the port. the value specified is the power delivered into a matched load. You can select either *Amplitude*

2 (Vpk) or *Amplitude 2 (dBm),* but not both. If *Amplitude 2 (dBm)* has a value, the *Amplitude 2 (Vpk)* field is grayed out. The value must be a real number. Units: dBm

Phase for Sinusoid 2

The phase at the specified *Delay time* for the second sinusoid. To achieve specified phase while still remaining continuous, the sinusoidal waveform might start before the given *Delay time*. The value must be a real number. Default: 0 Units: degrees.

Display multi sinusoid

Important

The check box is a display feature in the form only. If there are properties that are set and the display check box is not selected, although you will not see the parameters in the form, these parameters will still be netlisted, and will still have an effect on the simulation.

Multi sinusoid is a way of generating up to nine unmodulated signals at the same time from the port component. It works in all the large-signal analyses, and is usually used with harmonic balance or qpss-harmonic balance for the simulation of systems with multiple carriers.

When *Display multi sinusoid* is selected, the properties form expands to show more properties. The *Number of Frequencies* field specifies the number of carriers you want to generate. Up to nine carriers can be generated by the port component. For each one of the frequencies, a set of parameters patterned after: *Sinusoid Frequency 1, Sinusoid Ampl 1 (Vpk), Sinusoid Ampl 1 (dBm),* S*inusoid Phase 1, Sinusoid Maxharm 1* is displayed for each tone you want to generate.

When *Display multi sinusoid* is selected, the frequency specified in the *Frequency1* parameter should be the middle frequency of the group of frequencies you want to generate. It must be one of the frequencies that could actually be generated by the port component. If four frequencies are to be generated, then the *Frequency1* property could be either the second or third frequency in the series.

Consider an example where 2.43GHz, 2.44GHz, and 2.46GHz need to be generated for a composite triple beat measurement. The actual center frequency is 2.445GHz. Since this is not one of the frequencies in the series that is actually generated, it cannot be picked. Pick one of the frequencies that could be generated on either side of the actual center frequency. Here, the center frequency could be either 2.44GHz or 2.45GHz. Even though 2.45GHz is not

being generated in this example, it is one of the frequencies that could be generated by the series, and thus it is eligible to be selected.

The frequency specified in the *Frequency 2* field should be the spacing frequency. All the frequencies specified in the multi-frequency section need to be separated by the spacing frequency. In the example above, the spacing frequency is 10MHz.

Up to nine frequencies can be generated in one port. When this feature is used with HB or QPSS, only specify the *Frequency1* and *Frequency2* frequencies in the frequencies section. The number of harmonics should be set to about two times the total number of tones that are generated in the multi-frequency section.

Figure 10-11 Display multi sinusoid

Number of Frequencies

Number of sinusoid frequencies to be specified.

Sinusoid Frequency 1

The frequency of the first sinusoidal waveform (carrier frequency) in Hertz.

Sinusoid Ampl 1 (Vpk)

The peak amplitude of the sinusoidal waveform that you generate in volts. When you specify the voltage on a *port*, you are specifying the voltage when the port is properly terminated, and not the voltage on the internal voltage source. Thus, the voltage on the internal source is set to twice the value specified on the *port*.

Sinusoid Ampl 1 (dBm)

The amplitude of the first sinusoidal waveform when specified in dBm.

Sinusoid Phase 1

The phase at the specified delay time. To achieve a specified phase and still remain continuous, the sinusoidal waveform might start before the given delay time. For example, to generate a cosine wave, set this parameter to pi/2. The units are radians.

Sinusoid Maxharm 1

Do not set this parameter. Harmonics are usually set in the hb or qpss *Choosing Analyses* form.

Below is a plot of the output when the variable prf is set to -40 (dBm).

The amplitudes of the tones at 2.4G, 2.41G, 2.42G and so on are at the numerical noise floor of the simulation.

When these three frequencies are applied to a real system, the nonlinearity of the system produces mixing products. For example, 2.43G mixes with 2.44G to produce 10M. 2.44G mixes with 2.46G to produce 20M. 2.43G mixes with 2.46G to produce 30M. 2.43G, 2.44G, and 2.46G are first order terms, so 10M, 20M, and 30M are second order terms.

Now the 10M, 20M, and 30M terms can mix with 2.43G, 2.44G, and 2.46G. For example, 2.44G mixes with 10M to produce 2.45G. 2.44G is a first order term, and 10M is a second order term, so 2.45G is a third order term. 2.43G can mix with 20M to add an additional third

order term at 2.45G. This leads to many harmonics that are actually produced, as shown below. The vertical axes have the same scale.

Exponential Waveform Parameters

To generate an exponential waveform from the `port` component, set the CDF parameter *Source type=exp*, as shown in the next figure.

Figure 10-12 Source type=exp in the Edit Object Properties form

The exponential waveform can generate one exponential pulse, and cannot generate a periodic signal. This it is not usable with SpectreRF.

When you specify the voltage on a *port*, you are specifying the voltage when the port is properly terminated, and not the voltage on the internal voltage source. Thus, the voltage on the internal source is set to twice the value specified on the *port*.

Zero value

The *Zero value* property is the time-zero value of the exponential waveform in volts.

One value

The *One value* property specifies the voltage that the signal transitions to at time equals zero plus. The default is one volt, which is unlikely to be what you want. Always set this property.

Rise time start

This property specifies the simulation time in seconds to begin the transition from the zero to the one value. The default is zero seconds.

Rise time constant

The *Rise time constant* specifies the time for the exponential waveform to reach 63 percent of the way from the zero value to the one value. The default is 0.075 percent of the stop time in the transient analysis.

Fall time start

The *Fall time start* specifies the simulation time to begin the transition from the one state to the zero state. The default is one percent of the transient analysis stop time.

Fall time constant

The *Fall time constant* specifies the simulation time for the exponential waveform to reach 63 percent of the way from the one value to the zero value. The default is one percent of the stop time in the transient analysis.

Bit Waveform Parameters

To generate bit sequence/string from the port component, select bit in the *Source Type* cyclic field. The bit source has four states: `1`, `0`, `m`, and `z`, which represent the high, low, middle voltage/current and high impedance state respectively. It allows patterns defining a sequence of bits. When the *m* state is specified, the output voltage is set halfway between the zero and one voltage.

Figure 10-13 Source type=bit in the Edit Object Properties form

Delay Time

The *Delay time* property specifies the time the first bit in the string should be held before the first bit is generated.

Note: When the delay time property is set, the time of the first transition is at the delay time plus the period of the first bit, which is set by the *Period of waveform* property.

Zero Value

The *Zero value* property sets the voltage when the pattern specifies a zero state. The pattern will be defined shortly.

Important

For the zero and one value, the port is assumed to be terminated with an open circuit. The internal voltage source is not doubled as it is for other waveform types.

One Value

The *One value* property sets the voltage of the port when the pattern specifies a one.

Period of Waveform

The bit waveform type generates a bit stream where the period defines the period of one of the bits in the stream. For example, if the period property is set to 1n, then every nanosecond the bit switches to the next bit in the sequence.

Rise Time

The *Rise time* property specifies the transition time from the zero state to the one state.

Fall Time

The *Fall time* property specifies the transition time from the one state to the zero state.

Pattern Parameter data

This is the bit string, which is a series of the four states; `1`, `0`, `m`, and `z`. An example is 101100 (not separated by spaces) which would generate a 1, then a zero, then 2 ones in a row, and then two zeros in a row.

rptstart

This specifies the starting bit number when repeating. Default value is `1`, which causes the first bit in the sequence to be the starting bit for the repeated pattern. The data repeats from the specified bit to the end of the bit string. The value of the parameter should be an integer from 1 to the length of the bit string.

rpttimes

This specifies the number of times the pattern should be repeated after the first sequence is generated. The default value is `0`. which specifies that the pattern should not be repeated. When its value is `negative`, the string repeats forever. If a finite number of repeats is specified and the simulation time increases to a time after the last bit is generated, the output will maintain the state of the last bit in the sequence.

Bit Waveform Examples

Here is the circuit that is used for all the examples.

The port with the sourcetype set to bit is on the left. Then there is a 1MOhm resistor, and a DC voltage source set to 900mV.

Example 1

Here is a property list for the first example.

Here is the output signal which is *Out1* in the schematic.

The delay time is 5n, and the bit period is 1n. This causes the signal to start at the first bit which is 1, and hold it for 5nsec. Then, it produces the first bit, which is 1 for the next nanosecond. If you look closely, you can see that the middle of the transition occurs at exactly 6nsec. The start of the transition is half the falltime before. The sequence is 1010110000mmzzzzzzzzzz. You can see the 0, 1, 0 transitions at 7nsec, 8nsec, and 9nsec. Then you see the two 1 states from 9nsec to 11 nsec, and the four 0 states from 11nsec to 15nsec. The next two states are middle states at 0.6v from 15nsec to 17 nsec. Then the output goes to high impedance, where the 900mv DC level at the output is seen. Rpttimes is set to 1, which repeats the pattern once, and the starting bit is the first bit in the sequence, which is a 1. At the end of the 900mv section, the signal transitions to the 1 state, and the entire pattern is repeated again. After the second pattern is done, the last state which is a z state is held after that.

Example 2

Here is the property list for the second example.

Everything is the same as the first example except for the repeat start, which is set to 2. This causes the second bit in the sequence, which is a 0, to be the starting bit. At the end of the first pattern, the signal repeats from the second bit in the sequence.

Example 3

Here is the property list for the third example.

Here, there is no repeat.

PRBS Waveform Parameters

PRBS is an acronym for pseudo-random binary sequence. The PRBS source has two modes. It can be used to generate a pseudo-random sequence, or you can specify the beginning state and tap gains for a Fibonacci PRBS generator. To select this type of output, set the *Source Type* to *PRBS*.

For either mode, first you must define the output characteristics.

Delay Time

This specifies a time where the source stays at the zero value, and does nothing. The default is zero seconds. At the end of the delay time, the first bit will be produced.

Zero Value

This specifies the analog voltage to generate in the zero state. The default is zero volts.

Important

Note that for the port, the output is assumed to be open, and not matched for the *Zero value* and the *One value*.

One Value

This specifies the analog voltage to generate in the one state. The default is one volt.

Period of Waveform

This specifies the time of a single bit. This measures the time from the middle of the transition to the middle of the transition between the zero and one state for the duration of a single bit.

Rise Time

This is the zero percent to 100 percent risetime for the transition from the zero state to the one state.

Fall Time

This is the zero percent to 100 percent falltime for the transition from the one state to the zero state.

Using the Internal Random Number Generator

To use the internal random number generator, specify any integer in the seed field. Below is the output when the seed is one (top) and 2 (bottom).

There is no default value for the seed, and this is ignored if the *Pattern Parameter data* and the *Feedback Shift Register* both have values.

Using the Fibonacci Shift Register with Tap Gain

To use this feature, you must set the pattern and tap information. The general configuration of a 6-bit Fibonacci shift register is as follows:

The boxes on the top are the shift register bits. The non-inverters in the middle can either be switched on or off, and are controlled by the taps vector. Taps zero and N (The taps at both ends of the shift register) are always on, which enables the circular connections. The shift register is clocked every period, and the bit shifts right. Depending on the setting of the feedback taps, different bits of the shift register can change the next bit that is loaded into the beginning (left) of the shift register. This logic is asynchronous. The adders at the bottom do modulo-2 addition, which is equivalent to an exclusive-or logic function.

Important

There are different conventions for numbering the taps. Please note that in Spectre, the taps are numbered from left to right, starting at zero.

Pattern Parameter Data

This is a string of zero and one bits that is used to preload the shift register. Only zero and one are allowed for the PRBS generator. X and Z are not allowed. An example is 000111, which would load the left three bits with the zero state, and the rightmost three bits to the one state. If the size of the shift register is larger than the data, the data is loaded into the shift register right-justified, and the leftmost bits are set to zero. If the size of the shift register is smaller than the data, the rightmost bits get loaded in, and the leftmost bits are dropped off.

There is no default for this parameter, and if the *Feedback Shift Register* field is blank, this will be ignored.

Feedback Shift Register

This is a list of the taps to set to one. In the above example, the numbers 1, 2, 3, 4, 5, and 6 can be specified. By convention, the first number is the number of bits in the shift register, and the next series of numbers control the gain of the taps. In Spectre, the numbers can be specified in any order, and the largest number sets the size of the shift register. For example, 6 4 2 (separated by spaces) specifies a 6 bit shift register with G2 and G4 set to one. The rest of the gains are set to zero, or an open connection. Also note that 6 4 and 4 6 give the same sequence. The largest number in the string sets the size of the shift register, and the rest of the numbers define which taps are on and off. If the *Feedback Shift Register* field has a single number, this defines the size of the shift register, and the values in the *Pattern Parameter data* field will endlessly circle through the shift register.

Reference Value

This property is not implemented at the current time.

Random Delay Time

This property is not implemented at the current time.

Detailed Example of Setting the Taps

Consider an example where the *Pattern Parameter data* is set to 000011 and the *Feedback Shift Register* is set to 6 4. This defines the shift register to be 6 bits wide, and the feedback tap G4 is enabled.

The internal bits get set, as shown below. When the tap is on, it is a connection. When it is off, it is an open. When the adders that are not shown have a zero on the top connection, they take the bit from the right, add zero, which just brings that bit to the left. The other adders just become wired connections.

The taps vector enables G4, which brings the fourth bit from the left to the adder for that section. The adder sees a zero and a 1, so the output is 1. This 1 will be loaded in to the shift register at the next clock event.

At the next clock, the 1 from the output of the adder is loaded into the leftmost bit, and the contents shift right.

The output of the right bit is still 1, and the 4th bit from the left is still 0, so the output of the adder stays at 1.

At the next clock, the 1 is loaded in the leftmost bit, and the data shifts right.

Now, the adder sees zero on both inputs, so the output changes to zero. This is loaded in the next clock, and the data shifts right.

If you look at the sequence of the bits on the right side of the shift register so far, 1 1 0 0 has been generated. The sequence is from left to right. The first bit is a 1.

Continuing this exercise, it can be shown that the bits on the output should be 1 1 0 0 0 0 1 1 0 0 1 1 1 1 1 1 0 0 0 0 for the first 20 bits.

Now assume a bit period of 1nsec, The first bit, which is a 1, is generated from zero seconds to 1 nsec. The second bit, which is a 1, is generated from 1nsec to 2nsec. The third bit, which is a zero, is generated from 2nsec to 3nsec, The first transition from a 1 to a zero occurs at

2nsec. The bit sequence above would generate transitions at 2n, 6n, 8n, 10n, and 16n, which is shown in the result below.

Noise Parameters

Important

The check box is a display feature in the form only. If there are properties that are set and the display check box is not selected, although you will not see the parameters in the form, these parameters will still be netlisted, and will still have an effect on the simulation.

The noise parameters include *Noise temperature*, *Noise Entry Method*, *Noise file name*, and *Noise/Frequency points*.

Noise temperature

The *Noise temperature* of the *port* resistor in degrees Celsius. If this is not specified, the *Noise temperature* is assumed to be the temperature that is set by the temperature option. When you compute the noise figure of a circuit driven at its input by a *port*, set the *Noise temperature* of the *port* to 16.85C (290K). This setting matches the standard IEEE definition of noise figure. In addition, disable all other sources of noise in the *port*, such as the Spectre parameters *Noise file name* and eliminate any entries you have in the fields that appear when you select *Noise/Frequency points*. If you want a noiseless *port*, set the *Noise temperature* to absolute zero or below, and do not specify a noise file or Noise/Frequency points. Absolute zero is -273C. The default noise temperature is 290C..

Noise Entry Method

You can select one of two ways to enter noise data, either by specifying a *File* name or entering a series of *Noise/Frequency points*.

Important

Entering noise points define AM noise only. There is no mechanism in the port for including PM noise from a noise file.

Noise file name

If *Noise Entry Method* = *File* is selected, you enter the name of the file containing the noise data in the form of frequency-noise pairs. In your file, list the frequency-noise pairs as one pair per line with a space or tab between the frequency and noise values. The first entry is the frequency, and the second value is volts-squared per Hertz.

Important

Frequencies outside the range specified in the noise file are interpolated, and not zero. If you want zero, make sure that you specify zero over the range you want. Remember that in pnoise, a wide range of noise frequencies are typically evaluated, and this interpolation can cause more noise than expected in the analysis. If you want to see what the source is actually generating, make a circuit with a port and a terminating resistor, and run linear noise over a very wide frequency range. Then plot the noise at the output of the port.

Important

If you want just the noise specified in the file and nothing more, set the noise temperature of the port to -273. The default is to generate the noise from the file and add to it the noise from the resistor in the port. Setting the noise temperature to -273 makes the noise fro the resistor be zero.

Figure 10-14 Display noise parameters: Noise Entry Method=File

Important

The check box is a display feature in the form only. If there are properties that are set and the display check box is not selected, although you will not see the parameters in the form, these parameters will still be netlisted, and will still have an effect on the simulation.

Important

If you want just the noise specified in the noise file and nothing more, set the noise temperature of the port to -273. The default is to generate the noise from the file and add to it the noise from the resistor in the port. Setting the noise temperature to -273 makes the noise from the resistor be zero.

Num. of noise/freq pairs

If the *Noise Entry Method* is set to *Noise/Frequency points*, you specify the *Number of noise/freq pairs*. The form expands to let you type in the designated *Freq* and *Noise* values. The noise values must be in volts-squared per Hz, and frequency in Hz. The maximum number of points in the form is 10.

The example below has the *Number of noise/freq pairs* set to 3.

Figure 10-15 Display noise parameters: Noise Entry Method=Noise/Frequency Points

Important

If you want just the noise specified in the *Num. of noise/freq pairs* properties and nothing more, set the noise temperature of the port to -273. The default is to generate the noise from the noise properties and add to it the noise from the resistor in the port. Setting the noise temperature to -273 makes the noise from the resistor be zero

Small-Signal Parameters

Display small signal params

Important

The check box is a display feature in the form only. If there are properties that are set and the display check box is not selected, although you will not see the parameters in the form, these parameters will still be netlisted, and will still have an effect on the simulation.

If selected, the *Edit Object Properties/Add Instance* form expands to show the small-signal properties *PAC Magnitude*, *PAC Magnitude (dBm)*, *PAC phase*, *AC Magnitude*, *AC phase*, and *XF Magnitude*.

Figure 10-16 Display small signal params

When you specify the voltage on a *port*, you are specifying the voltage when the port is properly terminated, and not the voltage on the internal voltage source. Thus, the voltage on the internal source is set to twice the value specified on the *port*. The same is true for the values for the transient, AC, and PAC signals. However, the amplitude of the sine wave in the PAC and transient analysis can alternatively be specified as the power in dBm delivered by the `port` when terminated with the reference resistance. The *PAC magnitude* property is used by pac, hbac, and qpac. The *XF Magnitude* property is used by xf, pxf, hbxf, and qpxf.

PAC Magnitude

The peak periodic AC analysis magnitude. Setting this value to unity is a convenient way of computing the transfer function from this source to the output.

You can select either *PAC magnitude* or *PAC magnitude (dBm)*, but not both. If *PAC magnitude* has a value, the *PAC magnitude (dBm)* field is grayed out. The PAC magnitude property is used by pac, hbac, and qpac.

PAC Magnitude (dBm)

The periodic AC analysis magnitude in dBm (alternative to *PAC magnitude*). You can select either *PAC magnitude* or *PAC magnitude (dBm)*, but not both. If *PAC magnitude (dBm)* has a value, the *PAC magnitude* field is grayed out. The *PAC magnitude(dBm)* property is used by pac, hbac, and qpac.

PAC phase

The periodic AC analysis phase. The value must be a real number. Default: 0 Units: degrees.

Typically, only one source in the circuit has a *PAC magnitude* set to a value other than zero, and many times it has a *PAC magnitude*=1 and *PAC phase*=0. However, there are situations where more than one source has a nonzero *PAC magnitude*. For example,

applying a differential small-signal input could be done with two sources with the *PAC magnitude*s set to 0.5 and the *PAC phase*s set to 0 and 180. The *PAC Phase* property is used by pac, hbac, and qpac

You do not specify the PAC frequency in the *port Edit Object Properties* form. Instead, you set the PAC frequency in the PAC *Choosing Analyses form*. For example, when making an IP3 measurement, you set the PAC frequency to a variable value in the *Choosing Analyses* form. Then, you enter the same variable in the *PAC Amplitude* (or *PAC Amplitude dBm*) field of the *port Edit Object Properties* form.

AC Magnitude

The peak small-signal voltage for the ac analysis in volts.

AC phase

The ac small-signal phase in degrees.

Typically, only one source in the circuit has *AC Magnitude* set to a value other than zero, and many times it has an *AC magnitude*=1 and *AC phase*=0. However, there are situations where more than one source has a non-zero *AC magnitude*. For example, you can apply a differential small-signal input with two sources with the *AC magnitude*s set to 0.5 and the *AC phase*s set to 0 and 180.

XF Magnitude

This property is seldom used. It specifies the transfer function analysis magnitude. This is used by xf, pxf, hbxf, and qpxf.

Temperature Effect Parameters

Temperature effect parameters include *Linear temp. coefficient*, *Quadratic temp. coefficient*, and *Nominal temperature*.

Display temperature params

Important

The check box is a display feature in the form only. If there are properties that are set and the display check box is not selected, although you will not see the parameters in the form, these parameters will still be netlisted, and will still have an effect on the simulation

If the check box is selected, the *Edit Object Properties*/*Add Instance* form expands and the following three parameters appear in the form: *Linear temp. coefficient, Quadratic temp. coeff.,* and *Nominal temperature*.

Figure 10-17 Display temperature params

Linear temp. coefficient

This property sets the first order (linear) temperature coefficient of all the voltages specified for any waveform in the port. 0.01 entered in the *Linear temp. coefficient* field is interpreted as a one percent change per degree Celsius. If the number is positive, the voltages increase with temperature. If the number is negative, the voltages decrease with temperature. If the temperature moves far enough, the values can become negative.

Quadratic temp. coeff.

This property sets the second order (quadratic) temperature coefficient of all the *voltages* the source generates. 0.01 causes a change in value of one percent per degree squared.

Nominal temperature

The *Nominal temperature* (tnom)for the linear and quadratic temperature coefficients.

Simulating Tabulated S-Parameters Using the Nport Component

Nport from analogLib provides a generic way to incorporate an S-Parameter file into a circuit simulation. The nport component allows you to set the number of ports, and when this is done, the schematic symbol re-draws with the number of ports shown. There is also n1port, n2port, n3port, and n4port in analogLib, which have a fixed number of ports. These components are provided for legacy purposes only, and should not be used in current simulations.

For the frequency-domain based analyses, such as ac, noise, xf, stb, pz, harmonic balance, hbac, hbxf, and hbnoise, the S-Parameter file is used directly in the frequency domain. For the DC and transient based simulations dc, tran, and shooting, a time-domain model must be created from the frequency-domain description in the S-Parameter file. This is a non-trivial process.

In general, there are two different ways to create a time-domain model from an S-Parameter file. One way is to do a rational fit, which attempts to match the frequency domain description using a combination of poles, zeros, and gain blocks. The other way uses a convolution-based approach. This approach calculates the impulse response, and then creates a time-domain model based on the impulse response.

Regardless of the method, there are fundamental requirements for the S-Parameter file. These are:

- The file must contain a frequency point that is low enough so that when the DC point is extrapolated, the DC point correctly defines your circuit behavior.

  This is a fundamental requirement. To run a transient, first you need to run the DC analysis to capture the time-zero timepoint. If the first frequency in the file is not zero frequency, then a zero-frequency point must be extrapolated. If the first frequency in the S-Parameter file is not low enough to accurately represent the real DC characteristics of the system, the DC extrapolation will be incorrect.

- The phase of the DC point must be zero or 180 degrees.

- The frequency-domain data must be dense enough to accurately describe the transfer function.

  Smooth transfer functions without local complex behavior might need relatively few datapoints. Systems with complicated behavior need to have enough datapoints to capture the behavior well enough so that when the data is interpolated between the given points, and the interpolation error is small.

For bandpass filters, make sure that data from near zero Hz through at least the frequency of the highest harmonic in the simulation is present in the S-Parameter file.

- The frequency data must exist to a high enough frequency so that harmonics of the system are accurately simulated.

  If your S-Parameter file has data only through the second harmonic of the input signal and you are calculating six harmonics in a large-signal analysis, just note that the S-Parameter data above the second harmonic is extrapolated. The simulation is not based on real data, and as a result, it is different from the real system. If you want your simulation to be accurate for the harmonics you are simulating, make sure that the S-Parameters themselves go high enough in frequency.

- As a strong suggestion, regardless of the source of the data, plot the data from your S-Parameter file before you simulate. The plot for all the S-Parameters should plot in a clockwise arc with no sudden jumps. These discontinuities cause large inaccuracies in the transient model. Spectre now checks for discontinuities in the S-Parameter data.

To plot an S-Parameter file directly in the waveform tool, the file format must be in Touchstone1 format. Here are the steps required to plot an S-Parameter form an S-Parameter file.

1. From the ADE window, select *Tools-Waveform*.
2. In the waveform tool, select *File-Open Results*, and browse to the location of the S-Parameter file.

3. When the file is selected, the waveform tool has the path to the results at the top-left and the S-Parameter filename on the bottom. If you started form ADE, you may also have the path to the psf file on the upper left.

4. Double-click the filename. The S-Parameters in the file are displayed.

5. On the upper left, select *Impedance* or *Admittance*. This tells the waveform tool to plot on aa Smith Chart.

6. Click one of the S-Parameters. Click the right mouse button and select *Move to Plot Signal*. The signal plots.

Important

S11 above has visible piecewise-linear behavior at low frequencies. This S-Parameter file will produce incorrect time-domain results at the frequencies between the inflection points in the S-Parameter plot.

Potential Problems with S-Parameter Files

Not all frequency domain descriptions are causal. This is a fundamental requirement for the time-domain model to be accurate. When Spectre reads in an S-Parameter description and causality is being checked, the transfer function from the highest frequency in the S-Parameter file through three times the highest frequency in the S-Parameter file is adjusted

to enforce causality. Frequencies above three times the frequency in the S-Parameter file should be avoided.

Convolution-Based Method

The convolution-based method is strongly suggested for most simulations. This method is used when the interpolation method is set to linear or spline. The default interpolation method is linear, and the convolution-based method is the default.

The convolution-based method samples the frequency-domain data, and then calculates the impulse response for that frequency description. Then a Fourier transform is used to create the DC and time-domain model.

Considerable development has been made in MMSIM12.1 so that the only thing that should be required is to set the S-parameter file name, the interpolation method, and the number of ports. The default settings are strongly recommended for all other parameters. By default, the impulse responses are cached. This allows the reuse of previously computed impulse response files, thereby saving a lot of time in simulations after the first time the simulation is run. If multiple instances in the schematic use the same S-Parameter file, the impulse response is calculated only once. The impulse response may need to be recalculated if you specifically set some of the properties that are available in the nport. Except for the *Passivity* parameter and the *Accuracy* parameter, changing any other property is highly discouraged unless you are an expert in simulating S-Parameters. The old Impulse response will not be erased. If you switch back to the previous settings in the nport, those files will still be available, and will still be reused. The location of the files depends on the setting of the *nportfiledir* option which is documented later.

Using the nport Component

Use the *nport* component to read in S-parameter data. Follow these steps to prepare the component for use in SpectreRF simulations.

Figure 10-18 nport Component Edit Object Properties Form

1. Select the *nport* component from the *analogLib*.
2. Place the *nport* component in the Schematic window.
3. Set the number of ports that are in the S-Parameter file.
4. In the Schematic window, highlight the new component and then choose *Edit – Properties – Objects, or use the bindkey you have set up for setting properties*.

The *Edit Object Properties* form for the *nport* component appears.

5. Click the file browser button located to the right of *Browse* and select the S-data file in the properties form. From the browser, select the S-Parameter file.

Alternatively, type in the S-Parameter data filename. If you specify just the filename in the port component (not the complete filename starting from slash), then in ADE, select *Setup - Simulation Files*, and enter the path to the directory that contains the S-

Parameter file in the *Include Paths* section. This is necessary so that the file can be found.

6. Although the properties form is labeled S-Parameter data file, it can also be a Y-Parameter or Z-Parameter file. Spectre recognizes all three formats. It is labeled this way because S-Parameter files are more common than Y or Z-Parameter files.

7. Specify whether you want the passivity check to run. Setting *Passivity* to *no* causes no passivity check to be run. Setting *Passivity* to *check*, which is the default, will run a passivity check, and the result of the check will be sent to the Spectre output window. The S-Parameter file will be used as specified. If *Passivity* is set to *enforce*, the passivity check will be run, and the S-Parameter file will be changed as necessary to ensure the passivity of the result. Note that this will change the S-Parameters as specified in the file if a non-passive entry is detected.

8. Set the *Interpolation method* cyclic field.

   The *linear* and *spline* values are usually preferable to *rational.* In general, the recommended method is *linear.*

   Linear and spline control the sampling of the S-Parameter data for the convolution-based method. In both methods, the S-parameter data is sampled using a linear frequency spacing from zero to three times the highest frequency in the S-Parameter data file in order to calculate the impulse response of the transfer function.

Important

Note that when the S-Parameter file is for a physically large structure like a package model, because of the physical size of the structure, the inductance drops starting in the range of 100KHz to 1MHz. To correctly model this behavior, you require data in the S-parameter file that accurately describes this behavior, and it requires a huge number of frequency sample points in order to get a correct time-domain model.

When *linear* is selected as the interpolation method, linear interpolation is used to get a datapoint needed in the sample that is not directly in the S-Parameter file. Spline uses a cubic spline algorithm. Cubic spline can occasionally introduce errors when there are rapid changes in the transfer functions defined in the S-Parameter file near the sample point.

When the data in the S-Parameter file is dense, linear and spline produce comparable results. When the data in the S-Parameter file is sparse, non-physical results can be introduced because of the wiggles that can be introduced by a polynomial fit. Linear is usually a better choice when the data is sparse.

When you select *rational*, four new fields are displayed. For information about these fields, see Controlling Rational Fit Accuracy on page 1564.

9. If exceptional accuracy is desired, select *Conservative* for the Accuracy default. This will cause more frequency-domain sample points, and produce a more accurate impulse response at the cost of runtime. *Default* accuracy is set tight enough that *Conservative* should only be needed in rare instances.

When the DC or transient-based simulation is run, Spectre provides information about the S-Parameter file and the impulse response file in the output window.

Advanced transient convolution parameters

These properties are provided for S-Parameter simulation experts only.

Max sampling points

Max sampling points defines the maximum number of frequency points to be sampled in the adaptive algorithm. The default is 131072. In every case, we have encountered so far, the actual number of samples taken by the adaptive algorithm is much smaller than the default. In extremely unusual cases, it can be raised to 262144.

Max frequency of interest

The property *Max frequency of interest* should not be changed unless you are an expert in the simulation of S-Parameter files.

Max frequency of interest (fmax) controls the highest frequency for the frequency domain sampling of the S-Parameter file. Leave this property at the default. The default is three times the highest frequency in the S-Parameter file.

Impulse response truncation

Impulse response truncation is used to deliberately cut off the tail of the impulse response which might theoretically continue to infinite time. Leave this property at the default of 1e-4, which corresponds to a gain of -80 dB.

Causality Correction

The *Causality Correction* drop-down list contains three choices: `no`, `fmax`, or `auto`. `fmax` is the default and is highly recommended. Causality enforcement is required in order to have reasonable results from an nport in either the DC or transient-based analyses. Causality correction is performed by setting the transfer function between the highest frequency in the S-Parameter file and three times this frequency so that the data becomes causal. The data within the frequencies specified in the S-Parameter file is unchanged. Setting causality to `fmax` or `auto` overrides the setting of the high frequency extrapolation property.

- `No` does not add a causality check.

Important

Setting the causality check to *no* is incredibly risky unless you are absolutely sure that the S-Parameter file is causal as described.

- `Fmax` retains the data in the frequency range of the S-parameter file, and then adds a transfer function above the frequency range in the S-parameter file to force the system to be causal. This transfer function extends to the setting of *Max frequency of interest*, which defaults to three times the highest frequency in the S-Parameter

file. If you suspect that the maximum frequency of interest needs to be changed, use causality *Auto* instead, if you are not an expert.

- `Auto` applies the causality correction in a similar manner to choosing fmax. `Auto` can also vary the maximum frequency of interest if it needs to get a causal time-domain model.

DC extrapolation

The *DC extrapolation* can be set to `constant` or `unwrap`. `Constant` projects the first point down to zero frequency at exactly the same level. `Unwrap` does an estimation based on the first few frequency points in the S-Parameter file. The default is *unwrap*.

Rarely Used Parameters

These properties should never be needed.

S-Parameter data format

S-Parameter data format is not needed because Spectre can directly parse an S, Y, or Z-parameter file in Spectre, Touchstone1, Touchstone2, or CITI file format.

Note that the Touchstone1 file format now provides support for multiple impedances at each port. This is done in the header of the file. When R is listed, then multiple values can be entered up to the maximum number of ports in the file. If a number is omitted, then that number takes on the value of the first R entry.

For example, `#GHz RI R 50 1 1 50` means the frequency numbers are in GHz. The data is in real-imaginary format. The resistance is 50 Ohms on port 1 and 4, and 1 Ohm on ports 2 and 3.

`#GHz RI R 50 10 20 30` means 50 ohms on port 1, 10 ohms on port 2, 20 ohms on port 3, and 30 ohms on port 4.

`#GHz RI R 50 10` means 50 ohms on ports 1, 3and 4, and 10 ohms on port 2.

Causal s-param output file

Enter a filename beginning with slash (/) in this field. Specifying a filename causes the S-Parameter data after causality correction to be placed in this file. This file can then be plotted directly in the waveform display tool. Most of the time, the data within the S-Parameter file frequencies is not disturbed, and the data between the highest frequency and three times the highest frequency is the data that is added to make the time-domain model causal. Below is

a comparison of the raw data and the causality-corrected data. The red trace is the original data.

This has the data plotted on a linear X-Axis.

This has the data plotted on the Smith Chart. The original data is in red, and the causality-corrected data is in blue.

Here is the original data in the Smith Chart.

Passivity tolerance

This property is only used when the *Passivity* property is set to *check* or *enforce*. Passivity tolerance does not need to be set. The default is 1e-6. This defines how close to unity gain should be modified by the passivity check and enforcement. Passivity will be enforced and/or reported when the gain is (`1 - Passivity tolerance`) or greater.

Data truncation threshold

Leave this property at the default of 1e-3, which corresponds to -60dB gain. When the cross coupling terms become smaller than the data truncation threshold, they are ignored. Cross coupling is the coupling from one port to another port.

Frequency sampling interval

Frequency sampling interval sets the delta frequency for the sampling from zero to the maximum frequency of interest. Leave this property at the default value. With adaptive sampling, this should never be necessary. If used, this delta should be a power of two divisor of the maximum frequency of interest.

Multiplier

Multiplier specifies how many nport devices to put in parallel. This is rarely used.

Scale Factor

This scales the frequency of the S-Parameter file. For example, many S-Parameter files have the frequency in GHz. In this case, set the *Scale factor* to 1e9.

High freq extrapolation

This property should not be used. This property is ignored when causality correction is applied. The *High freq extrapolation* field can be set to *constant* or *linear*. If the causality check is not run, *constant* maintains the same amplitude and phase as the last point in the S-parameter file to infinite frequency. *linear* keeps the amplitude constant at the last frequency point, but the phase increases linearly with frequency.

Flag for matrix form input

Flag for matrix form of input should not be needed to be set. In the past, each time the simulation ran, the impulse response was calculated for every port of every instance of the nport every time the simulation was started. In some cases, especially with a large number of ports, this could take considerable time. This flag was provided so the step of calculating the impulse response could be skipped. Since the impulse response is cached and available for re-use at any time, this property should never be needed.

Additional parameter list

This field is typically used by beta customers to unlock new features. Note that when this feature is used, a warning message will be issued. This warning can be ignored.

Controlling Rational Fit Accuracy

Rational fitting is the process of adding multiple poles, zeros, and gain blocks to match the frequency response that is defined in the S-Parameter file. Note that the S-Parameter file may have distributed effects. Approximating these distributed effects with poles and zeros can require a system with a very high order. Matching an arbitrary frequency-domain transfer function with poles and zeros is a difficult and complicated process.

At the current time, it is usually better to use the convolution-based algorithms when the interpolation method is set to *linear* or *spline*.

If you do use the rational fit, always test the rational model that is produced by running sp analysis using a circuit composed of an input port, an output port, and an nport with the interpolation method set to *rational*. Run the sp analysis, and plot all the S-Parameters. Look for differences between the original S-Parameter file and the S-Parameters with the rational fit.

Also, the algorithm is known to take a very long time to run for S-Parameter files with a large number of ports.

The *nport* component has three parameters to control the accuracy of the rational interpolation process,

- *relerr* (The *Relative error* field)
- *abserr* (The *Absolute error* field)
- *ratorder* (The *Rational order* field)

You can use these parameters to trade off accuracy against model size and simulation time. In general, the more stringent the accuracy requirement, the higher the model order. Higher-order models require longer extraction and simulation time. SpectreRF can automatically generate a model that meets a specified accuracy requirement. You can also specify the model order directly to SpectreRF.

Using the relerr and abserr Parameters

The *relerr* parameter is used to control the accuracy of the rational fit when the gain is larger than the gain specified in abserr. Usually, *relerr* is on the order of about 0.01, which sets a 1% error in the rational fit when the gain is large. Usually, *abserr* is on the order of 1e-3 to 1e-4, which specifies that when the gain is less than *abserr*, allow a lot of error in the rational fit. 1e-3 is -60dB gain, and 1e-4 is -80 dB gain. The *abserr* parameter lowers the order of the model by allowing fewer poles and zeros when the gain is small.

Consider the data shown by the solid line in Figure 10-19. In this example:

- *relerr* can control the error in the passband
- *abserr* can control the error in the stopband

If you are not interested in the details of the filter behavior in the stopband, you can set *abserr* to the level shown and these details are ignored in the interpolation.

Figure 10-19 Using relerr and abserr

This example has approximately a 10% ripple in the passband. To ensure adequate error control there, set *relerr* to much less than 0.1, for example, 0.01.

The dashed line in Figure 10-19 shows an interpolation function with the error control levels specified in Figure 10-19: *abserr*=2e^{-4}, and *relerr*=0.01. Details of the data below the *abserr* threshold are not resolved by the rational interpolation.

To recover the details in the stopband, set *abserr* to about 1e^{-5} for this example.

If there is noise in the data, set *relerr* and *abserr* above the respective noise levels. Otherwise, Spectre and SpectreRF attempt to interpolate the noise, resulting in very high order models and very long simulation times.

Remember that if you set *relerr* to zero, then, from the formula above, pure absolute error is used. Conversely, if you set *abserr* to zero, the error control is based solely on the errors relative to the magnitude of the input data.

Using the rational order(ratorder) Parameter

In general, it is usually best to specify only the accuracy parameters, *relerr* and *abserr*, and to let SpectreRF automatically select *rational order,* the order of the rational approximation. However, if you have information about your data set, you can direct SpectreRF to use a specific order of approximation in the rational interpolation. For example, if you know that your tabulated S-parameter data represents a sixth-order filter, you might instruct SpectreRF to use a seventh or eighth order fit. The slightly higher order gives SpectreRF flexibility to adjust for any noise or non-ideal behavior in the data.

When you specify the *ratorder* parameter, SpectreRF tries to meet the error requirements specified by *relerr* and *abserr*, but is limited to the number of poles and zeros it can use to achieve them. If the error specified cannot be achieved with the number of poles and zeros you have specified, a warning message is generated to let you know that the error requirements were not met.

ROM data file

If this file is specified, when the rational fit is performed, the results of the rational fit are stored in the specified file as a Spectre format reduced order model.

Troubleshooting

Certain S-parameter data sets might cause difficulty for the rational interpolation process. Types of data to avoid are:

- Data specified only over a very narrow frequency range. Time-domain simulation requires time-domain models with a wider range of values; values that lie outside the narrow frequency range. Data extrapolation might be difficult and is always risky.
- Very noisy data. Noisy data might lead to large, unreliable time-domain models.
- Data on a very sparse frequency grid. Accurate interpolation of such data might be impossible.
- Data with long ideal delays.
- Data representing idealized lossless elements, such as lossless transmission lines.

Assessing the Quality of the Rational Interpolation

If you suspect a problem with the rational interpolation process, you can investigate it using the *sp* analysis using the following steps:

1. Construct a test schematic consisting of an *nport* component with interpolation set to rational, as discussed in "Using the nport Component" on page 1550.

2. Next add the appropriate number of *port* components to the schematic. An example with two ports is shown below.

3. Perform an *sp* analysis on the *nport* component and look for anomalies.

   Refer to Using S-parameter Input Files *on page 2045* for an example of how to set up and run this type of simulation.

Large swings in interpolated values and S-parameter magnitudes greater than one both suggest a problem.

Large changes in the interpolated data result from an inaccurate fit. Usually when this happens, there are local peaks or suckouts in the S-Parameter data.

S-parameter values greater than one result from a non-passive (energy-generating) model that might create unstable time-domain solutions. Be particularly critical of anomalies near the zero frequency (DC). Most S-Parameter files do not have zero frequency data, so the DC point must be extrapolated. Sometimes in rational fit, the DC point can be inaccurate.

When the anomalous behavior occurs *within* the frequency range of the data in the S-parameter data file, it usually indicates an inaccurate rational interpolation. Anomalous behavior below the first frequency point and above the last frequency point are uncommon, but always needs to be checked. If you have this anomalous behavior, linear interpolation is suggested.

Possible Problems with Rational Fit

Because the rational fit can have unexpected behavior, it is suggested that you run an S-Parameter simulation over a very wide frequency range. Check for those behaviors in the sp analysis.

A comparison between rational and linear interpolation is shown below. Note that S21 and S11 both go considerably above unity gain (0dB) above about 60GHz for the rational fit. The data in this file stops at 25GHz. This data is from a passive structure.

Magenta and green are the original data. Blue and red are the rational fit.

Also note the differences in the low frequency data shown below.

1. Verify that all the conditions listed at the beginning of the Troubleshooting section are met.
2. Try decreasing *relerr* or *abserr* or both.
3. Try specifying a higher-order interpolation with the *ratorder* parameter.
4. Remember that measured data can contain fine details that might require a higher order than you might expect from a casual inspection of the data.
5. If the anomalous behavior occurs *outside* the tabulated frequency range, try changing the *abserr*, *relerr*, or *ratorder* parameters. Sometimes anomalies can be removed by using a more accurate fit. However, there are limits to the ability to extrapolate outside the frequency interval you specify. You might need to specify additional data points in the S-Parameter file to fix the problem.

Difficulty can also sometimes be seen in running the transient analysis. Below is a *Choosing Analyses* form that shows a setup to run the transient to 10nsec.

The simulation runs. and the data plots, but note that only part of the waveform has been calculated.

When you look in the Spectre output log, the iterations have stopped because the solution goes above 1GigaVolt.

Model Reuse

Rational Fit

When *rational* is set for the interpolation method, Spectre caches the rational model in the simulation directory. As long as you do not change the S-Parameter file, the rational fit will be reused.

Linear and Spline Fit

Prior to MMSIM 12.1, every instance of every nport calculated the impulse response at the beginning of the simulation. In some cases, this added a lot of time in order to accomplish this. Starting with MMSIM 12.1, only the impulse responses that need to be calculated are performed. For example, if there are multiple instances of nport with the same S-Parameter file and the same nport settings, the impulse response is calculated only once.

The impulse response data is cached by default so that as long as the S-Parameter data and the properties on the nport do not change, the cached data is reused in subsequent simulations. When the properties on the nport instance are modified, then a new impulse response is needed. The impulse response is recalculated and cached without eliminating the original cached data file. This is done so that if you go back to the original settings, you do not need to re-run the impulse response.

Additionally, file locking has been implemented to allow multiple users to have access to a shared impulse response directory. This is controlled by the *nportfiledir* option described below.

There are two options that control caching. These options are located in the ADE options located in *Simulation-Options-Analog*.

Nportirreuse

This option has the default of *yes*. Keep this at the default to save time. This option enables caching of the impulse response.

nportirfiledir

This controls the location of the cached impulse response files. The default (blank) sets the location to `/home/<username>/.cadence/dfII/mmsim`. When a directory name is specified in the *nportirfiledir* option, the impulse response files are cached in the specified directory. This can be a shared directory between users, as long as all the users have write permissions in that directory.

Dcblock, dcfeed, indq, and capq

Ideal components are provided to allow what-if analysis without needing details of specific structures to allow the determination of what is needed to meet the specifications.

These components are provided for any analysis in Spectre except shooting. This allows the use of inductors and capacitors with Q in s-parameter, ac, and hbac analyses, along with the other small-signal Spectre and SpectreRF analyses.

Dcfeed

Dcfeed is an ideal short at DC and an ideal open at all other frequencies. For the transient analysis, dcfeed is an inductor with a default value of 1uh.

The symbol and the properties form are shown below. The *Model Name* field should be left blank.

An example comparing the dcfeed, and inductor, and a wire is shown below.

This example shows three cascode amplifiers with different connections to the power supply for the upper transistor. The left amplifier has a dcfeed. The center amplifier has an inductor. The right amplifier has a wire. In the amplifiers above, the resistor to the left has the output point set to 1K ohm.

When the inductor and the dcfeed are essentially open, the gain of the circuit is set by the resistor. When the impedance is less than 1Kohms, the load impedance is dominated by the impedance of the DCfeed or inductor. In this way, we can deduce what is happening at low frequency by running a simulation.

The amplitude of the input is deliberately large so the amplifiers tend toward the mixer behavior. The input signals are at 2.45GHz and 2.451GHz. The difference frequency is 1MHz. The 1 MHz signal will be analyzed so the behavior of the dcfeed, inductor, and wire can be evaluated.

At all frequencies other than DC, the dcfeed is open. The inductor impedance goes up with frequency. At 1MHz, the impedance of the 1uh inductor is 6.28 ohms.

Below is the simulation result from harmonic balance near 2.45GHz. The result agrees for the dcfeed and inductor. With a wire, essentially no signal is present.

In the table above, the left result is the output signal level in dBm for the amplifier with the dcfeed, the middle is the signal level in dBm for the amplifier with the inductor, and the right

is the output signal level in dBm for a wire connected to the power supply. The third output is at the numerical noise floor of the simulation. Below is the spectral plot near 2.4GHz.

Near 1MHz, the story is different.

In the table above, the left result comes from the amplifier with the dcfeed. The center is with the inductor, and the right is with the wire. The dcfeed is open at 1MHz, so the signal level is set by the 1K resistor. Because the inductor has an impedance of only 6.28 ohms at 1MHz, the signal is much smaller than for the dcdfeed. With a wire, you see the numerical noise floor of the simulation.

The spectral plot is shown below.

Dcblock

Dcblock is a near open (1Gohm) at DC and an ideal short at all other frequencies. For the transient analysis, dcfeed is a capacitor with a default value of 1uF.

The schematic symbol and properties list are shown below.

Below is an example with three cascode amplifiers that use different connections to the output load resistor. The left amplifier has a dcblock in series with the load resistor. The middle

amplifier has a capacitor in series with the load resistor. The amplifier on the right has a wire connected to the load resistor. The output resistance is 10KOhms.

Here is a table with the DC values from the three cases. The left column is the DC voltage on the load resistor with the dcblock, the center column is the DC voltage with the capacitor, and the right column is the DC voltage with the wire.

	Expression	Value	Expression	Value	Expression	Value
1	value(v("/Out4...	28.99E-6	value(v("/Out5...	0.000	value(v("/Out6...	2.842

The voltage with the dcblock is only a few microvolts. This is because it is a 1Gohm resistor at DC. The capacitor is a perfect open at DC, so the voltage is exactly zero. With a wire, the DC bias level appears at the load resistor.

Here, is the simulation results from harmonic balance. Instead of dB, the vertical axis is linear volts. Note that the DC level is small for the dcblock. This is the top trace. The dcblock is a

1e9 ohm resistor at dc, and an ideal short at all non-zero frequencies. The center trace is from the amplifier with the capacitor, and the bottom trace is from the circuit with the wire.

indq

Indq is a series inductor and resistor. There are three different modes with different frequency behavior in each mode. Below is the test circuit for the plots that will come later. PORT1 is

defined as port1 in a single port S-parameter analysis so that the Z-Parameters for the inductor can be directly plotted.

The properties list for *mode 1* is shown below.

The inductor with Q works in all the analyses except for shooting.

The indq component has three modes, with different behavior in each mode. For all three modes, you need to specify the inductance, Q, and the frequency for the inductance and Q specification.

Mode 1

In this mode, the indq model is a series RL branch with R=constant and L=constant at the value given by the *Inductance* property. The value of the property *Rdc in mode2 and 3* is not used in this mode.

Special case: specifying Q=0 or F=0 makes the indq component just an inductor with the value L without any series resistance.

The resistance is calculated as follows:

$$R = \frac{\omega_Q L}{Q}$$

Below is a plot of the inductance, resistance, and Q with respect to frequency. A vertical marker has been placed at 1GHz, which is the frequency for L and Q. The inductance and resistance are constant, and the Q rises with frequency.

Mode 2

In this mode, both the resistance and the inductance are frequency dependent.

$$R(\omega) = R_{DC} + R'' \sqrt{\omega}$$

$$L(\omega) = L\infty + R'' \sqrt{\frac{1}{\omega}}$$

If the *Rdc* in *mode 2 and 3* property is set to zero or is left blank, the DC resistance is 0.05*the resistor value that is set at the frequency defined by the *Frequency for L and Q* property.

R_{DC}, R'', and $L(\infty)$ being constants are the classic forms of the skin-effect expressions for the resistance (grows as the square root of frequency) and inductance (the internal inductance decays inversely proportional to the square root of frequency). These expressions specify a causal frequency dependence of impedance. At the frequency specified in the *Frequency for L and Q property*, L and Q are equal to the values specified for *Inductance* and *Quality factor*.

The transient analysis cannot use frequency-dependent impedances, so it uses the *inductance* specified in the properties form, and the R_{DC} *in Mode 2 and 3* property for the resistance.

Below is the property form for mode 2.

Below is the inductance, resistance, and Q. Note the log frequency for inductance.

Mode 3

The advantage of mode 3 is that the non-physical frequency dependence of R near f=0 (DC) present with mode=2 is corrected. In this mode, R(f) has a reasonable frequency derivative near f=0. However, the expressions are non-causal as R varies as a function of frequency while L remains constant.

In this mode, L is constant at the value specified and

$$R = \sqrt{(R_{DC})^2 + \left(\frac{\omega L}{Q}\right)^2}$$

If the *Rdc* in *mode 2 and 3* property is set to zero, the DC resistance is 0.05*the resistor value that is set at the frequency defined by the *Frequency for L and Q* property.

Because the transient analysis cannot handle frequency-dependent resistance, the transient analysis uses the *inductance* specified in the properties form, and R_{DC} for the resistance.

The properties list for mode 3 is shown below.

The inductance, resistance, and Q in mode 3 are plotted versus frequency, as shown below.

Here is an expanded view of Q at low frequency. Q drops off at low frequency.

capq

Capq is a parallel capacitor and resistor. There are three different modes with different behavior in each mode. In all three modes, the capacitor is constant at the value specified in the properties form. Below is the test circuit for the plots that will come later. PORT1 is defined

as port1 in a single port S-parameter analysis so that the Y-Parameters for the capacitor can be directly plotted.

Mode 1

In this mode, the capq model is a parallel R C branch with R=constant and C=constant at the value given by the *Capacitance* property.

The conductance is constant as given below.

$$Real(Y) = \frac{C\omega_Q}{Q}$$

ω_Q above is the frequency for the Q value in radians per second.

This model is used in all the analyses, except shooting, where it is not supported.

Below is the properties list for the capq with mode 1 selected.

Below is a plot of the conductance, and Q with respect to frequency in mode 1. A vertical marker has been placed at 1GHz, which is the *frequency for Q*. The capacitance and resistance is constant, and the Q rises linearly with frequency.

Mode 2

In mode 2, the conductance is frequency dependent as shown in the equation below.

$$Real(Y) = \frac{C\sqrt{\omega \times \omega_Q}}{Q}$$

The capacitance is constant in all modes.

ω_Q above is the frequency where Q is specified in radians per second.

Mode 2 is non-causal. Because the resistance is frequency dependent, the transient analysis uses the capacitance specified in the properties list with no resistor in parallel.

Below is the properties form for mode 2.

Below is the conductance and Q versus frequency. Vertical markers have been placed at 1GHz and 40GHz. Mode 2 is a compromise between a linear Q increase with frequency in

mode 1, and a constant Q in mode 3. The Q goes up with frequency, but not as fast as in mode 1.

Mode 3

In mode 3, C is constant at the value specified in the properties form, and the conductance is

$$Real(Y) = \frac{\omega C}{Q}$$

Mode 3 is non-causal. Because the resistance is a function of frequency, the transient analysis uses just the capacitance term.

The properties list for mode 3 is shown below.

The conductance and Q versus frequency is plotted below.

In mode 3, the capacitance is constant, and the Q is constant. The resistance is a function of frequency.

Reference: S-Parameter Equations

This section describes the equations used by the VIVA Calculator to plot data generated by Spectre S-parameter simulations.

Using the VIVA Calculator in the Analog Design Environment, you can plot the following S-parameter data:

- Network Parameters
- Two-Port Scalar Quantities

- Two-Port Gain Quantities
- Two-Port Network Circles
- Equation for VSWR (Voltage Standing Wave Ratio)
- Equation for GD (group delay)

Use the buttons located in the *Function* area of the S-parameter *Direct Plot Form* to specify the type of data to plot.

Network Parameters

You can plot S, Y, Z, and H network parameters.

S-parameters, Y-parameters, and Z-parameters (denoted as SP, YP, and ZP on the *Direct Plot Form*) are defined for circuits with any number of ports. H-parameters (denoted as HP on the *Direct Plot Form*) are defined only for two-port circuits.

You can plot parameters on polar charts, Smith charts, or on rectangular plots after applying a *Modifier* option. The dB conversion uses $20 \log_{10} X$ because the parameters represent scalar ratios (for example, voltage).

Equations for Network Parameters

For the ZP, YP, and HP parameters, Spectre returns S-parameters to the analog design environment. The environment converts them, as needed, to the equivalent Z, Y, and H matrixes using standard published methods. Spectre calculates S-parameter values.

SP (S-parameter) values

SP (S-parameter) values are calculated by Spectre.

ZP (Z-parameter) equation

The Z-parameter equation is as follows:

$$Z_m = [Z_{ref}][I + S_m][I - S_m]^{-1}[Z_{ref}]$$

Where

- S_m is the N-port S-parameter matrix

- I is the N x N identity matrix
- Z_{ref} is the characteristic impedance of the port
- Z_m is the resulting Z-parameter matrix

YP (Y-parameter) equations

The Y-parameter equations are as follows:

$$Y_m = [Y_{ref}][I - S_m][I + S_m]^{-1}[Y_{ref}]$$

Where

- S_m is the N-port S-parameter matrix
- I is the N x N identity matrix
- Y_{ref} is a diagonal matrix defined as

$$Y_{ref} = \frac{1}{\sqrt{\Re e\{Z_i\}}}$$

Where

- Z_i is the terminating impedance at port i
- Y_m is the resulting Y-parameter matrix

The HP (H-parameter) equations

The HP (H-parameter) equations only apply to two-port circuits.

D is

$$D = (1 - S_{11})(1 + S_{22}) + S_{21}S_{12}$$

H_{11} is

$$H_{11} = \frac{[(1+S_{11})(1+S_{22}) - S_{21}S_{12}]Z_{ref1}^{2}}{D}$$

H_{21} is

$$H_{21} = \left(\frac{-2S_{21}}{D}\right)\frac{Z_{ref1}}{Z_{ref2}}$$

H_{12} is

$$H_{12} = \left(\frac{2S_{21}}{D}\right)\frac{Z_{ref1}}{Z_{ref2}}$$

and H_{22} is

$$H_{22} = \frac{(1-S_{11})(1-S_{22}) - S_{21}S_{12}Z_{ref}^{2}}{D}$$

Where

- Z_{ref1} is the terminating impedance at port 1
- Z_{ref2} is the terminating impedance at port 2

Two-Port Scalar Quantities

The two-port scalar quantities include

- NF_{min} is minimum noise figure
- NF is noise figure
- R_n is equivalent noise resistance
- G_{min} is Optimum Noise reflection coefficient

- B_{1f} is alternative stability factor
- K_f is stability factor

You can plot two-port scalar quantities only against frequency. In addition, you can plot them only on rectangular charts. Figure 10-20 on page 1603 illustrates a generic two-port circuit that defines impedances and reflection coefficients.

Figure 10-20 A Generic Two-Port Circuit

In Figure 10-20:

- Z_s is the source impedance
- Γ_s is the input reflection coefficient
- Z_L is the load impedance
- Γ_L is the load reflection coefficient
- Z_O is the characteristic impedance

Equations for Two-Port Scalar Quantities

G_{min} (Optimum Noise Reflection Coefficient) Equation

G_{min} (also known as Γ_{min} in the literature) is the reflection coefficient associated with minimum noise figure. You can plot G_{min} on a Smith chart or, by using the *Modifier* field, on a rectangular chart.

$$G_{min} = \Gamma_{min} = \frac{Z_{on} - Z_o}{Z_{on} + Z_o}$$

Where

- Z_o is the characteristic impedance
- Z_{on} is the source impedance associated with minimum noise figure (NF_{min})

NF_{min} (Minimum Noise Figure) and NF (Noise Figure) Equations

The NF_{min} and NF plots are controlled by the *Modifier* option in the *Direct Plot Form*.

Plot noise figure (NF) in dB by setting *Modifier* to *dB10*

- Plot noise factor (F) by setting *Magnitude*

Use NF_{min} and NF only for two-port circuits.

$$NF = 10\log_{10}F$$

Where

- F is the noise factor
- NF_{min} is the minimum noise figure

The NF_{min} values are calculated by Spectre

The NF (noise figure) equation is calculated by the Analog Design Environment from NF_{min}, G_{min} (Γ_{min}), and r_n. You can specify the optional source reflection coefficient Γ_S as an argument if you use the Analog Design Environment calculator. From the *Direct Plot Form*, the Analog Design Environment assumes Γ_S to be 0 (input matched to reference termination).

$$NF = NF_{min} + \frac{4r_n\left|r_s - r_{min}\right|^2}{\left(1 - \left|r_s\right|^2\left|1 + r_{min}\right|^2\right)}$$

Where

$$r_n = \frac{R_n}{Z_o}$$

Here r_n is the *normalized* equivalent noise resistance.

Rn (equivalent noise resistance)

R_n plots equivalent noise resistance. The R_n values are calculated by Spectre.

Stability Factors

K_f and B_{1f} plot the Rollet stability factor and its intermediate term. Use these parameters only for two-port circuits.

D is

$$D = S_{11}S_{22} - S_{21}S_{12}$$

K_f is

$$K_f = \frac{1 - |S_{11}|^2 - |S_{22}|^2 + |D|^2}{2|S_{21}||S_{12}|}$$

and B_{1f} is

$$B_{1f} = 1 + |S11|^2 - |S22|^2 - |D|^2$$

Two-Port Gain Quantities

The following gain quantities are valid only for two-port circuits:

G_A (available gain) is the power gain obtained by optimally matching the output of the network.

G_P (power gain) is the power gain obtained by optimally matching the input of the network.

G_T (transducer gain) shows the insertion effect of a two-port circuit. This quantity is used in amplifier design.

G_{umx} (maximum unilateral transducer power gain)

G_{max} (maximum available gain) shows the transducer power gain when there exists a simultaneous conjugate match at both ports.

G_{msg} (maximum stable gain) shows the gain that can be achieved by resistively loading the two-port such that $k = 1$ and then simultaneously conjugately matching the input and output ports. For conditionally stable two-ports, you can approach the maximum stable gain as you reduce the input and output mismatch. If you attempt a simultaneous conjugate match and $k < 1$, the two-port oscillates.

