

# Spectre<sup>®</sup> Circuit Simulator Reference

**Product Version 23.1**  
**September 2023**

## Spectre Circuit Simulator Reference

---

© 2023 Cadence Design Systems, Inc.

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

All rights reserved. Associated third party license terms may be found at <install\_dir>/doc/OpenSource/\*

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.

**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. Cadence is committed to using respectful language in our code and communications. We are also active in the removal and/or replacement of inappropriate language

## Spectre Circuit Simulator Reference

---

from existing content. This product documentation may however contain material that is no longer considered appropriate but still reflects long-standing industry terminology. Such content will be addressed at a time when the related software can be updated without end-user impact.

## Spectre Circuit Simulator Reference

---

**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

---

<a href="#">Preface</a> .....	11
<a href="#">Related Documents</a> .....	12
<a href="#">Typographic and Syntax Conventions</a> .....	12
<a href="#">References</a> .....	13
<a href="#">Additional Learning Resources</a> .....	14

## 1

<a href="#">Introducing the Spectre Circuit Simulator</a> .....	15
<a href="#">Spectre Circuit Simulator</a> .....	16
<a href="#">Spectre Circuit Simulator Features</a> .....	17
<a href="#">Spectre Netlist Format</a> .....	21
<a href="#">Benefits of Using the Spectre Circuit Simulator</a> .....	23
<a href="#">Spectre Accelerated Parallel Simulator</a> .....	25
<a href="#">Benefits of Spectre APS</a> .....	25
<a href="#">Spectre eXtensive Partitioning Simulator</a> .....	27
<a href="#">Benefits of Spectre XPS</a> .....	27

## 2

<a href="#">Command-Line Options</a> .....	29
<a href="#">Spectre Command-Line Options</a> .....	30
<a href="#">Default Values</a> .....	45
<a href="#">Default Parameter Values</a> .....	45

## 3

<a href="#">Analysis Statements</a> .....	47
<a href="#">AC Analysis (ac)</a> .....	48
<a href="#">ACMatch Analysis (acmatch)</a> .....	53
<a href="#">Alter a Circuit, Component, or Netlist Parameter (alter)</a> .....	58
<a href="#">Alter Group (altergroup)</a> .....	60
<a href="#">Check Parameter Values (check)</a> .....	63

## Spectre Circuit Simulator Reference

---

<u>Checklimit Analysis (checklimit)</u>	64
<u>Setting for Simulink-MATLAB co-simulation (cosim)</u>	67
<u>Costomerrpreset Analysis (customerrpreset)</u>	68
<u>DC Analysis (dc)</u>	69
<u>DC Device Matching Analysis (dcmatch)</u>	74
<u>Device Characterization (devchrz)</u>	80
<u>Immediate Set Dspf Options (dspf_options)</u>	81
<u>Envelope Following Analysis (envlp)</u>	82
<u>Harmonic Balance Steady State Analysis (hb)</u>	97
<u>HB AC Analysis (hbac)</u>	119
<u>HB Noise Analysis (hbnoise)</u>	127
<u>HB S-Parameter Analysis (hbsp)</u>	138
<u>HB Stability Analysis (hbstb)</u>	146
<u>HB XF Analysis (hbxf)</u>	151
<u>Circuit Information (info)</u>	158
<u>LF Analysis (lf)</u>	165
<u>Load Pull Analysis (loadpull)</u>	170
<u>Monte Carlo Analysis (montecarlo)</u>	175
<u>Noise Analysis (noise)</u>	196
<u>Optimize Analysis (optimize)</u>	202
<u>Immediate Set Options (options)</u>	203
<u>Periodic AC Analysis (pac)</u>	268
<u>Periodic Noise Analysis (pnoise)</u>	276
<u>Periodic S-Parameter Analysis (psp)</u>	287
<u>Periodic Steady-State Analysis (pss)</u>	295
<u>Periodic STB Analysis (pstb)</u>	323
<u>Periodic Transfer Function Analysis (pxf)</u>	328
<u>PZ Analysis (pz)</u>	336
<u>Quasi-Periodic AC Analysis (qpac)</u>	342
<u>Quasi-Periodic Noise Analysis (qpnoise)</u>	347
<u>Quasi-Periodic S-Parameter Analysis (qpssp)</u>	355
<u>Quasi-Periodic Steady State Analysis (qpss)</u>	363
<u>Quasi-Periodic Transfer Function Analysis (qpxf)</u>	382
<u>Reliability Analysis (reliability)</u>	388
<u>Deferred Set Options (set)</u>	406
<u>Shell Command (shell)</u>	412

## Spectre Circuit Simulator Reference

---

<u>S-Parameter Analysis (sp)</u>	413
<u>Stability Analysis (stb)</u>	420
<u>Reliability Stress Analysis (stress)</u>	429
<u>Sweep Analysis (sweep)</u>	434
<u>Time-Domain Reflectometer Analysis (tdr)</u>	439
<u>THERMAL Analysis (thermal)</u>	442
<u>Transient Analysis (tran)</u>	446
<u>Special Current Saving Options (uti)</u>	469
<u>Transfer Function Analysis (xf)</u>	471

## 4

<u>Other Simulation Topics</u>	477
<u>AHDL Linter Usage (ahdllint)</u>	479
<u>Using analogmodel for Model Passing (analogmodel)</u>	483
<u>Behavioral Source Use Model (bsource)</u>	485
<u>Checkpoint - Restart (checkpoint)</u>	494
<u>Configuring CMI Shared Objects (cmiconfig)</u>	496
<u>Built-in Mathematical and Physical Constants (constants)</u>	498
<u>Convergence Difficulties (convergence)</u>	500
<u>The dcopt command line option (dcopt)</u>	502
<u>encryption (encryption)</u>	503
<u>Expressions (expressions)</u>	506
<u>The fastdc command line option (fastdc)</u>	511
<u>Fault List for Transient Fault Analysis (faults)</u>	512
<u>Dynamic Force Voltage Node (force_voltage)</u>	514
<u>User Defined Functions (functions)</u>	516
<u>Global Nodes (global)</u>	517
<u>IBIS Component Use Model (ibis)</u>	518
<u>Initial Conditions (ic)</u>	525
<u>The Structural if-statement (if)</u>	526
<u>Include File (include)</u>	528
<u>Spectre Netlist Keywords (keywords)</u>	530
<u>Library - Sectional Include (library)</u>	533
<u>Library Multi-Technology Simulation Mode (lmts)</u>	534
<u>Tips for Reducing Memory Usage (memory)</u>	538

## Spectre Circuit Simulator Reference

---

<u>Multi-Technology Simulation Mode (mts)</u>	539
<u>Node Sets (nodeset)</u>	540
<u>Parameter Soft Limits (param_limits)</u>	541
<u>Netlist Parameters (parameters)</u>	544
<u>Parameter Set - Block of Data (paramset)</u>	547
<u>The postlayout command line option (postlayout)</u>	548
<u>Pspice include File (pspice_include)</u>	549
<u>Dynamic Release Voltage Node (release_voltage)</u>	550
<u>Tips for Reducing Memory Usage with SpectreRF (rfmemory)</u>	552
<u>Output Selections (save)</u>	557
<u>Savestate - Recover (savestate)</u>	560
<u>Sensitivity Analyses (sens)</u>	564
<u>Options for Sparam Standalone Checking and Fitting Tool (sparam)</u>	566
<u>SpectreRF Summary (spectrerf)</u>	567
<u>Stitch Flow Use Model (stitch)</u>	568
<u>Subcircuit Definitions (subckt)</u>	574
<u>Vec/Vcd/Evcd Digital Stimulus (vector)</u>	579
<u>Verilog-A Usage and Language Summary (veriloga)</u>	582

## 5

<u>Circuit Checks</u>	595
<u>Dynamic Subckt Instance Activity Check (dyn_activity)</u>	597
<u>Dynamic Active Node Check (dyn_actnode)</u>	599
<u>Dynamic Capacitor Voltage Check (dyn_capv)</u>	601
<u>Dynamic DC Leakage Path Check (dyn_dcpath)</u>	603
<u>Dynamic Delay Check (dyn_delay)</u>	606
<u>Dynamic Diode Voltage Check (dyn_diodev)</u>	609
<u>Dynamic Excessive Element Current Check (dyn_exi)</u>	611
<u>Dynamic Excessive Rise, Fall, Undefined State Time Check (dyn_exrf)</u>	613
<u>Dynamic Floating Node Statistical Check (dyn_float_tran_stat)</u>	616
<u>Dynamic Floating Node Induced DC Leakage Path Check (dyn_floatdcpath)</u>	622
<u>Dynamic Float Crosstalk Check (dyn_floatxtalk)</u>	631
<u>Dynamic Glitch Check (dyn_glitch)</u>	635
<u>Dynamic HighZ Node Check (dyn_highz)</u>	638
<u>Dynamic MOSFET Voltage Check (dyn_mosv)</u>	644



## Spectre Circuit Simulator Reference

---

<u>Dynamic Node Capacitance Check (dyn_nodecap)</u>	646
<u>Dynamic Noisy Node Check (dyn_noisynode)</u>	648
<u>Dynamic Oscillation Check (dyn_osc)</u>	650
<u>Dynamic Power Density Check (dyn_powerdensity)</u>	652
<u>Dynamic Pulse Width Check (dyn_pulsewidth)</u>	654
<u>Dynamic Resistor Voltage Check (dyn_resv)</u>	656
<u>Dynamic Rise Fall Edge Check (dyn_risefall)</u>	658
<u>Dynamic Setup and Hold Check (dyn_setuphold)</u>	661
<u>Dynamic Subckt Port Voltage/Current Check (dyn_subcktport)</u>	664
<u>Dynamic Subckt Port Power Check (dyn_subcktpwr)</u>	666
<u>Dynamic Uninitialized latch (dyn_uninilatch)</u>	669
<u>Static Capacitor Check (static_capacitor)</u>	671
<u>Static Capacitor Voltage Check (static_capv)</u>	673
<u>Static Coupling Impact Check (static_coupling)</u>	675
<u>Static DC Leakage Path Check (static_dcpv)</u>	677
<u>Static Diode Voltage Check (static_diodev)</u>	679
<u>Static ERC Check (static_erc)</u>	681
<u>Static Highfanout Check (static_highfanout)</u>	684
<u>Static HighZ Node Check (static_highz)</u>	686
<u>Static MOSFET Voltage Check (static_mosv)</u>	688
<u>Static NMOS to vdd count (static_nmos2vdd)</u>	690
<u>Static NMOS Forward Bias Bulk Check (static_nmosb)</u>	692
<u>Static Always Conducting NMOSFET Check (static_nmosvgs)</u>	695
<u>Static PMOS to gnd count (static_pmos2gnd)</u>	697
<u>Static PMOS Forward Bias Bulk Check (static_pmosb)</u>	699
<u>Static Always Conducting PMOSFET Check (static_pmosvgs)</u>	702
<u>Static RCDelay Check (static_rcdelay)</u>	704
<u>Static Resistor Check (static_resistor)</u>	707
<u>Static Resistor Voltage Check (static_resv)</u>	709
<u>Static Stack Check (static_stack)</u>	711
<u>Static Subckt Port Voltage Check (static_subcktport)</u>	713
<u>Static Transmission Gate Check (static_tgate)</u>	715
<u>Static Topology Check (static_topology)</u>	717
<u>Static Voltage Domain Conflict Check (static_vconflict)</u>	719
<u>Static Voltage Domain Device Check (static_voltldomain)</u>	721

A

References ..... 723

---

# Preface

---

This manual assumes that you are familiar with the development, design, and simulation of integrated circuits and that you have some familiarity with SPICE simulation. It contains information about the Spectre<sup>®</sup> circuit simulator.

Spectre is an advanced circuit simulator that simulates analog and digital circuits at the differential equation level. The simulator uses improved algorithms that offer increased simulation speed and greatly improved convergence characteristics over SPICE. Besides the basic capabilities, the Spectre circuit simulator provides significant additional capabilities over SPICE. Verilog<sup>®</sup>-A uses functional description text files (modules) to model the behavior of electrical circuits and other systems. Spectre RF Simulation option adds several new analyses that support the efficient calculation of the operating point, transfer function, noise, and distortion of common RF and communication circuits, such as mixers, oscillators, sample holds, and switched-capacitor filters.

This preface discusses the following topics:

- [Related Documents](#) on page -12
- [Typographic and Syntax Conventions](#) on page -12
- [References](#) on page 13

## Related Documents

The following can give you more information about the Spectre circuit simulator and related products:

- To learn more about the equations used in the Spectre circuit simulator, consult the *Spectre Circuit Simulator Components and Device Models Reference* manual.
- The Spectre circuit simulator is often run within the Cadence® 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 ADE Explorer User Guide*.
- For more information about using the Spectre circuit simulator with Verilog-A, see the *Verilog-A Language Reference* manual.
- If you want to see how SpectreRF is run under the analog circuit design environment, read *SpectreRF Simulation Option User Guide*.
- For more information about RF theory, see *SpectreRF Simulation Option Theory*.
- For more information about how you work with the design framework II interface, see *Design Framework II Help*.
- For more information about specific applications of Spectre analyses, see *The Designer's Guide to SPICE & Spectre*<sup>1</sup>.

## Typographic and Syntax Conventions

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

`literal`

Nonitalic words indicate keywords that you must enter literally. These keywords represent command (function, routine) or option names, file names and paths, and any other sort of type-in commands.

---

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

# Spectre Circuit Simulator Reference

## Preface

---

*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.

 **Important**

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

## References

Text within brackets ([ ]) are references. See [Appendix A, “References”](#) for more information.

## Additional Learning Resources

Cadence provides various [Rapid Adoption Kits](#) that you can use to learn how to employ Virtuoso applications in your design flows. These kits contain workshop databases, designs, and instructions to run the design flow.

Cadence offers the following training courses on the Spectre circuit simulator:

- [Spectre Circuit Simulator](#)
- [Spectre Simulations Using Virtuoso ADE](#)

For further information on the training courses available in your region, visit the [Cadence Training](#) portal. You can also write to [training\\_enroll@cadence.com](mailto:training_enroll@cadence.com).

**Note:** The links in this section open in a new browser. The course links initially display the requested training information for North America, but if required, you can navigate to the courses available in other regions.

---

# Introducing the Spectre Circuit Simulator

---

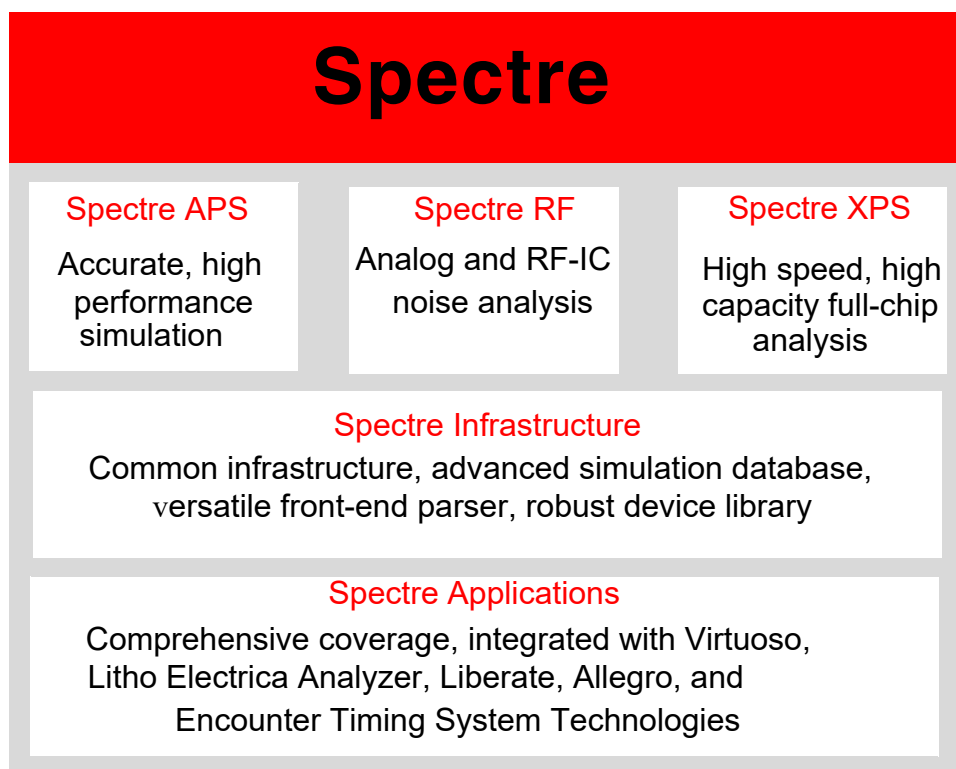
This chapter discusses the following topics:

- Spectre Circuit Simulator
  - Spectre Circuit Simulator Features
  - Spectre Netlist Format
  - Benefits of Using the Spectre Circuit Simulator
- Spectre Accelerated Parallel Simulator
  - Benefits of Spectre APS
- Spectre eXtensive Partitioning Simulator
  - Benefits of Spectre XPS

## Spectre Circuit Simulator

The Spectre® circuit simulator is a modern circuit simulator that provides high-precision SPICE simulation for pre- and post-layout analog RF and mixed-signal designs. Spectre is fully integrated with the Virtuoso custom design platform and provides a comprehensive set of detailed transistor-level analyses in multiple domains for faster convergence on the design goals. The advance architecture of Spectre enables low memory consumption and high-capacity analysis.

In addition to baseline simulation functionalities, Spectre supports the Accelerated Parallel Simulator (APS), and the eXtensive Partitioning Simulator (XPS) technologies that utilize the same Spectre simulation infrastructure — netlist format, analysis and options syntax, device models, output formats, feature functions, and so on.





## **Spectre Circuit Simulator Features**

The Spectre circuit simulator provides the following features.

### **Proven Circuit Simulation Techniques**

Spectre uses proprietary techniques — including adaptive time step control, sparse matrix solving, and multi-processing of MOS models — to provide high performance while maintaining sign-off accuracy. It includes native support for both Spectre and SPICE syntax, providing you the flexibility to use the Spectre technology for any design flow without worrying about the design format. In addition, it converges to results that are “silicon-accurate” by modeling extensive physical effects in devices for deep sub-micron processes.

### **Comprehensive Statistical Analysis**

Spectre bridges the gap between manufacturability and time to market nodes by providing a comprehensive set of statistical analysis tools tailored to IC design at advanced process nodes. Advanced Monte Carlo algorithms enable smart selection of process and design parameters to characterize the yield with significantly reduced simulation runs. The DC Match capability efficiently analyzes local process mismatch effects and identifies the yield-limiting devices and parameters. Tight integration between the Spectre Circuit Simulator and the Virtuoso Analog Design Environment offers user-friendly interactive setup and advanced visualization of statistical results.

### **Transient Noise Analysis**

Spectre provides transient noise analysis for accurate calculation of the large signal noise in nonlinear non-periodic circuits. All noise types are supported, including thermal, shot, and flicker.

### **Built-in Verilog-A and MDL**

The Spectre Circuit Simulator offers design abstraction for faster convergence on results, including behavioral modeling capabilities in full compliance with Verilog-A 2.0. The compiled Verilog-A implementation is optimized for compact device models, thus offering comparable performance to built-in device models.

In addition to supporting standard SPICE measurement functions (`.measure`), it offers a measurement description language (MDL) to automate cell and library characterization. Spectre MDL enables the designer to post-process the results and tune the simulator to provide the best performance/accuracy trade-off for a specific measurement.

### Advanced Device Modeling and Support

The Spectre Circuit Simulator supports MOS, BJT, specialty transistor models, resistors, capacitors, inductors, transformers and magnetic cores, lossy and lossless transmission lines, independent and controlled voltage and current sources, and Z and S domain sources.

The Spectre Circuit Simulator provides a user-defined compiled model interface (CMI). It allows for the rapid inclusion of user-defined models for a “model once, use everywhere” capability. It offers curve tracer analysis capability for rapid model development and debugging.

The Spectre circuit simulator supports the following models:

- MOSFET models, including the latest versions of BSIM3, BSIM4, PSP, HISIM, MOS9, MOS11, and EKV
- Silicon-on-insulator (Sol), including the latest versions of BTASOI, SSIMSOI BSIMSOI, BSIMSOI PD, and BSIM-IMG
- High-voltage MOSFET models, including the latest versions of HVMOS, LDMOS, and HiSim\_HV
- TMI models from TSMC
- Bipolar junction transistor (BJT) models, including latest versions of VBIC, HICUM L0, HICUM L2, Mextram, HBT, and Gummel-Poon models
- GaAs MESFET models, including the latest versions of GaAs, TOM2, TOM3, and Angelov
- Rensselaer Polytechnic Institute (RPI)’s Poly and Amorphous Silicon Thin-Film models
- Diode, JFET, FinFET, and flash cell models
- Verilog-A compact device models
- Specialized reliability models (AgeMOS) for HCI and NBTI analysis

### RF Simulation

Spectre RF, an option to the Spectre Circuit Simulator, provides a set of comprehensive RF analyses built on two production-proven simulation engines: harmonic balance and shooting-Newton. Spectre RF supports all industry-standard models. Spectre RF provides the following capabilities:

- Harmonic balance-based analyses, optimized for high dynamic range, high-capacity circuits with distributed components

## Spectre Circuit Simulator Reference

### Introducing the Spectre Circuit Simulator

---

- Shooting-Newton-based analysis, optimized for strongly non-linear circuits
- Advanced fast envelope analysis supporting all analog and digital modulation techniques
- Rapid IP2 and IP3 calculation based on perturbation technology
- Periodic noise analysis for accurate calculation of noise in nonlinear time variant circuits with detailed analysis options including modulated noise, sampled noise, and jitter
- Full spectrum periodic noise that provides a fast and silicon-accurate Pnoise analysis for circuits with sharp transitions
- Noise and distortion summary to identify the contribution of each device to the total output noise, harmonic, or inter-modulation distortion
- Small signal analysis that includes AC, transfer function, S-Parameters, and stability based on a periodic or quasiperiodic operating point
- Monte Carlo, corner-case, and parametric sweep analysis

### Advanced Transmission Line Library

Signal-integrity issues can be difficult and time consuming to identify, analyze, and resolve for high-speed designs. The Spectre RF rftline (RF transmission line) library enables the designer to perform signal-integrity analysis of the design in context of the package and PCB trace.

Spectre rFTlineLib provides a comprehensive set of multi-layer transmission lines and models. Spectre rftline models are based on rigorous 2-D electromagnetic simulations and include state-of-the-art descriptions of dielectric and conductor losses, delivering accurate models that are tightly integrated into Virtuoso ADE. An intuitive and easy-to-use graphical editor provides the ability to accurately define and graphically capture the substrates.

### Wireless Analysis

The modern mobile platform with exponentially evolving wireless standards is increasing the complexity of wireless RFIC designs. To meet specification requirements and productivity goals, you must evaluate the system-level performance metrics in an integrated, automated, and easy-to-use simulation-based flow.

Spectre RF wireless analysis feature provides a fully automated flow integrated in Virtuoso ADE, enabling you to apply the standard-compliant modulation sources and measure the output to calculate system-level performance.

## **Spectre Circuit Simulator Reference**

### **Introducing the Spectre Circuit Simulator**

---

The simulation is based on an advanced, accurate, and fast envelope following algorithm in Spectre RF. The wire analysis is designed with the RFIC designer in mind. It provides an automated setup of simulation parameters and standard-specific post-processing, eliminating the hassle and tedious nature of working with changing wireless standard sources. Spectre RF wireless analysis provides a rich set of visualization that includes EVM, BER, and spectrum. A broad set of wireless standards-compliant library sources is supported.

### **Co-simulation with Simulink**

The MathWorks Simulink interface to Spectre Circuit Simulator offers system and circuit designers a unique integrated environment for design and verification. Designers can insert their analog and RF schematics and post-layout netlist directly in the system-level block diagram and run a co-simulation between Simulink and Spectre technologies. Designers can reuse the same Simulink testbench from system-level design to post-layout verification, minimizing unnecessary format conversion while maintaining accuracy throughout the design flow.

### **Multi-Mode Simulation Toolbox for MATLAB**

Multi-Mode Simulation toolbox for MathWorks MATLAB reads PSF and SST2 files directly in MATLAB. You benefit from the set of MATLAB mathematical functions to post-process simulation results from Spectre Circuit Simulator, Spectre APS, Spectre XPS, and AMS Designer. All sweep types are supported in the toolbox, including Monte Carlo and parametric. Special data structures are used to store RF signals and harmonics resulting from PSS and QPSS analysis. Furthermore, the Spectre Simulation toolbox complements the rich MATLAB libraries with communication product-specific post-processing functions such as Fast Fourier Transform, third-order intercept point, and 1dB gain compression point.

### **Post-layout Simulation**

The Spectre Circuit Simulator enables analog and RF block and subsystem post-layout verification with speed near that of pre-layout simulation. An accurate parasitic reduction technique enhances the simulation performance of parasitic-dominant circuits by a significant amount over traditional SPICE-level simulation.

The technology enables designers to trade off accuracy and performance using a simple user-friendly setup.

## Spectre Netlist Format

A netlist is an ASCII file that lists the components in a circuit, the nodes that the components are connected to, and the parameter values. The netlist is created in a text editor such as `vi` or `emacs` or from one of the environments that support the Spectre simulator. The Spectre simulator uses a netlist to simulate a circuit.

### Sample Netlist

```
// BJT ECP Oscillator
simulator lang=spectre

Iee (e 0)    isource dc=1mA
Vcc (cc 0)   vsource dc=5

Q1 (cc b1 e) npn
Q2 (out b2 e) npn

L1 (cc out) inductor l=1uH

C1 (cc out) capacitor c=1pf
C2 (out b1) capacitor c=272.7pF
C3 (b1 0) capacitor c=3nF
C4 (b2 0) capacitor c=3nF

R1 (b1 0) resistor r=10k
R2 (b2 0) resistor r=10k

ic cc=5

model npn bjt type=npn bf=80 rb=100 vaf=50 \
cjs=2pf tf=0.3ns tr=6ns cje=3pf cjc=2pf

OscResp tran stop=80us maxstep=10ns
```

← Comment (indicated by //)

← Indicates the file contains a Spectre netlist (see the next section). Place below first line.

Instance statements

← Control statement (sets initial conditions)

Model statement

← Analysis statement

## Spectre Circuit Simulator Reference

### Introducing the Spectre Circuit Simulator

---

#### Elements of a Spectre Netlist

The following table briefly explains the components, models, analyses, and control statements in a Spectre netlist.

Netlist Element	Description
Title Line	The first line is taken to be the title. It is used verbatim when labeling output. Any statement you place in the first line is ignored as a comment.
Simulation Language	The second line of the sample netlist indicates that the netlist is in the Spectre Netlist Language, instead of SPICE.
Instance Statements	<p>The next section in the sample netlist consists of instance statements. To specify a single component in a Spectre netlist, you place all the necessary information for the component in a netlist statement. Netlist statements that specify single components are called instance statements.</p> <p>To specify single components within a circuit, you must provide the following information:</p> <ul style="list-style-type: none"><li>■ A unique component name for the component</li><li>■ The names of nodes to which the component is connected</li><li>■ The master name of the component (identifies the type of component)</li><li>■ The parameter values associated with the component</li></ul>
Control Statements	The next section of the sample netlist contains a control statement, which sets initial conditions.

## Spectre Circuit Simulator Reference

### Introducing the Spectre Circuit Simulator

---

#### Model Statements

Some components allow you to specify parameters common to many instances using the `model` statement. The only parameters you need to specify in the instance statement are those that are generally unique for a given instance of a component.

You need to provide the following for a model statement:

- The keyword `model` at the beginning of the statement
- A unique name for the model (reference by master names in instance statements)
- The master name of the model (identifies the type of model)
- The parameter values associated with the model

#### Analysis Statements

The last section of the sample netlist has the analysis statement. An analysis statement has the same syntax as an instance statement, except that the analysis type name replaces the master name. To specify an analysis, you must include the following information in a netlist statement:

- A unique name for the analysis statement
- Possibly a set of node names
- The name of the type of analysis you want
- Any additional parameter values associated with the analysis

Refer to the *Spectre Classic Simulator, Spectre APS, Spectre X, and Spectre XPS User Guide* for detailed information about the elements of the Spectre Netlist.

## Benefits of Using the Spectre Circuit Simulator

The Spectre Circuit Simulator provides the following benefits:

- Provides high-performance, high-capacity SPICE-level analog and RF simulation with out-of-the-box tuning for accuracy and faster convergence.

## **Spectre Circuit Simulator Reference**

### Introducing the Spectre Circuit Simulator

---

- Facilitates the trade-off between accuracy and performance through user-friendly simulation setup applicable to most complex analog and custom-digital ICs.
- Enables accurate and efficient post-layout simulation.
- Supports out-of-the-box S-Parameter models, enabling simulation of complex n-port devices.
- Delivers signal integrity analysis capability with an advanced transmission line library and graphical editor.
- Provides a platform to measure and analyze system-level performance metric.
- Performs application-specific analysis of RF performance parameters (spectral response, gain compression, intermodulation distortion, impedance matching, stability, and isolation).
- Offers advanced statistical analysis to help design companies improve the manufacturability and yield of ICs at advanced process nodes without sacrificing time to market.
- Delivers fast interactive simulation setup, cross-probing, visualization, and post-processing of simulation results through tight integration with the Virtuoso Analog Design Environment.
- Ensures higher design quality using silicon-accurate, industry-standard, foundry-certified device models shared across the simulation engines.



## **Spectre Accelerated Parallel Simulator**

The Spectre Accelerated Parallel Simulator (Spectre APS) maintains baseline Spectre simulation accuracy. It fully supports all Spectre functionalities and provides advanced performance for the next generation of analog and RF simulations. It delivers significant scalable performance and capacity with accurate results across a broad range of complex analog, RF, and mixed-signal blocks. It also provides accurate results for sub-systems with sizes up to millions of transistors and passive parasitic elements. Spectre APS provides all the transistor-level analysis capabilities available in Spectre Circuit Simulator.

In addition, its proprietary parallel simulation technology delivers scalable multi-core processing capability on modern multi-core compute platforms. Spectre APS:

- Supports all analysis capabilities offered in Spectre Circuit Simulator.
- Offers advanced parallel simulation on a single multi-core compute platform.
- Supports distributed, advanced parallel simulation across a cluster of multi-core computer platforms.
- Enables parasitic stitching and reduction for post-layout design and verification, providing additional performance gain for analog and RF designs dominated by parasitics.
- Supports multi-core harmonic balance, shooting-Newton, and envelope analysis.
- Supports Electromigration and IR drop analysis.
- Supports static and dynamic circuit checks.

### **Benefits of Spectre APS**

Spectre APS provides the following benefits:

- Provides significant single-core performance with an identical use model and full Spectre accuracy for everyday simulation of complex and/or large block designs, leading to faster convergence.
- Enables high-precision simulation for large post-layout analog and RF designs and subsystems dominated by parasitic devices.
- Delivers scalable performance leveraging a single machine or cluster of machines with multi-core architectures, allowing higher levels of analog design integration and verification and a quick turnaround time on simulation.

## **Spectre Circuit Simulator Reference**

### Introducing the Spectre Circuit Simulator

---

- Enables fast and accurate analysis of complete transceivers and large post-layout RF IC blocks by significantly improving the performance and capacity of harmonic balance analysis using a multi-core compute platform.

## **Spectre eXtensive Partitioning Simulator**

Spectre® eXtensive Partitioning Simulator (Spectre XPS) is a newer generation transistor-level circuit simulator emphasizing high simulation performance and large simulation capacity, with a vision to fundamentally address the design and verification needs of full-chip low-power designs at advanced process nodes.

Spectre XPS incorporates a completely new circuit partitioning and multi-rate technology, and further extends the simulation performance and simulation capacity to effectively enable full-chip simulation at advanced process nodes. In particular, Spectre XPS supports a variation analysis capability to provide circuit designers practical assessment on design robustness.

The Spectre XPS simulation technology is integrated into the Spectre binary, sharing the same product infrastructure with Spectre and Spectre APS. The Spectre XPS use model is identical to that of Spectre and Spectre APS, with the netlist syntax, device models, analysis setups, and output formats being fully compatible. It provides transient simulation capability for SRAM timing and power analysis, EM/IR analyses with an advanced power network solver, advanced parasitic reduction for faster post-layout simulation, and a set of circuit-checking features.

### **Benefits of Spectre XPS**

Spectre XPS provides the following benefits:

- Provides high performance and capacity pre-and post-layout simulation for design and IP characterization at the block and chip level.
- Provides a comprehensive set of transistor-level electrical rule checks.
- Delivers advanced EM and IR drop analysis for optimal throughput.
- Supports large and complex post-layout designs delivering a significant reduction in simulation runtime compared to the traditional FastSPICE simulator.
- Uses proven Spectre use-model for simplified setting up and post processing of results.
- Fully integrated into the Virtuoso Analog Design Environment (ADE).