Equations for Two-Port Gain Calculations

G_A (Available Gain) Equations

Available gain equations for output conjugately matched.

$$G_A = \frac{|S_{21}|^2\left(1 - |\Gamma_S|^2\right)}{|1 - S_{11}\Gamma_S|^2\left(1 - |\Gamma_2|^2\right)}$$

Where

$$\Gamma_2 = S_{22} + \frac{S_{12}S_{21}\Gamma_S}{1 - S_{11}\Gamma_S}$$

Note: When you use the S-parameter *Direct Plot Form*, Γ_S is set to zero, and therefore available gain (G_A) is plotted as:

$$G_A = \frac{|S_{21}|^2}{1 - |S_{22}|^2}$$

To plot G_A for non-zero values of Γ_S, use the Analog Design Environment calculator.

G_P (Power Gain) Equations

Power gain equations for input conjugately matched.

$$G_P = \frac{|S_{21}|^2\left(1-|\Gamma_L|^2\right)}{|1-S_{22}\Gamma_L|^2\left(1-|\Gamma_1|^2\right)}$$

Where

$$\Gamma_1 = S_{11} + \frac{S_{12}S_{21}\Gamma_L}{1-S_{22}\Gamma_L}$$

Note: When you use the S-parameter *Direct Plot Form*, Γ_L is set to zero, and therefore the power gain, G_P, is plotted as:

$$G_P = \frac{|S_{21}|^2}{1-|S_{11}|^2}$$

To plot G_P for nonzero values of Γ_L, use the Analog Design Environment calculator.

G_T (Transducer Gain) Equations

$$G_T = \frac{\left(1-|\Gamma_S|^2\right)|S_{21}|^2\left(1-|\Gamma_L|^2\right)}{|(1-S_{11}\Gamma_S)(1-S_{22}\Gamma_L)-S_{12}S_{21}\Gamma_S\Gamma_L|^2}$$

Note: When using the S-parameter *Direct Plot Form*, the Analog Design Environment assumes that the source (Γ_S) and load (Γ_L) reflection coefficients are zero. G_T, therefore, plots the insertion gain.

$$G_T = |S_{21}|^2$$

Using the calculator, you can plot G_T and specify the source and load terminations.

G_{max} (Maximum Available Gain) Equations

For $K > 1$

$$G_{max} = \left|\frac{S_{21}}{S_{12}}\right|\left[K - \sqrt{K^2 - 1}\right]$$

For $K <= 1$

$$G_{max} = \frac{|S_{21}|}{|S_{12}|}$$

Where K is the stability factor, K_f

G_{umx} (Maximum Unilateral Transducer Power Gain) Equation

$$G_{umx} = \frac{|S_{21}|^2}{(1 - |S_{11}|^2)(1 - |S_{22}|^2)}$$

G_{msg} (Maximum Stable Power Gain) Equation

$$G_{msg} = \frac{|S_{21}|}{|S_{12}|}$$

Two-Port Network Circles

NC plots constant noise contours at the input of a two-port circuit. *GAC* plots constant gain contours at the input port, and *GPC* plots constant gain contours at the output port. Gain contour values reflect an optimum match at the opposing port.

Noise and Gain circles can be plotted at a single dB value for a range of frequencies or at a single frequency for a range of dB values. If you do not enter values for the frequency range, a circle is plotted for every simulated frequency for which a circle with the specified value exists.

SSB plots stability circles at the input port, and *LSB* plots stability circles at the output port. You can also specify a limited frequency range for these contours.

Equations for Two-Port Network Circle

NC (Noise Circle) Equations

$$N_i = \frac{(F_i - F_{min})\left|1 + \Gamma_{min}\right|^2}{4r_n}$$

where

$$\Gamma_{min} = G_{min}$$

The center is calculated using

$$C_N = \frac{\Gamma_{min}}{1 + N_i}$$

The radius is calculated using

$$r_N = \sqrt{\frac{N_i^2 + N_i\left(1 - \left|\Gamma_{min}\right|^2\right)}{1 + N_i}}$$

Where i is the index number

GAC (Available Gain Circle) Equations

The center is calculated using

$$C_A = \frac{g_a(S_{11}{}^* - D^* S_{22})}{1 + g_a\left(\left|S_{11}\right|^2 - |D|^2\right)}$$

The radius is calculated using

$$r_A = \frac{\sqrt{1 - 2K_f\left|S_{21}S_{12}\right|g_a + \left|S_{12}S_{21}\right|^2 g_a^2}}{\left|1 + g_a\left(\left|S_{11}\right|^2 - |D|\right)^2\right|}$$

Where

$$G_a = \frac{G_A}{\left|S_{21}\right|^2}$$

And

$$D = S_{11}S_{22} - S_{12}S_{21}$$

GPC (Power Gain Circle) Equations

The center is calculated

$$C_P = \frac{g_p(S_{22}{}^* - D^* S_{11})}{1 + g_p\left(\left|S_{22}\right|^2 - |D|^2\right)}$$

The radius is calculated using

$$r_P = \frac{\sqrt{1 - 2K_f\left|S_{21}S_{12}\right|g_p + \left|S_{12}S_{21}\right|^2 g_p^2}}{1 + g_p\left(\left|S_{22}\right|^2 - |D|^2\right)}$$

Where

$$g_p = \frac{G_P}{\left|S_{21}\right|^2}$$

And

$$D = S_{11}S_{22} - S_{12}S_{21}$$

LSB (Load Stability Circle) Equations

The center is calculated using

$$C_L = \frac{S_{11}D^* - S_{22}{}^*}{|D|^2 - \left|S_{22}\right|^2}$$

The radius is calculated using

$$r_L = \left| \frac{S_{12}S_{21}}{|D|^2 - |S_{22}|^2} \right|$$

Where

$$D = S_{11}S_{22} - S_{12}S_{21}$$

SSB (Source Stability Circle) Equations

The center is calculated using

$$C_S = \frac{S_{22}D^* - S_{11}^*}{|D|^2 - |S_{11}|^2}$$

The radius is calculated using

$$R_S = \left| \frac{S_{12}S_{21}}{|D|^2 - |S_{11}|^2} \right|$$

Where

$$D = S_{11}S_{22} - S_{12}S_{21}$$

Equation for VSWR (Voltage Standing Wave Ratio)

VSWR is calculated from the S-parameters. You can plot the *VSWR* at any port in the circuit on a rectangular chart.

$$VSWR_i = \frac{1 + |S_{ii}|}{1 - |S_{ii}|}$$

Where `i` is the port number.

Equation for ZM (Input Impedance)

You can plot input impedance if all other ports are matched.

$$Z_m = \frac{1 + S_{ii}}{1 - S_{ii}} R$$

Where `R` the reference impedance of the port of interest and `i` is the port number.

Equation for GD (group delay)

GD (group delay) approximates the derivative of the phase with respect to frequency, normalized to 360 degrees. Units for group delay are in seconds.

$$G_d \cong \frac{-d\phi}{d\omega}$$

Group delay is calculated from the phase of the corresponding S-parameter (for example, GD_{21} corresponds to S_{21}).

References

To learn the technical details about how SpectreRF converts S-parameter data to a time-domain description, see the article “Robust rational function approximation algorithm for model generation,” by C. P. Coelho, J. R. Phillips, and L. M. Silveira. This article appeared in the proceedings of the 36th Design Automation Conference, New Orleans, LA, June 1999.

11

Cosimulation with MATLAB® and Simulink®

This chapter describes how to set up and use a cosimulation link between the MATLAB® and Simulink® system-level simulation environment and Virtuoso® Spectre® circuit simulator RF analysis (SpectreRF). The sections in this chapter are:

- Introduction to Cosimulation with MATLAB on page 1616
- Software Requirements on page 1616
- Setting Up and Running a Cosimulation on page 1616
- Connecting the Coupler Block Into the System-Level Simulink Schematic on page 1617
- Determining How You Want to Start and Run the Cosimulation on page 1621
- Generating a Netlist for the Lower-Level Block on page 1621
- Running the Cosimulation on page 1630
- MATLAB Support Matrix on page 1632

Introduction to Cosimulation with MATLAB

Cosimulation combines the best of system-level simulation with lower-level analog and RF simulation. Simulink provides large libraries of DSP algorithms for generating complicated signals and post processing while SpectreRF supports transient and envelope analysis of common RF and communication circuits such as mixers, oscillators, sample and holds, and switched capacitor filters at both the transistor and behavioral levels.

Cosimulation makes it easier

- To detect concept errors early
- To detect design flaws before tape-out
- To quickly correct issues and re-simulate

The system-level design in Simulink serves as a golden reference. System designers can use the Simulink system-level design as a testbench for implementing and simulating the design. Unfortunately, when system-level designs are simulated by themselves, the effects originating from subsystems are often not considered. With cosimulation, system designers can create low-level models of critical analog blocks and use these models one at a time to analyze the performance impact of individual blocks on the system-level simulation.

Software Requirements

Cadence recommends that you use Cadence software version MMSIM 7.1 or later.

Setting Up and Running a Cosimulation

To prepare for and run a cosimulation, you

1. Connect the coupler block into the system-level Simulink schematic.
2. Determine how you want to start and run the cosimulation.
3. Generate a netlist for the lower-level block that reflects how you want to start and run the cosimulation.
4. Run the cosimulation.

These steps are described in greater detail in the following sections.

Before you continue, however, be sure that the programs are ready to run.

➤ Add the SpectreRF and MATLAB/Simulink engine install paths to the `MATLABPATH` environment variable.

You can automate this step by adding the appropriate command to your `.cshrc` file. For example, if you are using a C shell, you can add the command

```
setenv MATLABPATH `cds_root spectre`/tools/spectre/simulink:${MATLABPATH}
```

Connecting the Coupler Block Into the System-Level Simulink Schematic

To prepare the Simulink part of the design for cosimulation,

1. Start MATLAB, by typing

   ```
   matlab &
   ```

   The Simulink library opens.

   This library contains the coupler module (distinguished by the Cadence logo and labeled `SpectreRF Engine`). You can insert the coupler module in any Simulink design by dragging and dropping it from this library.

2. Open your testbench or high-level design.

3. Drag and drop the `SpectreRF Engine` coupler block into the testbench and place it in the design.

4. Connect the `SpectreRF Engine` block into the design.

   To make a signal connection, move the mouse pointer over the module pin. The pointer changes to a cross. Press the left mouse button, move the cursor to the destination pin and release the button.

5. Double-click the `SpectreRF Engine` block.

This opens the Function Block Parameters: SpectreRF Engine window where you can edit the coupler block's parameter values.

The fields have the following meanings:

Field	Meaning
Number of input pins	The number of input pins to the `SpectreRF Engine` block (0…100). The input to the block is the input signal from the Simulink design.
Number of output pins	The number of output pins from the `SpectreRF Engine` block (0…100). The output from the block is the signal generated by the SpectreRF simulator.
Frame size	The number of samples per frame (any positive integer or -1; default: -1). Allows you to use the coupler in a loop where frame size inheritance does not work. Simulink automatically detects conflicting frame sizes. Set this value to -1 to inherit the frame size from the input.

Sample time	Sample period of the block, in seconds (default: -1). The *Sample time* is the frame period, not the time between the samples inside the frame. Allows the coupler to have only outputs, where the Simulink coupler acts as a source and defines a *Sample time*. Also allows the coupler to have only inputs. Set this value to -1 to inherit the sample period of the connected blocks. See the MATLAB and Simulink *Help* menu for more information on *Sample time* propagation.
Socket port	The number of the service that is identified by TCP/IP. Normally, the system reverses port numbers less than 1024, so this value can range from 1024 to 65535 (default: 38520).
Simulation response timeout	Maximum time, in seconds, to wait for an answer from the SpectreRF simulator during simulation. Increase this time if the simulator requires a long calculation time for each sample or frame (default: *120*).
Socket mode	TCP or UDP. Choose UDP if frame size is less 50. TCP and UDP are two different translate protocols in TCP/IP. Without CRC, UDP is usually faster in a good net environment with small data packages and TCP is usually better for large data packages.
Spectre command	Used to run a SpectreRF simulation. This parameter enables a use mode where SpectreRF is called internally by the Simulink simulation.
Show engine port labels	If checked, the `SpectreRF Engine` shows label information.

6. Set the number of input pins and output pins as well as any other parameters you need to set.

7. Note the Spectre command field but do not change the value now.

   Depending on how you choose to run the cosimulation, you might need to return to this field and type in a `spectre` command.

8. Click *OK*.

   The form closes and the `SpectreRF Engine` block is updated with the correct number of pins.

9. (Optional) Make other necessary changes in the Simulink testbench window.

For example,

- Choose *Format – Port/Signal Displays – Signal Dimensions*.

  This switches the signal dimension display on so you can see details of framed signals.

- Choose *View – Simulink library* to display the library browser. Double-click *Math Operations*.

  The library appears. Here you can select Simulink converters, such as a `Complex to Real-Imag` block. Drag and drop blocks into the schematic and connect them as necessary.

Double-click any placed blocks and set appropriate values.

10. Choose *Simulation – Configuration Parameters*.

The Configuration Parameters form opens.

11. In the *Stop time* field, set the stop time.

12. Click *OK*.

The Configuration Parameters form closes.

13. Save the modified design by choosing *file – save as*.

This completes the necessary modifications to the Simulink testbench.

Determining How You Want to Start and Run the Cosimulation

There are three ways to run a cosimulation, after all the setup is finished.

1. You can start the two applications (SpectreRF and MATLAB) separately.

   This method is appropriate if you need to be able to modify both the system-level Simulink design and the analog circuit.

2. You can start ADE and arrange to have MATLAB start automatically.

   This method is appropriate if you are an analog design who needs to validate a circuit with system-level design input and output.

3. You can start MATLAB and arrange to have SpectreRF start automatically.

   This method is appropriate if you are a system-level designer, because, after set up, you can use the SpectreRF Engine just like another block in Simulink.

The setup differs for each of these approaches so it is useful to decide which is most appropriate for your design before continuing.

Generating a Netlist for the Lower-Level Block

The previous section describes how to insert a coupler block into the system-level design. That defines one end of the connection but you must still establish a connection with the lower-level analog block that is simulated by SpectreRF. To do that, you insert a `cosim` statement into the netlist, either by hand or by using the ADE environment.

Preparing the Netlist When Using ADE

This section describes how to prepare a netlist for cosimulation using the Analog Design Environment (ADE).

1. Start the virtuoso tool.

```
virtuoso &
```

The recommended version of `virtuoso` is release IC6.1.4 or later.

2. Open the schematic view of the cell.

   For example,

3. Open the Virtuoso Analog Design Environment from the schematic editor by choosing *Launch – ADE L*.

4. Choose *Setup – Matlab/Simulink – Setting*.

   The Cosimulation Options form appears.

5. In the Cosimulation Options form, click the *Select* button located beside *Cosimulation inputs*. Switch to the schematic viewer, where you see the following information below the schematic.

   ```
   Select source instance as cosimulation inputs. Press Esc when done.
   ```

   Select the sources (which are connected to the outputs from the Simulink level of the design), then press the *Esc* key.

6. In the Cosimulation Options form, click the *Select* button beside *Cosimulation outputs*. Switch to the schematic viewer, where you see the following information below the schematic.

```
Select Net/Terminal as cosimulation outputs. Press Esc when done
```

Select the outputs (which are connected to the inputs at the Simulink level of the design), then press the *Esc* key.

The Cosimulation Options form looks something like this:

7. Ensure that the value of *Cosimulation socket port* is the same as the port value of the `SpectreRF Engine` block defined in "Connecting the Coupler Block Into the System-Level Simulink Schematic" on page 1617.

8. In the Cosimulation Options form, turn on *Enabled*.

9. Examine the possible values for the *Start MATLAB* field.

Value	Behavior
now	ADE launches a MATLAB session and opens a MATLAB desktop that is no different than it would be if you launched it manually. When you use the *now* value, you must start the simulation in the Simulink design before starting the SpectreRF simulation. Using the *now* value is a good way to test whether ADE can start MATLAB and open the design successfully.
before Simulation	ADE launches an internal MATLAB session and no MATLAB desktop appears. ADE opens the specified MATLAB design and initiates the MATLAB simulation before Spectre simulation starts. ADE starts MATLAB once but cannot start it again before you close the first session. Cadence suggests using the *now* value as a test before you run with the *before Simulation* value. The *now* and *before Simulation* values share one log file, which can be opened by choosing *Setup – Matlab/ Simulink – Log file*. You can use the log file to monitor the simulation when no MATLAB desktop is visible.
no	The *MATLAB start command*, *MATLAB startup directory*, and *MATLAB design name* fields become active. When you use the *no* value, you must start the simulation in the Simulink design before starting the SpectreRF simulation.

10. Continue the process of preparing the netlist according to how you want to start the applications that run the cosimulation.

To use this starting method...	Follow the guidance in this section...
Start the two applications (SpectreRF and MATLAB) separately.	"Preparing to Start the Two Applications Separately" on page 1625
Start ADE and arrange to have MATLAB start automatically.	"Preparing to Start ADE Manually and MATLAB Automatically" on page 1625

To use this starting method...	Follow the guidance in this section...
Start MATLAB and arrange to have SpectreRF start automatically.	"Preparing to Start MATLAB Manually and SpectreRF Automatically" on page 1627

Preparing to Start the Two Applications Separately

1. Set the *start MATLAB* value to *no*.
2. Click *OK* to close the Cosimulation Options form.
3. Use the Choosing Analyses form to set up the analysis.

   The *Stop Time* can be any value. The SpectreRF simulator synchronizes the stop time with Simulink.

   The Choosing Analyses window looks something like this:

4. Create a netlist and make sure the Simulink `cosim` statement appears in it.

Preparing to Start ADE Manually and MATLAB Automatically

1. Set the value of the *Start MATLAB* field to *before Simulation* in the Cosimulation Options form.

2. Type the name of the MATLAB design into the *MATLAB design name* field.

   Such designs have an extension of `.mdl`. For example, `env_d.mdl`. The MATLAB design must be in a location that is included in the `MATLABPATH` environment variable.

3. Set the value of the *Start MATLAB* field to *now*.

   A new MATLAB application starts and helps you open the Simulink design.

4. Double-click the `SpectreRF Engine` in the design and set the parameters.

   a. Set the *Sample time*. The Sample time can be set to -1, if you are uncertain about the appropriate time to use.

   b. Set *Socket Port* to the value you set in the Cosimulation Options form.

   c. Leave the *Spectre Command* field empty.

   This field is used when you wish to start SpectreRF automatically after starting MATLAB manually.

   d. Click *OK* to close the SpectreRF Engine form.

5. In the Cosimulation Options form, set *Start MATLAB* to *before Simulation*.

6. Click *OK* to close the Cosimulation Options form.

7. Use the Choosing Analyses form to set up the analysis.
8. Choose *File - Save* to save the design in MATLAB. Then exit from the MATLAB that was opened by ADE L.

Preparing to Start MATLAB Manually and SpectreRF Automatically

1. Set the value of the *Start MATLAB* field to *now*.

   A new MATLAB application starts and helps you open the Simulink design.

2. Double-click the `SpectreRF Engine` in the design and set the parameters.

   a. Set the *Sample time*. The Sample time can be set to -1, if you are uncertain about the appropriate time to use.

   b. Set *Socket Port* to the value you set in the Cosimulation Options form.

   c. Type a command similar to the following into the *Spectre Command* field.

   `spectre` *`netlist_file`* `=log spectre.out`

   The Function Block Parameters: SpectreRF Engine window looks something like this:

   d. Click *OK* to close the Function Block Parameters: SpectreRF Engine form.

3. In the Cosimulation Options form, set *Start MATLAB* to *no*.

4. Click *OK* to close the Cosimulation Options form.
5. Use the Choosing Analyses form to set up the analysis.
6. Choose *File - Save* to save the design in MATLAB. Then exit from the MATLAB that was opened by ADE L.

Preparing the Netlist Without Using a Graphical User Interface

The steps in this section are necessary only if SpectreRF needs to start MATLAB automatically. If you are using the “Start MATLAB and arrange to have SpectreRF start automatically” starting method, ensure that there is no `cosim` statement in the netlist.

1. Edit the netlist file.

   For example,

   ```
   vi tutorial1/tutorial1.scs.
   ```

2. Add a `cosim` statement to the netlist.

   For example,

   ```
   matlab cosim server="bj2lnx20"  port=38525 inputs=["VI:wave" "VQ:wave"]
   outputs=[TX_AFE_OUT_BB_I TX_AFE_OUT_BB_Q]
   ```

   The parameters for the `cosim` statement are described below:

Table 11-1 Parameters of the cosim Statement

Parameter	Meaning
`design`	Refers to the file Simulink associates with the netlist. `design="env_d.mdl"` means the testbench whose file is `env_d.mdl`.

Table 11-1 Parameters of the cosim Statement, *continued*

`inputs`	Input vector to identify the flow from MATLAB to `SpectreRF Engine`. The format is `[“instance_name:wave” “instance_name1:wave” …]` where `wave`, is a key word. **Note:** The sequence of instances in square brackets follows the same one as the `SpectreRF Engine` port labels. This label information can be found by double-clicking the `SpectreRF Engine` and then by checking the *Show engine port labels* at the bottom in Function Block Parameters of the `SpectreRF Engine`. After the inputs are set in the netlist, the sources associated with the inputs are meaningless and their parameters do not affect the simulation.
`outputs`	Output vector from `SpectreRF Engine` to MATLAB. This vector can be the net name or terminal name of interest. If net name is used, only voltage is given; if terminal name is used, only current is output. To get current in an ENVLP analysis, you must add probes in the topology.
`port`	The port on the server running MATLAB. Its value should be the same as the value in the `SpectreRF Engine` in your current MATLAB design.

`server`	The name of the machine that MATLAB starts. It can be set with the machine name or with the IP address of that machine. The accepted form is
	`server = ”155.110.110.110”`
	and
	`server = “bj2lnx20”`
	`server=localhost` means that Spectre and MATLAB use the same machine.

`silent`	Tells MATLAB whether to open the window during simulation. If `silent=yes`, the testbench window is opened.
`timeout`	The period of time MATLAB spends waiting for a response from the Spectre simulator. `timeout=60` stops MATLAB if it does not receive any response.

3. Close and save the netlist.

   After the above steps, the netlist is ready for simulation.

Running the Cosimulation

With the coupler connected into the MATLAB design and with an appropriate netlist for the low-level design, you are ready to run the cosimulation. Note that these starting methods work only when the design is prepared as described earlier in this chapter.

To use this starting method...	Follow the guidance in this section...
Start the two applications (SpectreRF and MATLAB) separately.	"Starting the Two Applications Separately" on page 1630
Start SpectreRF and arrange to have MATLAB start automatically.	"Starting SpectreRF Manually and MATLAB Automatically" on page 1631
Start MATLAB and arrange to have SpectreRF start automatically.	"Starting MATLAB Manually and SpectreRF Automatically" on page 1631

Starting the Two Applications Separately

1. Open the high-level design or testbench in the MATLAB design window.

2. In the MATLAB design window, choose *Simulation – Start*.

   The MATLAB desktop issues the following message.

   ```
   block 'modified/SpectreRF Engine': (COSIM_OK) Waiting for incoming connection
   on port 38520, timeout: 60 sec ...
   ```

   Then quickly do step 3.

   Note: The time interval between step 2 and step 3 must be within the *Simulation response timeout* defined as Function Block Parameters of the `SpectreRF Engine` (by double-clicking the `SpectreRF Engine`).

3. In the ADE window, click *Netlist and Run* or enter a spectre command at the command line. For example,

   ```
   spectre tutorial1/tutorial1.scs
   ```

   The cosimulation begins. The MATLAB desktop issues a message similar to the following when simulation ends.

   ```
   block 'modified/SpectreRF Engine': (COSIM_OK) Simulation finished
   ```

Starting SpectreRF Manually and MATLAB Automatically

To run the cosimulation by starting SpectreRF,

- If you are using ADE, open the low-level design in the ADE window.

  a. Click *Netlist and Run*.

  The cosimulation begins. The MATLAB desktop issues a message similar to the following when simulation ends.

  ```
  block 'modified/SpectreRF Engine': (COSIM_OK) Simulation finished
  ```

- If you are using standalone SpectreRF,

  a. At the command line, enter a command similar to the following.

  ```
  spectre tutorial1/tutorial1.scs
  ```

  The cosimulation begins.

Starting MATLAB Manually and SpectreRF Automatically

a. Start MATLAB.

```
matlab&
```

b. Choose *Simulation – Start*.

The MATLAB desktop issues a message similar to the following.

```
block 'env_d/SpectreRF Engine': (COSIM_OK) Waiting for incoming connection
on port 38525, timeout: 60 sec ...
block 'env_d/SpectreRF Engine': (COSIM_OK) Launch Spectre with commad
'spectre tutorial2.scs =log spectre.out'”
```

The cosimulation runs.

c. After the cosimulation finishes, review the MATLAB/Simulink output, close the MATLAB desktop, and exit from MATLAB.

MATLAB Support Matrix

Table 11-2 MATLAB Toolbox

MMSIM	MATLAB Release	Supported Platform	
		Linux 32 / 64	***Solaris 64***
	R2007a	Supported	Supported
	R2007b	Supported	Supported
	R2008a	Supported	Supported
MMSIM12.1	R2008b	Supported	Supported
	R2009a	Supported	Supported
	R2009b	Supported	Supported
	R2010a	Supported	Not Supported [1]
	R2010b	Supported	Not Supported [1]
	R2011a	Supported	Not Supported [1]
	R2011b	Supported	Not Supported [1]
	R2012a	Supported	Not Supported [1]

1. From MATLAB versions 2010a onwards, Solaris Platform is no longer supported.

Table 11-3 MATLAB Cosimulation

MMSIM	MATLAB Release	Supported Platform	
		Linux 32 / 64	***Solaris 64***
	R2007a	Supported	Supported
	R2007b	Supported	Supported
	R2008a	Supported	Supported
MMSIM12.1	R2008b	Supported	Supported
	R2009a	Supported	Supported
	R2009b	Supported	Supported
	R2010a	Supported	Not Supported [1]

R2010b	Supported	Not Supported [1]
R2011a	Supported	Not Supported [1]
R2011b	Supported	Not Supported [1]
R2012a	Supported	Not Supported [1]

1. From MATLAB versions 2010a onwards, Solaris Platform is no longer supported.

A

Design Workshops

This section explains how to set up your software and environment to run the examples in this user guide. The section also describes the procedure for accessing the latest version of Virtuoso® Spectre® circuit simulator RF analysis (SpectreRF), which ships in the MMSIM release stream.

Before you perform the SpectreRF analyses, you need to set up the component files and start the Cadence® software.

Setting Up Environment Variables and the Path Statement

Set the following environment variables for SpectreRF.

```
setenv CDS_rfExamples ‘cds_root icms‘
setenv CDS_Netlisting_Mode Analog
```

`CDS_rfExamples` defines the path to the piece-wise linear (PWL) model files in the Cadence software installation hierarchy.

Using SpectreRF from the MMSIM Hierarchy

Starting with the 5.1.41 USR1 release, you have the option to obtain the Virtuoso Spectre Circuit Simulator (Spectre), SpectreRF, and other simulators, from the MMSIM release stream. While the version of Spectre and SpectreRF that shipped with 5.1.41 USR1 continue to ship with the remaining 5.1.41 releases, documentation for new features and most bug fixes are provided exclusively with the MMSIM release stream. The first MMSIM release, MMSIM6.0, was released at the same time as the 5.1.41 USR1 update.

You must download and install the MMSIM simulators in a separate simulation installation hierarchy than the hierarchy you use for the Cadence software.

The SpectreRF examples use the following environment variables to point to the two installation hierarchies,

`MMSIMHOME`	Path to the installation hierarchy for the MMSIM simulators
`CDSHOME`	Path to the installation hierarchy for the Cadence software

To use the MMSIM simulators, put all paths to the MMSIM simulators such as

```
<path_to_MMSIM_simulators>/tools/bin
```

before any paths to the dfII software in your $PATH statement.

The SpectreRF examples use the following path

```
path = ( $MMSIMHOME/tools/bin \
$CDSHOME/tools/bin  \
$CDSHOME/tools/dfII/bin \
$CDSHOME/tools/java \
$CDSHOME/tools/java/bin \
$path )
```

Accessing the Most Current SpectreRF Documentation

The documentation for the latest features of SpectreRF is always found in the MMSIM hierarchy. If you are using the MMSIM version of SpectreRF, access the SpectreRF documentation from the MMSIM hierarchy.

Important

Note that the help buttons on the forms lead you to the IC version of the documentation.

Creating a Local Editable Copy of the rfExamples Library

Make a copy of the *rfExamples* library and save it in a directory in your account. Change the name and access mode of your local copy so you have write access to your local copy of the library. This permits you to edit the schematics and other files in this library as you follow the examples.

The *rfExamples* library is located at

```
<CDSHOME>/tools/dfII/samples/artist/rfExamples
```

where `CDSHOME` is the installation directory for your Cadence software.

the Cadence Libraries

The Cadence Libraries are defined in the UNIX text file *cds.lib*. You can edit this file by using the library path editor or by using a UNIX shell window.

Using the Library Path Editor

To access the Library Path Editor, use the following procedure.

1. In a UNIX window, type `virtuoso &` to start the Cadence software.

   The Command Interpreter Window (CIW) appears.

2. In the CIW, choose *Tools – Library Path Editor*.

   The Library Path Editor appears.

Library Path Editor Window

In the Library Path Editor, follow the instructions at the bottom of the form.

3. Type a name for each required library and the associated path to the library in the software installation hierarchy.You need the libraries listed in "Library Path Editor Window" on page 1638.

4. Choose *File – Save* to save your definitions in the `cds.lib` file.

   When you are using Open Access (OA), in addition to saving the `cds.lib` file an OA `lib.defs` file is also saved.

5. Exit the Library Path Editor.

Using a UNIX Shell Window

To set up the libraries in a UNIX shell window, use the following procedure:

1. In a UNIX shell window, open the `cds.lib` file for editing using *vi*, *emacs* or a similar text editor.

The `cds.lib` file is in your installation directory.

2. In the `cds.lib` file, define the Cadence provided libraries.
3. Define a user library where the sample circuits can be tested.

   You can label your test library any name you choose. This example assumes that you have called the library *my_rfExamples*. The name *my_dir* represents the directory into which you copied the *rfExamples* library.

   You must use the names *basic*, *sample* and *analogLib*. You cannot rename these libraries.

   After these steps, the definitions in the *cds.lib* file look similar to the following:

```
DEFINE rfExamples $CDSHOME/tools/dfII/samples/artist/rfExamples
DEFINE analogLib $CDSHOME/tools/dfII/etc/cdslib/artist/analogLib
DEFINE ahdlLib $CDSHOME/tools/dfII/samples/artist/ahdlLib
DEFINE rfLib $CDSHOME/tools/dfII/samples/artist/rfLib
DEFINE my_rfExamples /home/belinda/my_libs/rfExamples
DEFINE sample $CDSHOME/tools/dfII/samples/cdslib/sample
DEFINE basic $CDSHOME/tools/dfII/etc/cdslib/basic
DEFINE passiveLib $CDSHOME/tools/dfII/samples/artist/passiveLib
DEFINE pllLib $CDSHOME/tools/dfII/samples/artist/pllLib
```

Setting Up For Simulation

Opening a Circuit in the Schematic Window

To open a circuit in the schematic window,

1. In the CIW, choose *File – Open*.

   The Open File form appears.
2. In the Open File form, choose *my_rfExamples* in the *Library Name* menu.
3. Choose the circuit you want to work with from the list in the *Cells* list box. the *ne600p* circuit is used in this example.

The completed Open File form appears like the one below.

4. Click *OK*.

The Schematic window opens to display the circuit. In this case, the *ne600p* mixer appears.

Opening the Simulator Window

To open the Simulator window,

1. In the Schematic window, choose *Launch– ADE L*.

The ADE window opens. This window is also called the Virtuoso® Analog Design Environment.

Note: You can also choose *Tools – Analog Environment – Simulation* in the CIW to open the ADE window without opening the design. You can open the design later by choosing *Setup – Design* in the ADE window and then choosing *ne600p* in the Choosing Design form.

Choosing Simulator Options

To set up the simulator options,

1. Choose *Setup – Simulator/Directory/Host* in the ADE window.

   The Choosing Simulator/Directory/Host form appears.

2. In the Choosing Simulator/Directory/Host form, specify the following:

   a. Choose *spectre* for the *Simulator*.

   b. Type the name of the project directory, if necessary.

c. Highlight the *local* or the *remote* button to specify the *Host Mode*.

d. (Optional) For remote simulation, type the name of the host machine and the remote directory in the appropriate fields.

The completed form appears like the one below.

3. In the Simulator/Directory/Host form, click *OK*.

Specifying Outputs to Save

To specify the outputs that you want to save,

1. In the ADE window, choose *Outputs – Save All.*

The Save Options form appears.

2. In the *Select signals to output section*, be sure *allpub* is highlighted.

Setting Up Model Libraries

To set up the model libraries,

1. In the ADE window, choose *Setup – Model Libraries*.

   The Model Library Setup form appears.

2. In the *Model Library File* field, type the full path to the model file including the file name, `<CDSHOME>/tools/dfII/samples/artist/models/spectre/rfModels.scs`.

   When you type the path to the model library, you must expand `<CDSHOME>` and type in the actual path to your software installation hierarchy.

   The completed form appears like the one below.

3. In the Model Library Setup form, click *Add*.
4. Click *OK*.

Editing Design Variable Values

To edit the design variable values,

1. Verify the current variables and their values in the Design Variables area in the ADE window.

2. In the ADE window, use the following procedure to set the design variables to the values required for the simulation.

   (See the description of each simulation for the required variable names and values.) This example changes the value of the variable `frf`.

   a. In the ADE window, choose *Variables – Edit*.

The Editing Design Variables form appears.

b. In the *Table of Design Variables*, select a variable to edit.

c. The variable name and value display in the *Name* and *Value (Expr)* fields.

d. In the *Value (Expr)* field, type `900M` (or `920M`) for the value of `frf` and click *Change*.

e. Edit the variable's value in the *Value (Expr)* field. Then click *Change*.

f. The variable value changes in the *Table of Design Variables.*

g. Repeat to edit more variables.

h. In the Editing Design Variables form, click *OK*.

i. View the new variable value in the *Design Variables* section of the ADE window.

Low-noise Amplifiers

To use this chapter, you must be familiar with the SpectreRF simulator analyses and have a working knowledge of low-noise amplifier design. For more information about the SpectreRF simulator analyses, se.

Analyses and Measurement Examples in this Chapter

This section shows you how to simulate and plot the following:

- Voltage Gain
- Output Voltage Distribution
- S-parameter Analysis, including the Voltage Standing Wave Ratio
- S-parameter Noise Analysis, including
- Noise Figure and Minimum Noise Figure
- Equivalent Noise Resistance
- Load and Source Stability Circles
- Noise Circles
- Noise Figure Calculations with the Pnoise Analysis
- The 1dB Compression Point
- The Third-Order Intercept Point
- Conversion Gain and Power Supply Rejection with the Periodic Transfer Function Analysis

Simulating the lnaSimple Example

Opening the lnaSimple Circuit in the Schematic Window

1. In the Command Interpreter Window (CIW), choose *File – Open*.

   The Open File form appears.

2. Choose *my_rfExamples* for *Library Name*.

3. Choose *lnaSimple* in the *Cells* list box.
4. Choose *schematic* for *View Name*.

   The filled-in form appears like the one below.

5. In the Open File form, click *OK.*

The Schematic window now shows the *InaSimple* schematic.

6. In the Schematic window, choose *Tools – Analog Environment*.

The Virtuoso Analog Design Environment (ADE) window appears.

Choosing Simulator Options

1. In the ADE window, choose *Setup – Simulator/Directory/Host*.

   The Choosing Simulator/Directory/Host form appears.

2. Choose *spectre* for *Simulator*.
3. Type the name of the project directory, if necessary.
4. Highlight the *Host Mode* that corresponds to your situation.
5. If you choose *remote* or *distributed*, fill in the additional fields as necessary.

The completed form looks like this.

6. In the Choosing Simulator/Directory/Host form, click *OK*.
7. In the ADE window, choose *Outputs – Save All*.

   The Save Options form appears.
8. In the *Select signals to output (save)* section, be sure *allpub* is highlighted.

9. Click *OK*.

   The Save Options form closes.

Setting Up Model Libraries

1. In the ADE window, choose *Setup – Model Libraries*.

   The Model Library Setup form appears.

2. In the *Model Library File* field, type the full path to the model file including the file name, `rfModels.scs`.

3. In the Model Library Setup form, click *Add*.

   The completed form looks like this.

4. In the Model Library Setup form, click *OK*.

Calculating Voltage Gain with PSS

The most important characteristic of an amplifier is its gain. In the following example, a PSS analysis determines the voltage gain of the low noise amplifier.

Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.

2. If necessary, set up the simulator options.

3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file.

4. In the ADE window, choose *Analyses – Disable* to disable any analyses you ran previously. (Check the ADE window to verify whether or not an analysis is enabled.)

Editing the Schematic

1. In the Schematic window, click the `RF` voltage source.

2. In the Schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form for the port appears.

3. If necessary, in the *Source Type* field type `sine` and click *OK* in the Edit Object Properties form.

4. In the Schematic window, choose *Design – Check and Save*.

Setting Up the PSS Analysis

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, highlight *pss*.

The form changes to let you specify data for the PSS analysis.

3. In the *Fundamental Tones* area, the *Beat Frequency* button is highlighted by default. Highlight the *Auto Calculate* button.

   The fundamental frequency for this analysis, which is `900M`, appears in the *Beat Frequency* field.

4. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type a reasonable value, such as 10, in the *Number of harmonics* field. (You can choose a different number.)

The PSS Choosing Analyses form looks like this.

5. Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.
6. Highlight *Enabled*, if necessary.
7. In the Choosing Analyses form, click *OK*.

Running the Simulation

1. To run the PSS analysis, choose *Simulation – Netlist and Run* in the ADE window.

   The output log file appears and displays information about the simulation as it runs.
2. Look in the output log file to be sure the simulation completes successfully.

Plotting Voltage Gain

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.
2. In the Direct Plot form, highlight *Replace* for *Plotting Mode*.
3. Highlight *pss* for *Analysis*.
4. Highlight *Voltage Gain* for *Function*.

The Direct Plot form appears below.

The Direct Plot form changes to reflect your choice.

Notice that the *Select* cyclic field changes to display *Output and Input nets*.

Notice the message at the bottom of the form *Select Input Harmonic on this form...*.

5. Highlight *dB20* for *Modifier*.
6. Highlight *900M* in the *Input Harmonic* list box.
7. Calculate the voltage gain as *Vout/Vin* by following the instructions at the bottom of the Direct Plot form as illustrated in the next two steps—click first on the output net then click the input net.
8. Following the message at the bottom of the form,

Select Numerator Output Net on schematic...

Select the `Vout` output net.

9. Following the message at the bottom of the form,

Select Denominator Input Net on schematic...

Select the `Vin` input net at the RF source.

The voltage gain plot appears in waveform window.

10. To display the voltage gain value, in the waveform window, place your cursor at the top of the line representing the 900 MHz fundamental (the tallest, leftmost line).

The x and y axis values displayed in the lower-left corner of the waveform window show that the voltage gain at 900 MHz is 11.45 dB.

Calculating Output Voltage Distribution with PSS

The output voltage distribution of an amplifier is important in analyzing intermodulation distortion when the two RF input signals are close together. To model this situation, you modify a parameter to make the voltage source component generate two separate frequencies.

Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file
4. In the ADE window, choose *Analyses – Disable* to disable any analyses you ran previously. (Check the ADE window to verify whether or not an analysis is enabled.)

Editing the Schematic

1. In the Schematic window, click the `RF` voltage source.

2. In the Schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears.

3. In the *Second frequency name* field, type `fund2`, or any name that you choose.
4. In the *Frequency 2* field, type `920M`.
5. In the *Amplitude 2 (dBm)* field, type `prf`.

   This sets the amplitude of the second tone equal to that of the first tone.
6. In the Edit Object Properties form, click *OK*.
7. In the Schematic window, choose *Design – Check and Save*.

Setting up the PSS Analysis

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. Click *pss* for the *Analysis* choice.

   The Choosing Analyses form changes to display fields for the PSS analysis.

3. In the *Fundamental Tones* area, highlight the *Auto Calculate* button. The *Beat Frequency* button is highlighted by default.

   The *Beat Frequency* field displays `20 M` because there are two RF tones—one at 900 M and the second at 920 M.

   The top of the Choosing Analyses form appears as follows.

4. In the *Output harmonics* cyclic field, choose *Number of Harmonics* and type `60` in the *Number of Harmonics* field.

   The largest common multiple of 900 Mhz and 920 MHz is 20MHz. Using 60 harmonics extends the results out to 1.2 GHz, but you can choose another number if you want.

5. Highlight *moderate* as the *Accuracy Defaults* value.
6. Make sure that *Enabled* is highlighted.
7. The bottom of the Choosing Analyses form appears as follows.

8. In the Choosing Analyses form, click *OK*.

Running the Simulation

1. In the ADE window, choose *Simulation – Netlist and Run* to run the PSS analysis.

   The output log file appears and displays information about the simulation as it runs.
2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the Output Voltage Distribution

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.
2. In the Direct Plot form, highlight *Replace* for *Plot Mode*.
3. Highlight *pss* for *Analysis*.
4. Highlight *Voltage* for *Function*.
5. Notice that the *Select* cyclic field displays *Net*.

6. Highlight *spectrum* for *Sweep* value.
7. Highlight *peak* for *Signal Level*.
8. Highlight *dB20* for *Modifier*.
9. Following the message at the bottom of the form,

   Select Net on schematic...

   Select the `Vout` amplifier output net.

In the waveform window, you can see spikes at 900 MHz, 920 MHz (-9.3 dB) and a third spike at 20 Mhz. The next strongest harmonics are lower than -36 dB.

S-parameter Analysis for Low Noise Amplifiers

Before you can perform S-parameter analyses of the low noise amplifier, you need to set up for the example.

Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file.

4. In the ADE window, choose *Analyses – Disable* to disable any analyses you ran previously. (Check the ADE window to verify whether or not an analysis is enabled.)

Editing the Schematic

1. In the Schematic window, select the `RF` voltage source.
2. In the Schematic window, choose *Edit – Properties – Objects*.

   The Edit Object Properties form appears.
3. In the Edit Object Properties form, be sure the *Source type* is set to `sine`, delete the *Frequency 2* value, if there is one, and click *OK*.

   The input voltage source now has only one frequency at `900MHz`. *Frequency 2* is zero.
4. In the Schematic window, choose *Design – Check and Save*.

Setting up the S-parameter Analysis

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.
2. In the Choosing Analyses form, click *sp* for *Analysis*.

   The form changes to let you specify data for an S-parameter analysis.
3. Highlight *Frequency* for *Sweep Variable*.
4. Highlight *Start-Stop* for *Sweep Range* and type `100M` for *Start* and `1.2G` for *Stop*.
5. In the *Sweep Type* cyclic field, choose *Linear.*
6. Click the *Number of Steps* button and type `100` in the *Number of Steps* field.
7. Highlight *Enabled*.

8. The filled-in form looks like this.

9. In the Choosing Analyses form, click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the ADE window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting S-Parameters

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.

2. Select *New SubWin* for *Plotting Mode*.
3. Highlight *sp* for *Analysis*.
4. Highlight *SP* for *Function*.

   The top of the Direct Plot form looks like this.

5. Highlight *Rectangular* for *Plot Type*.
6. Highlight *dB20* for *Modifier*.

The bottom of the SP Direct Plot form looks like this.

7. Following the message at the bottom of the form,

To plot, press Sij-button on this form...

Select the S-parameters to plot:

a. Click *S11*.

S11 is plotted in the waveform window.

b. Click *S22*.

S22 is plotted in the right half of the waveform window.

The values of S11 and S22 measure whether input and output are matched.

You can choose among several different plot types in the Direct Plot form. The following plots show S11 and S22 with the *Plot Type* set to *Polar*.

You can also choose S12 and S21 for plotting in the Direct Plot form. S12 is the reverse gain and a measure of isolation.

Plotting the Voltage Standing Wave Ratio

In the Direct Plot form do the following.

1. Highlight *New SubWin* for *Plotting Mode*.
2. Highlight *sp* for *Analysis*.
3. Highlight *VSWR* for *Function*.
4. Click *VSWR1*.

   VSWR1 is plotted in the waveform window.
5. Click *VSWR2*.

   VSWR2 is plotted in the right half of the waveform window.

The two voltage standing wave ratio plots display in the waveform window.

Linear Two-Port Noise Analysis with S-Parameters

Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file.
4. In the ADE window, choose *Analyses – Disable* to disable any analyses you ran previously. (Check the ADE window to verify whether or not an analysis is enabled.)

Editing the Schematic

1. In the Schematic window, click the `RF` voltage source.

2. In the Schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears.

3. In the *Source type* field, type `dc`.

4. Click *OK*.
5. In the Schematic window, choose *Design – Check and Save*.

Note on the isnoisy Parameter

The *isnoisy* parameter, which you can set on the Edit Object Properties form by using the *Generate noise?* field, specifies whether a resistor generates noise. If you set the parameter to `yes` or `no`, the parameter setting appears in the netlist. If you leave the parameter unspecified by using the blank value in the cyclic field, the default setting is `yes` but the parameter does not appear in the netlist.

Setting up the S-parameter Analysis

1. In the ADE window, choose *Analyses – Choose*.
2. In the Choosing Analyses form, click *sp* for the *Analysis* choice.
3. Click *Select* in the *Ports* pane. Then, in the schematic, click on the input port (`RF`)

and then on the output port (`PORT0`)

In response, `/RF /PORT0` appears in the Ports field.

4. Highlight *Frequency* for the *Sweep Variable*.
5. Highlight *Start-Stop* for the *Sweep Range*. Type `800M` in the *Start* field and `1.1G` in the *Stop* field.
6. In the *Sweep Type* cyclic field, choose *Linear* for the sweep type and highlight *Number of Steps*. Type `30` in the *Number of Steps* field.
7. The top of the SP Choosing Analyses form appears as follows.

8. Highlight *yes* for *Do Noise*.

The form changes to let you specify values for the *Output port* and *Input port* fields.

9. To select the Output port,
   a. Click the *Select* button next to the *Output port* field.
   b. In the Schematic window, select the output port.

      `/PORT0` displays in the *Output port* field.
10. To select the Input port,
   a. Click the *Select* button next to the *Input port* field.
   b. In the Schematic window, select the RF port.

      `/RF` displays in the *Input port* field.
11. For the *Mode*, select *Single-Ended*.
12. Make sure that *Enabled* is highlighted.

    The bottom of the SP Choosing Analyses form appears as follows.

13. In the Choosing Analyses form, click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the ADE window.

   The output log file appears and displays information about the simulation as it runs.
2. Look in the CIW for a message that says the simulation is completed successfully.

Plotting the Noise Figure and Minimum Noise Figure

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.

2. In the Direct Plot form, highlight *Replace* for *Plotting Mode*.

3. Highlight *sp* for *Analysis*.

4. Highlight *NF* for *Function*.

5. Highlight *dB10* for *Modifier*.

   The completed form looks like this.

6. Following the message at the bottom of the form,

   Press Plot button on this form...

Click *Plot*.

The plot for the noise figure appears in the waveform window.

To add the minimum noise figure plot to the waveform window, in the Direct Plot form, do the following:

1. Highlight *Append* for *Plot Mode*.
2. Highlight *sp* for *Analysis*.
3. Highlight *NFmin* for *Function*.
4. Highlight *dB10* for *Modifier*.

The completed form looks like this.

5. Following the message at the bottom of the form,

   Press Plot button on this form...

   Click *Plot*.

The plot for the minimum noise figure is added to the noise figure plot in the waveform window.

Plotting the Equivalent Noise Resistance

In the Direct Plot form, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *sp* for *Analysis*.
3. Highlight *Rn* for *Function*.

The completed form looks like this.

4. In the Direct Plot form, click *Plot*.

The equivalent noise resistance is plotted in the waveform window.

Plotting Load and Source Stability Circles

Load and source stability circles show the boundaries between values of load and source impedance that cause instability and values that do not. Either the inside or the outside of the circles might represent the stable region, depending on other values. If, as in the example below, the values of S-parameters S_{11} and S_{22} are less than one for a 50 Ohm impedance system, the center of the normalized Smith chart falls within the stable region.

To Plot the Load Stability Circles

In the Direct Plot form, do the following:

1. Highlight *Replace* for *Plot Mode.*
2. Highlight *sp* for *Analysis*.
3. Highlight *LSB* for *Function*.
4. Highlight *Z-Smith* for *Plot Type*.

   The Direct Plot form changes to accept *Frequency Range (Hz)* values.
5. For the *Frequency Range (Hz)* values
   a. Type `800M` in the *Start* field.
   b. Type `1G` in the *Stop* field.
   c. Type `100M` in the *Step* field.

   The Direct Plot form looks like this.

6. Following the message at the bottom of the form,

   Press Plot button on this form...

   Click *Plot*.

   The load stability circles are plotted.

To Plot the Source Stability Circles

In the Direct Plot form, do the following:

1. Highlight *Append* for *Plot Mode*.
2. Highlight *sp* for *Analysis*.
3. Highlight *SSB* for *Function*.
4. Highlight *Z-Smith* for *Plot Type*.
5. For the *Frequency Range (Hz)* values

a. Type `800M` in the *Start* field.

b. Type `1G` in the *Stop* field.

c. Type `100M` in the *Step* field.

The Direct Plot form looks like this.

6. Following the message at the bottom of the form,

Press Plot button on this form...

Click *Plot*.

The source stability circles are plotted in the waveform window along with the load stability circles.

7. To see the plot as it appears below, in the waveform window, choose *Zoom – Fast Zoom Out* three times consecutively.

In the center of the Smith Chart, the values of S11 and S22 are greater than 1, so the center of the Smith chart is part of the unstable region. Therefore, all points inside the load stability circle and all points outside the source stability circle belong to the unstable region.

Plotting the Noise Circles

In the Direct Plot form, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *sp* for *Analysis*.

3. Highlight *NC* for *Function*.
4. Highlight *Z-Smith* for *Plot Type*.
5. Highlight *Noise Level (dB)* for *Sweep*.
6. Type `900M` for *Frequency (Hz)*.
7. For the *Level Range (dB)* values
   a. Type `-30` in the *Start* field.
   b. Type `30` in the *Stop* field.
   c. Type `5` in the *Step* field.

The Direct Plot form looks like this.

8. Following the message at the bottom of the form,

Press Plot button on this form...

Click *Plot*.

The noise circles are plotted in the waveform window.

Noise Calculations with PSS and Pnoise

Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file.
4. In the ADE window, choose *Analyses – Disable* to disable any analyses you ran previously. (Check the ADE window to verify whether or not an analysis is enabled.)

Editing the Schematic

1. In the Schematic window, click the `RF` voltage source.

2. In the Schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears.

3. In the *Source type* field, type `dc`.

4. Click *OK*.
5. In the Schematic window, choose *Design – Check and Save*.

Setting up the PSS and Pnoise Analyses

➤ In the ADE window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

Setting up the PSS Analysis

1. In the Choosing Analyses form, click *pss*.
2. The *Beat Frequency* button is highlighted by default. Type `900M` in the *Beat Frequency* field.

   When the *Source type* is `dc`, you must enter the fundamental frequency manually.
3. Be sure the *Auto Calculate* button is *not* highlighted.
4. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type in a reasonable value, such as `20`, in the *Number of harmonics* field.
5. Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.
6. Highlight *Enabled*.

The Choosing Analyses form for PSS appears as follows.

Setting up the Pnoise Analysis

1. At the top of the Choosing Analyses form, click *pnoise*.

   The form changes to let you specify data for the periodic noise (Pnoise) analysis.

2. In the *Output Frequency Sweep Range (Hz)* cyclic field, choose *Start – Stop*. Type `800M` in the *Start* field and `1.2G` in the *Stop* field.

3. In the *Sweep Type* cyclic field, choose *Linear* and click the *Number of Steps* button. Type `201` in the *Number of Steps* field.

4. In the *Sidebands* cyclic field choose *Maximum sideband* and type `20` in the *Maximum sideband* field.

5. In the Output cyclic field, choose *voltage*.

   The value is appropriate because the reference sideband is an LNA. The form changes to display positive and negative output node fields.

6. To select the *Positive Output Node*

   a. Click the *Select* button next to the *Positive Output Node* field.

   b. In the Schematic window, select the amplifier output net.

   `/net28` displays in the *Positive Output Node* field.

   c. Leave the *Negative Output Node* field empty. It defaults to `/gnd!`.

   You can set the *Negative Output Node* to a different value by clicking on the *Negative Output Node Select* button and then clicking on the output node in the schematic.

7. To select the *Input Voltage Source*

a. Click the *Select* button next to the *Input Voltage Source* field.

b. In the Schematic window, select the RF port.

`/RF` displays in the *Input Voltage Source* field.

8. In the *Reference side-band* field, type `0`.

You specify the reference sideband as `0` for an LNA because an LNA has no frequency conversion from input to output.

9. Highlight *Enabled*.

The Choosing Analyses form for Pnoise appears as follows.

10. In the Choosing Analyses form, click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the ADE window.

   The output log file appears and displays information about the simulation as it runs.
2. Look in the CIW for a message that says the simulation is completed successfully.

Plotting the Noise Calculations

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.
2. In the Direct Plot form, select *Replace* for *Plotting Mode*.
3. Highlight *pnoise* for *Analysis*.

   The form changes to display fields for the Pnoise analysis.
4. Highlight *Noise Figure* for *Function*.

The completed form looks like this.

5. Follow the message at the bottom of the form

Press plot button on this form...

Click *Plot*.

The plot appears in the waveform window. The Noise Figure at 900MHz is about 4.19 dB.

6. Close the Direct Plot form and the waveform window.

Printing the Noise Summary

1. In the ADE window, choose *Results – Print – Noise Summary*.

   The Results Display Window and the Noise Summary form appear.

2. In the Noise Summary form, highlight *spot noise* for *Type*.

3. Type `900M` in the *Frequency Spot (Hz)* field.

4. Click *Include All Types* in the *FILTER* list box.

   The *Include All Types* is the *FILTER* list box default, however, you must click the button to activate the choice.

5. In the *truncate* cyclic field, select *none* in the *TRUNCATE & SORT* section.

The filled-in Noise Summary form looks like this.

6. In the Noise Summary form, click *OK*.

The noise contributors now appear in the Results Display window.

Results Display Window

Window Expressions Info Help cādence

Device	Param	Noise Contribution	% Of Total
/RF	rn	4.46308e-18	38.11
/Q4	rb	3.09923e-18	26.47
/Q4	ib	1.58232e-18	13.51
/Q4	ic	1.05968e-18	9.05
/R21	rn	3.93139e-19	3.36
/Q4	re	2.89681e-19	2.47
/R6	rn	2.42592e-19	2.07
/R3	rn	1.83436e-19	1.57
/PORT0	rn	1.51426e-19	1.29
/R0	rn	1.25381e-19	1.07
/R22	rn	1.11598e-19	0.95
/Q3	ic	7.59155e-21	0.06
/Q3	rb	5.81939e-22	0.00
/Q4	rc	3.58162e-22	0.00
/Q4	fn	2.81693e-22	0.00
/Q3	rc	1.46749e-22	0.00
/Q3	ib	1.19067e-24	0.00
/R23	rn	5.84176e-29	0.00
/R19	rn	1.52636e-30	0.00
/R10	rn	3.91857e-31	0.00
/Q2	rb	1.90196e-32	0.00
/Q2	ic	1.57603e-32	0.00
/Q2	rc	2.17163e-34	0.00
/Q2	ib	5.90494e-37	0.00
/RF	ext_file_noise	0	0.00
/R6	fn	0	0.00
/R3	fn	0	0.00
/R23	fn	0	0.00
/R22	fn	0	0.00
/R21	fn	0	0.00
/R19	fn	0	0.00
/R10	fn	0	0.00
/R0	fn	0	0.00
/Q3	re	0	0.00
/Q2	re	0	0.00
/PORT0	ext_file_noise	0	0.00

Spot Noise Summary (in V^2/Hz) at 900M Hz Sorted By Noise Contributors
Total Summarized Noise = 1.17105e-17
Total Input Referred Noise = 4.20218e-19
The above noise summary info is for pnoise data

24

This summary helps you determine the percentage of noise contributed by the different devices in the circuit.

Plotting the 1dB Compression Point

Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file.
4. In the ADE window, choose *Analyses – Disable* to disable any analyses you ran previously. (Check the ADE window to verify whether or not an analysis is enabled.)

Editing the Schematic

1. In the Schematic window, click the `RF` voltage source.

2. In the Schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears.

Direct Plot Form
Plotting Mode
Replace
Analysis
pss
pxf
Function
Voltage Gain
Transimpedance
Sweep
spectrum
sideband
Modifier
Magnitude
Phase
dB20
Real
Imaginary
Add To Outputs
freqaxis = absin
> Select Port or Voltage Source on schematic...
OK
Cancel
Help

3. In the *Source type* field, type `sine`.
4. Type `900M` for the *Frequency* value.
5. Type `prf` for the *Amplitude (dBm)* value.
6. Be sure the *Amplitude* field is empty.
7. In the Edit Objects Properties form, click *OK.*
8. In the Schematic window, choose *Design – Check and Save*.

Setting up the PSS Analysis

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.
2. In the Choosing Analyses form, click *pss*.

   The form changes to let you specify data for the PSS analysis.
3. Type `100M` for the *Fundamental Frequency*.

   The *Beat Frequency* button is highlighted by default.
4. Be sure the *Auto Calculate* button is *not* highlighted.
5. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type in a reasonable value, such as `10`, in the *Number of harmonics* field.

The top of the PSS form appears like the one below.

6. Highlight *moderate* for the *Accuracy Default (errpreset).*
7. Highlight the *Sweep* button.

   The form changes to let you specify data for the sweep.
8. In the *Sweep* cyclic field, choose *Variable*.
9. Click the *Select Design Variable* button.

   The Select Design Variable form appears.

10. In the Select Design variable form, highlight *prf* and click *OK*.

The *prf* variable is the amplitude, in dBm, of the RF input source. If you are not sure of this, use the Edit Object Properties form to examine the *Amplitude (dBm)* field for the `RF` voltage source.

11. Choose *Start-Stop* for the *Sweep range* and type `-30` in the *Start* field and `10` in the *Stop* field.

12. Choose *Linear* for the *Sweep Type* and click the *Number of Steps* button. Type `10` as the number of steps in the field.

The bottom of the PSS form appears like the one below.

13. In the Choosing Analyses form, click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the ADE window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the 1dB Compression Point

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.

2. In the Direct Plot form, highlight *pss* for *Analysis*.
3. Highlight *Compression Point* for *Function*.
4. Type `-25` for the *Input Power Extrapolation Point (dBm)*.

   You learn what value to use for the extrapolation point through experience. If you do not specify a value, the plot defaults to the minimum variable value.

5. Choose *Input Referred 1dB Compression*.
6. Click *900M* in the *1st Order Harmonic* list box.

The filled-in form looks like this.

Direct Plot Form
Plotting Mode
Replace
Analysis
pss
Function
Voltage
Current
Power
Voltage Gain
Current Gain
Power Gain
Transconductance
Transimpedance
Compression Point
IPN Curves
Power Contours
Reflection Contours
Harmonic Frequency
Power Added Eff.
Power Gain Vs Pout
Comp. Vs Pout
Node Complex Imp.
THD
Select
Port (fixed R(port))
Format
Output Power
Gain Compression (dB)
1
"prf" ranges from -30 to 10
Input Power Extrapolation Point (dBm)
-25
Input Referred 1dB Compression
1st Order Harmonic
5 500M
6 600M
7 700M
8 800M
9 900M
10 1G
Add To Outputs
> Select Port on schematic...
OK
Cancel
Help

7. In the Schematic Window, click the `PORT0` output port.

The plot appears in the waveform window.

The SpectreRF simulator calculates the 1dBCompressionPoint as -10.3614.

Calculating the Third-Order Intercept Point with Swept PSS

The method illustrated here uses only swept PSS analysis. An alternative method, which runs more quickly if the input signals are very close together, uses both swept PSS and PAC analyses and is described in Third-Order Intercept Measurement with Swept PSS and PAC on page 1893.

Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file.
4. In the ADE window, choose *Analyses – Disable* to disable any analyses you ran previously. (Check the ADE window to verify whether or not an analysis is enabled.)

Editing the Schematic

1. In the Schematic window, click the input voltage source and then choose *Edit – Properties – Objects*.

   The Edit Object Properties form appears.
2. In the Edit Object Properties form, choose *sine* for the *Source Type*.
3. Type `fund2`, or any name you choose, in the *Second frequency name* field.
4. Type `prf` for the *Amplitude (dBm)* and *Amplitude 2 (dBm)* values.
5. Type `900M` for the *Frequency* value.
6. Type `920M` for the *Frequency 2* value.
7. Be sure the *Amplitude* and *Amplitude2* fields are empty.
8. In the Edit Object Properties form, click *OK*.
9. In the Schematic window, choose *Design – Check and Save*.

Setting up the Swept PSS Analysis

➤ In the ADE window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

In the Choosing Analyses form, do the following and then click *OK*:

1. In the Choosing Analyses form, click *pss*.
2. In the *Fundamental Tones* list box, be sure the *Auto calculate* button is highlighted.

   The fundamental frequency, 20M, is displayed.

   The *Beat Frequency* button is highlighted by default.
3. In the *Output harmonics* cyclic field, choose *Number of harmonics* for the *Output harmonics* choice and type in a reasonable value, such as `60`, in the field.

   The top of the PSS form looks like this.

4. Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.
5. Highlight the *Sweep* button.

   The form changes to let you specify data for the sweep.
6. In the *Sweep* cyclic field, choose *Variable*.
7. Click the *Select Design Variable* button.

   The Select Design Variable form appears.
8. In the Select Design Variable form, highlight *prf* and click *OK*.
9. Choose *Start-Stop* for the *Sweep Range*, and type `-30` as the *Start* value and type 10 as the *Stop* value.
10. Choose *Linear* for the *Sweep Type* and click the *Step Size* button. Type `5` as the step size.

The bottom of the PSS form looks like this.

11. Click *OK*.

The Choosing Analyses form closes.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the ADE window.

The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the Third-Order Intercept Point

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.

2. Highlight *Replace* for *Plotting Mode*.
3. Highlight *pss* for *Analysis*.
4. Highlight *IPN Curves* for *Function*.
5. Highlight *Variable Sweep ("prf")* for *Circuit Input Power*.
6. Choose *Input Referred IP3*.
7. Type `-25` for the *Input Power Extrapolation Point (dBm)*.

   You learn what value to use for the extrapolation point through experience. If you do not specify a value, the plot defaults to the minimum variable value.

8. In the *order* cyclic field, choose *order 3rd*.
9. In the *3rd Order Harmonic* list box, click *940M*.
10. In the *1st Order Harmonic* list box, click *900M*.

    Because the two input frequencies are 900MHz and 920MHz, the two-tone, third-order harmonics are 880MHz and 940MHz. These correspond to the 45th and 47th harmonics.

The filled-in form looks like this.

Direct Plot Form
Plotting Mode
Replace
Analysis
pss
Function
Voltage
Current
Power
Voltage Gain
Current Gain
Power Gain
Transconductance
Transimpedance
Compression Point
IPN Curves
Power Contours
Reflection Contours
Harmonic Frequency
Power Added Eff.
Power Gain Vs Pout
Comp. Vs Pout
Node Complex Imp.
THD
Select
Port (fixed R(port))
Circuit Input Power
Single Point
Variable Sweep
("prf")
"prf" ranges from -30 to 10
Input Power Extrapolation Point (dBm)
-25
Input Referred IP3
Order
3rd
3rd Order Harmonic
1st Order Harmonic
5 500M
6 600M
7 700M
8 800M
9 900M
10 1G
Add To Outputs
Replot
> Select Port on schematic...
OK
Cancel
Help

11. In the Schematic window, click the PORT 0 output port.

The IPN point is plotted in the waveform window.

12. Click *Replot.*

The SpectreRF simulator plots the first-order and third-order curves and identifies the intersection of their slopes.

The third-order interpolation point for this example is 2.16227 dB.

Calculating Conversion Gain and Power Supply Rejection with PSS and PXF

Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file.

4. In the ADE window, choose *Analyses – Disable* to disable any analyses you ran previously. (Check the ADE window to verify whether or not an analysis is enabled.)

Editing the Schematic

1. In the Schematic window, choose *Edit – Properties – Objects*.

   The Edit Object Properties form appears.

2. In the Schematic window, click the RFinput voltage source.

   The Edit Object Properties form changes to let you specify data for the input voltage source.

3. In the Edit Object Properties form, if necessary, type `dc` for the *Source Type*.
4. Click *OK*.
5. in the schematic editing window, choose *Design – Check and Save*.

Setting up the PSS and PXF Analyses

➤ In the ADE window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

Setting up the PSS Analysis

1. In the Choosing Analyses form, highlight *pss*.
2. The *Beat Frequency* button is highlighted by default. Type `900M` in the *Beat Frequency* field.

   Because the source type is dc, you must manually enter the fundamental frequency.

3. Be sure the *Auto Calculate* button is not highlighted.
4. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type in a reasonable value, such as `4`, in the field.
5. Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.

The Choosing Analyses form looks like this.

Setting up the PXF Analysis

1. At the top of the Choosing Analyses form, click *pxf*.

   The form changes to let you specify data for the PXF analysis.

2. In the *Output Frequency Sweep Range(Hz)* cyclic field, choose *Start-Stop*, and type `10M` and `1.2G` for the *Start* and *Stop* values, respectively in the fields.

3. In the *Sweep Type* cyclic field, choose *Linear* and highlight the *Number of Steps* button.

4. Type `101` for *Number of Steps* in the field.

5. In the *Sidebands cyclic field*, choose *Maximum sideband* for the *Sidebands* choice. Type `0` in the field.

6. Choose *voltage* for the *Output* choice.

7. Click the *Select* button for the *Positive Output Node*, and then click the output signal in the Schematic window.

8. Leave the *Negative Output Node* field empty.

   This field defaults to */gnd!* You can set the *Negative Output Node* to a different value by clicking on the *Negative Output Node Select* button and then clicking on the output node in the schematic.

9. Check the *Enabled* checkbox for the small-signal analysis, if necessary.

The pxf section of the form looks like this.

10. In the Choosing Analyses form, click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the ADE window.

The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting Conversion Gain and Power Supply Rejection

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.

2. In the Direct Plot form, highlight *Replace* for *Plotting mode*.
3. Highlight *pxf* for *Analysis*.
4. Highlight *Voltage Gain* for *Function*.
5. Highlight *dB20* for *Modifier*.

   The filled-in form looks like this.

6. In the Schematic window, click the RF source component.

   The transfer function for the LNA is plotted in the waveform window.

   The conversion gain at 900M is approximately 14.4 dBV.

7. In the Direct Plot form, choose *Replace* for *Plotting Mode*.

8. To see the power supply rejection, click the vdb DC voltage source in the schematic.

Be sure to click the components, not the wires.

The plot shows that the power supply rejection at 900M is approximately -106 dBV.

Oscillators

Autonomous circuits, such as oscillators, are time-invariant circuits with time-varying responses. Thus, autonomous circuits generate non-constant waveforms even though they are not driven by a time-varying stimulus.

You cannot specify the analysis period for autonomous circuits because you do not know the precise oscillation period in advance of the simulation. Instead, you specify an estimate of the oscillation period for the simulation. The PSS analysis uses your estimate to compute the precise period and the periodic solution waveforms.

Phases of Autonomous PSS Analysis

A PSS analysis has two phases:

- A transient analysis phase to initialize the circuit

  The transient analysis phase is divided into three intervals:

  - A beginning interval that starts at `tstart`, which is normally 0, and continues through the onset of periodicity for the independent sources
  - A stabilization interval of length `tstab`

    For driven circuits, the stabilization interval is optional. For autonomous circuits, `tstab` needs to be set to an interval that gives the circuit a good initial condition.
  - A final interval that is four times the estimated oscillation period specified in the PSS Analysis form

    During the final interval, the PSS analysis monitors the waveforms in the circuit and improves the estimate of the oscillation period.

- A Shooting phase to compute the periodic steady state solution

  During the Shooting phase, the circuit is simulated repeatedly over one period. The length of the period and the initial conditions are modified to find the periodic steady state solution.

See *Autonomous PSS Analysis* in *Virtuoso Spectre Circuit Simulator RF Analysis Theory* for more information on the autonomous PSS analysis algorithm.

Phase Noise and Oscillators

Oscillators tend to amplify any noise present near the oscillation frequency. The closer the noise frequency is to the oscillation frequency, the greater the noise amplification. Noise amplified by the oscillator in this manner is called *phase noise*. Phase noise is the most significant source of noise in oscillators, and because phase noise is centered about the oscillation frequency, filtering can never completely remove it.

You can understand phase noise if you recognize that the phase of an oscillator is arbitrary because there is no drive signal to lock to. Any waveform that is a solution to an oscillator can be shifted in time and still be a solution. If a perturbation disturbs the phase, nothing restores the phase, so it drifts without bound. If the perturbation is random noise, the drift is a random walk. Furthermore, the closer the perturbation frequency is to the oscillation frequency, the better it couples to the phase and the greater the drift. The perturbation need not come from random noise. Noise might also couple into the oscillator from other sources, such as the power supplies.

The first two examples in this chapter both examine phase noise.

Starting and Stabilizing Oscillators

To simulate an oscillator using PSS analysis, you must first start the oscillator by supplying either:

- A brief impulse stimulus

  The stimulus should couple strongly into the oscillatory mode of the circuit and poorly into other long-lasting modes such as bias circuitry.

- A set of initial conditions for the components of the oscillator's resonator

Regardless of which technique you use to start the oscillator, allow the oscillator to run long enough to stabilize before you start the Shooting phase and compute the steady state solution. Adjust the *tstab* parameter to supply the additional stabilization period.

See *Oscillators and Autonomous PSS Analysis* in the *Virtuoso Spectre Circuit Simulator RF Analysis Theory* for more information on oscillator simulation. For details about the autonomous PSS analysis see *Autonomous PSS Analysis* in the *Virtuoso Spectre Circuit Simulator RF Analysis Theory*.

The tline3oscRF Oscillator Circuit

This example computes the periodic steady state solution and the phase noise for the *tline3oscRF* circuit shown in Figure A-1.

Figure A-1 Schematic for the tline3oscRF Oscillator Circuit

This circuit supplies the brief impulsive stimulus needed to start the oscillator, so you do not need to set initial conditions. You supply the data necessary for PSS analysis to estimate the period when you specify the fundamental frequency.

Simulating the tline3oscRF Oscillator Circuit

Opening the tline3oscRF Circuit in the Schematic Window

1. In the Command Interpreter Window (CIW), choose *File – Open*.

   The Open File form appears. Filling in the Open File form opens the schematic.

2. For *Library Name,* choose *my_ rfExamples* or whatever you called the editable copy of the *rfExamples* library you created.

3. Choose *schematic* for *View Name*.
4. In the *Cells* list box, highlight *tline3oscRF*.

   tline3oscRF appears in the *Cell* field.
5. Highlight *edit* to choose the *Mode*.

   The completed Open File form appears like the one below.

6. Click *OK*.

   The Schematic window for the *tline3oscRF* oscillator opens.
7. In the Schematic window, choose *Tools – Analog Environment*.

The ADE window opens.

You can also choose *Tools – Analog Environment – Simulation* in the CIW to open the ADE window without opening the design. You can open the design later by choosing *Setup – Design* in the ADE window and choosing the *tline_3oscRF* in the Choosing Design form.

Choosing Simulator Options

1. In the ADE window, choose *Setup – Simulator/Directory/Host*.

   The Choosing Simulator/Directory/Host form appears.

2. In the Choosing Simulator/Directory/Host form, choose *spectre* for *Simulator*.
3. Type in the name of the project directory, if necessary.
4. Highlight the appropriate mode for the *Host Mode*.
5. For remote or distributed simulations, type the required information into the fields that become available for each choice.

The completed form looks like this.

6. Click *OK*.

Setting Up Model Libraries

1. In the ADE window, choose *Setup – Model Libraries*.

   The Model Library Setup form appears.

2. In the *Model Library File* field, type the full path to the `rfModels.scs` model file including the file name, `rfModels.scs`.

3. Click *Add*.

4. In the *Model Library File* field, type the full path to the `tline_s17_8470_40G.scs` model file including the file name, `tline_s17_8470_40G.scs` and click *Add*.

The completed form looks like this.

5. In the Model Library Setup form, click *OK*.

Periodic Steady State and Phase Noise with PSS and Pnoise

This example computes the periodic steady state solution and phase noise for the *tline3oscRF* oscillator circuit. You perform a `PSS` analysis first because the periodic steady state solution must be determined before you can perform a `Pnoise` small-signal analysis to determine the phase noise.

Setting Up the Simulation

1. If necessary, open the tline3oscRF circuit.
2. If necessary, specify the full path to the model files.
3. If necessary, in the ADE window use *Analysis - Disable* to disable any analyses you ran previously. (Check the *Analyses* area in the ADE window to verify whether or not any analyses are enabled.)

Setting Up the PSS Analysis

Note: If your oscillator circuit does not contain a stimulus to start the oscillator, you must run a transient analysis before you run this PSS analysis. From the transient analysis you save the node voltages and use them as initial conditions to start the oscillator in the PSS analysis.

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, click *pss*.

   The form changes to display options needed for `PSS` analysis.

   In the *Fundamental Tones* area, the *Beat Frequency* button is highlighted by default. Be sure the *Auto Calculate* button is *not* highlighted.

   In the field next to the *Beat Frequency* and *Beat Period* buttons, type your best estimate of the oscillation frequency. (Your estimate can be the node voltages saved from the transient analysis mentioned above.)

   Type `1.4G` in this example.

3. In the *Output harmonics* cyclic field, choose Number of harmonics and type in a reasonable number, such as `10`.

   The top of the PSS Choosing Analyses form appears below.

4. Highlight *moderate* for the *Accuracy Defaults (errpreset)*.
5. Highlight *Oscillator*.

   The form changes to let you specify the two nodes needed for oscillator simulation.
6. Click the *Select* button next to the *Oscillator node* field. Then click the appropriate wire in the Schematic window to choose the oscillator node. In this example, choose *net5*.

   `/net5` appears in the *Oscillator node* field.
7. Leave the *Reference node* field empty.

   The *Reference node* field defaults to `/gnd!`. You can set *Reference node* to a different value by clicking on the *Reference node Select* button and then choosing the appropriate node in the schematic.
8. Verify that *Enabled* is highlighted.

The bottom of the PSS Choosing Analyses form appears below.

Setting Up the Pnoise Analysis

1. At the top of the Choosing Analyses form, click *pnoise*.

   The form changes to let you specify data for the Pnoise analysis.

2. In the *Sweeptype* cyclic field, choose *Relative*. Enter `1` in the *Relative Harmonic* field.

   This choice specifies that the *Frequency Sweep Range (Hz)* values you choose represent frequency values away from the fundamental frequency. For example, if you specify `2K`, you choose a value `2K` away from the fundamental frequency.

3. Choose *Start-Stop* for the type of *Frequency Sweep Range (Hz)*. Type in reasonable values, such as `1K` and `100M` as the *Start* and *Stop* values.

4. In the *Sweep Type* cyclic field, choose *Logarithmic*. Use a logarithmic sweep when you plot phase noise because of the size of the phase noise values.

5. Highlight *Number of Steps* and type `201` in the field.

   Note: Be sure to use a nonzero starting value for logarithmic sweeps.

6. In the *Sidebands* cyclic field, choose *Maximum Sideband* and type `7` in the *Maximum sideband* field. The default parameter value is `maxsideband=7`.

7. Begin with a *Maximum sideband* value of 7. Then, in subsequent simulations, increase the value to see whether the output noise changes. Continue to increase the *Maximum sideband* value until the output noise stops changing.

   For a fundamental oscillator, *Maximum sideband* must be at least `1` to see any flicker noise up conversion. In general, small values for *Maximum sideband* are not recommended.

   Be sure to use a nonzero *Maximum sideband* value.

   The top of the Pnoise Choosing Analyses form appears below.

8. In the *Output* cyclic field, choose *voltage*.

9. Click the *Select* button next to the *Positive Output Node* field.Then click the appropriate wire in the Schematic window to choose net 5.

`/net5` appears in the *Positive Output Node* field.

10. Leave the *Negative Output Node* field empty.

This field defaults to `/gnd!`. You can set the *Negative Output Node* to a different value by clicking on the *Negative Output Node Select* button and then choosing the output node in the schematic.

11. In the *Input Source* cyclic field, choose *none*.

The bottom of the Pnoise Choosing Analyses form appears below.

12. Verify that the Pnoise analysis is Enabled. Then click *OK* in the Choosing Analyses form.

The `PSS` and `Pnoise` options you choose appear in the *Analyses* list box in the ADE window.

Running the Simulation

1. In the ADE window, choose *Simulation – Netlist and Run* to run the simulation.

The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the Fundamental Frequency

These steps describe how to plot the fundamental frequency.

1. Choose *Results – Direct Plot – Main Form* in the ADE window.

   The Direct Plot form appears.

2. Highlight *Replace* for *Plot Mode*.
3. Highlight *pss* for *Analysis.*
4. Highlight *Harmonic Frequency* for *Function*.

The form changes to display the *Harmonic Frequency* list box.

5. As the prompt at the bottom of the form instructs, highlight the first harmonic in the *Harmonic Frequency* list box.

6. Click *Plot*.

The waveform window displays the fundamental frequency.

Notice that the fundamental frequency value of `2.022G` is also displayed in the *Harmonic Frequency* list box in the Direct Plot form.

Plotting the Periodic Steady State Solution

If necessary, in the ADE window choose *Results – Direct Plot – Main Form* to open the Direct Plot form.

To plot the periodic steady state solution waveform for distortion, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. If necessary, highlight *pss* for *Analysis*.
3. Highlight *Voltage* for *Function*.
4. Highlight *Time* for *Sweep*.

5. Notice that the default choice in the *Select* cyclic field is *Net*. Also notice the *Select net on schematic* prompt at the bottom of the form.

6. In the Schematic window, click the appropriate wire to choose net 5.

The waveform window displays the sweep of voltage versus time—the periodic steady state solution.

Plotting the Phase Noise

If necessary, in the ADE window choose *Results – Direct Plot – Main Form* to open the Direct Plot form.

To plot the phase noise waveform, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pnoise* for *Analysis*.

   The form changes to display fields relevant for the `Pnoise` analysis.

3. Highlight *Phase Noise* for *Function*.

4. Click *Plot*.

The waveform window displays the phase noise versus frequency with the frequency in logarithmic scale. When you plot phase noise, you get more useful information if you use the logarithmic scale.

The oscDiff Circuit: A Balanced, Tunable Differential Oscillator

This example computes the fundamental frequency, output noise, and phase noise for the *oscDiff* circuit, the balanced, tunable differential oscillator shown in Figure A-2.

Figure A-2 Schematic for the oscDiff Oscillator Circuit

The *oscDiff* circuit generates sinusoids in antiphase for differential nodes *VoL* and *VoR*. The oscillation frequency is primarily governed by inductor *L0* and capacitor *C3*. The *C3*

capacitance value is voltage-dependent and obeys Equation A-1, the standard Schottky barrier capacitance equation.

(A-1)

$$Ctune = \frac{Cj0}{\left(1 + \frac{Vcntrl}{phi}\right)^{gamma}}$$

where

Ctune	*C3* capacitance value
Cj0	Zero-bias junction capacitance
Vcntrl	Applied varactor voltage
phi	Barrier potential
gamma	Junction grading coefficient

Simulating the oscDiff Circuit

Opening the oscDiff Circuit in the Simulation Window

1. In the Command Interpreter Window (CIW), choose *File – Open*.

   The Open File form appears. Filling in the Open File form opens the schematic.

2. For *Library Name,* choose *my_ rfExamples* or whatever you called the editable copy of the *rfExamples* library you created.

3. Choose *schematic* for *View Name*.

4. In the *Cells* list box, highlight *oscDiff*.

   oscDiff appears in the Cell field.

5. Highlight *edit* for *Mode*.

The completed Open File form appears like the one below.

6. Click *OK*.

   The Schematic window for the oscDiff oscillator opens as shown in Figure A-2 on page 1754.

7. In the Schematic window, choose *Tools – Analog Environment*.

The ADE window appears.

You can also choose *Tools – Analog Environment – Simulation* in the CIW to open the ADE window without opening the design. You can open the design later by choosing *Setup – Design* in the ADE window and choosing the *oscDiff* in the Choosing Design form.

Choosing Simulator Options

1. In the ADE window, choose *Setup – Simulator/Directory/Host*.

   The Choosing Simulator/Directory/Host form appears.

2. In the Choosing Simulator/Directory/Host form, choose *spectre* for *Simulator*.
3. Type in the name of the project directory, if necessary.
4. Highlight the appropriate button to specify the *Host Mode*.
5. For remote or distributed simulation, fill in the additional fields that become available.

The completed form appears like the one below.

6. In the Choosing Simulator/Directory/Host form, click *OK*.

Setting Up Model Libraries

1. In the ADE window, choose *Setup – Model Libraries*.

   The Model Library Setup form appears.

2. In the Model Library File field, type the full path to the model file including the file name, *rfModels.scs*.

3. Click *Add*.

   The completed form looks like this.

4. Click *OK*.

Fundamental Frequency, Output Noise, and Phase Noise with PSS and Pnoise

This example computes the fundamental frequency, output noise, and phase noise for the oscDiff differential oscillator circuit. You perform a `PSS` analysis first because the periodic steady-state solution must be determined before you can perform a `Pnoise` small-signal analysis to determine the fundamental frequency, output noise, and phase noise.

Setting Up the Simulation

1. If necessary, open the *oscDiff* circuit.
2. If necessary, specify the full path to the model files.
3. If necessary, in the ADE window choose *Analyses – Disable* to disable any analyses you ran previously. (Check the *Analyses* area in the ADE window to verify whether or not any analyses are enabled.)

Editing Design Variables

1. In the ADE window, choose *Variables – Edit*.

   The Editing Design Variables form appears. For this example, you use this form to set the value of the variable *Ctune*.
2. In the Table of Design Variables list box, highlight Ctune.
3. Type 3.5p in the Value (Expr) field.
4. Click *Change*.

The completed form looks like this.

5. Click *OK*.

   Ctune appears in the *Design Variables* section of the ADE window with its modified value

Setting Up the PSS Analysis

Note: If your oscillator circuit does not contain a stimulus to start the oscillator, you must run a `transient` analysis before you run this `PSS` analysis. From the `transient` analysis, you save the node voltages and use them as initial conditions to start the oscillator in the `PSS` analysis.

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, choose *pss* for the analysis.

   The form changes to display options needed for the `PSS` analysis. You perform a `PSS` analysis first because the periodic steady state solution must be determined before you can perform a `Pnoise` small-signal analysis to determine the phase noise.

3. In the *Fundamental Tones* area, the *Beat Frequency* button is highlighted by default. Be sure the *Auto Calculate* button is *not* highlighted.

4. In the field next to the *Beat Frequency* and *Beat Period* buttons, type your best estimate of the oscillation frequency. (Your estimate can be the node voltages saved from the transient analysis mentioned above.)

Type `1.9G` in this example.

5. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type `5` in the field.

6. The top of the Periodic Steady State Analyses form appears below.

7. Highlight *moderate* for the *Accuracy Default (errpreset)*.

8. Type `100.5n` in the *Additional Time for Stabilization (tstab)* field.

   You can use a slightly different value as long as it is above 100n.

9. If you want to save the results of the initial transient analysis, set *Save Initial Transient Results (saveinit)* to *yes*.

10. Highlight *Oscillator*.

    The form changes to let you specify the two nodes needed for oscillator simulation.

11. Click the *Select* button next to the *Oscillator node* field. Then click the appropriate wire in the Schematic window to choose the oscillator node. In this example, choose the node labelled *VoR*.

    `/VoR` appears in the *Oscillator node* field.

12. Click the *Select* button next to the *Reference node* field. Then click the appropriate wire in the Schematic window to choose the reference node. In this example, choose the node labeled *VoL*.

`/VoL` appears in the *Reference node* field. Because this is a differential oscillator, the reference node is not `/gnd!` as it was in the *tline3oscRF* example.

The bottom of the Choosing Analyses form for PSS looks like the following.

Verify that *Enabled* is highlighted for the `PSS` analysis.

Setting Up the Pnoise Analysis

1. At the top of the Choosing Analyses form, click *pnoise*.

The form changes to let you specify data for the `Pnoise` analysis.

2. In the *Sweeptype* cyclic field, choose *Relative*. Enter 1 in the *Relative Harmonic* field.

This choice specifies that the *Frequency Sweep Range (Hz)* values you choose represent frequency values away from the fundamental frequency. For example, if you specify `2K`, you choose a value 2K away from the fundamental frequency.

3. In the *Frequency Sweep Range(Hz)* cyclic field, choose *Start-Stop* and type `1` and `100M` as the *Start* and *Stop* values, respectively.

4. In the *Sweep Type* cyclic field, choose *Logarithmic*. Use a logarithmic sweep when you plot phase noise because of the size of the phase noise values.

5. Highlight *Points Per Decade* and type `5` in the field.

   Note: Be sure to use a nonzero starting value for logarithmic sweeps.

6. In the *Sidebands* cyclic field, choose *Maximum sideband* and type `7` in the *Sidebands* field. The default value is 7.

   Be sure to use a nonzero *Maximum sideband* value.

   The top of the Choosing Analyses form for the `Pnoise` analysis looks like this.

7. In the *Output* cyclic field, choose *voltage*.

8. Click the *Positive Output Node Select* button. Then click the appropriate wire in the Schematic window to choose *VoR*.

   `/VoR` appears in the *Positive Output Node* field.

9. Click the *Negative Output Node Select* button. Then click the appropriate wire in the Schematic window to choose *VoL*.

   `/VoL` appears in the *Negative Output Node* field. Because this is a differential oscillator, the negative output node is not `/gnd!` as it was for the *tline3oscRF* oscillator.

10. In the *Input Source* cyclic field, choose *none*.

    The bottom of the Choosing Analyses form for the `Pnoise` analysis looks like this.

11. In the Choosing Analyses form, verify that the *Enabled* field at the bottom of the Pnoise form is highlighted. Then click *OK* at the top of the Choosing Analyses form.

    The `pss` and `pnoise` analyses you set up appear in the *Analyses* list box in the ADE window.

Running the Simulation

1. In the ADE window, choose *Simulation – Netlist and Run* to run the simulation.

The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the Fundamental Frequency

To plot the fundamental frequency, do the following steps.

1. Choose *Results – Direct Plot – Main Form* in the ADE window.

   The Direct Plot form appears.

2. Highlight *Replace* for *Plot Mode*.

3. Highlight *pss* for *Analysis*.

4. Highlight *Harmonic Frequency* for *Function*.

   The Direct Plot form changes to display available frequencies in the *Harmonic Frequency* list box.

5. Follow the prompt at the bottom of the Direct Plot form,

   Select Harmonic Frequency on this form...

   then highlight the first harmonic in the *Harmonic Frequency* list box.

The Direct Plot form appears as follows.

6. Follow the prompt at the bottom of the Direct Plot form,

Press plot button on this form...

then click *Plot*.

The waveform window displays the fundamental frequency.

Notice that the fundamental frequency value for the first harmonic of `1.939 G` is also displayed in the Harmonic Frequency list box in the Direct Plot form.

Plotting the Output Noise and Phase Noise

If necessary, in the ADE window choose *Results - Direct Plot - Main Form* to open the Direct Plot form.

To plot the output noise, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. If necessary, highlight *pnoise* for *Analysis*.

3. Highlight *Output Noise* for *Function*.
4. Highlight *V/sqrt(Hz)* for *Signal Level*.
5. Highlight *dB20* for *Modifier*.

6. Follow the prompt at the bottom of the Direct Plot form,

   Press plot button on this form...

   then click *Plot*.

The output noise is plotted in the waveform window.

To plot phase noise on the same graph with output noise, do the following:

1. Highlight *Append* for *Plot Mode*.
2. Highlight *Phase Noise* for *Function*.
3. Click *Plot*.

The phase noise plot is added to the output noise plot.

By plotting the phase noise on the same graph as the output noise, you can see the linear relationship between the output noise and the phase noise. The output noise is scaled by the carrier amplitude to produce the phase noise value.

The Van der Pol Circuit: Measuring AM and PM Noise Separation

As a simple example, consider the `vdp_osc` circuit, a feedback amplifier circuit shown in Figure A-3. This example separates oscillator noise into AM and PM components. It also calculates USB and LSB noise for the oscillator.

Figure A-3 Schematic for the vdp_osc Feedback Amplifier Circuit

The `vdp_osc` is a parallel RLC circuit with a nonlinear transconductance represented by a polynomial voltage controlled current source. At small capacitor voltages, the transconductance is negative; that is, it is an active device which creates positive feedback that seeks to increase the voltage on the capacitor. At larger capacitor voltages, where the transconductance term goes into compression, it effectively acts as a positive resistor (that is, it creates negative feedback) and limits the voltage. You use a cubic polynomial model for the circuit.

Simulating the vdp_osc Circuit

Opening the vdp_osc Circuit

1. In the Command Interpreter Window (CIW), choose *File – Open*.

   The Open File form appears. Filling in the Open File form opens the vdp_osc schematic.

2. For *Library Name*, choose *my_rfExamples* or whatever you called the editable copy of the *rfExamples* library.

3. Choose *schematic* for *View Name*.

4. In the *Cells* list box, highlight *vdp_osc*.

vdp_osc appears in the Cell field.

5. Highlight *edit* for *Mode*.

   The completed Open File form appears like the one below.

6. Click *OK*.

   The Schematic window for the `vdp_osc` oscillator opens.

Editing Properties for the Inductor

1. In the Schematic window, select the L1 inductor.
2. With the L1 inductor selected, choose *Edit-Properties-Objects*.

The Edit Object Properties form displays for the L1 inductor.

3. Edit the *Initial condition* value by replacing the `0` with `1`.

4. Click *OK* in the Edit Object Properties form.
5. Choose *Design – Check and Save* in the Schematic window.

Opening the Simulation Window

1. In the Schematic window, choose *Tools – Analog Environment*.

The ADE window appears.

You can also choose *Tools – Analog Environment – Simulation* in the CIW to open the ADE window without opening the design. You can open the design later by choosing *Setup – Design* in the ADE window and choosing *vdp_osc* in the Choosing Design form.

Measuring AM and PM Conversion with PSS and Pnoise

This example computes AM and PM conversion for the `vdp_osc` oscillator circuit. You perform a `PSS` analysis first because the periodic steady-state solution must be determined before you can perform a `Pnoise` small-signal analysis to determine the AM and PM noise components from the results of the `PSS` and `Pnoise` analyses.

Setting Up the Simulation

1. If necessary, open the vdp_osc circuit.
2. If necessary, in the ADE window use *Analysis - Disable* to disable any analyses you ran previously. (Check the *Analyses* area in the ADE window to verify whether or not any analyses are enabled.)

Setting Up the PSS Analysis

Note: If your oscillator circuit does not contain a stimulus to start the oscillator, you must run a `transient` analysis before you run this `PSS` analysis. From the `transient` analysis, you save the node voltages and use them as initial conditions to start the oscillator in the `PSS` analysis (see step 3).

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, choose *pss* for the *Analysis*.

   The form changes to display options needed for the `PSS` analysis. You perform a `PSS` analysis first because the periodic steady state solution must be determined before you can perform a `Pnoise` small-signal analysis to determine the phase noise.

3. In the *Fundamental Tones* area, highlight *Beat Period.* Be sure the *Auto Calculate* button is *not* highlighted.
4. In the field next to *Beat Period*, type `2m`. This is an estimate based either on the results of a prior transient analysis or quick manual calculations from LC oscillator equations.
5. In the *Output harmonics* cyclic field, choose *Number of harmonics.* Type `10` in the *Number of harmonics* field.

The top of the PSS Choosing Analyses form appears below.

6. Highlight *moderate* for the *Accuracy Defaults (errpreset)*.
7. In the *Additional Time for Stabilization field (tstab)*, type `700n`.
8. Highlight *Oscillator*.

   The form changes to let you specify the two nodes needed for oscillator simulation.

9. Click the *Oscillator node Select* button. Then click the appropriate wire in the Schematic window to choose the oscillator node. In this example, choose the node labelled `1`.

`/1` appears in the *Oscillator node* field.

10. Click the *Reference node Select* button. Then click the appropriate wire in the Schematic window to choose the reference node. In this example, choose the node where the ground is located.

11. `/gnd!` appears in the *Reference node* field.

You can also leave the *Reference node* field empty as the *Reference node* field defaults to `/gnd!`.

The bottom of the Choosing Analyses form for PSS looks like the following.

12. Verify that *Enabled* is checked for the `PSS` analysis and click *Apply*.

Correct any errors reported for the `PSS` analysis after you click *Apply*.

Setting Up the Pnoise Analysis

1. At the top of the Choosing Analyses form, click *pnoise*.

The form changes to let you specify data for the `Pnoise` analysis.

2. In the *Sweeptype* cyclic field, choose *Relative*. Enter `1` in the *Relative Harmonic* field.

This choice specifies that the *Frequency Sweep Range (Hz)* values you choose represent frequency values away from the fundamental frequency. For example, if you specify `2K`, you choose a value `2K` away from the fundamental frequency. This example uses 1, the first harmonic, because this is the best way to present phase noise for an oscillator which shows up next to the oscillator's fundamental frequency.

3. In the *Output Frequency Sweep Range(Hz)* cyclic field, choose *Start-Stop* and type `5` and `10k` as the *Start* and *Stop* values, respectively.

4. In the *Sweep Type* cyclic field, choose *Logarithmic*. Use a logarithmic sweep when you plot phase noise because of the size of the phase noise values.

5. Highlight *Number of Steps* and type `501` in the *Number of Steps* field.

Note: Be sure to use a nonzero starting value for a logarithmic sweep.

This example uses 501 steps, which allows you to see all the small features and get very accurate noise calculations. Because this is a very small circuit, we can use any number of steps. Your circuit might require a different number of steps.

In an actual circuit, it is typically more efficient to start off by setting *Number of Steps* to 2 or 3, as you iterate through the simulations needed to determine an appropriate value for the *Maximum sideband*. After that value is determined, you can set the *Number of Steps* to whatever value you need to obtain an accurate noise calculation.

6. In the *Sidebands* cyclic field, choose *Maximum sideband* and type `30` in the *Sidebands* field.

Be sure you always use a nonzero *Maximum sideband* value. The default value is `7`. A large *Maximum sideband* value provides more accurate noise calculations by allowing for more harmonic folding. (In this small circuit, however, the large *Maximum sideband* value provides no advantage.)

In an actual circuit, you determine an appropriate value for *Maximum sideband* by experimentation, gradually increasing the value in a series of simulations until the change in additional accuracy diminishes to an amount that satisfies your requirements. As noted in step 5, you can run this series of simulations with a reduced *Number of Steps* and, after determining the *Maximum sideband*, increase the *Number of Steps* to obtain an accurate noise calculation.

The top of the Choosing Analyses form for the Pnoise analysis looks like this.

7. In the *Output* cyclic field, choose *voltage*.
8. Notice the informational message that displays in the CIW.

```
*info*  Since voltage was selected as the output measurement
        technique for the pnoise analysis, spectre will NOT
        subtract any load resistance contribution from the Noise
        Figure calculation. If this is undesirable, please select
        probe as the output measurement technique and select a port.
```

9. Click the *Positive Output Node Select* button. Then click the appropriate wire in the Schematic window to choose */1*.

`/1` appears in the *Positive Output Node* field.

10. Click the *Negative Output Node Select* button. Then click the appropriate wire in the Schematic window to choose the ground.

`/gnd!` appears in the *Negative Output Node* field.

11. In the *Input Source* cyclic field, choose *none*.

In this case, because we are calculating Total Noise and its AM and PM components, the Input Source is not needed. (Using an Input Source generates NF, IRN, and other parameters.) Input Source is not usually used in oscillator simulations.

12. In the *Noise Type* cyclic field, choose *modulated*.

The following text appears below the *Noise Type* cyclic field.

modulated: separation into USB, LSB, AM, and PM components

The bottom of the Choosing Analyses form for the `Pnoise` analysis looks like this.

13. Verify that the *Enabled* field for the `Pnoise` analysis is highlighted. Then click *Apply* at the top of the Choosing Analyses form.
14. Correct any errors reported for the `Pnoise` analysis after you click *Apply*.
15. Click *OK*.

    The Choosing Analyses form closes.

    The `pss` and `pnoise` analyses you set up appear in the *Analyses* list box in the ADE window.

Running the Simulation

1. In the ADE window, choose *Simulation – Netlist and Run* to run the simulation.

   The output log file appears and displays information about the simulation as it runs.
2. Look in the CIW for a message that says the simulation completed successfully.

Plotting Modulated Pnoise with the dBc Modifier

The dBc modifier on the Direct Plot form plots the noise power in decibels, relative to the carrier, on the y axis.

To plot the modulated pnoise, you start by plotting USB noise and then plot AM and PM noise in the same waveform window.

➤ If necessary, open the Direct Plot form. by choosing *Results – Direct Plot – Main Form* in the ADE window.

The Direct Plot form appears.

Plotting USB Noise

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pnoise modulated* for *Analysis*.

   This selects the results of the modulated pnoise analysis for display.
3. Highlight *USB* for *Noise Type*.

   This selects to plot the standard pnoise analysis results.
4. Highlight *Output Noise* for *Function*.

   This selects to plot Output noise.
5. Highlight *dBc* for *Modifier*.

   Selecting dBc plots the noise power in decibels, relative to the carrier, on the y axis.

The Direct Plot form appears as follows.

6. Follow the prompt at the bottom of the Direct Plot form

   Press plot button on this form...

   and click *Plot*.

The waveform window displays the USB noise function.

The upper sideband noise appears in the Direct Plot form.

Plotting AM Noise

Plot the AM noise in the same waveform window from the same Direct Plot form.

1. Highlight *Append* for *Plot Mode*.
2. Keep *pnoise modulated* for *Analysis*.
3. Highlight *AM* for *Noise Type*.

   When you highlight *AM* for *Noise Type*, all *Function* selections other than *Output Noise* go away.

4. Keep *dBc* for *Modifier*.

The Direct Plot form appears as follows.

5. Follow the prompt at the bottom of the Direct Plot form

Press plot button on this form...

and click *Plot*.

The waveform window displays the USB and AM noise functions.

Plotting PM Noise

Finally, plot the PM noise in the same waveform window from the same Direct Plot form.

1. Keep *Append* for *Plot Mode*.
2. Keep *pnoise modulated* for *Analysis*.
3. Highlight *PM* for *Noise Type*.
4. Keep *Output Noise* for *Function*.
5. Keep *dBc* for *Modifier*.

The Direct Plot form appears as follows.

6. Follow the prompt at the bottom of the Direct Plot form

Press plot button on this form...

and click *Plot*.

The waveform window displays the USB, AM, and PM noise functions.

Measuring Jitter with PSS and Pnoise Jitter Analyses

As an example, consider *oscDiff*, the feedback amplifier circuit shown in Figure A-4 on page 1792.

Figure A-4 Schematic for the OscDiff Circuit

Opening the oscDiff Circuit

1. In the Command Interpreter Window (CIW), choose *File – Open*.

   The Open File form appears. Filling in the Open File form opens the oscDiff schematic.

2. For *Library Name,* choose *my_ rfExamples* or whatever you called the editable copy of the *rfExamples* library you created.

3. Choose *schematic* for *View Name*.

4. In the *Cells* list box, highlight *oscDiff*.

   oscDiff appears in the *Cell* field.

5. Highlight *edit* for *Mode*.

   The completed Open File form appears like the one below.

6. Click *OK*.

The Schematic window for the `oscDiff` oscillator opens.

See "The oscDiff Circuit: A Balanced, Tunable Differential Oscillator" on page 1754 for information about this circuit.

Opening the Simulation Window

1. In the Schematic window, choose *Tools – Analog Environment*.

The ADE window appears.

You can also choose *Tools – Analog Environment – Simulation* in the CIW to open the ADE window without opening the design. You can open the design later by choosing *Setup – Design* in the ADE window and choosing *oscDiff* in the Choosing Design form.

Editing Design Variables

1. In the ADE window, choose *Variables – Edit*.

   The Editing Design Variables form appears. For this example, you use this form to set the value of the variable *Ctune*.

2. In the Editing Design Variables form, in the *Table of Design Variables* list box, highlight *Ctune*.

   Ctune displays in the *Name* field.

3. Type `3.5p` in the *Value (Expr)* field.

4. Click *Change*.

   The completed form looks like this.

5. Click *OK*.

   Ctune appears in the *Design Variables* section of the ADE window with its modified value

Setting Up the Simulation

1. If necessary, see "Opening the oscDiff Circuit in the Simulation Window" on page 1755.
2. If necessary, see "Choosing Simulator Options" on page 1757.
3. If necessary, see "Setting Up Model Libraries" on page 1758.
4. If necessary, in the ADE window use *Analysis - Disable* to disable any analyses you ran previously. (Check the *Analyses* area in the ADE window to verify whether or not any analyses are enabled.)

Specifying Outputs to Save

1. In the ADE window, choose *Outputs – Save All.*

   The Save Options form appears.

2. In the *Select signals to output section*, be sure *allpub* is highlighted.

3. Click *OK*.

Setting Up the PSS Analysis

Note: If your oscillator circuit does not contain a stimulus to start the oscillator, you must run a `transient` analysis before you run this `PSS` analysis. From the `transient` analysis, you

save the node voltages and use them as initial conditions to start the oscillator in the `PSS` analysis.

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, choose *pss* for the analysis.

   The form changes to display options needed for the `PSS` analysis. You perform a `PSS` analysis first because the periodic steady state solution must be determined before you can perform a `Pnoise` small-signal analysis to determine the phase noise.

3. In the *Fundamental Tones* area, the *Beat Frequency* button is highlighted by default. Be sure the *Auto Calculate* button is *not* highlighted.

4. In the field next to the *Beat Frequency* and *Beat Period* buttons, type your best estimate of the oscillation frequency. (Your estimate can be the node voltages saved from the transient analysis mentioned above.)

   For this example, type `1.9G`.

5. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type `10` in the field.

The top of the PSS Choosing Analyses form appears below.

6. Highlight *conservative* for the *Accuracy Default (errpreset)*.
7. Type `100.6n` in the *Additional Time for Stabilization (tstab)* field.

   You can use a slightly different value as long as it is above 100.
8. If you want to save the results of the initial transient analysis, set *Save Initial Transient Results (saveinit)* to *yes*.

9. Highlight *Oscillator*.

   The form changes to let you specify the two nodes needed for oscillator simulation.

10. Click the *Oscillator node Select* button. Then click the appropriate wire in the Schematic window to choose the oscillator node. In this example, choose the node labelled *VoR*.

    `/VoR` appears in the *Oscillator node* field.

11. Click the *Reference node Select* button. Then click the appropriate wire in the Schematic window to choose the reference node. In this example, choose the node labeled *VoL*.

    `/VoL` appears in the *Reference node* field. Because this is a differential oscillator, the reference node is not `/gnd!` as it was in the *tline3oscRF* example.

    The bottom of the Choosing Analyses form for PSS looks like the following.

12. Verify that *Enabled* is highlighted for the `PSS` analysis and click *Apply*.

Setting Up the Pnoise Analysis

1. At the top of the Choosing Analyses form, click *pnoise*.

   The form changes to let you specify data for the `Pnoise` analysis.

2. In the *Sweeptype* cyclic field, choose *relative*. Enter 1 in the *Relative Harmonic* field.

This choice specifies that the *Frequency Sweep Range (Hz)* values you choose represent frequency values away from the fundamental frequency. For example, if you specify `2K`, you choose a value 2K away from the fundamental frequency.

3. In the *Output Frequency Sweep Range(Hz)* cyclic field, choose *Start-Stop* and type `100` and `200M` as the *Start* and *Stop* values, respectively.
4. In the *Sweep Type* cyclic field, choose *Logarithmic*. Use a logarithmic sweep when you plot phase noise because of the size of the phase noise values.
5. Highlight *Points Per Decade* and type `5` in the field.

   Be sure to use a nonzero starting value for logarithmic sweeps.
6. In the *Sidebands* cyclic field, choose *Maximum sideband* and type `20` in the *Sidebands* field. The default value is 20.

   Be sure to use a nonzero *Maximum sideband* value.

   The top of the Choosing Analyses form for the `Pnoise` analysis looks like this.

7. In the *Output* cyclic field, choose *voltage*.

8. Notice the informational message that displays in the CIW.

```
*info*  Since voltage was selected as the output measurement
        technique for the pnoise analysis, spectre will NOT
        subtract any load resistance contribution from the Noise
        Figure calculation. If this is undesirable, please select
        probe as the output measurement technique and select a port.
```

9. Click the *Positive Output Node Select* button. Then click the appropriate wire in the Schematic window to choose *VoR*.

   `/VoR` appears in the *Positive Output Node* field.

10. Click the *Negative Output Node Select* button. Then click the appropriate wire in the Schematic window to choose *VoL*.

    `/VoL` appears in the *Negative Output Node* field. Because this is a differential oscillator, the negative output node is not `/gnd!` as it was for the *tline3oscRF* oscillator.

11. In the *Input Source* cyclic field, choose *current*.

12. Click the *Input Current Source Select* button. Then click the appropriate component in the Schematic window to choose /I2.

    `/I2` appears in the *Input Current Source* field.

13. In the *Reference side-band* cyclic field, choose *Enter in Field* and type `-1` in the *Reference side-band* field.

14. In the *Noise Type* cyclic field, choose *Jitter.*

15. The following information displays.

    jitter: jitter measurement at the output

    FM jitter for autonomous circuit

The bottom of the Choosing Analyses form for the `Pnoise` analysis looks like this.

16. Verify that the *Enabled* field at the bottom of the Pnoise form is highlighted. Then click *Apply* at the top of the Choosing Analyses form.

17. Click *OK*.

    The `pss` and `pnoise` analyses you set up appear in the *Analyses* list box in the ADE window.

Analyses

	Type	Enable	Arguments
1	pss	✓	1.9G 10 /VoR /VoL
2	pnoise	✓	20 100 200M 5 /VoR /VoL

Running the Simulation

1. In the ADE window, choose *Simulation – Netlist and Run* to run the simulation.

The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Measuring Jitter

Plotting Phase Noise

1. Choose *Results – Direct Plot – Main Form* in the ADE window.

   The Direct Plot form appears.

2. Highlight *Replace* for *Plotting Mode*.
3. Highlight *pnoise jitter* for *Analysis*.

   This selects the results of the pnoise jitter analysis for display and changes the appearance of the Direct Plot form.

4. Highlight *Phase Noise* for *Function*.
5. Highlight *dBc* for *Modifier*.

The Direct Plot form looks like this.

6. In the Direct Plot form, click *Plot*.
7. In the waveform window, set the x axis to a logarithmic scale.

   Using a logarithmic scale on the x axis makes it easier for you to figure out the range of frequencies over which to measure simple jitter.

   a. In the waveform window, select the x axis.

   b. Choose *Axis – Log* to choose a logarithmic scale for the x axis.

The waveform window displays the phase noise curve.

Find the White FM Noise or the 1/f^2 Range of the Phase Noise

The `-20dB/decade` function provides a way to find the frequency range where white FM noise is significant. The function also makes it also possible to see the presence of the flicker noise in the system.

1. In the Direct Plot Form, select *Append* in the *Plotting Mode* cyclic field.
2. Highlight *pnoise jitter* for *Analysis*.
3. Highlight *dBc* for *Modifier*.
4. Highlight *-20dB/dec Line* for *Function*.

   The form changes to display the *Integration Limits* section.

5. Set the *Frequency (Hz)* to 1M., a value in the range where the shape of the phase noise curve is changing from `1/f` to `1/f`^{2} to `1/f`^{3}.

6. Ignore the *Start Frequency (Hz)* and *Stop Frequency (Hz)* fields. They are not used in the plotting step that follows.

   The Direct Plot form appears as follows.

7. Follow the prompt at the bottom of the Direct Plot form

Press plot button on this form...

and click *Plot*.

The waveform window displays both the phase noise curve and the -20dB/decade line.

8. Estimate the $1/f^2$ section of the phase noise curve. You can see the FM flicker noise at the frequencies below the $1/f^2$ range (200-300 kHz).

Plotting Cycle-to-Cycle Jitter

1. In the Direct Plot Form, select *Replace* in the *Plotting Mode* cyclic field.
2. Highlight *pnoise jitter* for *Analysis*.
3. Highlight *Jcc* for *Function*.

   The form changes to add fields where you can enter more data.

4. The *Number of Cycles [k]* defaults to 1.
5. Highlight *rms* for *Signal Level*.
6. Highlight *Second* for *Modifier*.

   Select the *Modifier* value based on the units you are using in your design specifications.
7. The *Freq. Multiplier* field defaults to 1.
8. Fill in the *Start Frequency (Hz)* and *Stop Frequency (Hz)* fields.

   The appropriate values depend on the methodology. Sometimes appropriate values are included in the jitter specifications or in the verification methodology. The best values to use also depend on the circuit properties. For a small *Number of Cycles [k]*, such as a short observation time of one or a few periods, the lower frequency noise does not contribute much to either Jc or Jcc jitter. For a larger *Number of Cycles [k]*, the lower frequency noise and the lower limit of integration are important. See *Application Note. Long Term jitter measurements* for information on how to set up the lower limit of the integration.

   To continue the example, type 100K into the *Start Frequency (Hz)* field and type 40M into the *Stop Frequency (Hz)* field.

The Direct Plot form looks like this.

9. Click *Plot*.

10. The waveform window displays the Jcc value.

The Jcc value is added to the waveform.

The phase noise is numerically integrated during the jitter computations. It accounts for all noise contributors that exist in the schematic.

There is a significant difference between RMS phase jitter and cycle or cycle-to-cycle jitter.

- RMS jitter is easily generated by integrating the phase noise over the required frequency range. The contributions from different frequency ranges correspond to the PSD of the phase noise.
- The situation is different for Jc and Jcc. Regardless of the phase noise PSD, the lower frequency contributions are mostly negligible for the small K cycle, short term jitter computations and become important only as the observation time increases.

Troubleshooting for Oscillator Circuits

If your oscillator simulation fails to converge, try the following techniques to help the SpectreRF simulator find the solution. You can find more information about each of these techniques in the SpectreRF Theory document available in the online documentation library.

- Increase the value of the `tstab` parameter.
- Be sure that you successfully started the oscillator.

  PSS fails to converge if the circuit cannot sustain an autonomous oscillation. Start the oscillator so that it responds with a signal level between 25 percent and 100 percent of the expected final level. Also, do not “kick” the oscillator so hard that its response is unnecessarily nonlinear. If possible, avoid exciting response modes that are unrelated to the oscillation, especially those associated with longtime constants.
- Improve your estimate of the period.

  Be careful that the period you specify is not too short. Overestimating the period is better than underestimating the period.
- When possible, use *gear2only* as the value of the `method` parameter.
- Increase the value of the `maxperiods` parameter to increase the maximum number of iterations for the Shooting methods.
- If the Shooting iteration approaches convergence and then fails, increase the value of the `steadyratio` parameter.

  The `steadyratio` parameter controls how much the final solution can deviate from being periodic.
- Change the value of the `tolerance` parameter for periodic small-signal analyses.

  Changing the value of the `tolerance` parameter does not change the accuracy of the final solution, but it does increase the chance that the periodic small-signal analyses converge.
- Change the value of the `itres` parameter for the large-signal PSS analyses.

  The `itres` parameter adjusts the level of accuracy used when solving nonlinear systems of equations. Loosen the `itres` parameter but make sure you get convergence for the nonlinear problem. If your PSS simulation converges, the final result is accurate.

  The default value for the `itres` parameter is $1e^{-4}$ which can sometimes be too tight for particular circuits. You can the increase the `itres` parameter value as long as it remains less than 1.

Tighten the normal simulation tolerances by changing the values of the `maxstep`, `reltol`, `iteratio`, and `errpreset` parameters. Avoid setting `errpreset` to `liberal`.

Mixers

The SpectreRF simulator can simulate circuits, such as mixers, that show frequency conversion effects. This chapter uses a commercially available integrated circuit mixer, the ne600p, configured as a down converter to illustrate how the SpectreRF simulator can determine the characteristics of your design. The ne600p circuit examples illustrate both the capabilities and the requirements of the SpectreRF simulator. The examples show you how to perform a simulation with an RF to IF ratio of almost 12 to 1. The examples also show you how to be sure that all time-varying independent signal sources have a common integral multiple (40 MHz in this case).

In the mixer examples, you plot the following nonlinear characteristics of the `ne600p` mixer.

Measurements	Analyses
Total Harmonic Distortion Measurement with PSS	PSS
Compression Distortion Summary with PSS and PAC	PSS and PAC
Rapid IP3 Measurement with PSS and PAC	PSS and PAC
Noise Figure Measurement with PSS and Pnoise	PSS and Pnoise
Noise Figure Measurement and Periodic S-parameter Plots with PSS and PSP	PSS and PSP
Conversion Gain and Power Supply Rejection with PSS and PXF	PSS and PXF
Calculating the 1 dB Compression Point with Swept PSS	Swept PSS
Third-Order Intercept Measurement with Swept PSS and PAC	Swept PSS and PAC
Intermodulation Distortion Measurement with QPSS	QPSS
Noise Figure with QPSS and QPnoise	QPSS and QPnoise
IP3 and Compression Distortion Summary Measurements for a Mixer	PSS and Specialized PAC
Rapid IP2 and IM2 Distortion Measurements for a Gilbert Direct Conversion Mixer	PSS and Specialized PAC
Rapid IP3 and Compression Distortion Summary Measurements for a Power Amplifier	Specialized AC

To use this chapter, you must be familiar with the SpectreRF simulator analyses as well as know about mixer design. For more information about the SpectreRF simulator analyses, see *SpectreRF Simulation Option Theory*.

To make these measurements, you run one of the following combination of analyses.

- A Periodic Steady-State analysis (PSS), and a periodic small-signal analysis: Periodic AC (PAC) analysis, Periodic Transfer Function (PXF) analysis, Periodic Noise (Pnoise) analysis, or Periodic S-parameter (PSP) analysis.
- A Quasi-Periodic Steady-State (QPSS) Analysis and the small-signal Quasi-Periodic Noise (QPnoise) analysis.

The ne600p Mixer Circuit

The ne600p integrated circuit is commercially available and is designed for low power communication systems from 800-1200 MHz. The receiver is a single balanced mixer containing four bipolar transistors. The schematic for the ne600p mixer circuit is shown in Figure A-5.

Figure A-5 Schematic for the ne600p Mixer Circuit

The high-level local oscillator (LO) signal (~0 dBm at 50 Ohms) enters through the emitter follower q57. The transistor q56 is in common base configuration. The q56 emitter follows the q57 emitter. When the LO signal goes high, both emitters also go high and the B-E voltage on q56 decreases. As a result, the two BJTs form a balanced pair.

The collector currents of q56 and q57 are 180 degrees out of phase. The q58 pair feed the low-level (-30 to -20 dBm) RF signal to the balanced pair. The IF signal is drawn from the open collector output of q56 through a bandpass passive network.

The following tables list some measured values for different aspects of the ne600p mixer.

Measurement	Measured
LO frequency (Hz)	1 GHz
RF frequency (Hz)	920 MHz
IF frequency (Hz)	80 MHz
LO power	0 dBm
RF power	-30 dBm
Conversion gain	-2.6 dB
Noise figure	16 dB
Input 1dB compression point	-4 dBm
Input IP3 (from swept power)	6 dBm
Input IP3 (from PAC analysis)	6 dBm

Design Variable	Default Value
prf (RF power)	-30 dBm
vlo (LO magnitude)	316.2 m
frf (RF frequency)	920 MHz
flo (LO frequency)	1 GHz

Before you use the ne600p circuit from the sample library, be sure that the *frf* design variable is set to the appropriate value, either 920 MHz or 900 MHz, depending on the simulation. If you adjust the RF frequency to the center of the IF tuned circuit, approximately 100 MHz, the conversion gain becomes -2.57dB and the other values change slightly.

Setting Up to Simulate the ne600p Mixer

Before you start, perform the setup procedures described earler in this manual.

Opening the ne600p Mixer Circuit in the Schematic Window

In the CIW, choose *File – Open*.

The Open File form appears.

1. In the Open File form, choose *my_rfExamples* in the *Library Name* menu. Choose the editable copy of the rfExamples library you created as described in earlier in this manual.
2. Choose *ne600p* in the *Cells* list box.

   The completed Open File form appears like the one below.

3. Click *OK*.

The Schematic window for the *ne600p* mixer appears.

4. In the Schematic window, choose *Launch– ADE L*.

The Virtuoso Analog Design Environment window opens.

You can also choose *Tools – Analog Environment – Simulation* in the CIW to open the Virtuoso Analog Design Environment window without opening the design. You can open the design later by choosing *Setup – Design* in the Virtuoso Analog Design Environment window and choosing the *ne600p* in the Choosing Design form.

Choosing Simulator Options

1. Choose *Setup – Simulator/Directory/Host* in the Virtuoso Analog Design Environment window.

   The Choosing Simulator/Directory/Host form appears.

2. Specify the following:

   a. Choose *spectre* for the *Simulator*.

   b. Type the name of the project directory, if necessary.

   c. Highlight the *Host Mode* that corresponds to your situation.

   d. If you choose *remote* or *distributed*, fill in the additional fields as necessary.

The completed form appears similar to the one below.

3. Click *OK*.

The Choosing Simulator/Directory/Host form closes.

4. In the Virtuoso Analog Design Environment window, choose *Outputs – Save All*.

The Save Options form appears.

5. In the *Select signals to output* section, be sure *allpub* is highlighted.

6. Click *OK*.

   The Save Options form closes.

Setting Up Model Libraries

1. In the Virtuoso Analog Design Environment window, choose *Setup – Model Libraries*.

The Model Library Setup form appears.

2. In the *Model Library File* field, type the full path to the model file including the file name, `<CDSHOME>/tools/dfII/samples/artist/models/spectre/rfModels.scs`.
3. Click *Add*.

   The Model Library Setup form looks like the following.

4. Click *OK*.

   The Model Library Setup form closes.
5. In the Virtuoso Analog Design Environment window, disable any analyses you ran previously.

Setting Design Variables

Use the following procedure to set the design variables to the values required for each simulation. (See the description of each simulation for the required value.)

This example sets values for `frf` and `flo`.

1. In the Virtuoso Analog Design Environment window, highlight `frf` in the *Design Variable* list box.
2. Choose *Variables – Edit*.

3. The Editing Design Variables form appears.

4. Highlight `frf` in the *Table of Design Variables* list box.
5. In the *Value (Expr)* field, type `920M` for the value of `frf` and click *Change*.
6. Highlight `flo` in the *Table of Design Variables* list box.
7. In the *Value (Expr)* field, type `1G` for the value of `flo` and then click *Change*.
8. In the Editing Design Variables form, click *OK*.

   The form closes.

Total Harmonic Distortion Measurement with PSS

In this example, a single PSS analysis determines the Total Harmonic Distortion (THD) for the mixer.

Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)

4. If necessary, set the design variables `frf` to `920M` and `flo` to `1G`. (Check the Design Variables section to verify the current design variable values.)

Editing the Schematic

A critical part of this analysis is the correct use of the programmable voltage sources in the `ne600p` circuit. The RF voltage source is based on the *port* sample component. You must change the behavior of this component for each analysis.

1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for the *rf* voltage source and modify the schematic for this simulation.