## **Spectre Circuit Simulator Reference**

### Introducing the Spectre Circuit Simulator

---

---

# **Command-Line Options**

---

This chapter describes the command-line options for Spectre.

## Spectre Command-Line Options

The `spectre` command takes the following syntax at the command line:

```
spectre options inputfile
```

If no options are specified, the Spectre® circuit simulator saves the `.print` file in the current working directory, and saves the `.measure` and `.mt0` files in the `.raw` subdirectory of the netlist directory.

The Spectre circuit simulator reads default values for all the command line arguments marked with a dagger (†) from the UNIX environment variable `%S_DEFAULTS`.

Command option	Description
<code>-help</code>	Lists the Spectre command options and their descriptions. In addition, lists the available components, analyses, and design checks. You can use <code>-h</code> as an abbreviation of <code>-help</code> .
<code>-help <i>name</i></code>	Displays help information for the specified component or analysis name. If <i>name</i> is <code>all</code> , the help information for all components and analyses is displayed. You can use <code>-h</code> as an abbreviation of <code>-help</code> .
<code>-helpsort <i>name</i></code>	Displays the help information for the specified component or analysis <i>name</i> and sorts all the parameters alphabetically. You can use <code>-hs</code> as an abbreviation of <code>-helpsort</code> .
<code>-helpfull <i>name</i></code>	Displays detailed information for the specified component or analysis <i>name</i> , including parameter types and range limits. You can use <code>-hf</code> as an abbreviation of <code>-helpfull</code> .
<code>-helpsortfull <i>name</i></code>	Displays detailed information for the specified component or analysis <i>name</i> , including parameter types and range limits. In addition, it sorts all parameters by name. You can use <code>-hsf</code> as an abbreviation of <code>-helpsortfull</code> .
<code>-param</code>	Ignores the file containing the suggested parameter range limits. You can use <code>-p</code> as an abbreviation of <code>-param</code> .

## Spectre Circuit Simulator Reference

### Command-Line Options

---

Command option	Description
<code>+param file</code>	Reads the specified <i>file</i> for suggested parameter range limits. You can use <code>+p</code> as an abbreviation of <code>+param</code> .
<code>+paramdefault file</code>	<p>Reads the default values of the parameters from the specified <i>file</i> and applies them. However, if same parameters are specified in the netlist, their values override the default values in the specified file. This option can be set multiple times. Each line in the parameter file should contain three entries: primitive name, parameter name, and value. For example:</p> <pre>tran start 2e-3</pre> <p>The primitive name is limited to analyses primitives, which includes the options analysis. The value should be a constant. Expressions are not allowed.</p>
<code>=paramdefault file</code>	Reads the specified file and ignores all previously specified files. Refer to <code>+paramdefault</code> for details.
<code>-paramdefault</code>	Disables the reading of any parameter defaults that have been specified.
<code>-log</code>	Displays the log information on the standard output (shell) only and does not copy it to a log file. You can use <code>-l</code> as an abbreviation of <code>-log</code> .
<code>+log file</code>	Displays the log information on the standard output (shell) and copies it to the specified log <i>file</i> . You can use <code>+l</code> as an abbreviation of <code>+log</code> .
<code>=log file</code>	Sends the log information to the specified <i>file</i> only and does not display it on the standard output (shell). You can use <code>=l</code> as an abbreviation of <code>=log</code> .
<code>-raw raw</code>	Saves the simulation results in the specified file or directory named <i>raw</i> . In <i>raw</i> , <code>%C</code> , which is specified in the directory or file name is replaced by a circuit name. You can use <code>-r</code> as an abbreviation of <code>-raw</code> .

## Spectre Circuit Simulator Reference

### Command-Line Options

Command option	Description
<code>-format <i>fmt</i></code>	Generates raw data in the format <i>fmt</i> . You can use <code>-f</code> as an abbreviation of <code>-format</code> . Possible values for <i>fmt</i> are <code>nutbin</code> , <code>nutascii</code> , <code>psfbin</code> , <code>psfascii</code> , <code>psfbinf</code> , <code>psfxl</code> , <code>sst2</code> , <code>fsdb</code> , <code>fsdb5</code> , <code>wdf</code> , <code>uwi</code> , and <code>tr0ascii</code> .
<code>+rtsf</code>	Enables the fast waveform viewing mode for <code>psf</code> output. You can use this option only if you have specified the <code>-f psfbin</code> , <code>-f psfbinf</code> or <code>-f psfxl</code> format options.
<code>-outdir <i>path</i></code>	Changes the default location of Spectre output files. It does not change the location of raw directory if explicitly specified with the <code>-raw</code> option, and of files that contain slashes in the name.
<code>-uwifmt <i>name</i></code>	Specifies a user defined output format. To specify multiple formats use <code>:</code> as a delimiter. This option is valid only when waveform format is defined as <code>uwi</code> using the <code>-format</code> option.
<code>-uwilib <i>lib</i></code>	Absolute path to the user-defined output format library. This option is used together with <code>-uwifmt</code> . Use <code>:</code> to specify more than one library.
<code>+checkpoint</code>	Turns on the checkpoint capability. You can use <code>+cp</code> as an abbreviation of <code>+checkpoint</code> .
<code>-checkpoint</code>	Turns off the checkpoint capability. You can use <code>-cp</code> as an abbreviation of <code>-checkpoint</code> .
<code>+savestate</code>	Turns on the savestate capability. You may use <code>+ss</code> as an abbreviation of <code>+savestate</code> .
<code>-savestate</code>	Turns off the savestate capability. You may use <code>-ss</code> as an abbreviation of <code>-savestate</code> .
<code>-recover</code>	Does not restart the simulation, even if a checkpoint file exists. You can use <code>-rec</code> as an abbreviation of <code>-recover</code> .
<code>+recover [=filename]</code>	Restarts the simulation from a checkpoint or savestate file. Savestate file will be used if both files exist. You can use <code>+rec [=filename]</code> as an abbreviation of <code>+recover [=filename]</code> .



## Spectre Circuit Simulator Reference

### Command-Line Options

Command option	Description
<code>-cols <i>N</i></code>	Sets screen width (in characters) to <i>N</i> . This is needed only if the simulator cannot determine screen width automatically, and if default value of 80 is not acceptable. Spectre cannot determine screen width if output is redirected to a file or a pipe. You can use <code>-c</code> as an abbreviation of <code>-cols</code> .  <b>Note:</b> Spectre cannot determine the screen width if the output is redirected to a file or a pipe.
<code>-colslog <i>N</i></code>	Sets the log-file width based on the number of characters specified. The default is 80 characters.
<code>-%<i>X</i></code>	In quoted strings within the netlist, replaces % <i>X</i> with nothing where <i>X</i> is any uppercase or lowercase letter.
<code>+%<i>Xstring</i></code>	In quoted strings within the netlist, replaces % <i>X</i> with <i>string</i> , where <i>X</i> is an uppercase or lowercase letter. You can modify the string by using the <code>:x</code> operators.  Example: <code>+%KmyFile.txt</code> , if the netlist contains <code>file=../models/%K</code> , the file path is expanded to <code>../models/myFile.txt</code> .
<code>+error</code>	Prints error messages.
<code>-error</code>	Does not print error messages.
<code>+varedefnerror</code>	Prints error messages if Verilog-A modules are redefined.
<code>+warn</code>	Prints warning messages on the screen.
<code>-warn</code>	Does not print warning messages on the screen.
<code>-maxwarns <i>N</i></code>	Maximum number of times a particular type of warning message will be issued per analysis. You may use <code>-maxw</code> as an abbreviation of <code>-maxwarns</code> .
<code>-maxnotes <i>N</i></code>	Maximum number of times a particular type of notice message will be issued per analysis. You can use <code>-maxn</code> as an abbreviation of <code>-maxnotes</code> .

## Spectre Circuit Simulator Reference

### Command-Line Options

Command option	Description
<code>-maxwarnstolog <i>N</i></code>	Maximum number of times a particular type of warning message will be printed to log file per analysis. You can use <code>-maxwtl</code> as an abbreviation of <code>-maxwarnstolog</code> .
<code>-maxnotestolog <i>N</i></code>	Maximum number of times a particular type of notice message will be printed to log file per analysis. You may use <code>-maxntl</code> as an abbreviation of <code>-maxnotestolog</code> .
<code>+note</code>	Prints notices on the screen.
<code>-note</code>	Does not print notices on screen.
<code>+info</code>	Prints informational messages.
<code>-info</code>	Does not print informational messages.
<code>-debug</code>	Does not print debugging messages.
<code>+debug</code>	Prints debugging messages.
<code>-timer</code>	Turn off all timers.
<code>+timer</code>	Turn on all timers.
<code>+diagnose</code>	Enables diagnostic mode that provides debugging information to help you resolve simulation issues related to performance or convergence.
<code>+diagnose_minstep=1e-12</code>	Prints debugging information if the step size is less than the specified value, while running transient analysis in diagnostic mode.
<code>+transteps</code>	Prints all transient steps in diagnostic mode.
<code>+diagnose_top=<i>N</i></code>	Prints top <i>N</i> ( <i>N</i> ≥2) signals that limit step size or cause convergence failures in diagnostic mode.
<code>+diagnose_fpe</code>	Identify devices that cause floating point exceptions and print the bias voltages of the devices during dc and transient analyses in diagnostic mode. This option enforces single thread simulation.
<code>+detect_negcap</code>	Identify negative capacitance when convergence difficulty happens during transient analysis in diagnostic mode.

## Spectre Circuit Simulator Reference

### Command-Line Options

Command option	Description
<code>+fix_bad_pivot</code>	Reorder matrix when a bad pivot is detected during matrix factorization in transient analysis.
<code>-slave &lt;cmd&gt;</code>	Starts the attached simulator using the command <i>cmd</i> .
<code>-slvhost &lt;hostname&gt;</code>	Runs the attached simulator on machine <i>hostname</i> . Defaults to local machine.
<code>-V</code>	Prints version information.
<code>-W</code>	Prints subversion information.
<code>-cmiversion</code>	Prints CMI version information.
<code>-cmiconfig file</code>	Reads the specified <i>file</i> for information to modify the existing CMI configuration.
<code>-alias &lt;name&gt;</code>	Gives <i>name</i> to the license manager as the name of the simulator invoked.
<code>-E</code>	Runs the C preprocessor on an input file. In SPICE mode, the first line in the file must be a comment.
<code>-D&lt;x&gt;</code>	Defines string <i>x</i> and runs the C preprocessor.
<code>-D&lt;x=y&gt;</code>	Defines string <i>x</i> to be <i>y</i> and runs the C preprocessor.
<code>-U&lt;x&gt;</code>	Clears string <i>x</i> and runs the C preprocessor.
<code>-I&lt;dir&gt;</code>	Runs the C preprocessor and searches the directory <i>dir</i> for include files.
<code>+sensdata &lt;file&gt;</code>	Sends the sensitivity analyses data to <i>file</i> .
<code>+multithread</code>	Enables the multithreading capability. Spectre automatically detects the number of processors and selects the proper number of threads to use. (See note on the <code>options</code> help page about using multithreading). <code>+mt</code> can be used as an abbreviation of <code>+multithread</code> .
<code>+multithread=N</code>	Enables the multithreading capability. <i>N</i> is the specified number of threads. A maximum of 64 threads are allowed. <code>+mt</code> can be used as an abbreviation of <code>+multithread</code> .

## Spectre Circuit Simulator Reference

### Command-Line Options

Command option	Description
<code>-multithread</code>	Disables the multithreading capability. By default, multithreading is disabled for Spectre. However, it is enabled for Spectre APS. <code>-mt</code> can be used as an abbreviation of <code>-multithread</code> .
<code>+mtmode &lt;value&gt;</code>	Selects the multithreading mode. Possible values are <code>passive</code> and <code>active</code> .
<code>-processor</code>	Sets the CPU affinity of a process similar to Linux <code>taskset</code> command. It specifies a numerical list of processors that may contain multiple items, separated by comma, for example, <code>-processor 0-3,5,7</code> . Specification of numerical value out of range for current system results in the process termination with <i>Invalid argument</i> error message. You can use <code>-proc</code> as an abbreviation of <code>-processor</code> .
<code>-interactive</code>	Runs Spectre in the non-interactive mode, that is, process the input file and then return. You can use <code>-inter</code> as an abbreviation of <code>-interactive</code> .
<code>+interactive</code>	Runs Spectre in the interactive mode based on the type specified. You can use <code>+inter</code> as an abbreviation of <code>+interactive</code> .
<code>+interactive=type</code>	Runs in the interactive mode of the type specified. You can use <code>+inter</code> as an abbreviation of <code>+interactive</code> . Possible values for <i>type</i> are <code>skill</code> or <code>mpsc</code> .
<code>+mpsession=sessionName</code>	Specifies the <i>sessionName</i> for running an interactive session using multiprocess SKILL (MPS). This option must be specified when using the <code>+interactive=mpsc</code> setting.
<code>+mpshost=sessionHost</code>	Specifies the name of the <i>sessionHost</i> that will be used for an interactive session using MPS.
<code>-64</code>	Runs the 64-bit Spectre binary.
<code>-32</code>	Runs the 32-bit Spectre binary.
<code>-mdlcontrol</code>	Ignores the MDL control file. You can use <code>-mdl</code> as an abbreviation of <code>-mdlcontrol</code> .

## Spectre Circuit Simulator Reference

### Command-Line Options

Command option	Description
<code>+mdlcontrol</code>	Runs Spectre with the default MDL control file. You can use <code>+mdl</code> as an abbreviation of <code>+mdlcontrol</code> .
<code>+mdlcontrole</code>	Runs Spectre with the default MDL control file. You can use <code>+mdle</code> as an abbreviation of <code>+mdlcontrole</code> .
<code>=mdlcontrol <i>file</i></code>	Specifies the location of the MDL control file. You can use <code>=mdl</code> as an abbreviation of <code>=mdlcontrol</code> .
<code>=mdlcontrole <i>file</i></code>	Specifies the location of the MDL control file. You can use <code>=mdle</code> as an abbreviation of <code>=mdlcontrole</code> .
<code>-checklimitfile <i>file</i></code>	Writes assert violations to <i>file</i> . In <i>file</i> , %C is replaced by the circuit name. You can use <code>-cl</code> as an abbreviation of <code>-checklimitfile</code> .
<code>-dochecklimit</code>	Disables the checklimit capability. You can use <code>-docl</code> as an abbreviation of <code>-dochecklimit</code> .
<code>+dochecklimit</code>	Enables the checklimit capability. You can use <code>+docl</code> as an abbreviation of <code>+dochecklimit</code> .
<code>-dynchecks</code>	Disables the dynchecks capability.
<code>+lqtimeout <i>value</i></code>	Specifies the duration (in seconds) for which Spectre should wait to retrieve a license. When set to 0, Spectre waits until license is available. You can use <code>+lqt</code> as an abbreviation of <code>+lqtimeout</code> .
<code>+lqsleep <i>value</i></code>	Specifies the wait duration between two attempts made by Spectre to check out a license when queuing. Setting the value to a positive number overrides the default wait duration of 30 seconds. You can use <code>+lqs</code> as an abbreviation of <code>+lqsleep</code> .
<code>+lqmmtoken</code>	Enables queuing for the token license capability. When specified, Spectre registers the token request with the license server and waits for authorization. In addition, Spectre ignores all non-token licenses during the wait time since only token licenses are queued.

## Spectre Circuit Simulator Reference

### Command-Line Options

Command option	Description
<code>+lsuspend</code>	Enables the license suspend/resume capability. When Spectre receives <code>SIGTSTP</code> it checks in all the licenses before it gets suspended. The licenses are checked out again when <code>SIGCONT</code> is received. You may use <code>+lsusp</code> as an abbreviation of <code>+lsuspend</code> .
<code>+lmode value</code>	Checks out the licenses according to the specified value during initialization phase. Possible values are <code>&lt;num&gt;</code> and <code>RF</code> . <code>&lt;num&gt;</code> is a numeric value greater than 1, which specifies the number of token licenses to be checked out together. For example, to run Spectre APS with the multi-core option, specify <code>+lmode 4</code> . <code>RF</code> checks out the required licenses for the RF functionality. This value is case insensitive. If a simulation requires more licenses than the value specified using <code>+lmode</code> , the additional licenses are checked out after the initialization phase.
<code>+lorder value</code>	<p>Specifies the license check out order. Values are case insensitive. Possible values are:</p> <p><code>PRODUCT</code>: Spectre attempts to check out the product+options combination licenses only.</p> <p><code>MMSIM</code>: Spectre attempts to check out Virtuoso_Multi_Mode_Simulation (Spectre MMSIM) tokens only.</p> <p><code>PRODUCT:MMSIM</code>: Spectre attempts to check out the product licenses first and then Spectre MMSIM tokens.</p> <p><code>MMSIM:PRODUCT</code>: Spectre attempts to check out the Spectre MMSIM tokens first and then product licenses.</p> <p>Default is <code>PRODUCT:MMSIM</code>.</p>
<code>-ahdlcom value</code>	<b>Note:</b> Starting with MMSIM15.1, support for the <code>-ahdlcom</code> option has been removed.
<code>-va,define MACRO[=value]</code>	Defines a macro with higher priority than the one defined in Verilog-A files.

## Spectre Circuit Simulator Reference

### Command-Line Options

Command option	Description
<code>-rf_ahdl_functionality</code>	Specifies the value for AHDL functionality in RF analysis. Possible values are 141 and 151. Default value is 151.
<code>-ahdllint [=value]</code>	Enables Verilog-A linter check. Possible values are <code>no</code> , <code>warn</code> , <code>static</code> , <code>error</code> , and <code>force</code> . The default value is <code>warn</code> . This option is not available in Spectre base. It is available only in Spectre APS and Spectre XPS.
<code>-ahdllint_maxwarn [=value]</code>	Specifies the maximum number of Verilog-A linter warning messages to be reported by the simulator for each message ID. The default value is 5.
<code>-ahdllint_summary_maxentries [=value]</code>	Specifies the maximum number of messages to display in dynamic linter summary. The default value is 25.
<code>-ahdllint_warn_id=value</code>	Specifies a message ID and applies the <code>-ahdllint_maxwarn</code> option only to the specified ID. This option should always be used with the <code>ahdllint_maxwarn</code> option and should be directly specified after it at the command line, otherwise, the option will be ignored.
<code>-ahdllint_log &lt;file&gt;</code>	Specifies the file name to which the Verilog-A linter messages need to be written. By default, the Verilog-A linter messages are written to the output log file.
<code>-ahdllibdir path</code>	Redirects the library files generated by AHDL to the specified path. If both <code>-ahdllibddir</code> and <code>-outdir</code> options are specified, the library files generated by AHDL are stored under the path specified by the <code>-ahdllibdir</code> option and other output files are stored in the path specified by the <code>-outdir</code> option.

## Spectre Circuit Simulator Reference

### Command-Line Options

Command option	Description
<code>-ahdlsourceramp</code>	Disables source ramping for Verilog-A models. Source ramping of Verilog-A models is enabled by default because it helps with convergence for most designs. However, in certain corner cases, the default behavior could lead to convergence difficulties, which can be avoided by using this option.
<code>-ahdlshipdbdir=<i>sharing_path</i></code>	Specifies the sharing directory for Verilog-A library compilation. If this option is specified, the simulator will search for reusable library in the <i>sharing_path</i> . If library is found, reuse it, otherwise, compile a new library in the local directory.
<code>-ahdlshipdbmode=read_only or create_or_update</code>	Indicates whether to create or update the library under <i>sharing_path</i> specified by <code>-ahdlshipdbdir</code> . Possible values are <code>read_only</code> and <code>create_or_update</code> . <code>read_only</code> is the default behavior for <code>-ahdlshipdbdir</code> appearing alone. <code>create_or_update</code> will copy the newly-built library in local directory to <i>sharing_path</i> if no any library already or if the library is outdated.
<code>+errpreset=<i>value</i></code>	Selects a predetermined collection of parameter settings. Possible values are <code>liberal</code> , <code>moderate</code> , and <code>conservative</code> .
<code>+aps</code>	Enables Spectre APS mode.
<code>+aps=<i>value</i></code>	Enables Spectre APS mode and overrides the <code>errpreset</code> value in all transient analyses to apply the specified value. Possible values are <code>liberal</code> , <code>moderate</code> , or <code>conservative</code> .
<code>++aps</code>	Enables <code>++aps</code> mode. Unlike the <code>+aps</code> mode, the <code>++aps</code> mode uses a different time-step control algorithm as compared to Spectre. This results in improved performance, while satisfying error tolerances and constraints.



## Spectre Circuit Simulator Reference

### Command-Line Options

Command option	Description
<code>++aps=<i>value</i></code>	Enables the ++aps mode and overrides the errpreset value in all transient analyses to apply the specified value. Possible values are <code>liberal</code> , <code>moderate</code> , or <code>conservative</code> .
<code>+preset=<i>value</i></code>	Enables Spectre X. The most accurate mode is <code>cx</code> , and the highest performing mode is <code>vx</code> . Possible values are <code>cx</code> , <code>ax</code> , <code>mx</code> , <code>lx</code> , <code>vx</code> .
<code>+postlpreset=<i>value</i></code>	Defines parasitic optimization in Spectre X. If not set, Spectre X enables parasitic optimization if post-layout content is detected. If set, parasitic optimization is enforced. The most accurate parasitic optimization is <code>cx</code> , and the highest performing parasitic optimization is <code>vx</code> . Parasitic optimization can be disabled with <code>off</code> . Possible values are <code>off</code> , <code>cx</code> , <code>ax</code> , <code>mx</code> , <code>lx</code> , <code>vx</code> .
<code>-preset_override</code>	Spectre X by default ignores solver options defined in the netlist. The option <code>-preset_override</code> forces Spectre X to honor solver options defined in the netlist. The option can also be used to honor individual solver options: <code>-preset_override=reltol,method</code> .
<code>+xdp=<i>value</i></code>	Enables Spectre X distributed, multi-processing simulation. Supports RSH/SSH in manual mode: <code>+xdp=rsh ssh +hosts="host1:1 host2:3"</code> . In farm mode, <code>+xdp</code> needs to be set and the job distribution system controls the host/core usage.

## Spectre Circuit Simulator Reference

### Command-Line Options

---

Command option	Description
<code>+hosts "&lt;hostname&gt;: number of processes"</code>	<p>Defines the machine specifications that will be used for distributed processing. This option must be used when you specify the <code>rsh</code> or <code>ssh</code> values with the <code>+xdp</code> option. The format of the machine specification is the same for both <code>rsh</code> and <code>ssh</code>. Each machine specification, which can be separated by the space delimiter contains the name of the machine and number of cores to be used on the machine in the following format:</p> <p><code>"host_name:number_of_processes"</code>.</p> <p>For example:</p> <p><code>+hosts "hostA:4 hostB:4"</code>.</p> <p>This setting specifies that four processes will be used on both <code>hostA</code> and <code>hostB</code> for performing the simulation.</p>
<code>+ms</code>	Enables Spectre MS mode.
<code>+xps</code>	Enables the Spectre XPS mode.
<code>+xps=s[value]</code>	Enables Spectre XPS SPICE accuracy mode. The optional value is the speed setting.
<code>+speed=value</code>	The simulation group speed setting. The lower number means conservative setting.

## Spectre Circuit Simulator Reference

### Command-Line Options

---

Command option	Description
<code>+query=value</code>	<p>Queries the recommended number of threads or licenses required to run simulation on the current computer or on another computer with similar configuration.</p> <p>Possible values are:</p> <ul style="list-style-type: none"><li>■ "mtinfo: prints the number of threads required to run the simulation</li><li>■ "meminfo: prints an estimate of the memory that will be required to run the simulation</li><li>■ "all: prints the recommended number of threads, the memory usage estimate, and all possible license combinations for the design used in simulation</li><li>■ "alllic: prints all possible license combinations for the design used for simulation</li><li>■ "tokenlic: prints the number of MMSIM tokens required. This is the default value.</li></ul>
<code>+liclog</code>	<p>Writes the license check-in and check-out information in the log file.</p>
<code>+disk_check [=value]</code>	<p>Suspend the simulation when available disk storage is less than N bytes. Here, N is the value specified using the optional [=value] of <code>+disk_check</code> or netlist option <code>disk_check_thresh</code>. For example, <code>+disk_check=1e10</code>.</p>
<code>-clearcache</code>	<p>Clears the Spectre cache directory <code>/home/&lt;username&gt;/ .cadence/mmsim</code>. You can use <code>-cc</code> as an abbreviation for <code>-clearcache</code>.</p>
<code>-disableCPP</code>	<p>Disable CPP preprocesses in situations where Spectre implicitly calls CPP, such as <code>-I</code> and <code>-D</code>. However, if <code>-E</code> is specified, <code>-disableCPP</code> has no effect.</p>

## Spectre Circuit Simulator Reference

### Command-Line Options

Command option	Description
<code>+dcopt [=value]</code>	<p>Speeds up the simulation in Spectre and Spectre APS for post-layout circuits that consist of a large number of parasitic resistors and capacitors, which require significant time in DC simulation. This option can also be used to solve the non-convergence issues in DC simulations of any other circuit.</p> <p><b>Note:</b> In some cases, the DC solution obtained using the <code>dcopt</code> may not be as accurate as the true DC solution.</p>
<code>+lite</code>	<p>Enables a lightweight simulation session where most or all of the computationally intensive features (device check asserts, info analysis, current probing, power probing, instance preservation, and so on) are disabled. By default, <code>++aps</code> is used as the simulation engine. This can be a useful tool in isolating the computational time used by simulation engine during a simulation performance debugging.</p>
<code>+config &lt;file&gt;</code>	<p>Enables the user to append all Spectre commands defined in <i>file</i> to the netlist. <i>file</i> may contain one or more Spectre netlist lines. Multiple <code>+config</code> on the same Spectre command line are supported. They are incrementally processed.</p>
<code>=config &lt;file&gt;</code>	<p>Enables the user to append all Spectre commands defined in <i>file</i> to the netlist. All <code>+config</code> statements on the same Spectre command line are ignored.</p>
<code>-config</code>	<p>Disable all <code>+config</code> statements on the same Spectre command line. Disables all preceding <code>=config</code> statements on the same Spectre command line.</p>
<code>+pre_config &lt;file&gt;</code>	<p>Enables the user to prepend all Spectre commands defined in <i>&lt;file&gt;</i> to the netlist. <i>&lt;file&gt;</i> may contain one or more Spectre netlist lines. Multiple <code>+pre_config</code> on the same Spectre command line are supported. They are incrementally processed.</p>

## Spectre Circuit Simulator Reference

### Command-Line Options

Command option	Description
<code>+top &lt;value&gt;</code>	When specified with a subcircuit name, all elements, models, and so on will be exposed to the top level and the subcircuit will not exist anymore. As a result, instantiation from this subcircuit will be illegal. Instances inside the subcircuit will automatically connect to the instance at the top level if they connect to the same node.
<code>-hpc</code>	Disables the hpc option.

If you do not specify an input file, the Spectre simulator reads from standard input. When `+/` pairs of `spectre` command options are available, the default is the first value given in the previous list. For further information about the percent code options, `+%` and `-%`, see *Chapter 15, “Managing Files”*, in the *Spectre Classic Simulator*, *Spectre APS*, *Spectre X*, and *Spectre XPS User Guide*.

## Default Values

The Spectre simulator reads default values for all the command line arguments marked with a dagger (†) from the UNIX environment variable `%S_DEFAULTS`. The name of the simulator as called replaces `%S`. Typically, this name is `spectre`, and the Spectre simulator looks for `spectre_DEFAULTS`. However, the name can be different if you move the executable to a file with a different name or if you call the Spectre simulator through a symbolic or hard link with a different name. This feature lets you set different default values for each name you use to call the Spectre simulator.

If the variable `%S_DEFAULTS` does not exist, `SPECTRE_DEFAULTS` is used instead. The command line arguments always override any specifications from the `options` statement in the circuit file. The `options` statement specifications, in turn, override any specifications in the environment variable.

## Default Parameter Values

Many Spectre parameters have default values, and sometimes you will need to know them so you can determine whether they are acceptable for your simulation. You can find the default values for component, analysis, and control statement parameters by consulting the documentation for the statement in Spectre online help (`spectre -h`). Values given for parameters in the online help are the default values.

## Spectre Circuit Simulator Reference

### Command-Line Options

---

The following examples show some defaults for different types of parameters from the Spectre online help:

---

Parameter	Description
<code>nf=1.0</code>	Forward emission coefficient.
<code>etchc</code>	Narrowing due to etching for capacitances.
<code>homotopy=all</code>	Method used when no convergence occurs on initial attempt of DC analysis. You can specify methods and their orders by specifying a vector setting such as <code>homotopy=[source ptran gmin]</code> . Possible values are <code>none</code> , <code>gmin</code> , <code>source</code> , <code>dptran</code> , <code>ptran</code> , and <code>all</code> .

---

### ***Related Topics***

[Description of Spectre Predefined Percent Codes](#)

[Customizing Percent Codes](#)

[Creating Filenames from Parts of Input Filenames](#)

---

# Analysis Statements

---

This chapter describes the parameters for various analyses in Spectre.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

## AC Analysis (ac)

### Description

AC analysis linearizes the circuit about the DC operating point and computes the response to a given small sinusoidal stimulus.

Spectre can perform AC analysis while sweeping a parameter. The parameter can be frequency, temperature, component instance parameter, component model parameter, or netlist parameter. If changing a parameter affects the DC operating point, the operating point is recomputed at each step. You can sweep the circuit temperature by giving the parameter name as `temp`, without a `dev` or `mod` parameter. In addition, you can sweep a netlist parameter by giving the parameter name without a `dev` or `mod` parameter. After the analysis is complete, the modified parameter returns to its original value.

### Syntax

```
Name ac parameter=value ...
```

### Parameters

<code>prevoppoint</code>	<code>no</code>	Use the operating point computed in the previous analysis. Possible values are <code>no</code> and <code>yes</code> .
--------------------------	-----------------	---

#### Sweep interval parameters

<code>start</code>	<code>0</code>	Start sweep limit.
<code>stop</code>		Stop sweep limit.
<code>center</code>		Center of sweep.
<code>span</code>	<code>0</code>	Sweep limit span.
<code>step</code>		Step size, linear sweep.
<code>lin</code>	<code>50</code>	Number of steps, linear sweep.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

dec		Points per decade.
log	50	Number of steps, log sweep.
values	[...]	Array of sweep values.
valuesfile		Name of the file containing the sweep values.

#### Sweep variable parameters

dev		Device instance whose parameter value is to be swept.
mod		Model whose parameter value is to be swept.
param		Name of parameter to sweep.
freq	(Hz)	Frequency when a parameter other than frequency is being swept.

#### State-file parameters

readns		File that contains an estimate of the DC solution (nodeset).
write		DC operating point output file at the first step of the sweep.
writefinal		DC operating point output file at the last step of the sweep.

#### Initial condition parameters

force	none	The set of initial conditions to use. Possible values are none, node, dev and all.
-------	------	--

## Spectre Circuit Simulator Reference

### Analysis Statements

---

readforce		File that contains initial conditions.
skipdc	no	Skip DC analysis. Possible values are no and yes.
useprevic	no	If set to yes or ns, use the converged initial condition from previous analysis as ic or ns. Possible values are no, yes and ns.

#### Output parameters

save		Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, none and nooutput.
nestlvl		Levels of subcircuits to output.
oppoint	no	Determines whether operating point information should be computed:if yes, where should it be printed (screen or file). Operating point information is not printed if the operating point computed in the previous analysis remains unchanged. Possible values are no, screen, logfile and rawfile.

#### Convergence parameters

restart	yes	Restart the DC solution from scratch if any condition has changed. If not, use the previous solution as an initial guess. Possible values are no and yes.
---------	-----	---

#### Annotation parameters

annotate	sweep	Degree of annotation. Possible values are no, title, sweep, status and steps.
title		Analysis title.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>perturbation</code>	<code>linear</code>	The type of AC analysis. Default is linear for normal AC analysis. <code>im2ds</code> is for im2 distortion summary and <code>ds</code> is for distortion summary. Possible values are linear, ds, ip3, ip2, im2ds, multiple_beat and triple_beat.
<code>flin_out</code>	0 Hz	Frequency of linear output signal.
<code>fim_out</code>	0 Hz	Frequency of IM output signal.
<code>out1</code>	NULL	Output signal 1.
<code>out2</code>	NULL	Output signal 2.
<code>contriblist</code>	NULL	Array of device names for distortion summary. When <code>contriblist=[""]</code> , distortion from each non-linear device is calculated.
<code>maxharm_nonlin</code>	4	Maximum harmonics of input signal frequency induced by non-linear effect.
<code>rfmag</code>	0	RF source magnitude.
<code>rfdbm</code>	0	RF source dBm.
<code>rf1_src</code>	NULL	Array of RF1 source names for IP3/IP2/IM2DistortionSummary.
<code>rf2_src</code>	NULL	Array of RF2 source names for IP3/IP2/IM2DistortionSummary.

You can define sweep limits by specifying the end points or the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. You can specify a step size parameter (`step`, `lin`, `log` or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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. All frequencies are in Hertz.

The small-signal analysis begins by linearizing the circuit about an operating point. By default, this analysis computes the operating point, if it is not known, or recomputes it if any significant

## Spectre Circuit Simulator Reference

### Analysis Statements

---

component or circuit parameter has changed. However, if an operating point was computed during a previous analysis, you can set `prevoppoint=yes` to avoid recomputing it. For example, if `prevoppoint=yes` and the previous analysis was a transient analysis, the operating point is the state of the circuit at the final time point.

Nodesets help find the DC or the initial transient solution. You can specify nodesets in the circuit description file with `nodeset` statements or in a separate file by using the `readns` parameter. When nodesets are specified, Spectre computes an initial guess of the solution by performing DC analysis, while forcing the specified values on to nodes by using a voltage source in series with a resistor whose resistance is `rforce`. Spectre then removes these voltage sources and resistors and computes the required solution from this initial guess.

Nodesets have two important uses. First, if a circuit has two or more solutions, nodesets can bias the simulator towards computing the required solution. Second, this is a convergence aid. By estimating the solution of the largest possible number of nodes, you might be able to eliminate a convergence problem or significantly speed up convergence.

When you simulate the same circuit multiple times, it is recommended that you use both `write` and `readns` parameters and assign the same file name to both parameters. DC analysis then converges quickly even if the circuit has changed since the last simulation, and the nodeset file is automatically updated.

During the initial operating point DC analysis, you may force certain circuit variables to use the values given in the `ic` file, `ic` statements, or `ic` parameter on the capacitors and inductors. The `ic` parameter controls the interaction of various methods of setting the force values. The effects of individual settings are as follows:

`force=none`: All initial conditions are ignored.

`force=node`: The `ic` statements are used, and the `ic` parameters on the capacitors and inductors are ignored.

`force=dev`: The `ic` parameters on the capacitors and inductors are used, and the `ic` statements are ignored.

`force=all`: Both `ic` statements and `ic` parameters are used, with the `ic` parameters overriding the `ic` statements.

If you specify an `ic` file with the `readforce` parameter, force values from the file are used and any `ic` statements are ignored.

After you specify the initial conditions, Spectre computes the DC operating point with the specified nodes forced to the given value by using a voltage source in series with a resistor whose resistance is `rforce` (see `options`).

## ACMatch Analysis (acmatch)

### Description

ACMatch analysis linearizes the circuit about the DC operating point and computes the variations of AC responses due to the statistical parameters defined in statistical blocks. Only mismatch parameters are taken into account.

### Syntax

```
Name [node1] [node2] acmatch parameter=value ...
```

### Parameters

#### Analysis parameters

meth	1e-3	Threshold of the relative contribution to be exported in output file. The ratio is determined by individual contribution divided by total contribution. Both magnitude and phase are taken into account.
oprobe		The mismatch variations of the current signal through this component is chosen as the output. It can be one of the following: vsource, inductor, switch, tline, iprobe, ccvs, vcvs, pccvs, and pvcvs.

#### AC frequency sweeping parameters

start	0	Start sweep limit.
stop		Stop sweep limit.
center		Center of sweep.
span	0	Sweep limit span.
step		Step size, linear sweep.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

lin	50	Number of steps, linear sweep.
dec		Points per decade.
log	50	Number of steps, log sweep.
values	[...]	Array of sweep values.
valuesfile		Name of the file containing the sweep values.

#### Output parameters

groupby	param	Choose the way of how to group the total sigma. Total sigma of each contributor can be grouped either by statistics parameters or by subckt instances. Possible values are param and inst.
where	screen	Specifies where the results should be printed. Possible values are screen, logfile, file and rawfile.
file		The name of the file in which the results are printed, if where=file.
save		Signal levels in output. Possible values are all, lvl, allpub, lvlpub, selected, none and nooutput.
oppoint	no	Determines whether operating point information should be computed. If yes, specifies where should it be printed (screen or file). Operating point information is not printed if the operating point computed in the previous analysis remains unchanged. Possible values are no, screen, logfile and rawfile.
write		DC operating point output file at the first step of the sweep.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>writefinal</code>		DC operating point output file at the last step of the sweep.
-------------------------	--	---

#### Initial condition parameters

<code>readns</code>		File that contains an estimate of DC solution (nodeset).
<code>prevoppoint</code>	<code>no</code>	Determines whether to use the operating point computed in the previous analysis. Possible values are no and yes.
<code>force</code>	<code>none</code>	The set of initial conditions to use. Possible values are none, node, dev and all.
<code>readforce</code>		File that contains the initial conditions.
<code>skipdc</code>	<code>no</code>	Skip DC analysis. Possible values are no and yes.
<code>useprevic</code>	<code>no</code>	If set to yes or ns, use the converged initial condition from the previous analysis as ic or ns. Possible values are no, yes and ns.
<code>restart</code>	<code>yes</code>	Restart the DC solution from scratch if any condition has changed. If not, use the previous solution as an initial guess. Possible values are no and yes.

#### Annotation parameters

<code>annotate</code>	<code>sweep</code>	Degree of annotation. Possible values are no, title, sweep, status and steps.
<code>title</code>		Analysis title.

ACMatch takes each parameter defined in the statistics blocks and applies variation one at a time to find the sensitivity of the output with respect to the parameter. Based on the sensitivities computed by this procedure, the total variation of the output is computed by

## Spectre Circuit Simulator Reference

### Analysis Statements

---

adding up variation contributions from each parameter, assuming that the parameters are mutually independent.

You can define sweep limits by specifying the end points or the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. You can specify a step size parameter (`step`, `lin`, `log` or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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. All frequencies are in Hertz.

The small-signal analysis begins by linearizing the circuit about an operating point. By default, this analysis computes the operating point, if it is not known, or recomputes it if any significant component or circuit parameter has changed. However, if an operating point was computed during a previous analysis, you can set `prevoppoint=yes` to avoid recomputing it. For example, if `prevoppoint=yes` and the previous analysis was a transient analysis, the operating point is the state of the circuit at the final time point.

Nodesets help find the DC or the initial transient solution. You can specify nodesets in the circuit description file with `nodeset` statements or in a separate file by using the `readns` parameter. When nodesets are specified, Spectre computes an initial guess of the solution by performing DC analysis, while forcing the specified values on to nodes by using a voltage source in series with a resistor whose resistance is `rforce`. Spectre then removes these voltage sources and resistors and computes the required solution from this initial guess.

Nodesets have two important uses. First, if a circuit has two or more solutions, nodesets can bias the simulator towards computing the required solution. Second, this is a convergence aid. By estimating the solution of the largest possible number of nodes, you might be able to eliminate a convergence problem or significantly speed up convergence.

When you simulate the same circuit multiple times, it is recommended that you use both `write` and `readns` parameters and assign the same file name to both parameters. DC analysis then converges quickly even if the circuit has changed since the last simulation, and the nodeset file is automatically updated.

During the initial operating point DC analysis, you may force some of the circuit variables to the values given in the `ic` file, `ic` statements, or `ic` parameter on the capacitors and inductors. The `ic` parameter controls the interaction of various methods of setting the force values. The effects of individual settings are as follows:

`force=none`: All initial conditions are ignored.

`force=node`: The `ic` statements are used, and the `ic` parameters on the capacitors and inductors are ignored.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

`force=dev`: The `ic` parameters on the capacitors and inductors are used, and the `ic` statements are ignored.

`force=all`: Both `ic` statements and `ic` parameters are used, with the `ic` parameters overriding the `ic` statements.

If you specify an `ic` file with the `readforce` parameter, force values from the file are used and any `ic` statements are ignored.

After you specify the initial conditions, Spectre computes the DC operating point with the specified nodes forced to the given value by using a voltage source in series with a resistor whose resistance is `rforce` (see `spectre -h` options).

## Alter a Circuit, Component, or Netlist Parameter (alter)

### Description

The `alter` statement changes the value of any modifiable component or netlist parameter for any analyses that follow. The parameter to be altered can be circuit temperature, a device instance parameter, a device model parameter, a netlist parameter, or a subcircuit parameter for a particular subcircuit instance. You can:

alter the circuit temperature by giving the parameter name as `param=temp` without a `dev`, `mod` or `sub` parameter.

alter a top-level netlist parameter by giving the parameter name without a `dev`, `mod` or `sub` parameter.

alter a subcircuit parameter for a particular subcircuit instance by specifying the subcircuit instance name with the `sub` parameter, and the subcircuit parameter name with the `param` parameter.

Each `alter` statement can change only one parameter.

### Syntax

```
Name alter parameter=value ...
```

### Parameters

<code>mod</code>	Device model.
<code>dev</code>	Device instance.
<code>sub</code>	Subcircuit instance.
<code>param</code>	Name of parameter to be altered.
<code>value</code>	New value for parameter.
<code>annotate</code>	Degree of annotation. Possible values are <code>no</code> and <code>title</code> .

## Spectre Circuit Simulator Reference

### Analysis Statements

---

reelaborate	yes	Activates the re-elaboration process, which triggers a related expression evaluation throughout the circuit. To speed up simulation, disable this parameter by setting it to no in successive alter statements. Possible values are no and yes.
subckt		Subcircuit definition.
vary		Variation parameter name, used to set the value for the specified instance, similar to mismatch value. The specified value will be held until changed any another vary or a montecarlo analysis..
justignore		ignore dangling instance.
component		component instance.

## Alter Group (altergroup)

### Description

The `altergroup` statement changes the values of any modifiable model, instance or netlist parameter for any analyses that follow. Within an alter group, you can specify model statements, instance statements, parameter statements and options statements (only supports `temp`, `tnom`, and `scale`). These statements should be bound within braces. The opening brace is required at the end of the line defining the `altergroup`. Altergroups cannot be specified within subcircuits. The following statements are not allowed within altergroups (`analyses`, `export`, `paramset`, `save`, and `sens`).

Within an altergroup, each device (instance or model) is first set to default and then the device parameters are updated. For netlist parameters, the expressions are updated and evaluated.

For subcircuit within altergroup, all instances of the subcircuits are modified when running altergroup. There are strict checks that do not allow changes to topology.

You can include files into the altergroup and can use the `simulator lang=spice` directive. See `spectre -h include` for more information. A model defined in the netlist should have the same model name and primitive type (`bsim2`, `bsim3`, `bjt`) in the altergroup. An instance defined in the netlist, should have the same instance name, terminal connections, and primitive type. For model groups, you can change the number of models in the group. However, you cannot change from a model to a model group and vice versa. See `spectre -h bsim3v3` for details on model groups.

### Syntax

```
Name altergroup parameter=value ...
```

### Parameters

<code>annotate</code>	Degree of annotation. Possible values are <code>no</code> and <code>title</code> .
<code>title</code>	Altergroup title.

### Example

```
FastCorner altergroup {
```

## Spectre Circuit Simulator Reference Analysis Statements

---

```
parameters p2=1 p3=p1+2
myopt options temp=27
model myres resistor r1=1e3 af=1
model mybsim bsim3v3 lmax=p1 lmin=3.5e-7
m1 (n1 n2 n3 n4) mybsim w=0.3u l=1.2u
}
```

The list of public devices supported by altergroup is as follows:

```
angelovangelov_ganasm_ganassert
bht bht0bjt bjt301
bjt500bjt500tbjt503bjt504
bjt504tbjt505bjt505tbjt3500
bjt3500tbjtd504bjtd504tbjtd505
bjtd505tbjtd3500bjtd3500tbsim1
bsim2bsim3bsim3v3bsim4
bsim6bsimbulkbsimcmgbsimimg
bsimsoibsimsoi_sbsource_resistorcapacitor
capqccapsimccccccvcs
dcblockdcfeeddelta_gatedio500
diodediode_cmcekvekv3
ekv3_nqsek3v3_r4ekv3_rfekv3_s
fracpolegaashbthemt
hisim2hisim2_vahisim_diodehisim hv
hisim_igbthisimhmghi_simhv_vahisimsoi
hisimsoi_bthisimsoi_fbhvm0sigbt0
inductorintcapiprobeisource
jfetjfet100jfet100qjfetidg
jfetidgtjjjuncapjuncap200
juncap_eldoldmoslutsoilutsoit
mos1mos2mos3mos40
mos40tmos902mos903mos903c
mos903emos903tmos1101emos1101et
mos1102emos1102etmos2002mos2002e
mos2002etmos2002tmos3100mos3100t
mos11010mos11010tmos11011mos11011t
mos11020mos11020tmos11021mos11021t
mosvarmslinemutual_inductormvsg_gan
nodcapovcheckovcheck6pattern
pcccsppccvscopy_resport
printpsitftpsp102psp102e
psp103psp103tppsp1020psp1021
pspnqs102epspnqs103pspnqs1020pspnqs1021
pvccspvcvsr2r3
rdiffresistorrlck_matrixspmos
ssim_mbctlinetom2tom3
tom3vltransformerucsd_hbtutsoi2
vbicvccsvcvsvsource
wprobewsources
```

The list of public devices not supported by altergroup is as follows:

```
a2d atftbendbend2
bit bsource_capacitorcktromconductor
corecornercrosscurve
d2a_delaydielectricibis_buffer
iswitchjittereventlocmos0
mos15mtlinenportpdbh2dProxy
relaysccccccvssprobepport
stackupstepsvccssvcvs
```

## Spectre Circuit Simulator Reference Analysis Statements

---

switchteevswitchwinding  
zcccszccvszvccszvcvs

## Check Parameter Values (check)

### Description

The `check` analysis checks the values of component parameters to ensure that they are reasonable. This analysis reduces the cost of data entry errors. Various filters specify which parameters are checked. You can perform checks on input, output, or operating-point parameters. Use this analysis in conjunction with the `+param` command-line argument, which specifies a file that contains component parameter soft limits.

### Syntax

```
Name check parameter=value ...
```

### Parameters

<code>what</code>	<code>all</code>	The parameters that should be checked. Possible values are none, inst, models, input, output, all and oppoint.
-------------------	------------------	--

## Checklimit Analysis (`checklimit`)

### Description

A `checklimit` analysis allows the enabling or disabling of individual or group of `asserts` specified in the netlist. Use this analysis in conjunction with the `assert` statements in the netlist to perform checks on parameters of device instances, models, subcircuits or expressions.

Multiple `checklimit` analyses can be defined in the netlist. The enabled checks will be applied to all subsequent analyses until the next `checklimit` analysis is encountered.

### Syntax

```
Name checklimit parameter=value ...
```



## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Parameters

enable	[...]	Array of checks to be enabled. The patterns can have wildcard symbols. Default is all.
disable	[...]	Array of checks to be disabled. The patterns can have wildcard symbols. Default is none.
severity		Severity of the checks. Possible values are none, notice, warning, error and fatal.
title		Analysis title.
checkallasserts	yes	If all checks should be enabled or disabled. checkallasserts=no disables all checks. Possible values are no and yes.
asserts	[...]	Specifies a group of asserts whose parameters (specified using the param parameter) and values (specified using the value parameter) need to be changed.
param		Specifies the parameter setting for an individual assert or a group of asserts specified using the 'asserts' parameter. Possible values are max, min, check_windows, maxvio_perinst, maxvio_all, duration, level, extreme and skip_dev_inside_subckt.
value		Specifies the value of the parameter (specified using the 'param' parameter) for the assert (specified using the 'asserts' parameter) that needs to be changed.
filter	none	Set filter=progressive to report less assertion violations by omitting less severe repetitions. Set filter=extreme to only report top peak/duration/area violations for each assert. If severity level of an assert is set to 'error' or 'fatal', then for that assert filtering is disabled. Possible values are none, progressive and extreme.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Boundary parameters

boundary_type	time	Boundary type. Possible values are time and sweep.
start		Start time or sweep boundary of the checks.
stop		Stop time or sweep boundary of the checks.
check_windows	[...]	Boundary time or sweep windows of the checks. Array should have an even number of values [b_begin1 b_end1 b_begin2 b_end2 ...].
reset_times	[...]	Reset times vector for asserts. This option will be ignored if any assert statement has duration parameter, or contains time dependent expressions like avg() or integ().

## Setting for Simulink-MATLAB co-simulation (cosim)

### Description

Setting for Simulink-MATLAB co-simulation.

### Syntax

Name `cosim parameter=value ...`

### Parameters

<code>server</code>		MATLAB/Simulink server name.
<code>port</code>	<code>38520</code>	Co-simulink listen port.
<code>timeout</code>	<code>60 s</code>	Socket timeout in seconds. Default is 60s.
<code>inputs</code>	<code>[...]</code>	Array of input names.
<code>outputs</code>	<code>[...]</code>	Array of output node names.
<code>design</code>		MATLAB/Simulink design name. If the design name is specified, MATLAB is launched automatically.
<code>silent</code>	<code>yes</code>	Launch MATLAB with parameter <code>-nodesktop</code> . Possible values are no and yes.
<code>ratio</code>	<code>0</code>	Hold time ratio between two sample points. Default is 0.

## Costomerrpreset Analysis (customerrpreset)

### Syntax

Name customerrpreset parameter=value ...

### Parameters

maxstepratio	Maximum time step ratio for custom 'errpreset'.
reltolratio	Reltol ratio for custom 'errpreset'.
errpreset	Selects a reasonable collection of parameter settings. Possible values are liberal, moderate and conservative.
relref	Reference used for the relative convergence criteria. The default is derived from <code>errpreset</code> . Possible values are pointlocal, alllocal, sigglobal and allglobal.
method	Integration method. The default is derived from <code>errpreset</code> . Possible values are euler, trap, traponly, gear2, gear2only, trapgear2 and trapeuler.
lteratio	Ratio used to compute LTE tolerances from Newton tolerance. The default is derived from <code>errpreset</code> .

## DC Analysis (dc)

### Description

DC analysis finds the DC operating-point or DC transfer curves of the circuit. To generate transfer curves, specify a parameter and a sweep range. The swept parameter can be circuit temperature, a device instance parameter, a device model parameter, a netlist parameter, or a subcircuit parameter for a particular subcircuit instance. You can:

sweep the circuit temperature by giving the parameter name as `param=temp` without a `dev`, `mod` or `sub` parameter.

sweep a top-level netlist parameter by giving the parameter name without a `dev`, `mod` or `sub` parameter.

sweep a subcircuit parameter for a particular subcircuit instance by specifying the subcircuit instance name with the `sub` parameter, and the subcircuit parameter name with the `param` parameter.

After the analysis is complete, the modified parameter returns to its original value.

### Syntax

```
Name dc parameter=value ...
```

### Parameters

Sweep interval parameters

<code>start</code>	<code>0</code>	Start sweep limit.
<code>stop</code>		Stop sweep limit.
<code>center</code>		Center of sweep.
<code>span</code>	<code>0</code>	Sweep limit span.
<code>step</code>		Step size, linear sweep.
<code>lin</code>	<code>50</code>	Number of steps, linear sweep.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

dec		Points per decade.
log	50	Number of steps, log sweep.
values	[...]	Array of sweep values.
valuesfile		Name of the file containing the sweep values.
hysteresis	no	Perform DC hysteresis sweep. When set to yes, a reverse sweep will automatically be added to the DC sweep. Possible values are no and yes.

#### Sweep variable parameters

dev		Device instance whose parameter value is to be swept.
mod		Model whose parameter value is to be swept.
param		Name of parameter to sweep.

#### State-file parameters

force	none	Determine whether to force values for DC. Uses the values from the device and node ICs. Possible values are none, node, dev and all.
readns		File that contains estimate of DC solution (nodeset).
readforce		File that contains force values.
write		File to which solution at first step in sweep is written.
writefinal		File to which solution at last step in sweep is written.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>useprevic</code>	<code>no</code>	If set to yes or ns, use the converged initial condition from previous analysis as ic or ns. Possible values are no, yes and ns.
------------------------	-----------------	--

#### Output parameters

<code>save</code>		Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, none and nooutput.
-------------------	--	---

<code>nestlvl</code>		Levels of subcircuits to output.
----------------------	--	----------------------------------

<code>print</code>	<code>no</code>	Print node voltages. Possible values are no and yes.
--------------------	-----------------	--

<code>oppoint</code>	<code>no</code>	Should operating point information be computed; if yes, where should it be printed (screen or file). Operating point information is not printed if sweep parameter <code>param</code> is set ( if user wants to enable this kind of output for dc sweep, use option <code>enable_dcswEEP_op_info=yes</code> ). Possible values are no, screen, logfile and rawfile.
----------------------	-----------------	---

<code>check</code>	<code>yes</code>	Check operating point parameters against soft limits. Possible values are no and yes.
--------------------	------------------	---

#### Convergence parameters

<code>homotopy</code>	<code>all</code>	Method used when no convergence occurs on initial attempt of DC analysis. You can specify methods and their orders by specifying a vector setting such as <code>homotopy=[source ptran gmin]</code> . Possible values are none, gmin, source, dptran, ptran and all.
-----------------------	------------------	--

<code>newton</code>	<code>normal</code>	Users can specify newton methods such as <code>newton=none</code> , and <code>newton=normal</code> . Possible values are none and normal.
---------------------	---------------------	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

restart	yes	Restart from scratch if any condition has changed. If not, use the previous solution as initial guess. Possible values are no and yes.
maxiters	150	Maximum number of iterations.
maxsteps	10000	Maximum number of steps used in homotopy method.

#### Annotation parameters

annotate	sweep	Degree of annotation. Possible values are no, title, sweep, status, estimated, steps, iters, detailed, rejects and alliters.
title		Analysis title.

#### Emir output parameters

emirformat	none	Format of the EM/IR database file. Possible values are none and vavo.
emirfile		Name of the EM/IR database file. The default is '%A_emir_vavo.db'. The file is printed to a raw directory.
swpuseprevic	yes	DC sweep uses the solution of previous point as starting point. Possible values are no, yes and ns.
component		component instance.

You can define sweep limits by specifying the end points or the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. In addition, you can specify a step size parameter (`step`, `lin`, `log`, or `dec`) and determine whether the sweep is linear or logarithmic. If you do not specify a step size parameter, the sweep is linear when the ratio of stop to start values is less than 10 and



## Spectre Circuit Simulator Reference

### Analysis Statements

---

logarithmic when this ratio is 10 or greater. If you specify the `oppoint` parameter, Spectre computes and prints the linearized model for each nonlinear component.

Nodesets help find the DC or the initial transient solution. You can specify nodesets in the circuit description file with `nodeset` statements or in a separate file by using the `readns` parameter. When nodesets are specified, Spectre computes an initial guess of the solution by performing DC analysis, while forcing the specified values on to nodes by using a voltage source in series with a resistor whose resistance is `rforce`. Spectre then removes these voltage sources and resistors and computes the required solution from this initial guess.

Nodesets have two important uses. First, if a circuit has two or more solutions, nodesets can bias the simulator towards computing the required solution. Second, this is a convergence aid. By estimating the solution of the largest possible number of nodes, you might be able to eliminate a convergence problem or significantly speed up convergence.

When you simulate the same circuit multiple times, it is recommended that you use both `write` and `readns` parameters and assign the same file name to both parameters. DC analysis then converges quickly even if the circuit has changed since the last simulation, and the nodeset file is automatically updated.

You can set the 'force' parameter and specify the values to force the DC analysis. The values used to force signals are specified by using the `force` file, the `ic` statement, or the `ic` parameter on the capacitors and inductors. The force parameter controls the interaction of various methods of setting the force values. The effects of individual settings are as follows:

`force=none`: All initial conditions are ignored.

`force=node`: The `ic` statements are used, and the `ic` parameter on the capacitors and inductors are ignored.

`force=dev`: The `ic` parameters on the capacitors and inductors are used, and the `ic` statements are ignored.

`force=all`: Both the `ic` statements and the `ic` parameters are used, with the `ic` parameters overriding the `ic` statements.

If you specify a `force` file with the `readforce` parameter, force values read from the file are used, and any `ic` statements are ignored.

After you specify the force conditions, Spectre performs DC analysis with the specified nodes forced to the given value by using a voltage source in series with a resistor whose resistance is `rforce` (see `options`).

## DC Device Matching Analysis (dcmatch)

### Description

DCMATCH analysis performs device matching analysis on the operating point for a specified output. It computes the deviation in the operating point of a circuit due to processing or environmental variation in modeled devices. If `method=standard` is specified, a set of `dcmatch` parameters in model cards are required for each supported device contributing to the deviation. The analysis applies device matching models to construct equivalent mismatch current sources to all the devices that are modeled. These current sources are assumed to have zero mean and certain variance. The variance of the current sources is computed according to device matching models. Next, the 3-sigma variance of DC voltages or currents (due to the mismatch current sources) is computed at the specified outputs. The simulation result displays the devices rank ordered by their contribution to the outputs. In addition, for MOSFET devices, it displays threshold voltage mismatch, current factor mismatch, gate voltage mismatch, and drain current mismatch. For bipolar devices, it displays base-emitter junction voltage mismatch. For resistors, it displays resistor mismatches.

The analysis replaces multiple simulation runs that determine changes in accuracy with any changes in size. It automatically identifies the set of critical matched components during circuit design. For example, when there are matched pairs in the circuit, the contribution of two matched transistors is equal in magnitude but opposite in sign. Typical usage is to simulate the output offset voltage of operational amplifiers, estimate the variation in bandgap voltages, and predict the accuracy of current steering DACs.

For `method=standard`, DCMATCH analysis can be applied to following devices: BSIM3V3, BSIM4, BSIMSOI, EKV, PSP102, PSP103, BJT, VBIC, BHT, RESISTOR, PHY\_RES, R3, and resistor-type `bsource`. If `method=statistics` is specified, a statistics block is required to list the parameters that are considered as randomly varying. By default, all statistics parameters found in statistics blocks are considered in computation, unless the option `variations` is explicitly specified. Device matching models are not used in this case and `dcmatch` model parameters are not required.

### Syntax

```
Name ... dcmatch parameter=value ...
```

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Parameters

nth		Relative mismatch contribution threshold value.
where	screen	Where DC-Mismatch analysis results should be printed. Possible values are screen, logfile, file and rawfile.
file		File name for results to be printed if where=file is used.

#### Probe parameters

oprobe		Compute mismatch at the output defined by this component.
--------	--	---

#### Port parameters

portv		Voltage across this probe port is output of the analysis.
porti		Current through this probe port is output of the 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, linear sweep.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>lin</code>	<code>50</code>	Number of steps, linear sweep.
<code>dec</code>		Points per decade.
<code>log</code>	<code>50</code>	Number of steps, log sweep.
<code>values</code>	<code>[...]</code>	Array of sweep values.
<code>valuesfile</code>		Name of the file containing the sweep values.

#### Sweep variable parameters

<code>dev</code>		Device instance whose parameter value is to be swept.
<code>mod</code>		Model whose parameter value is to be swept.
<code>param</code>		Name of parameter to sweep.

#### State-file parameters

<code>readns</code>		File that contains an estimate of DC solution (nodeset).
<code>useprevic</code>	<code>no</code>	If set to yes or ns, use the converged initial condition from previous analysis as ic or ns. Possible values are no, yes and ns.

#### Output parameters

<code>save</code>		Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, none and nooutput.
<code>nestlvl</code>		Levels of subcircuits to output.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

oppoint	no	Determines whether the operating point information should be computed. If yes, where should it be printed (screen or file). Operating point information is not printed in case of the following conditions: (1) operating point is computed in the previous analysis and is unchanged (2) sweep parameter param is set. Possible values are no, screen, logfile and rawfile.
---------	----	--

#### Convergence parameters

prevoppoint	no	Use operating point computed on the previous analysis. Possible values are no and yes.
restart	yes	Restart the DC solution from scratch if any condition has changed. If not, use the previous solution as initial guess. Possible values are no and yes.

#### Annotation parameters

annotate	sweep	Degree of annotation. Possible values are no, title, sweep, status and steps.
title		Analysis title.

#### Miscellaneous parameters

version	0	Use BSIM short-channel mismatch equation for BSIM3 and BSIM4 devices, if the value is set to 1 and 3. If set to 0 and 2, do not use BSIM short-channel mismatch equation. Values 2 and 3 are compatible with Monte Carlo analysis while 0 and 1 are not. This option works only when method=standard. Possible values are 0 to 3.
---------	---	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

method	standard	Proceed with standard device mismatch models, or utilize the parameters defined in statistics blocks to compute output variation. Possible values are standard and statistics.
nsigma	3	Specifies the variation of each statistics parameter (in the number of sigma). This option works only if method=statistics.
variations	all	Selects the type of statistical parameters that are involved in dcmatch. The option is valid only if method=statistics. Possible values are process, mismatch and all.
deltascale	1	Scaling factor of offset applied to mismatch and process parameters.
nullmfactorcorrelation	no	Controls the mismatch variation correlation of parallel devices defined by m-factor. If set to yes, devices are assumed to get uncorrelated mismatch variations. If set to no, devices are assumed to get the same mismatch variation. Possible values are no and yes.

The dcmatch analysis will find a DC operating point first. If the DC analysis fails, the dcmatch analysis also fails. The parameter mth is a threshold value relative to maximum contribution. Any device contribution less than (mth \* maximum) is not reported, where maximum is the maximum contribution among all the devices of a given type.

Examples:

```
dcmm1 dcmatch mth=1e-3 oprobe=vd porti=1
dcmm2 dcmatch mth=1e-3 oprobe=r3 portv=1
dcmm3 n1 n2 dcmatch mth=1e-3 where=rawfile stats=yes
dcmm4 n3 0 dcmatch mth=1e-3 where=file file="%C:r.info.what"
sweep1 sweep dev=mp6 param=w start=80e-6 stop=90e-6 step=2e-6 {
dcmm5 dcmatch oprobe=vd mth=1e-3 where=rawfile }
```

## Spectre Circuit Simulator Reference

### Analysis Statements

---

dcmm6 n3 0 dcmatch mth=0.01 dev=x1.mp2 param=w start=15e-6 stop=20e-6 step=1e-6

dcmm7 n3 0 dcmatch mth=0.01 param=temp start=25 stop=100 step=25

**Note:** Note: porti allows you to select a current associated with a specific device given in oprobe as an output. All types of devices are supported for oprobe. For inductor, vsource, switch, controlled voltage source, iprobe, etc., porti can only be set to one, because these devices are two-terminal devices (one port); and for tline porti can be set to one or two, because it is a four-terminal device (two ports).

## Device Characterization (devchrz)

### Description

This analysis computer the characterization in the transistor level

The following are brief descriptions of the types of parameters you can request with the `devchrz` statement:

```
my_characterization devchrz what=tmilde
```

### Syntax

```
Name devchrz parameter=value ...
```

### Parameters

<code>what</code>	<code>tmilde</code>	The parameters that should be printed. Possible values are none, <code>tmilde</code> and all.
-------------------	---------------------	---



## Immediate Set Dspf Options (dspf\_options)

### Description

The immediate set options statement sets or changes various program control options. These options take effect immediately and are set while the circuit is read. For more options, see the description of individual analyses.

**Note:** Note: Options that are dependent on netlist parameter values do not maintain their dependencies on those netlist parameters.

In many cases, a particular option can be controlled by either a command-line or netlist specification. In the situation where both are used, the command-line option takes priority over the setting in the netlist options statement.

### Syntax

```
Name dspf_options parameter=value ...
```

### Parameters

Global Nodes Parameters

Process Nets Parameters

Process Floating Caps Parameters

Temperature Parameters

Memcell Detection Parameters

Other Parameters

RCR Related Parameters

## Envelope Following Analysis (envlp)

### Description

This analysis computes the envelope response of a circuit based on the specified analysis `clockname`, `period` or `fund`. If `clockname` is specified, the simulator automatically determines the clock period by looking through all the sources with the specified name. The envelope response is computed over the interval from `start` to `stop`. If the interval is not a multiple of the clock period, it is rounded off to the nearest multiple before the stop time. The initial condition is taken to be the DC steady-state solution, if not given.

Envelope following analysis is most efficient for circuits where the modulation bandwidth is orders of magnitude lower than the clock frequency. This is typically the case, for example, in circuits where the clock is the only fast varying signal and other input signals have a spectrum whose frequency range is orders of magnitude lower than the clock frequency. The down conversion of two closely placed frequencies can also generate a slow-varying modulation envelope.

Envelope following analysis is capable of handling both autonomous (non-driven) and driven (non-autonomous) circuits. Autonomous circuits are time-invariant circuits that have time-varying responses. Therefore, autonomous circuits generate non-constant waveforms even though they are not driven by a time-varying stimulus. Driven circuits require time-varying stimulus to generate a time-varying response. The most common example of an autonomous circuit is an oscillator.