2. In the Schematic window, click the *rf* voltage source.

The Edit Object Properties form changes to display information for the voltage source.

3. In the Edit Object Properties form, be sure that *sine* is specified as the *Source type*.

A portion of the completed Edit Object Properties form for the rf port appears like the one below:

4. In the Edit Object Properties form, click *OK*.
5. In the Schematic window, choose *Design – Check and Save*.

Setting Up the PSS Analysis

1. Choose *Analyses – Choose* in the Virtuoso Analog Design Environment window.

   The Choosing Analyses form appears.
2. In the Choosing Analyses form, click *pss*.
3. In the *Fundamental Tones* list box, *Beat Frequency* is highlighted by default. Be sure *Auto Calculate* is also highlighted.

   The *Beat Frequency* is calculated. The `40M` value is the minimum period for which both RF at `920 MHz` and LO at `1 GHz` are periodic (or for which RF and LO are integer multiples of the fundamental frequency).
4. In the *Output harmonics* cyclic field, choose *Number of harmonics*. Type `30` in the *number of harmonics* field.

   This entry expands the plotted frequency range to include the areas of interest around both `40 MHz` and `1 GHz`. (`30` x `40 MHz` equals `1.2 GHz`, or `0` to `1.2 GHz`, and includes all important frequencies.)

The top of the PSS Choosing Analyses form is as follows.

5. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.
6. Verify that *Enabled* is highlighted.

   The bottom of the PSS Choosing Analyses form is as follows.

7. Click *Apply*.

8. Click *OK.*

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting and Calculating Total Harmonic Distortion

1. Choose *Results – Direct Plot – Main Form* in the Virtuoso Analog Design Environment window.

   The Direct Plot form appears.

2. In the Direct Plot form, select *Append* in the *Plotting Mode* cyclic field.
3. Highlight *pss* for *Analysis*.
4. Highlight *Voltage* for *Function*.
5. In the *Select* cyclic field, select *Differential Nets* to plot voltage against frequency.

   The message at the bottom of the form changes.

6. For *Sweep*, highlight *spectrum*.
7. For *Signal Level*, highlight *peak*.
8. For *Modifier*, highlight *dB20*.

The Direct Plot form appears as follows.

9. To plot the voltage against frequency, click the *Pif* and *Prf* wires in the Schematic Window.

a. Following the instructions at the bottom of the Direct Plot Form,

```
Select Positive Net On Schematic...
```

Select the positive net, *Pif*, on the schematic.

Pif is highlighted in the schematic window and the instructions at the bottom of the Direct Plot form change.

b. Following the instructions at the bottom of the Direct Plot Form,

```
Select Negative Net on schematic...
```

Select the negative net, *Prf*, on the schematic.

Prf is highlighted in the schematic window.

The waveform window display appears like the one below.

10. Measure the `Vif` values at the second, fourth, and sixth harmonics. That is take measurements at `80 MHz`, `160 MHz`, and `240 MHz`.

To measure `Vif` at each harmonic, place the cursor at the top of the second, forth, and sixth harmonic in turn. For the selected harmonic, the X value in MHz and Y value in dB appears at the bottom left corner of the waveform window.

Measurements for the three selected harmonics are as follows.

Harmonic Number	Frequency Value	Vif Value
$h2$ — the 2nd harmonic	80 MHz	-45.48 dB
$h4$ — the 4th harmonic	160 MHz	-111.1 dB
$h6$ — the 6th harmonic	240 MHz	-116.2 dB

11. Calculate the total harmonic distortion (THD) with the following formula:

$$THD = dB10[(h4)^2 + (h6)^2] - dB20(h2)$$

where

$h2$ is the value of `Vif` at the 2nd harmonic: -45.48 dB

$h4$ is the value of `Vif` at the 4th harmonic: -111.1 dB

$h6$ is the value of `Vif` at the 6th harmonic: -116.2 dB

The value of `THD` is 10.93768 `dB`.

Compression Distortion Summary with PSS and PAC

You can combine a PSS analysis with a small-signal PAC analysis to measure and print the compression distortion summary for selected components in the ne600p circuit.

PAC analysis calculates the distortion contributed by each selected component in the schematic.

The distortion summary is a quantitative measurement of how much distortion a device adds to the output signal. Distortion is measured as

$$dB = 10 \log\left(\frac{V_{out}}{c_1 V_{RF}}\right)$$

Where

V_{RF} and V_{out} are amplitudes of the input and output signals, respectively.

C_1 is the ideal linear conversion gain.

Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)

In this figure, the PSS analysis is disabled.

4. If necessary, set the design variables.

`flo`	1G
`prf`	-30
`frf`	920M

5. Check the Design Variables section to verify the current design variable values.

Design Variables

	Name	Value
1	ifres	50
2	re	450
3	rc	1K
4	prf	-30
5	vlo	316.2m
6	frf	920M
7	flo	1G
8	plo	-10

Editing the Schematic

Modify properties of the rf voltage source in the ne600p mixer schematic to be sure the PSS analysis response is limited to the LO signal only.

1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF properties and modify the schematic for this simulation.

2. In the Schematic Window, click the *rf* voltage source.

3. In the Edit Object Properties form, make the following changes and click *OK*.

4. Choose *dc* as the *Source type*, if necessary.

5. Click *Display small signal parameters* to display small signal parameters associated with the rf source.

6. Verify that the *PAC Magnitude (dBm)* small signal parameter is set and that the value is `prf`. The variable *`prf`* appears in the Variables section of the Virtuoso Analog Design Environment window.

   The *PAC magnitude (dBm)* parameter measures the magnitude of the RF input at the RF source.

   For non-RF sources, you should not use the *PAC magnitude (dBm)* parameter or you should set it to 0.

7. Click *OK*.

   The Edit Object Properties form closes.

8. In the Schematic window, choose *Design – Check and Save*.

Setting up the PSS and PAC Analyses

➤ In the Virtuoso Analog Design Environment window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

Setting Up the PSS Analysis

1. In the Choosing Analyses form, click *pss* for the *Analysis*.
2. At the lower right corner of the *Fundamental Tones* section, highlight *Auto Calculate*.

   The *Beat Frequency* is calculated as `1G`.

   For *Output Harmonics*, select *Number of harmonics* and enter 10.

   The top of the PSS Choosing Analyses form appears as follows.

3. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.
4. Highlight *Enabled*.

5. The bottom of the PSS Choosing Analyses form appears as follows.

6. Click *Apply* in the PSS Choosing Analyses form.

Setting Up the PAC Analysis

1. At the top of the Choosing Analyses form, highlight *pac*.

   The Choosing Analyses form changes to let you specify data for a PAC analysis.

2. In the *Input Frequency Sweep Range (Hz)* cyclic field, choose *Single-Point*.
3. Type `920M` in the *Freq* field.
4. In the Sidebands field, select Maximum Sideband and enter 10.

The top of the PAC Choosing Analyses form looks as follows.

5. In the *Specialized Analyses* cyclic field, choose *Compression Distortion Summary*.

   The PAC Choosing Analyses form changes to display fields for setting up the Compression Distortion Summary.

   A message also displays reminding you to set the pacmag parameter on the RF source in the schematic.

6. Click *Close* to dismiss the message and display the *Compression Distortion Summary* fields for the PAC analysis.

7. Click *Select*, just above the *Contributor Instances* field.

   A message appears in the Virtuoso Schematic Editing window, telling you to

   `Select contributor instances. Press Esc when done.`

8. Select the transistors in the mixer as the components to measure for the distortion summary.

   a. Select *q56 “NPNupper”* in the schematic.

   b. Select *q57 “NPNupper”* in the schematic.

   c. Select *q58a “NPNlower”* in the schematic.

   d. Select *q58b “NPNlower”* in the schematic.

9. Press *Esc*, when you are done selecting the transistors.

   The four selected components display in the *Contributor Instances* field.

10. In the *Frequency of Linear Output Signal* field type 80M.

11. In the *Maximum Non-linear Harmonics* field, type 4. This value specifies the maximum harmonic used to represent the nonlinear response to RF signals.

12. Next to *Output*, click *Voltage*.

13. Next to *Out+* click *Select*. Then click the appropriate net in the Schematic window to choose *Pif*.

`/Pif` appears in the *Out+* field. this is the output node for measurement.

Leave the *Out-* field empty.

The *Out-* field defaults to `/gnd!` You can set *Out-* to a different value by clicking on the *Select* button and then choosing an output node in the schematic.

The bottom of the PAC Choosing Analyses form looks as follows.

14. Highlight *Enabled*.

15. Click *Apply*.

16. Click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Printing the Compression Distortion Summary

➤ In the Virtuoso Analog Design Environment window, choose *Results - Print - PAC Distortion Summary*.

The Results Display Window opens and displays the Distortion Summary.

The PAC Compression Distortion Summary report helps you determine the amount of distortion generated by each selected component from your schematic.

Rapid IP3 Measurement with PSS and PAC

In this example, IP3 is quickly computed for the ne600p circuit using PSS and PAC analyses. This method uses extra function evaluations and small signal PAC analysis to evaluate higher order terms.

The rapid IP3 measurement computes the nonlinear response to RF signals around the periodically time-varying operating point. This IP3 measurement is much faster than the IP3 measurement procedure described in "Third-Order Intercept Measurement with Swept PSS and PAC" on page 1893.

Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)

In this figure, the PSS analysis is disabled.

4. If necessary, set the design variables.

flo	1G
prf	-30
frf	1.1G

5. Check the Design Variables section to verify the current design variable values.

Design Variables

	Name	Value
1	ifres	50
2	re	450
3	rc	1K
4	prf	-30
5	vlo	316.2m
6	frf	1.1G
7	flo	1G
8	plo	-10

Editing the Schematic

Modify properties of the rf voltage source in the ne600p mixer schematic to be sure the PSS analysis response is limited to the LO signal only.

1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF properties and modify the schematic for this simulation.

2. In the Schematic Window, click the *rf* voltage source.

3. In the Edit Object Properties form, make the following changes and click *OK*.

4. Choose *dc* as the *Source type*, if necessary.

5. In the Schematic window, choose *Design – Check and Save*.

Setting up the PSS and PAC Analyses

➤ In the Virtuoso Analog Design Environment window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

Setting Up the PSS Analysis

1. In the Choosing Analyses form, click *pss* for the *Analysis*.
2. At the lower right corner of the *Fundamental Tones* section, highlight *Auto Calculate*.

   The *Beat Frequency* is calculated as `1G`.
3. For *Output Harmonics*, select *Number of harmonics* and enter 10.

The top of the PSS Choosing Analyses form appears as follows.

4. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.
5. Highlight *Enabled*.

   The bottom of the PSS Choosing Analyses form appears as follows.

6. Click *Apply* in the PSS Choosing Analyses form.

Setting Up the PAC Analysis

1. At the top of the Choosing Analyses form, highlight *pac*.

   The Choosing Analyses form changes to let you specify data for a *PAC* analysis.

2. In the *Input Frequency Sweep Range (Hz)* cyclic field, choose *Start-Stop*.

3. Type `frf` in the *Start* field.

4. Type `frf+30M` in the *Stop* field.

   If the *Stop* field is grayed out, set *Specialized Analyses* to *None* before typing in the field.

5. In the *Sidebands* field, select *Maximum Sideband* and enter 2.

6. The top of the PAC Choosing Analyses form looks as follows.

7. In the *Specialized Analyses* cyclic field, choose *Rapid IP3*.

   The PAC Choosing Analyses form changes to display fields for setting up the Rapid IP3 calculation.

8. For the *Source Type* select *port*.

9. Select the RF sources and set up their frequencies.

   The second RF port can be the same as the first.

   a. To the right of *Input Sources 1* click *Select*. Then click the rf source in the Schematic window.

   b. Press *esc* to display `/rf` in the *Input Sources 1* field.

   c. In the *Freq* field, enter the input frequency `frf`.

      This is the same as the *Start* value.

   d. To the right of *Input Sources 2* click *Select*. Then click the same rf source in the Schematic window.

   e. Press *esc* to display `/rf` in the *Input Sources 2* field.

   f. In the *Freq* field, enter the input frequency `frf+30M`.

      This is the same as the *Stop* value.

10. In the *Input Power (dBm)* field, enter `prf`. This input power value applies to both sources.

11. In the *Frequency of IM Output Signal* field, type `160M`, which is the IM3 frequency.

12. In the *Frequency of Linear Output Signal* field, type `100M`, which is the IM1 frequency.

13. In the *Maximum Non-linear Harmonics* field, type 4. This number specifies the maximum harmonic used to represent nonlinear response to RF signals.

14. Specify the output nodes for IP3 measurement.

    a. Next to *Output* highlight *Voltage*.

    b. To the right of the *Out+* field, click *Select* and select the appropriate net in the Schematic window to choose *Pif*.

       `/Pif` appears in the *Out+* field. This is the output node for measurement.

       Leave the *Out-* field empty.

       This field defaults to `/gnd!` You can set *Out-* to a different value by clicking on the *Select* button and then choosing an output node in the schematic

The bottom of the PAC Choosing Analyses form looks as follows.

15. Highlight *Enabled*.
16. Click *Apply*.
17. Click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the Rapid IP3 Curve

1. In the Virtuoso Analog Design Environment window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.

2. Select *Replace* in the *Plotting Mode* cyclic field.
3. Highlight *pac* for *Analysis*.

   The form changes to display information for the PAC analysis.
4. Highlight *Rapid IP3* for *Function*.

   The form changes again.
5. Highlight *Add to Outputs*.

   Notice that `freqaxis` = `absout` for this plot.
6. Follow the prompt at the bottom of the form

   `Press plot button on this form...`
7. Press *Plot*.

The waveform window display appears as shown below.

Noise Figure Measurement with PSS and Pnoise

You can combine a PSS analysis with a small-signal Pnoise analysis to determine the noise figure for the ne600p circuit.

Pnoise analysis calculates the total noise at the output of the circuit. The equation for the noise figure is given in the SpectreRF Theory document. The SpectreRF Pnoise analysis

computes the single sideband noise figure (`-1` in this case). The total noise can vary with the number of harmonics you choose because each harmonic contributes a noise component.

Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)

In this figure, the analysis is disabled.

If necessary, set the design variables `frf` to `920M` and `flo` to `1G`. (Check the Design Variables section to verify the current design variable values.)

Editing the Schematic

Modify the schematic to be sure the PSS analysis is the response of the ne600p mixer to *only* the LO signal.

1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF properties and modify the schematic for this simulation.

2. In the Schematic Window, click the *rf* voltage source.

3. In the Edit Object Properties form, choose *dc* as the *Source type*, if necessary, and click *OK*.

4. In the Schematic window, choose *Design – Check and Save*.

Setting up the PSS and Pnoise Analyses

➡ In the Virtuoso Analog Design Environment window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

Setting Up the PSS Analysis

1. In the Choosing Analyses form, click *pss* for the *Analysis*.
2. At the lower right corner of the Fundamental Tones section, highlight *Auto Calculate*.

   The *Beat Frequency* is now displayed as `1G`. The *Beat Frequency* button is highlighted by default.
3. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type 0 in the field.

   Note: Pnoise requires that you set the *Number of harmonics* value to `0` to determine the circuit's response to LO only.

   The top of the PSS Choosing Analyses form appears as follows.

4. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.

5. Highlight *Enabled*.

   The bottom of the PSS Choosing Analyses form appears as follows.

6. Click *Apply*.

Setting Up the Pnoise Analysis

1. At the top of the Choosing Analyses form, highlight *pnoise*.

   The Choosing Analyses form changes to let you specify data for a *Pnoise* analysis.

2. In the *Output Frequency Sweep Range (Hz)* cyclic field, choose *Start-Stop*. Type `1K` in the *Start field* and `2G` in the *Stop* field.

   This frequency range covers the frequencies of interest, but avoids the value 0. You cannot include the value 0 in a logarithmic sweep.

3. In the *Sweep Type* cyclic field, choose *Logarithmic* for the sweep type, highlight *Points Per Decade* and type `10` in the *Points Per Decade* field.

4. In the *Sidebands* section, choose *Maximum sideband* and type `30` for the number of sidebands.

   With this setting, you specify that 30 sidebands contribute noise to the output. The value is taken from the first example (*PSS analysis only*) in this chapter.

   The top of the Pnoise Choosing Analyses form looks as follows.

5. In the *Output* cyclic field, choose *voltage* for the *Output* value.
6. Click the *Select* button for *Positive Output Node*, then click the appropriate wire in the Schematic window to choose *Pif*.

   `/Pif` appears in the *Positive Output Node* field.
7. Leave the *Negative Output Node* field empty.

   This field defaults to `/gnd!` You can set the *Negative Output Node* to a different value by clicking on the *Negative Output Node Select* button and then choosing the output node in the schematic.
8. In the *Input Source* cyclic field, choose *voltage*.
9. Click the *Select* button for *Input Voltage Source*, then click the *rf* port in the Schematic window.

   `/rf` appears in the *Input Voltage Source* field.
10. In the *Reference Side-band* cyclic field, choose *Enter in field*.
11. Type `-1` in the *Reference Sideband* field.

    In this field, you specify the difference between the input and output frequencies in the whole *frf*. The *Reference Sideband* must be `-1` because this is a down converter. Other possible choices are `0` and `+1`.
12. Leave *sources* in the *Noise Type* cyclic field to plot single sideband noise.
13. Highlight *Enabled*.

The Pnoise Choosing Analyses form looks as follows.

14. Click *Apply*.

15. Click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the Noise Figure

1. To open the Direct Plot form, choose *Results – Direct Plot – Main Form* in the Virtuoso Analog Design Environment window.
2. In the Direct Plot form, select *Replace* in the *Plotting Mode* cyclic field.
3. Highlight *pnoise* for *Analysis*.

   The Direct Plot form changes to display choices for results of the Pnoise analysis.
4. Highlight *Noise Figure* for *Function*.

   The completed Direct Plot form appears like the one below.

5. Follow the prompt at the bottom of the form

   `Press plot button on this form...`

Click *Plot* in the Direct Plot form.

The plot appears in the waveform window.

The waveform window displays the noise figure:

6. To determine the noise figure at different frequencies, move the cursor along the noise figure curve in the waveform window. In the above plot, the noise figure is around `16.31 dB` at `80 MHz`.

Noise Figure Measurement and Periodic S-parameter Plots with PSS and PSP

The periodic S-parameter (PSP) analysis computes scattering and noise parameters for n-port circuits that exhibit frequency translation. Such circuits include mixers, switched-capacitor filters, samplers, phase-locked loops, and similar circuits.

In this example, you follow a large-signal PSS analysis with a small-signal PSP analysis to analyze noise folding terms induced by the RF input for the ne600p mixer circuit.

This PSP analysis computes

- Periodic S-parameters

- Periodic noise correlation matrices
- Noise figure
- Equivalent noise parameters

The PSP analysis is a small-signal analysis like the SP analysis, except that, as is true for PAC and PXF analysis, the circuit is first linearized about a periodically varying operating point as opposed to a simple DC operating point. Linearizing the circuit about a periodically time-varying operating point allows for the computation of S-parameters between circuit ports that convert signals from one frequency band to another.

The PSP analysis can also calculate noise parameters in frequency-converting circuits. PSP computes noise figure (both single-sideband and double-sideband), input referred noise, equivalent noise parameters, and noise correlation matrices. The noise features of the PSP analysis include noise folding effects due to the periodically time-varying nature of the circuit. This is also true for the Pnoise analysis, but not for the SP analysis.

Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)

4. If necessary, set the design variables `frf` to `900M` and `flo` to `1G`. (Check the Design Variables section to verify the current design variable values.)

Design Variables

	Name	Value
1	ifres	50
2	re	450
3	rc	1K
4	prf	-30
5	vlo	316.2m
6	frf	900M
7	flo	1G
8	plo	-10

Editing the Schematic

The rf source used in the ne600p mixer circuit has a *Source type* property that you can set to either *sine* or *dc* depending on your application. Suppose that your RF input signal is at 900 MHz, the LO is at 1 GHz, and the IF is at 100 MHz. Before you perform a PSP analysis, you must first perform a PSS analysis.

In many cases for small-signal analysis, it is sufficient to treat the RF input as a small-signal. For example, most gain measurements are performed under small-signal conditions in which case you set the *Source type* to *dc*.

When it is important to analyze additional noise folding terms induced by the rf input, you treat the rf source as a large signal and set the *Source type* to *sine*.

In this example, you assume the rf source is a large signal so you edit the rf port on the schematic and set the *Source type* parameter to *sine*.

1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF properties and modify the schematic for this simulation.

2. In the Schematic window, click the *rf* voltage source.

   The Edit Object Properties form changes to display information for the voltage source.

3. In the Edit Object Properties form, choose *sine* as the *Source type*, if necessary, and click *OK*.

4. In the Schematic window, choose *Design – Check and Save*.

Setting up the PSS and PSP Analyses

Use the Choosing Analyses form, to set up two analyses, first PSS then PSP.

➤ Choose *Analyses – Choose* in the Virtuoso Analog Design Environment window.

The Choosing Analyses form appears.

Setting Up the PSS Analysis

1. At the top of the Choosing Analyses form, highlight *pss*.
2. In the *Fundamental Tones* list box, be sure the *Auto Calculate* button is highlighted.

The *Beat Frequency* is now displayed. (The *Beat Frequency* button is highlighted by default.) The 100 M value for beat frequency is the minimum period for which both RF at 900 MHz and LO at 1 GHz are periodic (or for which RF and LO are integer multiples of the fundamental frequency).

3. In the *Output harmonics* cyclic field, choose *Number of harmonics.* Type `30` in the *Number of harmonics* field.

This entry expands the plotted frequency range to include the areas of interest around both 100 MHz and 1 GHz. (30 $\times$ 100 MHz equals 3GHz, or 0 to 3GHz, and includes all important frequencies.)

4. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.

5. Be sure the *Enabled* checkbox is checked.

The correctly filled out form looks like this.

6. Click *Apply* to check the information you entered in the Choosing Analyses form for pss.

Setting Up the PSP Analysis

1. In the *Analysis* section at the top of the Choosing Analyses form, highlight *PSP*.

The Choosing Analyses form changes to let you specify data for the *PSP* analysis.

2. In the *Sweeptype* cyclic field, choose *relative*.

   The *Sweeptype* parameter controls the way that frequencies are swept in the PSP analysis. When you specify a *relative* sweep, the sweep is relative to the analysis harmonics (not the PSS fundamental frequency).

   For PSP analysis, the frequencies of the input and response are usually different. This is an important difference between the PSP and SP analyses. Because the PSP computation involves inputs and outputs at frequencies that are relative to multiple harmonics, the `freqaxis` and `sweeptype` parameters behave differently in PSP analysis than they behave in PAC and PXF analyses.

3. In the *Frequency Sweep Range (Hz)* cyclic field, choose *Start-Stop* and type `-20M` in the *Start* field and `30M` in the *Stop* field.

   The PSP analysis is performed from 20 MHz below the RF center frequency to 30 MHz above it. This frequency range accounts for inputs on the rf port in the range of -920 MHz to -870 MHz. Noise parameters (such as noise figure) are computed in a 50 MHz band around the frequency specified by the output harmonic.

4. Leave the *Sweep Type* cyclic field set to *Automatic*.

5. Highlight *Select Ports* to select the input and output ports on the schematic.

   The first port, the input port, is the *rf* port.

   The second port, the output port, is the *rif* port.

6. Select the input port.

   a. Type `1` in the first field, the *Port#* field. It is directly above the *Select Port* button.

b. Click *Select Port* and follow the prompt at the bottom of the Schematic window.

`Select source...`

c. In the Schematic window, click the appropriate port to choose `/rf`.

`/rf` appears in the *Name* field.

7. Click *Choose Harmonic*.

The Choose Harmonic form displays with a list of harmonics (by index and frequency) for the *rf* port.

8. In the Choose Harmonic form, scroll through the list and highlight the harmonic with index `-9`.

9. Click *OK*.

The Choose Harmonic form closes. In the *Select Ports* area of the Choosing Analyses form, `-9` displays in the *Harm*. field.

10. In the *Select Ports* area, click *Add*.

Information for the input port displays in the *Select Ports* list box.

11. Select the output port.

a. Type `2` in the first field, the *Port#* field. It is directly above the *Select Port* button.

b. Click *Select Port* and follow the prompt at the bottom of the Schematic window.

`Select source...`

c. In the Schematic window, click the appropriate port to choose `/rif`.

`/rif` appears in the *Name* field.

12. Click *Choose Harmonic*.

The Choose Harmonic form displays with a list of harmonics (by index and frequency) for the *rif* port.

13. In the Choose Harmonic form, scroll through the list and highlight the harmonic with index `1`.

14. In the Choose Harmonic form, click *OK*.

The Choose Harmonic form closes. In the *Select Ports* area of the Choosing Analyses form, `1` displays in the *Harm*. field.

15. In the *Select Ports* area, click *Add*.

Information for the output port is added to the *Select Ports* list box.

With the 1 GHz LO, small-signal inputs around 900 MHz (harmonics +/-9 of the PSS fundamental) appear at around 100 MHz (or harmonics -/+1 of the fundamental). You select harmonic 1 as the output harmonic, and harmonic -9 as the input harmonic because a single complex-exponential input

$$e^{i\omega_s t}$$

on the lower side of 900 MHz (harmonic -9) appears as the upper sideband of harmonic 1 at around 100 MHz.

Harmonics -9 and 1 are separated by 10 fundamental periods, which corresponds to the LO frequency of 1 GHz.

16. Below *Do Noise* near the bottom of the form, highlight *yes* to calculate noise parameters as part of the PSP analysis.

17. Type `50` in the *Maximum Sideband* field.

Setting *Maximum Sideband* to `50` accounts for noise folding up to 5 GHz in frequency.

All noise computations in PSP analysis involve noise folding effects. The *maxsideband* parameter specifies the maximum sideband to include for summing noise contributions either up-converted or down-converted to the output at the frequency of interest.

18. Highlight *Enabled*.

19. Click the *Options* button to display the PSP Options form.

20. Highlight *out* for the *freqaxis* parameter.

Selecting *out* changes the scale used for the output axis so that it runs from 80 MHz to 130 MHz.

The *freqaxis* parameter specifies whether the results should be output versus the absolute value of the input frequency (*absout*), the input frequency (*in*), or the output frequency (*out*).

21. Click *OK* in the Periodic S-parameter Options form.

The completed Periodic S-parameter Analysis section of the Choosing Analyses form appears like the one below.

22. In the Choosing Analyses form, click *OK*.

Information about the `pss` and `PSP` analyses appears in the Analyses section of the Virtuoso Analog Design Environment window.

Running the Simulation

You can now run the simulation. In addition to computing the periodic S-parameters, the periodic noise correlation matrixes are also computed, as are the noise figure and the equivalent noise parameters.

To run the simulation,

1. Choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the Noise Figure

1. To open the Direct Plot form, choose *Results – Direct Plot – Main Form* in the Virtuoso Analog Design Environment window.
2. In the Direct Plot form, highlight *Replace* for *Plot Mode*.
3. Highlight *PSP* for *Analysis*.

   The form changes to display choices appropriate for PSP analysis results.

4. Highlight *NF* for *Function* to display the Noise Figure.

   The form changes to display choices appropriate for noise figure.

The completed Direct Plot form appears like the one below.

5. Follow the prompt at the bottom of the form

```
Press plot button on this form...
```

Click *Plot* to display the Noise Figure.

The plot appears in the waveform window.

6. To determine the noise figure at different frequencies, move the cursor along the noise figure curve and observe the x and y values displayed at the top of the waveform window.

Plotting Periodic S-Parameters

1. If necessary, open the Direct Plot form, choose *Results – Direct Plot – Main Form* in the Virtuoso Analog Design Environment window.

   In the Direct Plot form, do the following:

2. Highlight *Replace* for *Plot Mode*.
3. Highlight *PSP* for *Analysis*.

4. Highlight *SP* for *Function* to display periodic S-parameters.

   The top of the completed Direct Plot form appears like the one below.

5. Highlight *Z-Smith* for *Plot Type* to display periodic S-parameters.

The bottom of the completed Direct Plot form appears like the one below.

6. Follow the prompt at the bottom of the form

```
To plot, press Sij-button on this form...
```

Click *S22* to plot output match.

Recall from basic S-parameter theory that for a 2-port circuit S22 represents the match at port 2. In this example S22 represents the match at the `if` port.

The plot appears in the waveform window.

The waveform window display appears like the one below:

At the center of the Smith chart (at the1), the match is perfect. The match is better as the curve gets closer to the center of the Smith chart. Notice that frequency changes with match. The displayed frequency and position on the Smith chart change as you move the cursor along the curve.

The frequency where the output match is the best is approximately `105.4`MHz, and the reflection coefficient at that frequency is `341.2`m at a phase of `9.017` degrees

Conversion Gain and Power Supply Rejection with PSS and PXF

You can combine a PSS analysis with a periodic small-signal transfer function PXF analysis to determine the conversion gain of the down converter. With changes to the plotting form, you can also calculate the power supply rejection and local oscillator feedthrough.

Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)
4. If necessary, set the design variables `frf` to `920M` and `flo` to `1G`. (Check the Design Variables section to verify the current design variable values.)

Editing the Schematic

You must modify the schematic to set the RF source to a DC source to be sure the PSS analysis is the response of the `ne600p` mixer to only the LO signal.

1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF properties and modify the schematic for this simulation.

2. In the Schematic window, click the *rf* voltage source.

3. In the Edit Object Properties form, choose *dc* as the *Source type*, if necessary, and click *OK*.

4. In the Schematic window, choose *Design – Check and Save*.

Setting Up the PSS and PXF Analyses

1. In the Virtuoso Analog Design Environment window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

Setting Up the PSS Analysis

1. In the Choosing Analyses form, highlight *pss* for the *Analysis*.
2. In the *Fundamental Tones* section, be sure the *Auto Calculate* button is highlighted.

   The *Beat Frequency* is now displayed as `1G`. LO is the only time-varying signal in the circuit, so it becomes the fundamental frequency. The *Beat Frequency* button is highlighted by default.
3. In the *Output Harmonics* cyclic field, choose *Number of Harmonics* and type `0` in the *Number of Harmonics* field.

You must turn off harmonic generation because it is required to set up the subsequent PXF analysis.

4. Highlight *conservative* for the *Accuracy Defaults (errpreset)* setting.
5. Highlight *Enabled*.

6. The PSS Choosing Analyses form looks like this.

7. Click *Apply*.

Setting Up the PXF Analysis

1. At the top of the Choosing Analyses form, click *pxf*.
2. The Choosing Analyses form changes to let you specify data for the PXF analysis.
3. In the *Output Frequency Sweep Range (Hz)* cyclic field, choose *Start – Stop*, and type `1M` in the *Start* field and `300M` in the *Stop* field.

   This range specification leaves space between the plots, but still shows the dramatic gain changes in the first 100 Mhz of each plot.
4. In the *Sweep Type* cyclic field, choose *Linear* and highlight *Number of Steps*. Type `50` for the *Number of Steps*.

   Larger numbers of total points increase the resolution of the plot but also require a longer simulation time.
5. In the *Sidebands* cyclic field, choose *Maximum sideband* and type `3` as the value.

   The top of the PXF Choosing Analyses form looks like this.

6. In the *Output* section, highlight *voltage*.
7. Click the *Select* button for *Positive Output Node*. Then click the appropriate wire in the Schematic window to choose *Pif*.

   `/Pif` appears in the *Positive Output Node* field.
8. Leave the *Negative Output Node* field empty.

   This field defaults to `/gnd!` You can set the *Negative Output Node* to a different value by clicking on the *Negative Output Node Select* button and then choosing the output node in the schematic.
9. Highlight *Enabled*.
10. The bottom of the PXF Choosing Analyses form looks like this.

11. In the Choosing Analyses form, click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

   The output log file appears and displays information about the simulation while it runs.
2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the Conversion Gain

1. Choose *Results – Direct Plot – Main Form* in the Virtuoso Analog Design Environment window.

   The Direct Plot form appears.

2. Highlight *Replace* for *Plot Mode*.
3. Highlight *pxf* for *Analysis*.
4. Highlight *Voltage Gain* for *Function*.
5. Highlight *dB20* for *Modifier*.

   The completed PXF Direct Plot form looks like this.

6. Follow the prompt at the bottom of the form

   `Select Port or Voltage Source on schematic...`

   Click the *Prf* source on the schematic.

The plot appears in the waveform window.

7. To determine the gain at different frequencies, move the cursor along the curve and use the readout at the top of the waveform window.

   For example, the gain is about `-5.46 dB` at `920 MHz`.

Plotting the Power Supply Rejection

In the Direct Plot form, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pxf* for *Analysis*.

3. In the Schematic window, click the *vcc* power supply at the top left of the schematic.

The waveform window appears as follows.

4. To determine the rejection at different frequencies, move the cursor along the curve and use the readout in the lower-left corner of the waveform window.

Calculating the 1 dB Compression Point with Swept PSS

In this example, a swept PSS analysis determines the `1dB` compression point for the `ne600p` mixer configured as a down converter.

Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)
4. If necessary, set the design variables `frf` to `920M` and `flo` to `1G`. (Check the Design Variables section to verify the current design variable values.)

Editing the Schematic

1. In the Schematic window, click the *rf* voltage source.
2. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF properties and modify the schematic for this simulation.
3. Choose *sine* for *Source type*.
4. Type `prf` in the *Amplitude 1 (dBm)* field.
5. Click *OK*.
6. In the Schematic window, choose *Design – Check and Save*.

Setting Up the Swept PSS Analysis

1. In the Virtuoso Analog Design Environment window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, choose *pss* for the *Analysis*.
3. In the *Fundamental Tones* list box, be sure the *Auto Calculate* button is highlighted.

   The *Beat Frequency* is now displayed as `40M`. Now that the RF signal is active again, the fundamental frequency of the circuit goes back to `40 MHz`. The *Beat Frequency* button is highlighted by default.
4. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type `2` in the *Number of harmonics* field.

   Only two harmonics are required to determine the 1 dB compression point.

   The top of the Swept PSS Choosing Analyses form looks like the following.

5. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.
6. Highlight the *Sweep* button.

   The form changes to let you specify data for the sweep.
7. In the *Sweep* cyclic field, choose *Variable*.
8. Click the *Select Design Variable* button.

The Select Design Variable form appears.

9. In the Select Design Variable form, highlight *prf* and click *OK*.

The *Variable Name* `prf` appears in the Choosing Analyses form. Selecting the `prf` variable sweeps RF input.

Choose *Start-Stop* for the *Sweep Range*, and then type `-30` in the *Start* field and `10` in the *Stop* field.

Both the signal source and the sweep are done in dBm. You learn where to set the sweep limits and how many points to include in the sweep with experience.

10. Choose *Linear* for the *Sweep Type*, and specify `10` for the *Number of Steps*.

11. Highlight *Enabled*.

The top of the Swept PSS Choosing Analyses form looks like the following.

12. In the Choosing Analyses form, first click *Apply* then click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the 1 dB Compression Point

The first sideband of the IF signal is the second harmonic of the 40 MHz fundamental. To plot the 1 dB compression point, perform the following:

1. Choose *Results – Direct Plot – Main Form* in the Virtuoso Analog Design Environment window.

   The Direct Plot form appears.

2. Highlight *Replace* for *Plotting Mode*.

3. Highlight *pss* for *Analysis*.

4. Highlight *Compression Point* for *Function*.

5. The top of the Direct Plot form looks like this.

   The form changes to display fields for compression point calculation.

   Type `1` for *Gain Compression (dB)*.

6. Type `-25` for *Input Power Extrapolation Point (dBm)*.

This value specifies the point where the ideal amplification curve intersects the output curve. You must estimate where this point is located. If you do not specify a value, the plot defaults to the minimum variable value.

7. In the cyclic field, choose *Input Referred 1dB Compression*.

8. Follow the prompt at the bottom of the form

   `Select 1st Order Harmonic on this form...`

9. In the *1st Order Harmonic* list box, highlight harmonic *2 (80 MHz)*.

   This is the second harmonic of the 40 MHz fundamental frequency, which is the IF frequency.

   The bottom of the Direct Plot form looks like this.

10. Follow the new prompt at the bottom of the form

    `Select Port on schematic...`

    In the Schematic window, click the *if* port.

The waveform window display looks like this.

The SpectreRF simulation plots the signal curve and an ideal linear slope and automatically calculates a value of `-4.86155 dBm` for the 1 dB compression point.

If you have trouble reading the label attached to the 1 dB point, you can use the cursor to drag it to another location.

Third-Order Intercept Measurement with Swept PSS and PAC

In this example, you combine a swept PSS analysis with a Periodic AC (PAC) analysis to produce data for an IP3 plot.

To calculate an IP3 measurement, you could use settings similar to those used for the 1 dB compression point, but with another tone for the second RF signal. That would be a swept PSS with three tones. The method described here, however, runs more quickly because

- It processes the second RF tone only during the PAC analysis
- It considers only two of the second RF tone sidebands

  The swept PSS analysis for the 1 dB compression point example considered all sidebands for all signals.

Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)
4. If necessary, set the design variables `frf` to `920M` and `flo` to `1G`. (Check the Design Variables section to verify the current design variable values.)

Editing the Schematic

1. In the Schematic window, choose *Edit – Properties – Objects*.

   The Edit Object Properties form appears. You use this form to modify the schematic by changing the list of CDF properties.
2. To modify the schematic for this simulation, click the rf voltage source in the Schematic window.
3. Highlight *Display second sinusoid*, remove any values that are set from previous analyses, then remove the highlight.
4. Choose *sine* for *Source type*.

5. Type `frf` for *Frequency 1*.
6. Type `prf` in the *Amplitude 1 (dBm)* field.
7. Highlight *Display small signal params*.

   The form changes to let you specify small signal parameters.
8. Type `prf` for the *PAC Magnitude (dBm)* value.

   This simulation uses dBm values rather than magnitude.

   The completed Edit Object Properties form appears like the one below.

9. Click *OK*.
10. In the Schematic window, choose *Design – Check and Save*.

Setting Up the PSS and PAC Analyses

➤ In the Virtuoso Analog Design Environment window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

Setting Up the PSS Analysis

1. In the Choosing Analyses form, choose *pss* for the *Analysis*.
2. In the *Fundamental Tones* section, be sure the *Auto Calculate* button is highlighted.

   The value 40M is specified as the *Beat Frequency*. The PAC analysis is responsible for the two tones, and the PSS analysis is now a single signal analysis. Consequently, the fundamental frequency is set to the original 40 MHz. The *Beat Frequency* button is highlighted by default.
3. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type 2 in the field.

The top of the swept PSS Choosing Analyses form looks like this.

4. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.
5. Highlight the *Sweep* button.

   The form changes to let you specify data for the sweep.
6. In the *Sweep* cyclic field, choose *Variable*.
7. Click the *Select Design Variable* button.

   The Select Design Variable form appears.
8. Highlight `prf` in this form and click *OK*.
9. Choose *Start-Stop* for the *Sweep Range* value, and then type `-25` in the *Start field* and `5` in the *Stop* field.
10. Highlight *Linear* for the *Sweep Type*, highlight *Step Size,* and then type `5` in the *Step Size* field.
11. Highlight *Enabled*.

The bottom of the swept PSS Choosing Analyses form looks like this.

12. Click *Apply*.

Setting Up the PAC Analysis

1. At the top of the Choosing Analyses form, highlight *pac*.

   The Choosing Analyses form changes to let you specify data for a *pac* analysis.

2. In the *Specialized Analyses* cyclic field, choose *None*.
3. Type `921M` for the *Freq* value.
4. In the *Sidebands* cyclic field, choose *Array of indices* and in the *Additional indices* field type `-21` and `-25` with a space between them.

Given a fundamental tone of 40 MHz, the LO at 1 GHz, and two RF tones at 920 MHz and 921 MHz, the sidebands of `-25` and `-21` represent respectively the first-order harmonic of the IF output at 79 MHz (921 -25*40 = 79) and the third-order harmonic at 81 MHz (921 - 21*40 = 81).

5. Check the *Enabled*checkbox.

   The PAC Choosing Analyses form looks like this.

6. Click *Apply*.

7. Click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

   Note: This example compares two signals only 1MHz apart. An analysis of two signals so close together would have taken much longer with the previous analysis.

   The output log file appears and displays information about the simulation as it runs.

2. Check the output log file to be sure the simulation completes successfully.

Plotting the IP3 Curve

1. In the Virtuoso Analog Design Environment window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.

2. Select *Replace* for *Plotting Mode*.

3. Highlight *pac* for *Analysis*.

   The form changes to display information for the PAC analysis.

4. Highlight *IPN Curves* for *Function*.

   The form changes again.

5. For *Circuit Input Power*, choose *Variable Sweep ("prf")*.

6. Type `-15` for *Input Power Extrapolation Point (dBm)*.

   This value is the intercept point for the ideal amplification extrapolation. If you do not specify a value, the plot defaults to the minimum variable value.

7. In the cyclic field, choose *Input Referred IP3*.

8. Follow the prompt at the bottom of the form

   ```
   Select 3rd Order Harmonic on this form...
   ```

   Highlight *-21 81M* in the *3rd Order Harmonic* list box.

9. Follow the new prompt at the bottom of the form

   ```
   Select 1st Order Harmonic on this form...
   ```

Highlight *-25 79M* in the *1st Order Harmonic* list box.

The completed form looks like this.

Notice that *freqaxis = absout* near the bottom of the form.

Follow the new prompt at the bottom of the form

```
Select Port on schematic...
```

In the Schematic window, click the *if* port.

The waveform window display appears as shown below. (You can click and drag to move the two labels with arrows, *ep* and *Input Referred IP3*, to improve visibility.)

The general equation used to compute the input-referred third-order intercept point is

$$IIP3 = V_{rf} + \frac{(V_{if(-25)} - V_{if(-21)})}{2}$$

where V_{rf} is the value for the input power extrapolation point (dBm).

Intermodulation Distortion Measurement with QPSS

The QPSS analysis lets you consider the effects of a few harmonics in intermodulation distortion calculations. This example describes how to run a QPSS analysis with the ne600p mixer circuit.

With the QPSS analysis, you can compute the distortion of moderately sized signals, as opposed to the small signals you investigated with the swept PSS analysis with PAC approach where you assumed there were no significant distortion effects from the small signal harmonics. With QPSS, you specify a large, or clock, signal for the fundamental and one or more moderately sized signals whose distortion effects you want to measure.

Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)
4. If necessary, set the design variables `frf` to `900M` and `flo` to `1G`. (Check the Design Variables section to verify the current design variable values.)

Editing the Schematic

1. In the Schematic window, choose the *rf* input port and then choose *Edit – Properties – Objects*.

   The Edit Object Properties form for the *rf* port appears.
2. Choose *Display small signal params* and remove any remaining values from any previous analysis.
3. Choose *sine* for *Source type*.
4. Choose *Display second sinusoid* and specify the following when the form changes to display new fields:
   a. Type `fund2`, or any name you choose, in the *Frequency name 2* field.
   b. Type `920M` for the *Frequency 2* value.
   c. Type `prf-10` for the *Amplitude 2 (dBm)* value.

The completed form looks like this.

CDF Parameter	Value	Display
Resistance	50 Ohms	off
Reactance		off
Port number	1	off
DC voltage		off
Source type	sine	off
Frequency name 1	frf	off
Frequency 1	frf Hz	off
Amplitude 1 (Vpk)		off
Amplitude 1 (dBm)	prf	off
Phase for Sinusoid 1		off
Sine DC level		off
Delay time		off
Display second sinusoid	✓	off
Frequency name 2	fund2	off
Frequency 2	920M Hz	off
Amplitude 2 (Vpk)		off
Amplitude 2 (dBm)	prf-10	off
Phase for Sinusoid 2		off

5. In the Edit Object Properties form, click *OK*.
6. In the Schematic window, choose *Design – Check and Save*.

Setting Up the QPSS Analysis

1. In the Virtuoso Analog Design Environment window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. At the top of the Choosing Analyses form, highlight *qpss*.

   The form changes to let you specify information for the QPSS analysis. There are three tones in the *Fundamental Tones* list box, two that were present in the original specifications and one that you added.

   To update the fundamental tones, perform the following steps:

3. In the *Fundamental Tones* list box, highlight the tone named `frf`.

   The `frf` tone and its associated values appear in the fields below the list box.

4. If necessary, choose *Moderate* from the *Signal* cyclic field.

5. In the *Harms* data entry field, highlight the current value and then type `2` to specify the range of harmonics for the `frf` tone.

   An increase in the harmonic range for a moderate tone, such as `frf`, directly increases the simulation run time.

6. In the list box, highlight the tone named `fund2` (or whatever name you gave it).

   The new values for `frf` display in the list box. The `fund2` tone and its associated values appear in the fields below the list box.

7. If necessary, choose *Moderate* from the *Signal* cyclic field.

8. In the *Harms* data entry field, highlight the current value and then type `2` to specify the range of harmonics for the `fund2` tone.

9. In the Fundamental Tones list box, highlight the tone named `flo`.

   The new values for `fund2` display in the list box. The `flo` tone and its associated values appear in the fields below the list box.

10. Choose *Large* from the *Signal* cyclic field. In addition, highlight the current value in the *Harms* field and then type `1` to specify the range of harmonics for the `flo` tone. Click *Clear/Add*.

    The new values for `flo` display in the list box. the harmonic range of 1 gives 3 harmonics (-1, 0,1) for the large tone.

    By choosing *Large* as the *Signal* value, you specify `flo` to be the *Large* or clock signal. Each QPSS analysis must have one tone set to be the *Large* signal. When you choose a signal for the large signal, choose a tone that is larger, less sinusoidal, or more nonlinear than the other tones.

You must choose at least one harmonic for each signal that you want to include in a QPSS analysis. A signal with an harmonic range (*Harms*) value of 0 is ignored by the simulation. In general,

- The harmonic range (*Harms)* value for the *Large* tone should generally be at least 5 which guarantees11 harmonics. In some cases, for example in a down converting situation such as this one, *Harms* can be as low as 1.
- The *Harms* value for *Moderate* tones should be approximately 2 or 3. For *Moderate* tones, increasing the *Harms* value increases the simulation run time.

  Setting the *Harms* value to 2, yields up to the 3rd order intermodulation terms; setting the *Harms* value to 3, yields up to the 5th intermodulation terms. For higher order intermodulation terms, increase the *Harms* value accordingly. However, you should avoid setting the *Harms* value unnecessarily high because it increases the simulation time.

When, as in this example, you specify *Harms* values of 1 for the *Large* signal and 2 for the *Moderate* signals, you get `maxharms = [1,2,2]` which gives you 3 harmonics (-1, 0, 1) for the *Large* tone and 5 harmonics (-2, -1, 0, 1, 2) for each *Moderate* tone. As a result, you get noise from 3 x 5 x 5 = 75 frequency sidebands.

11. Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.

12. Be sure the *Enabled* checkbox is checked.

The completed form appears like the one below.

13. Click *Apply*.

14. Click *OK*.

Selecting Simulation Outputs

1. In the Virtuoso Analog Design Environment window, choose *Outputs – To Be Saved – Select on Schematic*.

   If you want to plot current or power at the end of the simulation, you must explicitly save the currents necessary for the calculations. The most economical way to do this, in terms of simulation time, is to choose specific currents on the schematic.

2. In the Schematic window, click the appropriate terminals to choose *rf* and *rif*.

   The terminals are circled in the Schematic window after you choose them.

The selected terminals also display in the Virtuoso Analog Design Environment window.

3. Click the *Esc* key when you are finished selecting terminals.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

   The output log file appears and displays information about the simulation as it runs.

2. Check the CIW for a message that says the simulation completed successfully.

Plotting the Voltage and Power

➤ In the Virtuoso Analog Design Environment window, choose *Results – Direct Plot – Main Form*.

The Direct Plot form appears.

Plotting Voltage

To plot voltage do the following in the Direct Plot form:

1. Select *Replace* for *Plotting Mode*.
2. Highlight *qpss* for *Analysis*.
3. Highlight *Voltage* for *Function*.
4. Ensure that the *Select* cyclic field displays *Net*.
5. Highlight *peak* for *Signal Level*.
6. Highlight *dB20* for *Modifier*.

The completed form looks like this.

7. Follow the prompt at the bottom of the form

```
Select Net on schematic...
```

Click the net connecting to the *rf* terminal.

The voltage plot for the *rf* terminal displays in the waveform window.

Because this is a down-converting mixer, the cluster of frequencies from 20 M to 120 M near the Y axis of the plot are of interest. To find a frequency and its associated voltage, place the cursor at the tip of a line. The X and Y values that represent the frequency and the voltage, respectively, appear at the lower left corner of the window. If you place the

cursor at the points shown above, you display the following voltage and frequency values:

Difference Translation	Frequency	Voltage
920M - 900M	20M	-100.2dB20(V)
1G - 920M	80M	-132.5db20(V)
1G - 900M	100M	-122.6db20(V)

Plotting Power

To plot the power, do the following in the Direct Plot form:

1. Select *Replace* for *Plotting Mode*.
2. Highlight *Power* for *Function*.
3. Ensure that the *Select* cyclic field displays *Terminal*.
4. Highlight *dBm* for *Modifier*.
5. Follow the message at the bottom of the form.

   `Select Instance Terminal on schematic...`

   In the Schematic window, click the *rf* terminal.

The plot for the power appears in the waveform window.

6. If you want to see the voltage and power for the *rif* terminal, repeat the steps to create the plots for the *rif* terminal, but choose the *rif* terminal in the schematic.

   Note: If you want to run other simulations in this chapter, reset the design variable *frf* to `920 M`.

Noise Figure with QPSS and QPnoise

You can combine a QPSS analysis with a small-signal QPnoise analysis to determine the degradation of the noise figure for the ne600p mixer circuit caused by the two interfering signals present at the input of the mixer. The noise figure for the ne600p mixer without the interferer signals is determined in "Noise Figure Measurement with PSS and Pnoise" on

page 1850. The QPSS and QPnoise analyses described in this example include the interferer signals which allow more noise foldings to occur into the output IF band.

The Quasi-Periodic Noise, or QPnoise analysis, is similar to the conventional noise analysis, except that it includes frequency conversion and intermodulation effects. Hence is it useful for predicting the noise behavior of mixers, switched-capacitor filters, and other periodically or quasi-periodically driven circuits.

Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Virtuoso Analog Design Environment window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Analyses section to verify whether or not an analysis is enabled.)
4. If necessary, set the design variables `frf` to `900M` and `flo` to `1G`. (Check the Design Variables section to verify the current design variable values.)

Editing the Schematic

1. In the Schematic window, click the *rf* input port and then choose *Edit – Properties – Objects*.

   The Edit Object Properties form for the *rf* input port appears. You use this form to change the list of CDF properties and modify the schematic for this simulation.

In the Edit Object Properties form, do the following:

1. Choose *Display small signal params* and remove any remaining values from the previous analysis.
2. Because the `rf` signal is a moderate signal, choose *sine* for *Source type*.
3. Choose *Display second sinusoid* and specify the following when the form changes to display new fields:
   a. Type `fund2`, or any name you choose, in the *Frequency name 2* field.
   b. Type `920M` for the *Frequency 2* value.
   c. Type `prf - 10` for the *Amplitude 2 (dBm)* value.

The completed form looks like this.

CDF Parameter	Value	Display
Resistance	50 Ohms	off
Reactance		off
Port number	1	off
DC voltage		off
Source type	sine	off
Frequency name 1	frf	off
Frequency 1	frf Hz	off
Amplitude 1 (Vpk)		off
Amplitude 1 (dBm)	prf	off
Phase for Sinusoid 1		off
Sine DC level		off
Delay time		off
Display second sinusoid	✔	off
Frequency name 2	fund2	off
Frequency 2	920M Hz	off
Amplitude 2 (Vpk)		off
Amplitude 2 (dBm)	prf-10	off
Phase for Sinusoid 2		off

4. In the Edit Object Properties form, click *OK*.
5. In the Schematic window, choose *Design – Check and Save*.

Setting Up the QPSS and QPnoise Analyses

In the following sections you set up two analyses, first *qpss* then *qpnoise*.

➤ In the Virtuoso Analog Design Environment window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

Setting Up the QPSS Analysis

1. At the top of the Choosing Analyses form, highlight *qpss*.

   The form changes to let you specify information for the QPSS analysis.

2. In the *Fundamental Tones* list box, highlight the tone named `frf`.

   The `frf` tone and its associated values appear in the fields below the list box.

3. If necessary, choose *Moderate* from the *Signal* cyclic field.

4. In the *Harms* data entry field, highlight the current value and then type 2 to specify the range of harmonics for the `frf` tone.

   An increase in the harmonic range for a moderate tone directly increases the simulation run time.

5. In the list box, highlight the tone named `fund2`.

   The new values for `fund2` display in the list box. The `fund2` tone and its associated values appear in the fields below the list box.

6. If necessary, choose *Moderate* from the *Signal* cyclic field.

7. In the *Harms* data entry field, highlight the current value and then type 2 to specify the range of harmonics for the `fund2` tone.

8. In the list box, highlight the tone named `flo`.

   The new values for `fund2` display in the list box. The `flo` tone and its associated values appear in the fields below the list box.

9. Choose *Large* from the *Signal* cyclic field. In addition, highlight the current value in the *Harms* field and type `5` to specify the range of harmonics for the `flo` tone. Click *Clear/Add*.

   The new values for `flo` display in the list box. The harmonic range of 5 gives 11 harmonics (-5, ... ,0, ... +5) for the large tone. In this case where a QPnoise analysis is following the QPSS analysis, the *Harms* value must be sufficiently large to guarantee an accurate noise analysis. Generally, 5 is a sufficiently large *Harms* value. Entering a large *Harms* value for the large tone does not affect the simulation run time.

   By choosing *Large* as the *Signal* value, you specify `flo` to be the large or clock signal. Each QPSS analysis must have one tone set to be the large signal. When you choose a

large signal, select a tone that is larger, less sinusoidal, or more nonlinear than the other tones.

You must choose at least one harmonic for each signal that you want to include in a QPSS analysis. A signal with a harmonic range value of 0 is ignored by the simulation. In general,

- The harmonic range (*Harms)* value for the *Large* tone should be at least 5, guaranteeing 11 harmonics.
- The *Harms* value for *Moderate* tones should be approximately 2 or 3. For *Moderate* tones, increasing the *Harms* value increases the simulation run time.

  When, as in this example, you specify *Harms* values of 5 for the *Large* signal and 2 for the *Moderate* signals, you get `maxharms = [5,2]` which gives you 11 harmonics for the Large tone and 5 harmonics (-2, -1, 0, 1, 2) for the Moderate tone. As a result, you get noise from 11 x 5 = 55 frequency sidebands. SpectreRF considers all these combinations when performing QPnoise calculations.

10. Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.

11. Be sure the *Enabled* checkbox is checked.

The completed form appears like the one below.

12. Click *Apply*.

Setting Up the QPnoise Analysis

1. In the Analysis section at the top of the Choosing Analyses form, highlight *qpnoise*.

   The Choosing Analyses form changes to let you specify data for a Quasi-Periodic Noise Analysis.

2. In the *Output Frequency Sweep Range (Hz)* cyclic field, choose *Start-Stop*.

3. Type `50M` in the *Start* field and `150M` in the *Stop* field.

   This frequency range covers the frequencies of interest.

4. In the *Sweep Type* cyclic field, choose *Linear* for the sweep type and highlight *Number of Steps*. Type `20` in the *Number of Steps* field.

5. In the *Sidebands* cyclic field, choose *Maximum clock order* and type `5`.

   Maximum clock order is related to large signal harmonics range. A *Maximum clock order* of 5 insures that sufficient large-signal harmonics are processed. You can increase this value.

6. In the *Output* cyclic field, choose *probe* for the *Output* value.

7. Click the *Select* button for *Output Probe Instance*. Then click the appropriate component in the Schematic window to choose *Pif*.

   `/rif` appears in the *Output Probe Instance* field.

8. In the *Input Source* cyclic field, choose *port*.

9. Click the *Select* button for *Input Port Source*. Then click the appropriate component in the Schematic window to choose *rf*.

   `/rf` appears in the *Input Port Source* field.

10. Type `1 0 0` in the *Reference side-band* field. One entry for each tone separated by spaces.

    These numbers define a vector whose indices have a one-to-one correspondence with the fundamental tones. For this example, the correspondence is illustrated in the following table.

Ref Sideband Vector	1	0	0
Fundamental Tones	LO	FUND2	RF
Frequencies	1 GHz	920 MHz	900 MHz

The reference sideband vector `1 0 0` mixes with the LO tone only to get the output.

The completed *Quasi Periodic Noise Analysis* section of the Choosing Analyses form appears like the one below.

11. In the Choosing Analyses form, click *OK*.

Selecting Simulation Outputs

1. In the Virtuoso Analog Design Environment window, choose *Outputs – To Be Saved – Select on Schematic*.

If you want to plot current or power at the end of the simulation, you must explicitly save the currents necessary for the calculations. The most economical way to do this, in terms of simulation time, is to select the specific currents to save on the schematic.

2. In the Schematic window, click the appropriate terminals to choose *rf* and *rif*.

The terminals are circled in the Schematic window after you choose them.

The selected terminals also appear in the Virtuoso Analog Design Environment window.

3. Click the *Esc* key when you are done selecting terminals.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Virtuoso Analog Design Environment window.

   The output log file appears and displays information about the simulation as it runs.

2. Check the CIW for a message that says the simulation completed successfully.

Plotting the Noise Figure

1. In the Virtuoso Analog Design Environment window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form window appears.

2. Highlight *Replace* for *Plot Mode*.
3. Highlight *qpnoise* for *Analysis*.
4. Highlight *Noise Figure* for *Function*.

The completed Direct Plot form for Qpnoise appears like the one below.

5. Follow the prompt at the bottom of the form

   `Press plot button on this form...`

   Click *Plot* in the Direct Plot form.

   The plot appears in the waveform window.

The waveform window appears like the one below:

6. To determine the noise figure at different frequencies, move the cursor along the noise figure curve in the waveform window. In the above plot, the noise figure is about `73.24dB` at `80 MHz`.

Plotting the Output Noise

In the Direct Plot form, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *qpnoise* for *Analysis*.
3. Highlight *Output Noise* for *Function*.

4. Highlight V/sqrt(Hz) for Signal Level.
5. Highlight *dB20* for *Modifier*.

   The completed Direct Plot form appears like the one below.

6. Follow the prompt at the bottom of the form

   `Press plot button on this form...`

   Click *Plot* in the Direct Plot form.

   The plot appears in the waveform window.

The waveform window appears like the one below:

7. To determine the output noise at different frequencies, move the cursor along the output noise curve in the waveform window. In the above plot, the output noise is about `-119.2 dB` at `80 MHz`.

Note: Before you run most other simulation examples in this chapter, be sure to reset the *Design Variable* `frf` to `920M`.

IP3 and Compression Distortion Summary Measurements for a Mixer

Mixer distortion limits the sensitivity of a receiver if there is a large interfering signal present within the bandwidth of the RF input filter (a characteristic known as *selectivity*).

Consider measuring Rapid IP3 and Compression Distortion Summary for the *NE600* using perturbation-based PAC analysis. The *NE600* mixer schematic from the *rfExamples* library is shown in Figure A-6.

Figure A-6 The NE600 Mixer Schematic

When choosing the RF input signals *frf1* and *frf2*, you can perform a PAC analysis to determine the bandwidth of the circuit. Then you can place the RF input signals in the middle of the bandwidth. The RF input signals *frf1* and *frf2* should be close enough in frequency so that their intermodulation terms are also within the bandwidth.

Suppose the RF input signal is at *frf1* = 900 MHz, the LO signal is at 1 GHz, and the IFsignal is at 100 MHz. Another RF signal is at *frf2* = 925 M, so one of the 3rd intermodulation products is at 2*frf1* - *frf2* = 875 M and the corresponding IF signal is at 125 M. Before you perform the Rapid IP3 measurement, you must run a PSS analysis. For small-signal analysis,

it is sufficient to treat the RF input as a small-signal (for example, by setting the *Source Type* to *dc*).

Measuring Rapid IP3

The Rapid IP3 mixer measurement is calculated using SpectreRF in the Analog Design Environment (ADE).

Stimuli:

Apply a large sinusoidal signal at the LO port (*PORT2)*	*flo*
Apply a moderate *dc* source at the RF port (*PORT1*)	*frf1*

Parameters:

Set the parameters

flo	`1G`
prf	`-30`
frf1	`900M`
frf2	`925M`

Simulation/Analyses:

1. Set up a PSS analysis.

   Beat frequency: *flo*

   Output harmonics: *number of harmonics 30*.

   Note: There are two simulation engines: *Shooting* and *Harmonic Balance*. If you choose *Harmonic Balance*, the harmonics set in the PSS analysis affect the accuracy of the IP3 result.

2. Set up a PAC analysis.

   Specialized Analyses: *Rapid IP3*

Source Type: *port*

Input Sources 1: */PORT1* **freq:** *frf1*

Input Sources 2: */PORT1* **freq:** *frf2*

Input Power (dBm): *prf*

Frequency of IM Output Signal: *flo-(2*frf1-frf2)*

Frequency of Linear Output Signal: *flo-frf1*

Maximum Non-Linear Harmonics: *4*

Output: *Voltage*

Out+: /*pif*

Out-: /gnd!.

Note: If output is current in a port, you must use the `save` statement to indicate that the port current needs to be computed. Otherwise, Spectre won't calculate it.

3. Run the PSS and the PAC simulations:
4. Display the results with the Direct Plot form.

   To plot Rapid IP3, click *PAC* analysis. Select *Rapid IP3*. IP3 is plotted as in Figure A-7.

Figure A-7 Rapid IP3 Measurement

Measuring Compression Distortion Summary

The Compression Distortion Summary mixer measurement is calculated using SpectreRF in the Analog Design Environment (ADE).

Stimuli:

Apply a large sinusoidal signal at the LO port (*PORT2)*. Use the RF port (*PORT1*) for a moderate DC source, set the parameter for *PAC magnitude (dBm)* in the *Small Signal Parameters to -30 dBm*. This value will be used as the input power in the following small signal analysis.

Parameters:

Set the parameters

flo `1G`

frf `900M`

Simulation/Analyses:

1. Set up a PSS analysis.

   Beat frequency: *flo*

   Output harmonics: *number of harmonics 30*.

   Note: There are two simulation engines: *Shooting* and *Harmonic Balance*. If you choose *Harmonic Balance*, the harmonics set in PSS analysis will affect the accuracy of the Compression Distortion Summary result.

2. Set up a PAC analysis.

   Start: *frf*

   Specialized Analyses: *Compression Distortion Summary*

   Contributor Instances:

   Frequency of Linear Output Signal: *flo-frf*

   Maximum Non-Linear Harmonics: *4*

   Output: *Voltage*

   Out+: */pif*

   Out-: */gnd!*

   Note: If output is current in a port, you must use the `save` statement to indicate that the port current needs to be computed. Otherwise, Spectre won't calculate it.

   When no devices are selected, all the nonlinear devices will be calculated. Because PAC needs to be run for each device specified, computation could be time-consuming if the device list is large. You should select only the most important devices to the output. We do not recommended running the distortion summary for all nonlinear devices in the circuit.

3. Run the PSS and the PAC simulations.

4. Display/Data Analysis.

   You can find the information in the *PAC Distortion Summary* which you can access by choosing *Results – Print.* The result is shown in Figure A-8.

Figure A-8 PAC Compression Distortion Summary

This summary lists the various distortion contributors and how much distortion each contributes to the output.

If no devices are selected, all the nonlinear devices will be calculated. You can quickly identify top distortion contributors, and you can look up the distortion contribution for a particular device.

Rapid IP2 and IM2 Distortion Measurements for a Gilbert Direct Conversion Mixer

Figure A-9 Scheme of a Direct Conversion Receiver

Direct conversion is an alternative architecture to the highly integrated and low-power super-heterodyne receiver. Its fundamental advantage is that the received signal is amplified and filtered at baseband rather than at some high intermediate frequency. A typical front-end frame for a direct conversion receiver is shown in Figure A-9.

The direct conversion mixer (outlined by the dashed rectangle in Figure A-9) is the key component in this receiver. If the local oscillator frequency (LO) equals to (zero IF) or approaches (low IF) the carrier frequency (RF), the mixer translates signals from the RF band to baseband, generally resulting in two mutually exclusive groups of signals.

In an ideal situation, the mixer output would be an exact replica of the input signal. In reality, mixer output is distorted due to non-linearity in the mixer, caused by BJT, MOS etc. Linearity is important to the receiver's performance. The most important nonlinearities of a direct conversion mixer are IP2 and IM2, which you can efficiently measure using SpectreRF perturbation technology.

The mixer measurements described here are calculated using SpectreRF in ADE. The design investigated is a Gilbert double-balanced mixer shown in Figure A-10.

Figure A-10 Gilbert Double-Balanced Mixer

The core of the mixer is the Gilbert cell, constructed from NPN Q2, Q3, Q4 and Q5. The NPN Q0 and Q1 are connected as common-base, and act to import RF signals.

The input RF signal enters a Balun transmission line block (implemented by AHDL). Two output difference signals feed into the emitters of Q0 and Q1 respectively.

The PNP Q6 and Q7 act as current mirrors supplying the Gilbert cell.

The circuit runs with a local oscillator at *flo=1.9 G*. Input RF signal at *frf1=1.9012 G* and the corresponding IF at *frf1-flo=1.2 M*. Another RF signal is at *frf2=1.901 G*. One of the IM2 is at *frf1-frf2=200 K*.

Measuring Rapid IP2

The rapid IP2 mixer measurement is described in the following sections. It is calculated using SpectreRF in ADE.

Stimuli:

The circuit has two ports providing large sinusoidal LO signals. They have same frequency *flo=1.9G* and amplitude *ampl=316.2m* with different phase. One is *0°* and the other is *180°*.

Parameters:

Set the parameters

flo	`1.9G`
frf1	`1.9012G`
frf2	`1.901G`
prf	`-30`

Simulation/Analyses:

1. Set up a PSS analysis.

   Beat frequency: *flo*

   Output harmonics: *number of harmonics 20*.

   Note: There are two simulation engines: *Shooting* and *Harmonic Balance*. If you choose *Harmonic Balance*, the harmonics set in PSS analysis will affect the accuracy of the small signal analyses.

2. Set up a PAC analysis.

   Specialized Analyses: *Rapid IP2*

   Source Type: *port*

   Input Sources 1: */rf* **freq:** *frf1*

   Input Sources 2: */rf* **freq:** *frf2*

Input Power (dBm): *prf*

Frequency of IM Output Signal: *frf1-frf2*

Frequency of Linear Output Signal: *frf1-flo*

Maximum Non-Linear Harmonics: *4*

Output: *Voltage*

Out+: *out2*

Out-: *out1*.

Note: If output is current in a port, you must use the `save` statement to indicate that the port current needs to be computed. Otherwise, Spectre does not calculate it.

3. Run the PSS and PAC simulations.
4. Display/Data Analyses.

   Display results using *Results-Direct Plot*.

   To plot Rapid IP2, click *PAC* analysis. Select *Rapid IP2*. The IP2 for the mixer is plotted in Figure A-11.

Figure A-11 PAC Rapid IP2 Plot for the Mixer

Measuring IM2 Distortion Summary

The IM2 distortion summary mixer measurement is described in the following section. It is calculated using SpectreRF in ADE.

Stimuli:

The circuit has two ports providing large sinusoidal LO signals. They have the same frequency *flo=1.9G* and amplitude *ampl=316.2m* with different phase. One is *0°* and the other is *180°*.

Parameters:

Set the parameters

flo	`1.9G`
vlo	`316.2m`
frf1	`1.901G`
frf2	*frf1*`+200K`
prf	`-30`

Simulation/Analyses:

1. Set up a PSS analysis.

   Beat frequency: *flo*

   Output harmonics: *number of harmonics 20*.

   Note: There are two simulation engines: *Shooting* and *Harmonic Balance*. If you choose *Harmonic Balance*, the harmonics set in PSS analysis will affect the accuracy of the small signal analyses.

2. Set up a PAC analysis.

   Specialized Analyses: *IM2 Distortion Summary*

   Source Type: *port*

   Input Sources 1: */rf* **freq:** *frf1*

   Input Sources 2: */rf* **freq:** *frf2*

   Input Power (dBm): *prf*

   Frequency of IM Output Signal: *frf1-frf2*

   Maximum Non-Linear Harmonics: *4*

   Output: *Voltage*

   Out+: *out2*

   Out-: *out1*.

   Note: If output is current in a port, you must use the `save` statement to indicate that the

port current needs to be computed. Otherwise, Spectre does not calculate it.

3. Run the PSS and the PAC simulations.
4. Display/Data Analysis

   Choose *Results-Print* to display the *IM2 Distortion Summary* as shown in Figure A-12.

Figure A-12 PAC IM2 Distortion Summary for the Mixer

Rapid IP3 and Compression Distortion Summary Measurements for a Power Amplifier

One of the main concerns in power amplifier (PA) design is the nonlinearity effect. The nonlinearity of a PA can degrade the quality of the Tx signal and interfere with adjacent channels. Typically, IP3 and adjacent channel power ratio (ACPR) are used to characterize PA nonlinearity.

In this example,

- Rapid IP3 is used to measure the PA's IP3.

- The contribution of every nonlinear device to the PA's total nonlinearity is evaluated using the Compression Distortion Summary.

Compared with measurements made for the mixer, IP3 measurement for a PA or LNA can use AC-based Rapid IP3 and Compression Distortion Summary because their operating point is time-independent. In this case, AC-based measurement is more efficient than PAC-based Rapid IP3 and Compression Distortion Summary.

The PA used in the following measurements is shown in Figure A-13. This PA is an edited version of the *EF_example* schematic in the *rfExamples* library. It consists of three blocks: input match, the input amplifier (I4) and the output amplifier (I2).

Figure A-13 The Power Amplifier Schematic (EF_example)

Subcircuit I4 is shown in Figure A-14. This input amplifier subcircuit is called *EF_PA_istg* and it is in the *rfExamples* library.

Figure A-14 Schematic for Subcircuit I4 (the Input Amplifier Subcircuit)

Subcircuit 12 is shown in Figure A-15. This output amplifier subcircuit is called *EF_PA_ostg* and it is in the *rfExamples* library.

Figure A-15 Schematic for Subcircuit 12 (the Output Amplifier Subcircuit)

The Rapid IP3 Measurement

The Rapid IP3 PA measurement is calculated using SpectreRF in ADE.

Stimuli:

For the RF port (PORT3), set *dc* as the *Source Type*. Set the parameter for *AC magnitude* to `7m` Volts in the *small signal params* section. This value will be used as the input magnitude in the AC small signal analysis.

Parameters:

Set the parameters

frf1	*1G*
frf2	*frf1 + 0.002G*
prf	*-30*

Simulation/Analyses:

1. Set up an AC analysis.

   Specialized Analyses: *Rapid IP3*

   Source Type: *port*

   Input Sources 1: */PORT3* **freq:** *frf1*

   Input Sources 2: */PORT3* **freq:** *frf2*

   Input Power (dBm): *prf*

   Frequency of IM Output Signal: *2frf1-frf2*

   Frequency of Linear Output Signal: *frf2*

   Maximum Non-Linear Harmonics: *4*

   Output: *Voltage*

   Out+: */RFOUT*

   Out-: */gnd!*

   Note: If output is current in a port, you must use the `save` statement to indicate that the port current needs to be computed. Otherwise, Spectre won't calculate it.

2. Run the AC simulation.

3. Display/Data Analyses.

   To display the Rapid IP3 results,

   a. Open the Direct Plot form.

   b. Click *AC* analysis.

c. Select *Rapid IP3*.

d. Click *plot*.

Rapid IP3 for the PA is plotted as shown in Figure A-16.

Figure A-16 AC Rapid IP3 Plot for the PA

Measuring Compression Distortion Summary

The Compression Distortion Summary PA measurement is calculated using SpectreRF in ADE.

Stimuli:

For the RF port (PORT3), set *dc* as the *Source Type*. Set the parameter for *AC magnitude* to `7m` Volts in the *small signal params* section. This value will be used as the input magnitude in the AC small signal analysis.

Parameters:

Set the parameters:

frf *1.002G*

prf *-30*

Simulation/Analyses:

1. Set up an AC analysis.

   Start: *frf*

   Specialized Analyses: *Compression Distortion Summary*

   Contributor Instances:

   Frequency of Linear Output Signal: *frf*

   Maximum Non-Linear Harmonics: *4*

   Output: *Voltage*

   Out+: */RFOUT*

   Out-: */gnd!*

   Note: If output is current in a port, user must use the `save` statement to indicate that the port current needs to be computed. Otherwise, Spectre won't calculate it.

Display/Data Analysis:

➤ Display Rapid IP3 results by selecting *Results-Print-AC Distortion Summary*.

The result is shown in Figure A-17. It can be seen that the distortion from subcircuit I2 devices is more significant than that from the subcircuit I4 devices. This may be because subcircuit I2 is the second stage amplifier in the PA and it performs the main power amplifier function. It also should be noticed that the total distortion is lessened for the interaction between devices.

Figure A-17 AC Compression Distortion Summary for the PA

Transmitters

This chapter tells you how to use several Virtuoso® Spectre® circuit simulator RF analysis (SpectreRF) transmitter design features. It emphasizes

- Setting Up an Envelope analysis (Envlp)

  The Envelope analysis example explains

  - How to interpret Envlp analysis results
  - How to visually detect distortion quickly

- Measuring ACPR and PSD. PSD (Transmitted Power Spectral Density) is characterized by the ACPR number (Adjacent Channel Power Ratio).

  The ACPR and PSD example explains

  - How to use the ACPR wizard.
  - How to estimate PSD.

- Measuring Load Pull Contours and Reflection Coefficients

  The Load-Pull example explains

  - How to select an optimal power amplifier load
  - How to determine whether the input matching network needs to be re-designed for the optimal load

- USing S-parameter Input Files

  The S-parameter example explains

  - How to generate tabulated S-parameters
  - How to include tabulated S-parameters as input in a SpectreRF analysis

- Measuring AM and PM Conversion with Modulated PAC, AC and PXF Analyses

  The AM/PM conversion example explains how to determine the amplitude and phase modulation effects in RF circuits using the Modulated PAC and PXF analyses.

  - How to create the EX_AMP example.
  - How to set up the modulated PAC and PXF analyses

- Measuring Jitter with PSS and Pnoise Analyses

This example explains how to measure different jitter measurements using the Pnoise Jitter measurement.

- How to set up or create the EX_AMP circuit.
- How to set up the Pnoise jitter analyses

Envelope Analysis

This example tells you how to set up and run the Envelope analysis then explains why the time results look somewhat unusual.

Before you start, perform the setup procedures described in Chapter 3.

Opening the EF_example Circuit in the Schematic Window

1. In the CIW, choose *File – Open*.

   The Open File form appears.

2. In the Open File form, do the following:

   a. Choose *my_rfExamples* for *Library Name,* the editable copy of *rfExamples*.

      Select the editable copy of the *rfExamples* library you created following the instructions in Chapter 3.

   b. In the *Cell Name* list box, highlight *EF_example*.

   c. Choose *schematic* for *View Name*.

   d. Highlight *edit* for *Mode*.

After these steps the Open File form looks like this.

e. Click *OK*.

The Schematic window appears with the *EF_example* schematic. This is a simple direct-conversion transmitter with ideal I/Q modulators.

Opening the Simulation Window

1. In the Schematic window, choose *Tools – Analog Environment*.

   The ADE window opens.

Note: You can also choose *Tools – Analog Environment – Simulation* in the CIW to open the ADE window without opening the design. You can open the design later by choosing *Setup – Design* in the ADE window and choosing *EF_example* in the Choosing Design form.

Setting Up the Model Libraries

1. In the ADE window, choose *Setup - Model Libraries.*

   The Model Library Setup form appears.

2. In the *Model Library File* field, type the full path to the model file including the file name, where `CDSHOME` is the installation directory for the Cadence software.

   `<CDSHOME>/tools/dfII/samples/artist/models/spectre/rfModels.scs.`

3. Click *Add*.

The Model Library Setup form looks like the following.

4. Click *OK*.

Editing PORT0 and PORT1 in the EF_example Schematic

The *EF_example* circuit uses the programmable voltage source, *port*. The RF voltage source is based on the *port* sample component. You can easily change the behavior of this programmable component.

In the example, you edit `PORT0` and `PORT1` on the left side of the schematic.

1. In the Schematic window, select `PORT0`. (The port in the top, left corner of the *EF_example* schematic.)

2. In the Schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears and changes to display information for `PORT0`. You use this form to change the list of CDF (component description format) properties for the `PORT0` and modify the schematic for this simulation.

3. In the Edit Object Properties form, edit the *PWL filename* of `PORT0`. Type the following path, where *`CDSHOME`* is the installation directory for the Cadence software. Then click *Apply*.

```
<CDSHOME>/tools/dfII/samples/artist/rfLib/cdma_2ms_idata.pwl
```

The section of the Edit Object Properties form that includes the *PWL file name* field looks like the following.

4. In the Schematic window, select `PORT1`. (The port in the bottom, left corner of the *EF_example* schematic.)

The Edit Object Properties form changes to display information for `PORT1`.

5. In the Edit Object Properties form, edit the *PWL filename* of `PORT1`. Type the following path, where *`CDSHOME`* is the installation directory for the Cadence software. Then click *Apply*.

   ```
   <CDSHOME>/tools/dfII/samples/artist/rfLib/cdma_2ms_qdata.pwl
   ```

6. Click *OK*.

7. Choose *Design - Check and Save* in the Schematic window.

Setting Up an Envelope Analysis

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, highlight *envlp*.

   The Choosing Analyses form changes to let you specify values for the Envelope analysis.

3. In the Choosing Analyses form, do the following.

   a. Click *Clock Name*.

   b. Click *Select Clock Name.*

   The Select Clock Name form appears.

   c. In the Select Clock Name form, click `fff` then click *OK.*

   In the *Clock Name* field, `fff` appears.

   d. Type `300u` for *Stop Time*.

   e. In the for *Output Harmonics* cyclic field, choose *Number of harmonics* and type 1 in the adjacent field.

   f. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.

   g. Verify that *Enabled* is highlighted.

The completed form looks like this.

h. Click *OK*.

4. In the ADE window, choose *Simulation – Netlist and Run*.

Look in the CIW for messages saying that the simulation has started and completed successfully. Watch the simulation log file for information as the simulation runs.

Looking at the Envelope results

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form and the waveform window appear.

2. In the Direct Plot form, do the following:

   a. Choose *Replace* for *Plotting Mode*.

   b. Highlight *envlp* for *Analysis*.

   c. Highlight *Voltage* for *Function*.

   d. Highlight *time* for *Sweep*.

   e. The *Select* cyclic field displays *Net* and a prompt displays at the bottom of the form.

      > *Select Net on Schematic...*.

The completed form looks like this.

3. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the transmitter output net (/RFOUT)

The voltage waveform for *RFOUT* appears in the waveform window.

4. To get a closer look at a small section of the waveform, in the waveform window, right click and drag to select a small section of the waveform. (The area you select covers a narrow vertical rectangle.)

You might have to do this several times to get a plot similar to the one shown.

The plot changes to display a number of vertical lines with a wavy line running through them.

- Each vertical line is a time point where a detailed calculation was performed.
- The wavy line connects these points

An Envelope analysis runs much faster than a Spectre transient analysis because Envelope analysis skips carrier cycles when it can do so while still satisfying numerical tolerances.

5. To get a closer look, right click and drag to select one of the vertical lines. You might have to do this several times.

   After several selections, you are able to see the detailed simulation plot for one complete cycle. What you see should be similar to the following waveform depending on the areas you selected.

Following the Baseband Signal Changes Through an Ideal Circuit

The modulation riding on the RF carrier is the baseband signal, the information to be transmitted. The baseband signal determines the amplitude and phase of the RF carrier. In transmitter design, it is important to determine how the transmitter might alter the baseband signal.

This section illustrates how to extract and plot the baseband signal at several points in the circuit.

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form and the waveform window appear.

2. In the Direct Plot form, do the following:

   a. Choose *Replace* for *Plotting Mode*.

   b. Highlight *envlp* for *Analysis*.

   c. Highlight *Voltage* for *Function*.

   d. The *Select* cyclic field displays *Net* and a prompt at the bottom of the form. displays

   ```
   > Select Net on Schematic.
   ```

   e. Highlight *time* for *Sweep*.

3. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the I-modulator source net (the I-modulator input net.

Note: You might have to click *Zoom – Fit* in the waveform window before you are able to see the waveform.

The plot for the modulation source waveform appears.

4. In the Direct Plot form, highlight *Append* for *Plotting Mode*.

This adds new waveforms to any waveforms currently displayed in the waveform window.

5. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the I-modulator output net as shown in the following schematic.

This is the in-phase carrier modulated by the I-component of the baseband signal. In this example,

- The in-phase carrier is `cos(`ω_c`t)`, where ω_c is the carrier frequency in radians per second.
- The quadrature carrier is `sin(`ω_c`t)`.

The I-modulator output waveform is added to the waveform window.

6. In the waveform window, choose *Axis – Strips*.

This changes the waveform window to display multiple waveforms in strips, one above the other.

7. In the waveform window, right click and drag over a narrow section of both waveforms.

You can see the modulation wave within the output wave. It is coincidental that the top waveform and smooth curve in the bottom waveform look alike. If the individual cycles in

the bottom waveform were sampled at a different phase, the smooth curve in the bottom waveform might look different.

Note: The appearance of this plot, as well as the appearance of subsequent plots, depends on the section of the waveform window in which you perform the click and drag.

8. In the Direct Plot form, do the following:

a. Highlight *harmonic time* for *Sweep*.

b. Highlight *Real* for *Modifier*.

c. Highlight *1* for *Harmonic Number*.

d. The *Select* cyclic field displays *Net* and a prompt at the bottom of the form.

> Select Net on Schematic....

Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the I-modulator output net.

9. The bottom plot in the waveform window is the baseband waveform recovered from the modulated RF carrier. Aside from a linear scale factor, It looks exactly like the top waveform because the modulator is ideal.

Following the Baseband Signal Changes Through a Non-Ideal Circuit

To see the change in the signal as it passes through the entire length of the circuit, which is not ideal, follow the instructions in this section.

1. In the waveform window, choose *File - Close* to close the waveform window.
2. In the Direct Plot form, do the following:
   a. Choose *Replace* for *Plotting Mode*.
   b. Highlight *envlp* for *Analysis*.
   c. Highlight *Voltage* for *Function*.
   d. Highlight *time* for *Sweep*.
   e. The *Select* cyclic field displays *Net* and a prompt at the bottom of the form

      `> Select Net on Schematic....`
3. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the I-modulator source net.

The plot for the modulation source waveform appears.

You might have to choose *Zoom – Fit* in the waveform window before you are able to see the waveform.

4. In the waveform window, choose *Axes – To Strip*.

   This changes the waveform window to display multiple waveforms in strips, one above the other.

5. In the Direct Plot form, highlight *Append* for *Plotting Mode*.

   This adds new waveforms to any waveforms currently displayed in the waveform window.

6. Following the prompt at the bottom of the Direct Plot form, in the Schematic window, click the transmitter output net (RFOUT).

The transmitter output waveform is added to the waveform window.

7. In the waveform window, right click and drag over a narrow section of both waveforms.

You can see the modulation wave within the output wave. It is coincidental that the bottom waveform and smooth curve in the top waveform look alike. If the individual cycles in the top waveform were sampled at a different phase, the smooth curve in the top waveform might look different.

Note: The appearance of this plot and of the subsequent plots depends on the section of the waveform window in which you perform the click and drag.

8. In the Direct Plot form, do the following:

   a. Highlight *harmonic time* for *Sweep*.

   b. Highlight *Real* for *Modifier*.

   c. Highlight *1* for *Harmonic Number*.

9. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the transmitter output net (RFOUT).

The waveform at the transmitter output appears at the top of the waveform window. It shows a greater change in the signal.

Plotting the Complete Baseband Signal

1. In the Direct Plot form, do the following:

a. Choose *Replace* for *Plotting Mode*.

b. Highlight *envlp* for *Analysis*.

c. Highlight *Voltage* for *Function*.

d. The *Select* cyclic field displays *Net* and a prompt at the bottom of the form.

```
> Select Net on Schematic.
```

e. Highlight *harmonic time* for *Sweep*.

f. Highlight *Real* for *Modifier*.

g. Highlight *1* for *Harmonic Number*.

2. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the adder output net.

A plot for the real portion of the adder output waveform appears in the waveform window.

3. In the Direct Plot form, make the following changes:

a. Choose *Append* for *Plotting Mode*.

b. Highlight *Imaginary* for *Modifier*.

4. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the adder output net.

A plot for the imaginary portion of the adder output waveform is added to the waveform window.

5. In the waveform window, right click and drag to select a portion of the waveform window over a narrow portion of the x-axis.

You can now clearly see both the real part and the imaginary part of the baseband signal.

Plotting the Baseband Trajectory

This section describes how to display the baseband trajectory, the plot of one waveform against the other. The baseband waveforms recovered from the modulated RF carrier, as displayed in the previous section, do not directly reveal much information about how the transmitter affects them.

The baseband trajectory reveals much more information about the kind of distortion the transmitter introduces. The procedure described in this section displays first the input baseband trajectory followed by the output baseband trajectory. A comparison of the two trajectories reveals whether the power amplifiers in this example introduce any phase shift.

Displaying the Input Baseband Trajectory

1. In the waveform window, choose *Window - Reset*.
2. In the Direct Plot form, do the following:
   a. Choose *Replace* for *Plotting Mode*.

b. Highlight *envlp* for *Analysis*.

c. Highlight *Voltage* for *Function*.

d. The *Select* cyclic field displays *Net* and a prompt at the bottom of the form.

```
> Select Net on Schematic.
```

e. Highlight *harmonic time* for *Sweep*.

f. Highlight *Real* for *Modifier*.

g. Highlight *1* for *Harmonic Number*.

The completed form looks like this.

3. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the adder output net.

A plot for the real portion of the adder output waveform appears in the waveform window.

4. In the Direct Plot form, change the following:

   a. Choose *Append* for *Plotting Mode*.

   b. Highlight *Imaginary* for *Modifier*.

5. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the adder output net.

A plot for the imaginary portion of the adder output waveform is added to the real portion in the waveform window.

6. In the waveform window, choose *Axes – X Axis*.

The X Axis form appears.

7. In the *Plot vs.* cyclic field of the X Axis form, select the real portion of the waveform. Choose the selection that contains *re(V)*. In this case, choose

```
1    h-1;   v    /net64;  envlp  re(V)
```

8. Click *OK*.

The input baseband trajectory, undistorted by the power amplifiers, appears in the waveform window.

Displaying the Output Baseband Trajectory

1. In the ADE window, choose *Tools – Waveform*.

   A second waveform window appears.

2. In the Direct Plot form, do the following:

   a. Highlight *Replace* for *Plotting Mode*.

   b. Highlight *envlp* for *Analysis*.

   c. Highlight *Voltage* for *Function*.

   d. The *Select* cyclic field displays *Net* and a prompt at the bottom of the form.

   ```
   > Select Net on Schematic.
   ```

   e. Highlight *harmonic time* for *Sweep*.

   f. Highlight *Real* for *Modifier*.

   g. Highlight *1* for *Harmonic Number*.

The completed form looks like this.

3. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the transmitter output net (/RFOUT).

A plot of the real portion of the transmitter output waveform appears in the new waveform window.

4. In the Direct Plot form, change the following:

a. Highlight *Append* for *Plotting Mode*.

b. Highlight *Imaginary* for *Modifier*.

5. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click the transmitter output net (RFOUT).

A plot for the imaginary portion of the transmitter output waveform is added to the real portion in the waveform window.

6. In the waveform window, choose *Axes – X Axis*.

The X Axis form appears.

7. In the *Plot vs.* cyclic field of the X Axis form, select the real portion of the waveform. Choose the selection that contains *re(V)*. In this case, choose

```
1    h-1;    v    /RFOUT;  envlp  re(V)
```

8. Click *OK*.

The output baseband trajectory appears in the second waveform window.

As shown in Figure A-18 on page 1982, the output baseband trajectory (on the right) is the input baseband trajectory (on the left) scaled linearly and rotated.

Figure A-18 Input and Output Baseband Trajectories

In other words, the output baseband signal is the input baseband signal multiplied by a complex constant. The input and output waveforms look different because of the rotation, not because of some non-linear distortion. A common non-linear distortion, such as saturation, would make the outer edges of the trajectory lie on a circle.

Measuring ACPR and PSD

Adjacent Channel Power Ratio (ACPR) is a common measure of the power a transmitter emits outside its allotted frequency band. ACPR is the ratio of the power in an adjacent band to the power in the allotted band.

$$ACPR = \frac{\text{Power in an adjacent band}}{\text{Power in the allotted band}}$$

Regardless of how you choose the frequencies and bands for the ACPR measurement, ACPR is always extracted from the power spectral density (PSD) of the transmitted signal. PSD is a frequency-by-frequency average of a set of discrete Fourier transforms (DFTs) of the baseband signal. Here, the baseband signal is the harmonic-time result of an *envlp* analysis.

The ACPR Wizard

The ACPR Wizard simplifies the complicated calculations needed to measure ACPR. It determines the appropriate *ENVLP* simulation parameters and `psdbb` function arguments from the information you supply in the ACPR wizard form. The ACPR wizard form contains a minimum number of clearly labeled fields so you can easily supply the required information.

After you provide the information and press *Apply* or *OK* in the ACPR wizard, the calculated values are used to fill in the *ENVLP* Choosing Analyses form and the *ENVLP* Options form. You can then run the simulation and analyze the PSD and ACPR data.

Open the ACPR wizard in one of two ways.

- In the ADE window, choose *Tools – RF – Wizards – ACPR*

  or

- In the *ENVLP* Choosing Analyses form, press *Start ACPR Wizard*.

In either case the ACPR Wizard displays.

ACPR Wizard
Clock Name
Update From Hierarchy
How to Measure
Net
Net
Select
Power
Power Density
Channel Definitions
Custom
Main Channel Width (Hz)
Adjacent frequencies are specified relative to the center of main channel
name
from (Hz)
to (Hz)
low
high
Add
Change
Delete
Simulation Control
Symbol Start(Sec)
0.0
Symbol Stop(Sec)
Symbol Stop = Symbol Start +(Strobe Period * Window Size * Repetitions)
Strobe Period
Window Size
64
Window Size should be the n-th power of 2
Repetitions
2
Resolution Bandwidth (Hz)
Default
Resolution Bandwidth = 1/(Strobe Period * Window Size)
Windowing Function
Cosine4

Measuring ACPR

The ACPR wizard minimizes the number of items you need to supply. Its fields are easy to understand and fill in. After you provide the ACPR wizard field values, these values are used to calculate and fill in the field values on both the *ENVLP* Choosing Analyses and Options forms. The ACPR wizard computes the appropriate simulation parameters and `psdbb` arguments from the information you enter in the ACPR wizard.

To use the ACPR wizard,

1. Fill in the ACPR wizard form.
2. Click *OK* or *Apply* in the ACPR wizard to calculate field values for both the *ENVLP* Choosing Analyses and *ENVLP* Options forms.
3. Run the *ENVLP* analysis. When the simulation completes, the *ENVLP* analysis results are plotted.

This example illustrates how to use the ACPR wizard and to set up and run the Envelope analysis. This example uses the *EF_example* circuit from your writable copy of the *rfExamples* library.

Before you start, perform the setup procedures described in Chapter 3 if you have not yet set up the writable copy of the *rfExamples* library.

Set up the *EF_example* schematic and the simulation environment.

1. Open the *EF_example* schematic as described in "Opening the EF_example Circuit in the Schematic Window" on page 1947.
2. Open the ADE window as described in "Opening the Simulation Window" on page 1949.
3. Set Up the Model Libraries as described in "Setting Up the Model Libraries" on page 1949.
4. Edit the *PWL file names* for *PORT0* and *PORT1* as described in "Editing PORT0 and PORT1 in the EF_example Schematic" on page 1950.

Setting Up the ACPR Wizard and the ENVLP Analysis

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.
2. In the Choosing Analyses form, highlight *envlp*.

The Choosing Analyses form changes to let you specify values for the Envelope analysis.

3. In the *ENVLP* Choosing Analyses form, click *Start ACPR Wizard*.
4. Leave the *ENVLP* Choosing Analyses form open in the background.

The ACPR wizard appears.

ACPR Wizard
Clock Name
Update From Hierarchy
How to Measure
Net
Net
Select
Power
Power Density
Channel Definitions
Custom
Main Channel Width (Hz)
Adjacent frequencies are specified relative to the center of main channel
name
from (Hz)
to (Hz)
low
high
Add
Change
Delete
Simulation Control
Symbol Start(Sec)
0.0
Symbol Stop(Sec)
Symbol Stop = Symbol Start +(Strobe Period * Window Size * Repetitions)
Strobe Period
Window Size
64
Window Size should be the n-th power of 2
Repetitions
2
Resolution Bandwidth (Hz)
Default
Resolution Bandwidth = 1/(Strobe Period * Window Size)
Windowing Function
Cosine4

5. In the ACPR wizard, do the following.

a. In the *Clock Name* cyclic field, select *fff*.

The clock name identifies the source of the modulated signal.

b. In the *How to Measure* cyclic field, select *Net*.

You can measure ACPR

- For a single *Net* (Select one net, *Net*)
- Between *Differential Nets* (Select two nets, *Net*+ and *Net*-)

c. To select a single output net in the Schematic window, click *Select* to the right of the *Net* field. In the Schematic window, select the transmitter output net, *RFOUT*.

The output net name, */RFOUT*, displays in the *Net* field.

The top of the ACPR wizard appears below.

d. In the *Channel Definitions* cyclic field, select the *IS-95* standard.

- The *Main Channel Width (Hz)* field is calculated as `1.2288M`.
- The *adjacent frequencies* are determined and display in the list box.

e. Note the text on the ACPR Wizard form,

Adjacent frequencies are specified relative to the center of the main channel

- *Channel Definitions* presets include *Custom*, *IS-95*, and *W-CDMA*
- *Main Channel Width (Hz)* specifies a frequency band in Hz

The middle of the ACPR wizard appears below.

Use the edit fields below the list box and the *Add*, *Change* and *Delete* buttons to modify channel definitions in the list box.

You can hand edit or enter adjacent frequency names and *upper* and *lower* boundaries. Specify adjacent frequencies relative to the center of the main channel. All channel widths must be greater than zero.

f. In the *Stabilization Time (Sec)* field, enter `72n`.

`7.2e`$^{-08}$ displays in the *Stabilization Time (Sec)* field and `2` displays in the *Repetitions* field.

The *Stabilization Time* is the length of time in seconds to wait before using the data for analysis. *Repetitions* is the number of times to repeat the discrete Fourier transform for averaging.

Increasing the number of repetitions makes the power density curve smoother, but at the expense of longer simulation time and increased data file size.

g. Click *Calculate* to determine the *Resolution Bandwidth (Hz)*.

`7500` displays in the *Resolution Bandwidth (Hz)* field. *Resolution Bandwidth* specifies the spacing of data points on the resulting power density curve, in Hz. Reducing *Resolution Bandwidth* increases simulation time and data file size.

h. In the *Windowing Function* cyclic field, select *Cosine4*.

Windowing Function presets include *Blackman, Cosine2, Cosine4, ExtCosBell, HalfCycleSine, HalfCycleSine3, HalfCycleSine6, Hamming, Hanning, Kaiser, Parzen, Rectangular* and *Triangular.*

The *Simulation Control* section at the bottom of the ACPR Wizard looks like the following.

6. In the ACPR Wizard, click *Apply*

When you click *OK* or *Apply* in the ACPR wizard, values are calculated and appear in the *ENVLP* Choosing Analyses form.

If necessary choose *Choose – Analyses* in the ADE window to display the ENVLP Choosing Analyses form.

Values in the *ENVLP* Choosing Analyses form are as follows.

- The *Clock Name* is the same in both forms. In this case `fff`.
- *Stop Time* for *ENVLP* is calculated as `0.0002669271`.
- For the *ENVLP* analysis O*utput Harmonics, Number of Harmonics* is selected in the cyclic field and the *Number of Harmonics* is set to `1`.
- The *ENVLP* analysis is *Enabled*
- The *ENVLP* analysis *errpreset* is *moderate*.

Values in the *ENVLP* Options form are as follows.

- The *ENVLP* option *start* is set to blank.
- The *ENVLP* option *modulationbw* is calculated as `1098000.0`
- The *ENVLP* option *strobeperiod* is calculated as `2.604167e-07`.

  Note: You can modify values on the *ENVLP* Choosing Analyses and Options forms but your changes are not propagated back to the ACPR wizard.

7. In the *ENVLP* Choosing Analyses form, click *Options* to open the *ENVLP* Options form.

Notice that values for the *modulationbw* and *strobeperiod* parameters are calculated and the *start* parameter is blank.

The section for the *start* and *modulationbw* parameters looks like this.

The section for the *strobeperiod* parameter looks like this.

8. Click *OK* in the Envelope Following Options form.
9. Verify that *Enabled* is highlighted and click *Apply* in the *ENVLP* Choosing Analyses form.
10. Click *OK* in the *ENVLP* Choosing Analyses form.

    The Choosing Analyses form closes.
11. Click *OK* in the ACPR Wizard.

    The ACPR Wizard closes.

The ADE window reflects the *Analysis* and *Outputs* information from the ACPR wizard, the *ENVLP* Choosing Analyses form and the calculations that resulted.

12. In the ADE window, choose *Simulation – Netlist and Run*.

Look in the CIW for messages saying that the simulation has started and completed successfully. Watch the simulation log file for information as the simulation runs.

When the simulation successfully completes, an ACPR value for each channel appears in the *Value* column in the ADE window *Output* section.

- The *ACPR* `psd` */RFOUT Value* is a *wave* which is plotted.
- The *ACPR lower* channel value is `-61.56`.
- The *ACPR upper* channel value is `-60.07`.

When the simulation finishes, the PSD plot displays as in Figure A-19.

Figure A-19 PSD Plot Generated by the ACPR Wizard

Estimating PSD From the Direct Plot Form

The power spectral density (PSD) is always estimated because the information riding on the carrier is a stochastic process and the Fourier transform of a stochastic process is ill defined. No matter how you chose to define the spectral nature of a stochastic process, it always involves an averaging process. Any empirically derived average is an estimate because you can never take an infinite number of samples.

PSD is a frequency-by-frequency average of a set of discrete Fourier transforms (DFTs) of the baseband signal. Here, the baseband signal is the harmonic-time result of an *ENVLP* analysis.

This example shows how to estimate PSD given the results of the *ENVLP* analysis you just performed.

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.

2. In the Direct Plot form, choose *New Win* for *Plotting Mode*.

This plots PSD in a new waveform window.

3. Highlight *ENVLP* for *analysis.*
4. Highlight *Voltage* for *Function*.
5. Highlight *spectrum* for *Sweep*.

   The *Power Spectral Density Parameters* section opens at the bottom of the form.
6. Highlight *Magnitude* for *Modifier*.
7. Choose *1* for *Harmonic Number*.

The top of the Direct Plot form looks like the following.

8. In the Bottom of the Direct Plot form, enter information for the *Power Spectral Density Parameters*

a. Press *Get From Data* to calculate the values for the *Time Interval* values, *From* is `0.0` and *To* is `0.0002669271`.

b. Type `5M` for *Nyquist half-bandwidth*.

c. Type `.1M` for *Frequency bin width*.

d. Type `3M` for *Max. plotting frequency*.

e. Type `-3M` for *Min. plotting frequency*.

f. In the *Windowing* cyclic field, select *Cosine4*.

g. In the *Detrending* cyclic field, select *None*.

The *Power Spectral Density Parameters* area at the bottom of the *ENVLP* Direct Plot form is as follows.

h. Following the prompt at the bottom of the form,

```
> Select Net on schematic.
```

Click the transmitter output net (*/RFOUT*).

The PSD plot displays.

Figure A-20 PSD mag (V) from the ENVLP Direct Plot Form

To plot the modified PSD plot,

1. In the Direct Plot form, make the following changes:

a. Choose *New Win* for *Plotting Mode*.

b. Highlight *dB10* for *Modifier*.

The top of the Direct Plot form looks like this.

c. Select the RFOUT net in the Schematic window.

The modified psd plot displays, as shown in Figure A-21.

Figure A-21 Estimated PSD from the ENVLP Direct Plot Form

Compare Figure A-21 with the PSD plot shown in Figure A-20 on page 2002.

Reference Information for ACPR and PSD Calculations

The process outlined in this section is complex largely because the *ENVLP* analysis parameters are not directly related to the calculator `psdbb` function arguments.

- The *ENVLP* parameters include nyquist half-bandwidth, frequency bin width and time interval.
- The *psdbb* function arguments are the total number of samples, the window size and the bin-width. All three parameters are in terms of the number of DFT time samples.

However, to make optimum use of the simulation data, it is necessary that the simulation parameters and the `psdbb` function arguments be compatible.

The Power Spectral Density (PSD) Parameters

When you select *spectrum* for *Sweep* in the Direct Plot form, the *Power Spectral Density Parameters* section opens at the bottom of the Direct Plot form. The *Power Spectral Density Parameters* section displays the analysis parameters that control the PSD estimate.

The *Power Spectral Density Parameters* section is shown in Figure A-21.

PSD Parameters on the ENVLP Direct Plot Form

The PSD Parameters are described in Table A-1.

Table A-1 Power Spectral Density Parameters From the Direct Plot Form

PSD Parameter	Description
Time Interval	The *Time Interval* parameters specify the time record to analyze in the frequency domain. The time interval should be long enough to support the required frequency resolution. Use *Get From Data* to calculate the *From* and *To* values from the ACPR wizard data.
Nyquist half-bandwidth	The *Nyquist half-bandwidth* parameter is half the sampling frequency used in the DFTs. Make the nyquist half-bandwidth large enough to prevent aliasing. The true spectrum of the baseband signal should be negligible at this frequency and beyond it.
Frequency bin width	The *Frequency bin width* parameter specifies the required frequency resolution. ■ When this value is too small; the resulting PSD looks noisy and the PSD has a jagged appearance. ■ When this value is too large; the resulting PSD might be softened when the spectrum should have sharp edges. *Windowing Function* presets include *Blackman, Cosine2, Cosine4, ExtCosBell, HalfCycleSine, HalfCycleSine3, HalfCycleSine6, Hamming, Hanning, Kaiser, Parzen, Rectangular* and *Triangular.*
Windowing	The *Windowing* parameter specifies which windowing function to apply before performing the DFTs.
Detrending	The *Detrending* parameter has one of three values: *None*, *Mean*, or *Linear*.

If necessary, the waveform is first interpolated to generate evenly spaced data points in time. The data point spacing is the inverse of the DFT sampling frequency. The PSD is computed by

- Breaking the time interval up into overlapping segments
- Multiplying each segment, time point by time point, by the specified *Windowing* function.

Windowing reduces errors caused by a finite time record. It is impossible to work with an infinite time record. Direct use of an unwindowed finite time record is equivalent to multiplying the infinite record by a rectangular pulse that lasts as long as the data record. Multiplication in the time domain corresponds to convolution in the frequency domain. The Fourier transform of a rectangular pulse is a *sinc* function. Considering the frequency domain convolution, the side lobes of the sinc function cause parts of the true spectrum to leak into the frequency of interest, that is, the frequency of the main lobe. Ideally, the sinc function would be a Dirac delta function but that requires an infinite time record. Good window functions have smaller side lobes than the sinc function.

The DFT is performed on each windowed segment of the baseband waveform. At each frequency, the DFTs from all segments are averaged together. Fewer segments means fewer data points in the average at a particular frequency. The length of each segment is inversely proportional to the *Frequency bin width*, which is why a small *Frequency bin width* produces a jagged PSD. A smaller *Frequency bin width* means a longer time segment.

Fewer long segments fit into the given time interval so there are fewer DFTs to average together. In the extreme, there is only one segment and no averaging. Without averaging, the PSD is the square of the magnitude of the DFT of a stochastic process. At the other extreme, large *Frequency bin widths* produce lots of points to average at each frequency but there are fewer frequencies at which to average because fewer large bins fit into the Nyquist frequency. The PSD is smoother but it does not have as much resolution.

PSD is always estimated because the information riding on the carrier is a stochastic process and the Fourier transform of a stochastic process is ill defined. No matter how you chose to define the spectral nature of a stochastic process, it always involves an averaging process. Any empirically derived average is an estimate because you can never take an infinite number of samples.

The calculator *psdbb* function performs a discrete Fourier transform (DFT) on the voltage curve which produces the power spectral density (PSD) curve. Integrate the PSD curve to calculate power in the channel.

The Calculator psdbb Function

The calculator *psdbb* function, which estimates PSD, derives the function parameters it requires from the PSD parameters and values you supply in the ENVLP Direct Plot form.

- *Nyquist half-bandwidth*
- *Frequency bin width*
- *Time Interval*

See Table A-1 for more information about the PSD parameters on the Direct Plot form.

The calculator *psdbb* function parameters are

- The total number of samples
- The window size
- The bin-width

All three parameters are in terms of the number of DFT time samples.

The calculations defined in Table A-2 generate the `psdbb` parameters. The calculated *psdbb* parameters are printed in the CIW window.

Table A-2 Parameter Values Calculated by the psdbb Calculator Function

psdbb Calculation	**Source of the Data**
L = *To* −*From*	*To* and *From* are the values from the *Time Interval* *To* and *From* fields on the *ENVLP* Direct Plot form.
f_{max}	*Nyquist half-bandwidth* and *Frequency bin width* are values from these fields on the *ENVLP* Direct Plot form.
#bins = floor(L*binwidth)	#bins >= 1. Here, *floor* means *to take the integer part of*, i.e. truncate to the nearest integer.
2^m * (#bins) > 2 * L * f_{max}	Compute the smallest m.
windowsize=2^m	
#bins * windowsize	Compute the number of samples.

You might want to use the `psdbb` function directly when strobing time-harmonic results to eliminate interpolation error. Use of the `psdbb` function is described in the calculator documentation.

Calculating ACPR

This section describes how to calculate the ACPR (Adjacent Channel Power Ratio) by hand. The ACPR Wizard performs these calculations for you.

ACPR is measured with respect to any x_1 and x_2, as y_1 - y_2. You can calculate the ACPR for any two x-axis values by subtracting their associated y-axis values.

For example, given the following two spectral parameters,

- The adjacent channel is 2.5 MHz from the carrier
- You choose to define ACPR in terms of power at just two frequencies,
  - 2.5 MHz
  - 0 MHz

Use the cursor to determine the RF output power at 2.5 MHz and 0 MHz frequencies. As you slide the cursor along the curve, you read the X and Y values off the top of the waveform window:

```
X:2.5M    Y:-115.8        v /RFOUT;envlp mag(V)
```

and

```
X:0    Y:-77.33        v /RFOUT;envlp mag(V)
```

So

- The power at 2.5 MHz equals -115.8 dB
- The power at 0 MHz equals -77.33 dB

Remember that the horizontal scale is frequency offset from the carrier fundamental. The difference between these measurements equals -38.47 dB.

$$ACPR = -115.8 - (-77.33) = -38.47$$

For this definition of ACPR, ACPR = -38.47 dB.

Other definitions of ACPR are possible. You can compute them by adjusting the spectral parameters and applying the calculator to the spectral plot.

Calculating PSD

For a constant baseband signal, calculate power spectral density as shown in Table A-3.

Table A-3 Power Spectral Density for a Constant Baseband Signal

Baseband signal	$1 + j \times q$ where $j = \sqrt{-1}$

Table A-3 Power Spectral Density for a Constant Baseband Signal

Passband signal	$i \times \cos(w \times t) - q \times \sin(w \times t)$
Baseband signal power	$i \times i + q \times q \qquad volts \times volts$
Passband signal power	$i \times \frac{i}{2} + q \times \frac{q}{2} = (1/\ 2) \times (baseband""power)$

Envelope Spectral analysis computes these values per Hz. You can also express these values as

$$(rms""\text{passband}""\text{volts}) \times (rms""\text{passband}""\text{volts})\ /\ (Hz)$$

Envelope Time-Harmonic analysis computes peak volts because they can be directly compared with the modulating signals.

For this example

- You see a waveform displayed in the waveform window as V^2 / (Hz) versus frequency.
- You can think of this as

$$(rms""\text{passband}""\text{volts}) \times (rms""\text{passband}""\text{volts})\ /\ (Hz)$$

Important

The displayed spectrum is the estimated PSD of the complex envelope divided by two. The division by two is included because the envelope is expressed in units of peak carrier volts, but power in the carrier equals the square of the peak divided by two. It is convenient to express the envelope in peak units because you can then directly compare it against an input baseband signal.

PSD and the Transmitted Baseband Signal

When you measure ACPR, it is crucial that you drive the transmitter with the proper baseband signals. The baseband signals driving the transmitter dominate the transmitted PSD. In most cases, the baseband signals are produced by digital filters so the digital filters constrain the

spectrum of the input baseband signal. Distortion in the transmitter causes the spectrum to grow where it should not, hence the need for an ACPR measurement.

It is not practical to model digital filters in SpectreRF because SpectreRF cannot simulate state variables inside Verilog®-A modules. Consequently, for now, you must pre-compute and store the baseband inputs and then read them into the SpectreRF analysis through *ppwlf* sources found in the *analogLib*, as shown in "Computing the Spectrum at the Adder Output" on page 2011. The *ppwlf* sources also read SPW format, so you can also generate and record the input baseband waveforms using SPW.

The *rfLib* contains three sets of stored baseband waveforms, *cdma*, *dqpsk*, and *gsm*. These waveforms were created with the baseband signal generators in the *measurement* category of the *rfLib*.

If you want to measure ACPR with the noise floor much more than 40 dB below the peak of the output power spectral density, you must create baseband drive signals with a noise floor at or below the required noise floor. If you use a DSP tool such as SPW to create the signals, the filters in the baseband signal generator must operate perhaps hundreds of times faster than those in the actual generator.

Sometimes an ACPR specification exceeds the ACPR of the baseband drive signals. To see if the transmitter meets specifications in that event, it must be driven with an unrealistic baseband signal. Otherwise the signals do not have enough resolution. The noise floor depends heavily on interpolation error.

The Fourier analysis used to compute the power spectral density uses evenly spaced time points. If data does not exist at one of the Fourier time points, the Fourier algorithm must interpolate to create the missing Fourier time point.

You can strobe the harmonic time results to eliminate interpolation of the output but you cannot eliminate interpolation of the baseband drive signals. The only way you can reduce interpolation errors at the input is to use ultra-high-resolution drive signals so that no matter where the interpolation occurs, the error is small. For now, you must generate the ultra-high-resolution drive signals yourself.

Computing the Spectrum at the Adder Output

The following results were obtained by

1. Generating high resolution baseband signals in SPW
2. Storing the high resolution baseband signals
3. Reading the high resolution baseband signals into an *ENVLP* analysis through the *PWL* sources

The SPW FIR filters operate at 300 times the chip rate.

Figure A-22 shows the spectrum at the output of the adder. In this circuit, the RF signal is undistorted at the adder output. The spectrum was computed three times. Each time with a different value for *number of samples*.

The ENVLP analysis was run with the following parameters:

reltol	`1e`$^{-5}$
strobeperiod	`100n`

Note: If the *strobeperiod* does not equal an integer number of clock cycles, it is internally truncated to an integer number of clock cycles.

For the `psdbb` function

Time interval	`100u` to `1m`
Window size	`1024`
Windowing	*Hanning*

Note: *Window size* must be a power of 2. If you enter a value that is not a power of 2, the `psdbb` function truncates the value to the nearest power of 2.

With *strobeperiod* set to `100` ns (100 clock cycles), the simulation produced 9000 evenly spaced samples inside the time interval. The spectra were calculated using 8500, 8750, 9000, 9250, and 9500 points for the *number of samples*. Figure A-22 clearly shows interpolation errors when the number of samples used in the `psdbb` function does not equal the actual number of samples.

Figure A-22 Adder Output Spectrum Computed with Three Different Numbers of Samples

Figure A-23 compares the input and output PSD plots using the high resolution drive signals and the following set of `psdbb` parameters:

From	`100u`
To	`1m`
Number of Samples	9000
Window Size	1024
Window Type	Hanning

In Figure A-23, the upper PSD plot is the power amplifier output and clearly shows spectral regrowth when compared to the input PSD plot.

Figure A-23 Comparison of Input and Output PSD plots

Measuring Load-Pull Contours and Load Reflection Coefficients

This section describes how to generate load pull contours and how to determine whether you must redesign the input matching network for the optimal load.

A *load-pull contour* is a set of points on a Smith chart representing all the loads that dissipate a given amount of power. Load-pull contours help you match a load to a power amplifier for maximum power transfer. Just as a topographical map shows a hiker where the mountain peak lies and how steep the climb is, load-pull contours show which load dissipates the most power and how sensitive that power is to small load perturbations.

To properly use Load-Pull results, you must understand how the Load-Pull feature defines load reflection coefficients. Load reflection coefficients are computed using the PSS analysis.

The *load reflection coefficients* are computed from the load impedance, which is computed as the ratio of the voltage across the load to the current flowing into the load at the RF carrier frequency. Suppose the RF carrier is 1 GHz and the load waveforms are distorted. The impedance is not computed from small-signal perturbations about an operating point. Rather, the impedance is computed as the ratio of the 1 GHz components of the load voltage and current waveforms simulated by a PSS analysis. Because the load is passive and linear, the load reflection coefficient computed in this manner equals the small-signal, or incrementally computed, load impedance.

This is not necessarily true for the *input reflection coefficient* because the input circuitry can be non-linear and it can also contain input offset voltages and currents. However, because matching networks are usually designed only for the RF fundamental, defining the reflection coefficient as the ratio of fundamental Fourier components of the large signals is often justified.

Creating and Setting Up the Modified Circuit (EF_LoadPull)

This example tells you how to

- Create the *EF_LoadPull* schematic for this example. The *EF_LoadPull* schematic is a modified copy of the *EF_example* schematic from the *rfExamples* library.
- Set up and run the necessary PSS and Parametric analyses
- Measure load-pull contours for the modified *EF_LoadPull* schematic
- Measure load reflection coefficients for the modified *EF_LoadPull* schematic

Before you start, make sure you have performed the setup procedures for the writable *rfExamples* library, as described in Chapter 3.

Creating a New Empty Schematic Window

1. In the CIW choose *File – New – Cellview*.

   The Create New File form appears.

2. In the Create New File form, do the following:

   a. Choose *my_rfExamples* for *Library Name*.

   Select *my_rfExamples*, the editable copy of the *rfExamples* library you created following the instructions in Chapter 3.

   In the *Cell Name* field, enter *EF_LoadPull*.

   b. In the *View Name* field, enter *schematic*.

   c. In the *Tool* cyclic field, select *Composer-Schematic*.

   d. Click *OK*.

   A new, empty Schematic window named *EF_LoadPull* opens.

Opening and Copying the EF_example Circuit

Now open another schematic window containing the *EF_example* circuit. Then copy the *EF_example* circuit into this empty *EF_LoadPull* Schematic window.

1. In the CIW, choose *File – Open*.

   The Open File form appears.

2. In the Open File form, do the following:

   a. Choose *my_rfExamples* for *Library Name*.

      Select the editable copy of the *rfExamples* library you created following the instructions in Chapter 3.

   b. In the *Cell Name* list box, highlight *EF_example*.

   c. Choose *schematic* for *View Name*.

   d. Highlight *edit* for *Mode*.

   e. Click *OK*.

   The Schematic window appears with the *EF_example* schematic as shown in Figure A-23. This is a simple direct-conversion transmitter with ideal I/Q modulators.

The EF_example Schematic

3. In the *EF_example* Schematic window, choose *Edit - Copy* and follow the prompts at the bottom of the Schematic window.

4. Following the prompt,

   `> point at object to copy`

   left click and drag to create a box around the entire *EF_example* circuit.

   The *EF_example* components are highlighted in yellow.

5. Following the prompt,

   `> point at reference point for copy`

   click inside the outlined elements.

6. Following the prompt,

   `> point at destination point for copy`

   move the cursor to the empty *EF_LoadPull* Schematic window and click there. This drags a copy of the *EF_example* circuit into the empty Schematic window.

   The *EF_LoadPull* Schematic window now contains a copy of the *EF_example* schematic.

7. If necessary, choose *Window - Fit* to center the *EF_LoadPull* circuit in the Schematic window.

8. In the *EF_example* Schematic window, choose *Window - Close*.

   The *EF_example* Schematic window closes.

Opening the Simulation Window for the EF_LoadPull Schematic

➤ Open the ADE window from the *EF_LoadPull* Schematic window as described for the *EF_example* schematic in "Opening the Simulation Window" on page 1949.

Setting up the Model Libraries for the EF_LoadPull Schematic

➤ Set up the Model Libraries for the *EF_LoadPull* Schematic as described for the *EF_example* schematic in "Setting Up the Model Libraries" on page 1949.

Editing the EF_LoadPull Schematic

To prepare the schematic, follow the instructions in the following sections.

Delete Components and Wires from the EF_LoadPull Schematic

In the *EF_LoadPull* Schematic window, delete components and their associated connecting wires as shown in Figure A-24.

Figure A-24 Components and Wires to Delete for the EF_LoadPull Schematic

If you need assistance with methods for editing the schematic, see the *Virtuoso® Schematic Composer™ User Guide*.

1. In the *EF_LoadPull* Schematic window, choose *Edit – Delete.*
2. Click a component or wire to delete it.
3. Press the *Esc* key when you are done deleting.
4. After deleting the indicated components and wires, the *EF_LoadPull* Schematic should look like Figure A-25 on page 2020.

Figure A-25 EF_LoadPull Schematic After Deleting Components

5. In the *EF_LoadPull* Schematic window, choose *Edit – Move* to move *PORT5* closer to capacitor *C1* and to move *PORT6* away from capacitor *C7* to make room for the *PortAdaptor* component you add later.

   a. Left click and drag to create a rectangle around *PORT5* and its *GND*.

      PORT5 and the *Gnd* are highlighted in yellow.

   b. Click inside the yellow line and drag the outlined components to move them close to capacitor *C1* as shown in Figure A-26 on page 2022.

   c. Left click and drag to create a rectangle around *PORT6* and its *GND*.

      PORT6 and the *Gnd* are highlighted in yellow.

   d. Click inside the yellow line and drag the components to move them away from capacitor *C7* as shown in Figure A-26 on page 2022.

      This makes room for the *PortAdapter*.

6. Choose *Window - Fit* to center the edited schematic in the window.

Place the PortAdaptor

1. In the *EF_LoadPull* Schematic window, choose *Add – Instance*.

The Add Instance form appears.

2. In the Add Instance form *Library* field, type `rfExamples`.
3. In the *Cell* field, type `portAdapter`.
4. In the *View* field, type `Symbol`.

   As you move your cursor from the form to the *EF_LoadPull* Schematic window, a copy of the *portAdapter* moves with the cursor.
5. Left click to place the *portAdapter* as shown in Figure A-26 on page 2022.
6. After you place the *portAdapter* in the schematic, press the *Esc* key to remove the *portAdapter* symbol from your cursor and close the Add Instance form.

Figure A-26 The EF_LoadPull Schematic

Wire the Schematic

In the *EF_LoadPull* Schematic window, you need to connect *PORT5* to Capacitor *C1* and connect the *portAdapter* between Capacitor *C7* and *PORT6*. To do so,

1. In the Schematic window, choose *Add - Wire (Narrow)*.
2. Click the terminal on *PORT5* then click the terminal on *C1*.
3. Click the terminal on *C7* then click the terminal on the *PortAdaptor.*
4. Click the terminal on the *PortAdaptor* then click the terminal on *PORT6.*
5. Press the Esc key to stop wiring.

   The edited schematic looks like the one in Figure A-26.