When applied to autonomous circuits, envelope following analysis requires you to specify a pair of nodes, `p` and `n`. In fact, this is how envelope following analysis determines whether it is being applied to an autonomous or a driven circuit. If the pair of nodes is supplied, envelope assumes the circuit is autonomous; if not, the circuit is assumed to be driven.

The analysis generates two types of output files, a voltage versus time (`td`) file and an amplitude/phase versus time (`fd`) file for each of the specified harmonics of the clock fundamental.

Fast mode envelope analysis is used to simulate RF power amplifier (PA) with I/Q orthogonal modulation. Like normal envelope analysis, the time scale difference between I/Q signals and carrier is very large. Fast envelope analysis has larger speed up performance than normal envelope. Fast envelope has two modes, `level1` and `level2`. `Level1` is used to simulate the circuit without memory effect. `Level2` has been discontinued. If selected, the simulator will automatically switch to `level1`. If the circuit has strong nonlinear-memory effect, fast envelope can be inaccurate. In this situation, the only accurate way is to use regular envelope analysis. Fast envelope only outputs the `fd` result of specified nodes (assigned by the parameter 'output') at harmonic 1 of carrier.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

Fast mode envelope analysis is not supported in the Shooting engine; its parameters apply only to the harmonic balance engine.

### Syntax

Name [p] [n] envlp parameter=value ...

### Parameters

#### Envelope fundamental parameters

clockname		Name of the clock fundamental.
modulationbw	(Hz)	Modulation bandwidth.
resolutionbw	(Hz)	Resolution bandwidth: if set, overwrites the stoptime to be at least 1/resolutionbw.

#### Simulation interval parameters

stop	(s)	Stop time.
start	0 s	Start time.
tstab	0 s	Initial stabilization time: can be used to change the phase that envelope starts shooting.
period	(s)	Period of the clock fundamental: if set, clockname can be ignored. It is the estimated period for autonomous circuits.
fund	(Hz)	Alternative to period. Frequency of the clock fundamental frequency.
outputstart	start s	Output is saved only after this time is reached.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Time-step parameters

maxstep	(s)	Maximum time step for inner transient integration. Default is derived from <code>errpreset</code> .
envmaxstep	(s)	Maximum outer envelope step size. Default is derived from <code>errpreset</code> .
fixstepsize	no	Use this option to fix envelope step size for speeding up envelope analysis. Possible values are no and yes.
stepsize	4	The number of cycles skipped between two steps when <code>fixstepsize</code> is yes. The time interval between the two steps will be $(stepsize+1)*T_c$ , where $T_c$ is the clock period. For shooting, autonomous, and fm envelope, it is rounded off to an integer.
stepperiod		The interval (in seconds of envelope following time) between two steps when <code>fixstepsize</code> is yes. Should be greater than period of clock. For autonomous FM or shooting envelope, it is rounded off to the nearest integer multiple of clock period.

#### Initial-condition parameters

ic	all	What should be used to set initial condition. Possible values are dc, node, dev and all.
skipdc	no	If set to yes, there will be no DC analysis for initial transient. Possible values are no and yes.
readic		File that contains initial transient condition.
useprevic	no	If set to yes or ns, use the converged initial condition from previous analysis as ic or ns. Possible values are no, yes and ns.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Convergence parameters

readns		File that contains estimate of initial DC solution.
cmin	0 F	Minimum capacitance from each node to ground.

#### State-file parameters

write		File to which initial transient solution is to be written.
writefinal		File to which final transient solution is to be written.
swapfile		Temporary file that holds the matrix information used by Newton's method. It tells Spectre to use a regular file, rather than virtual memory, to hold the matrix information. Use this option if Spectre does not have enough memory to complete this analysis. This parameter is now valid only for shooting. It does not apply to harmonic balance..

#### Envelope Integration method parameters

envmethod	gear2only	Envelope Integration method. Possible values are euler, trap, traponly, gear2, gear2only and trapgear2.
-----------	-----------	---

#### Integration method parameters

method	gear2only	Inner transient integration method. Possible values are euler, trap, traponly, gear2, gear2only and trapgear2.
oscic	default	Oscillator IC method. It determines how the starting values for the oscillator are determined. Possible values are default and lin.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Accuracy parameters

<code>errpreset</code>	<code>moderate</code>	Selects a reasonable collection of parameter settings. Possible values are liberal, moderate and conservative.
<code>relref</code>		Reference used for the relative convergence criteria. Default is derived from <code>errpreset</code> . Possible values are pointlocal, alllocal, sigglobal and allglobal.
<code>lteratio</code>		Ratio used to compute LTE tolerances from Newton tolerance. Default is derived from <code>errpreset</code> .
<code>smoothcheck</code>	<code>yes</code>	When set to no, the smoothness of the envelope is not checked. The skip cycles are as many as possible. If 'fixstepsize=yes', 'smoothcheck' is set to 'no' automatically.. Possible values are no and yes.
<code>itres</code>	<code>1e-2</code>	Relative tolerance for linear solver.
<code>inexactNewton</code>	<code>no</code>	Inexact Newton method. Possible values are no and yes.
<code>steadyratio</code>		Ratio used to compute steady-state tolerances from LTE tolerance. Default is derived from <code>errpreset</code> .
<code>envlteratio</code>		Ratio used to compute envelope LTE tolerances. Default is derived from <code>errpreset</code> .

#### Annotation parameters

<code>annotate</code>	<code>sweep</code>	Degree of annotation. Possible values are no, title, sweep, status and steps.
<code>title</code>		Analysis title.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Output parameters

harms		If <code>harmonicbalance</code> is set to no, it is the number of clock harmonics to output and the default value is 1. If <code>harmonicbalance</code> is set to yes, it is the <code>maxharm</code> of the clock fundamental and the default value is 3.
harmsvec	[...]	Array of desired output clock harmonics. Alternative form of <code>harms</code> that allows selection of specific harmonics. For multi-carrier envelope, each group of elements with size equal to that of <code>funds</code> is a selection of specific harmonic combinations of fundamental frequencies.
outputtype	both	Output type. Possible values are both, envelope and spectrum.
save		Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, none and nooutput.
nestlvl		Levels of subcircuits to output.
compression	no	Perform data compression on output. Possible values are no and yes.
strobeperiod	(s)	The output strobe interval (in seconds) of envelope following time. For Shooting Envelope, the actual strobe interval is rounded off to an integer multiple of the clock period.
transtrobeperiod	(s)	The output strobe interval (in seconds) of envelope time. The value of the parameter must be less than the cycle period. Those strobe timepoints in all cycles will output when the parameter is working. It is valid for shooting and HB.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Newton parameters

<code>maxiters</code>	5	Maximum number of Newton iterations per transient integration time step.
<code>envmaxiters</code>		Maximum number of Newton iterations per envelope step. For time domain Shooting envelope, the default is 3. For Harmonic Balance Envelope, the default is 40.
<code>restart</code>	no	Restart the DC solution from scratch if any condition has changed. If not, use the previous solution as initial guess. Possible values are no and yes.

#### Circuit age

<code>circuitage</code>	(Years)	Stress time. Age of the circuit used to simulate hot-electron degradation of MOSFET and BSIM circuits.
<code>fmspeedup</code>	0	The level to speed up the envelope analysis for frequency modulated signal. Default is 0 for standard envelope following and 1 for fmmod sources speed up.
<code>saveinit</code>	no	If set, the waveforms for the initial transient (tstab) before envelope are saved. Possible values are no and yes.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Fast envelope parameters

<code>fastmode</code>	<code>off</code>	This parameter controls the accuracy/performance tradeoffs of envelope analysis. When <code>fastmode=off</code> , a transistor-level accurate envelope algorithm is used. When <code>fastmode=level1</code> or <code>level2</code> , a block-level envelope algorithm is used that provides faster simulation and lesser memory occupation. Level1 mode provides the fastest operation, but ignores eventual modulation signal memory effects. The latter are accounted by Level2 mode at a little less operation speed.. Possible values are <code>off</code> , <code>level1</code> and <code>level2</code> .
<code>fastmethod</code>	<code>passband</code>	This parameter applies only in fast envelope mode. The passband method models only magnitude-dependent (AM-AM and AM-PM) effects . The baseband method includes both magnitude- and phase-dependent (PM-AM and PM-PM) effects. Use the passband method when you model PAs. Use the baseband method when the circuit includes transistor-level modulators, demodulators, and generally whenever the output depends both on the magnitude and phase of the input signal. Possible values are <code>passband</code> and <code>baseband</code> .
<code>writenv</code>		The file to which fast mode envelope data is written. It can be reused by 'readenv' if the circuit remains unchanged, except for the decrease in 'srci' or 'srcq' scale. Can only be used in fast envelope.
<code>readenv</code>		File from which fast mode envelope data is read. It can be used only if the circuit remains unchanged after the file is created, except for the decrease in 'srci' or 'srcq' scale. Can only be used in fast envelope.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>srci</code>	<code>[...]</code>	I branch baseband modulation source, whose signal corresponds to the real part of baseband signal. If two sources are assigned, it means that the inputs are differential signals and the first source is positive. Only the pwl type of source is supported. Can only be used in fast envelope.
<code>srcq</code>	<code>[...]</code>	Q branch baseband modulation source, whose signal corresponds to the imaginary part of baseband signal. If two sources are assigned, it means that the inputs are differential signals and the first source is positive. Only the pwl type of source is supported and can be used only in fast envelope.
<code>srcw</code>	<code>[...]</code>	Wireless source. Can be used only in fast envlp currently.
<code>sweepmethod</code>	<code>coarse</code>	This parameter controls the sweep algorithm during the fast envelope circuit characterization. When <code>sweepmethod</code> is <code>coarse</code> , <code>fine</code> , or <code>userdefined</code> , it uses a fixed number of sweep points. When <code>sweepmethod</code> is <code>adaptive</code> , it determines the number of sweep points automatically. <code>coarse</code> uses 7 magnitude points (plus 12 phase points when <code>fastmethod=baseband</code> ); <code>fine</code> uses 10 magnitude points (plus 16 phase points when <code>fastmethod=baseband</code> ); when <code>sweepmethod=userdefined</code> , use the <code>sweepnum</code> parameter to define the grid. Possible values are <code>coarse</code> , <code>fine</code> , <code>userdefined</code> and <code>adaptive</code> .

## Spectre Circuit Simulator Reference

### Analysis Statements

---

sweepnum	[ . . . ]	This parameter controls the number of sweep points for fast envelope characterization when sweepmethod=userdefined. When fastmethod=passband, sweepnum is a one-element integer array which sets the number of magnitude sweep points. When fastmethod=baseband, it is a two-element integer array which sets the number of magnitude and phase sweep points. In passband mode, default sweepnum=[7]. In baseband mode, default sweepnum=[7 12]. In most cases, default values are adequate to capture the main and adjacent channel power accurately. Nevertheless, it is recommended to increase sweepnum gradually to ensure that the results do not change. A reasonable strategy is to start at sweepnum=[7 12] and increase to [10 16] until results converge. The upper limit of the magnitude and phase sweep points is [49 48].
output	[ . . . ]	Fast mode envelope output nodes. Fast mode envelope only generates fd result ( complex solution of a certain harmonic versus time ) and can be used only in fast envelope.
outputharmonic	[1]	Output harmonic vector in fast mode envelope analysis. Default value is 1.
outputallharms	no	By default, fast mode envlp analysis outputs only one harmonic, as determined by the 'outputharmonic' parameter. When 'outputallharms' is set to 'yes', all harmonics are calculated and sent to output. In addition, when 'outputallharms' is set to 'yes' and the input excitation is single-tone, SpectreRF also calculates and stores the output time domain waveform. This parameter applies to fast mode envlp analysis only. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Fast envelope noise model parameters

<code>noisemethod</code>	<code>off</code>	<p>This parameter controls the noise model in fast envelope mode. <code>noisemethod</code> does not apply in regular envelope (when <code>fastmode=off</code>). By default, <code>noisemethod=off</code> and the circuit is treated as noiseless. If <code>noisemethod=level1</code>, the noise model is generated from linear noise analysis performed about the DC operating point. If <code>noisemethod=level2</code>, the noise model is calculated from a more rigorous periodic noise analysis. The root-mean-square value of the magnitude of the input signal and the mean of the phase of the input signal, are used as the operating point in periodic noise analysis. The level2 model is preferred in almost all the practical situations. The level1 model is faster to extract and may be useful in certain situations under nearly-linear operating condition. Possible values are <code>off</code>, <code>level1</code> and <code>level2</code>.</p>
<code>fstart</code>	<code>1k</code>	<p>This parameter sets the starting sweep frequency for the fast envelope noise sweep.</p>
<code>dec</code>	<code>3</code>	<p>This parameter controls the noise frequency sweep for the fast envelope noise model. It is applicable when <code>noisemethod=level1</code> or <code>level2</code>. When set, the simulator performs a linear or periodic noise analysis around output harmonic, using <code>dec</code> log steps per decade in the frequency interval given by <code>[fstart, 1/(2*Tstep)]</code>. If <code>lin</code> and <code>dec</code> are both specified, log sweep is used and the parameter <code>lin</code> is ignored.</p>
<code>lin</code>	<code>10</code>	<p>This parameter controls the noise frequency sweep for the fast envelope noise model. It is applicable when <code>noisemethod=level1</code> or <code>level2</code>. When set, the simulator performs a linear or periodic noise analysis around output harmonic, using <code>lin</code> equal steps in the frequency interval given by <code>[fstart, 1/(2*Tstep)]</code>.</p>

## Spectre Circuit Simulator Reference

### Analysis Statements

---

noiseseq	no	This parameter controls BER/EVM/FSKErr calculation in wprobe when noisemethod=level1 or level2 in fast envlp wireless simulation. When set, the simulator provides two sets of data for wprobe to calculate two BER/EVM/FSKErr values. One is impacted only by circuit distortion, the other is impacted by both circuit distortion and noise. Possible values are no and yes.
----------	----	--

#### Wireless fundamental parameters

wsource		Wireless source in envlp analysis. It must be selected within wsource instances.
wprobe	[...]	The wprobe vector in envlp analysis. It is designed to measure evm or ber, and so on.

#### Harmonic Balance Envelope parameters

funds	[...]	Array of fundamental frequency names for fundamentals that will be used for Harmonic Balance Envelope.
maxharms	[...]	Array of number of harmonics of each fundamental that will be considered for Harmonic Balance Envelope.
freqdivide		Large signal frequency division.
fundfreqs	[...]	Array of fundamental frequencies to use in multi-carrier envelope.
harmonicbalance	no	Use Harmonic Balance Envelope. Possible values are no and yes.
flexbalance	no	The same parameter as harmonicbalance. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>oversamplefactor</code>	<code>1</code>	Oversample sample device evaluations for Harmonic Balance Envelope.
-------------------------------	----------------	---

<code>oversample</code>	<code>[...]</code>	Array of oversample factors for each tone for Harmonic Balance Envelope.
-------------------------	--------------------	--

#### Tstab save/restart parameters

<code>saveperiod</code>		Save the tran analysis periodically on the simulation time.
-------------------------	--	---

<code>saveclock</code>	<code>1800 s</code>	Save the tran analysis periodically on the wall clock time.
------------------------	---------------------	---

<code>savetime</code>	<code>[...]</code>	Save the analysis states into files on the specified time points.
-----------------------	--------------------	---

<code>savefile</code>		Save the analysis states into the specified file.
-----------------------	--	---

<code>recover</code>		Specify the file to be restored.
----------------------	--	----------------------------------

#### AMS-envlp co-sim parameters

<code>resetenv</code>	<code>no</code>	Use this option to reset envelope data after D2A/A2D events for AMS-envlp co-simulation. Possible values are no and yes.
-----------------------	-----------------	--

<code>ignoredclk</code>	<code>no</code>	Use this option to ignore digital clock if the clock rate is in the same order as envelope clock for AMS-envlp co-simulation. Possible values are no and yes.
-------------------------	-----------------	---

<code>trancycles</code>	<code>5</code>	The number of transient cycles for AMS-envlp co-simulation. This is the number of cycles around D2A/A2D events' time point. Default value is 5.
-------------------------	----------------	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### envlp-PAC parameters

<code>pacnames</code>	<code>[...]</code>	Names of <code>pac</code> , <code>pnoise</code> , <code>psp</code> , or <code>pxf</code> analyses to be performed at each time point in the <code>pactimes</code> array. Not for AMS.
<code>pactimes</code>	<code>[...] s</code>	Times when analyses specified in <code>pacnames</code> array are performed. Not for AMS.

If `period` or `fund` is not specified, the simulator examines all the sources whose name matches the clock name specified in the analysis line by the `clockname` parameter to determine the clock frequency. If more than one frequency is found, the greatest common factor of these frequencies is used as the clock frequency.

The maximum envelope step size is affected by many parameters. It can be directly limited by `envmaxstep`. It is also limited by `modulationbw`. You provide an estimate of the modulation bandwidth. The simulator puts at least eight points within the modulation period. It is recommended that you use `strobeperiod` to get equally spaced envelope points, which will improve the noise floor in power spectrum density computation.

The `harms` and `harmsvec` parameters affect the simulation time in an insignificant way. The spectrum is calculated for all the specified harmonics for all sampled integration cycles as the envelope following analysis marches on. For each harmonic, a file is generated. If `harmonicbalance` is no, `harms` is typically set to 1 or 2 because the high-order harmonics are not accurate.

Most parameters of this analysis are inherited from either transient or PSS analysis and their meanings are consistent. However, a few of them need to be clarified. The effect of `errpreset` on certain envelope following analysis parameters is shown in the following table.

For 'conservative' autonomous envelope, default values for 'method' and 'envmethod' set to 'traponly' to avoid numerical damping of the oscillator.

In this table,  $T$  is the period of the clock.

---

#### Parameter defaults as a function of `errpreset`

---

<code>errpreset</code>	<code>maxstep</code>	<code>envmaxstep</code>	<code>reltol</code>	<code>relref</code>	<code>steadyratio</code>	<code>envlteratio</code>
------------------------	----------------------	-------------------------	---------------------	---------------------	--------------------------	--------------------------

## Spectre Circuit Simulator Reference

### Analysis Statements

---

-----

liberal	T/20	Interval/10	0.01	sigglobal	0.1	0.35
moderate	T/20	Interval/25	0.001	sigglobal	0.1	3.5
conservative	T/50	Interval/50	0.0001	allocal	1.0	35.0

The default value for `compression` is no. The output file stores data for every signal at every timepoint for which Spectre calculates a solution. Spectre saves the X-axis data only once, because every signal has the same x value. If `compression=yes`, Spectre writes data to the output file only if the signal value changes by at least two times the convergence criteria. 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 the 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.



## Harmonic Balance Steady State Analysis (hb)

### Description

This analysis uses harmonic balance (in the frequency domain) to compute the response of circuits with one fundamental frequency (periodic steady-state, PSS) or multiple fundamental frequencies (quasi-periodic steady-state, QPSS). The simulation time required for an HB analysis is independent of the time-constants of the circuit. This analysis also determines the circuit's periodic or quasi-periodic operating point, which can then be used during a periodic or quasi-periodic time-varying small-signal analysis, such as HBAC, HBSP, or HBNOISE.

Usually, harmonic balance (HB) analysis is a very efficient way to simulate weak nonlinear circuits. In addition, HB analysis works better than shooting analysis (in the time domain) for frequency-dependent components, such as delay, transmission line, and S-parameter data.

An HB analysis consists of two phases. The first phase calculates an initial solution, which the second phase then uses to compute the periodic or quasi-periodic steady-state solution, by using the Newton method.

The two most important parameters for HB analysis are 'funds' and 'maxharms'. The 'funds' parameter accepts a list of names of fundamentals that are present in the sources. These names are specified in the sources by the 'fundname' parameter. If only one name appears, the analysis is an HB PSS analysis. On the other hand, if more than one name appears, the analysis is an HB QPSS analysis. The 'maxharms' parameter accepts a list of numbers of the harmonics that are required to adequately model the responses due to the different fundamentals.

The annotate parameter has two values specific to HB analysis: detailed\_hb and internal\_hb. when annotate is set to detailed\_hb or internal\_hb, additional analysis debug information will be printed to log files. In the case of internal\_hb, encrypted debug information is stored in the internal log file. Both options are valid for pss and qpss analyses with flexbalance=yes.

### Syntax

```
Name [p] [n] hb parameter=value ...
```

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Parameters

##### HB fundamental parameters

<code>funds</code>	<code>[ . . . ]</code>	Array of fundamental frequency names for fundamentals to use in analysis.
<code>fundfreqs</code>	<code>[ . . . ]</code>	Array of fundamental frequencies to use in analysis.
<code>maxharms</code>	<code>[ . . . ]</code>	Array of number of harmonics of each fundamental to consider for each fundamental.
<code>autoharms</code>	<code>no</code>	Activates automatic harmonic number calculation in harmonic balance. Applies only if <code>tstab&gt;0</code> or if <code>autotstab=yes</code> . If a steady-state is reached, Spectre does a spectrum analysis to calculate the optimal number of harmonics for HB. The minimum number of harmonics is specified by <code>maxharms</code> . If steady-state is not reached to sufficient tolerance, <code>autoharms</code> may be disabled. Possible values are <code>no</code> and <code>yes</code> .
<code>selectharm</code>		Name of harmonics selection methods. Default is <code>diamond</code> when <code>maximorder</code> is set; otherwise, default is <code>box</code> . Possible values are <code>box</code> , <code>diamond</code> , <code>funnel</code> , <code>axis</code> , <code>widefunnel</code> , <code>crossbox</code> , <code>crossbox_hier</code> and <code>arbitrary</code> .
<code>evenodd</code>	<code>[ . . . ]</code>	Array of even, odd, or all strings for moderate tones to select harmonics.
<code>maximorder</code>		Maximum intermodulation order of harmonics in <code>diamond</code> , <code>funnel</code> , <code>widefunnel</code> , <code>crossbox</code> and <code>crossbox_hier</code> cuts. For example, given a two-tone case, a harmonic, <code>[hx hy]</code> , is in the <code>diamond</code> , <code>funnel</code> or <code>widefunnel</code> harmonic set when <code> hxl + hyl  &lt;= maximorder</code> . And it is in the <code>crossbox</code> or <code>crossbox_hier</code> set when <code> hxl  &lt;= maximorder</code> and <code> hyl  &lt;= maximorder</code> .

## Spectre Circuit Simulator Reference

### Analysis Statements

---

mappingfft	no	This parameter is valid only for funnel harmonic cut now. When it is set to yes, it is possible to save more memory but maybe at accuracy expense. Possible values are no and yes.
axisbw		The width of the band part of wide funnel harmonic cut along axis.
minialiasorder		The order of mini aliasing harmonic order.
selharmvec	[...]	Array of harmonics indices.
freqdivide		Large signal frequency division.
freqdividevector	[...]	Array of frequency division factors for each tone. This parameter overrides freqdivide.

#### Simulation interval parameters

tstab	0.0 s	Extra stabilization time after the onset of periodicity for independent sources.
autotstab	no	Activates the automatic initial transient (tstab) in harmonic balance and PSS shooting. If set to yes, the simulator decides whether to run tstab and for how long. Typically, the initial length of tstab is 50 periods; however, it could be longer depending on the type of circuit and its behavior. If steady-state is reached (or nearly reached), tstab terminates early. Possible values are no and yes.
envlp_autotstab	no	Activates the automatic initial envlp (tstabenvlp) in PSS shooting. If set to yes, the simulator decides how long to run tstabenvlp. Typically, the initial length of tstabenvlp is 50 periods; however, it could be longer depending on the type of circuit and its behavior. If steady-state is reached (or nearly reached), tstabenvlp terminates early. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

autosteady	no	Activates the automatic steady state detection during initial transient (tstab) in harmonic balance and PSS shooting. When steady state is reached (or nearly reached), tstab terminates early. This parameter applies only when tstab>0 or when autotstab=yes. autosteady=no 0 turns this feature off; autosteady=yes 1 activates this feature; autosteady=2 runs autosteady with lower steady state tolerance than autosteady=1. autosteady=2 may help pss convergence but with higher tstab costs. Possible values are 0, 1, 2, no and yes.
------------	----	--

tstabenvlp_autosteady	no	Activates the automatic steady state detection during tstabenvlp in PSS shooting. When steady state is reached, tstabenvlp terminates early. This parameter applies only when tstabenvlpstop>0 or when envlp_autotstab=yes. tstabenvlp_autosteady=no 0 turns this feature off; tstabenvlp_autosteady=yes 1 activates this feature; tstabenvlp_autosteady=2 runs autosteady with lower steady state tolerance than tstabenvlp_autosteady=1. tstabenvlp_autosteady=2 may help pss convergence but with higher tstabenvlp costs. Possible values are 0, 1, 2, no and yes.
-----------------------	----	--

tstabenvlpstop	0.0 s	Determines the stop time of envelope tstab.
----------------	-------	---

tstabenvlpstep	0.0 s	Determines the step size of envelope tstab.
----------------	-------	---

#### Time-step parameters

maxstep	(s)	Maximum time step. The default is derived from errpreset.
---------	-----	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

transres	1e-9*stop s	Transition resolution. The transient analysis attempts to stop at corners of input waveforms (for example, corners of rising/falling edge of a pulse). If such events occur within a time less than transres, the analysis combines the events into one and forces only one time point. The rest of the steps are determined by error control. This may lead to loss of detail.
----------	-------------	---

#### Initial-condition parameters

ic	all	The value to be used to set the initial condition. Possible values are dc, node, dev and all.
skipdc	no	If set to yes, there is no DC analysis for initial transient. Possible values are no, yes and sigrampup.
readic		File that contains initial condition.
oscic	default	Oscillator IC method. It determines how the starting values for the oscillator are calculated. <code>oscic=lin</code> provides you an accurate initial value, but it takes time; <code>oscic=lin_ic</code> is not recommended, which is the older version of <code>oscic=lin</code> for shooting analysis for backward compatibility; <code>oscic=fastic</code> is fast, but it is less accurate. <code>'oscic=skip'</code> directly uses the frequency provided by you as the initial guess frequency. It is only for the two-tier method. Possible values are default, lin, lin_ic, fastic and skip.
useprevic	no	If set to yes or ns, use the converged initial condition from previous analysis as ic or ns. Possible values are no, yes and ns.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

tone_homotopy	[...]	Array of homotopy options for convergence assistance with multi-divider cases, when hbbhomotopy=aggregation. The number of entries should be equal to that of tones. The possible values for each entry are 0, 1 and 2. '0' means no convergence assistance is applied for that tone. '1' means only transient stabilization (tstab) is run for that tone and '2' means both tstab and a single-tone HB simulation should be run to obtain the initial guess for multi-tone HB simulation.
---------------	-------	--

#### Convergence parameters

readns		File that contains an estimate of the initial transient solution.
cmin	0 F	Minimum capacitance from each node to ground.
hbbhomotopy	tone	Name of Harmonic Balance homotopy selection methods. Possible values are tstab, source, gsweep, tone, inctone, aggregation and steptone.
gstart	1.e-7	Start conductance for hbbhomotopy of gsweep.
gstop	1.e-12	Stop conductance for hbbhomotopy of gsweep.
glog	5	Number of steps, log sweep for hbbhomotopy of gsweep.
sweepic	none	IC extrapolation method in sweep HB analysis. Possible values are none, linear and log.
oscmetho		Osc Newton method for autonomous HB. Possible values are onetier and twotier.
pinnode		Node to pin during autonomous HB simulation.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>pinnode_rank</code>		Harmonic rank to pin during autonomous HB simulation.
<code>pinnode_mag</code>		This parameter gives an estimate of the magnitude of the pin node voltage. Default value is 0.01.
<code>pinnode_minus</code>		Second node to pin during autonomous HB simulation. Needed only when differential nodes exist in oscillator.
<code>backtracking</code>	<code>yes</code>	This parameter is used to activate the backtracking utility of Newton's method. The default is <code>yes</code> . Possible values are <code>no</code> , <code>yes</code> and <code>forced</code> .

#### Output parameters

<code>save</code>		Signals to output. Possible values are <code>all</code> , <code>lvl</code> , <code>allpub</code> , <code>lvlpub</code> , <code>selected</code> , <code>none</code> and <code>nooutput</code> .
<code>nestlvl</code>		Levels of subcircuits to output.
<code>saveinit</code>	<code>no</code>	If set to <code>yes</code> , the waveforms for the initial transient before steady state are saved. Possible values are <code>no</code> and <code>yes</code> .
<code>harmoutputlist</code>	<code>[...]</code>	Array of harmonics indices for hb output.

#### Integration method parameters

<code>tstabmethod</code>		Integration method used in stabilization time. The default is <code>traponly</code> for autonomous circuits, or is derived from <code>errpreset</code> for driven circuits. Possible values are <code>euler</code> , <code>trap</code> , <code>traponly</code> , <code>gear2</code> and <code>gear2only</code> .
--------------------------	--	--

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Accuracy parameters

errpreset		Selects a reasonable collection of parameter settings. Possible values are liberal, moderate and conservative.
maxperiods		Maximum number of iterations allowed before convergence is reached in shooting or harmonic balance Newton iteration. For PSS and QPSS, the default is 20 for driven circuits, and 50 for oscillators; For HB, the default is 100.
max_innerkrylov_ size	20	The maximum iteration number allowed in inner gmres solver in two-level gmres linear solver. The default value is 12 for large signal and 20 for small signal analysis.
max_outerkrylov_ size	4	The maximum iteration number allowed in inner gmres solver in two-level gmres linear solver. The default value is 4.
itres	1e-4 for shooting , 0.9 for HB	Controls the residual for iterative solution of linearized matrix equation at each Newton iteration. Tightening the parameter can help with the Newton convergence, but does not affect the result accuracy. The value should be between [0, 1]. Default value for shooting APS flow is 1e-3.
krylov_size	10	The minimum iteration count of the linear matrix solver used in HB large-signal analysis. After reaching krylov_size iterations, the iteration is forced to terminate because of the poor rate of convergence. Increase krylov_size if the simulation reports insufficient norm reduction errors in GMRES.
oversamplefactor	1	Oversample device evaluations.
oversample	[...]	Array of oversample factors for each tone. This parameter overrides oversamplefactor.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

excludeconvgwith BK	yes	Possible values are no and yes.
hbpartition_defs	[...]	Define HB partitions.
hbpartition_fund ratios	[...]	Specify HB partition fundamental frequency ratios.
hbpartition_harm s	[...]	Specify HB partition harmonics.
hbprecond_solver	basicsol ver	Choose a linear solver for the GMRES preconditioner. Possible values are basicsolver, blocksolver, autotest, blockdense, blocksolver2 and directsolver.
lowmem	0	Harmonic balance low memory mode; Possible values are 0, 1, or number ( $\geq 10$ ). The default value is '0', the low memory mode is turned off; if '1' is set, the standard low memory mode is turned on; If a number no less than 10 is set, Spectre interprets the value as the memory requested in GigaBytes.

#### Annotation parameters

annotate	sweep	Degree of annotation. Possible values are no, title, sweep, status, estimated, steps, iters, detailed, rejects, alliters, detailed_hb and internal_hb.
title		Analysis title.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Newton parameters

<code>restart</code>	<code>no</code>	Restart the DC/PSS/QPSS solution if set to 'yes'; if set to 'no', reuse the previous solution as an initial guess; if set to 'firstonly', restart if it is the first point of sweep (supported only in HB). The default value is 'no' for HB and 'yes' for shooting. Possible values are no, yes and firstonly.
----------------------	-----------------	---

#### Circuit age

<code>circuitage</code>	(Years)	Stress time. Age of the circuit used to simulate hot-electron degradation of MOSFET and BSIM circuits.
-------------------------	---------	--

#### State-file parameters

<code>write</code>		File to which initial transient solution (before steady-state) is written.
<code>writeshb</code>		File to which final harmonic balance steady-state solution is to be written. Small-signal analyses, such as hbac,hbsp, and hbnoise can read the steady-state solution from this file directly instead of running the hb analysis again.
<code>readhb</code>		File from which final harmonic steady-state solution is to be read. Small signal analyses such as hbac,hbsp, and hbnoise can read in the steady-state solution from this file directly instead of running the hb analysis again.
<code>selharmread</code>		File from which arbitrary harmonic data needs to be read.
<code>selharmwrite</code>		File to which harmonic data needs to be written.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Tstab save/restart parameters

saveperiod		Save the tran analysis periodically on the simulation time.
saveperiodhistory	no	Maintains the history of saved files. If yes, maintains all the saved files. Possible values are no and yes.
saveclock	(s)	Save the tran analysis periodically on the wall clock time. The default is 1800s for Spectre. This parameter is disabled in the APS mode by default.
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.