The EF_LoadPull Schematic

Edit CDF Properties for both the PortAdaptor and Port6

In the *EF_LoadPull* Schematic, the *portAdapter* and its terminating port, *Port6*, must have the same reference resistance.

1. In the *EF_LoadPull* Schematic window, select the *PortAdapter*.
2. In the *EF_LoadPull* Schematic window, choose *Edit – Properties – Objects*.

   The Edit Object Properties form appears with information for the *Port Adapter* displayed.
3. In the Edit Object Properties form, do the following and click *Apply*.
   - **a.** Type *frf* for *Frequency*.
   - **b.** Type *theta* for *Phase of Gamma (degrees)*.
   - **c.** Type *mag* for *Mag of Gamma (linear scale)*.
   - **d.** Type *r0* for *Reference Resistance*.

The completed form looks like this.

CDF Parameter	Value
Frequency	frf
Phase of Gamma (degrees)	theta
Mag of Gamma (linear scale)	mag
Reference Resistance	r0
Gamma Phase Offset (deg)	0
Gamma Mag Offset (linear)	0

4. In the *EF_LoadPull* Schematic window, select *Port6*.

   The Edit Object Properties form changes to display data for *Port6*.

5. In the Edit Object Properties form for *Port6*, type *r0* for *Resistance* and click *OK*.

Select Outputs To Save

1. In the ADE window for the *EF_LoadPull* Schematic, choose *Outputs – To Be Saved – Select on Schematic*.

2. In the *EF_LoadPull* Schematic window, click each terminals that is circled in Figure A-27.

Figure A-27 Outputs to Save in the EF_LoadPull Schematic

After you click a terminal, it is circled in the schematic as shown in Figure A-27. The selected outputs are also displayed in the ADE window Outputs area as shown in Figure A-28.

Figure A-28 Outputs Area in the Simulation Window

Outputs

#	Name/Signal/Expr	Value	Plot	Save	March
1	PORT3/PLUS		no	yes	no
2	I3/out		no	yes	no

Editing Variables

1. In the ADE window, choose *Variables – Copy from Cellview*.

   The *Design Variables* list box in the ADE window changes to reflect the copied variables from the schematic as shown in Figure A-29.

Figure A-29 Design Variables Area Showing Variable Names

2. In the ADE window, choose *Variables - Edit*.

The Editing Design variables form appears.

3. In the Editing Design Variables form, do the following.

a. Highlight one of the variables In the *Table of Design Variables* list box.

The variable name displays in the *Name* field.

b. Type the variable's associated value from the following list into the *Value (Expr)* field.

- `theta 0`
- `r0 50`
- `mag 0`
- `frf 1G`

c. Click *Change*.

d. When you have given values to all four variables, click *OK*.

The Design Variables list box in the ADE window now displays both variable names and the value associated with each name.

Save the Changes to the EF_LoadPull Schematic

➤ In the Schematic window, choose *Design – Check and Save* to save the current state of the *EF_LoadPull* schematic.

Setting Up and Running the PSS and Parametric Analyses

This example tells you how to

- Set up the swept PSS analysis to sweep the design variable `theta`.
- Set up the Parametric analysis to sweep the design variable `mag`.
- Run the parametric and swept PSS analyses.

- Plot the results.

Setting Up the PSS Simulation

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, highlight *pss*.

   The form changes to display options for PSS simulation.

3. In the Choosing Analyses form, if information for *frf* is not displayed in the *Fundamental Tones* list box, type `frf` in the *Name* field below the list box, type `1G` in the *Expr* field below the list box and click *Clear/Add*.

4. Click *Update From Hierarchy* to update the values in the *Fundamental Tones* list box by searching all levels of the hierarchy. If you do not click *Update From Hierarchy*, only the top level of the hierarchy is searched.

5. Click *Auto Calculate* to automatically calculate and enter a value in the *Beat Frequency* field. 1G displays in the *Beat Frequency* field.

6. Choose *Number of harmonics* for *Output harmonics* and type `9` in the adjacent field.

   The top of the PSS analysis form looks like this.

7. Highlight *moderate* for *Accuracy Defaults (errpreset).*

8. Highlight the *Sweep* button and choose *Variable* in the associated cyclic field.
9. Type `theta` for *Variable Name* (or click *Select Design Variable*, highlight *theta* in the form that appears and click *OK*).
10. Choose *Start – Stop* for *Sweep Range*.
11. Type `0` for the *Start* value and `359` for the *Stop* value.
12. Choose *Linear* for *Sweep Type*.
13. Highlight *Number of Steps* and type `20` in the adjacent field.
14. Verify that *Enabled* is highlighted.

    The bottom of the PSS analysis form looks like this.

   a. Click *Apply*. Then click *OK*.

Setting Up the Parametric Analysis and Performing the Analyses

1. In the ADE window, choose *Tools – Parametric Analysis*.

   The Parametric Analysis form appears.

2. In the Parametric Analysis form, type *mag* for *Variable Name*.
3. Choose *From/To* in the *Range Type* cyclic field.
4. Type `0` for *From* and 0.`95` for *To* in the adjacent fields.
5. Choose *Linear* in the *Step Control* cyclic field.
6. Type `10` in the adjacent *Total Steps* field.

   The completed form looks like this.

7. In the Parametric Analysis form, choose *Analysis – Start*.
8. Look in the CIW for a message that says the simulation has completed successfully.

Displaying Load Contours

1. In the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.
2. In the Direct Plot form, select *Replace* in the *Plotting Mode* cyclic field.
3. Highlight *pss* for *Analysis*.
4. Highlight *Power Contours* for *Function*.

The top of the Direct Plot form looks like the following.

5. Highlight *Magnitude* for *Power Modifier*.
6. Leave *Maximum Power* and *Minimum Power* blank.
7. Type `9` for *Number of Contours*, if necessary.
8. Type `50.0` for *Reference Resistance*, if necessary.
9. Following the prompt at the bottom of the form,

   ```
   > Select Output Harmonic on this form...,
   ```

   highlight *1 1G* for *Output Harmonic*.

   You would select a different harmonic if the PSS fundamental frequency were smaller than `1 GHz`. For example, if another part of the circuit is driven at `1.5 Hz`, the

fundamental is `500 MHz`. But because the part of the circuit where you want to do load-pull analysis operates at `1 GHz`, in order to plot the correct contours, the contours associated with `1 GHz`, you must specify the second harmonic of the fundamental, `2*500 MHz = 1 GHz`.

The *Select* cyclic field displays *Single Power/Refl Terminal* and the prompt at the bottom of the form displays

```
> Select Instance Terminal on Schematic.
```

The bottom of the Direct Plot form looks like the following.

10. Follow the prompt at the bottom of the Direct Plot form,

```
> Select Instance Terminal on Schematic
```

by clicking, in the schematic, the terminal at the port adapter as shown in Figure A-30.

Figure A-30 Terminal at Port Adapter

Note: Subsequent instructions call this terminal the *terminal at the port adapter*.

After you select the terminal, the plot for the load contours appears in the waveform window. Figure A-30 shows the entire plot.

Complete Load Contour Plot

A small x appears at the maximum power point, which in this case lies near the center of the smallest (yellow) constant power contour. See Figure A-31 on page 2035 for a detailed view.

Figure A-31, an enlarged area taken from Figure A-30, which more clearly shows the X that marks the maximum power point (inside the yellow contour).

Figure A-31 Enlarged View Showing Max Power Point (X)

If you place the cursor on the X, you can read the following information across the top of the waveform window.

```
Real:2.479      Imag:-4.737      Freq:-360     p="1.04m";Constant Power Contours
```

This indicates that a normalized load impedance of about 2.48 - j4.737 dissipates the most power. The X appears at the maximum power point.

Adding the Reflection Contours to the Plot

You might want to maximize load power subject to a constraint on the magnitude of the amplifier's input reflection coefficient. Such a constraint can prevent unstable interactions with the preceding stage.

You can overlay load-pull contours with contours of constant input reflection coefficient magnitude. The optimal load corresponds to the reflection coefficient that lies on both the

largest power load-pull contour and on a constant input reflection coefficient contour that is within the constraint. Here *largest power* means the contour corresponding to the largest amount of power delivered to the load.

To use the topographical map analogy again, this is like overlaying constant elevation contours with constant temperature contours. The optimum objective is similar to trying to find the highest point on the mountain where the temperature is above 60 degrees.

1. If necessary, in the ADE window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.

2. Select *Append* in the *Plotting Mode* cyclic field.

3. Highlight *pss* for *Analysis*.

4. Highlight *Reflection Contours* for *Function*.

   The top of the completed form looks like this.

5. Type `9` for *Number of Contours*, if necessary.
6. Type `50.0` for *Reference Resistance*, if necessary.
7. Highlight *1 1G* for *Output Harmonic*.
8. In the *Select* cyclic field, select *Separate Refl and RefRefl Terminals*. The prompt at the bottom of the form changes to

   ```
   > Select Reflection Instance Terminal on Schematic
   ```

   The bottom of the completed form looks like this.

9. Following the prompt at the bottom of the Direct Plot form,

   ```
   > Select Reflection Instance Terminal on Schematic
   ```

   in the Schematic window, click the terminal at port 5, as shown in Figure A-32.

Figure A-32 Terminal at Port 5

Note: Subsequent instructions call this terminal the *terminal at Port 5*.

The prompt on the bottom of the Direct Plot form is

```
> Select Reflection Instance Terminal on Schematic....
```

Follow the prompt. In the Schematic window, click the *terminal at the port adapter* (See A-33).

Figure A-33 Terminal at Port Adapter

The plot for reflection contours is added to the waveform window.

Assume that, for stability, the input reflection coefficient should be less than or equal to 0.218. For the curves shown above, the optimal normalized load impedance is approximately

3.908 -j5.073. Assuming you were free to draw smaller constant power contours, the true constrained optimal power point would occur when the constant power contour is just tangent to the constant input reflection coefficient contour of 0.218.

In the Direct Plot Form for *Reflection Contours*, *Separate Refl & RefRefl Terminals* is selected to measure *Reflection Contours* as contours of constant reflection (magnitude) as seen from the power amplifier's input (measured at *port 5,* the input port) with respect to the reflection of the power amplifier's load (measured at the input to the port adapter.)

In this case,

- In response to the first prompt,

  ```
  > Select Reflection Instance Terminal on Schematic
  ```

  Select the input to the power amplifier, the terminal of *port 5* in the schematic. (See Figure A-32 on page 2038)

- In response to the second prompt

  ```
  > Select Reflection Instance Terminal on Schematic
  ```

  Select the output of the power amplifier, the input terminal to the *port adapter* in the schematic. (See Figure A-33 on page 2038)

So the *port adapter's* reflection (both `mag` and `phase`) is swept, which is the same as sweeping the power amplifier's load. In the Smith chart you see input reflection plots as seen by *port 5*.

The reflection contour plot shows the effect of the power amplifier's load on the input reflection while the power contour plot shows the effect of the power amplifiers's load on the power gain.

For reflection contours, the following terminal selections apply.

1. For *Single Refl/RefRefl Terminal* select one terminal.

   One reflection coefficient of that terminal is computed as

   $$gamma1 = \frac{Z - Z0}{Z + Z0}$$

   where *Z* is the large signal impedance at the fundamental.

   The resulting Smith chart plots contours where `gamma1` is constant.

2. For *Single Refl/RefRefl Term and ref Term* select two terminals.

This is the differential case for number 1. The different voltages of the two nets are used in the large signal impedance calculation.

3. For *Separate Refl and RefRefl Terminals* select two terminals

   Two reflection coefficients for each terminal are computed as

$$gamma1 = \frac{Z1 - Z0}{Z1 + Z0}$$

$$gamma2 = \frac{Z2 - Z0}{Z2 + Z0}$$

   where *Z1* and *Z2* are the large signal impedance for each terminal at the fundamental.

   The resulting Smith chart plots gamma2 and contours where `gamma1` is constant.

4. For + - *Refl and* + - *Ref Refl Terminals* select 4 terminals

   This is the differential case for number 3.

For power contours, the following terminal selections apply.

1. For *Single Power/Ref Terminals* select one terminal.

   The power (*P1*) and the reflection coefficient (`gamma1`) of that terminal are computed. The resulting Smith chart plots `gamma1`and contours where `P1` is constant.

2. For *Single Power/Refl Term and ref Term*

   This is the differential case for number 1.

3. For *Separate Power and Refl Terminals* select 2 terminals

   The power of the first terminal (*P1*) and the reflection coefficients of the second terminal (`gamma2`) are computed. The resulting Smith chart plots `gamma2` and contours where *P1* is constant.

4. For + - *Power and* + - *Refl Terminals*

   This is the differential case for the third case, *Separate Refl and RefRefl Terminals*.

Are Constant Power Contours the Same as Constant Gain Contours?

In general, *constant power contours* are not the same as *constant power gain contours*. The simulator does not maintain any input impedance match while it sweeps the load to generate load-pull contours. The resulting contours are not constant power gain contours, but are only constant power contours. However, if the input reflection coefficient and input power

do not change much over the load sweep range, the generated contours are good approximations of constant gain contours. By plotting constant power contours in the input reflection coefficient plane, you can see how both change when you sweep load reflection coefficient. If the input power does not vary with load, the constant power contours are also constant power gain contours. If the input power does vary with the load, the constant power contours tell you how much you have to change the input matching network during the load sweep to maintain a constant input power.

1. In the Direct Plot form, do the following:
   - **a.** Highlight *Replace* for *Plot Mode*.
   - **b.** Highlight *Power Contours for Function*.
   - **c.** Highlight *Single Power/Ref Terminal*.
   - **d.** Highlight *Magnitude* for *Power Modifier*.
   - **e.** Type `9` for *Number of Contours*, if necessary.
   - **f.** Type `50.0` for *Reference Resistance*, if necessary.
   - **g.** Highlight 1 G for *Output Harmonic*.
2. In the Schematic window, click the terminal at Port 5 (See Figure A-32 on page 2038).
3. The plot of input reflection coefficients appears as in Figure Figure A-34 on page 2043.

Figure A-34 Input Reflection Coefficients Generated by Sweeping Load

4. In the waveform window, choose *Zoom – ZoomIn* and then click and drag with the mouse to form a rectangle that includes the area you want to see.The magnified view is shown in Figure A-34.

Magnified View of Input Reflection Coefficients

The real part of the input impedance varies from about 1.3 to 1.6 in normalized units and the imaginary part varies from 0.15 to 0.4 in normalized units. The input power varies from 927 nW to 967 nW. You find these numbers by placing the cursor on the end points of the outside contours. You read the impedance and power/reflection for that cursor location at the top of the window.

The numbers vary slightly as the load is swept. The objective here is to determine how much the input reflection coefficient and the input power vary as the load varies. If the input quantities vary significantly, then the generated load pull contours do not coincide with the constant power gain contours because the input power does not remain constant. The amount of variation in the input reflection coefficient tells you by how much you would have

to re-tune the input matching network to maintain constant input power as you sweep the load. You have to decide whether this much variation matters.

Moving to Differential Mode

Up until now, the examples used in this chapter have been single-ended. Because many circuits are differential, Figure A-35 shows a simple way to modify the example circuit to make it differential.

Figure A-35 Simple Differential Circuit

If your circuit has one or more differential ports, you can translate between single-ended and differential circuitry using linear-dependent controlled sources. The extra circuitry added to Figure A-35 shows a modified version of the circuit where the output has been modified to make it differential. The transition from single-ended to differential output was made using the controlled sources *F0* and *E1* (circled in Figure A-35).

The controlled sources, *cccs* and *vcvs*, are available in *analogLib*. The gain of *cccs* is -1 to be consistent with the polarity of the *vcvs*. The *vcvs* (*E1*) is the *Name of voltage source* parameter in the *cccs*.

Using S-parameter Input Files

In this example, you create an S-parameter data file then you enter it back into the original circuit. The S-parameter data file replaces that portion of the circuit originally used to create the S-parameters. To enter tabulated S-parameters from some other source, record the data

in the format shown in the S-parameter data file and select the appropriate numerical options in the *nport*.

This example shows you how to

- Set up and simulate the *sparamfirst* circuit and create an S-parameter data file (`sparam.practice`) for the *sparamfirst* schematic
- Add an *n2port* device in the frequency domain to a modified copy of the *sparamfirst* schematic.
- Run another simulation of the modified *sparamfirst* circuit where the *n2port* component reads in the S-parameter data file
- Plot and compare the two sets of data produced.
  - The output of the original branch of the circuit
  - The output of the modified branch of the circuit where the *n2port* component replaced the components in the original branch. The *n2port* reads in the `sparam.practice` S-parameter file you created.

  When you superimpose the two plots, it is clear that the S-parameter file produces the same plot as the original components.

Setting Up the EF_example Schematic for the First Simulation

This example requires that you

- Open the *EF_example* schematic
- Create a new, empty cellview named *sparamfirst*
- Copy part of the *EF_example* schematic and paste it in the *sparamfirst* schematic
- Simulate the new schematic.

  This first simulation generates the S-parameters of a linear time-invariant part of the original circuit (*EF_example*). The S-parameter file is stored so you can use it in the second simulation to import S-parameters.

Opening the EF_example Schematic

1. Open the *EF_example* schematic as described in "Opening the EF_example Circuit in the Schematic Window" on page 1947.

The *EF_example* schematic is the original schematic used in the first Envelope example in this chapter. Leave the *EF_example* schematic open in the background.

Creating and Editing the New Schematic

1. In the CIW, choose *File – New – Cellview.*

   The Create New File form appears.

2. In the New File form, do the following:

   a. Choose *my_rfExamples* for *Library Name,* the editable copy of *rfExamples*.

   Select the editable copy of the *rfExamples* library you created following the instructions in Chapter 3.

   b. In the *Cell Name* field, type *sparamfirst ,* a name for the new view.

   c. In the *View Name* field, type `schematic`.

   d. Choose *Composer-Schematic* for *Tool*.

e. Click *OK*.

After you click *OK,* the new empty Schematic window for *sparamfirst* appears.

3. In the *EF_example* Schematic window, copy the part of the schematic shown in Figure A-35 and paste it into the new Schematic window.

Portion of EF_example Schematic to Copy

a. In the *EF_example* Schematic window, choose *Edit - Copy* and follow the prompts at the bottom of the Schematic window.

b. Following the prompt,

```
> point at object to copy
```

left click and drag to create a box around the part of the *EF_example* circuit indicated in A-35.

The selected *EF_example* components are highlighted in yellow.

c. Following the prompt,

```
> point at reference point for copy
```

click inside the outlined elements.

d. Following the prompt,

```
> point at destination point for copy
```

move the cursor to the *sparamfirst* schematic window and click there. This copies the selected part of the *EF_example* circuit into the empty Schematic window.

The empty Schematic window now contains the portion of the *EF_example* schematic shown in A-35.

e. If necessary, choose *Window - Fit* to center the copied section of the *EF_example* circuit in the *sparamfirst* Schematic window.

4. In the original *EF_example* Schematic window, choose *Window - Close*.

5. The *EF_example* Schematic window closes.

Adding Components to the Schematic

1. In the *sparamfirst* Schematic window, choose *Add – Instance.*

The Add Instance form appears.

2. In the Add Instance form, click *Browse*.

The Library Browser – Add Instance form appears.

3. In the Library Browser – Add Instance form, do the following:

   a. Select the *analogLib* and highlight *Show Categories*.

   b. Scroll down and select *Sources* in the *Category* list box.

      The *Category* list box expands to display several subcategories under *Sources*.

   c. Scroll down and select the *Ports* subcategory under *Sources*.

   d. Select *port* for *Cell*.

   e. Select *symbol* for *View*.

      A copy of the port component moves with the cursor into the new Schematic window.

   f. Place two *port* symbols in the *sparamfirst* Schematic window as shown in Figure A-35.

Placement and Port Numbering in the sparamfirst Schematic

4. In the Library Browser – Add Instance form, do the following:

   a. Select *analogLib* for *Library*.

   b. Select *Everything* for *Category*.

   c. Select *gnd* for *Cell*.

   d. Select *symbol* for *View*.

   e. Place two *gnd* components in the Schematic window as shown in Figure A-35. Press *Esc* when you are done.

5. In the Schematic window, choose *Add – Wire (narrow)* and wire up the new components in the *sparamfirst* Schematic window. Press *Esc* when you are done.
6. In the new Schematic window, choose *Edit – Properties – Objects*.
7. For each of the *port* components in the new schematic, do the following in the Edit Object Properties form:
   a. Select the *port*.
   b. Type the appropriate number in the *Port number* field.

      Figure A-35 shows the appropriate number to type. Leave the Port resistance at its default of 50 Ohms.
   c. Click *Apply*.
8. In the new Schematic window, choose *Design – Check and Save*.

Setting Up the sparamfirst Schematic

1. In the *sparamfirst* Schematic window, open the ADE window as described in "Opening the Simulation Window" on page 1949.
2. In the ADE window, set up the model libraries as described in "Setting Up the Model Libraries" on page 1949.

Running the SP Simulation

Run an SP analysis to write out an S-parameter file, `sparam.practice`, that describes the *sparamfirst* circuit.

1. In the ADE window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.
2. In the Choosing Analyses form, do the following:
   a. Highlight *sp* for *Analysis*.
   b. Highlight *Frequency* for *Sweep Variable*.
   c. Highlight *Start – Stop* for *Sweep Range*.
   d. Type `100K` for *Start* and `10G` for *Stop*.
   e. Choose *Automatic* for *Sweep Type*.

   f. Highlight *no* for *Do Noise*.

3. In the Choosing Analyses form, click *Options.*

The S-parameter Options form appears.

4. In the OUTPUT PARAMETERS section of the S-parameter Options form, type the path for the output S-parameter file in the *file* field. Click *Apply*. Click *OK*.

5. In the Choosing Analyses form, click *Apply*. Click *OK*

6. In the ADE window, choose *Simulation – Netlist and Run*.

7. Check the CIW for a message that says the simulation completed successfully.

The sp analysis wrote an S-parameter file, `sparam.practice`, to the directory you indicated.

The S-Parameter File

Use the S-parameter output file, `sparam.practice`, you created as input for the next simulation. You can use a text editor to open and examine the format of the S-parameter output file. You can use this format to create S-parameter files from other sources.

The top of the `sparam.practice` S-parameter file looks like this.

```
; S-parameter data file /hm/belinda/sparam.practice.
; Tue Feb 28 18:10:27 2006
; Number of ports is 2

reference resistance
                 port2 = 50.000000
                 port1 = 50.000000

format freq:  s1:1(real,imag)   s2:1(real,imag)
              s1:2(real,imag)    s2:2(real,imag)

1.00000000e+05:    0.99996,      -0.00628293      4.02664e-05,      0.00628293
               4.02664e-05,      0.00628293         0.99996,      -0.00634576
1.25892541e+05:    0.999936,     -0.00790956      6.38165e-05,      0.00790956
```

```
                 6.38165e-05,      0.00790956         0.999936,     -0.00798866
1.58489319e+05:    0.999899,     -0.00995718      0.000101139,      0.00995717
                 0.000101139,       0.00995717         0.999898,     -0.0100567
```

Setting Up and Running the Second sp Simulation

In this simulation, use the S-parameter output file, `sparam.practice`, from the first simulation as input to the second simulation. The S-parameter file models the components in the `sparamfirst` Schematic.

1. If necessary, open the `sparamfirst` schematic.

2. Add components to the *sparamfirst* schematic you created for the SP analysis as shown in Figure A-36.

Figure A-36 Wired sparamfirst Schematic

The components to add are described in the following table. See "Adding Components to the Schematic" on page 2050 for information on adding components to a schematic.

Table A-4 Components to Add to the sparamfirst Schematic

Library	Category	Subcategory	Cell	View	Number to Add
analogLib	Sources	Ports	n2port	symbol	1
analogLib	Sources	Ports	port	symbol	2
analogLib	Everything	None	gnd	symbol	4

3. Wire the components as shown in Figure A-36.
4. Edit the properties on the two new ports to assign port numbers 3 and 4 as shown in Figure A-36. Then edit the properties on the new *n2port*.
5. In the Schematic window select the *port* on the lower left.
6. In the Schematic window, choose *Edit – Properties – Objects.*
7. In the Edit Object Properties form, do the following:
   - **a.** Type 3 in the *Port number* field.

     Figure A-36 on page 2056 shows the appropriate number to type. Leave the *Port resistance* at the default of 50 Ohms.
   - **b.** Click *Apply*.
   - **c.** In the Schematic window, select the *port* on the lower right.
   - **d.** Type 4 in the *Port number* field.
   - **e.** Leave the *Port resistance* at the default of 50 Ohms.
   - **f.** Click *Apply*.
8. In the Schematic window, select the *n2port*.

   The Edit Object Properties form changes to display information for the *n2port*.
9. In the Edit Object Properties form, do the following and click *OK*:
   - **a.** In the *S-parameter data file* field, type the absolute path to the S-parameter file you created in the first simulation.

     `/hm/belinda/sparam.practice`
   - **b.** Choose *rational* for *Interpolation method*.

     The form changes to let you add additional information.
   - **c.** Type `.001` for *Relative error*.
   - **d.** Type $1e^{-6}$ for *Absolute error*.
   - **e.** Type `6` for *Rational order*.
   - **f.** Select *no* for *Thermal Noise*.
   - **g.** Select *spectre* for *S-parameter data format*.

The completed Edit Object Properties form for the *n2port* looks like this.

CDF Parameter	Value	Display
S-parameter data file	belinda/sparam.practice	off
Multiplier		off
Scale factor		off
Interpolation method	rational	off
Relative error	.001	off
Absolute error	1e-6	off
ROM data file		off
Rational order	6	off
No. of Harmonics for PSS		off
Thermal Noise	no	off
Use smooth data windowing		off
S-parameter data format	spectre	off
Thermal noise model		off

10. In the Schematic window, choose *Design – Check and Save*.
11. In the ADE window, choose *Simulation – Netlist and Run*.

Plotting Results

1. In the ADE window, choose *Results – Direct Plot – Main Form.*

   The Direct Plot form appears.
2. In the Direct Plot form, choose *Replace* for *Plotting Mode*.
3. Highlight *sp* for *Analysis*.
4. Highlight *SP* for *Function*.

5. Highlight *Z-Smith* for *Plot Type*.

The Direct Plot form looks like the following.

a. Click *S11*.

The plot for S11 appears in the waveform window.

6. In the Direct Plot form, do the following:

a. Choose *Append* for *Plot Mode*.

b. Click *S33*.

The plot for `S33` is appended to the `S11` plot. The two plots lie one on top of the other which shows that the two plots are identical. Thus the results produced by the first simulation are the same as those produced by the second simulation which used the S-parameter input file.

You can also check other plots for equivalency. For example, you can do the following to plot S21 and S43.

1. Choose *Replace* for *Plotting Mode.*

2. Plot S21 as described in "Plotting Results" on page 2058.

3. Plot S43 as follows.

a. Choose *Append* for *Plotting Mode*.

b. Select *4* in the first cyclic field to the right of *S*.

c. Select *3* in the second cyclic field to the right of *S*.

d. Click *S* to create the plot.

Figure A-4 is the plot produced by appending `S431` to the `S21` plot.

Plot Showing S21 and S43

Using the S-parameter Input File with a SpectreRF Envlp Analysis

Use the `sparam.practice` S-parameter format data file as input to an Envelope analysis. Again, it compares results of an *ENVLP* analysis using the S-parameter file to results of an *ENVLP* analysis using the original components.

Setting Up the Schematic

1. Open the *EF_example* schematic. Choose *File – Open* in the CIW.
2. Open the *sparamfirst* schematic and move it to the background.

Figure A-37 The sparamfirst Schematic

3. In the *EF_example* schematic. choose *Design – Save As* in the Schematic window.

   The Save As form appears.
4. In the Save As form, do the following:

a. In the *Library Name* field, type the name of your local, editable copy of the *rfExamples* library.

b. In the *Cell Name* field, type *EF_example_copy,* the name for the copy of the *EF_example* schematic.

c. Click *OK*.

You now have a copy of the *EF_example* schematic called *EF_example_copy*.

In the *EF_example* Schematic window., choose *Window – Close* to close the *EF_example* schematic.

5. Open the *EF_example_copy* schematic.

6. In the *EF_example_copy* Schematic, copy everything to the right of the adder and paste the copy below the original.

Important

Be sure to remove the *RFOUT* label from the duplicate branch or the two branches are shorted together at their outputs.

7. To delete RFOUT from the bottom branch, choose *Edit – Delete,* highlight *RFOUT* and then press *Esc*.

8. In the lower branch of the *EF_example_copy* schematic, delete the circuitry you used to create the S-parameter file in the *sparamfirst* schematic, see Figure A-38.

Figure A-38 Components to Delete from the EF_example_copy Schematic

9. Choose *Edit – Delete* and follow the prompts at the bottom of the *EF_example_copy* schematic window.

10. Following the prompt,

    ```
    > point at object to delete
    ```

    left click and drag to create a box around the indicated part of the *EF_example_copy* circuit.

    The selected *EF_example_copy* components are deleted.

11. Press *Esc* to stop deleting.

12. Replace the deleted components in the *EF_example_copy* schematic with the *n2port* component and the two attached *gnd* (ground) components from the *sparamfirst* schematic.The *n2port* and *gnd* components are shown in Figure A-37 on page 2063.

    a. In the *sparamfirst* schematic.window, choose *Edit – Copy* to copy the *n2port* and the two attached *gnd* components. Follow the prompts at the bottom of the *sparamfirst* schematic window.

       Note: (Close the *sparamfirst* schematic window when you are done.)

    b. As you move the cursor into the *EF_example_copy* Schematic window, a copy of the *n2port* and *gnd* components moves with the cursor. Place the components in the *EF_example_copy* Schematic window as shown in Figure A-38

13. In the Schematic window, choose *Add – Wire (narrow)* and wire up the new components in the *EF_example_copy* Schematic window. Press *Esc* when you are done.

The EF_example_copy Schematic with n2port Component in Place

14. In the Schematic window, select the *n2port* component and choose *Edit – Properties – Object*. Then verify that the *n2port* component has the following properties.

a. The absolute path to the S-parameter data file displays in the S-parameter data file field. In this example,

`/hm/belinda/sparam.practice`

b. *Interpolation method* is *Rational*.

c. *Relative error* is `.001`.

d. *Absolute error* is $1e^{-6}$.

e. *Rational order* is `6`.

f. *Thermal Noise* is *no*.

g. *S-parameter data format* is *spectre*.

The completed Edit Object Properties form for the *n2port* looks like this.

CDF Parameter	Value	Display
S-parameter data file	belinda/sparam.practice	off
Multiplier		off
Scale factor		off
Interpolation method	rational	off
Relative error	.001	off
Absolute error	1e-6	off
ROM data file		off
Rational order	6	off
No. of Harmonics for PSS		off
Thermal Noise	no	off
Use smooth data windowing		off
S-parameter data format	spectre	off
Thermal noise model		off

15. In the Schematic window, choose *Design – Check and Save*.

Setting Up and Running the Simulation

Set up and run an *ENVLP* analysis as described in the sections listed here.

- "Opening the Simulation Window" on page 1949
- "Setting Up the Model Libraries" on page 1949
- "Editing PORT0 and PORT1 in the EF_example Schematic" on page 1950
- "Setting Up an Envelope Analysis" on page 1952.

When you set up the *ENVLP* analysis, use 30u for the *Stop Time* value.

The completed *ENVLP* Choosing Analyses form looks like this.

➤ In the ADE window, choose *Simulation – Netlist and Run*.

Check the CIW for a message that says the simulation completed successfully.

Displaying the ENVLP Following Results

1. In the ADE window, choose *Results – Direct Plot – envlp*.

   The Direct Plot form appears.

2. In the Direct Plot form, choose *Replace* for *Plotting Mode*.
3. Highlight *envlp* for *analysis.*
4. Highlight *Voltage* for *Function*.

The top of the form looks like this.

5. Highlight *Harmonic Time* for *Sweep*.
6. Highlight *Real* for *Modifier*.
7. Choose *1* for *Harmonic Number*.

The bottom of the form looks like this.

The prompt on the bottom of the Direct Plot form is

```
> Select Net on Schematic....
```

8. Follow the prompt. In the Schematic window, click one of the output nodes as shown in "Output Nodes" on page 2072.
9. Change the *Plot Mode* to *Append.*
10. Click the other output node as shown in "Output Nodes" on page 2072.

    You can also confirm that the imaginary parts of the waveforms match.

Figure A-39 Output Nodes

The coincident plots in the waveform window show that the output is the same at both nodes. When the S-parameters are correct, the *n2port* device accurately simulates the circuitry represented by the S-parameters.

Measuring AM and PM Conversion with Modulated PAC, AC and PXF Analyses

This section describes AM and PM small signal characterization in SpectreRF. AM/PM conversion measurements allow you to investigate the amplitude and phase characterization of RF circuits. The modulated PAC, AC and PXF analyses calculate conversion gain and other characteristics between AM, PM, and SSB sources and AM, PM, and SSB outputs.

AM/PM conversion computes transfer functions and gain measurements involving AM and PM inputs and outputs. In general, there are three possible types of inputs and outputs for which you might want to compute transfer functions.

- Unmodulated or single sideband (SSB)
- Amplitude modulated sinusoids (AM)
- Phase modulated sinusoids (PM)

The Modulated Analysis Settings

In total, a modulated analysis can calculate 9 cross conversion metrics from 3 types of inputs (AM, PM and SSB) to 3 types of outputs (AM, PM and SSB). Use the *Modulated Analysis* section in the PAC and PXF Choosing Analyses forms to set up your analyses.

- *Input Type* (for PAC analysis) and *Output Type* (for PXF analysis) allow you to choose whether to measure all 9 modulated conversions (choose *SSB/AM/PM*) or only 3 conversions (choose *SSB*).
- *Output Modulated Harmonic List* (for PAC output modulations) and *Input Modulated Harmonic List* (for PXF modulated sources) specify a vector of harmonic indexes. You can type the indexes separated by spaces or you can choose them from a scrolling list.
- *Input Modulated Harmonic* (for PAC) and *Output Modulated Harmonic* (for PXF) specify a single harmonic index for PAC input source modulation or for PXF output modulation. You can type the index or you can choose it from a scrolling list. This choice appears when you select the *SSB/AM/PM Input* or *Output Type*.

  For *SSB Input Type* (PAC) or *Output Type* (PXF), specify an *Output Upper Sideband* or *Input Upper Sideband.* You can type the sideband index or you can choose it from a scrolling list.
- For modulated PXF analysis, select the *Output Probe Instance* or the *Positive* and *Negative voltage Output Nodes* from the schematic.

This example illustrates how to

- Create the *EF_AMP* schematic, a modified copy of the *EF_example* schematic.
- Save a copy of the *EF_AMP* schematic in the *my_rfExamples* library. You use this schematic for other examples later.
- Set up and run the necessary PSS, modulated PAC and modulated PXF analyses.
- Compute transfer functions and gain measurements and display the resulting information with the Direct Plot form.

After you run the simulation, use the Direct Plot form to plot the modulated analysis results.

Creating the EF_AMP Circuit

Create a modified copy of the *EF_example* schematic, *EF_AMP* and save the *EF_AMP* schematic in the *my_rfExamples* library you created. You use the *EF_AMP* schematic in the Jitter example.

Create a New Empty Schematic Window

Open the *EF_example* schematic in a schematic window. Open another empty schematic window, name it EF_AMP and copy the *EF_example* schematic into the EF_AMP schematic window.

1. In the CIW, choose *File – New – Cellview*.

The Create New File form appears.

2. In the Create New File form, do the following:

   a. Choose *my_rfExamples* for *Library Name*.

      Select *my_rfExamples*, the editable copy of the *rfExamples* library you created following the instructions in Chapter 3.

      In the *Cell Name* field, enter *EF_AMP*.

   b. In the *View Name* field, enter *schematic*.

   c. In the *Tool* cyclic field, select *Composer-Schematic*.

   d. Click *OK*.

A new, empty Schematic window named *EF_AMP* opens.

Open and Copy the EF_example Schematic

Open the *EF_example* schematic and copy it into the empty *EF_AMP* Schematic window. Then edit *EF_AMP* before simulating.

1. In the CIW, choose *File – Open*.

The Open File form appears.

2. In the Open File form, do the following:

   a. Choose *my_rfExamples* for *Library Name*.

   Select the editable copy of the *rfExamples* library you created following the instructions in Chapter 3.

   b. In the *Cell Name* list box, highlight *EF_example*.

   c. Choose *schematic* for *View Name*.

   d. Highlight *edit* for *Mode*.

   e. Click *OK*.

   The Schematic window appears with the *EF_example* schematic.

3. In the *EF_example* Schematic window, choose *Edit - Copy* and follow the prompts.

4. Following the prompt at the bottom of the Schematic window,

   `> point at object to copy`

   left click and drag to create a box around the entire *EF_example* circuit.

   The *EF_example* components are highlighted.

5. Following the prompt,

   `> point at reference point for copy`

   click inside the outlined elements.

6. Following the prompt,

   ```
   > point at destination point for copy,
   ```

   move the cursor to drag a copy of the entire circuit into the *EF_AMP* Schematic window and click there.

   The *EF_AMP* Schematic window now contains a copy of the *EF_example* schematic.

7. In the *EF_AMP* Schematic window, if necessary, choose *Window - Fit* to scale and center the circuit in the Schematic window.

8. In the *EF_example* Schematic window, choose *Window - Close*.

   The *EF_example* Schematic window closes.

Editing the EF_AMP Schematic

In the *EF_AMP* Schematic window, delete components and their associated connecting wires as shown in Figure A-40. The final *EF_AMP* Schematic should look like Figure A-42 on page 2079.

Figure A-40 Components and Wires to Delete from the EF_LoadPull Schematic

Note: If you need assistance with methods for editing the schematic, see the *Virtuoso® Schematic Composer™ User Guide*.

Delete Components and Wires

1. In the *EF_AMP* Schematic window, choose *Edit – Delete.*
2. Click a component or wire to delete it.
3. Press the *Esc* key when you are done deleting.
4. After you delete the components and wires, the EF_AMP schematic looks like Figure A-41.

Figure A-41 EF_AMP Schematic After Deleting Components

5. In the Schematic window, choose *Edit – Move* to move both *PORT 3* and *VCC* as shown in "The Edited EF_AMP Schematic" on page 2079.
6. To move *PORT 3* closer to capacitor *C!,* click and drag to create a rectangle around *PORT 3* and its ground.

   PORT 3 and the *Gnd* are highlighted.
7. Click and drag the components to move them close to capacitor `C1` as shown in "The Edited EF_AMP Schematic" on page 2079.
8. To move *VCC* below and between the two device-level amplifier models, click and drag to create a rectangle around *VCC*.

   VCC is highlighted in yellow.
9. Click and drag the components to move them below and between the two device-level amplifier models as shown in "The Edited EF_AMP Schematic" on page 2079.

10. Choose *Window - Fit* to center the edited schematic in the window.

Wire the Schematic.

1. In the Schematic window, connect *PORT3* to Capacitor *C1*.

a. In the Schematic window, choose *Add - Wire (Narrow)*.

b. Click the terminal on *PORT 3* then click the terminal on *C1*.

c. Press the Esc key to stop wiring.

2. Choose *Window -- Fit* to center the edited schematic in the window.

The edited schematic looks like the one in Figure A-42.

Figure A-42 The Edited EF_AMP Schematic

Edit CDF Properties for PORT3 and RFOUT

In this section, you edit CDF properties for both *PORT 3* and *RF_OUT*.

1. In the Schematic window, select *PORT 3*.

2. Choose *Edit – Properties – Objects*.

The Edit Object Properties form appears with information for *PORT 3* displayed.

3. In the Edit Object Properties form, do the following.

a. Type *fin* for *Frequency 1*.

The completed form looks like this.

b. Highlight *Display small signal params*.

The small signal parameters section of the form opens up.

c. Type *pac_dbm* for *PAC Magnitude (dBm)*.

The completed form looks like this.

4. Click *Apply* in the Edit Object Properties form.
5. In the Schematic window, select *RF_OUT*.

   The Edit Object Properties form changes to display data for *RF_OUT*.
6. In the Edit Object Properties form, do the following.
   a. Highlight *Display small signal params*.

      The small signal parameters section of the form opens up.
   b. Type *pxfout_mag* for *XF Magnitude*.

The completed form looks like this.

7. Click *OK* in the Edit Object Properties form.
8. In the Schematic window, choose *Design - Check and Save*.

Setting Up the EF_AMP Circuit

Before you start, perform the setup procedures described in Chapter 3.

1. Set up and run the necessary PSS, modulated PAC and modulated PXF analyses on the *EF_AMP* schematic.
2. Compute transfer functions and gain measurements and display the resulting information with the Direct Plot form.

Opening the Simulation Window for the EF_AMP Circuit

1. In the EF_AMP Schematic window, choose *Tools – Analog Environment*.

The ADE window opens.

2. In the ADE window, choose *Variables – Copy From Cellview*.

   The variable names display in the Design Variables area of the ADE window.

Edit the Variable Values

➤ Following the directions in for setting up the examples and edit the variables to have the following values.

pxfout_mag	1
pac_dbm	0
fin	1G

Check the Design Variables area in the ADE window to display the variable values.

You can also choose *Tools – Analog Environment – Simulation* in the CIW to open the ADE window without opening the design. You can open the design later by choosing *Setup – Design* in the ADE window and choosing *EF_example* in the Choosing Design form.

Setting up the Model Libraries for the EF_AMP Circuit

1. In the ADE window, choose *Setup - Model Libraries.*

   The Model Library Setup form appears.

2. In the *Model Library File* field, type the full path to the model file including the file name. *CDSHOME* is the installation directory for the Cadence software.

   ```
   <CDSHOME>/tools/dfII/samples/artist/models/spectre/rfModels.scs
   ```

3. Click *Add*.

The Model Library Setup form looks like the following.

4. Click *OK*.

Selecting Outputs To Save

1. In the ADE window, choose *Outputs – To Be Saved – Select on Schematic*.
2. In the Schematic window, click the terminals that are circled in Figure A-43.

Figure A-43 Outputs to Save in the EF_AMP Schematic

After you click a terminal, it is circled in the schematic. The selected outputs are also displayed in the Schematic window Outputs area as shown in Figure A-44.

Figure A-44 Outputs Area in Simulation Window

Outputs

#	Name/Signal/Expr	Value	Plot	Save	March
1	PORT3/PLUS		no	yes	no
2	VCC/PLUS		no	yes	no
3	RF_OUT/PLUS		no	yes	no

Plotting mode: Replace

3. In the ADE window, choose *Outputs-Save All*.

4. The Save Options form displays.

5. Set the following values

save	*allpub*
currents	*selected*
useprobes	*yes*

6. Click *OK*.

Setting Up and Running the PSS, PAC Modulated, and PXF Modulated Analyses

➤ In the ADE window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

Setting up the PSS Analysis

1. In the Choosing Analyses form, highlight *pss*.

   The form changes to display options for PSS simulation.

2. Highlight the *Auto Calculate* button to automatically calculate and enter the value `1G` in the *Beat Frequency* field.

3. Choose *Number of harmonics* for *Output harmonics* and type `10` in the adjacent field.

The top of the PSS analysis form looks like this.

4. Highlight *moderate* for Accuracy Defaults *(errpreset)*.
5. Verify that *Enabled* is highlighted in the PSS Choosing Analyses form.

The bottom of the PSS Choosing Analyses form looks like this.

6. Click the *Options* button to display the Options form for the PSS analysis.
7. In the Options form, scroll to the *Output Parameters* section, highlight *all* for *save.* The *Output Parameters* section of the Options form looks like the following.

8. In the PSS Options form, click *OK*.
9. In the PSS Analysis form, click *Apply*.

Setting up the PAC Modulated Analysis

1. In the Choosing Analyses form, highlight *pac*.

   The form changes to display options for PAC simulation.

2. For *Sweeptype*, select *Relative* in the cyclic field and type `1` in the *Relative Harmonic* field.

3. For *Frequency Sweep Range*, select *Start-Stop* in the cyclic field. Type `100` in the *Start* field and `400M` in the *Stop* field.

4. For *Sweep Type*, select *Logarithmic* in the cyclic field, highlight *Points Per Decade* and type `20` in the field.

   The top of the PAC analysis form looks like this.

5. For *Sidebands*, select *Maximum sideband* in the cyclic field. Type `3` in the field.

6. In the *Specialized Analyses* cyclic field select *Modulated* to open the modulated analysis section of the PAC Choosing Analyses form.

7. For *Input Type* choose *SSB/AM/PM* in the cyclic field.

8. For *Output Modulated Harmonic List*, press *Choose* to display the Choose Harmonic form.

9. Scroll the list of harmonics and select harmonics with indexes 1, 2 and 3. Click to highlight the first harmonic. Press and hold down the *Control* key while you click to select harmonics 2 and 3. Then click OK.

`1`, `2`, and `3` display in the *Output Modulated Harmonic List* field.

You can also simply type the values, separated by spaces, in the *Output Modulated Harmonic List* field.

10. For *Input Modulated Harmonic*, press *Choose* to display the Choose Harmonic form.

11. Scroll the list of harmonics and select the harmonic with index of 1. Click to highlight the harmonic 1. Then click *OK*.

`1` displays in the *Input Modulated Harmonic* field.

You can also simply type the value in the *Input Modulated Harmonic* field.

12. Verify that *Enabled* is highlighted in the PAC Choosing Analyses form and click *Apply.*

The bottom of the PAC analysis form looks like this.

13. Click the *Options* button to display the Options form for the PAC analysis.

14. In the *Output Parameters* section of the Options form, highlight, *out* for *freqaxis* and *all* for *save.*

The Output *Parameters* section of the Options form looks like the following.

15. In the PAC Options form, click *OK*.

16. In the PAC Analysis form, click *Apply*.

Setting up the PXF Modulated Analysis

1. In the Choosing Analyses form, highlight *pxf*.

The form changes to display options for PXF analysis.

2. For *Sweeptype*, select *relative* in the cyclic field and type `1` in the *Relative Harmonic* field.

3. For *Frequency Sweep Range*, select *Start-Stop* in the cyclic field. Type `100` in the *Start* field and `400M` in the *Stop* field.

4. For *Sweep Type*, select *Logarithmic* in the cyclic field, highlight *Points Per Decade* and type `20` in the field.

   The top of the PXF analysis form looks like this.

5. For *Sidebands*, select *Maximum sideband* in the cyclic field and type `3` in the field.

6. For *Output*, highlight *probe*. Click *Select* and follow the prompt at the bottom of the Schematic window

   ```
   Select output probe instance...
   ```

   and select the RF port on the right side of the schematic.

`/RF_OUT` displays in the *Output Probe Instance* field.

7. Highlight *Modulated Analysis* to open the modulated analysis section of the PXF Choosing Analyses form.

8. For *Output Type* choose *SSB/AM/PM* in the cyclic field.

9. For *Input Modulated Harmonic List*, press *Choose* to display the Choose Harmonic form.

10. Scroll the list of harmonics and select the harmonic with index of 1. Click to highlight the harmonic 1. Then click OK.

    `1` displays in the *Input Modulated Harmonic List* field.

    You can also simply type the value in the *Input Modulated Harmonic List* field.

11. For *Output Modulated Harmonic*, press *Choose* to display the Choose Harmonic form.

12. Scroll the list of harmonics and select the harmonic with index of 1. Click to highlight the harmonic 1. Then click OK.

`1` displays in the *Output Modulated Harmonic* field.

You can also simply type the value in the *Input Modulated Harmonic* field.

The bottom of the PXF analysis form looks like this.

13. Verify that *Enabled* is highlighted in the PXF Choosing Analyses form.

14. Click the *Options* button to display the Options form for the PXF analysis.

15. In the *Output Parameters* section of the Options form *in* for *freqaxis* and *all* for *save. sources* is already highlighted for *stimuli* because it is the default.

The Output *Parameters* section of the Options form looks like the following.

16. Click OK in the PXF Options form.

17. Click OK in the PXF Choosing Analyses form.

The Choosing Analyses form closes. The ADE window displays the analysis information you have set up.

Running the Simulations

1. To run the simulations, choose *Simulation – Netlist and Run* in the ADE window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting and Calculating PAC Modulated Results

1. Choose *Results – Direct Plot – Main Form* in the ADE window.

   The Direct Plot form appears

2. In the Direct Plot form, highlight *Replace* for *Plot Mode*.
3. Highlight *pac modulated* for *Analysis*.

   The form changes to reflect the modulated PAC analysis.
4. In the *Input* cyclic field, select *AM*.
5. In the *Modulated Input Harmonic* cyclic field, select *1G 1*. This is the only available choice.

   The *USB* field displays `1G -- 1.4G`.
6. In the *Output* cyclic field, select *AM*.
7. In the *Modulated Output Harmonic* cyclic field, select *1G 1*.

   The *USB* field displays `1G -- 1.4G`.
8. Highlight *Voltage* for *Function*.
9. Highlight *peak* for *Signal Level*.
10. Highlight *dB20* for *Modifier*.

11. The Direct Plot form displays as follows.

12. Select *Net* in the *Select* cyclic field. Follow the prompt at the bottom of the form

`Select Net on schematic`

Select the net labeled *RFOUT* on the right side of the schematic.

The waveform looks as follows.

To add AM/PM to the plot, make the following changes in the Direct Plot form.

1. Change Plot Mode to Append.
2. Change the *Output* cyclic field to *PM*.
3. In the *Modulated Output Harmonic* cyclic field, select *1G 1*.

   The *USB* field displays `1G -- 1.4G`.

The Direct Plot form changes as follows.

4. Follow the prompt at the bottom of the form

   `Select Net on schematic`

   Select the net labeled *RFOUT* again.

   The waveform looks as follows.

Plotting and Calculating PXF Modulated Results

1. If necessary, choose *Results – Direct Plot – Main Form* in the ADE window.

   The Direct Plot form appears.

2. In the Direct Plot form, highlight *Replace* for *Plot Mode*.

3. Highlight *pxf modulated* for *Analysis*.

4. In the *Input* cyclic field, select *AM*.

5. In the *Modulated Input Harmonic* cyclic field, select *1G 1*. This is the only available choice.

   The *USB* field displays `1G -- 1.4G`.

6. In the *Output* cyclic field, select *AM*.

7. In the *Output Harmonic* cyclic field, select *1G 1*.

   The *USB* field displays `1G -- 1.4G`.

8. Highlight *Voltage Gain* for *Function*.

9. Highlight *dB20* for *Modifier*.

10. The Direct Plot form displays as follows.

11. Follow the prompt at the bottom of the form

```
Select Port or voltage source on schematic
```

Select the input port labeled *PORT3* on the left side of the schematic.

The waveform looks as follows.

To add the VCC voltage source to the plot,

1. Change Plot Mode to Append.
2. Select the voltage source labeled *VCC* in the lower center of the schematic.

The VCC voltage source is added to the plot.

This plot compares two different voltage sources from the schematic.

Measuring Jitter with PSS and Pnoise Analyses

This section describes jitter measurement.

Setting Up the EF_AMP Circuit

Before you start, perform the setup procedures described in Chapter 3.

- If you have a copy of the *EF_AMP* schematic in your *my_rfExamples* library, simply open it. If you do not have a copy of the *EF_AMP* schematic, See "Creating the EF_AMP Circuit" on page 2074 for information on creating the EF_AMP circuit.
- Open the simulation window for the EF_AMP schematic as described in "Opening the Simulation Window for the EF_AMP Circuit" on page 2082.
- Set up the model libraries as described in "Setting up the Model Libraries for the EF_AMP Circuit" on page 2084.

- Select Outputs to save as described in "Selecting Outputs To Save" on page 2085.
- Set up and run the necessary PSS and modulated Pnoise analyses on the *EF_AMP* schematic.
- Compute transfer functions and gain measurements and display the resulting information with the Direct Plot form.

Opening the EF_AMP Circuit in the Schematic Window

1. In the CIW, choose *File – Open*.

   The Open File form appears.

2. In the Open File form, do the following:

   a. Choose *my_rfExamples* for *Library Name*.

      Select the editable copy of the *rfExamples* library you created following the instructions in Chapter 3.

   b. In the *Cell Name* list box, highlight *EF_AMP*.

   c. Choose *schematic* for *View Name*.

   d. Highlight *edit* for *Mode*.

   e. Click *OK*.

The Schematic window appears with the *EF_AMP* schematic.

Opening the Simulation Window for the EF_AMP Circuit

1. In the EF_AMP Schematic window, choose *Tools – Analog Environment*.

The ADE window opens.

Setting Up and Running the PSS and Pnoise Analyses

➤ In the ADE window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

Setting up the PSS Analysis

1. In the Choosing Analyses form, highlight *pss*.

   The form changes to display options for PSS simulation.

2. Highlight the *Auto Calculate* button to automatically calculate and enter the value `1G` in the *Beat Frequency* field.

3. Choose *Number of harmonics* for *Output harmonics* and type `10` in the adjacent field. Output Harmonics are not critical for this analysis.

The top of the PSS analysis form looks like this.

4. Highlight *moderate* for Accuracy Defaults *(errpreset)*.
5. Verify that *Enabled* is highlighted in the PSS Choosing Analyses form.

   The bottom of the PSS Choosing Analyses form looks like this.

6. Click the *Options* button to display the Options form for the PSS analysis.

7. In the Options form, scroll to the *Output Parameters* section, highlight *all* for *save.* The *Output Parameters* section of the Options form looks like the following.

8. In the PSS Options form, click *OK*.
9. In the PSS Analysis form, click *Apply*.

Setting up the Pnoise Jitter Analysis

1. In the Choosing Analyses form, highlight *pnoise*.

   The form changes to display options for pnoise simulation.

2. For *Sweeptype*, select *Relative* in the cyclic field and type `1` in the *Relative Harmonic* field.
3. For *Frequency Sweep Range*, select *Start-Stop* in the cyclic field. Type `10` in the *Start* field and `500M` in the *Stop* field.
4. For *Sweep Type*, select *Logarithmic* in the cyclic field, highlight *Points Per Decade* and type `5` in the field.

The top of the pnoise analysis form looks like this.

5. For *Sidebands*, select *Maximum sideband* in the cyclic field. Type 7 in the field.
6. Choose *voltage* in the *Output* cyclic field.

   The following message displays in the CIW.

```
Since voltage was selected as the output measurement
technique for the pnoise analysis, spectre will NOT
subtract any load resistance contribution from the Noise
Figure calculation. If this is undesirable, please select
probe as the output measurement technique and select a port.
```

7. Click *Select* for the *Positive Output Node*, and then click the *RFOUT* net in the Schematic window.
8. Leave the *Negative Output Node* field empty.

   This field defaults to */gnd!* You can set the *Negative Output Node* to a different value by clicking *Select* and then clicking on the output node in the schematic.
9. Choose *port* in the *Input Source* cyclic field.
10. Click *Select* for the *Input Port Source*, and then click *port 3* in the Schematic window.

11. In the *Reference side-band* cyclic field, select *Enter in field* and type `0` in the field.

12. In the *Noise Type* cyclic field select *Jitter* to open the jitter section of the Pnoise Choosing Analyses form.

The form indicates that you are measuring PM jitter at the output for a driven circuit and that the output signal is */RFOUT*.

13. Set the *Threshold Value* to `0.05`. Select the threshold value based on knowledge of the results of and earlier run of the PSS analysis. Base your selection on knowledge of your circuit.

14. Select *All* in the *Crossing Direction* cyclic field. This captures both up and down crossing points.

Select *Crossing Direction* equal to *all* cautiously because it directly affects the simulation speed. If the output signal has many transitions, the Pnoise analysis is slowed.

15. Verify that *Enabled* is highlighted in the *Pnoise* Choosing Analyses form and click *Apply.*

16. Click the *Options* button to display the Options form for the *Pnoise* analysis.

17. In the *Output Parameters* section of the Options form, highlight *all* for *save*.

The Output *Parameters* section of the Options form looks like the following.

18. In the Pnoise Options form, click *OK*.

19. In the Pnoise Analysis form, click *OK*.

The Choosing Analyses form.closes. The ADE window displays the analysis information you have set up.

Running the Simulations

1. To run the simulations, choose *Simulation – Netlist and Run* in the ADE window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the Jitter Measurement

➤ Choose *Results – Direct Plot – Main Form* in the ADE window.

The Direct Plot form appears.

Plot the Upward Transition and Event Time

In the Direct Plot form, plot the upward transition.

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pnoise jitter* for *Analysis*.

   The form changes to reflect the pnoise jitter analysis.

3. Highlight *Threshold Xing* for *Function*.
4. In the *Signal Level* cyclic field, select *187.3p.*
5. Click *Plot.*

   The up transition and its location display.

Plot the Downward Transition and Event Time

In the Direct Plot form, plot the downward transition.

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pnoise jitter* for *Analysis*.

   The form changes to reflect the pnoise jitter analysis.
3. Highlight *Threshold Xing* for *Function*.
4. In the *Signal Level* cyclic field select *581.5p.*
5. Click *Plot.*

The down transition and its location display.

Plot Edge to Edge Jitter

In the Direct Plot form, plot the edge to edge jitter.

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pnoise jitter* for *Analysis*.
3. Highlight *Jee* for *Function*.

   The form changes to reflect the Function.
4. In the *Event Time* cyclic field, select *187.3p.*
5. Highlight *rms* for *Signal Level*.
6. Highlight *UI* for *Modifier*.

UI, the Unit Interval, plots time jitter scaled by the period.

7. Click *Plot.*

   Edge to edge jitter displays.

Selecting the unit interval, *UI*, selects time jitter scaled by the period. This plots the frequency dependent jitter power spectral density. The plot also indicates the integrated, or total, jitter as a number scaled by the selected modifier, UI in this case.

Jitter at the up and down crossings is very close. The primary difference is the amount of noise at the lower frequency range, which does not matter after integration. In addition, the slope at the crossing, which we are using, is slightly different.

The *Integration Limits* display at the bottom of the Direct Plot form. In this instance the *Start Frequency* is 10 Hz and the *Stop Frequency* is 500M Hz.

Plot Cycle to Cycle Jitter for Both Event Times

In the Direct Plot form, plot cycle to cycle jitter.

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pnoise jitter* for *Analysis*.
3. Highlight *Jcc* for *Function*.

The form changes to reflect the cycle to cycle jitter analysis.

The *Number of cycles [k]* field displays *1.*

4. In the *Event Time* cyclic field select *581.5p.*
5. Highlight *rms* for *Signal Level*.
6. Highlight *Second* for *Modifier*.

   The *Freq. Multiplier* displays as `1` and the *Integration Limits* display as `10` and `500M` Hz.

The Direct Plot form displays as follows.

7. Click *Plot.*

   Cycle to cycle jitter for the 581.5p event time displays.

```
Jcc[Second][k=1]@(10,500M)Hz:rms = 2.4782p
```

In the Direct Plot form, make the following changes.

1. Highlight *Append* for *Plot Mode*.
2. In the *Event Time* cyclic field select *187.3p.*
3. Click *Plot.*

   Cycle to cycle jitter displays for both event times.

```
Jcc[Second][k=1]@(10,500M)Hz:rms = 2.4782p

Jcc[Second][k=1]@(10,500M)Hz:rms = 2.47729p
```

Jitter at the up and down crossings is very close. The primary difference is the amount of noise at the lower frequency range, which does not matter after integration. In addition, the slope at the crossing, which we are using, is slightly different.

The *Integration Limits* display at the bottom of the Direct Plot form. In this instance the *Start Frequency* is 10 Hz and the *Stop Frequency* is 500M Hz.

Plot Jc for Both Event Times

In the Direct Plot form, plot Jc jitter.

1. Highlight *Replace* for *Plot Mode*.

2. Highlight *pnoise jitter* for *Analysis*.
3. Highlight *Jc* for *Function*.

   The form changes to reflect the Jc jitter analysis.

   The *Number of cycles [k]* field displays *1.*
4. In the *Event Time* cyclic field select *187.3p.*
5. Highlight *rms* for *Signal Level*.
6. Highlight *UI* for *Modifier*.

   The *Freq. Multiplier* displays as `1` and the *Integration Limits* display as `10` and `500M` Hz.

The Direct Plot form displays as follows.

7. Click *Plot.*

   Jc displays.

```
Jc[UI][k=1]@(10,500M)Hz:rms = 1.45833m
```

Plot Jc for the 581.5p Event Time

In the Direct Plot form,

1. Highlight *Append* for *Plot Mode*.
2. Change the *Event Time* cyclic field selection to *581.5p.*
3. Click *Plot.*

   Jc display for the second event time is appended to the plot.

```
Jc[UI][k=1]@(10,500M)Hz:rms = 1.45833m

Jc[UI][k=1]@(10,500M)Hz:rms = 1.45894m
```

Jitter at the up and down crossings is very close. The primary difference is the amount of noise at the lower frequency range, which does not matter after integration. In addition, the slope at the crossing, which we are using, is slightly different.

The *Integration Limits* display at the bottom of the Direct Plot form. In this instance the *Start Frequency* is 10 Hz and the *Stop Frequency* is 500M Hz.

Virtuoso Spectre Circuit Simulator and Accelerated Parallel Simulator RF Analysis User Guide

B

Top-down RF Design Methodology

This section describes a methodology for designing analog RF subsystems that fit into larger DSP systems. In particular, this section describes how to use a canonical set of top-down behavioral baseband models for exploring RF architectures in the analog design environment. These models come from the following categories in *rfLib*

- Category *top_dwnBB* contains models of common RF function blocks.
  - The default view of each model is the baseband view (called *veriloga*).
  - All models in this category also have a differential passband view (called *veriloga_PB*).

  The only exceptions are the BB_loss and VGA_BB models. They are meant only for baseband analysis and have no passband view.
- Category *top_dwnPB* contains single-ended passband versions of the baseband models.
- Category *measurement* contains the instrumentation block and baseband signal generator models used to make RF measurements. These elements are not part of an RF architecture. They simply facilitate RF measurements and diagnostics.
- Category *testbenches* contains the test circuits used in this chapter to define the model specifications in the *rfLib*. Where possible, the models are specified in terms of standard RF measurements. The most precise way to describe a measurement is with a test circuit, set up instructions, and sample measurements. The circuits in the testbenches category serve that purpose

These models provide RF designers with a fast method to map RF system specifications into detailed RF designs. The baseband models facilitate fast evaluation of candidate RF architectures specified with DSP metrics. The passband views of the baseband models provide a behavioral system testbench for checking detailed designs of individual RF system components.

Baseband models are behavioral models and all behavioral models sacrifice some accuracy for increased simulation speed. Such sacrifices are usually acceptable in architectural studies because many implementation-dependent details do not affect high level decisions. The

modeling approach taken in top-down design is to simulate only those effects that drive the decisions at hand.

Baseband modeling in no way replaces passband modeling. Some effects missed by equivalent baseband models can affect high level decisions. However, the application of baseband models early and passband models later minimizes the number of slow simulations needed at the lower levels of design abstraction. Baseband models help you to quickly weed out designs that would surely fail tests simulated with passband models.

The success of a modeling approach to top-down design hinges on knowing how the models fit into the design flow and knowing exactly what each modeling parameter means. This chapter has two goals:

- To describe the top-down design flow, from a modeling perspective, for baseband modeling
- To define, as clearly and concisely as possible, the parameters that specify the models

Top-Down Design of RF Systems

Ideally, the digital signal processing, or DSP, team specifies an RF subsystem that fits snugly into the DSP system. A *snug fit* means that

- The specified RF subsystem does exactly what it needs to do at the lowest possible cost
- A functional specification exists that describes requirements for the RF subsystem

In a top down design flow like the one shown in Figure B-1 on page 2132, the DSP team writes a functional specification for an RF subsystem that has not yet been designed. The functional specification describes what the RF subsystem should and should not do without describing how to build the RF subsystem.

The functional specification supplied by the DSP team describes the RF subsystem at the highest possible level of abstraction. At this point behavioral models can be specified rather than measured. This early in the design cycle, the functional specification might well be incomplete or inconsistent. A good top-down design flow can detect problems, such as omissions and inconsistencies in the design, early in the design cycle when they are easier and less expensive to fix. Problems detected later in the design cycle can be much more costly and very difficult to resolve.

Using the functional specification supplied by the DSP team and the behavioral baseband models from *rfLib*, the RF system designers can easily explore RF architectures in the analog design environment. The baseband models facilitate fast evaluation of candidate RF architectures specified with DSP metrics. By switching to the passband views of the

baseband models, the RF design team maps DSP measurements to RF measurements. The passband views of the baseband models provide a behavioral system testbench for checking detailed designs for individual system components.

Using the functional specification and exploring and testing with the baseband and passband models, the RF team can efficiently create a detailed design specification that fully describes the RF subsystem. The design specification can include detailed instructions for building the RF subsystem. At this stage of the design cycle, everything that is known about the design is described at the lowest level of abstraction.

You can now extract behavioral models of a detailed design from simulated measurements. The problem remains that detailed designs usually do not exist until the project is complete. To jump directly to a detailed design implies that the design flow is bottom up. Bottom up flows are important in many projects, but not in all.

DSP and RF designers sometimes have trouble communicating through specifications because the two groups deal with different metrics. For example, DSP designers deal with *bit error rates* and *error vector magnitude statistics* whereas RF designers deal with *intercept points* and *noise figures*.

The new models described here are designed to help RF system designers in two ways.

- First, the baseband models enable RF system designers to quickly explore the RF architectural space, as specified by the DSP metrics, while letting the RF engineers specify the RF system components with RF metrics. The circuit implementations of the RF system components are easier to design and test when the components are specified with standard RF metrics.
- Second, the baseband models can be switched quickly to a passband views where the RF system model can generate end-to-end RF metrics. With end-to-end metrics, the new view can quickly simulate how the detailed design of a particular RF system component affects end-to-end performance.

Use Model for Top Down Design

The following steps outline the RF design process with focus on the early phases of the design as illustrated in Figure B-1 on page 2132.

Figure B-1 The Top Down Design Flow and Use Model

Specify the RF Subsystem in Terms of DSP Metrics

Before you begin the RF subsystem top-down design flow, the DSP design team should completely specify the RF subsystem in terms of DSP metrics. This preliminary step distinguishes the end of the DSP design flow from the beginning of the RF top down design flow and formally hands-off the RF subsystem design specifications to the RF design team.

Explore Candidate Architectures with Baseband Models

The first step in top down RF design is to select a candidate RF architecture. An RF architecture is a set of interconnected RF function blocks that, taken together, describe how

a receiver or transmitter operates. You specify each function block in terms of standard RF metrics such as IP3, gain, bandwidth, and noise figure.

The models you use early in the design cycle as you explore candidate RF architectures must run fast. Each simulations can span hundreds of symbols and each symbol can easily span thousands of RF carrier cycles. The space defined by the function block specifications in each candidate architecture is far too vast to explore with slow, highly precise models. Models used for architectural exploration must quickly reduce the design space down to a size that can be explored with more precision.

The most efficient models for architectural exploration suppress the RF (IF) carrier and are called *baseband* models. In contrast, the *passband* models (introduced in the next step) do not suppress carriers.

You can use the Circuit Optimizer during architectural exploration to help balance the function block specifications for a candidate architecture. For example, you can use the Circuit Optimizer to minimize RMS EVM while ensuring that other measurements stay within acceptable limits.

When you have determined the nominal specifications for each function block, you must put tolerances around them. In the analog world *specifications without tolerances are meaningless***.** The tolerance space is usually explored with some mix of experience, feasibility, a variety of analyses, and outright arbitrary decisions.

There are several ways that you can use the baseband models to test candidate tolerances as well as to determine some tolerances analytically.

One way to test a candidate set of tolerances is to run a Monte Carlo analysis on the metric of interest, like RMS, EVM, or signal-to-noise ratio (SNR).

Another approach is to use the Circuit Optimizer *in reverse*, as a de-optimizer, to determine worst case performance.

Yet another approach is to compute each tolerance separately from a parametric plot. When you have determined all but 2 or 3 tolerances, you can use a multidimensional parametric analysis to map out the performance space and easily identify the remaining tolerances.

Switch to Passband Models and Create an RF System Testbench

The second step in top down RF design is to create a passband view of the system model.

The passband system model performs two functions:

- Confirms that the filters perform as expected.

- Creates an end-to-end testbench that you can use to design the individual function blocks.

For computational efficiency during system passband testing, at any one time, model one or two selected function blocks at the device level. Model all other blocks in the system behaviorally using passband models.

Derive the tolerances by performing the same Monte Carlo analysis or Circuit Optimizer analysis you used to test the function block tolerances in the first step, but this time replace the DSP metrics with end-to-end RF metrics. After you know how far the end-to-end RF metrics can vary, you can insert a device-level model of a function block into the testbench to see how close it drives the system toward violating a derived end-to-end RF specification.

Implement the Function Blocks with Active and Passive Devices

The last step in the top down design process is to implement the function blocks with device models. Because the function blocks are specified in terms of standard RF metrics, you can easily measure the modeling parameters to make sure they fall within the specified tolerances. You can also insert the measured parameters back into the baseband model of the system to check the DSP metrics, or insert the device-level model directly into the passband testbench to check the derived end-to-end RF specifications.

Baseband Modeling

A baseband model for an RF function block simulates what happens to the baseband representation of a signal as it passes through the block. A baseband model maps input baseband signal trajectories into output baseband signal trajectories. If you sample a baseband signal periodically in time and plot the samples in the complex plane, the resulting scatter plot shows the symbol constellation.

Figure B-2 on page 2135 mathematically defines a baseband representation of a passband signal. The *i* and *q* signals are the real and imaginary parts of a complex signal that rides on the two phases of an RF carrier.

Figure B-2 Baseband Representation of a Passband Signal

Baseband models simulate only what happens to the carrier fundamental. Consequently, they only account for non-linearities with odd symmetry. Non-linearities with even symmetry produce no output at the carrier fundamental; they affect the carrier fundamental only when cascaded. For example, a second order non-linearity in one block can create a DC offset at it's output. Upon passing through a subsequent block with another second order non-linearity, the DC offset can mix with the carrier to affect the output carrier fundamental. You should model cascaded blocks producing unfiltered even harmonics as a single baseband model rather than as separate baseband models cascaded together. The non-linearities that most often dominate performance have odd symmetry.

Example Comparing Baseband and Passband Models

The example in this section walks you through an Envelope analysis that illustrates the relationship between baseband and passband models. Following the simulation, you plot the baseband equivalent output signals as computed by the baseband and passband circuits.

The *BB_test_bench* schematic shown in Figure B-3 on page 2136 illustrates the difference between passband and baseband modeling. This circuit is located in the *rfExamples* library.

Figure B-3 The BB_test_bench Schematic

The *BB_test_bench* circuit shows a passband circuit (across the top of the schematic) and its baseband equivalent circuit (across the bottom of the schematic). The same baseband signals drive both circuits but only the passband circuit mixes the baseband signals up to RF. The power amplifier is not matched to either input or output impedances and both impedances are reactive.

Before you start, perform the setup procedures described in Chapter 3.

Opening the Baseband Test Bench Circuit

1. In the CIW, choose *File – Open*.

   The Open File form appears.

2. In the Open File form,

a. Choose *rfExamples* in the *Library Name* cyclic field. (Choose the editable copy of *rfExamples* you created as described in Chapter 3.)

b. Choose *BB_test_bench* in the *Cell Names* list box. Note that the *View Name* cyclic field displays *Schematic*.

c. The completed Open File form appears like the one below.

3. Click *OK*.

The Schematic window for the *BB_test_bench* appears.

4. In the Schematic window, choose *Tools – Analog Environment*.

The ADE window opens.

You can also choose *Tools – Analog Environment – Simulation* in the CIW to open the ADE window without opening the design. You can open the design later by choosing *Setup – Design* in the ADE window and choosing the *BB_test_bench* in the Choosing Design form.

Choosing Simulator Options

1. Choose *Setup – Simulator/Directory/Host* in the ADE window.

   The Simulator/Directory/Host form appears.

2. In the Simulator/Directory/Host form, specify the following:

   a. Choose *spectre* for the *Simulator*.

   b. Type the name of the project directory, if necessary.

   c. Highlight the *local*, *remote*, or *distributed* button to specify the *Host Mode*.

For remote simulation, type the name of the host machine and the remote directory in the appropriate fields. For distributed simulation, fill out the other fields that appear.

The completed form appears like the one below.

3. In the Simulator/Directory/Host form, click *OK*.
4. In the ADE window, choose *Outputs – Save All.*

   The Save Options form appears.

5. In the *Select signals to output (save)* section, be sure *allpub* is highlighted.

6. In the Save Options form, click *OK*.

Setting Up Model Libraries

1. In the ADE window, choose *Setup – Model Libraries*.

   The Model Library Setup form appears.

2. In the *Model Library File* field, type the full path to the model file including the file name, `rfModels.scs`.

3. In the Model Library Setup form, click *Add*.

The completed form appears like the one below.

4. In the Model Library Setup form, click *OK*.
5. In the ADE window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the ADE window to verify whether or not an analysis is enabled.)

Setting Up the Envelope Analysis

1. Choose *Analyses – Choose* in the ADE window.

   The Choosing Analyses form appears.
2. In the Choosing Analyses form, click *envlp.*
   - **a.** Enter *ff* in the *Clock Name* field.
   - **b.** Enter `10u` in the *Stop Time* field.
   - **c.** In the *Output Harmonics* cyclic field, select *Number of harmonics*.
   - **d.** Enter `1` in the *Number of harmonics* field.
   - **e.** Select *moderate* for the Accuracy Defaults.

The correctly filled out form appears below.

3. In the Choosing Analyses form, click *OK*.

Running the Simulation

1. In the ADE window, choose *Simulation – Netlist and Run*.

   The output log file appears and displays information about the simulation as it runs.

Look in the CIW for a message that says the simulation completed successfully.

Plotting the Baseband Equivalent Output Signals

1. In the ADE window, choose *Results-Direct Plot-Main Form*.

   The Direct Plot form appears.

2. In the Direct Plot form, do the following:

   a. Highlight *Replace* for *Plotting Mode*.

   b. Highlight *envlp* for *Analysis*.

   c. Highlight *Voltage* for *Function*.

   d. Highlight *time* for *Sweep*.

   The *Net* selection appears in the *Select* cyclic field and the label *Description: Envelope Voltage vs Time* appears.

   e. Following the message at the bottom of the Direct Plot form

   Select Net on schematic...

   Click the `baseband_I_out` net.

The first trace appears in the waveform window.

3. In the Direct Plot form, do the following:

a. Highlight *Append* for *Plotting Mode*.

b. Leave *Voltage* set for *Function* and *time* set for *Sweep*.

c. Following the message at the bottom of the Direct Plot form

Select Net on schematic...

Click the `baseband_Q_out` net.

The second trace is added to the waveform window. Both baseband equivalent output signals for the baseband model are plotted.

4. In the Direct Plot form, do the following:

a. Leave *Append* for *Plotting Mode* and *Voltage* for *Function.*

b. Highlight *harmonic time* for *Sweep*.

c. Highlight *Real* for *Modifier*.

d. Following the message at the bottom of the form,

```
Select Harmonic Number on this form...
```

Select 1 for harmonic number.

e. Following the next message at the bottom of the form

```
Select Net on schematic...
```

Click the `RF_out` net.

A third trace is added to the waveform window.

5. In the Direct Plot form, do the following:

a. Leave *Append* for *Plot Mode*, *Voltage* for *Function*, *harmonic time* for *Sweep*, and 1 for *Harmonic Number*.

b. Highlight *Imaginary* for *Modifier*.

6. Following the message at the bottom of the form

   `Select Net on schematic...`

   Click the `RF_out` net again.

   A fourth trace is added to the waveform window. Both baseband equivalent output signals for the passband model are added to the plot.

In the waveform display window you should now see what at first appears to be two traces. When you look more closely, you should see that each trace is actually two traces, one nearly on top of the other, making a total of four traces.

The plot resulting from this example illustrates how well baseband modeling corresponds to the time-varying fundamental Fourier component computed by Envelope analysis and raises two questions:

- Why use baseband models when Envelope analysis gives the same results?

- Why not use baseband models all the time?

Running a transient analysis with only the baseband models answers the first question. If from the ADE window you deactivate the passband circuit by setting the `carrier_pb` variable to zero, disable the Envelope analysis, and set up and run a 10µs transient analysis, you observe the same baseband results, but the transient simulation runs over 100 times faster.

Examining the Envelope results answers the second question. If you look closely at both waveforms you notice that the baseband waveforms clip at a slightly lower level than the Envelope waveforms. This is because hard limiting of the carrier generates higher-than-third-order harmonics and the behavioral baseband model only simulates third order non-linearities.

rfLib Library Overview

The *rfLib* include three kinds of models to support baseband modeling:

- Instrumentation models
- Non-linear memoryless models
- Linear models with memory

The instrumentation models provide stimuli, diagnostics, and performance metrics relevant to the DSP system.

Both the linear models with memory and the non-linear memoryless models simulate the function blocks in an RF architecture and are specified in terms of common RF metrics. The RF function block models include input referenced white Gaussian noise as specified by noise figure. The *rfLib* includes models for the following RF function blocks—amplifiers, mixers, filters, and phase shifters; where *filters* includes single resistors, capacitors, and inductors.

The non-linear models simulate AM/AM conversion [1] with a third-order polynomial that saturates at the peak of the transfer curve. The polynomial is specified by the gain and either the input-referred IP3 or the output-referred 1 dB compression point. Only the non-linear baseband models simulate AM/PM conversion. AM/PM conversion [1] is an important effect that is hard, if not impossible, to simulate with passband behavioral models. Figure B-4 on page 2149 shows the basic baseband non-linearity.

Figure B-4 Basic, Baseband Non-Linearity

The linear models are the key to simulating loading effects at baseband. In RF integrated circuits, loading effects are important because it is often hard to integrate impedance matching networks. The baseband models of reactive elements differentiate our approach from the spreadsheet-based approaches to RF system design. The baseband capacitor and inductor models (*cap_BB* and *ind_BB* in *top_dwnBB*) let you simulate reactive loading effects in the time domain, where non-linearities are more naturally modeled.

The baseband models of reactive elements also play a key role in modeling filters. Most digital communications text books [1,2] explain that you can model a passband transfer function at baseband by simply frequency-shifting the transfer function. What these books do not describe is how to implement the resulting transfer function in a general circuit simulator such as Virtuoso® Spectre® circuit simulator RF analysis (SpectreRF). The shifted transfer function usually lacks complex conjugate symmetry about zero frequency and therefore has a complex impulse response.

The first consequence of modeling RF function blocks at baseband is that all equivalent baseband models have four terminals instead of two:

- One set of terminals represents the in-phase signals, $i_{in}(t)$ and $i_{out}(t)$
- The other set of terminals represents the quadrature signals, $q_{in}(t)$ and $q_{out}(t)$

  Both sets of terminals are illustrated in Figure B-2 on page 2135.

The mathematics illustrated in Figure B-5 on page 2151 and Figure B-6 on page 2152 summarize the ideas behind a time-varying coordinate transformation that models reactive elements at baseband. The mathematics apply to capacitors as well as inductors.

There is a well-documented but little-known electro-mechanical analogy for the derivation of the inductor baseband equivalent model. The four inductor terminals resemble the stator windings of a two-phase rotating machine with shaft speed equal to the RF carrier frequency. Modulation is mathematically analogous to the flux linking a stator winding due to currents in orthogonal rotor windings. The flux depends on the shaft angle just as a modulated signal depends on the carrier phase. Transforming the vectorial equation for $v=Ldi/dt$ to the rotor reference frame suppresses the RF carrier and introduces a *speed voltage* [3,4,5,6,7,8,9], or back electro-motive force (back EMF), that couples the differential equations.

An expression for the real current (i.e. the passband current) appears in Figure B-6 on page 2152. The real current is modeled as the projection of a two-dimensional rotating vector onto a stationary axis, the *real* axis. The vector rotates with an angular velocity equal to the RF carrier frequency.

Figure B-5 Passband Current for an Inductor

$$i_{real} = i_i(t)\cos(\omega_{rf}t) - i_q(t)\sin(\omega_{rf}t) = \mathrm{Real}[(i_i+j*i_q)*e^{j\omega_{rf}t}]$$

$$i_{imag} = i_i(t)\sin(\omega_{rf}t) + i_q(t)\cos(\omega_{rf}t) = \mathrm{Imag}[(i_i+j*i_q)*e^{j\omega_{rf}t}]$$

$$i=(i_i+j*i_q)*e^{j\omega_{rf}t}$$

$$v=(v_i+j*v_q)*e^{j\omega_{rf}t}$$

$$v=L\,di/dt$$

$$(v_i+j*v_q)\cancel{e^{j\omega_{rf}t}}=L*[d(i_i+j*i_q)/dt + j*\omega_{rf}*(i_i+j*i_q)]\cancel{e^{j\omega_{rf}t}}$$

The rotating vector also has a projection onto another stationary axis orthogonal to the real axis. In the baseband literature, the orthogonal projection is the Hilbert transform of the real signal. The constitutive relationship of the inductor, *v=Ldi/dt*, is expressed in terms of coordinates in a reference frame that rotates with the vector.

Figure B-6 on page 2152 shows the constitutive inductor relationship between voltage and current in the rotating reference frame. Note that the trigonometric terms, the terms that slow simulation speed, are gone and the two projections are now coupled through *speed voltages*. The term speed voltage comes from the fact that the voltages depend on the angular speed of the rotating reference frame. In motor theory, that speed is the shaft speed. Speed voltage is similar to the back EMF in a motor. Because of speed voltages, baseband models of filters and reactive elements must have their carrier frequency specified. The carrier frequency is the frequency for which the baseband signals are referenced. For example, the carrier frequency for an RF filter would be the RF frequency while the carrier frequency for an IF filter would be the IF frequency.

Figure B-6 Relationship Between Voltage and Current for an Inductor

The baseband counterparts of the passband filter models are built up from inductors and capacitors modeled in the rotating reference frame.

In the complex expression for $v=L\,di/dt$, if you replace d/dt with $j\omega$, you find that the impedance of the inductor changes from $jL\omega$ to $jL(\omega+\omega_{rf})$. The same holds for capacitors, which means a filter transfer function, $H(\omega)$, has a baseband equivalent equal to $H(\omega+\omega_{rf})$. This is simply the original passband transfer function shifted to the left by an amount equal to the carrier frequency. Our time domain baseband models are consistent with the text book frequency domain explanation of baseband modeling.

Use Model and Design Example

This section describes how to use the baseband models during the architectural design phase. The following examples show you how to

- Construct a baseband model for a simple receiver

- Use the Circuit Optimizer to balance specifications among the function blocks
- Create a passband testbench for the receiver

The design goals were chosen arbitrarily. The example is meant simply to illustrate how to use the library and is a derivative of the design found in [10]. If you find that some parameters are not specified, leave them as default values. You construct the receiver from left to right, from input to output.

Opening a New Schematic Window

1. In the CIW, choose *File – New – Cellview*.

   The Create New File form appears.

2. In the Create New File form,

   a. Choose *my_rfExamples* in the *Library Name* cyclic field. (Choose the editable copy of *rfExamples* you created as described in Chapter 3.)

   b. Enter `receiver_example` in the *Cell Name* field.

   c. Select *Composer-Schematic* in the *Tool* cyclic field. `schematic` appears in the *View Name* field.

   d. The completed form appears like the one below.

3. Click *OK*.

An empty Schematic window for the *receiver_example* appears.

Opening the Analog Environment

1. In the Schematic window, choose *Tools – Analog Environment*.

   The ADE window opens.

   The *Library*, *Cell*, and *View* names appear in the *Design* section of the ADE window.

2. Set the simulator options from the Simulator window as described in "Choosing Simulator Options" on page 2138.

3. Set up the model libraries from the Simulator window as described in "Setting Up Model Libraries" on page 2140.

Constructing the Baseband Model for the Receiver

Construct the receiver in the Schematic window by adding blocks from left to right, from input to output, as listed in Table B-1.

Except for the resistor, ground, and port models (which come from the *analogLib*), all blocks come from the *rfLib*. Unless otherwise instructed, leave the port resistances at their default value of 50 Ohms.

Table B-1 Blocks Used to Create the Receiver

Block Name and Reference	Element Name	Library and Category
CDMA signal source — See Adding the CDMA Signal Source	`CDMA_reverse_xmit`	From the *measurement* category in *rfLib*.
Resistor—attach to CDMA signal source	`res`	From *analogLib*
Driver — See Adding the Driver	`BB_driver`	From the *measurement* category in *rfLib*.
Low noise amplifier — See Adding the Low Noise Amplifier	`LNA_BB`	From the *top_dwnBB* category in *rfLib*.
Butterworth bandpass filter — See Adding a Butterworth Band Pass Filter	`BB_butterworth_bp`	From the *top_dwnBB* category in *rfLib*
RF-to-IF mixer — See Adding an RF-to-IF Mixer	`dwn_cnvrt`	From the *top_dwnBB* category in *rfLib*
Butterworth bandpass filter — See Adding Another Butterworth Bandpass Filter	`BB_butterworth_bp`	From the *top_dwnBB* category in *rfLib*
IQ demodulator — See Adding an IQ Demodulator	`IQ_demod_BB`	From the *top_dwnBB* category in *rfLib*
Butterworth lowpass filters (create two) — See Adding Two Butterworth Lowpass Filters	`butterworth_lp`	From the *top_dwnPB* category in *rfLib*
Instrumentation model — See Adding an Instrumentation Block	`offset_comms_instr`	From the *measurement* category in *rfLib*.
Terminator — See Adding an Instrumentation Terminator	`instr_term`	From the *measurement* category in *rfLib*
Grounds—attach to RF-to-IF mixer, IQ demodulator, and Instrumentation model	`gnd`	From *analogLib*

Adding the CDMA Signal Source

Add the first receiver block, a CDMA signal source (`CDMA_reverse_xmit`), to the schematic.

1. In the Schematic window, choose *Add – Instance*.

   The Add Instance form appears. It may be empty or it may display information for a previously added element. The default for the *View* field is `symbol`.

2. In the Add Instance form, click *Browse*.

   The Library Browser - Add Instance form appears.

3. In the Library Browser - Add Instance form,

   a. If necessary, click *Show Categories* to display the *Category* column so you can view the elements (or cells) in the *rfLib* by category.

   b. In the *Libraries* column, click *rfLib* to display categories of elements in *rfLib*.

      The *Everything* category is displayed by default and all cells in *rfLib* are listed in the *Cells* column. (In the Add Instance form, *rfLib* displays in the *Library* field.)

   c. In the *Category* column, click *measurement* to list only the cells in the *measurement* category.

   d. In the *Cell* column, click *CDMA_reverse_xmit*.

In the Library Browser, cell *CDMA_reverse_xmit* and it's default view *symbol* are both selected.

In the Add Instance form,

- `rfLib` appears in the *Library* field
- `CDMA_reverse_xmit` displays in the *Cell* field
- `symbol` displays in the *View* field

The CDF parameters for the element and their default values appear at the bottom of the form.

4. To place a *CDMA_reverse_xmit* block in the schematic,

a. Move the cursor over the Schematic window.

The outline for the *CDMA_reverse_xmit* symbol is attached to the cursor.

b. Move the cursor near the top left corner of the schematic and click to place the *CDMA_reverse_xmit* block.

This block models a CDMA signal source.

c. Click *Esc* to remove the symbol from the cursor.

Adding a Resistor to the CDMA Signal Source

Because this example does not use the binary output nodes (`i_bin_node` and `q_bin_node`) on the CDMA signal source, connect a resistor between these nodes to avoid unused pin warnings.

1. In the *Libraries* column of the Library Browser - Add Instance form, click *analogLib* to display elements in *analogLib*.

   If *Show Categories* is selected, the *Everything* category is displayed by default and all cells in *analogLib* are listed in the *Cells* column.

2. Scroll through the list of cells in *analogLib* to locate the resistor cell, *res*.

3. Click *res* in the *Cell* column.

The cell *res* and it's default view *symbol* are both selected.

In the Add Instance form,

- `analogLib` displays in the *Library* field
- `res` displays in the *Cell* field
- and `symbol` displays in the *View* field

The CDF parameters for the element and their default values appear at the bottom of the form.

4. Move the cursor over the Schematic window.
5. Click to place the top resistor terminal in line with the top binary output node (`i_bin_out`) on the lower right side of the *CDMA_reverse_xmit* block
6. Click *Esc* to remove the symbol from the cursor.

Wiring the Resistor to the CDMA Signal Source

Wire the resistor to the binary outputs, `i_bin_node` and `q_bin_node`, of the CDMA signal source.

1. To wire the resistor to the *CDMA_reverse_xmit* block, in the Schematic window choose *Add - Wire (narrow)*.
2. Click `i_bin_node` on the `CDMA_reverse_xmit` block then move the cursor and click the top node of the resistor.
3. Click `q_bin_node` on the `CDMA_reverse_xmit` block then move the cursor and click the bottom node of the resistor.
4. Click *Esc* to stop wiring.

The CDMA signal source and resistor wired together appear as follows.

Adding the Driver

Add a driver block to the right of the CDMA signal source block.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.
2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.
3. In the Library Browser - Add Instance form, make the following selections.

Library	Category	Cell	View
rfLib	*measurement*	*BB_driver*	*symbol*

At the top of the Add Instance form,

- `rfLib` displays in the *Library* field,
- `BB_driver` displays in the *Cell* field and
- `symbol` displays in the *View* field.

- The CDF parameters and their default values appear at the bottom of the Add Instance form.

4. Move the cursor over the Schematic window.
5. Click to place the *BB_driver* to the right of the *CDMA_reverse_xmit* block. Align the input pins of the driver with the analog output pins of the CDMA signal source.
6. Click *Esc* to remove the symbol from the cursor.

Wiring the Signal Source to the Driver

1. To wire the *BB_driver* block to the *CDMA_reverse_xmit* block, in the Schematic window choose *Add – Wire (narrow)*.
2. Click `i_out_node` on the *CDMA_reverse_xmit* block then click `I_in` on the *BB_driver* block.
3. Click `q_out_node` on the *CDMA_reverse_xmit* block then click `Q_in` on the *BB_driver block*.
4. Click *Esc* to stop wiring.

The schematic now appears as follows.

Modifying Parameter Values for the Driver

Edit the value of the *BB_driver* CDF parameter *dBm-out@1v peak in driver* as follows.

1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the values of CDF (component description format) properties for the driver and modify the schematic for this simulation.

2. In the Schematic window, click the *BB_driver* block.

3. The Edit Object Properties form changes to display information for the *BB_driver* block

4. Change the *dBm-out@1v peak in* parameter value as follows.

Parameter Name	Value
dBm-out@1v peak in	*-16*

The driver converts 1 peak volt from the CDMA signal source to –16 dBm referenced to the output resistance of the driver.

5. Click *OK* in the Edit Object Properties form.

   The form closes.

Adding the Low Noise Amplifier

Add a low noise amplifier to the right of the driver.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

3. In the Library Browser - Add Instance form, make the following selections. (If necessary, click *Show Categories* at the top of the Library Browser, to display the *Category* column.)

Library	Category	Cell	View
rfLib	top_dwnBB	LNA_BB	symbol

In the Library Browser, cell *LNA_BB* and it's default view *symbol* are both selected.

At the top of the Add Instance form,

- `rfLib` displays in the *Library* field
- `LNA_BB` displays in the *Cell* field
- `symbol` displays in the *View* field

The CDF parameters for the element and their default values display at the bottom of the LNA_BB Add Instance form.

4. Move the cursor over the Schematic window. The outline for the LNA symbol is attached to the cursor. Align the input pins of the LNA with the output pins of the driver.
5. Click to place the *LNA_BB* block to the right of the *BB_driver* block.
6. Click *Esc* to remove the symbol from the cursor.

Wiring the Driver to the LNA

1. To wire the *LNA_BB* block to the *BB_driver* block, in the Schematic window choose *Add - Wire (narrow)*.
2. Click `I_out` on the *BB_driver* block then click `I_in` on the *LNA_BB* block.
3. Click `Q_out` on the *BB_driver* block then click `Q_in` on the *LNA_BB* block.
4. Click *Esc* to stop wiring.

The schematic now appears as follows.

Modifying Parameter Values for the LNA

1. Edit the CDF parameter values for the LNA.

   a. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for the LNA and modify the schematic for this simulation.

   b. In the Schematic window, click the *LNA*.

   The Edit Object Properties form changes to display information for the LNA.

   c. Change the CDF parameter values for the LNA as follows.

Parameter Name	Value
Available pwr gain [dB]	`lna_gain`
Input resistance	`50`
Output resistance	`300`
Input referred IP3 [dBm]	`lna_ip3`
Noise figure [dB]	`10`

Parameter Name	Value
am/pm sharpness	`2`
cmp [dBm]	`lna_ip3`
|radians| @ cmp	`.05`
|radians| @ big input	`.7`
{1, 0, -1} for {cw, none, ccw}	`1`

2. Click *OK* in the Edit Object Properties form.

   The form closes.

Adding a Butterworth Band Pass Filter

Add a Butterworth band pass filter to the right of the low noise amplifier.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   Information for the LNA is still displayed in the form.

2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

3. In the Library Browser - Add Instance form, make the following selection.

Library	Category	Cell	View
rfLib	*top_dwnBB*	*BB_butterworth_bp*	*symbol*

   In the Library Browser, cell *BB_butterworth_bp* and it's default view are both selected.

   At the top of the Add Instance form,

   - `rfLib` displays in the *Library* field
   - `BB_butterworth_bp` displays in the *Cell* field
   - `symbol` displays in the *View* field.

- The CDF parameters for the Butterworth band pass filter and their default values display at the bottom of the Add Instance form.

4. Move the cursor over the Schematic window. The outline for the filter symbol is attached to the cursor. Align the input pins of the filter with the output pins of the LNA.
5. Click to place the *BB_butterworth_bp* block to the right of the *LNA_BB* block.
6. Click *Esc* to remove the symbol from the cursor.

Wiring the LNA to the Filter

1. To wire the *BB_butterworth_bp* block to the *LNA_BB* block, in the Schematic window choose *Add – Wire (narrow)*.
2. Click `I_out` on the *LNA_BB* block then click `ini` on the *BB_butterworth_bp* block.
3. Click `Q_out` on the *LNA_BB* block then click `inq` on the *BB_butterworth_bp* block.
4. Click *Esc* to stop wiring.

The schematic now appears as follows.

Modifying Parameter Values for the Band Pass Filter

Edit the CDF parameter values for the Butterworth band pass filter.

1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for the filter and modify the schematic for this simulation.

2. In the Schematic window, click the filter.

   The Edit Object Properties form changes to display information for the filter.

3. Edit the parameter values to match those in Table 2-2.

4. Click *OK*.

Table 2-2 CDF Parameter Values for the Butterworth Filter

Parameter Name	Value
Filter order	`3`
Input impedance	`50`
Output impedance	`50`
Center frequency (Hz)	`frf`

Table 2-2 CDF Parameter Values for the Butterworth Filter

Parameter Name	Value
Relative bandwidth	`rf_rbw`
Insertion loss (dB)	`3`
Carrier frequency	`frf`

Specify the *Carrier frequency* parameter value for the baseband equivalent model of the Butterworth band pass filter, just as you do for any reactive element. As shown in Figure B-6 on page 2152, the carrier frequency is used to compute speed voltages. Because filters are built up from inductors and capacitors which have speed voltages, you must specify the carrier frequency for filters.

When a filter follows an RF-to-IF mixer, its *Carrier frequency* parameter value is the IF frequency.

- The *Carrier frequency* is the frequency value to which the baseband signals are referenced.
- The *Center frequency* is the frequency for which a filter is designed. The *Center frequency* parameter value for a bandpass filter does not have to equal the *Carrier frequency* parameter value.

Adding an RF-to-IF Mixer

Add an RF-to-IF mixer (*dwn_cnvrt*) block to the right of the bandpass filter block.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

3. In the Library Browser - Add Instance form, make the selections indicated in Table 2-3.

Table 2-3 Library Browser selections for the RF to IF Mixer

Library	Category	Cell	View
rfLib	*top_dwnBB*	*dwn_cnvrt*	*symbol*

In the Library Browser, cell *dwn_cnvrt* (the RF-to-IF mixer) and it's default view *symbol* are both selected.

At the top of the Add Instance form,

- `rfLib` displays in the *Library* field
- `dwn_cnvrt` displays in the *Cell* field
- `symbol` displays in the *View* field.

The CDF parameters for the down converter and their default values display at the bottom of the form.

4. Move the cursor over the Schematic window. The outline for the RF-to-IF Mixer symbol is attached to the cursor. Align the input pins of the mixer with the output pins of the filter.
5. Click to place the *dwn_cnvrt* block to the right of the *BB_butterworth_bp* block.
6. Click *Esc* to remove the symbol from the cursor.

Grounding the phase_err Pin on the Mixer

It is necessary to ground the phase error (*phase_err*) pin on the bottom of the RF-to-IF mixer.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, type

   - `analogLib` in the *Library* field
   - `gnd` in the *Cell* field
   - `symbol` in the *View* field.

3. Move the cursor over the Schematic window.
4. Click to place the ground terminal in line with the phase error node (`phase_err`) on the bottom of the *dwn_cnvt* block.
5. Click *Esc* to remove the symbol from the cursor.

Wiring the Filter and Ground to the Mixer

1. To wire the *dwn_cnvrt* block to the *BB_butterworth_bp* block and the ground, in the Schematic window choose *Add - Wire (narrow)*.
2. Click `outi` on the *BB_butterworth_bp* block then click `I_in` on the *dwn_cnvrt* block.
3. Click `outq` on the *BB_butterworth_bp* block then click `Q_in` on the *dwn_cnvrt* block.
4. Click the port on the *gnd* block then click `phase_err` on the *dwn_cnvrt* block.
5. Click *Esc* to stop wiring.

The schematic now appears as follows.

Modifying Parameter Values for the RF-to-IF Mixer

1. Edit the CDF parameter values for the RF-to-IF mixer (*dwn_cnvrt*) as listed in Table 2-4 on page 2173.

   a. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for *dwn_cnvrt* and modify the schematic for this simulation.

   b. In the Schematic window, click *dwn_cnvrt*.

   The Edit Object Properties form changes to display information for *dwn_cnvrt*.

   c. Change the parameter values to match those in Table 2-4.

   d. Click OK.

Table 2-4 CDF Parameter Values for the RF-to-IF Mixer

Parameter Name	Value
available power gain[dB]	`if_mx_gain`
Input resistance	`50`
output resistance	`50`
input referred ip3[dBm]	`if_mx_ip`
noise figure [dB]	`10`

Table 2-4 CDF Parameter Values for the RF-to-IF Mixer

Parameter Name	Value
RF frequency	`frf`
LO frequency	`flo1`
AM/PM input point[dBm]	`-30`
phase shift at cmp[rad]	`.7`
phase shift at infinity	`2`
sharpness factor	`2`
{1,0,-1} = {cw,none,ccw}	`0`

Adding Another Butterworth Bandpass Filter

Add another Butterworth band pass filter block to the right of the RF-to-IF Mixer block.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.
2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.
3. In the Library Browser - Add Instance form, make the following selection.

Library	Category	Cell	View
rfLib	*top_dwnBB*	*BB_butterworth_bp*	*symbol*

In the Library Browser, cell *BB_butterworth_bp* and it's default view are both selected.

At the top of the Add Instance form,

- `rfLib` displays in the *Library* field
- `BB_butterworth_bp` displays in the *Cell* field
- `symbol` displays in the *View* field.

The CDF parameters for the Butterworth band pass filter and their default values display at the bottom of the Add Instance form.

4. Move the cursor over the Schematic window. The outline for the filter symbol is attached to the cursor. Align the input pins of the filter with the output pins of the LNA.
5. Click to place the *BB_butterworth_bp* block to the right of the *LNA_BB* block.
6. Click *Esc* to remove the symbol from the cursor.

Wiring the Mixer to the Filter

1. To wire the *BB_butterworth_bp* block to the *dwn_cnvrt* block, in the Schematic window choose *Add - Wire (narrow)*.
2. Click `I_out` on the *dwn_cnvrt* block then click `ini` on the *BB_butterworth_bp* block.
3. Click `Q_out` on the *dwn_cnvrt* block then click `inq` on the *BB_butterworth_bp* block.
4. Click *Esc* to stop wiring.

The schematic now appears as follows.

Modifying Parameter Values for the Band Pass Filter

1. Edit the CDF parameter values for the Butterworth band pass filter as listed in Table 2-5 on page 2176.

   a. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for the filter and modify the schematic for this simulation.

   b. In the Schematic window, click the filter.

   The Edit Object Properties form changes to display information for the filter.

   c. Change the parameter values to match those in Table 2-5.

   d. Click OK.

Table 2-5 CDF Parameter Values for the Second Butterworth Filter

Parameter Name	Value
Filter Order	`3`
Input impedance	`50`
Output impedance	`50`
Center frequency (Hz)	`-frf+flo1`
Relative bandwidth	`if_rbw`

Table 2-5 CDF Parameter Values for the Second Butterworth Filter

Parameter Name	Value
Insertion loss (dB)	`1`
Carrier frequency	`-frf+flo1`

As for the first band pass filter, specify the carrier frequency for the baseband equivalent model of the Butterworth band pass filter, just as you do for any reactive element. As shown in Figure B-6 on page 2152, the carrier frequency is used to compute speed voltages. Because filters are built up from inductors and capacitors which have speed voltages, you must specify the carrier frequency for filters.

When a filter follows an RF-to-IF mixer, its carrier frequency is the IF frequency. The carrier frequency is the frequency to which the baseband signals are referenced. The center frequency of the bandpass filter does not have to equal the carrier frequency. The center frequency is the frequency for which a filter is designed.

Adding an IQ Demodulator

Add an IQ Demodulator (*IQ_demod_BB*) block to the right of the bandpass filter block.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

3. In the Library Browser - Add Instance form,

   a. If necessary, click *Show Categories* to display the *Category* column.

   b. Click *rfLib* to display elements in *rfLib*. The *Everything* category is displayed by default and all cells in *rfLib* are listed in the *Cells* column.

   c. In the *Category* column, click *top_dwnBB* to display cells in the *top_dwnBB* category.

   d. In the *Cell* column, click *IQ_demod_BB*.

In the Library Browser, cell *IQ_demod_BB* (the IQ demodulator) and it's default view *symbol* are both selected.

At the top of the Add Instance form, `rfLib` displays in the *Library* field, `IQ_demod_BB` displays in the *Cell* field and `symbol` displays in the *View* field. The CDF parameters for the IQ demodulator and their default values display at the bottom of the form.

4. Move the cursor over the Schematic window. The outline for the IQ demodulator symbol is attached to the cursor. Align the input pins of the IQ demodulator with the output pins of the filter and click to place the *IQ_demod_BB* block to the right of the second *BB_butterworth_bp* block.

5. Click *Esc* to remove the symbol from the cursor.

Grounding the phase_err Pin on the IQ Demodulator

It is necessary to ground the phase error (*phase_err*) pin on the bottom of the IQ demodulator.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, type

   - `analogLib` in the *Library* field
   - `gnd` in the *Cell* field
   - `symbol` in the *View* field.

3. Move the cursor over the Schematic window.

4. Click to place the ground terminal in line with the phase error node (`phase_err`) on the bottom of the *IQ_demod_BB* block.

5. Click *Esc* to remove the symbol from the cursor.

Wiring the Filter and Ground to the IQ Demodulator

1. To wire the *IQ_demod_BB* block to the *BB_butterworth_bp* block and the ground, in the Schematic window choose *Add - Wire (narrow)*.

2. Click `outi` on the *BB_butterworth_bp* block then click `I_in` on the *IQ_demod_BB* block.

3. Click `outq` on the *BB_butterworth_bp* block then click `Q_in` on the *IQ_demod_BB* block.

4. Click the port on the *gnd* block then click `phase_err` on the *IQ_demod_BB* block.

5. Click *Esc* to stop wiring.

   The schematic now appears as follows.

Modifying Parameter Values for the IQ Demodulator

1. Edit the CDF parameter values for the IQ Demodulator (*IQ_demod_BB*) as listed in Table 2-6 on page 2181.

   a. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for *IQ_demod_BB* and modify the schematic for this simulation.

   b. In the Schematic window, click *IQ_demod_BB*.

   The Edit Object Properties form changes to display information for *IQ_demod_BB*.

   c. Change the parameter values to match those in Table 2-6.

   d. Click OK.

Table 2-6 CDF Parameter Values for the IQ Demodulator

Parameter Name	Value
available I-mixer gain[dB]	0
available Q-mixer gain[dB]	0
Input resistance	50
output resistance	50

Table 2-6 CDF Parameter Values for the IQ Demodulator

Parameter Name	Value
I-[dBm] input referred IP3	40
Q-[dBm] input referred IP3	40
noise figure [dB]	2
quadrature error	0
I-sharpness factor	2
Q-sharpness factor	2
I_cmp	-30
Q_cmp	-30
I-radians@I_cmp	.7
Q-radians@Q_cmp	.7
I-radians@big I-input	2
Q-radians@big Q-input	2
I {1,0,-1} for {cw,none,ccw}	0
Q {1,0,-1} for {cw,none,ccw}	0

Adding Two Butterworth Lowpass Filters

1. Add two Butterworth low pass filters to the right of the IQ demodulator block.
   - Align one filter with the demodulator's `i_out` pin.
   - Align the other filter with its `q_out` pin.
2. In the Schematic window, choose *Add – Instance* to display the Add Instance form.
3. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.
4. In the Library Browser - Add Instance form, make the following selections.

Library	Category	Cell	View
rfLib	*top_dwnPB*	*butterworth_lp*	*symbol*

In the Library Browser, cell *butterworth_lp* and it's default view are both selected.

At the top of the Add Instance form,

- `rfLib` displays in the *Library* field
- `butterworth_lp` displays in the *Cell* field
- `symbol` displays in the *View* field.
- The CDF parameters for the Butterworth low pass filter and their default values display at the bottom of the Add Instance form.

5. Move the cursor over the Schematic window. The outline for the butterworth low pass filter symbol is attached to the cursor.

   Align the *in* pin of the first butterworth low pass filter with the *I_out* pin (the top pin) of the IQ demodulator and click to place the filter close to the demodulator. Align the *in* pin of the second butterworth low pass filter with the *Q_out* pin (the bottom pin) of the IQ demodulator. You have to place it further from the demodulator to align it with the *Q_out* pin.

6. Click *Esc* to remove the symbol from the cursor.

Wiring the IQ Demodulator to the Filters

1. To wire the *IQ_demod_BB* block to the *butterworth_lp* blocks, in the Schematic window choose *Add - Wire (narrow)*.
2. Click `I_out` on the *IQ_demod_BB* block then click `in` on the first *butterworth_lp* block.
3. Click `Q_out` on the *IQ_demod_BB* block then click `in` on the second *butterworth_lp* block.
4. Click *Esc* to stop wiring.

The schematic now appears as follows.

Modifying Parameter Values for Both Low Pass Filters

1. Edit the CDF parameter values for the Butterworth low pass filters as listed in Table 2-7 on page 2184.

   a. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for each filter and modify the schematic for this simulation.

   b. In the Schematic window, click the first low pass filter.

   The Edit Object Properties form changes to display information for the filter.

   c. Change the parameter values to match those in Table 2-7.

Table 2-7 CDF Parameter Values for the Butterworth Low Pass Filters

Parameter Name	Value
Filter Order	3
Input impedance	50
Output impedance	50
Corner frequency (Hz)	10M
Insertion loss (dB)	0

   d. Click Apply.

e. In the Schematic window, click the second low pass filter.

The Edit Object Properties form displays the information you entered for the filter as shown in Table 2-7.

f. Click OK.

Adding an Instrumentation Block

Add an instrumentation block (*offset_comms_instr*) block to the right of the low pass filter blocks.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

3. In the Library Browser - Add Instance form, make the following selections. (If necessary, click Show Categories at the top of the Library Browser, to display the Category column.)

Library	Category	Cell	View
rfLib	*measurement*	*offset_comms_instr*	*symbol*

In the Library Browser, cell *offset_comms_instr* and it's default view *symbol* are both selected.

At the top of the Add Instance form,

- `rfLib` displays in the *Library*
- `offset_comms_instr` displays in the *Cell* field
- `symbol` displays in the *View* field

The CDF parameters for the *offset_comms_instr* and their default values display at the bottom of the form.

4. Move the cursor over the Schematic window. The outline for the *offset_comms_instr* block symbol is attached to the cursor. Align the *I_in* and *Q_in* pins of the *offset_comms_instr* block with the *out* pins of the butterworth low pass filters and click to place the *offset_comms_instr* block to the right of the low pass filter blocks.
5. Click *Esc* to remove the symbol from the cursor.

Grounding the Reference Pins on the Instrumentation Block

It is necessary to ground the reference pins (*I_ref* and *Q_ref*) pin near the lower left corner of the instrumentation block.

Grounding the phase_err Pin on the Mixer

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, type

   - `analogLib` in the *Library* field
   - `gnd` in the *Cell* field
   - `symbol` in the *View* field.

3. Move the cursor over the Schematic window.
4. Click to place the ground terminal in line with the `Q_ref` node on the bottom of the *offset_comms_instr* block.
5. Click *Esc* to remove the symbol from the cursor.

Wiring the Filter and Ground to the Instrumentation Block

1. To wire the low pass filters to the *offset_comms_instr* block and the ground, in the Schematic window choose *Add - Wire (narrow)*.
2. Click `out` on the upper *butterworth_lp* block (the low pass filter connected to the `I_out` node on the IQ demodulator) then click `I_in` on the *offset_comms_instr* block.
3. Click `out` on the lower *butterworth_lp* block (the low pass filter connected to the `Q_out` node on the IQ demodulator) then click `Q_in` on the *offset_comms_instr* block.
4. Click the port on the *gnd* block then click the `Q_ref` node on the *offset_comms_instr* block.
5. Click the port on the *gnd* block then click the `I_ref` node on the *offset_comms_instr* block.
6. Click *Esc* to stop wiring.

The schematic now appears as follows.

Modifying Parameter Values for the Instrumentation Block

1. Edit the CDF parameter values for the *offset_comms_instr* as listed in Table 2-8 on page 2189.

   a. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for *offset_comms_instr* and modify the schematic for this simulation.

   b. In the Schematic window, click the *offset_comms_instr* block.

   The Edit Object Properties form changes to display information for *offset_comms_instr* block.

   c. Change the parameter values to match those in Table 2-8.

d. Click OK.

Table 2-8 CDF Parameter Values for the Instrumentation Block

Parameter Name	Value
symbols per second	`1228800`
I-sampling delay (secs)	`134n`
number of symbols	`2`
max eye-diag volts	`1`
min eye volts	`-1`
number of hstgm bins	`100`
I-noise (volts^2)	`0`
Q-noise (volts^2)	`0`
statistics start time	`30u`
input resistance	`50`

Adding an Instrumentation Terminator

Add an instrumentation termination (*instr_term*) block to the right of the instrumentation block. The *instr_term* block terminates the outputs on the instrumentation block and prevents unused pin warnings.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

3. In the Library Browser - Add Instance form, make the following selections. (If necessary, click Show Categories at the top of the Library Browser, to display the Category column.)

Library	Category	Cell	View
rfLib	*measurement*	*instr_term*	*symbol*

In the Library Browser, cell *instr_term* and it's default view *symbol* are both selected.

At the top of the Add Instance form,

- `rfLib` displays in the *Library*
- `instr_term` displays in the *Cell* field
- `symbol` displays in the *View* field

There are no CDF parameters for the *instr_term* cell.

4. Move the cursor over the Schematic window. The outline for the *instr_term* symbol is attached to the cursor. Move the *instr_term* block to the right of the instrumentation block and align the input pins of the instrumentation termination block with the output pins of the instrumentation block.
5. Click to place the *instr_term* block.
6. Click *Esc* to remove the symbol from the cursor.

Wiring the Termination Block to the Instrumentation Block

1. To wire the instrumentation (*offset_comms_instr)* block to the Instrumentation terminator (*instr_term)* block, in the Schematic window choose *Add - Wire (narrow)*.
2. Wire the aligned pins straight across.
3. Click *Esc* to stop wiring.

The schematic should look as follows.

4. In the Schematic window, choose *Design – Design Check and Save*.

   The completed schematic is verified and saved.

The schematic for the complete receiver model should look like the one in Figure B-7 on page 2192.

Figure B-7 Completed receiver model

Setting Variable Values for the Receiver Schematic

Copy the variables you entered as CDF parameters for the individual blocks from the receiver schematic to the ADE window. Then edit each variable to give it the value specified in Table B-9 on page 2193.

1. In the ADE window, choose *Variables – Copy From Cellview* to copy the variables from the receiver schematic to the Design Variables area on the ADE window.

The copied variables display in the *Design Variables* area in the ADE window.

2. In the ADE window, choose *Variables – Edit* to open the Editing Design Variables form.

Adding the Values to the Copied Variables

In the Editing Design Variables form, one by one, select each variable in the *Table of Design Variables* and associate with each one, the value listed in Table B-9.

Table B-9 Values for Receiver Variables

Variable	Value
`lna_gain`	15

Table B-9 Values for Receiver Variables

Variable	Value
`lna_ip3`	-5
`if_mx_gain`	10
`if_mx_ip`	35
`frf`	2.14G
`flo1`	2.354G
`if_rbw`	200m
`rf_rbw`	100m

To associate a value with a design variable

1. In the *Table of Design Variables*, click `lna_gain`.

   `lna_gain` displays in the *Name* field.

2. In the *Value (Expr)* field, enter the number `15`, the value from Table B-9 on page 2193.

3. Click *Change* to list the variable name and its value from the *Table of Design Variables.*

4. Repeat these steps for the remaining variables listed in the *Table of Design Variables* to associate the values from Table B-9 on page 2193 with the variable names.

5. Click *OK* in the Editing Design Variables form after you have added all the variable values.

The table of *Design Variables* in the ADE window is updated and the Editing Design Variables form is closed.

Design Variables

	Name	Value
1	flo1	2.354G
2	frf	2.14G
3	if_mx_gain	10
4	if_mx_ip	35
5	if_rbw	200m
6	lna_gain	15
7	lna_ip3	-5
8	rf_rbw	100m

Setting Up and Running a Transient Analysis

1. In the ADE window, choose *Analyses – Choose* to display the Choosing Analyses form.
2. In the Choosing Analyses form, if necessary, click *tran* to select a transient analysis.
3. In the Choosing Analyses form, enter `130u` in the *Stop Time* field.

4. Highlight *moderate* for *Accuracy Defaults (errpreset)*.

5. In the Choosing Analyses form, click *Options* to display the Transient Options form.
6. In the Transient Options form, enter `30u` in the *outputstart* field.

By delaying the output start, you remove start-up transients from the eye-diagrams and scatter plots.

7. Click *OK* in the Transient Options form.

8. Click *OK* in the Choosing Analyses form.
9. If you have not already done so, set up the simulator and model libraries with the following steps.
   a. Set the simulator options from the Simulator window as described in "Choosing Simulator Options" on page 2138.
   b. Set up the model libraries from the Simulator window as described in "Setting Up Model Libraries" on page 2140.
10. In the ADE window, choose *Simulation – Netlist and Run*.

    Messages display in the CIW. The simulation log window opens. Watch for messages stating that the simulation has completed successfully.

    Watch the CIW for messages stating the simulation is running and that it has completed successfully.

Examining the Results: Eye Diagram, Histogram, and Scatter Plot

In this section we examine the results of the transient analysis of the receiver.

Plotting the Eye Diagram (and Transient Response)

First plot an eye diagram.

1. In the ADE window, choose *Results – Direct Plot – Transient Signal*.

   The Virtuoso Schematic Editing window appears.
2. Following the prompts at the bottom of the window,

   > Select nodes or terminals, press <esc> to finish selection

   a. In the Schematic window, click the *sawtooth* net from the instrumentation (*offset_comms_instr*) block.

b. Click the *I_eye* net from the instrumentation (*offset_comms_instr*) block.

c. Press *Esc* to indicate that you have finished selecting outputs.

This opens a waveform window and creates a plot of the Transient Response of the net.

3. In the waveform window, double-click the X axis.

The Axis Attributes form displays:

a. In the *Plot vs* field, select the `sawtooth` net, such as `/net15`.

b. Click *OK*.

You see the eye-diagram shown in Figure B-8 on page 2199.

Figure B-8 Eye-diagram

The I-sampling delay parameter in the instrumentation block (*I_del*) is chosen with respect to this eye diagram. The delay is the time when the eye opens the widest.

The instrumentation block samples the input waveforms with this delay to compute all statistics and to produce scatter plots.

4. In the waveform window, choose *File – Close*.

   The waveform window closes.

Generate the Histogram

Now generate a histogram of the I-voltage at the sampling times.

1. In the ADE window, choose *Results – Direct Plot – Transient Signal*.

   The Virtuoso Schematic Editing window appears.

2. Following the prompts at the bottom of the window,

   a. Click the *eye_hist* net from the instrumentation (*offset_comms_instr)* block.

   b. Click the *eye_count_hist* net from *offset_comms_instr*.

   c. Press *Esc* to indicate that you have finished selecting outputs.

      This creates a plot in the waveform window.

3. In the waveform window, double-click the X axis.

   The Axis Attributes form displays:

   a. In the *Plot vs* field, select the *eye_hist* output, such as `/net17`.

   b. Click *OK*.

   This creates an unintelligible intermediate plot in the waveform window.

4. In the waveform window, double-click the trace in the graph.

   The Trace Attributes form displays.

   a. In the *Type/Style* cyclic field, select *Bars*.

   b. Click *OK*.

   You see a plot like Figure B-9 on page 2201.

Figure B-9 Histogram

5. In the waveform window, choose *File – Close*.

   The waveform window closes.

Generating the Scatter Plot

Generate a scatter plot of the received symbols.

1. In the ADE window, choose *Results – Direct Plot – Transient Signal*.

   This displays the waveform window.

2. Following the prompts at the bottom of the waveform window,

   In the Schematic window:

   a. Click the *I_scatter* output from the instrumentation (*offset_comms_instr)* block.

   b. Click the *Q_scatter* output from *offset_comms_instr*.

   c. Click *Esc* to indicate that you have finished selecting outputs.

      This creates a plot in the waveform window.

3. In the waveform window, double-click the X axis.

The Axis Attributes form displays:

a. In the *Plot vs* field, select the *I_scatter* output, such as `/net22`.

b. Click *OK*.

4. In the waveform window, double-click the trace in the graph.

The Trace Attributes form displays.

a. In the *Type/Style* cyclic field, select *Points*.

b. Click *OK*.

You see a scatter plot like Figure B-10 on page 2202.

Figure B-10 Scatter Plot

5. In the waveform window, choose *File – Close*.

The waveform window closes.

The Various Instrumentation Blocks

The CDMA source (*CDMA_reverse_xmit)* produced offset QPSK symbols. Offset QPSK modulation avoids traversing the origin by staggering the digital changes in the I and Q

signals. Running the baseband trajectory through the origin increases spectral regrowth in the transmitters.

The instrumentation block (*offset_comms_instr*) samples the I and Q signals at different times then plots the two staggered samples against each other. The resulting scatter plot shows the received symbols. A scatter plot of the unstaggered samples reveals only what is happening in one dimension, either the I or Q dimension.

For non-offset QPSK and QAM modulation schemes, use the *comms_instr* instrumentation block instead of the *offset_comms_instr* block.

Measuring RMS EVM

You can use the same instrumentation block (*offset_comms_instr*) to compute root-mean-squared error vector magnitude (RMS EVM). The error vector is the vectorial difference between the ideal received symbol and the actual received symbol.

- EVM (error vector magnitude) is the magnitude of the error vector.
- RMS EVM is the root-mean-squared value of a sequence of EVMs.

RMS EVM is one measure of a receiver's quality. RMS EVM can account for as much or as little distortion and noise as you like. The trick is to figure out where the ideal received symbol lies. You can do this using the *I_ref* and *Q_ref* inputs to the *offset_comms_instr* instrumentation block.