#### Compression parameters

xdbccompression	no	Sets the automatic gain compression analysis. In automatic gain compression analysis, Spectre automatically sweeps the input excitation until the gain, as defined by the analysis parameter xdbgain, compresses by the amount specified by the analysis parameter xdblevel. In gain compression analysis, Spectre outputs the hb solution at the calculated compression point only. Dependent analyses, such as hbnoise and hbac, are supported and calculated about the calculated compression level. Auxiliary output includes the gain and voltage/power compression curves. These outputs are available for analysis and post-processing in ADE. The possible values are yes and no. Default is no.
-----------------	----	--

## Spectre Circuit Simulator Reference

### Analysis Statements

---

xdblevel	[...]	Sets the gain compression level for compression analysis. The reference point for gain compression is the small-signal gain of the circuits, or as specified by the analysis parameter xdbref. Default is 1.
xdbgain	power	Chooses between the voltage gain and transducer power gain as the target for compression point calculation. When xdbgain=power, the gain is defined as $G \text{ (dB)} = P_{\text{load}} \text{ (dBm)} - P_{\text{available}} \text{ (dBm)}$ . When xdbgain=voltage, the gain is defined as $G \text{ (dB)} = \text{dB}20( V_{\text{load}} / V_{\text{source}} )$ . In both cases, Spectre sweeps the excitation source until xdbref - $G = \text{xdblevel}$ , where the analysis parameter xdbref defines the reference level for compression calculation. Possible values are power and voltage. Default is power.
xdbref	linear	Sets the reference point for gain compression calculations. When xdbref=linear, spectre uses the small-signal gain as the reference. When xdbref=max, spectre uses the maximum observed gain as the reference. Possible values are linear and max. Default is linear.
xdbsource		The instance name of the excitation source, which is swept automatically to reach the compression level. When xdbgain=power, the excitation source must be a port instance. When xdbgain=voltage, the excitation source must be a vsource instance.
xdbload		The instance name of the load termination. When xdbgain is power, xdbload can be a port, a resistor, or a current probe.
xdbnodep		The output terminals for voltage gain calculation when xdbgain=voltage. If either is left unspecified, the terminal is assumed to be the global ground.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

xdbnoden		The output terminals for voltage gain calculation when xdbgain=voltage. If either is left unspecified, the terminal is assumed to be the global ground.
xdbrefnode		The reference node when xdbload is a current probe. The default is the ground node.
xdbharm	[ . . . ]	The Integer array which specifies the harmonic indexes of the output voltage or power component.
xdbsteps	100	The maximum number of steps for the compression point search. The simulator terminates if xdbsteps exceeds before the compression point is found. The default is 100.
xdbmax		The maximum input power (or voltage) for the compression point search. Default is 30 dBm when xdbgain=power, and 2.0 V when xdbgain=voltage.
xdbstart		The starting input power (or voltage) for the compression point search. Default is (xdbmax-70) dBm when xdbgain=power, and xdbmax/20000 when xdbgain=voltage.
xdbtol	0.01	Sets the tolerance for compression analysis. This tolerance is used in compression curve fitting and calculating the compression point.
xdbrapid	no	Sets the automatic gain compression analysis in rapid mode. In this mode, Spectre does not trace the compression curve and calculates only the compression point.
xdbcpi		Sets the estimated input-referred compression point for rapid compression analysis.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>backoff</code>	<code>0.0</code>	The backoff point. If defined, an additional harmonic balance analysis will be performed after the compression analysis is done. Default is 0 dBm when <code>xdbgain=power</code> , and 0 V when <code>xdbgain=voltage</code> .
----------------------	------------------	---

#### Memory estimation parameters

<code>memoryestimate</code>	<code>no</code>	Sets the memory usage estimate for Harmonic Balance. If yes, a memory estimate is printed in the log file. You can use this memory estimate to plan the computing resources before submitting harmonic balance runs. In memory estimate mode, a short simulation is performed first, and the engine exits after printing the estimate in the log file without saving any results. If no, the simulation continues after the memory estimate is printed. Memory estimation is not recommended for simulations that require less than 500MB approximately. For PSS analysis, memory estimate mode does not apply unless <code>flexbalance=yes</code> . <code>memoryestimate=1</code> estimates the memory usage for large-signal analysis and <code>memoryestimate=2</code> estimates both large-signal analysis and small-signal simulations. Possible values are no and yes.
-----------------------------	-----------------	--

#### Oscillator tuning mode parameters

<code>tuneparam</code>		When set, <code>tuneparam</code> enables the tuning mode oscillator analysis. In the tuning mode analysis, a circuit parameter is automatically varied to reach the oscillation frequency specified by the <code>fundfreqs</code> parameter. The tuning parameter can be a device instance parameter (as determined by the parameters <code>tunedev</code> ) or a netlist parameter. This mode applies only to autonomous circuits (oscillators).
------------------------	--	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

tunedev		Sets the instance name of a device whose parameter (identified by tuneparam) will be varied such that the circuit oscillates at the specified frequency. Applies only in tuning mode autonomous analysis. Tunedev must be used with tuneparam.
tunerange	[...]	The tuning range of the parameter identified by tuneparam. Although tunerange is not required, it can aid in convergence if set. Tunerange also can be used with 'tunestep' or 'tunelin' to decide the acceptable discrete parameter set.
tunestep		Specify the step size between two adjacent discrete tuning points. Must be used with 'tunerange'.
tunelin		Specify the numbers of discrete tuning points. Must be used with 'tunerange'.
tunevalues	[...]	Specify the values of discrete tuning points.

#### LSSP parameters

lsspports	[...]	Specifies the list of ports on which the large-signal 2-port S-parameters are calculated.
lssp harms	[...]	Specifies the output harmonic for large-signal S-parameter calculations. The input harmonic is defined by the frequency parameters on the input port instance.
lsspfile		Identifies the file name for large-signal S-parameter output.
lsspdatafmt	touchstone	Sets the file format of the large-signal S-parameter output. Possible values are spectre and touchstone. Default is touchstone.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

lsspdatatype	msgphase	Sets the data format or the large-signal S-parameter output. Possible values are realimag, magphase and phase. Default is magphase.
--------------	----------	---

#### Dynamic parameters

param		Name of the parameter to be updated during pss/hb analysis (tstab stage). When param=paramName and param_vec=[t1 value1 t2 value2 ... valuen tn], paramName is set to value1 at t1, value2 at t2, and so on. Pss/hb analysis uses the last value in the param_vec(valuen) to find the steady state.
-------	--	---

paramset		Name of the dynamic parameter set.
----------	--	------------------------------------

param_vec	[...] s	The time_value points to param=name.
-----------	---------	--------------------------------------

param_file		The file that contains the time_value points to param=name.
------------	--	---

sub		Name of the subcircuit instance parameter specified in param=name.
-----	--	--

mod		Name of the device model parameter specified in param=name.
-----	--	---

dev		Name of the device instance parameter specified in param=name.
-----	--	--

param_step		Defines the frequency of updating the dynamic parameter values. If param_step=0, it updates the parameter value at a given time point.
------------	--	--

#### Osc macro source parameters

oscmacrogene	no	If set to yes, harmonic balance steady-state solution is saved. Possible values are no and yes.
--------------	----	---



## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>oscmacroprobe</code>		Specify probe of which the current data is to be saved.
<code>oscmacroout</code>	<code>[...]</code>	Specify nodes of which the voltage data is to be saved.
<code>oscmacrosave</code>		File to which harmonic balance steady-state solution is to be written.

The initial transient analysis provides a flexible mechanism to direct the circuit to a particular steady-state solution of interest and to avoid undesired solutions. The initial transient simulation also helps convergence by eliminating the large but fast decaying modes that are present in many circuits.

In some circuits, the linearity of the relationship between the initial and final states depends on when HB analysis begins. In practice, starting at a good point can improve convergence, and starting at a bad point can degrade convergence and slow down the analysis.

When HB analysis simulates oscillators, initialization is performed to obtain an initial guess of the steady-state solution and of the oscillating frequency. Two initialization methods are implemented, based on transient and linear analysis. When `oscic=default` is specified, transient initialization is used and the length of the transient is specified by `tstab`. You must start the oscillator by using initial conditions or by using a brief impulsive stimulus, just as you would if you were simulating the turn-on transient of the oscillator by using transient analysis. Initial conditions would be provided for the components of the oscillator's resonator. If an impulsive stimulus is used, it should be applied so as to couple strongly into the oscillatory mode of the circuit and poorly into any other long-lasting modes, such as those associated with bias circuitry. The Designers Guide to Spice and Spectre [K. S. Kundert, Kluwer Academic Publishers, 1995] describes in depth some techniques for starting oscillators. When `oscic=lin` is specified, linear initialization is used. In this method both oscillation frequency and amplitude are estimated based on linear analysis at DC solution. No impulsive stimulus or initial conditions are needed. Linear initialization is suitable for linear type of oscillators, such as LC and crystal oscillators. Note that `tstab` transient is still performed after linear initialization, though it can be significantly shortened or skipped. Either way, specifying a non-zero `tstab` parameter can improve convergence.

For the 'funds' parameter, the frequencies associated with fundamentals are figured out automatically by the simulator. An important feature is that each input signal can be a composition of more than one source. However, these sources must have the same fundamental name. For each fundamental name, the fundamental frequency is the greatest common factor of all frequencies associated with the name. Omitting a fundamental name in

## Spectre Circuit Simulator Reference

### Analysis Statements

---

the `funds` parameter is an error that stops the simulation. If `maxharms` is not given, a warning message is issued, and the number of harmonics defaults to 1 for each of the fundamentals in multi-tone simulation and 10 in single-tone simulation.

HB signal partition is a method of decomposing a circuit so that multi-rate behavior can be exploited to increase simulation performance. If every part of a circuit has the same spectrum structure, such as fundamental frequency and bandwidth, there is no need to apply signal partition. However, if the RF circuit has multiple tones, the signals in different parts can have various spectrum structures, such as different fundamental frequency and number of harmonics. With HB signal partition, you can divide the circuit into several parts based on the signals contained in them. The parameter 'hbpartition\_defs' defines the partitions. Each partition can be made up of one or more instances. For example,

```
hbpartition_defs = ["I9 I10" "I11 I12" "I13 I14"]
```

defines three partitions. The first partition consists of instance "I9" and "I10" while the second partition consists of instances "I11" and "I12". The third one has "I13" and "I14". The number of instances for each partition should not be less than 1 and there is no upper limit for the number. The principle for dividing a circuit is that the subcircuits or instances with the same spectrum properties should be put into one partition. The parameter 'hbpartition\_harms' specifies the maximum number of positive harmonics of each tone for every partition. For example,

```
hbpartition_harms=["10 0 0" "5 3 3" "3 3 3"]
```

So the maximum number of positive harmonics for the first partition is 10, 0 and 0, respectively. For the second partition, it is 5, 3 and 3. For the last one, it is 3, 3 and 3. The parameter 'hbpartition\_fundratios' indicates the fundamental frequency ratio of each tone for each partition. With these ratios, it is easy to know the fundamental frequencies of each tone of the partitions. For example,

```
hbpartition_fundratios=["2 1 1" "1 1 1" "1 1 1"]
```

If three global fundamental frequencies are defined as: LO=1GHz, RF1=1.1GHz and RF2=1.13GHz, it indicates the fundamental frequencies of each tone of the first partition is 2\*LO, 1\*RF1 and 1\*RF2, respectively. The second and third partition has the same frequencies for their each tone: 1\*LO, 1\*RF1 and 1\*RF2. However, you have to make sure that the global fundamental frequencies for each tone are the smallest among all the partitions and the ratios are integers.

The parameter `maxperiods` default value is set to 50 for HB.

The `errpreset` parameter lets you adjust the simulator parameters to fit your needs quickly. In most cases, it should also be the only parameter you need to adjust. If you want a fast simulation with reasonable accuracy, you can set `errpreset` to liberal (it is not recommended

## Spectre Circuit Simulator Reference

### Analysis Statements

to use liberal on an RF circuit). If you have some concern for accuracy, you can set errpreset to moderate. If accuracy is your main interest, you can set errpreset to conservative.

The following table shows the effect of errpreset on other parameters in HB with One-tone Driven Circuits, Multi-tone Driven Circuits and Autonomous Circuits:

Parameter defaults as a function of errpreset with different circuits

| One-tone Driven Circuits | Multi-tone Driven Circuits and Autonomous Circuits

errpreset		reltol(max)	Iteratio		reltol(max)	Iteratio
-----------	--	-------------	----------	--	-------------	----------

liberal		1e-3	3.5		1e-3	3.5
moderate		1e-3	3.5		1e-4	3.5
conservative		1e-4	*		1e-5	*

\* : Iteratio=10.0 for conservative errpreset by default. However, when the option reltol  $\leq 1e-4 \cdot 10.0 / 3.5$ , Iteratio is set to 3.5

for One-tone Driven Circuits, and when the option reltol  $\leq 1e-5 \cdot 10.0 / 3.5$ , Iteratio is set to 3.5 for Multi-tone Driven Circuits

and Autonomous Circuits.

The values of reltol are usually different in the tstab interval and in the hb interval. During tstab, reltol is set to the option reltol, whose default value is 1e-3. This part is not impacted by errpreset. If you want to change the value, set reltol in Spectre options. Any value more than 1e-3 is ignored.

The value of reltol in the hb interval is affected by both errpreset and the option reltol. errpreset sets the maximum value of reltol (as shown in the table above). If the option reltol is less than the maximum value, it is set to the option reltol. Otherwise, the maximum value is used. reltol value that is more than 1e-3 is ignored.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

If `errpreset` is not specified in the netlist, 'moderate' settings is used.

If the circuit you are simulating has infinitely fast transitions (for example, a circuit that contains nodes with no capacitance), Spectre might have convergence problems. To avoid this, you must prevent the circuit from responding instantaneously. You can accomplish this by setting `cmin`, the minimum capacitance to ground at each node, to a physically reasonable nonzero value. This often significantly improves Spectre convergence.

You can specify the initial condition for the transient analysis by using the `ic` statement or the `ic` parameter on the capacitors and inductors. If you do not specify the initial condition, the DC solution is used as the initial condition. The `ic` parameter on the transient analysis controls the interaction of various methods of setting the initial conditions. The effects of individual settings are as follows:

`ic=dc`: All initial conditions are ignored, and the DC solution is used.

`ic=node`: The `ic` statements are used, and the `ic` parameter on the capacitors and inductors is ignored.

`ic=dev`: The `ic` parameters on the capacitors and inductors are used, and the `ic` statements are ignored.

`ic=all`: Both `ic` statements and `ic` parameters are used, and the `ic` parameters override the `ic` statements.

If you specify an initial condition file with the `readic` parameter, initial conditions from the file are used, and any `ic` statements are ignored.

After you specify the initial conditions, Spectre computes the actual initial state of the circuit by performing a DC analysis. During this analysis, Spectre forces the initial conditions on nodes by using a voltage source in series with a resistor whose resistance is `rforce` (see `options`).

With the `ic` statement, it is possible to specify an inconsistent initial condition (one that cannot be sustained by the reactive elements). Examples of inconsistent initial conditions include setting the voltage on a node with no path of capacitors to ground, or setting the current through a branch that is not an inductor. If you initialize Spectre inconsistently, its solution jumps, that is, it changes instantly at the beginning of the simulation interval. You should avoid such changes because Spectre can have convergence problems while trying to make the jump.

You can skip DC analysis entirely by using the parameter `skipdc`. If DC analysis is skipped, the initial solution is trivial or is given in the file that you specified by using the `readic` parameter, or if the `readic` parameter is not specified, by the values specified on the `ic` statements. Device-based initial conditions are not used for `skipdc`. Nodes that you do not

## Spectre Circuit Simulator Reference

### Analysis Statements

---

specify with the `ic` file or `ic` statements start at zero. You should not use this parameter unless you are generating a nodeset file for circuits that have trouble in the DC solution; it usually takes longer to follow the initial transient spikes that occur when the DC analysis is skipped than it takes to find the real DC solution. The `skipdc` parameter might also cause convergence problems in the transient analysis.

The possible settings of parameter `skipdc` and their descriptions are as follows:

`skipdc=no`: Initial solution is calculated using normal DC analysis (default).

`skipdc=yes`: Initial solution is given in the file specified by the `readic` parameter or the values specified on the `ic` statements.

`skipdc=sigrampup`: Independent source values start at 0 and ramp up to their initial values in the first phase of the simulation. The waveform production in the time-varying independent source is enabled after the rampup phase. The rampup simulation is from `tstart` to `time=0` s, and the main simulation is from `time=0` s to `tstab`. If the `tstart` parameter is not specified, the default `tstart` time is set to  $-0.1 \cdot tstab$ .

Nodesets help the simulator find the DC or initial transient solution. You can specify nodesets in the circuit description file with `nodeset` statements, or in a separate file using the `readns` parameter. When nodesets are specified, Spectre computes an initial guess of the solution by performing DC analysis, while forcing the specified values on to nodes by using a voltage source in series with a resistor whose resistance is `rforce`. Spectre then removes these voltage sources and resistors and computes the true solution from this initial guess.

Nodesets have two important uses. First, if a circuit has two or more solutions, nodesets can bias the simulator towards computing the desired solution. Second, they are a convergence aid. By estimating the solution of the largest possible number of nodes, you might be able to eliminate a convergence problem or significantly speed up convergence.

Nodesets and initial conditions have similar implementation, but produce different effects. Initial conditions define the solution, whereas nodesets only influence it. When you simulate a circuit with a transient analysis, Spectre forms and solves a set of differential equations. Because differential equations have an infinite number of solutions, a complete set of initial conditions must be specified to identify the required solution. Any initial conditions that you do not specify are computed by the simulator to be consistent. The transient waveforms then start from initial conditions. Nodesets are usually used as a convergence aid and do not affect the final results. However, in a circuit with more than one solution, such as a latch, nodesets bias the simulator towards finding the solution closest to the nodeset values.

With parameter 'hbmhomotopy', you can specify harmonic balance homotopy selection methods. The possible values of parameter 'hbmhomotopy' and their descriptions are as follows:

## Spectre Circuit Simulator Reference

### Analysis Statements

---

'hbhomotopy=tstab': Simulator runs a transient analysis and generates an initial guess for harmonic balance analysis; it is recommended for nonlinear circuits or circuits with frequency dividers.

'hbhomotopy=source': For driven circuit, the simulator ignores tstab and accordingly increases the source power level; for oscillators, the simulator accordingly adjusts the probe magnitude until the probe has no effect on the oscillators. It is recommended for strongly nonlinear or high Q circuits.

'hbhomotopy=tone': This method is valid only for multi-tone circuit. The simulator first solves a single-tone circuit by turning off all the tones, except the first one, and then solves the multi-tone circuit by restoring all the tones and using the single-tone solution as its initial guess. It is recommended for multi-tone simulation with a strong first tone.

'hbhomotopy=inctone': Simulator first solves a single tone, then turns on moderate tones incrementally till all tones are enabled. It is recommended for circuits with one strong large tone.

'hbhomotopy=gsweep': A resistor, whose conductance is  $g$ , is connected with each node, and the sweep of  $g$  is controlled by  $gstart$ ,  $gstop$ , and  $glog$ . It is recommended for circuits containing high-impedance or quasi-floating nodes.

## HB AC Analysis (hbac)

### Description

The harmonic balance AC (HBAC) analysis computes transfer functions for circuits that exhibit single or multi-tone frequency translation. Such circuits include mixers, switched-capacitor filters, samplers, phase-locked loops, and so on. HBAC is a small-signal analysis like AC analysis, except that the circuit is first linearized about a periodically or quasi-periodically varying operating point, rather than about a simple DC operating point. Linearizing about a periodically or quasi-periodically time-varying operating point allows transfer-functions that include frequency translation, which is not the case when linearizing about a DC operating point because linear time-invariant circuits do not exhibit frequency translation. In addition, the frequency of the sinusoidal stimulus is not constrained by the period of the large periodic solution.

Computing the small-signal response of a periodically or quasi-periodically varying circuit is a two-step process. First, the small stimulus is ignored and the periodic or quasi-periodic steady-state response of the circuit to possibly large periodic stimulus is computed using HB analysis. As part of the HB analysis, the periodically or quasi-periodically time-varying representation of the circuit is computed and saved for later use. The second step is to apply the small stimulus to the periodically or quasi-periodically varying linear representation to compute the small signal response. This is done using the HBAC analysis. An HBAC analysis cannot be used independently; it must follow an HB analysis. However, any number of periodic or quasi-periodic small-signal analyses, such as HBAC, HBSP, or HBNOISE, can follow an HB analysis.

Modulated small signal measurements are possible using the Analog Design Environment(ADE). The `modulated` option for HBAC and other modulated parameters are set by ADE. HBAC analyses with this option produce results that can have limited use outside ADE. Direct Plot is configured to analyze these results and combine several wave forms to measure AM and PM response due to single sideband or modulated stimuli. For details, see the 'Spectre Circuit Simulator and Accelerated Parallel Simulator RF Analysis User Guide'.

**Note:** Note: Unlike other analyses in Spectre, the HBAC analysis can only sweep frequency.

### Syntax

Name ... `hbac parameter=value` ...

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### 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.
valuesfile		Name of the file containing the sweep values.
sweeptype	unspecified	Specifies if the sweep frequency range is the absolute frequency of input or if it is relative to the port harmonics. When the unspecified value is used, Spectre RF sweeps using relative when autonomous simulation is performed or when the analysis is PSP; for other cases Spectre RF sweeps the absolute value of the input. Possible values are absolute, relative and unspecified.
relharmvec	[...]	Sideband - vector of HB harmonics - to which relative frequency sweep should be referenced.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Sampled analysis parameters

<code>ptvtype</code>	<code>timeaveraged</code>	Specifies whether the PTV analysis will be traditional or sampled under certain conditions. Possible values are <code>timeaveraged</code> and <code>sampled</code> .
<code>sampleprobe</code>		The crossing event at this port triggers the sampled small signal computation.
<code>thresholdvalue</code>	<code>0</code>	Sampled measurement is done when the signal crosses this value.
<code>crossingdirection</code>	<code>all</code>	Specifies the transitions for which sampling must be done. Possible values are <code>all</code> , <code>rise</code> , <code>fall</code> and <code>ignore</code> .
<code>maxsamples</code>	<code>16</code>	Maximum number of sampled events to be processed during the sampled analysis.
<code>extrasampletimepoints</code>	<code>[...]</code>	Additional time points for sampled PTV analysis.

#### Output parameters

<code>sidevec</code>	<code>[...]</code>	Array of relevant sidebands for the analysis.
<code>maxsideband</code>	<code>7</code>	An alternative to the <code>sidebands</code> array specification, which automatically generates the array: <code>[-maxsideband ... 0 ... +maxsideband]</code> . For the shooting analysis, the default value is 7. For HB small signal analysis, the default value is the <code>harms/maxharms</code> setting in the HB large signal analysis. It is ignored in HB small signal when it is larger than the <code>harms/maxharms</code> value of large signal.
<code>freqaxis</code>		Specifies whether the results should be printed as per the input frequency, the output frequency, or the absolute value of the output frequency. Default is <code>absout</code> . Possible values are <code>absout</code> , <code>out</code> and <code>in</code> .

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>save</code>	Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, none and nooutput.
-------------------	---

<code>nestlvl</code>	Levels of subcircuits to output.
----------------------	----------------------------------

#### Convergence parameters

<code>tolerance</code>	Tolerance for linear solver. The default value is 1.0e-9 for shooting-based solver and 1.0e-4 for harmonic balance-based solver.
------------------------	--

<code>relativeTol</code>	Relative tolerance for harmonic balance-based linear solver. Default value is 1.0e-2.
--------------------------	---

<code>resgmrescycle</code>	<code>short</code>	Restarts GMRES cycle. Possible values are instant, short, long, recycleinstant, recycleshort, recyclelong and custom.
----------------------------	--------------------	---

<code>hbprecond_solver</code>	<code>autoset</code>	Select a linear solver for the GMRES preconditioner. Default is autoset. With autoset, the simulator will automatically select the appropriate preconditioner. The preconditioner affects the rate of convergence of the linear matrix solver used in periodic small-signal analysis. When autoset is selected, the simulator may decide to switch to a different preconditioner after the analysis begins. When that happens, the simulator may issue a warning instructing you to choose a different preconditioner during subsequent runs. Although not required, choosing a different preconditioner according to the simulator's instructions may speed up subsequent analyses. Possible values are basicsolver, blocksolver, autoset, blockdense, blocksolver2 and directsolver.
-------------------------------	----------------------	--

## Spectre Circuit Simulator Reference

### Analysis Statements

---

lowmem	0	Harmonic balance low memory mode; Possible values are 0, 1, or number ( $\geq 10$ ). The default value is '0', the low memory mode is turned off; if '1' is set, the standard low memory mode is turned on; If a number no less than 10 is set, Spectre interprets the value as the memory requested in GigaBytes..
krylov_size	200	This parameter is used to set maximum iteration count of the linear matrix solver used in periodic small-signal analysis. After reaching $1.25 \times \text{krylov\_size}$ iterations, the iteration is forced to terminate because of the poor rate of convergence. Increase the krylov_size if the simulation reports insufficient norm reduction errors in GMRES.

#### Annotation parameters

annotate	sweep	Degree of annotation. Possible values are no, title, sweep, status, steps and detailed_hb.
title		Analysis title.

#### Modulation conversion parameters

modulated	no	Compute transfer functions/conversion between modulated sources and outputs. Possible values are single, first, second and no.
inmodharmnum	1	Harmonic value for the input source modulation.
outmodharmvec	[...]	Harmonic list for the output modulations.
moduppersideband	1	Index of the upper sideband included in the modulation of an output for PAC and HBAC, or an input for PXF.
modsource		Refer the output noise to this component.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

perturbation	linear	The type of HBAC analysis. The default is linear for normal HBAC analysis. im2ds stands for im2 distortion summary and ds stands for distortion summary. Value multiple_beat is deprecated and will be removed in the future: use triple_beat instead. Possible values are linear, ds, ip3, ip2, im2ds, multiple_beat and triple_beat.
flin_out	0 Hz	Frequency of linear output signal.
fim_out	0 Hz	Frequency of IM output signal.
out1	NULL	Output signal 1.
out2	NULL	Output signal 2.
contriblist	NULL	Array of device names for distortion summary. When contriblist=[""], distortion from each non-linear device is calculated.
maxharm_nonlin	4	Maximum harmonics of input signal frequency induced by non-linear effect.
rfmag	0	RF source magnitude.
rfdbm	0	RF source dBm.
rf1_src	NULL	Array of RF1 source names for IP3/IP2/IM2.
rf2_src	NULL	Array of RF2 source names for IP3/IP2/IM2.
rf_src	NULL	Array of RF source names for triple beat analysis, triple beat is not supported in shooting engine, PSS must be run before PAC with 'flexbalance=yes' or 'harmonicbalance=yes'.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>fregs</code>	NULL	Array of RF source frequencies for triple beat analysis, triple beat is not supported in shooting engine, PSS must be run before PAC with 'flexbalance=yes' or 'harmonicbalance=yes'.
<code>rfampls</code>	NULL	RF source amplitudes; the units are dBm for ports, Voltage for v-sources and Ampere for i-sources.

You can select the set of periodic small-signal output frequencies of interest by setting either the `maxsideband` or the `sidevec` parameter. When there is only one tone in HB analysis, sidebands are *n* integer numbers,  $K_1, K_2, \dots, K_n$ , and the output frequency at each sideband is computed as follows:

$$f(\text{out}) = f(\text{in}) + K_i * \text{fund}(\text{hb})$$

where  $f(\text{in})$  represents the (possibly swept) input frequency and  $\text{fund}(\text{hb})$  represents the fundamental frequency used in the corresponding HB analysis. Thus, when analyzing a down-converting mixer, while sweeping the RF input frequency, the most relevant sideband for IF output is  $K_i = -1$ . When simulating an up-converting mixer, while sweeping IF input frequency, the most relevant sideband for RF output is  $K_i = 1$ . By setting the `maxsideband` value to  $K_{\text{max}}$ , all  $2 * K_{\text{max}} + 1$  sidebands from  $-K_{\text{max}}$  to  $+K_{\text{max}}$  are generated.

When there are multiple tones in HB analysis, sidebands are vectors. Consider that you have one large tone and one moderate tone in HB. A sideband,  $K_1$ , is represented as  $[K_{1\_1} \ K_{1\_2}]$ . Corresponding frequency is as follows:

$$K_{1\_1} * \text{fund}(\text{large tone of HB}) + K_{1\_2} * \text{fund}(\text{moderate tone of HB})$$

The assumption is that there are *L* large and moderate tones in HB analysis and a given set of *n* integer vectors representing the sidebands,  $K_1 = \{ K_{1\_1}, \dots, K_{1\_j}, \dots, K_{1\_L} \}$ ,  $K_2, \dots, K_n$ . The output frequency at each sideband is computed as follows:

$$f(\text{out}) = f(\text{in}) + \text{SUM}_{j=1\_to\_L} \{ K_{i\_j} * \text{fund}_{j\_}(\text{hb}) \},$$

where  $f(\text{in})$  represents the (possibly swept) input frequency and  $\text{fund}_{j\_}(\text{hb})$  represents the fundamental frequency used in the corresponding HB analysis. Therefore, when analyzing a down-converting mixer, while sweeping the RF input frequency, the most relevant sideband for IF output is  $\{-1, 0\}$ . When simulating an up-converting mixer, while sweeping IF input frequency, the most relevant sideband for RF output is  $\{1, 0\}$ . You enter `sidevec` as a sequence of integer numbers, separated by spaces. The set of vectors  $\{1 \ 1 \ 0\} \{1 \ -1 \ 0\} \{1 \ 1 \ 1\}$  becomes `sidevec=[ 1 1 0 1 -1 0 1 1 1]`. For `maxsideband`, only the large tone, which is the first fundamental, is affected by this entry. All the other tones, which are the moderate tones,

## Spectre Circuit Simulator Reference

### Analysis Statements

---

are limited by `maxharms` specified for an HB analysis. Given `maxharms=[k1max k2max ... knmax]` in HB and `maxsideband=Kmax`, all  $(2 \cdot K_{\text{max}} + 1) \cdot (2 \cdot k_{2\text{max}} + 1) \cdot (2 \cdot k_{3\text{max}} + 1) \cdot \dots \cdot (2 \cdot k_{n\text{max}} + 1)$  sidebands are generated.

The number of requested sidebands changes the simulation time substantially.

With HBAC, the frequency of the stimulus and of the response are usually different (this is an important area in which HBAC differs from AC). The `freqaxis` parameter is used to specify whether the results should be output versus the input frequency (`in`), the output frequency (`out`), or the absolute value of the output frequency (`absout`).

You can specify sweep limits by specifying the end points, or the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. You can specify a step size parameter (`step`, `lin`, `log`, or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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 use the `values` parameter to specify the values that the sweep parameter should take. If you provide 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.

## HB Noise Analysis (hbnoise)

### Description

The Periodic or Quasi-Periodic Noise (HBNOISE) analysis is similar to the conventional noise analysis, except that HBNOISE analysis includes frequency conversion effects. Hence, it is useful for predicting the noise behavior of mixers, switched-capacitor filters, and other periodically or quasi-periodically driven circuits. It is particularly useful for predicting the phase noise of autonomous circuits, such as oscillators.

HBNOISE analysis linearizes the circuit about the periodic or quasi-periodic operating point computed in the prerequisite HB analysis. It is the periodically or quasi-periodically time-varying nature of the linearized circuit that accounts for the frequency conversion. In addition, the effect of a periodically or quasi-periodically time-varying bias point on the noise generated by the various components in the circuit is also included.

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 `port` as the output probe. If the output is current (or flow), choose a `vsource` or `iprobe` as the output probe.

If the input-referred noise or noise figure is desired, specify the input source using the `iprobe` parameter. For input-referred noise, use either a `vsource` or `isource` as the input probe; for noise figure, use a `port` as the probe. 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 computes 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 its `noisetemp` must be set to 16.85C (290K). 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, 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.

The reference sideband (`refsideband`) specifies which conversion gain is used when computing input-referred noise, noise factor, and noise figure. The reference sideband specifies the input frequency relative to the output frequency with:

$$|f(\text{input})| = |f(\text{out}) + \text{refsideband frequency shift}|.$$

For periodic noise (only one tone in HB analysis), 'refsideband' is a number. Use `refsideband=0` when the input and output of the circuit are at the same frequency, such as

## Spectre Circuit Simulator Reference

### Analysis Statements

---

with amplifiers and filters. When `refsideband` differs from 0, the single side-band noise figure is computed.

While for quasi-periodic noise (multiple tones in HB analysis), reference sidebands are vectors. Assume that there is one large tone and one moderate tone in HB. A sideband  $K_i$  is a vector  $[K_{i\_1} \ K_{i\_2}]$ . It gives the frequency at

$$K_{i\_1} * \text{fund}(\text{large tone of HB}) + K_{i\_2} * \text{fund}(\text{moderate tone of HB})$$

Use `refsideband=[0 0 ...]` when the input and output of the circuit are at the same frequency, such as with amplifiers and filters.

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.

The noise analysis always computes the total noise at the output, which includes contributions from the input source and the output load. The amount of the output noise that is attributable to each noise source in the circuit is also computed and output individually. If the input source is identified (using `iprobe`) and is a `vsource` or `isource`, the input-referred noise is computed, which includes the noise from the input source itself. Finally, if the input source is identified (using `iprobe`) and is noisy, as is the case with ports, the noise factor and noise figure are computed. Therefore, if:

$N_o$  = total output noise

$N_s$  = 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

$G$  = gain of the circuit

$F$  = noise factor

$NF$  = noise figure

$F_{dsb}$  = double sideband noise factor

$NF_{dsb}$  = double sideband noise figure



## Spectre Circuit Simulator Reference

### Analysis Statements

---

$F_{IEEE}$  = IEEE single sideband noise factor

$NF_{IEEE}$  = IEEE single sideband noise figure

Then:

$IRN = \sqrt{N_o^2 / G^2}$

$F = (N_o^2 - N_i^2) / N_s^2$

$NF = 10 \cdot \log_{10}(F)$

$F_{dsb} = (N_o^2 - N_i^2) / (N_s^2 + N_{si}^2)$

$NF_{dsb} = 10 \cdot \log_{10}(F_{dsb})$

$F_{IEEE} = (N_o^2 - N_i^2 - N_{so}^2) / N_s^2$

$NF_{IEEE} = 10 \cdot \log_{10}(F_{IEEE})$ .

When the results are output,  $N_o$  is named `out`,  $IRN$  is named `in`,  $G$  is named `gain`,  $F$ ,  $NF$ ,  $F_{dsb}$ ,  $NF_{dsb}$ ,  $F_{IEEE}$ , and  $NF_{IEEE}$  are named `F`, `NF`, `Fdsb`, `NFdsb`, `Fieee`, and `NFieee` respectively.

The computation of gain and IRN for quasi-periodic noise in HBNOISE assumes that the circuit under test is impedance-matched to the input source. This can introduce inaccuracy into the gain and IRN computation.

An HBNOISE analysis must follow an HB analysis.

**Note:** Note: Unlike other analyses in Spectre, this analysis can only sweep frequency.

### Syntax

Name [p] [n] ... `hbnoise parameter=value` ...

The optional terminals (p and n) specify the output of the circuit. If you do not specify the terminals, you must specify the output with a probe component.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### 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.
valuesfile		Name of the file containing the sweep values.
sweeptype	unspecified	Specifies if the sweep frequency range is the absolute frequency of input or if it is relative to the port harmonics. When the unspecified value is used, Spectre RF sweeps using relative when autonomous simulation is performed or when the analysis is PSP; for other cases Spectre RF sweeps the absolute value of the input. Possible values are absolute, relative and unspecified.
relharmvec	[...]	Sideband - vector of HB harmonics - to which relative frequency sweep should be referenced.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

stophalfund	no	If yes, automatically choose stop frequency of pnoise/hbnoise to be half of the fundamental frequency obtained from pss/hb analysis. Only works for noisetype = sampled/pmjitter/timedomain. Possible values are no and yes.
-------------	----	--

#### Probe parameters

oprobe		Compute total noise at the output defined by this component.
iprobe		Refer the output noise to this component.
refsideband	[...]	Conversion gain associated with this sideband is used when computing input-referred noise or noise figure.
refsidebandoption	individual	Whether to view the sideband as a specification of a frequency or a specification of an individual sideband. Possible values are freq and individual.

#### Sampled analysis parameters

ptvtype	timeaveraged	Specifies whether the PTV analysis will be traditional or sampled under certain conditions. Possible values are timeaveraged and sampled.
extrasampletimepoints	[...]	Additional time points for sampled PTV analysis.
sampleprobe		The crossing event at this port triggers the sampled small signal computation.
sampleratio	1	The ratio between sampled frequency and fund frequency (sampled frequency/fund frequency)..

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>noiseskipcount</code>	<code>-1</code>	Calculate time-domain noise on only one of every <code>noiseskipcount</code> time points. When $< 0$ , the parameter is ignored. When $\geq 0$ , the simulator uses this parameter and ignores <code>numberofpoints</code> .
<code>noisetimepoints</code>	<code>[...]</code>	Additional time points for time-domain noise analysis.
<code>numberofpoints</code>	<code>5</code>	Number of time points of interest in the period where time domain PSD is calculated. Simulator divides the period evenly into N segments ( $N=\text{numberofpoints}$ ) and calculates time domain PSD on the starting time point of each segment. When $< 0$ , the parameter is ignored.
<code>thresholdvalue</code>	<code>0</code>	Sampled measurement is done when the signal crosses this value.
<code>crossingdirection</code>	<code>all</code>	Specifies the transitions for which sampling must be done. Possible values are <code>all</code> , <code>rise</code> , <code>fall</code> and <code>ignore</code> .
<code>maxsamples</code>	<code>16</code>	Maximum number of sampled events to be processed during the sampled analysis.
<code>measurement</code>	<code>NULL</code>	Specifies the jitter event list that will be measured.

#### Output parameters

<code>noisetype</code>	<code>timeaverage</code>	Specifies computation of time-averaged or time-sampled noise information. Possible values are <code>timeaverage</code> , <code>correlations</code> , <code>timedomain</code> , <code>pmjitter</code> and <code>sampled</code> .
------------------------	--------------------------	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

maxsideband		This parameter determines the maximum sideband that is included when computing noise, that is either up-converted or down-converted to the output by the periodic drive signal. This parameter also determines the size of the small signal system when hbnoise is performed. It is critical for the accuracy of hbnoise analysis. Using a small maxsideband may cause accuracy loss. The default value is the <code>harms/maxharms</code> setting in the HB large signal analysis.
noiseout	usb	Specify noise output. You can set a vector like <code>noiseout=[usb am pm]</code> . And all are using single sideband(SSB) convention, half of the total power. Possible values are usb, lsb, am and pm.
sidevec	[...]	Array of relevant sidebands for the analysis.
save		Signals to output. The option 'save' specifies the signals to be saved in the result. 'allpub' saves all signals at all levels of hierarchy in the schematic, including the internal signals of device models. 'all' works like 'allpub'. 'lvl' saves all signals through the level of hierarchy set in 'nestlvl' option. 'lvlpub' works like 'lvl'. 'selected' is not recommended to use here. Possible values are all, lvl, allpub, lvlpub and selected.
nestlvl		Levels of subcircuits to output.
saveallsidebands	no	Save noise contributors by sideband. Possible values are no and yes.
output_xf	no	If set to no, there are no xf results output in hbnoise. Possible values are no and yes.
stimuli	sources	Stimuli used for XF analysis in hbnoise. Possible values are sources and nodes_and_terminals.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>separatenoise</code>	<code>no</code>	Separate noise into sources and transfer functions. Possible values are no and yes.
<code>cyclo2txtfile</code>	<code>no</code>	Output cyclo-stationary noise to text file as input source of next stage. Possible values are no and yes.

#### Convergence parameters

<code>tolerance</code>		Tolerance for linear solver. The default value is 1.0e-9 for shooting-based solver and 1.0e-4 for harmonic balance-based solver.
<code>relativeTol</code>		Relative tolerance for harmonic balance-based linear solver. Default value is 1.0e-2.
<code>resgmrescycle</code>	<code>short</code>	Restarts GMRES cycle. Possible values are instant, short, long, recycleinstant, recycleshort, recyclelong and custom.
<code>hbprecond_solver</code>	<code>autoset</code>	Select a linear solver for the GMRES preconditioner. Default is autoset. With autoset, the simulator will automatically select the appropriate preconditioner. The preconditioner affects the rate of convergence of the linear matrix solver used in periodic small-signal analysis. When autoset is selected, the simulator may decide to switch to a different preconditioner after the analysis begins. When that happens, the simulator may issue a warning instructing you to choose a different preconditioner during subsequent runs. Although not required, choosing a different preconditioner according to the simulator's instructions may speed up subsequent analyses. Possible values are basicsolver, blocksolver, autoset, blockdense, blocksolver2 and directsolver.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

lowmem	0	Harmonic balance low memory mode; Possible values are 0, 1, or number ( $\geq 10$ ). The default value is '0', the low memory mode is turned off; if '1' is set, the standard low memory mode is turned on; If a number no less than 10 is set, Spectre interprets the value as the memory requested in GigaBytes..
ppv	no	If set to yes, save the oscillator PPV after performing noise analysis. Possible values are no and yes.
augmented	yes	This parameter will be removed in future release soon. The parameter 'noiseout' is recommended. If set to yes, the frequency-aware PPV method is used to calculate the total noise of the oscillator; if set to pmonly, only the PM part of the oscillator noise is calculated; if set to amonly, only the AM part of the oscillator noise is calculated. The output of AM/PM noise is using double sideband convention. Possible values are no, yes, pmonly and amonly.
lorentzian	cornerfreqonly	This option determines if the Lorentzian plot is used in the oscillator noise analysis. 'lorentzian=yes' is only valid for 'noisetype=timeaverage'. Possible values are no, cornerfreqonly and yes.
krylov_size	200	This parameter is used to set maximum iteration count of the linear matrix solver used in periodic small-signal analysis. After reaching $1.25 \times \text{krylov\_size}$ iterations, the iteration is forced to terminate because of the poor rate of convergence. Increase the krylov_size if the simulation reports insufficient norm reduction errors in GMRES.

#### Annotation parameters

annotate	sweep	Degree of annotation. Possible values are no, title, sweep, status, steps and detailed_hb.
----------	-------	--

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>title</code>		Analysis title.
<code>oscmacrogene</code>	<code>no</code>	If set to yes, harmonic balance steady-state and phase noise data are saved. Applies only if there is pm noise out. Possible values are no and yes.
<code>oscmacrosave</code>		File to which harmonic balance steady-state solution and phase noise data are to be written.

In practice, noise can mix with each of the harmonics of the periodic drive signal applied in the HB analysis and end up at the output frequency. However, the HBNOISE analysis includes only the noise that mixes with a finite set of harmonics that are typically specified using the `maxsideband` parameter.

If  $K_i$  represents sideband  $i$ , then for periodic noise:

$$f(\text{noise\_source}) = f(\text{out}) + K_i * \text{fund}(\text{hb})$$

For quasi-periodic noise with multi-tone in HB analysis, assuming that there is one large tone and one moderate tone,  $K_i$  is represented as  $[K_{i\_1} \ K_{i\_2}]$ . Corresponding frequency shift is as follows:

$$K_{i\_1} * \text{fund}(\text{large tone of HB}) + K_{i\_2} * \text{fund}(\text{moderate tone of HB})$$

If there are  $L$  large and moderate tones in HB analysis and a set of  $n$  integer vectors representing the sidebands:

Then

$$f(\text{noise\_source}) = f(\text{out}) + \text{SUM}_{j=1\_to\_L} \{ K_{i\_j} * \text{fund}_{j\_}(\text{hb}) \}$$

The `maxsideband` parameter specifies the maximum  $|K_{i\_j}|$  included in the HBNOISE calculation. For quasi-periodic noise, only the large tone, which is the first fundamental, is affected by this entry. All the other tones, which are the moderate tones, are limited by `maxharms` specified for an HB analysis.

The number of requested sidebands changes the simulation time substantially.

You can designate a voltage to be the output by specifying a pair of nodes on the HBNOISE analysis statement or by using the 'oprobe' parameter. Any component with two or more terminals can be a voltage probe. When there are more than two terminals, they are grouped in pairs, and you use the `portv` parameter to select the appropriate pair.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

Any component that naturally computes current as an internal variable can be a current probe. If the probe component computes more than one current, you use the `porti` parameter to select the appropriate current. You must not specify both `portv` and `porti`. If you specify neither, the probe component provides a reasonable default.

You can use the `stimuli` parameter to specify what serves as the inputs for the transfer functions. There are two choices: `stimuli=sources` and `stimuli=nodes_and_terminals`.

`stimuli=sources` indicates that the sources present in the circuit are to be used. You can use the `xfmag` parameters provided by the sources to adjust the computed gain to compensate for gains or losses in a test fixture. You can limit the number of sources in hierarchical netlists by using the `save` and `nestlvl` parameters.

`stimuli=nodes_and_terminals` indicates that all possible transfer functions are to be computed. This is useful when it is not known in advance which transfer functions are interesting. 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 from a small set of terminals are computed. If you want transfer functions from specific terminals, specify the terminals in the `save` statement. You must use the `:probe` modifier (for example, `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.

You can specify sweep limits by specifying the end points, or the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. You can specify a step size parameter (`step`, `lin`, `log`, or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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 use the `values` parameter to specify the values that the sweep parameter should take. If you provide 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.

## HB S-Parameter Analysis (hbsp)

### Description

The periodic or quasi-periodic SP (HBSP) analysis is used to compute scattering and noise parameters for n-port circuits such as mixers that exhibit frequency translation. It is a small-signal analysis similar to SP analysis, except that in HBAC and HBNOISE, the circuit is first linearized about a periodically varying operating point as opposed to a simple DC operating point. Linearizing about a periodically or quasi-periodically time-varying operating point allows the computation of S-parameters between circuit ports that convert signals from one frequency band to another. HBSP can also calculate noise parameters in frequency-converting circuits. In addition, HBSP computes noise figure (both single-sideband and double-sideband), input referred noise, equivalent noise parameters, and noise correlation matrices. Similar to HBNOISE, but unlike SP, the noise features of the HBSP analysis include noise folding effects due to the periodic 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 or quasi-periodic steady-state response of the circuit to possibly large periodic stimulus is computed using HB analysis. As part of the HB analysis, the periodically time-varying representation of the circuit is computed and saved for later use. The second step is to apply small-signal excitations to compute the n-port S-parameters and noise parameters. This is done using the HBSP analysis. HBSP analysis cannot be used independently; it must follow HB analysis. However, any number of periodic small-signal analyses such as HBAC, HBSP, HBNOISE, can follow an HB analysis.

**Note:** Note: Unlike other analyses in Spectre, this analysis can only sweep frequency.

### Syntax

```
Name hbsp parameter=value ...
```

### Parameters

Sweep interval parameters

start	0	Start sweep limit.
stop		Stop sweep limit.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

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.
valuesfile		Name of the file containing the sweep values.
sweepstype	unspecified	Specifies if the sweep frequency range is the absolute frequency of input or if it is relative to the port harmonics. When the unspecified value is used, Spectre RF sweeps using relative when autonomous simulation is performed or when the analysis is PSP; for other cases Spectre RF sweeps the absolute value of the input. Possible values are absolute, relative and unspecified.

#### Port parameters

ports	[...]	List of active ports. Ports are numbered in the specified order. For noise figure computation, the input is considered as port 1 and the output as port 2.
portharmsvec	[...]	List of harmonics that are active on specified list of ports. Must have a one-to-one correspondence with the 'ports' vector.
harmsvec	[...]	List of harmonics, in addition to the ones associated with specific ports by portharmsvec, that are active.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Output parameters

freqaxis		Specifies whether the results should be printed as per the input frequency, the output frequency, or the absolute value of the input frequency. Default is in. Possible values are absin, in and out.
file		Output file name. It just supports S-parameters in PSP, HBSP, and QPSP analysis.
datafmt	spectre	Data format of the S-parameter output file. Possible values are spectre, touchstone and touchstone2.
datatype	realimag	Data type of the S-parameter output file. Possible values are realimag, magphase and dbphase.
noisedata	no	Should noise data be saved to the S-parameter output file; if yes, in what format. Possible values are no and twoport.

#### Noise parameters

donoise	yes	Perform noise analysis. If oprobe is specified as a valid port, this parameter is set to yes, and a detailed noise output is generated. Possible values are no and yes.
---------	-----	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Probe parameters

<code>maxsideband</code>	This parameter determines the maximum sideband that is included when computing noise, that is either up-converted or down-converted to the output by the periodic drive signal. This parameter also determines the size of the small signal system when <code>hbnoise</code> is performed. It is critical for the accuracy of <code>hbnoise</code> analysis. Using a small <code>maxsideband</code> may cause accuracy loss. The default value is the <code>harms/maxharms</code> setting in the HB large signal analysis.
--------------------------	--

#### Convergence parameters

<code>tolerance</code>	Tolerance for linear solver. The default value is 1.0e-9 for shooting-based solver and 1.0e-4 for harmonic balance-based solver.
<code>relativeTol</code>	Relative tolerance for harmonic balance-based linear solver. Default value is 1.0e-2.
<code>hbprecond_solver</code> <code>autoset</code>	Select a linear solver for the GMRES preconditioner. Default is <code>autoset</code> . With <code>autoset</code> , the simulator will automatically select the appropriate preconditioner. The preconditioner affects the rate of convergence of the linear matrix solver used in periodic small-signal analysis. When <code>autoset</code> is selected, the simulator may decide to switch to a different preconditioner after the analysis begins. When that happens, the simulator may issue a warning instructing you to choose a different preconditioner during subsequent runs. Although not required, choosing a different preconditioner according to the simulator's instructions may speed up subsequent analyses. Possible values are <code>basicsolver</code> , <code>blocksolver</code> , <code>autoset</code> , <code>blockdense</code> , <code>blocksolver2</code> and <code>directsolver</code> .

## Spectre Circuit Simulator Reference

### Analysis Statements

---

lowmem	0	Harmonic balance low memory mode; Possible values are 0, 1, or number ( $\geq 10$ ). The default value is '0', the low memory mode is turned off; if '1' is set, the standard low memory mode is turned on; If a number no less than 10 is set, Spectre interprets the value as the memory requested in GigaBytes..
krylov_size	200	This parameter is used to set maximum iteration count of the linear matrix solver used in periodic small-signal analysis. After reaching $1.25 \times \text{krylov\_size}$ iterations, the iteration is forced to terminate because of the poor rate of convergence. Increase the krylov_size if the simulation reports insufficient norm reduction errors in GMRES.

#### Annotation parameters

annotate	sweep	Degree of annotation. Possible values are no, title, sweep, status, steps and detailed_hb.
title		Analysis title.
sprobes	[...]	List of s-probes.
sprobeharmsvec	[...]	List of sprobe harmonics. Set 2 harmonics for each sprobe..

To specify the HBSP analysis, the port and port harmonic relations must be specified. You can select the ports of interest by setting the `port` parameter, and select the set of periodic small-signal output frequencies of interest by setting `port_harmsvec` or `harmsvec` parameters. For a given set of  $n$  integer numbers representing the harmonics  $K_1, K_2, \dots, K_n$ , the scattering parameters at each port are computed at the following frequencies:

For periodic SP in one-tone HB analysis, frequency is:

$$f(\text{scattered}) = f(\text{rel}) + K_i * \text{fund}(\text{HB})$$

For quasi-periodic noise with multi-tone in HB analysis, sidebands are vectors. Consider that you have one large tone and one moderate tone in HB. Then the above sideband  $K_1$  will be represented as  $[K_1\_1 \ K_1\_2]$ . In this case, the corresponding frequency is:

## Spectre Circuit Simulator Reference

### Analysis Statements

---

$K1\_1 * \text{fund}(\text{large tone of HB}) + K1\_2 * \text{fund}(\text{moderate tone of HB}) = \text{SUM}_{j=1\_to\_L} \{K1\_j * \text{fund}_j(\text{HB})\}$

If there are L (1 large plus L-1 moderate) tones in HB analysis and a given set of n integer vectors representing the sidebands:

$K1 = \{ K1\_1, \dots, K1\_j, \dots, K1\_L \}, K2, \dots, Kn.$

If you specify the relative frequency, the scattering parameters at each port are computed at the frequencies:

$f(\text{scattered}) = f(\text{rel}) + \text{SUM}_{j=1\_to\_L} \{K1\_j * \text{fund}_j(\text{hb})\},$

where  $f(\text{rel})$  represents the relative frequency of a signal incident on a port,  $f(\text{scattered})$  represents the frequency to which the relevant scattering parameter represents the conversion, and  $\text{fund}(\text{one-tone HB})$  or  $\text{fund}_j(\text{multi-tone HB})$  represents the fundamental frequency used in the corresponding HB analysis.

During analysis of a down-converting mixer with a blocker and the signal in the upper sideband, we sweep the input frequency of the signal coming into RF port. In case of periodic SP with one-tone HB, the most relevant harmonic for RF input is  $Ki = 1$  and for IF output  $Ki = 0$ . Therefore, we can associate  $K2 = 0$  with the IF port and  $K1 = 1$  with the RF port.  $S21$  represents the transmission of signal from the RF to IF, and  $S11$  represents the reflection of signal back to the RF port. If the signal was in the lower sideband, a choice of  $K1 = -1$  is more appropriate. For quasi-periodic SP with multi-tone HB, the most relevant sideband for this input is  $Ki = \{1, 0\}$  - and for IF output  $Ki = \{0, 0\}$ . Therefore, we can associate  $K1 = \{1, 0\}$  with the RF port and  $K2 = \{0, 0\}$  with the IF port. If the signal was in the lower sideband, then a choice of  $K1 = \{-1, 0\}$  is more appropriate.

The parameter `portharmsvec` or `harmsvec` can be used to specify the harmonics of interest. If `portharmsvec` is specified, the harmonics must be in one-to-one correspondence with the ports, with each harmonic associated with a single port. If harmonics are specified in the optional `harmsvec` parameter, all possible frequency-translating scattering parameters associated with the specified harmonics are computed.

With HBSP, the frequencies of the input and of the response are usually different (this is an important area in which HBSP differs from SP). Because the HBSP computation involves inputs and outputs at frequencies that are relative to multiple harmonics or sidebands, the `freqaxis` and `sweepstype` parameters behave differently in HBSP than in HBAC and HBNOISE.

The `sweepstype` parameter controls the way the frequencies in the HBSP analysis are swept. Specifying a `relative` sweep indicates the sweep to be relative to the analysis harmonics or port sideband (not the HB fundamental) and specifying an `absolute` sweep indicates the sweep of the absolute input source frequency.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

For example, in case of periodic SP with one-tone HB and HB fundamental of 100MHz, `portharmsvec` is set to [9 1] to examine a downconverting mixer. Using `sweepstype=relative` and a sweep range of  $f(\text{rel})=0 \rightarrow 50\text{MHz}$ , S21 represents the strength of signal transmitted from the input port in the range 900 $\rightarrow$ 950MHz to the output port at frequencies 100 $\rightarrow$ 150MHz. Using `sweepstype=absolute` and sweeping the frequency from 900 $\rightarrow$ 950MHz calculates the same quantities, because  $f(\text{abs})=900 \rightarrow 950\text{MHz}$ ,  $f(\text{rel}) = f(\text{abs}) - K1 * \text{fund}(\text{hb}) = 0 \rightarrow 50\text{MHz}$ , with  $K1=9$ , and  $\text{fund}(\text{hb}) = 100\text{MHz}$ .

For quasi-periodic noise with multi-tone HB and HB fundamentals of 1000MHz (LO) and 966MHz (blocker in RF channel), `portharmsvec` could be set to [0 1 -1 1] to examine a downconverting mixer. Consider setting `sweepstype=relative` and a sweep range of  $f(\text{rel})=-10\text{MHz} \leftrightarrow 10\text{MHz}$ , S21 represents the strength of the signal transmitted from the input port in the range 956 $\rightarrow$ 976MHz to the output port at the frequencies 24 $\leftrightarrow$ 44MHz. Using `sweepstype=absolute` and sweeping the frequency from 966 $\leftrightarrow$ 976MHz calculates the same quantities, because  $f(\text{abs})=956 \leftrightarrow 976\text{MHz}$ ,  $f(\text{rel}) = f(\text{abs}) - (K1\_1 * \text{fund\_1}(\text{hb}) + K1\_2 * \text{fund\_2}(\text{hb})) = -10\text{MHz} \leftrightarrow 10\text{MHz}$ , with  $K1\_1=0$ ,  $K1\_2=1$ ,  $\text{fund\_1}(\text{hb}) = 1000\text{MHz}$ , and  $\text{fund\_2}(\text{hb}) = 966\text{MHz}$ .

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`).

HBSP analysis also computes noise figures, equivalent noise sources, and noise parameters. The noise computation, which is skipped only when `donoise=no`, requires additional simulation time. If:

No = total output noise at frequency  $f$

Ns = noise at the output due to the input probe (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

NI = noise at the output due to the output probe (the load)

IRN = input referred noise

G = gain of the circuit

F = noise factor (single side band)

NF = noise figure (single side band)

Fdsb = double sideband noise factor



## Spectre Circuit Simulator Reference

### Analysis Statements

---

NFdsb = double sideband noise figure

Fieee = IEEE single sideband noise factor

NFieee = IEEE single sideband noise figure

Then:

$$IRN = \sqrt{N_o^2/G^2}$$

$$F = (N_o^2 - N_i^2)/N_s^2$$

$$NF = 10 \cdot \log_{10}(F)$$

$$F_{dsb} = (N_o^2 - N_i^2)/(N_s^2 + N_{si}^2)$$

$$NF_{dsb} = 10 \cdot \log_{10}(F_{dsb})$$

$$F_{ieee} = (N_o^2 - N_i^2 - N_{so}^2)/N_s^2$$

$$NF_{ieee} = 10 \cdot \log_{10}(F_{ieee}).$$

When the results are output, IRN is named `in`, G is named `gain`, F, NF, Fdsb, NFdsb, Fieee, and NFieee are named `F`, `NF`, `Fdsb`, `NFdsb`, `Fieee`, and `NFieee`, respectively. Note that the gain computed by HBSP 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.

To ensure accurate noise calculations, the `maxsideband` or `sidebands` parameters must be set to include the relevant noise folding effects. `maxsideband` is only relevant to the noise computation features of HBSP.

You can specify sweep limits by specifying the end points, or the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. You can specify a step size parameter (`step`, `lin`, `log`, or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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 use the `values` parameter to specify the values that the sweep parameter should take. If you provide 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.

## HB Stability Analysis (hbstb)

### Description

The HBSTB analysis is used to evaluate the local stability of a periodically 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.

Evaluating the stability of a periodically varying circuit is a two-step process. In the first step, the small stimulus is ignored and HB analysis is used to compute the periodic steady-state response of the circuit to a possibly large periodic stimulus. As part of the HB analysis, the periodically time-varying representation of the circuit is computed and saved for later use. In the second step, a probe is used to compute the loop gain of the zero sideband. The local stability can be evaluated using gain margin, phase margin, or a Nyquist plot of the loop gain. To perform HBSTB analysis, a probe instance must be specified as `probe` parameter.

The loop-based algorithm requires that a `probe` be placed 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. For the time-varying property of the circuit, the loop gain at different places can be different, but all values can be used to evaluate the stability. The loop-based algorithm provides stability information for single-loop circuits and for multi-loop circuits in which a `probe` component can be placed on a critical wire to break all loops. For a typical multi-loop circuit, such a critical wire may not be available. The loop-based algorithm can be used only on individual feedback loops to ensure that they are stable.

The device based algorithm requires the `probe` be a gain instant, such as a bjt transistor or a mos transistor. The device-based algorithm evaluates the loop gain around the `probe`, which can be involved in mutiloops.

**Note:** Note: Unlike other analyses in Spectre, this analysis can only sweep frequency.

### Syntax

```
Name hbstb parameter=value ...
```

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Parameters

##### Sweep interval parameters

<code>start</code>	<code>0</code>	Start sweep limit.
<code>stop</code>		Stop sweep limit.
<code>center</code>		Center of sweep.
<code>span</code>	<code>0</code>	Sweep limit span.
<code>step</code>		Step size, linear sweep.
<code>lin</code>	<code>50</code>	Number of steps, linear sweep.
<code>dec</code>		Points per decade.
<code>log</code>	<code>50</code>	Number of steps, log sweep.
<code>values</code>	<code>[...]</code>	Array of sweep values.
<code>valuesfile</code>		Name of the file containing the sweep values.

##### Probe parameters

<code>probe</code>	Probe instance around which the loop gain is calculated.
<code>localgnd</code>	Node name of local ground. If not specified, the probe is referenced to global ground.

##### Output parameters

<code>save</code>	Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, none and nooutput.
-------------------	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>nestlvl</code>		Levels of subcircuits to output.
----------------------	--	----------------------------------

#### Convergence parameters

<code>tolerance</code>		Tolerance for linear solver. The default value is 1.0e-9 for shooting-based solver and 1.0e-4 for harmonic balance-based solver.
------------------------	--	--

<code>relativeTol</code>		Relative tolerance for harmonic balance-based linear solver. Default value is 1.0e-2.
--------------------------	--	---

<code>gear_order</code>	2	Gear order used for small-signal integration.
-------------------------	---	---

<code>solver</code>	turbo	Solver type. Possible values are std, turbo, std_hh and turbo_hh.
---------------------	-------	---

<code>oscsolver</code>	turbo	Oscillator solver type. It is recommended that you use ira for huge circuits. Possible values are std, turbo, ira and direct.
------------------------	-------	---

<code>resgmrescycle</code>	short	Restarts GMRES cycle. Possible values are instant, short, long, recycleinstant, recycleshort, recyclelong and custom.
----------------------------	-------	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>hbprecond_solver</code>	<code>autoset</code>	Select a linear solver for the GMRES preconditioner. Default is <code>autoset</code> . With <code>autoset</code> , the simulator will automatically select the appropriate preconditioner. The preconditioner affects the rate of convergence of the linear matrix solver used in periodic small-signal analysis. When <code>autoset</code> is selected, the simulator may decide to switch to a different preconditioner after the analysis begins. When that happens, the simulator may issue a warning instructing you to choose a different preconditioner during subsequent runs. Although not required, choosing a different preconditioner according to the simulator's instructions may speed up subsequent analyses. Possible values are <code>basicsolver</code> , <code>blocksolver</code> , <code>autoset</code> , <code>blockdense</code> , <code>blocksolver2</code> and <code>directsolver</code> .
<code>krylov_size</code>	<code>200</code>	This parameter is used to set maximum iteration count of the linear matrix solver used in periodic small-signal analysis. After reaching <code>1.25*krylov_size</code> iterations, the iteration is forced to terminate because of the poor rate of convergence. Increase the <code>krylov_size</code> if the simulation reports insufficient norm reduction errors in GMRES.

#### Annotation parameters

<code>annotate</code>	<code>sweep</code>	Degree of annotation. Possible values are <code>no</code> , <code>title</code> , <code>sweep</code> , <code>status</code> , <code>steps</code> and <code>detailed_hb</code> .
<code>title</code>		Analysis title.

You can specify sweep limits by specifying the end points, or the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. You can specify a step size parameter (`step`, `lin`, `log`, or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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 use the `values` parameter

## **Spectre Circuit Simulator Reference**

### **Analysis Statements**

---

to specify the values that the sweep parameter should take. If you provide 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.

## HB XF Analysis (hbxf)

### Description

A conventional transfer function analysis computes the transfer function from every source in the circuit to a single output. Unlike a conventional AC analysis that computes the response from a single stimulus to every node in the circuit, the Harmonic Balance Transfer Function or HBXF analysis computes the transfer functions from any source at any frequency to a single output at a single frequency. Thus, like HBAC analysis, HBXF analysis includes frequency conversion effects.

The HBXF analysis directly computes such useful quantities as conversion efficiency (transfer function from input to output at required frequency), image and sideband rejection (input to output at undesired frequency), and LO feed-through and power supply rejection (undesired input to output at all frequencies).

As with a HBAC, HBSP, and HBNoise analyses, a HBXF analysis must follow a HB analysis.

**Note:** Note: Unlike other analyses in Spectre, this analysis can only sweep frequency.

### Syntax

```
Name [p] [n] ... hbxf parameter=value ...
```

The optional terminals (p and n) specify the output of the circuit. If you do not specify the terminals, you must specify the output with a probe component.

### 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.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

lin	50	Number of steps, linear sweep.
dec		Points per decade.
log	50	Number of steps, log sweep.
values	[...]	Array of sweep values.
valuesfile		Name of the file containing the sweep values.
sweepstype	unspecified	Specifies if the sweep frequency range is the absolute frequency of input or if it is relative to the port harmonics. When the unspecified value is used, Spectre RF sweeps using relative when autonomous simulation is performed or when the analysis is PSP; for other cases Spectre RF sweeps the absolute value of the input. Possible values are absolute, relative and unspecified.
relharmvec	[...]	Sideband - vector of HB harmonics - to which relative frequency sweep should be referenced.

#### Probe parameters

probe	Compute every transfer function to this probe component.
-------	--

#### Sampled analysis parameters

ptvtype	timeaveraged	Specifies whether the PTV analysis will be traditional or sampled under certain conditions. Possible values are timeaveraged and sampled.
sampleprobe		The crossing event at this port triggers the sampled small signal computation.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>thresholdvalue</code>	<code>0</code>	Sampled measurement is done when the signal crosses this value.
<code>crossingdirection</code>	<code>all</code>	Specifies the transitions for which sampling must be done. Possible values are all, rise, fall and ignore.
<code>maxsamples</code>	<code>16</code>	Maximum number of sampled events to be processed during the sampled analysis.
<code>extrasampletimepoints</code>	<code>[...]</code>	Additional time points for sampled PTV analysis.
<code>sampleratio</code>	<code>1</code>	The ratio between sampled frequency and fund frequency (sampled frequency/fund frequency)..