To calculate RMS EVM, you

- Create a duplicate copy of the receiver chain from the *BB_driver* to the *IQ_demod_BB* including these two blocks
- Place the duplicate copy below the original receiver chain in the Schematic window
- Modify the duplicate receiver chain to make it as *ideal* as you like by changing parameter values for the individual function blocks.

  For example, to see the effect of just the LNA's IP3 value on RMS EVM, in the duplicate receiver chain make the LNA's IP3 absurdly large.

Constructing the Ideal Receiver Chain

The *ideal* receiver chain (the duplicated and modified receiver chain) is driven from the same input, the *CDMA_reverse_xmit* block, as the original receiver chain. The output of the ideal receiver chain drives the instrumentation block's *I_ref* and *Q_ref* inputs.

In the ideal receiver chain, you copy the first receiver chain and make every block ideal.

1. Remove the *gnd* from the *I_ref* and *Q_ref* pins on the instrumentation block.

   a. In the Schematic window, choose *Edit - Delete*.

   b. In the Schematic window, click each wire and the *Gnd* symbol attached to the *I_ref* and *Q_ref* pins.

   c. Click *esc* to stop deleting.

2. Duplicate the receiver chain from the *BB_driver* to the *IQ_demod_BB* inclusive. Do not duplicate the filters. Follow the prompts at the bottom of the Schematic window.

   a. In the Schematic window, choose *Edit – Copy*.

   b. In the Schematic window, draw a box around the blocks to copy by clicking to the left of and above the *BB_driver* block and dragging the cursor to a point below and to the right of the *IQ_demod_BB* block. Click again to complete the box.

   The blocks within the box are highlighted.

   c. Click within the highlighted area.

   A copy of the highlighted blocks in the receiver chain now moves with the cursor.

d. Place the duplicate receiver chain so that the output pins on the *IQ_demod_BB* are in line with the *I_ref* and *Q_ref* pins on the instrumentation block.

3. Wire the duplicate receiver chain to the CDMA signal source (*CDMA_reverse_xmit)* and the instrumentation block (*offset_comms_instr*).

a. In the Schematic window, choose *Add - Wire (narrow).*

b. Connect the *IQ_demod_BB* outputs on the duplicate receiver to the *I_ref* and *Q_ref* pins on the instrumentation block.

c. Connect the output pins on *CDMA_reverse_xmit* to the input pins of the duplicate *BB_driver*. This drives the duplicate receiver from the CDMA signal source.

d. Click *esc* to stop wiring.

The schematic with the duplicate receiver chain wired up looks like Figure B-11.

Figure B-11 Receiver Model with Duplicated Receiver Chain

Modifying Parameter Values to Make the Blocks Ideal

Now modify the parameter values for each block in the duplicate receiver chain to create ideal blocks. Block names, parameter names, and parameter values are given in Table B-10.

Table B-10 Parameter Values to Create an Ideal Receiver

Block Names	Parameter Names	New Parameter Values
`LNA_BB`	*Input referred IP3 [dBm]* *{1,0,-1} for {cw, none, ccw}*	`100` `0`
`BB_butterworth_bp`	*Cell Name*	`BB_loss`
`dwn_cnvrt`	*Input referred IP3 [dBm]*	`100`
`BB_butterworth_bp`	*Cell Name*	`BB_loss`
`IQ_demod_BB`	*I-[dBm] Input referred IP3* *Q-[dBm] Input referred IP3*	`100` `100`

1. In the Schematic window, choose *Edit – Properties – Objects* to open the Edit Object Properties form.
2. In the Schematic window, select the *LNA_BB* block.

   The Edit Object Properties form changes to display properties for the *LNA_BB* block.

   a. Set *Input referred IP3 [dBm]* to 100.

   b. Set *{1, 0,-1} for {cw, none, ccw}* to 0. (This eliminates AM/PM conversion.)

   c. Click *Apply*.
3. In the Schematic window, select the first RF *BB_butterworth_bp* block.

   The Edit Object Properties form changes to display properties for the *BB_butterworth_bp* block.

   a. Change the *Cell Name* to *BB_loss*.

   b. Click *Apply*.

   The properties and symbol change to those for the *BB_loss* block. The sole purpose of the *BB_loss* model is to replace a filter in an RMS EVM analysis.

The *Reference impedance* for the *BB_loss* block should equal the *Output impedance* of the *BB_butterworth_bp* bandpass filter block it replaces. The value should be 50 ohms for both blocks and you should not have to change it.

The *BB_loss* model retains the filter's loss but eliminates the filter's dynamics so you can see what, if any, affect the filter has on EVM through inter-symbol interference. To eliminate the loss as well as the dynamics, you might even replace the filter with straight wires. This example uses the *BB_loss* block instead.

4. In the Schematic window, select the *dwn_cnvrt* block.

   The Edit Object Properties form changes to display properties for the *dwn_cnvrt* block

   a. Set *Input referred IP3 [dBm]* to 100.

   b. Click *Apply*.

5. In the Schematic window, select the second *BB_butterworth_bp* block.

   The Edit Object Properties form changes to display properties for the *BB_butterworth_bp* block

   a. Change the *Cell Name* to *BB_loss*.

   b. Click *Apply*.

6. In the Schematic window, select the *IQ_demod_BB* block.

   The Edit Object Properties form changes to display properties for the *IQ_demod_BB* block

   a. Set *I-[dBm] input referred IP3* to 100.

   b. Set *Q-[dBm] input referred IP3* to 100.

   c. Click *Apply*.

7. In the Edit Object Properties form, click *OK* to close the form.

8. In the Schematic window, choose *Design – Check and Save* to check and save your modifications to the circuit.

Set Up and Run a Transient Analysis

Set up and run a transient analysis as described in "Setting Up and Running a Transient Analysis" on page 2195. Set the *Stop Time* to `130u` and the *outputstart* option to `30u`. Click *OK* in both the Transient Options and Choosing Analyses forms. Choose *Simulation – Netlist and Run* to run the transient analysis.

Look for messages in the CIW stating that the simulation is starting. Watch the simulation log window for messages that the simulation has completed successfully.

Plot the RMS EVM Output

After the simulation, plot the RMS_EVM output of the instrumentation block.

1. In the ADE window, choose *Results – Direct Plot – Transient Signal*.

   This displays the waveform window.

2. Following the prompts at the bottom of the waveform window,

   a. In the schematic, click the *rms_EVM* output net from the instrumentation (*offset_comms_instr)* block.

   b. Click *Esc* to indicate that you have finished selecting outputs.

   This creates the RMS EVM plot in the waveform window as shown in Figure B-12 on page 2208.

Figure B-12 RMS EVM

The RMS EVM trace starts at 30us, which is the `statistics start time` parameter of the instrumentation block. The `statistics start time` parameter keeps start-up transients out of the statistics.

The trace settles out at 25.84 Volts. This means that after 130us of data is collected, and ignoring the first 30us, the RMS EVM is 25.84%. The EVM measurement is normalized to the RMS magnitude of the ideal symbol then multiplied by 100 to express the measurement as a percentage.

3. In the waveform window, choose *File – Close*.

Computing Minimized RMS Noise Using the Optimizer

There is one more construction step before proceeding to the Circuit Optimizer application. You set up the Circuit Optimizer to minimize RMS noise subject to performance constraints. This step replicates the receiver chain yet one more time to generate the noise measurement.

1. Duplicate the original receiver chain from the *BB_driver* up to and including both low pass filters (*butterworth_lp*). Follow the prompts at the bottom of the Schematic window.

   a. In the Schematic window, choose *Edit – Copy*.

   b. In the Schematic window, draw a box around the blocks to copy by clicking to the left of and above the *BB_driver* block and dragging the cursor to a point below and to the right of the *butterworth_lp* filter blocks. Click again to complete the box.

      The blocks within the box are highlighted.

   c. Click within the highlighted area.

      A copy of the highlighted blocks in the receiver chain now moves with the cursor.

d. Place the duplicate receiver chain above the original receiver chain.

2. In the duplicate receiver chain, ground the *Q_in* pin on the *BB_driver* block.

a. In the Schematic window, choose *Add – Wire.*

b. Click the *Q_in* pin and run the wire to the *Gnd* symbol below the *dwn_cnvrt* block.

c. Click the *Gnd* symbol below the *dwn_cnvrt* block.

d. Click *esc* to stop wiring.

3. Add a 50mV DC voltage source to the left of the *BB_driver* block to drive the *I_in* pin on the *BB_driver* block. At the same time add a *gnd* symbol below the *port* in the schematic.

a. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

b. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

c. In the Library Browser - Add Instance form, click *analogLib*.

d. Scroll the elements in the *Cell* column and click *port*.

e. The outline for the *port* symbol is attached to the cursor. Move the *port* symbol to the left of the *BB_driver* block and click to place the *port* symbol.

f. Return to the Library Browser - Add Instance form and scroll the elements in the *Cell* column and click *gnd*.

g. The outline for the *gnd* symbol is attached to the cursor. Move the *gnd* symbol below the *port* symbol and click to place it there.

h. Click *Esc* to remove the *gnd* symbol from the cursor.

4. In the Schematic window, choose *Edit – Object – Properties* to modify the *port* symbol using the Edit Object Properties form.

a. In the Schematic window, click the *port* symbol.

The Edit Object Properties form changes to display information for the *port* symbol.

b. In the *Source Type* cyclic field, select *dc*.

c. In the *DC Voltage* field, enter 50m.

d. Highlight *Display small signal params* to display small signal parameters.

e. In the *AC Magnitude* field type `1V`.

f. In the *AC Phase* field type `0`.

g. Click *OK* in the Edit Object Properties form.

5. Load the low pass filters with a *res_BB* model from the *top_dwnBB* category of *rfLib*. Use the default parameters and ground the output pins.

a. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

b. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

c. In the Library Browser - Add Instance form, click *rfLib*.

d. Scroll the elements in the *Cell* column and click *res_BB*.

The outline for the *res_BB* symbol is attached to the cursor.

e. Move the *res_BB* symbol to the right of the two low pass filters (*butterworth_lp*) and click to place the *res_BB* symbol.

f. Return to the Library Browser - Add Instance form and click *analogLib.*

g. Scroll the elements in the *Cell* column and click *gnd*.

The outline for the *gnd* symbol is attached to the cursor.

h. Move the *gnd* symbol to the right of the *res_BB* symbol and click to place it there.

i. Click *Esc* to remove the symbol from the cursor.

6. In the duplicate receiver chain, wire the *port*, the *res_BB* block, and their *gnd* blocks.

a. In the Schematic window, choose *Add – Wire.*

b. Click the *I_in* pin on the *BB_driver*, then click the top pin on the *port*.

c. Click the bottom pin on the *port*, then click the *gnd* pin just below it.

d. Click the *out* pin on the top *butterworth_lp* filter, then click the *I_in* pin on the *res_BB* block.

e. Click the *out* pin on the lower *butterworth_lp* filter, then click the Q_*in* pin on the *res_BB* block.

f. Click the *I_out* pin on the *res_BB* block. Then click the top pin on the *gnd* located to it's right.

g. Click the Q_*out* pin on the *res_BB* block. Then click the top pin on the same *gnd*.

h. Click *esc* to stop wiring.

7. In the Schematic window, choose *Design – Check and Save* to check and save the schematic.

The schematic with the third receiver chain is shown in Figure B-13.

Figure B-13 Schematic with Noise Generating Receiver

Set Up and Run Transient and Noise Analyses

Set up a transient analysis as described in "Setting Up and Running a Transient Analysis" on page 2195. Set the *Stop Time* to `130u` and the *outputstart* option to `30u` and make sure that the transient analysis is enabled.

Set up a noise analysis as follows:

1. In the Choosing Analyses form, click *noise* to select a noise analysis.
2. For *Sweep Variable*, click *Frequency*.
3. For *Sweep Range*, click *Start-Stop*.
4. Set up the analysis to sweep frequency from 0 to100 MHz.
   a. For the starting frequency, in the *Start* field enter `0`.
   b. For the stop frequency, in the *Stop* field enter `100M`.
5. Set up the *Output Noise* source.

a. In the *Output Noise* cyclic field, select *voltage.*

b. To select the *Positive Output Node*, click *Select* next to the *Positive Output Node* field. Then, in the Schematic window, click the net next to the *I_in* pin on the *res_BB* block.

c. To select the *Negative Output Node*, click *Select* next to the *Negative Output Node* field. Then, in the Schematic window, click the net next to the *I_out* pin on the *res_BB* block.

6. Set up the *Input Noise* source.

a. In the *Input Noise* cyclic field, select *port.*

b. To select the *Input Port Source*, click *Select* next to the *Input Port Source* field. Then, in the Schematic window, click the *DC input port* model.

7. Click *Enabled*.

The *Output Noise* and *Input Noise* sections of the Noise analysis form look as follows. (The net instance numbers might be different, however.)

8. Verify that *Enabled* is highlighted and click *OK* in the Choosing Analyses form.

9. In the ADE window, check the Analysis area to verify that both the transient and noise analyses are set up properly and that they are both enabled.

10. In the ADE window, choose *Simulation – Netlist and Run* to start the simulations.

    Watch the messages in the CIW to verify that everything is set up properly and that the simulations start. Check the simulation log window to see that the simulations run and complete properly.

Set Up to Run the Circuit Optimizer

1. In the ADE window, choose *Tools – Calculator* to open the calculator window.

2. In the ADE window, choose *Tools – Optimization* to open the Circuit Optimizer window.

Add the First Goal

1. In the calculator window's function panel, select *rmsNoise*.
2. When the RMS Noise panel opens,
   a. In the *From* field, enter `0`.

b. In the *To* field, enter `100M`.

c. Click *OK*.

The expression

```
rmsNoise(0,100M)
```

displays in the buffer of the calculator window.

3. In the Circuit Optimizer window, choose *Goals – Add* to add the first goal.

The Adding Goals form opens.

4. Fill in the Adding Goals form.

a. In the *Name* field, type `rmsNoise`.

b. Click the *Get Expression* button to the right of the *Calculator* label.

The expression `rmsNoise(0 100000000)` displays in the *Expression* field.

c. Ensure that the *Direction* is minimize.

The default for the Optimizer is to minimize goals.

d. In the *Target* field, enter `0.1u`.

e. In the *Acceptable* field, enter `20u`.

f. Verify that the form is enabled.

The Adding Goals form looks like this.

g. Click *OK*.

The first goal is added to the Circuit Optimizer window.

Add the Second Goal

1. In the calculator window, under the *tran* tab, click the *vt* button.

   The Schematic Editing window becomes active.

2. In the Schematic window, click the *rms_EVM* output net of the instrumentation block.

   An expression similar to `VT("/net24")` displays in the calculator buffer.

3. In the calculator window's function panel, select *value*.

   When the Value panel appears

   a. Enter `130u` in the *Interpolate At* field.

   b. Click *OK*.

   The expanded expression `value(VT("/net24") 130u )` displays in the buffer of the calculator window.

4. In the Circuit Optimizer, choose *Goals – Add*.

   When the Adding Goals form opens

   a. In the *Name* field, enter `evm`.

   b. Click the *Get Expression* button to the right of the *Calculator* label.

   An expression similar to

```
value(VT("/net24") 0.00013)
```

displays in the *Expression* field.

c. In the *Direction* cyclic field, select <=

d. In the *Target* field, enter `25`.

e. In the *Acceptable* field, enter `10`.

f. Click the *% within Target* button.

g. Verify that *Enabled* is active.

The Adding Goals form looks like this.

h. Click *OK*.

The second goal is added to the Circuit Optimizer window.

Add the Third Goal

1. In the calculator window, click *Clear*.

   The buffer is cleared.

2. Under the *tran* tab, click the *vt* button.

   The Schematic Editing window becomes active.

3. In the Schematic window, click the *I_in* pin of the instrumentation block.

   An expression similar to `VT("/net14")` displays in the buffer of the calculator window.

4. In the calculator window's function panel, select *rms*.

   An expanded expression similar to

   ```
   rms(VT("/net14"))
   ```

   displays in the buffer of the calculator window.

   The objective is to keep the *rms* value of this signal level above 300 mV. Note that all goals must be scalars.

5. In the Circuit Optimizer, choose *Goals - Add*.

   The Adding Goals form opens.

6. Fill in the fields of the Adding Goals form.

   a. In the *Name* field, enter `sig_level`.

   b. Click the *Get Expression* button to the right of the *Calculator* label.

   And expression similar to

   ```
   rms(VT("/net14"))
   ```

   displays in the *Expression* field.

c. In the *Direction* cyclic field, select >=

d. In the *Target* field, enter `300m`.

e. In the *Acceptable* field, enter `10`.

f. Click the *% within Target* button.

g. Verify that the *Enabled* button is active.

The Adding Goals form looks like this.

h. Click *OK*.

The third goal is added to the Circuit Optimizer window.

Add the Circuit Variables to the Optimizer

Add the variables to the Circuit Optimizer window.

1. In the Circuit Optimizer window, choose *Variables – Add/Edit*.
2. When the Editing Variables form opens
   - **a.** In the *Name* list box, click the *lna_ip3* variable.

     The *lna_ip3* variable is highlighted in the list box and it's current value `-5` displays in the *Initial Value* field.
   - **b.** In the *Minimum Value* field, enter `-9`.
   - **c.** In the *Maximum Value* field, enter `10`.
   - **d.** If necessary, click *Enabled*.

     The Editing Variables form appears as follows.

   - **e.** Click *Apply*.

Information for the *lna_ip3* variable displays in the *Variables* section of the Circuit Optimizer.

3. Repeat this procedure to add all the variables and values listed in Table B-11

Table B-11 Variables and Values for the Optimizer

Variable Name	Initial Value	Minimum Value	Maximum Value
lna_ip3	-5	-9	10
lna_gain	15	10	30
if_mx_gain	10	1	50
if_rbw	200m	50m	300m
rf_rbw	100m	50m	300m

4. Click *OK* to close the Editing Variables form.

In the Circuit Optimizer window, the variables and optimization goals appear as shown in Figure B-14 on page 2224.

Figure B-14 Circuit Optimizer Setup

Run the Circuit Optimizer

Note: This Circuit Optimizer analysis might take several hours to complete.

1. In the Circuit Optimizer, choose *Optimizer – Run n* to display the *Run for Fixed Number of Iterations* form.
2. In the *Number of Iterations* field, move the slider to the right until *12* displays.

3. Click *OK.*

   The *Run for Fixed Number of Iterations* form closes and the Circuit Optimizer starts running.

   Watch for simulator startup messages in the CIW. Monitor the progress of the analysis in the log window.

   The waveform window opens when the first simulation completes. As simulations complete, the results are added to the plots in the waveform window.

Viewing the Circuit Optimizer Output

The Circuit Optimizer results displayed in the waveform window are shown in Figure B-15 on page 2225.

- The traces on the left show the EVM and RMS output signal level, and the RMS noise at each Circuit Optimizer iteration.
- The traces on the right show the variable function block specifications at each Circuit Optimizer iteration.

Figure B-15 Circuit Optimizer Results After 12 Iterations

Although the example is contrived, it illustrates the use model. After the Circuit Optimizer met the constraints it tried to minimize RMS noise.

1. Save the initial state of the Analog Design Environment in case you want to start over.

2. Then in the Circuit Optimizer window, click *Results – Update Design*. The last click updates the variables in the Analog Design Environment window with the last set of variables found by the Circuit Optimizer. You use these states in the passband view.

Summarizing the Design Procedure

To summarize, the semi-automated design procedure consists of

1. Setting up the measurements
2. Placing tolerances on the block parameters
3. Constraining the system performance
4. Identifying a quantity to minimize (or maximize)
5. Running the Optimizer
6. Evaluating the results

This is why the process is called *semi*-automated. After evaluating results for the first or second time you probably need to

1. Refine tolerances
2. Refine goals
3. Add or delete constraints
4. Add or delete variables

Each simulation covers 100s, or about 80 CDMA symbols. The suppressed carriers are an RF carrier at 2.14 GHz, an LO carrier at 2.354GHz, and an IF carrier at 214 MHz. The symbol rate is 1.2288 Mega-symbols per second.

Creating a Passband View of the Architectural Model

After you have designed an architecture, you can quickly create a passband view of the architectural model. (Currently, the passband behavioral models in the *top_dwnPB* category and in the passband view do not introduce any specifications that are not in the baseband models.)

The passband view checks for problems that might have escaped detection in the baseband view. For example, although the baseband view quickly assesses what filters do to the baseband signal, baseband models do not indicate whether the filters are indeed removing undesired carrier harmonics.

Baseband modeling is also not the best way to evaluate image rejection. Although the baseband model accurately simulates how the desired signal propagates through an image rejection receiver, it does not accurately simulate how much of the image signal propagates to the receiver output.

The passband view also creates a system testbench as mentioned in "Top-Down Design of RF Systems" on page 2130.

Procedures for Creating the Passband Model of the Receiver

The procedures described in this section illustrate how to

- Switch from a baseband to a passband view
- Make an end-to-end RF measurement
- Measure the one dB compression point

  The one dB compression point is usually a transmitter specification but it is used to demonstrate this flow because it is easier to set up.

1. Copy the original receiver model from the Circuit Optimizer analysis to a new schematic window. Copy everything from the LNA to the low pass filters.
2. Edit the properties of the IQ_demodulator (`IQ_demod_BB`) to set the last parameter, *flo*, to *-frf+flo1*. The baseband view does not need the local oscillator frequency but the passband view does.
3. Load the low pass filters with ports.
4. Connect a port across the LNA_BB inputs. Set the Frequency name to `fin`, the frequency to `frf`, and the amplitude to `power`. (Do not abbreviate `power` to `pwr` because `pwr` is a reserved variable and you do not get any warning. SpectreRF may complain about a mysterious indexed undefined variable that increments from run to run.)
5. Add loaded voltage-controlled-voltage sources as shown in Figure B-16 on page 2228 to observe intermediate differential voltages.

Figure B-16 Passband View

6. Check and save the schematic
7. Close the schematic window.
8. Bring up a Library Manager and select but do not open the schematic you just created.
9. In the Library Manager window click *File – New – Cell View*.
10. In the window that comes up, select Hierarchy Editor for the Tool. Click *OK*.
11. Type in `schematic` for the View then click *Use Template*.
12. Set the *Name* to `Spectre`.
13. Enter `veriloga_PB` as the first item in the View List, then click *OK*.
14. In the Hierarchy Editor window, click *File – Save*. This is important.
15. Click the *Open* button to bring up the schematic.
16. Bring up an Analog Design Environment tool and use the states from the last Circuit Optimizer iteration.
17. Recall the states you saved from the last Circuit Optimizer iteration. Add the `power` variable and set it to -16.
18. Delete the previous analyses and set up a PSS analysis. Enter a new Fundamental Tone named `LO` and make it 2.354GHz. *Auto Calculate* the *Beat Frequency* and type `1` for the *Number of harmonics*.
19. Run the analysis and observe the intermediate differential voltages. The model is indeed now a passband model. At the higher power levels the LNA output contains odd

harmonics of the RF carrier. The filter reflects the odd harmonics back to the LNA and does not let them propagate forward. The baseband model does not simulate the odd harmonics but it does simulate the intermodulation term between the second harmonic and fundamental that falls at the fundamental. One reason to simulate the passband view is to check for peak voltage levels that might exceed voltage ratings. The baseband models only simulate peak voltage at the carrier fundamental, not the absolute peak.

20. Set up a swept PSS analysis. Sweep *power* from -32 to 0 in 10 steps.

21. After the sweep finishes, click *Results – Direct Plot* in the Environment window, select *Compression point*, *1dB*, *Output Referred*. Select the 0 harmonic because the end-to-end system produces a baseband output. Then click the port loading the top output low pass filter. You should see the compression point plot shown in Figure B-17 on page 2229.

Figure B-17 End-to-end RF measurement, one dB compression point at the I-baseband output.

22. Repeat the last step but this time click the lower output port. You should see the compression plot in Figure B-18 on page 2230.

Figure B-18 End-to-end RF measurement, one dB compression point at the Q-baseband output.

Comparing Baseband and Passband Models

This section illustrates how to compare baseband and passband models by:

1. Setting up a Transient analysis with the passband view
2. Setting up a Transient analysis with the baseband view
3. Directly comparing the baseband and passband models.

You run one analysis of the baseband view and two analyses of the passband view. You perform the second passband analysis with tightened tolerances.

1. Save the passband schematic under a different name. You use the new copy.
2. Repeat steps 9 through 17 from the last recipe for the new copy but do not enter the `veriloga_PB` view in the *View List* yet. You do a baseband analysis first.

3. Delete the port driving the LNA.
4. Delete the loaded voltage-controlled-voltage sources.
5. You need to synthesize an antenna signal. Add an `IQ_mod_BB` from the `top_dwnBB` category. Set the I and Q gains to 0 dB. Set the 1dB compression points to 1000 so that the modulator is ideal. Instantiate it in front of the LNA with the pins aligned, then wire the pins straight across. Ground the `phase_err` pin.
6. Drive the `I_in` pin of the IQ modulator with a port. Set the port frequency to 2MHz and name the frequency BB1. Set the amplitude to -16dBm.
7. Do the same for the `Q_in` modulator input.
8. Load or duplicate the states from the 12-iteration Circuit Optimizer analysis but delete the Noise and Transient analyses.
9. Remove AM/PM conversion from the LNA by setting the last parameter in the properties list to zero.

   It is not fair to compare passband and baseband views with AM/PM conversion because the passband view does not capture it.
10. Set up a 1us Transient analysis with default options.
11. Run the analysis and plot the filtered baseband outputs, the outputs of the low pass filters.

    Note how fast the simulation runs. Save the results so you can plot them again later. [1]
12. Switch to the passband view by entering “veriloga_PB” in the View List in the Hierarchy Editor. Click the update button in the hierarchy editor.
13. After you switch to the `veriloga_PB` view, edit the IQ modulator properties to set *flo* to `frf`. Edit the demodulator properties and set its *flo* to `flo1-frf`.
14. Click *Results – Printing Plotting Options* then click the *Overlay Plots* button.
15. Overlay the passband results with the baseband results. You see the waveforms in Figure B-19 on page 2232. The comparison is not very good.
16. Rerun the analysis with conservative options and set reltol to 1e-6. This run takes longer.

---

1. Note that the baseband outputs are out of phase with each other, even though the baseband inputs are in phase. In the baseband model, changing the RF-IF mixer LO from “flo1” to “-flo1” fixes the sign problem. In the passband model, the IQ_demodulator flo should be frf-flo1. To maintain the convention, in the baseband model the IF filter’s carrier frequency should be frf-flo1.

17. Plot the results.

18. Recall the saved baseband results and overlay them with those from the last simulation. You see the waveforms in Figure B-20 on page 2233. The passband results now lay right on top of the baseband results but took much longer to compute! It was not obvious without the baseband results that the first passband simulation did not run with tight enough numerical tolerances.

Figure B-19 Passband and baseband results with default options in the passband analysis

Figure B-20 Passband and baseband results with tighter options in the passband analysis

Relationship Between Baseband and Passband Noise

Noise analysis at baseband can be confusing because factors of two appear in a number of places throughout the calculations. For example:

- Each passband node becomes two equivalent baseband nodes.
  - Does noise injected at a passband node split between the two equivalent baseband nodes?
  - If so, does the noise split evenly?
- As shown in the `BB_test_bench` example, baseband models simulate peak in-phase and peak quadrature components of the carrier.

  When analyzing signal-to-noise ratios, does that mean you have to use half the square of the baseband signals?
- The analog design environment displays single sided power spectral densities.

Because the baseband power spectral density is the two-sided passband density shifted down, is there another factor of two because we can only see the baseband density for positive frequencies?

- White noise at the input of a mixer mixes up to the carrier from DC but there is also noise at twice the carrier that mixers down to the carrier.

  Does the baseband model account for this?

These questions are answered in this section.

Before sorting out the factors of two, note that baseband noise analysis is valid only for small signals. If any element in the architecture operates in a non-linear fashion, the noise analysis might be inaccurate. This is due to the fact that a baseband noise analysis follows a DC operating point analysis, rather than a PSS analysis.

Instantaneous incremental gain in a passband static non-linear model dithers at the carrier frequency.

- When the carrier swings through zero, the incremental gain is large and noise at the input is amplified.
- When the carrier reaches its peak and drives the circuit into saturation, the incremental gain is smaller.

The average gain is greater than the minimum gain. The baseband model remains in the non-linear region because it only simulates peak voltages. Consequently, the incremental gain is always a minimum and the baseband model under-estimates the amount of noise propagating to the output. If the peak input signal drives the model into saturation, be sure to scale the baseband noise results accordingly.

Introduction to Analysis

The circuit discussed here is called `noise_test_circuit` and you can find it in the `rfExamples` library. The circuit looks as shown in Figure B-21 on page 2235.

Figure B-21 The noise_test_circuit in the rfExamples Library

The `noise_test_circuit` shows the relationship between baseband and passband noise. One branch consists of passband models. The other branch is a baseband equivalent of the first. You can assess noise at each of three observation points located in each branch of the circuit. At each observation point, you can examine both the noise and pnoise summaries.

The I and Q inputs are both driven by the same DC source so that you only have to view one baseband output, the other baseband output is identical by symmetry. Noise parameters in the passband and baseband models are identical. Aside from the behavioral blocks at the end of each branch, each behavioral block has noise injected at its input.

Preparation Steps for Analyses

1. Set up a PSS analysis.

   Because the local oscillator is inside the passband mixer models you have to manually enter the frequency (1GHz) into the PSS analysis form. Let the beat frequency be *autocalculated* and use 1 harmonic.

2. Set up a pnoise analysis.

   Set the start frequency equal to 0 Hz, the stop frequency equal to 100MHz, use a linear sweep with 100 steps. Set the Maximum sideband to 1. For the input source select *none*

and for the output use *voltage*. For the positive output node, select the I-input of the IQ_modulator in the passband branch. Use ground for the negative node.

3. Set up a noise analysis.

Sweep the frequency from 0 to 100MHz linearly in 100 steps. Use voltage for the output noise and select the I-input of the *IQ_mod_BB* component in the baseband branch for the positive node. Select ground for the negative node. Set the input noise port to any one of the ports in the circuit. Noise from that port does not affect either passband or baseband branches.

4. Run the analysis.

5. When the analysis finishes, go to the analog design environment simulation window and click *results – print – pss noise summary*.

6. In the form that appears, include All types, set *Type* to *integrated noise*, look at noise from 0 to 100MHz, select *truncate by number*, and view the top 2 noise contributors.

There are only 2 noise contributors at this point, noise at the I/Q inputs and noise from the low pass resistors.

7. Print the noise summary (which is different from the pnoise summary).

Again, print integrated noise (from 0 to 100MHz). Select All types and print noise from the top 2 noise contributors.

Figure B-22 on page 2237 shows the noise summaries. The two summaries agree because at this point in the circuit, both nodes are really baseband nodes.

Figure B-22 Noise Summaries for the First PSS pnoise and Noise Analyses

```
Device    Param            Noise Contribution    % Of Total

/I2       IQ_modulator_i   1.56676e-12           70.06
/R20      rn               6.69678e-13           29.94

Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 2.23644e-12
No input referred noise available


Device    Param          Noise Contribution    % Of Total
/I22      IQ_mod_BB_i    1.56676e-12           70.06
/R22      rn             6.69679e-13           29.94

Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 2.23644e-12
Total Input Referred Noise = Inf
```

8. Repeat the analyses.

This time select the `outi` pins on the low pass filters as outputs in the appropriate noise analyses. Again print the same noise summaries but this time look at the top 12 noise contributors in each summary.

Figure B-23 on page 2238 shows the noise summaries for the second analysis. The top 7 noise contributors in the baseband and passband branches agree. The remaining noise contributors are negligible, and should be negligible for the circuit because it has no AM/PM conversion.

Figure B-23 Noise Summaries for the Second PSS pnoise Analyses

```
Device      Param                     Noise Contribution    % Of Total

/I77        PA_PB                     7.49879e-10           82.29
/I2         IQ_modulator_i            9.69448e-11           10.64
/R20        rn                        4.14371e-11           4.55
/I87        butterworth_lp_i_noise    1.52928e-11           1.68
/I94        IQ_demodulator            7.64642e-12           0.84
/PORT13     rn                        3.77571e-14           0.00
/PORT14     rn                        3.77571e-14           0.00
/I2         IQ_modulator_q            2.33139e-15           0.00
/R21        rn                        9.96506e-16           0.00
/R22        rn                        0                     0.00
/R23        fn                        0                     0.00
/R23        rn                        0                     0.00

Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 9.11279e-10
No input referred noise available


Device      Param                     Noise Contribution    % Of Total
/I76        PA_HB_i                   7.49861e-10           82.29
/I22        IQ_mod_B3_i               9.69448e-11           10.64
/R22        rn                        4.14371e-11           4.55
/I86        butterworth_lp_i_noise    1.52928e-11           1.68
/I92        IQ_demod_HB_i             7.64642e-12           0.84
/PORT15     rn                        3.77571e-14           0.00
/PORT9      rn                        3.77571e-14           0.00
/I76        PA_HB_q                   1.80329e-14           0.00
/I22        IQ_mod_B3_q               2.33136e-15           0.00
/R23        rn                        9.96493e-16           0.00
/I92        IQ_demod_HB_q             1.54184e-31           0.00
/R23        fn                        0                     0.00

Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 9.11279e-10
Total Input Referred Noise = 0.0151541
```

9. Repeat the analysis again.

This time change the Pnoise sweep to run from 900MHz to 1.1GHz in 200 linear steps and select the power amplifier output as the noise output node. For the noise analysis, leave the sweep at zero to 100MHz, use 100 steps as before, and change the output node to be the I-output of the power amplifier.

Look at the top 7 noise contributors in each analysis. This time, integrate noise from 900MHz to 1.1GHz for the pnoise run and integrate noise from 0 to 100MHz for the noise run.

Figure B-24 on page 2239 shows the new summaries. Although the noise analyses agree at the ends of the branches, the noise analyses appear to disagree at a point where the baseband node is only a baseband equivalent, not a true baseband node.

Figure B-24 Noise Summaries for the Third PSS pnoise Analyses

```
Device      Param              Noise Contribution    % Of Total

/I77        PA_PB              2.3769e-09            83.70
/I2         IQ_modulator_i     1.53663e-10           5.41
/I2         IQ_modulator_q     1.53663e-10           5.41
/R20        rn                 6.568e-11             2.31
/R21        rn                 6.568e-11             2.31
/I84        IQ_demodulator     2.42376e-11           0.85
/PORT13     rn                 1.19682e-13           0.00

Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 2.83995e-09
No input referred noise available


Device     Param              Noise Contribution    % Of Total
/I76       PA_BB_i            2.3769e-09            83.69
/I22       IQ_mod_BB_i        3.07294e-10           10.82
/R22       rn                 1.31347e-10           4.62
/I82       IQ_demod_BB_i      2.42376e-11           0.85
/PORT9     rn                 1.19682e-13           0.00
/I76       PA_BB_q            5.71606e-14           0.00
/I22       IQ_mod_BB_q        7.38993e-15           0.00

Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 2.83997e-09
Total Input Referred Noise = 0.0148587
```

The *apparent* disagreement shown in Figure B-24 on page 2239 requires an explanation. Let us examine the noise contributors and try to answer some the questions we posed earlier.

Noise at the power amplifier output due to noise injected at a passband node

- Passband model contributors: *PA_PB* and *IQ_demodulator*, port 9.
- Baseband model contributors: *PA_BB_i*, *IQ_demod_BB_i*, port 13.

Passband and baseband counterparts contribute the same amount of noise. However, in the baseband model, from symmetry you see the same numbers if you look at the Q-node. This means the baseband model predicts twice as much total noise due to noise injected between the modulators.

This factor of two is intentionally introduced to maintain the correct signal-to-noise ratio. The baseband model simulates peak signals; the carrier is suppressed. Without the carrier, signal power equals the square of the peak rather than one half of the square. This factor of two is not as arbitrary as it seems. The baseband model predicts the correct noise after demodulation because the passband demodulator model includes an extra factor of two to offset the factor of two inherent in the demodulation process.

Let the modulated carrier be `i(t)*cos(wc*t) -q(t)*sin(wc*t) + noise(t)`.

Now consider the I-output. To generate the I-output, the demodulator multiplies the signal by `cos(wc*t)`. The only part that propagates through the subsequent filter is `(1/2)i(t) + noise(t)*cos(wc*t)`. To recover `i(t)`, the passband demodulator model must scale this sum by two. (The baseband demodulator does not need to scale by two to extract the baseband signal because the carrier is suppressed.)

Thus, noise at passband demodulator model output equals `2*noise(t)*cos(wc*t)`. The filtered noise power density is then `4*(`*`input noise density`*`)/2`. The factor of 1/2 comes from the cosine. The filtered output noise density is twice the input noise density.

In the baseband model, doubling the noise injected at the passband nodes was not simply a matter of convenience.

So, to answer questions 1 and 2,

- Yes, noise injected at a passband node splits evenly between the two equivalent baseband nodes

  but

- Because each split is doubled, the ratio of *peak* signal to total noise equals the true signal-to-noise ratio.

Question 3 is rendered moot by using the noise summary and integrating over the proper band. If the pnoise analysis only integrated from 1GHz to 1.1GHz, (instead of from 900MHz to 1.1GHz), there would be a mysterious factor of two error.

In the baseband model, phase noise entering on the phase error pin propagates to both the I and Q outputs. In the baseband model, the same noise power appears on just the one output. Again, the total noise in the baseband model is twice that of the passband model to maintain the correct signal-to-noise ratio.

Noise injected at the modulator input (resistors and modulator noise)

Total noise in the baseband model due to modulator and input resistor noise is twice what it is from just one phase. Thus, the total noise due to sources on the input sides of the modulators differ by a factor of two. This occurs because the passband model is a real multiplier, which modulates the noise. If the peak signal voltage agrees with the baseband model, the passband modulator model attenuates input noise most of the time. The important thing is that the signal-to-noise ratios in the passband and baseband models agree anywhere in the system.

Now, copy the circuit, remove the capacitors at the modulator inputs, and repeat the last set of noise analyses. You see that in the passband model, the input resistors, R20 and R21, together contribute twice as much as the baseband counterpart, R22. With the capacitors, R20 and R21 together contribute just as much as R22. Without the capacitors, the input noise is truly white over the frequencies of interest. The same thing happens to the modulator noise itself. Figure B-25 on page 2242 shows the results.

In particular, the modulator now also has noise at twice the carrier frequency and that noise mixes down to the carrier frequency. The baseband model is just that, a baseband model. The answer to question 4 is *no*. The baseband models do not account for noise, or signals, at carrier harmonics. The baseband equivalent noise analysis is valid only if noise injected into the modulators has no power beyond the local oscillator frequency. Phase noise injected at the phase noise pins should also be band limited.

Figure B-25 Noise Summaries with the input capacitors removed.

```
Device     Param              Noise Contribution     % Of Total

/I2        IQ_modulator_q     4.22571e-09            29.21
/I2        IQ_modulator_i     4.22571e-09            29.21
/I77       PA_PB              2.3769e-09             16.43
/R21       rn                 1.80619e-09            12.49
/R20       rn                 1.80619e-09            12.49
/I84       IQ_demodulator     2.42376e-11            0.17

Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 1.44651e-08
No input referred noise available


Device     Param              Noise Contribution     % Of Total
/I22       IQ_mod_BB_i        4.22561e-09            50.11
/I76       PA_BB_i            2.3769e-09             28.19
/R22       rn                 1.80615e-09            21.42
/I82       IQ_demod_BB_i      2.42376e-11            0.29
/PORT9     rn                 1.19682e-13            0.00
/I22       IQ_mod_BB_q        1.01619e-13            0.00

Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 8.43322e-09
Total Input Referred Noise = 9.76997e-11
```

C

Documents That Ship in the Software Hierarchy

In addition to the SpectreRF documents visible through CDNSHelp, there are many documents that ship in the software hierarchy along with workshops. To find these documents, go to

`$MMSIMHOME/tools/spectre/examples/SpectreRF_workshop`

The structure of the `SpectreRF_workshop` directory is given in the following table. To make the best use of this information, look at the summaries of contents to determine which document contains the information you need. That document will tell you which tar ball contains the corresponding database. The tar balls are also included in the `SpectreRF_workshop` directory, but are not listed here.

Document or directory	For a brief summary of contents, see
`SpectreRF_simulink_example.pdf`	SpectreRF_simulink_example.pdf on page 2244
`SpectreRF_AN` directory	
`EnvelopeAN.pdf`	EnvelopeAN.pdf on page 2244
`HB_AN.pdf`	HB_AN.pdf on page 2247
`JitterAN.pdf`	JitterAN.pdf on page 2249
`LSSP_AN.pdf`	LSSP_AN.pdf on page 2245
`LTJM_AN.pdf`	LTJM_AN.pdf on page 2247
`MatlabAN.pdf`	MatlabAN.pdf on page 2249
`MatlabWorkshop.pdf`	MatlabWorkshop.pdf on page 2245
`NS_AN.pdf`	NS_AN.pdf on page 2246
`PerturbationAN.pdf`	PerturbationAN.pdf on page 2248

Document or directory	For a brief summary of contents, see
PLL_Jitter_AN.pdf	PLL_Jitter_AN.pdf on page 2246
PSRR_Drv_AN.pdf	PSRR_Drv_AN.pdf on page 2246
PSRR_Osc_AN.pdf	PSRR_Osc_AN.pdf on page 2248
PstbAN.pdf	PstbAN.pdf on page 2250
readme.txt	readme.txt on page 2247
RF_Blocks_AN.pdf	RF_Blocks_AN.pdf on page 2248

SpectreRF_simulink_example.pdf

Title: *Co-Simulation with SpectreRF MATLAB/Simulink Tutorial*

This document is a tutorial that describes how to run a co-simulation between SpectreRF and MATLAB/Simulink.

Tutorial 1 demonstrates Simulink, SpectreRF co-simulation while the SpectreRF simulator runs a tran analysis, for the system-level module of a wireless LAN. The three steps required for the co-simulation include:

1. Inserting and configuring the SpectreRF Engine in the Simulink schematic.
2. Setting up the netlist to adopt the SpectreRF and MATLAB co-simulation.
3. Running the co-simulation.

Tutorial 2 demonstrates Simulink, SpectreRF co-simulation while the SpectreRF simulator runs an ENVLP analysis. This tutorial illustrates:

1. The use mode of manually running Spectre with ADE and calling MATLAB from ADE.
2. The use mode of manually running Simulink and activating Spectre through Simulink.

EnvelopeAN.pdf

Title: *A User's Guide to Envelope Following Analysis Using SpectreRF*

This document is an application note that describes how to use envelope following analysis with SpectreRF. The topics covered include:

1. Envelope following algorithm

2. Using the envelope following analysis
3. Parameters used in the analysis
4. Plotting results
5. An automatic gain control (AGC) example
6. Comparing simulation and measurement results for a GSM power amplifier
7. Using autonomous envelope following, with examples
8. Simulating circuits with FM sources

LSSP_AN.pdf

Title: *Measuring Large-Signal S-Parameters Using SpectreRF*

This document illustrates how to measure large signal S-parameters using Spectre and SpectreRF from within ADE. The contents include:

1. Calculating LSSP in terms of input power
2. Calculating LSSP in terms of beat frequency
3. Comparing large and small-signal S-parameters

MatlabWorkshop.pdf

Title: *SpectreRF Workshop: Using the SpectreRF MATLAB Toolbox*

This document is a workshop that describes how to use SpectreRF and the SpectreRF MATLAB toolbox to measure parameters that are important in the design verification of low noise amplifiers (LNA). The contents include:

1. Measuring the small signal gain (SP) of an LNA
2. Conducting a large signal noise analysis (PSS and Pnoise)
3. Measuring gain compression and total harmonic distortion (Swept PSS)
4. Measuring IP3 (PSS plus PAC)
5. Measuring IP3 (QPSS analysis with Shooting or the Harmonic Balance engine)
6. Measuring IP3 (Rapid IP3 using AC analysis)

7. Measuring IP3 (Rapid IP3 using PSS Plus PAC)

PLL_Jitter_AN.pdf

Title: *Application Note. Pll jitter measurements*

This document illustrates how to use the Spectre and SpectreRF simulators with ADE to measure jitter characteristics of phase-locked loop (PLL) circuits. The contents include:

1. Using SpectreRF to simulate the individual blocks and measure the jitter and operating parameters of the models
2. Creating or modifying the behavioral models of the blocks to incorporate the jitter
3. Time domain simulation of the PLL using the behavioral models of the blocks.
4. Post processing of the simulation results to obtain the jitter and noise characteristics of the entire PLL.

NS_AN.pdf

Title: *The Application Note of Noise Separation in Pnoise/Qpnoise Analysis*

This document is an application note that describes how to use noise separation features in Pnoise and Qpnoise analysis to extract noise sources. The contents include:

1. The principles of noise separation in RF circuits
2. The graphical user interface used for noise separation
3. The flow to follow to determine which noise sources add the most noise to the output
4. Understanding the noise source summary report
5. An example is provided that illustrates how to locate the noise sources that contribute the most noise to the output in an NE600 mixer.

PSRR_Drv_AN.pdf

Title: *Power Supply Rejection Ratio Characterization Using SpectreRF Driven Circuits*

This document illustrates how to characterize the power supply rejection ratio (PSRR) of driven circuits. The contents include:

1. How to set up the PXF analysis
2. Using Direct Plot for the PXF analysis to plot the periodic transfer function

readme.txt

This text document lists the main topic of each of the `.pdf` files in the `SpectreRF_workshop`.

HB_AN.pdf

Title: *Harmonic Balance Release Note*

This document illustrates how to use the Harmonic Balance (HB) method, which is very efficient in simulating circuits with only low order harmonics, such as low-noise amplifiers. The contents include:

1. The parameters used to specify HB for driven circuit large signal simulations
2. How to use HB from within ADE
3. Discussion of when it is most appropriate to use time domain Shooting and when it is better to use HB

LTJM_AN.pdf

Title: *Application Note. Long term jitter measurements*

This document describes the procedures used to measure long term jitter characteristics from within the Analog Design Environment (ADE). Long term jitter characterizes the variation in the accumulated width of a large number of clock periods. The contents include:

1. Plotting the phase noise power spectral density, which will be used to compute the long term jitter
2. Selecting the number of cycles to use for the measurement
3. Selecting the minimum frequency of the integration. Long term jitter is described as very sensitive to the lower frequency limit that is used in the integration of the phase noise.

PerturbationAN.pdf

Title: *Perturbation Based Measurements Using SpectreRF*

This document describes how to use perturbation based rapid IP3, rapid IP2, the IM2 distortion summary, and the compression distortion summary to measure the compression distortion and intercept points of RF circuits. Contents include:

1. A discussion of intermodulation distortion
2. A discussion of compression distortion
3. A discussion of the perturbation method used in the analyses
4. An example using IP3 and compression distortion summary measurements of a mixer
5. An example using IP2 and IM2 distortion summary measurements of a Gilbert direct conversion mixer
6. An example using IPa3 and compression distortion summary measurements of a power amplifier

PSRR_Osc_AN.pdf

Title: *Power Supply Rejection Ratio Characterization Using SpectreRF Autonomous Circuits*

This document describes how to measure the power supply rejection ratio (PSRR) of oscillators and other autonomous circuits. Knowing PSRR values can help you protect against noise sources such as power supply ripples. The same methodology can be used to measure the effect of substrate noise. Contents include:

1. Setting up the PXF analysis to measure the PSRR of a VCO.
2. A discussion of the different definitions of PSRR
3. A discussion of confirming the results of the PXF analysis by taking quasi-static measurements of the tuning sensitivity and frequency pushing of the VCO

RF_Blocks_AN.pdf

Title: *Using Verilog-A Baseband Models of RF Blocks Application Note*

This document describes how to create and use Verilog-A signal flow and conservative baseband models of RF blocks. Contents include:

1. Several examples exploring the correspondence between passband (PB) and baseband (BB) representations of RF signals
2. Examples illustrating baseband modeling of linear time invariant blocks
3. Examples illustrating baseband modeling of nonlinear blocks using an amplifier
4. Code examples for Verilog-A modules, including
   - An ideal baseband to passband converter
   - A behavioral model of a passband to baseband converter
   - A behavioral model of a deviation frequency meter
   - A baseband model of a passband low pass filter with transfer function `H(s) = a/(a+s)`
   - A signal flow baseband model of a nonlinear RF block based on the complex BB gain approach

JitterAN.pdf

Title: *Jitter Measurements Using SpectreRF Application Note*

This document illustrates how to measure the jitter characteristics of typical blocks, working within ADE. Examples are presented for both driven and autonomous circuits. Contents include:

1. A discussion of how time (or phase) jitter is defined. RMS jitter and peak-to-peak jitter measurements are introduced
2. An example illustrating how to measure jitter for an autonomous circuit (differential VCO)
3. An example illustrating how to measure phase noise and jitter at the output of a divider, in a circuit that uses a VCO with divide-by-2 prescaler

MatlabAN.pdf

Title: *Spectre/RF Matlab Toolbox Application Note*

This document describes how to use SpectreRF MATLAB toolbox to read in a simulation result and perform standard RF measurements. Contents include:

1. Instructions for obtaining and setting up the toolbox
2. A list and description of the basic functions included in the toolbox:
   - `cds_srr`
   - `cds_evalsig`
   - `cds_plotsig`
   - `cds_harmonic`
   - `cds_interpsig`
   - `cds_fft`
   - `cds_compression`
   - `cds_ipn`
3. An example illustrating how to obtain the 1dB compression point, the first order harmonic signal, and the total harmonic distortion
4. A discussion of compatibility with the Aptivia MATLAB functions

PstbAN.pdf

Title: *Stability analysis of Linear Periodical Time-Varying Circuit using SpectreRF PSTB Analysis Application Note*

This document discusses how to use STB analysis to evaluate the stability of a linear periodical time-varying circuit. Contents include:

1. An example illustrating how to run an STB and PSTB analysis on a VCO with an inherent nonlinearity
2. An example illustrating how to examine the local stability of an injection-locked oscillator

An example illustrating how to examine the global stability of an injection-locked oscillator

Index

Numerics

1dB compression point
low noise amplifier, plotting 877

A

analysis
descriptions 127, 179, 229, 253

B

batch mode, setting in .cdsinit file 672
before Simulation value of Start MATLAB field 794
Bench Type 674

C

Cadence libraries, setting up 805, 806
conventions, typographic and syntax 29
conversion gain measurement, mixer 1047
conversion gain, plotting 1052
power supply rejection, plotting 1054
cosimulation
generating netlist for lower-level block 791
inputs, selecting 792
options, enabling 793
outputs, selecting 792
running 800
setting up and running 786
simulation response timeout, setting 789
socket mode for, setting 789
socket port 793
socket port for, setting 789
software required for 786
Spectre command used in 789
start method for 791, 794
starting ADE manually and MATLAB automatically 795
starting MATLAB and Spectre RF separately 795, 800
starting MATLAB manually and Spectre RF automatically 797, 801
starting Spectre RF manually and MATLAB automatically 801
stop time 795
with MATLAB, introduction to 786
coupler block, connecting to system-level Simulink schematic 787
current mismatch, effect on PFD-CP 703

D

delay mismatch, effect on PFD-CP 704
Designer's Guide to SPICE and Spectre 28
divider jitter, generating Verilog-A model of 672
documents, shipped in the software hierarchy 1413

E

Enable PLL Macro Model option 674
equivalent noise resistance, low noise amplifier, plotting 855
examples
low noise amplifier 819
1dB compression point, plotting 877
noise calculations 864
output voltage distribution 832
PXF analysis 894
S-parameter analysis 838
voltage standing wave ratio, plotting 845
S-parameter Noise analysis 846
equivalent noise resistance, plotting 855
load stability circles, plotting 857
minimum noise figure 852
noise circles, plotting 861
noise figure 852
source stability circles 859
third-order intercept point, plotting 887

voltage gain
calculation 824
plotting 829
mixer
conversion gain measurement 1047
conversion gain, plotting 1052
power supply rejection, plotting 1054
harmonic distortion measurement 993
intermodulation distortion measurement
with QPSS 1072
noise figure measurement 1020
1dB compression point 1056
noise figure measurement with PSP 1029
periodic S-parameter plots 1029
third-order intercept measurement with PSS sweep and PAC 1063
oscillator simulation, of differential oscillator 924
output noise, plotting 937
oscillator simulation, of tline3oscRF
phase noise, plotting 921
steady state solution, plotting 918

F

Fast Frac-N VCO template 678
Fast VCO template 678
flicker noise computation 264
Format — Port/Signal Displays — Signal Dimensions 790
Fourier analysis 127
fractional-N VCO template 678
frame size, specifying 788
freq_meter instance, for measuring PLL noise 717
frequency synthesizer, block diagram of 672

H

harmonic distortion measurement, mixer 993

I

input pins, number of in SpectreRF Engine block 788
intermodulation distortion measurement 182
with QPSS, mixer 1072
isnoisy parameter 849

J

jitter variation with phase difference, in VCO 705
jitter, divider 672

L

libpll_sh.so 673
libpllMMpsd_sh.so 673
libpllPPVoscModel_sh.so 673
libpllTTpfd_cpModel_sh.so 673
load stability circle
low noise amplifier, plotting 857
low noise amplifier
1dB compression point, plotting 877
noise calculations 864
output voltage distribution 832
PXF analysis 894
S-parameter analysis 838
voltage standing wave ratio, plotting 845
S-parameter Noise analysis 846
equivalent noise resistance, plotting 855
load stability circles, plotting 857
minimum noise figure 852
noise circles, plotting 861
noise figure 852
source stability circles 859
third-order intercept point, plotting 887
voltage gain
calculation 824
plotting 829
low noise amplifier, simulation example 819
lteratio parameter 140

M

macro-models
  PLL simulation with 717
Math Operations library 790
MATLAB
  design name 794, 796
  start command 794
  startup directory 794
maxacfreq parameter 141
maxsideband parameter 257
minimum noise figure, low noise amplifier 852
mixer
  conversion gain measurement 1047
    conversion gain, plotting 1052
    power supply rejection, plotting 1054
  harmonic distortion measurement 993
  intermodulation distortion measurement with QPSS 1072
  noise figure measurement 1020
    1dB compression point 1056
  noise figure with PSP measurement 1029
  periodic S-parameter plots measurement 1029
  third-order intercept measurement with PSS sweep and PAC 1063

N

netlist
  generating for cosimulation 791
  generating for lower-level block in cosimulation 791
  preparing for cosimulation in ADE 791
  preparing for cosimulation without using a GUI 798
newlink OutputParamsPXF 234
no command value, of Start MATLAB field 794
noise calculations, low noise amplifier 864
noise circles
  low noise amplifier, plotting 861
noise figure measurement 263
  low noise amplifier 852
  mixer 1020
    1dB compression point calculating 1056
noise figure measurement with PSP, mixer 1029
Noise Summary form 874
noise temperature 755
noisefile parameter 263
noisetemp parameter 263, 755
noisevec parameter 263
Normal VCO template 678
now command value
  in MATLAB cosimulation 794
  of Start MATLAB field 794
  using to test whether ADE starts MATLAB 794

O

oprobe parameter 263
oscillators
  differential oscillator 924
    output noise, plotting 937
  starting in simulations 903
  tline3oscRF
    phase noise, plotting 921
    steady state solution, plotting 918
  troubleshooting simulations for 982
oscmm model 681
oscmm Symbol 682
oscmm_fast symbol 683
oscmm_frac symbol 685
oscmm_frac_fast Symbol 688, 689, 691, 692, 694, 695, 697, 698, 700
output noise, plotting with oscillators 937
output pins, number of in SpectreRF Engine block 788
output voltage distribution, low noise amplifier 832
overview, Spectre RF analyses 2

P

PAC 179
PAC analysis 179
  intermodulation distortion measurement with 182
  synopsis 181
parameters
  leratio 140
  maxacfreq 141

maxsideband 257
noisefile 263
noisetemp 263
noisevec 263
oprobe 263
period 137
refsideband 255
reltol 140
saveinit 138
sidebands 257
steadyratio 140
tinit 137
tonset 136
tstart 136
tstop 137
values 264
parameters tonset 136
period parameter 137
periodic AC analysis, *See* PAC analysis
periodic noise analysis, *See* Pnoise analysis
periodic S-parameter plots, mixer 1029
periodic steady-state analysis. *See* PSS analysis
periodic transfer function analysis, *See* PXF analysis
pfd_cp_bench 706
PFD-CP
input and output 702
nonideal effects on performance of 702
PFD-CP block, generating Verilog-A model of 672
PFD-CP macro-model, extracting 701, 706
PFD-CP macro-model, using 710
PFDCP Sweep field, use of 708
phase noise, discussion of 631
and SpectreRF simulation 645
frequently asked questions 652
further reading 657
models 634
troubleshooting 647
phase noise, with oscillators 903
plotting 921
pins, input, in SpectreRF Engine block 788
pins, output, in SpectreRF Engine block 788
PLL Bench 674
PLL Macro Model Wizard 674
PLL macro-model test bench (fractional-N), schematic of 719
PLL macro-model test bench (integer-N), schematic of 718
PLL simulation, challenges presented by 670
PLL test benches, found in pllMMLib 672
pllmmnoise.vcsv, produced if PSD plugin is used 718
PLLs
simulating with macro-models 717
pllTTpfd_cp symbol 710
plugin, command for using with Spectre 673
Pnoise 253
Pnoise analysis 253
flicker noise computation 264
noise figure computation 263
synopsis 257
PSS 127
PSS analysis 127
PSS Noise Summary form 874
PXF 229
PXF analysis 229
low noise amplifier 894
synopsis 234

R

refsideband parameter 255
related documents 1413
reltol parameter 140
resistor noise generation 849

S

sample time, setting for block used in cosimulation 789
saveinit parameter 138
setting up the software 803
setting up the Cadence libraries 805
using the UNIX shell window 806
Setup — Matlab/Simulink — Log file 794
Setup — Matlab/Simulink — Setting 792
sidebands parameter 257
Simulation — Configuration Parameters 790
Simulation — Start 800, 801
simulation response timeout for cosimulation 789
socket mode, for cosimulation 789
socket port, for cosimulation 789
source stability circle

low noise amplifier 859
S-parameter analysis, low noise amplifier 838
noise 846
voltage standing wave ratio, plotting 845
S-parameter Noise analysis, low noise amplifier
equivalent noise resistance plotting 855
load stability circles plotting 857
minimum noise figure 852
noise circles plotting 861
noise figure 852
source stability circles plotting 859
S-parameter simulation data, plotting
equations for S-parameter calculator 769
Spectre command, for cosimulation 789
Spectre RF
analyses, overview 2
SpectreRF Engine
block, number of input pins in 788
block, number of output pins in 788
showing port labels for 789
SpectreRF-workshop 1413
Start MATLAB field
before Simulation value 794
no value 794
now value 794
values for 794
starting the oscillator 903
steady state solution, plotting with oscillators 918
steadyratio parameter 140
stop time, for cosimulation 795
synopses
PAC 181
Pnoise 257
PXF 234

T

third-order intercept point
low noise amplifier, plotting 887
measurement with PSS sweep and PAC 1063
tinit parameter 137
tonset parameter 136
Tools — Analog Environment 792
troubleshooting, with oscillator simulation 982
tstab parameter 136
tstart parameter 136
tstop parameter 137
tuning curve, of VCO 675
typographical conventions 29

U

using in SpectreRF simulation
port parameter types 725

V

values parameter 264
VCO
effect of buffers and prescalars on simulation 680
jitter variation with phase difference in 705
noiseless macro-model of 680
white noise supported by 680
VCO block, generating CMI macromodel of 672
VCO extraction 675
VCO internal noise 676
VCO macro-model 676
VCO macro-model, extracting 677
VCO sweep, setting 679
VCO templates 678
VCO test bench for macro-model extraction 678
vco_vdd 680
vco_vout 680
vco_vss 680
vco_vtune 680
VCO, turning curve of 675
Vctrl sweep range 680
VDD sweep 680
vddstart VCO sweep parameter, vddstop VCO sweep parameter, vddsteps VCO sweep parameter 679
View — Simulink library 790
voltage gain
calculation, low noise amplifier 824
plotting, low noise amplifier 829
VSS sweep 680

W

white noise, in VCO 680