#### Output parameters

<code>sidevec</code>	<code>[...]</code>	Array of relevant sidebands for the analysis.
<code>maxsideband</code>		This parameter determines the maximum sideband that is included when computing noise, that is either up-converted or down-converted to the output by the periodic drive signal. This parameter also determines the size of the small signal system when hbnoise is performed. It is critical for the accuracy of hbnoise analysis. Using a small maxsideband may cause accuracy loss. The default value is the <code>harms/maxharms</code> setting in the HB large signal analysis.
<code>freqaxis</code>		Specifies whether the results should be printed as per the input frequency, the output frequency, or the absolute value of the input frequency. The default is <code>absin</code> . Possible values are <code>absin</code> , <code>in</code> and <code>out</code> .
<code>save</code>		Signals to output. Possible values are <code>all</code> , <code>lvl</code> , <code>allpub</code> , <code>lvlpub</code> , <code>selected</code> , <code>none</code> and <code>nooutput</code> .

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>nestlvl</code>		Levels of subcircuits to output.
----------------------	--	----------------------------------

<code>stimuli</code>	<code>sources</code>	Stimuli used for XF analysis in hbnoise. Possible values are <code>sources</code> and <code>nodes_and_terminals</code> .
----------------------	----------------------	--

#### Convergence parameters

<code>tolerance</code>		Tolerance for linear solver. The default value is 1.0e-9 for shooting-based solver and 1.0e-4 for harmonic balance-based solver.
------------------------	--	--

<code>relativeTol</code>		Relative tolerance for harmonic balance-based linear solver. Default value is 1.0e-2.
--------------------------	--	---

<code>resgmrescycle</code>	<code>short</code>	Restarts GMRES cycle. Possible values are <code>instant</code> , <code>short</code> , <code>long</code> , <code>recycleinstant</code> , <code>recycleshort</code> , <code>recyclelong</code> and <code>custom</code> .
----------------------------	--------------------	--

<code>hbprecond_solver</code>	<code>autoset</code>	Select a linear solver for the GMRES preconditioner. Default is <code>autoset</code> . With <code>autoset</code> , the simulator will automatically select the appropriate preconditioner. The preconditioner affects the rate of convergence of the linear matrix solver used in periodic small-signal analysis. When <code>autoset</code> is selected, the simulator may decide to switch to a different preconditioner after the analysis begins. When that happens, the simulator may issue a warning instructing you to choose a different preconditioner during subsequent runs. Although not required, choosing a different preconditioner according to the simulator's instructions may speed up subsequent analyses. Possible values are <code>basicsolver</code> , <code>blocksolver</code> , <code>autoset</code> , <code>blockdense</code> , <code>blocksolver2</code> and <code>directsolver</code> .
-------------------------------	----------------------	--

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>lowmem</code>	<code>0</code>	Harmonic balance low memory mode; Possible values are 0, 1, or number ( $\geq 10$ ). The default value is '0', the low memory mode is turned off; if '1' is set, the standard low memory mode is turned on; If a number no less than 10 is set, Spectre interprets the value as the memory requested in GigaBytes..
<code>lorentzian</code>	<code>cornerfreqonly</code>	This option determines if the Lorentzian plot is used in the oscillator noise analysis. 'lorentzian=yes' is only valid for 'noisetype=timeaverage'. Possible values are no, cornerfreqonly and yes.
<code>krylov_size</code>	<code>200</code>	This parameter is used to set maximum iteration count of the linear matrix solver used in periodic small-signal analysis. After reaching $1.25 \times \text{krylov\_size}$ iterations, the iteration is forced to terminate because of the poor rate of convergence. Increase the <code>krylov_size</code> if the simulation reports insufficient norm reduction errors in GMRES.

#### Annotation parameters

<code>annotate</code>	<code>sweep</code>	Degree of annotation. Possible values are no, title, sweep, status, steps and detailed_hb.
<code>title</code>		Analysis title.

The variable of interest at the output can be voltage or current, and its frequency is not constrained by the period of the large periodic solution. While sweeping the selected output frequency, you can select the periodic small-signal input frequencies of interest by setting either the `maxsideband` or the `sidevec` parameter. For a given set of  $n$  integer numbers representing the sidebands  $K_1, K_2, \dots, K_n$ , the input signal frequency at each sideband is computed as  $f(\text{in}) = f(\text{out}) + K_i \cdot f_{\text{fund}}(\text{pss})$ , where,  $f(\text{out})$  represents the (possibly swept) output signal frequency and  $f_{\text{fund}}(\text{HB})$  represents the fundamental frequencies used in the corresponding HB analysis. Thus, when analyzing a down-converting mixer and sweeping the IF output frequency,  $K_i = +1$  for the RF input represents the first upper-sideband, while  $K_i = -1$  for the RF input represents the first lower-sideband. By setting the `maxsideband` value to  $K_{\text{max}}$ , all  $2 \cdot K_{\text{max}} + 1$  sidebands from  $-K_{\text{max}}$  to  $+K_{\text{max}}$  are selected.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

The number of requested sidebands does not change substantially the simulation time. However, the `maxacfreq` of the corresponding HB analysis should be set to guarantee that  $|\max\{f(\text{in})\}|$  is less than `maxacfreq`; otherwise, the computed solution might be contaminated by aliasing effects. The HBXF simulation is not executed for  $|f(\text{out})|$  greater than `maxacfreq`. Diagnostic messages are printed for those extreme cases, indicating how `maxacfreq` should be set in the HB analysis. In majority of simulations, however, this is not an issue, because `maxacfreq` is never allowed to be smaller than 40x the HB fundamental.

With HBXF, the frequency of the stimulus and of the response are usually different (this is an important area in which HBXF differs from XF). The `freqaxis` parameter is used 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`).

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, and you 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 HBXF analysis statement.

Any component that naturally computes current as an internal variable can be a current probe. If the probe component computes more than one current, you use the `porti` parameter to select the appropriate current. It is an error to specify both `portv` and `porti`. If neither is specified, the probe component provides a reasonable default.

The `stimuli` parameter specifies the inputs for the transfer functions. There are two choices. `stimuli=sources` indicates that the sources present in the circuit should be used. The `xfmag` parameters provided by the sources may be used to adjust the computed gain to compensate for gains or losses in a test fixture. One can limit the number of sources in hierarchical netlists by using the `save` and `nestlvl` parameters. `stimuli=nodes_and_terminals` indicates that all possible transfer functions should be computed.

This is useful when it is not known in advance which transfer functions are interesting. 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 from a small set of terminals are computed. If transfer functions from specific terminals are required, specify the terminals in the `save` statement. You must use the `:probe` modifier (for example, `Rout:1:probe`) or specify `useprobes=yes` on the options statement. If transfer functions from all terminals are required, specify `currents=all` and `useprobes=yes` on the options statement.

You can specify sweep limits by specifying the end points, or the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the

## Spectre Circuit Simulator Reference

### Analysis Statements

---

size of each step. You can specify a step size parameter (`step`, `lin`, `log`, or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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 use the `values` parameter to specify the values that the sweep parameter should take. If you provide 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.

## Circuit Information (info)

### Description

The circuit information analysis outputs several types of information about the circuit and its components. You can use various filters to specify what information is output. You can create a listing of model, instance, temperature-dependent, input, output, and operating point parameters. You can also generate a summary of the minimum and maximum parameter values (by using `extremes=yes` or `only`). Finally, you can request that Spectre provides a node-to-terminal map (by using `what=terminals`) or a terminal-to-node map (by using `what=nodes`).

The following are brief descriptions of the types of parameters you can request with the `info` statement:

**Input parameters:** Parameters that you specify in the netlist, such as the given length of a MOSFET or the saturation current of a bipolar transistor (use `what=inst, models, input, or all`)

**Output parameters:** Parameters that are computed by Spectre, such as temperature-dependent parameters and the effective length of a MOSFET after scaling (use `what=output or all`)

**Operating-point parameters:** Parameters that depend on the actual solution computed (use `what=oppoint`)

**Initial condition file creation:** write node voltages and source currents into a file which can be read by the transient analysis `readic/readns` statement.

### Syntax

```
Name info parameter=value ...
```

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Parameters

what	oppoint	The parameters that should be printed. Possible values are none, inst, models, input, output, nodes, all, terminals, oppoint, captab, parameters, primitives, subckts, assert, allparameters, netlist, options, dumpall, bridges, opens, customreplace, custominsert, customopen, faultparam, allcap, netcap, moscap, dhe and simstat.
where	logfile	Where the parameters should be printed. Asserts can only be written to rawfile. Possible values are nowhere, screen, file, logfile and rawfile.
file	%C:r.inf o.what	File name when where=file.
save		Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, none and nooutput.
nestlvl		Levels of subcircuits to output.
extremes	yes	Print minimum and maximum values. Possible values are no, yes and only.
title		Analysis title.
descriptions	no	Print descriptions. Possible values are no and yes.
subckts	[...]	Customize for filtering out wanted subckts names from the netlist. Only support for output in logfile or file..
subckt_parameter s	[...]	Customize for filtering out wanted subckt parameter names from the netlist. Only support for output in logfile or file..

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Operating point parameters

oppoint	all	Operating point type when 'what=oppoint', all - operating point for devices, stacked devices, node voltages, and source currents. node - operating point for node voltages, and source currents. dev - operating point for devices. subckt - operating point for stacked devices. alldev - operating point for regular and stacked devices. stacked device parameter reporting requires Spectre option subcktoppoint=yes, and save statement for stacked device parameters of interest. Possible values are all, node, dev, subckt and alldev.
---------	-----	--

#### Initial condition file creation

optype	ic	Print Spectre format operating point file which can be read with readic/readns in transient statement. Time for writing operating point file needs to be defined in infotimes statement. Possible values are ic and nodeset.
--------	----	--

#### Fault generation parameters

faultblock		Name of the fault block when fault list generation requested.
faultdev	[...]	Fault devices defined by primitive names, subckt names, or model names.
faultcustom	[...]	Custom fault subcircuit defined by subckt model name from the netlist.
faultres	[...]	Resistance value for bridges or opens (default 100 Ohm for bridges, 1G Ohm for opens).
faultcap	0	Capacitance value for open faults.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>faultterminals</code>	<code>[...]</code>	List of device terminals to be considered for fault generation (default is all terminals of <code>faultdev</code> ).
<code>faultstuckat</code>	<code>[...]</code>	Supply node name(s) to generate stuck-at fault list when 'what=bridges'.
<code>faultextract</code>	<code>no</code>	Enables the generation of a layout-based fault list when DSPF files are included in the netlist. Set it to: 'no' to disable the generation of a layout-based fault list, only the schematic-based faults are generated, this is the default value; 'spf' to generate a list of faults that are compatible with a layout-based netlist; 'sch' to generate a list of faults that are compatible with a schematic-based netlist. Possible values are no, spf and sch.
<code>faultcmin</code>	<code>1e-20</code>	Defines the minimum capacitance value an extracted parasitic capacitor must have to be identified as a bridge defect during the generation of faults.
<code>faultrmin</code>	<code>0.001</code>	Defines the minimum resistance value an extracted parasitic resistor must have to be identified as an open defect during the generation of faults.
<code>subckt</code>	<code>[...]</code>	Faults are generated for all instances of the specified subcircuits.
<code>inst</code>	<code>[...]</code>	Faults are generated for the specified subcircuit instances.
<code>xsubckt</code>	<code>[...]</code>	All instances of the specified subcircuits are excluded from fault generation.
<code>xinst</code>	<code>[...]</code>	All specified subcircuit instances are excluded from fault generation.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

faultlevel		Defines the hierarchical level of the design to be considered during the fault list generation. By default, Spectre generates faults in subcircuits at all levels of the design.
faultlayer	[...]	Layer numbers on which layout-based faults are to be identified. Set this option to 0 to identify faults on all layers including the connectivity between layers. Set it to a set of layer numbers, [x y ...], to identify faults on those layers, or set it to a range, [(x,y)], to identify faults for all layers in the range, excluding the connectivity between layers in both cases.
faultmaxcount	100,000	Upper bound for number of faults to be generated before collapsing.
faultrule	none	Comply with IEEE 2427 requirements for fault list generation. Possible values are none and 2427.
faultuniverse		Name of the file to save the defect universe details after fault list generation.
faultcollapse	yes	If yes, only one fault will be included into the fault list for each set of equivalent faults having the same topology. Possible values are no and yes.
weight_expr		Expression to define fault weighting function.
weight_factor	1.	Value to define weighting factor for given block in the fault list.
faultduplicate	yes	If no, the equivalent faults will be excluded from the fault list. In addition, the duplicated faults will be excluded when several info analyses are specified for fault generation. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

faultactivethresh	0	The minimum activity ratio of the circuit element to identify the related faults as potentially undetectable. The value is calculated as the ratio of time points when the element was active to the total number of time points.
faultparam	[...]	Parameter names to generate fault list when 'what=faultparam'.
faultvalues	[...]	Parameter values for fault generation when 'what=faultparam'.
faultpercentage	[...]	Percentage in parameter value deviation for fault generation when 'what=faultparam'.
faultfile		File name containing parameter values when 'what=faultparam'.
faultstart	[...]	Parameter initial values for fault generation when 'what=faultparam'.
faultstep	[...]	Step size to sweep parameter values for fault generation when 'what=faultparam'.
faultstop	[...]	Parameter's final value for fault generation when 'what=faultparam'.

#### Design hierarchy extraction parameters for Midas Safety Platform

dheminarea	0 $\mu\text{m}^2$	Lower bound of area value for device to be considered during design hierarchy extraction.
dhesubckt	[...]	Design hierarchy is generated for all instances of the specified subcircuits.
dheinst	[...]	Design hierarchy is generated for the specified subcircuit instances.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

dhexsubckt	[...]	All instances of the specified subcircuits are excluded from the design hierarchy.
dhexinst	[...]	The specified subcircuit instances are excluded from the design hierarchy.
dheparams		Name of the file that provides the rules to calculate area for subcircuits when 'what=dhe'.

#### Captab parameters

detail	node	How detailed should the capacitance table be. Possible values are node, nodetoground and nodetonode.
sort	name	How to sort the capacitance table. Possible values are name and value.
filter		If set to "rc" nodes within RC net will be dropped. Possible values are none and rc.
threshold	0 F	Threshold value for printing capacitances (ignore capacitances smaller than this value).
skip_ground		Whether capt skip ground or not. Possible values are yes and no.
node	[...]	Nodes for which captab analyses will performed.
intrinsic_cap_merge	no	Merge the internal captab node to external node. Possible values are no and yes.

#### Example initial condition file creation

```
tran1 tran stop=10n infotimes=[3n 4n] infonames=[myinfo1 myinfo2]
myinfo1 info what=oppoint optype=ic where=file file='3n.ic'
myinfo2 info what=oppoint optype=nodeset where=file file='4n.ic'
```

## Spectre Circuit Simulator Reference

### Analysis Statements

---

## LF Analysis (lf)

### Description

The LF analysis begins by linearizing the circuit about an operating point. This analysis detects poles that may potentially cause stability problems, and identifies loops that associate to potentially problematic poles.

### Syntax

```
Name lf parameter=value ...
```

### 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.
valuesfile		Name of the file containing the sweep values.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Sweep variable parameters

dev	Device instance whose parameter value is to be swept.
mod	Model whose parameter value is to be swept.
param	Name of parameter to sweep.

#### State-file parameters

readns	File that contains estimate of DC solution (nodeset).
write	DC operating point output file at the first step of the sweep.
writefinal	DC operating point output file at the last step of the sweep.

#### Output parameters

oppoint	no	Determines whether operating point information should be computed:if yes, where should it be printed (screen or file). Operating point information is not printed if the operating point computed in the previous analysis remains unchanged. Possible values are no, screen, logfile and rawfile.
---------	----	--

#### Convergence parameters

restart	yes	Restart the DC solution from scratch if any condition has changed. If not, use the previous solution as initial guess. Possible values are no and yes.
---------	-----	--

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Annotation parameters

<code>annotate</code>	<code>sweep</code>	Degree of annotation. Possible values are no, title, sweep, status and steps.
-----------------------	--------------------	---

#### Miscellaneous parameters

<code>order</code>		Order of expansion around each pole to generate residue information. This option is used for model order reduction in node impedance computation.
--------------------	--	---

<code>sensitivity</code>	<code>3</code>	Sensitivity of the krylov method of pole detection and the accuracy of impedance computation. Higher values increase runtime while reduce the chance of missing a pole and improve the accuracy of impedance values. Possible values are 1, 2, 3, 4, and 5.
--------------------------	----------------	---

<code>solver_method</code>	<code>auto</code>	If set to <code>krylov</code> , the modified rational krylov method is used for pole detection; if set to <code>direct</code> , QZ method is used; if set to <code>auto</code> , <code>loopfinder</code> chooses the method for pole detection automatically. Possible values are <code>auto</code> , <code>direct</code> and <code>krylov</code> .
----------------------------	-------------------	---

<code>romsize</code>	<code>6000</code>	Minimum number of nodes for applying the Krylov method. This option is used if <code>solver_method=auto</code> .
----------------------	-------------------	--

<code>zmin</code>	<code>0.1 <math>\Omega</math></code>	Minimum DC impedance of interest in loop identification.
-------------------	--------------------------------------	--

<code>dampmax</code>	<code>0.7</code>	Maximum damping ratio, i.e. negative value of $\cos(\phi)$ where $\phi$ is the argument of the corresponding pole, for targeting loops. This option is used in pole detection.
----------------------	------------------	--

<code>fregtol</code>	<code>(Hz)</code>	Relative error tolerance for natural frequencies of loops if modified rational krylov method is used for pole detection.
----------------------	-------------------	--

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>freqmax</code>	<code>1e10 Hz</code>	Maximum natural frequency for targeting loops.
<code>freqmin</code>	<code>1e-2 Hz</code>	Minimum natural frequency for targeting loops.
<code>emptyloop</code>	<code>yes</code>	If set to <code>yes</code> , empty loops, if any, are printed in the output; if set to <code>no</code> , only non-empty loops are printed. Possible values are <code>no</code> and <code>yes</code> .
<code>doublekrylov</code>		If set to <code>no</code> , krylov flow searches the region of interest once; if set to <code>yes</code> , krylov flow searches the region of interest twice to minimize the change of missing poles. Possible values are <code>no</code> and <code>yes</code> .
<code>print_all_poles</code>	<code>no</code>	If set to <code>yes</code> , print full list of poles to the log file; if set to <code>no</code> , do not print full list of poles. This option is used in direct solver method. Possible values are <code>no</code> and <code>yes</code> .
<code>prevoppoint</code>	<code>no</code>	Use the operating point computed on the previous analysis. Possible values are <code>no</code> and <code>yes</code> .
<code>force</code>	<code>none</code>	The set of initial conditions to use. Possible values are <code>none</code> , <code>node</code> , <code>dev</code> and <code>all</code> .
<code>readforce</code>		File that contains initial conditions.
<code>skipdc</code>	<code>no</code>	Skip DC analysis. Possible values are <code>no</code> and <code>yes</code> .
<code>useprevic</code>	<code>no</code>	If set to <code>yes</code> or <code>ns</code> , use the converged initial condition from previous analysis as <code>ic</code> or <code>ns</code> . Possible values are <code>no</code> , <code>yes</code> and <code>ns</code> .

By default, this analysis computes the operating point, if it is not known, or recomputes it if any significant component or circuit parameter has changed. However, if an operating point was computed during a previous analysis, you can set `prevoppoint=yes` to avoid recomputing it. For example, if `prevoppoint=yes` and the previous analysis was a transient analysis, the operating point is the state of the circuit at the final time point.

Nodesets help find the DC or the initial transient solution. You can specify nodesets in the circuit description file with `nodeset` statements or in a separate file by using the `readns`



## Spectre Circuit Simulator Reference

### Analysis Statements

---

parameter. When nodesets are specified, Spectre computes an initial guess of the solution by performing DC analysis, while forcing the specified values on to nodes by using a voltage source in series with a resistor whose resistance is `rforce`. Spectre then removes these voltage sources and resistors and computes the required solution from this initial guess.

Nodesets have two important uses. First, if a circuit has two or more solutions, nodesets can bias the simulator towards computing the required solution. Second, this is a convergence aid. By estimating the solution of the largest possible number of nodes, you might be able to eliminate a convergence problem or significantly speed up convergence.

When you simulate the same circuit multiple times, it is recommended that you use both `write` and `readns` parameters and assign the same file name to both parameters. DC analysis then converges quickly even if the circuit has changed since the last simulation, and the nodeset file is automatically updated.

The loopfinder analysis does not work with frequency-defined or distributed devices properly. If there are frequency-defined or distributed devices, poles and node impedance are computed by approximating those devices as equivalent conductances and capacitances evaluated at 1Hz.

The default value of options `fregtol`, `order` and `doublekrylov` are controlled by the setting of the option `sensitivity`. The effect of `sensitivity` on other parameters is shown in the following table.

<code>sensitivity</code>	<code>fregtol</code>	<code>order</code>	<code>doublekrylov</code>
--------------------------	----------------------	--------------------	---------------------------

-----

5	1.0e-6	7	yes
4	1.0e-4	5	no
3	1.0e-3	3	no
2	3.0e-3	2	no
1	3.0e-3	2	no

## Load Pull Analysis (loadpull)

### Description

Loadpull sweeps the magnitude ( $\rho$ ) and phase ( $\phi$ ) of the load instance over a specified range to detect potentially unstable amplifier load impedances.

The component in the netlist that is used to set the magnitude and phase of the reflection coefficient of the loadpull simulation is called the load instance and

is different for shooting and harmonic balance. For harmonic balance, the load instance must be either a port or a portAdapter. For shooting pss, the load

instance must be a portAdapter.

In the loadpull statement, you can define the sweeps for the magnitude and phase (in degrees) for the reflection coefficient of the load instance. To calculate

the amplifier output, either hb or pss is required. The hb or pss statement is enclosed in curly brackets after the loadpull statement. Harmonic balance is

likely to be faster than shooting PSS for most amplifiers.

### Syntax

```
Name loadpull parameter=value ...
```

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Parameters

tuner		Set the instance to be swept in loadpull MWO flow, If tuner is a load port, loadpull is performed. If tuner is a source port, sourcepull analysis is performed. Only loadpull is supported currently.
sinst		Set the source instance in loadpull MWO flow when running loadpull analysis. The impedance file of source instance is used in MWO to compute the contours for measurements like PAE, Source Voltage, Source Current and so on.
maxharmoutput	3	The maximum harmonic number to be saved in the loadpull MWO dataset..
paramset		Name of parameter set in loadpull MWO flow, gmag and gtheta are set in paramset.
irfterm		Set the input RF terminal in loadpull MWO flow.
irfnet		Set the input RF net in loadpull MWO flow.
orfterm		Set the output RF terminal in loadpull MWO flow.
orfnet		Set the output RF net in loadpull MWO flow.
idcterm		Set the input DC terminal in loadpull MWO flow.
idcnet		Set the input DC net in loadpull MWO flow.
odcterm		Set the output DC terminal in loadpull MWO flow.
odcnet		Set the output DC net in loadpull MWO flow.
z0	50	Used only in harmonic balance only when the load instance is a port and sets the impedance of the load port. When a portAdapter is used as the load instance, this value is set in the portAdapter instance.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>inst</code>		Used only in harmonic balance when the load instance is a port. This is the instance name for the port that loads the amplifier. When a portAdapter is used as the load instance, the portAdapter that uses the rho and Phi values in the netlist is assumed to be the load instance.
<code>rho</code>		Name of parameter to rho sweep. The name is arbitrary and must be in the parameters list. When a portAdapter is used as the load instance, the parameter must be used in the portAdapter to set the reflection coefficient magnitude.
<code>rhostart</code>	0	Start sweep limit of rho.
<code>rhostop</code>		Stop sweep limit of rho.
<code>rhostep</code>		Step size, linear sweep of rho.
<code>rholin</code>	50	Number of steps, linear sweep of rho.
<code>rhovalues</code>	[...]	Array of sweep values of rho.
<code>phi</code>		Name of parameter to phi sweep. The name is arbitrary and must be in the parameters list. When a portAdapter is used as the load instance, the parameter must be used in the portAdapter to set the reflection coefficient angle.
<code>phistart</code>	0	Start sweep limit of phi.
<code>phistop</code>		Stop sweep limit of phi.
<code>phistep</code>		Step size, linear sweep of phi.
<code>philin</code>	50	Number of steps, linear sweep of phi.
<code>phivalues</code>	[...]	Array of sweep values of phi.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

=====

#### Examples

=====

#### Harmonic Balance:

```
lphb loadpull inst=PORT4 rho=mag rhostart=0.001 rhostop=0.991
+  rhostep=0.11 phi=theta phistart=-180 phistop=160 phistep=20
+  z0=50 {
    hb hb tstab=0 oversample=[1] fundfreqs=[(2.45G)] maxharms=[6]
+  errpreset=moderate annotate=status
}
```

#### Shooting PSS:

```
lppss loadpull rho=mag rhostart=0.001 rhostop=0.991
+  rhostep=0.11 phi=theta phistart=-180 phistop=160 phistep=20 {
    pss pss fund=2.45G harms=7 errpreset=moderate tstab=3n
+  annotate=status
}
```

#### Loadpull MWO flow:

```
lphb loadpull tuner=PORT2 sint=PORT1 maxharmoutput=3
+  irfterm=IPRB_SOURCE:in irfnet=source orfterm=IPRB_LOAD:in orfnet=load
+  idcnet=net5 odcterm=V1:p odcnet=net10 rho=gmag phi=gtheta
+  z0=50 paramset=ldpsethb {
    sweepFreq sweep param=fin paramtype=freq start=2.2G stop=2.4G step=0.1G{
    sweepPower sweep param=Pwr paramtype=pwr start=-10 stop=10 step=1{
    hb hb autotstab=yes oversample=[1] fundfreqs=[(fin)]
```

## Spectre Circuit Simulator Reference

### Analysis Statements

---

```
+    maxharms=[11] errpreset=conservative annotate=status  
    }  
    }  
}
```

## Monte Carlo Analysis (montecarlo)

### Description

The `montecarlo` analysis is a swept analysis with associated child analyses similar to the sweep analysis (see `spectre -h sweep`.) The Monte Carlo analysis refers to "statistics blocks" where statistical distributions and correlations of netlist parameters are specified (detailed information about statistics blocks is given below). For each iteration of the Monte Carlo analysis, new pseudo-random values are generated for the specified netlist parameters (according to their specified distributions) and the list of child analyses are then executed.

Expressions are associated with the child analyses. These expressions, which you constructed as scalar calculator expressions during Monte Carlo analysis setup, can be used to measure circuit metrics, such as the slew-rate of an op-amp. For each iteration during a Monte Carlo analysis, the expression results vary with the netlist parameters. Therefore, Monte Carlo analysis allows you to examine and predict circuit performance variations, which affect yield.

The statistics blocks allow you to specify batch-to-batch (process) and per-instance (mismatch) variations for netlist parameters. These statistically-varying netlist parameters can be referenced by models or instances in the main netlist and may represent IC manufacturing process variation or component variations for board-level designs. The following description gives a simplified example of the Monte Carlo analysis flow:

```
perform nominal run if requested

if any errors in nominal run then stop

foreach Monte Carlo iteration {
    if process variations specified then
        apply process variation to parameters

    if mismatch variations specified then
        foreach subcircuit instance {
            apply mismatch variation to parameters
        }
}
```

## Spectre Circuit Simulator Reference

### Analysis Statements

---

```
foreach child analysis {  
    run child analysis  
    evaluate expressions  
}  
}
```

### Syntax

Name montecarlo parameter=value ...

### Parameters

#### Analysis parameters

numruns	100	Number of Monte Carlo iterations to perform (does not include the nominal run).
firstrun	1	Starting iteration number.
runpoints	[...]	Specifies a set of iteration indices to be simulated. Indices can be any positive integer values or values defined by a range function.
saverunpointsfile		Save file that contains the actual run points (iteration indices).
loadrunpointsfile		Load file that contains the run points (iteration indices) to be simulated.
variations	process	Level of statistical variation to apply. Possible values are process, mismatch and all.
sampling	standard	Method of statistical sampling to apply. Possible values are standard, lds and lhs.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

numbins	0	Number of bins for latin-hypercube (lhs). The number is checked against numruns + firstrun - 1, and Max( numbins, numruns + firstrun - 1 ) is used.
seed		Optional starting seed for random number generator.
config		Yaml file to set up Monte-Carlo configurations.
scalarfile		Output file that contains output scalar data.
paramfile		Output file that contains output scalar data labels.
dut	[...]	If set, the specified subcircuit instance have process and mismatch variations applied and the unspecified instance only have process variations applied. All subcircuits instantiated under this instance also have process and mismatch enabled. By default, mismatch is applied to all subcircuit instances in the design and process is applied globally. This parameter allows the test-bench to change and not affect the variations seen by the actual design.
ignore	[...]	If set, no variation is applied to specified subcircuit instances. In addition, all subcircuits instantiated under this instance do not have variation enabled. By default, the mismatch is applied to all subcircuit instances in the design and the process is applied globally.
dutparams	[...]	If set, only the specified statistical parameters have process and mismatch variations applied.
ignoreparams	[...]	If set, the specified statistical parameters are excluded from applying process and mismatch variation.
json	no	Output JSON file saves the variations and analysis settings. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>accuracyaware</code>	<code>summary</code>	Specifies the mode of run-time monitoring and early-termination of a montecarlo simulation. In summary mode, the statistics of measurement is only listed after the Monte-Carlo analysis finishes. In iteration mode, the statistics is printed out after each step of Monte-Carlo analysis. In autostop mode, analysis is terminated based on the criteria specified by the options <code>minmaxpairs</code> , and <code>smooththresh</code> . Possible values are <code>summary</code> , <code>iteration</code> , <code>autostop</code> and <code>none</code> .
<code>minmaxpairs</code>	<code>[...]</code>	Pairs of values that are used to specify the min and max of each measurement defined in ocean expressions. Simulation is terminated when the current iteration generates a measurement that resides outside the region of <code>[min, max]</code> . It is not necessary that the number of pairs equals the number of measurement. However, each defined pair has to be aligned with the corresponding ocean measurement. Extra number of pairs or measurement will be ignored when deciding upon early termination. This option is only active when <code>accuracyaware=autostop</code> .
<code>smooththresh</code>	<code>0.0</code>	Specifies the smoothness threshold of an averaged ocean measurement. The average takes place within consecutive non-overlapping 200-iteration windows. Suggested value is <code>1e-4</code> for a reasonably converged signal-average. This test of smoothness is only active when <code>accuracyaware=autostop</code> .
<code>method</code>	<code>standard</code>	Method used to run montecarlo analysis. Standard is the regular montecarlo analysis with varying netlist parameters. VADE is variation aware design with directly varying device parameters. Possible values are <code>standard</code> and <code>vade</code> .

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>ignoremode</code>	<code>default</code>	Controls the type of variation that is applied to ignored devices, when <code>variations=all</code> . By default, the variation seen by ignored devices depends on whether the <code>'dut'</code> or <code>'ignore'</code> parameter is used. If <code>'dut'</code> is used, then all non-DUT devices will get process variation. If the <code>'ignore'</code> parameter is used, then devices in the <code>'ignore'</code> list will use nominal values, i.e. have no variation. When <code>ignoremode=nominalprocess</code> , the behavior is consistent for both <code>'dut'</code> and <code>'ignore'</code> . If <code>ignoremode=nominal</code> , then all devices in the <code>'ignore'</code> list or devices that are not in the <code>'dut'</code> list will use nominal values. If <code>ignoremode=process</code> , then all devices in the <code>'ignore'</code> list or devices that are not in the <code>'dut'</code> list will have process variation.. Possible values are <code>default</code> , <code>nominal</code> , <code>process</code> and <code>both</code> .
<code>stdscale</code>	<code>default</code>	Scale the standard deviation by the specified value. The default value is 1.0.
<code>process_stdscale</code>	<code>default</code>	Scale the standard deviation of process variation parameters by the specified value. The default value is 1.0.
<code>mismatch_stdscal e</code>	<code>default</code>	Scale the standard deviation of mismatch variation parameters by the specified value. The default value is 1.0.
<code>nscale</code>	<code>default</code>	N Scale for both of process and mismatch variation parameters in uniform distribution by the specified value. The default value is 1.0.
<code>process_nscale</code>	<code>default</code>	N Scale for process variation parameters in uniform distribution by the specified value. The default value is 1.0.
<code>mismatch_nscale</code>	<code>default</code>	N Scale for mismatch variation parameters in uniform distribution by the specified value. The default value is 1.0.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>dist</code>	<code>default</code>	Force all MonteCarlo random variation distributions to the specified type. Possible values are default, unif, gauss and gamma.
<code>processdist</code>	<code>none</code>	Set all MonteCarlo process variation distributions to be the specified type. The default type is none. Possible values are none, unif, gauss, gamma, lnorm, sunif and lunif.
<code>mismatchdist</code>	<code>none</code>	Set all MonteCarlo mismatch variation distributions to be the specified type. The default type is none. Possible values are none, unif, gauss, gamma, lnorm, sunif and lunif.
<code>ignore_type</code>	<code>[...]</code>	If set, no variation is applied to specified types. Default is none. Possible values are none and memcell.
<code>iteruseprevic</code>		Controls if MC iterations use previous IC solutions. Possible values are none and nominal.
<code>dumpvop</code>	<code>no</code>	Whether to dump process/mismatch with new order. Possible values are no and yes.
<code>loadvariationfile</code>		Load file in csv format that contains the variation (iteration indices) to be simulated.

### Saving Process Parameters

<code>saveprocessparameters</code>	Whether to save scalar data for statistically varying process parameters that are subject to process variation. Possible values are no and yes.
<code>processscalarfile</code>	Output file that contains process parameter scalar data.
<code>processparamfile</code>	Output file that contains process parameter scalar data labels.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>saveprocessvec</code>	<code>[...]</code>	Array of statistically varying process parameters (which are subject to process variation) to save as scalar data in <code>processscalarfile</code> .
-----------------------------	--------------------	---

#### Saving Mismatch Parameters

<code>savemismatchpara</code>	<code>no</code>	Whether to save scalar data for statistically varying mismatch parameters that are subject to mismatch variation. Possible values are <code>no</code> and <code>yes</code> .
-------------------------------	-----------------	--

<code>mismatchscalarfile</code>		Output file that contains mismatch parameter scalar data.
---------------------------------	--	---

<code>mismatchparamfile</code>		Output file that contains mismatch parameter scalar data labels.
--------------------------------	--	--

<code>dumpdependency</code>	<code>none</code>	Whether to save a dependency map. Possible values are <code>none</code> and <code>mismatch</code> .
-----------------------------	-------------------	---

<code>dependencymapfile</code>		Specify the name of the output file that contains a dependency map, which indicates the pairing of mismatch parameters and subcircuit instances.
--------------------------------	--	--

<code>dependencyscalarfile</code>		Output the random numbers that are used by mismatch parameters to a file.
-----------------------------------	--	---

<code>dependencyparamfile</code>		Output the mapping from mismatch parameters to corresponding subcircuit instances to a file.
----------------------------------	--	--

#### Flags

<code>donominal</code>	<code>yes</code>	Whether to perform nominal run. Possible values are <code>no</code> and <code>yes</code> .
------------------------	------------------	--

<code>addnominalresults</code>	<code>no</code>	Whether to add nominal run results to Monte Carlo run results. Possible values are <code>no</code> and <code>yes</code> .
--------------------------------	-----------------	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

paramdumpmode	no	Whether to fully dump process/mismatch parameters information. Possible values are no and yes.
dumpseed	no	Whether to dump seed parameters information. Possible values are no and yes.
nullmfactorcorrelation	no	Controls the mismatch variation correlation of parallel devices defined by m-factor. If set to yes, devices are assumed to get uncorrelated mismatch variations. If set to no, devices are assumed to get the same mismatch variation. Possible values are no and yes.
appendsd	no	Whether to append scalar data. Possible values are no and yes.
savefamilyplots	no	Whether to save data for family plots. If yes, this could require considerable disk space. Possible values are no and yes.
savedatainseparatedir	no	Whether to save data for an each plot in a separate directory. If yes, this could require considerable disk space. Possible values are no and yes.
evaluationmode	no	If set to yes, dump random numbers assigned to statistical parameters without running enclosed analyses. Possible values are no and yes.
diskstorage	no	If set to yes, mismatch data is stored on disk instead of memory while running a montecarlo analysis. Possible values are no and yes.
wfseparation	no	If set to yes, a separated directory by the name of <code>nom</code> is created for the nominal run and the directory names of individual iterations are simplified by removing the leading analysis names. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

seedscramble	no	If set to yes, a scrambling procedure is applied to the seed value to generate a better random number sequence in the sense of randomness. Possible values are no and yes.
distribute		Distribute a Monte Carlo analysis to reduce simulation time by using more computer cores across multiple computers. Possible values are no, fork, rsh, ssh, lsf, sge, nc and auto.
numprocesses		Specifies the number of jobs in distributed mode.
usesamesequence	no	If set to yes, the random number sequence is maintained for a seed, even if there is a non-empty dut/ignore list. Possible values are no and yes.
rngversion		The version of random number generator. Possible values are v151.

#### Annotation parameters

annotate	sweep	Degree of annotation. Possible values are no, title, sweep and status.
title		Analysis title.

#### Fast Monte Carlo parameters (for all FMC methods)

fmcmethod		The FMC method to use. Possible values are worstsample and yeldest.
fmcbudget		Budget number of actual simulations for FMC algorithm. For fmcmethod=worstsample, default is no such limit; for fmcmethod=yeldest, default is 3000..
fmcconfidence	0.95	Specify the confidence value in the range of (0.5, 1).

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Fast Monte Carlo parameters (only for Worst Sample method)

fmcmode	fast	The FMC mode to control the balance between performance and accuracy. Possible values are fast, conservative and explore.
fmcyield		Specify the FMC yield target in the range of (0.5, 1).
fmcsigma		Specify the FMC yield target by the sigma value.
fmcnumtailsamples	10	Request FMC to return the desired number of tail samples.
fmcinitruns	50	Specify number of runs in FMC initial group.
fmcnumfails	0	Stop a certain spec during FMC process if its number of failed iterations reaches fmcnumfails. Default is 0, where FMC will not check the number of failed iterations.
fmcmaxbufsize	1	Specify FMC maximum buffer size.
fmcscalarfile		Output file that contains FMC scalar data.
fmcdumpcontribution	0	dump device mismatch contribution lists into FMC log file, in descending order. Input a positive number to specify the number of devices to be dumped; input a negative number to dump all devices' contributions; input 0 to turn off this feature.
fmcmoments	no	Instructs FMC to perform computation of moments of the measurement distribution. Possible values are no and yes.

#### Fast Monte Carlo parameters (only for Yield Estimation method)

fmcsigmawidth	0.5	Specify the FMC yield estimation target width of confidence interval (in terms of sigma)..
---------------	-----	--



## Spectre Circuit Simulator Reference

### Analysis Statements

---

Detailed Description and Examples:

sampling=[standard | lhs]

Determines the sampling behavior. This parameter can be set to standard, the default value, or lhs. lhs invokes latin-hypercube (LHS) method, while standard defaults to the existing standard sampling behavior.

numbins=value

Controls the number of subdivisions used in the LHS method.

If numbins is not specified, the number of subdivisions of the sampling space in LHS will be numruns + firstrun - 1. This parameter is active only when sampling is lhs. If numbins is set to a non-zero integer, the number of subdivisions will be assigned to the greater of the two values numbins or numruns + firstrun - 1.

numruns:(default=100)

Specifies the number of Monte Carlo iterations to perform. The simulator performs a loop, running the specified child analyses, and evaluating any expressions numruns times.

runpoints:(no default)

Specifies the iteration indices to be simulated. Two types of settings are accepted: integers, and ranges. For example, runpoints=[10 range(15, 18) 20] is an acceptable setting, which is translated to perform the simulation of 10th, 15th, 16th, 17th, 18th, and 20th iterations.

seed:(no default)

Specifies the seed for the random number generator. By always specifying the same seed, you can reproduce a previous experiment. If you do not specify a seed, each time you run the analysis, you will get different results, that is, a different stream of pseudo-random numbers is generated.

scalarfile="filename"

Allows you to specify an ASCII file in which scalar data (results of expressions that resolve to scalar values) is written. The data from this file can be read and plotted in histograms by ADE. For each iteration of each Monte Carlo child analyses, Spectre (through Artill) writes a line to this ASCII file, which contains scalar data (one scalar expression per column, for example, slewrate or bandwidth). The default name for this file is of the form name.mcddata, where name is the name of the Monte Carlo analysis instance. This file contains only the matrix of

## Spectre Circuit Simulator Reference

### Analysis Statements

---

numeric values. If you are an ADE Monte Carlo user, you will be more familiar with the term `mcddata` file for the scalar file. Additionally, when the ADE Monte Carlo tool is used to generate the Spectre netlist file, Spectre merges the values of the statistically varying process parameters into this file that contains the scalar data (results of expressions). This means that ADE can later read the data and create scatter plots of the statistically varying process parameters against each other, or against the results of the expressions. In this way, you can see correlations between process parameter variations and circuit performance variations. This data merging occurs whenever the `scalarfile` and `processscalarfile` are written in the same directory.

`paramfile="filename"`

Contains the titles, sweep variable values, and the full expression for each of the columns in the `scalarfile`. If you are an ADE Monte Carlo user, you will be more familiar with the term `mcpparam` file for the `paramfile`. This file is created in the `psf` directory by default, unless you specify an alternative path with the file name.

`processscalarfile="filename"`

If `saveprocessparams` is set to yes, the process (batch-to-batch) values of all statistically varying parameters are saved to this scalar data file. You can use `saveprocessvec` to filter out a subset of parameters in which case Spectre will save only the parameters specified in `saveprocessvec` to the `processscalarfile`). The `processscalarfile` is equivalent to `scalarfile`, except that the data in the `scalarfile` contains the values of the scalar expressions, whereas the data in `processscalarfile` contains the corresponding process parameter values. The default name for this file is of the form `instname.process.mcddata`, where `instname` is the name of the Monte Carlo analysis instance. This file is created in the `psf` directory by default, unless you specify an alternative path in the filename. You can load `processscalarfile` and `processparamfile` into the ADE statistical post-processing environment to plot/verify the process parameter distributions. If you later merge the `processparamfile` with the data in the `scalarfile`, you can then plot scalar expressions values against the corresponding process parameters by loading this merged file into the ADE statistical postprocessing environment.

`processparamfile="filename"`

Contains the titles and sweep variable values for each of the columns in `processscalarfile`. These titles are the names of the process parameters.

`processparamfile` is equivalent to the `paramfile`, except that `paramfile` contains the name of the expressions, whereas `processparamfile` contains the names of the process parameters. The default name for this file is of the form `instname.process.mcpparam`, where `instname` is the name of the Monte Carlo analysis instance. This file is created in the `psf` directory by default, unless you specify an alternative location with filename.

`firstrun:(default=1)`

## Spectre Circuit Simulator Reference

### Analysis Statements

---

Specifies the index of the first iteration. If the first iteration is specified as some number `n` greater than one, then the beginning `n-1` iterations are `skipped`, that is, the Monte Carlo analysis behaves as if the first `n-1` iterations were run, but without actually performing the child analyses for these iterations. The subsequent stream of random numbers generated for the remaining iterations will be the same as if the first `n-1` iterations were actually run. By specifying the first iteration number and the same value for seed, you can reproduce a particular run or sequence of runs from a previous experiment (for example, to examine an outlier case in more detail).

`variations={process,mismatch,all}` (defaults to `process`).

Determines whether to apply only process (batch-to-batch) variations, or only mismatch (per-instance) variations, or both. This parameter assumes that you have specified appropriate statistical distributions in the statistics block. You cannot request that mismatch variations be applied unless you have specified mismatch statistics in the statistics block. You cannot request that process variations be applied unless you have specified process statistics in the statistics block.

`saveprocessvec=[rshsp TOX ...]`

If `saveprocessparams` is set to `yes`, this parameter saves the process (batch-to-batch) values of only those parameters that are listed in `saveprocessvec` to the `processparamfile`. This acts as a filter so that you do not save all process parameters to the file. If you do not want to filter the list of process parameters, do not specify this parameter.

`donominal={yes,no}`(defaults to `yes`).

Controls whether Spectre should perform a nominal run before starting the main Monte Carlo loop of iterations. If any errors are encountered during the nominal run (for example, convergence problems, incorrect expressions, and so on) then Spectre issues an appropriate error message and immediately abandons the Monte Carlo analysis.

If set to `no`, Spectre runs only the Monte Carlo iteration, and does not perform nominal analysis. If any errors are encountered during the Monte Carlo iterations, Spectre issues a warning and continues with the next iteration of the Monte Carlo loop.

`addnominalresults={yes,no}` (defaults to `no`).

Controls whether Spectre should append a nominal run results after the Monte Carlo run results in the data files.

`paramdumpmode={yes,no}`(defaults to `no`).

Controls whether Spectre should dump out each process/mismatch parameter distribution type, mean value, s.t.d. value, and correlation information into additional parameter files. The

## Spectre Circuit Simulator Reference

### Analysis Statements

---

names for these files are `instname.process_full.param`, `instname.mismatch_full.param`, `instname.process.correlate.param` and `instname.mismatch.correlate.param`, where `instname` is the name of the Monte Carlo analysis instance.

`dumpseed={yes,no}` (defaults to `no`).

Controls whether Spectre should dump out seed parameter and iteration number into `instname.seed` file, where `instname` is the name of the Monte Carlo analysis instance.

`nullmfactorcorrelation={yes,no}` (defaults to `no`).

Controls whether Spectre should emulate a 0% correlation for mismatch devices with `m-factor`.

`appendsd={yes,no}` (defaults to `no`).

Specifies whether to append scalar data to an existing scalarfile, or to overwrite the existing scalarfile. This flag applies to both the scalar file and the `processscalarfile`.

`savefamilyplots={yes,no}`.

If set to `yes`, a data file (for example, `psf`) is saved for each analysis for each Monte Carlo iteration, in addition to the expressions scalar results that are saved to the ASCII scalar data file at the end of each iteration. Saving the full data files between runs enables the cloud plotting feature (overlaid waveforms) in ADE. It also enables you to define/evaluate new calculator measurements after the simulation has been run using the ViVA XL calculator. This feature could result in a huge amount of data being stored to disk, and it is advised that you use this feature with care. If you do decide to use this feature, it is advisable to keep the saved data to a minimum. If this parameter is set to `no`, data files are overwritten by each Monte Carlo iteration.

`savedatainseparatedir={yes,no}`.

If set to `yes`, a data file (for example, `psf`) is saved for each analysis for each Monte Carlo iteration in a separate directory, in addition to the expressions scalar results that are saved to the ASCII scalar data file at the end of each iteration. This feature can result in a huge amount of data being stored to the disk. Therefore, it is recommended that you use this feature with care and keep the saved data to a minimum. If this parameter is set to `no`, data files are overwritten by each Monte Carlo iteration.

`annotate={no,title,sweep,status}`

Specifies the degree of annotation. Use the maximum value of `status` to print a summary of the runs that did not converge, had problems evaluating expressions, and so on.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Hints to Improve Performance:

To get better run performance on small-sized designs, it is recommended to use the following settings:

- 1) Set the global option "narrate=compact" to reduce the annotation messages
- 2) Reduce the number of saved signals, or use the "save=nooutput" option
- 3) Use just the measure statement to measure the simulation results

#### Examples

```
// do a Monte Carlo analysis, with process variations only
// useful for looking at absolute performance spreads
mc1 montecarlo variations=process seed=1234 numruns=200 {
    dcop1 dc          // a child analysis
    tran1 tran start=0 stop=1u    // another child analysis
    // expression calculations are sent to the scalardata file
    export slewrate=oceanEval("slewRate(v("vout"),10n,t,30n,t,10,90 )")
}
// do a Monte Carlo analysis, with mismatch variations only
// useful for detecting spreads in differential circuit
// applications, etc. Do not perform a nominal run.
mc2 montecarlo donominal=no variations=mismatch seed=1234 numruns=200 {
    dcop2 dc
    tran2 tran start=0 stop=1u
    export slewrate=oceanEval("slewRate(v("vout"),10n,t,30n,t,10,90 )")
}
// do both together...
mc3 montecarlo saveprocessparams=yes variations=all numruns=200 {
    dcop3 dc
    tran3 tran start=0 stop=1u
    export slewrate=oceanEval("slewRate(v("vout"),10n,t,30n,t,10,90 )")
}
```

#### Specifying Parameter Distributions Using Statistics Blocks

The `statistics` blocks are used to specify the input statistical variations for a Monte Carlo analysis. A statistics block may contain one or more `process` blocks (which represent batch-to-batch type variations) and/or one or more `mismatch` blocks (which represents on-

## Spectre Circuit Simulator Reference

### Analysis Statements

---

chip or device mismatch variations), in which the distributions for parameters are specified. Statistics blocks may also contain one or more correlation statements to specify the correlations between specified process parameters, and/or to specify correlated device instances (for example matched pairs). Statistics blocks may also contain a `truncate` statement that may be used for generating truncated distributions. The distributions specified in the process block are sampled once per Monte Carlo iteration and are typically used to represent batch-to-batch or process variations, whereas the distributions specified in the mismatch block are sampled on a per subcircuit instance basis and are typically used to represent device-to-device mismatch for devices on the same chip. In the case where the same parameter is subject to both process and mismatch variations, the sampled process value becomes the mean for the mismatch random number generator for that particular parameter.

**Note:** Note: Multiple statistics blocks can exist, and in that case the blocks either accumulate or overlay. Typically, process variations, mismatch variations, and correlations between process parameters are specified in one statistics block. A second statistics block should be specified where actual device instance correlations are specified (that is, specification of matched pairs).

Statistics blocks can be specified using combinations of the Spectre keywords `statistics`, `process`, `mismatch`, `vary`, `truncate`, and `correlate`. Braces `{}` are used to delimit blocks.

The following example shows statistics blocks, which are discussed below along with syntax requirements.

```
// define some netlist parameters to represent process parameters

// such as sheet resistance and mismatch factors

parameters rshsp=200 rshpi=5k rshpi_std=0.4K xisn=1 xisp=1 xxx=20000 uuu=200

// define statistical variations, to be used

// with a MonteCarlo analysis.

statistics {

    process { // process: generate random number once per MC run

        vary rshsp dist=gauss std=12 percent=yes

        vary rshpi dist=gauss std=rshpi_std // rshpi_std is a parameter

        vary xxx dist=lnorm std=12

        vary uuu dist=unif N=10 percent=yes
```

## Spectre Circuit Simulator Reference

### Analysis Statements

---

```
truncate tr=2.0 // +/- 2 sigma
...
}
mismatch { // mismatch: generate a random number per instance
    vary rshsp dist=gauss std=2
    vary xisn dist=gauss std=0.5
    vary xisp dist=gauss std=0.5
    truncate tr=7.0 // +/- 7 sigma
}
// some process parameters are correlated
correlate param=[rshsp rshpi] cc=0.6
// specify a global distribution truncation factor
truncate tr=6.0 // +/- 6 sigma
}
// a separate statistics block to specify correlated (i.e. matched) components
// where m1 and m2 are subckt instances.
statistics {
    correlate dev=[m1 m2] param=[xisn xisp] cc=0.8
}
// a separate statistics block to specify correlation with wildcard, where
// I*.M3 matches multiple subckt instances, for examples, I1.M3, I2.M3, I3.M3, etc..
// Only the asterisk (*) is recognized as a valid wildcard symbol.
statistics {
    correlate dev=[I*.M3] param=[misx mixy] cc=0.8
```

## Spectre Circuit Simulator Reference

### Analysis Statements

---

}

#### Specifying Distributions

Parameter variations are specified using the following syntax:

```
vary PAR_NAME dist=<type> {std=<value> | N=<value>} {percent=yes|no}
```

The following types of parameter distributions are available: gaussian, lognormal, uniform, set-uniform, and gamma, corresponding to the keywords `gauss`, `lnorm`, `unif`, `'sunif'`, and `'gamma'` respectively. For the `gauss`, the `lnorm` and the `'gamma'` distributions, you specify a standard deviation using the `std` keyword.

#### Gaussian Distribution

For the gaussian distribution, the mean value is taken as the current value of the parameter being varied, giving a distribution denoted by Normal (mean,std). Using the example above, parameter `rshpi` is varied with a distribution of Normal (5k,0.4k)

#### Lognormal Distribution

The lognormal distribution is denoted by:

$$\log(x) = \text{Normal}(\log(\text{mean}), \text{std})$$

where,  $x$  is the parameter being specified as having a lognormal distribution.

**Note:** Note: `log()` is the natural logarithm function. For parameter `xxx` in the example, the process variation is according to:

$$\log(\text{xxx}) = \text{Normal}(\log(20000), 12)$$

#### Uniform Distribution

The uniform distribution for parameter  $x$  is generated according to:

$$x = \text{unif}(\text{mean}-N, \text{mean}+N)$$

The mean value is the nominal value of the parameter  $x$ , and the parameter is varied about the mean with a range of  $\pm N$ . The standard deviation is not specified for the uniform distribution, but its value can be calculated by using the formula:  $\text{std}=N/\sqrt{3}$ .

#### Set-uniform Distribution



## Spectre Circuit Simulator Reference

### Analysis Statements

---

Set-uniform, or discrete uniform, distribution is specified by a finite set of values `values=[val1 val2 ... valN]` that are equally likely to be observed. Values can be provided one by one, or using `range(<min>, <max>, <step>)`, or using expression `<min>:<step>:<max>`.

#### Gamma Distribution

Similar to Gauss distribution, the mean value for gamma distribution is the current value of the parameter being varied, giving a distribution denoted by `Gamma(mean,std)`. The corresponding shape parameter 'alpha' and scale parameter 'beta' are:

$$\text{alpha} = (\text{mean} / \text{std})^2$$

$$\text{beta} = (\text{std})^2 / \text{mean}.$$

Notice: truncation factor is not available for gamma distribution, and it will always be complete gamma distribution for any set value of truncation factor.

#### Values as Percentages

The `percent` flag indicates whether the standard deviation `std` or uniform range `N` are specified in absolute terms (`percent=no`) or as a percentage of the mean value (`percent=yes`). For parameter `uuu` in the example above, the mean value is 200, and the variation is  $200 \pm 10\% \times (200)$  i.e.  $200 \pm 20$ . For parameter `rshsp`, the process variation is given by Normal (200,  $12\% \times (200)$ ), that is, Normal (200, 24). It is recommended that you do not use `percent=yes` with the lognormal distribution.

#### Process and Mismatch Variations

The statistics specified in a process block are applied at global scope, and the distributions are sampled once per Monte Carlo iteration. The statistics specified in a mismatch block are applied on a per-subcircuit instance basis, and are sampled once per subcircuit instance. If you place model cards and/or device instances in subcircuits, and add a mismatch block to your statistics block you can effectively model device-to-device mismatch for these devices/models.

#### Correlation Statements

There are two types of correlation statements that you can use: process parameter correlation statements and instance correlation statements.

##### Process Parameter Correlation

The syntax of the process parameter correlation statement is:

`correlate param=[list of parameters] cc=<value>`

## Spectre Circuit Simulator Reference

### Analysis Statements

---

This allows you to specify a correlation coefficient between multiple process parameters. You can specify multiple process parameter correlation statements in a statistics block to build a matrix of process parameter correlations. During a Monte Carlo analysis, process parameter values are randomly generated according to the specified distributions and correlations.

#### Mismatch Correlation (Matched Devices)

The syntax of the instance or mismatch correlation statements are:

```
correlate dev=[list of subcircuit instances] {param=[list of parameters]} cc=<value>
```

```
correlate dev=[<wildcard expr>] {param=[list of parameters]} cc=<value>
```

where, the device or subcircuit instances to be matched are listed in the list of subcircuit instances or regular expressions with asterisk (\*), and the list of parameters specifies exactly which parameters with mismatch variations are to be correlated.

The instance mismatch correlation statement is used to specify correlations for particular subcircuit instances. If a subcircuit contains a device, you can effectively use the instance correlation statements to specify that certain devices are correlated (that is, matched) and give the correlation coefficient. You can optionally specify exactly which parameters are to be correlated by giving a list of parameters (each of which must have had distributions specified for it in a mismatch block), or specify no parameter list, in which case all parameters with mismatch statistics specified are correlated with the given correlation coefficient. The correlation coefficients are specified in the <value> field and must be between +/- 1.0.

**Note:** Note: Correlation coefficients can be constants or expressions, as can `std` and `N` when specifying distributions.

#### Truncation Factor:

The default truncation factor for gaussian distributions (and for the gaussian distribution underlying the lognormal distribution) is 4.0 sigma. Randomly generated values that are outside the range of mean +/- 4.0 sigma are automatically rejected and regenerated until they fall inside the range. You can change the truncation factor using the `truncate` statement. The syntax is:

```
truncate tr=<value>
```

The following conditions should be considered while setting the truncate factor: The value of the truncation factor can be a constant or an expression.

Parameter correlations can be affected by using small truncation factors.

## **Spectre Circuit Simulator Reference**

### **Analysis Statements**

---

There are different truncate for process and mismatch blocks. If a truncate for process or mismatch block is not given, it will be set to the value of truncate in statistic block.

Notice: truncation factor is not available for gamma distribution, and it will always be complete gamma distribution for any set value of truncation factor.

## Noise Analysis (noise)

### Description

Noise analysis linearizes the circuit about the operating point and computes the noise spectral density at the output. If you identify an input source, the transfer function and the input-referred noise for an equivalent noise-free network are computed. If the input source is noisy, the noise figure is also computed.

The noise is computed at the output of the circuit. The output is specified with either a pair of nodes or a probe component. To specify the output of a circuit with a probe, specify it with 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 the input-referred noise is desired, specify the input source by using the `iprobe` parameter. Currently, only a `vsource`, an `isource`, or a `port` may be used as an input probe. If the input source is noisy, as is a `port`, the noise analysis computes 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 its `noisetemp` must be set to 16.85C (290K). 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, 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.

The noise analysis always computes the total noise at the output, which 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 is also computed and output individually. If the input source is identified, the input-referred noise is computed, which includes the noise from the input source itself. Finally, if the input source is identified and is noisy, the noise factor and noise figure are computed. Therefore, if:

No = total output noise

Ns = noise at the output due to the input probe (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

## Spectre Circuit Simulator Reference

### Analysis Statements

---

Then:

$$IRN = \sqrt{No^2 / G^2}$$

$$F = (No^2 - NI^2) / Ns^2$$

$$NF = 10 * \log_{10}(F)$$

When the results are output, No is named `out`, IRN is named `in`, G is named `gain`, F is named `F`, and NF is named `NF`.

Spectre can perform Noise analysis while sweeping a parameter. The parameter can be frequency, temperature, component instance parameter, component model parameter, or netlist parameter. If changing a parameter affects the DC operating point, the operating point is recomputed at each step. You can sweep the circuit temperature by giving the parameter name as `temp`, without a `dev` or `mod` parameter. In addition, you can sweep a netlist parameter by giving the parameter name without a `dev` or `mod` parameter. After the analysis is complete, the modified parameter returns to its original value.

### Syntax

```
Name [p] [n] noise parameter=value ...
```

The optional terminals (p and n) specify the output of the circuit. If you do not specify the terminals, you must specify the output with a probe component.

### Parameters

<code>prevoppoint</code>	<code>no</code>	Use the operating point computed in the previous analysis. Possible values are <code>no</code> and <code>yes</code> .
--------------------------	-----------------	---

#### Sweep interval parameters

<code>start</code>	<code>0</code>	Start sweep limit.
<code>stop</code>		Stop sweep limit.
<code>center</code>		Center of sweep.
<code>span</code>	<code>0</code>	Sweep limit span.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

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.
valuesfile		Name of the file containing the sweep values.

#### Sweep variable parameters

dev		Device instance whose parameter value is to be swept.
mod		Model whose parameter value is to be swept.
param		Name of parameter to sweep.
freq	(Hz)	Frequency when a parameter other than frequency is being swept.

#### Probe parameters

oprobe		Compute total noise at the output defined by this component.
iprobe		Input probe. Refer the output noise to this component.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### State-file parameters

readns		File that contains an estimate of the DC solution (nodeset).
write		DC operating point output file at the first step of the sweep.
writefinal		DC operating point output file at the last step of the sweep.

#### Initial condition parameters

force	none	The set of initial conditions to use. Possible values are none, node, dev and all.
readforce		File that contains initial conditions.
skipdc	no	Skip DC analysis. Possible values are no and yes.
useprevic	no	If set to yes or ns, use the converged initial condition from previous analysis as ic or ns. Possible values are no, yes and ns.

#### Output parameters

save		Signals to output. Possible values are all, lvl, allpub, lvlpub and selected.
nestlvl		Levels of subcircuits to output.
oppoint	no	Determines whether operating point information should be computed:if yes, where should it be printed (screen or file). Operating point information is not printed if the operating point computed in the previous analysis remains unchanged. Possible values are no, screen, logfile and rawfile.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>separatenoise</code>	<code>no</code>	Separate noise into sources and transfer functions. Possible values are no and yes.
----------------------------	-----------------	---

#### Convergence parameters

<code>restart</code>	<code>yes</code>	Restart the DC solution from scratch if any condition has changed. If not, use the previous solution as an initial guess. Possible values are no and yes.
----------------------	------------------	---

#### Annotation parameters

<code>annotate</code>	<code>sweep</code>	Degree of annotation. Possible values are no, title, sweep, status and steps.
-----------------------	--------------------	---

<code>title</code>	Analysis title.
--------------------	-----------------

You can define sweep limits by specifying the end points or the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. You can specify a step size parameter (`step`, `lin`, `log` or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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. All frequencies are in Hertz.

The small-signal analysis begins by linearizing the circuit about an operating point. By default, this analysis computes the operating point, if it is not known, or recomputes it if any significant component or circuit parameter has changed. However, if an operating point was computed during a previous analysis, you can set `prevoppoint=yes` to avoid recomputing it. For example, if `prevoppoint=yes` and the previous analysis was a transient analysis, the operating point is the state of the circuit at the final time point.

Nodesets help find the DC or the initial transient solution. You can specify nodesets in the circuit description file with `nodeset` statements or in a separate file by using the `readns` parameter. When nodesets are specified, Spectre computes an initial guess of the solution by performing DC analysis, while forcing the specified values on to nodes by using a voltage source in series with a resistor whose resistance is `rforce`. Spectre then removes these voltage sources and resistors and computes the required solution from this initial guess.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

Nodesets have two important uses. First, if a circuit has two or more solutions, nodesets can bias the simulator towards computing the required solution. Second, this is a convergence aid. By estimating the solution of the largest possible number of nodes, you might be able to eliminate a convergence problem or significantly speed up convergence.

When you simulate the same circuit multiple times, it is recommended that you use both `write` and `readns` parameters and assign the same file name to both parameters. DC analysis then converges quickly even if the circuit has changed since the last simulation, and the nodeset file is automatically updated.

During the initial operating point DC analysis, you may force certain circuit variables to use the values given in the `ic` file, `ic` statements, or `ic` parameter on the capacitors and inductors. The `ic` parameter controls the interaction of various methods of setting the force values. The effects of individual settings are as follows:

`force=none`: All initial conditions are ignored.

`force=node`: The `ic` statements are used, and the `ic` parameters on the capacitors and inductors are ignored.

`force=dev`: The `ic` parameters on the capacitors and inductors are used, and the `ic` statements are ignored.

`force=all`: Both `ic` statements and `ic` parameters are used, with the `ic` parameters overriding the `ic` statements.

If you specify an `ic` file with the `readforce` parameter, force values from the file are used and any `ic` statements are ignored.

After you specify the initial conditions, Spectre computes the DC operating point with the specified nodes forced to the given value by using a voltage source in series with a resistor whose resistance is `rforce` (see `options`).

Optimize Analysis (optimize)

Description

The `optimize` analysis automatically generates model parameters and component values from a set of electrical specifications or measured data.

Syntax

Name `optimize parameter=value ...`

Parameters

Optimize Parameters

<code>title</code>		Analysis title.
<code>annotate</code>	<code>title</code>	Degree of annotation. Possible values are no, title, sweep and status.

Passfail parameters

Example

Example:

```
result=max_i model=optimizemod .tran 1n 3n sweep optimize=optimize1
```

## Immediate Set Options (options)

### Description

The immediate set options statement sets or changes various program control options. These options take effect immediately and are set while the circuit is read. For more options, see the description of individual analyses.

**Note:** Note: Options that are dependent on netlist parameter values do not maintain their dependencies on those netlist parameters.

In many cases, a particular option can be controlled by either a command-line or netlist specification. In the situation where both are used, the command-line option takes priority over the setting in the netlist options statement.

### Syntax

```
Name options parameter=value ...
```

### Parameters

#### Tolerance parameters

reltol	0.001	Relative convergence criterion.
residualtol	1.0	Tolerance ratio for residual (multiplies reltol).
vabstol	1.0e-6 V	Convergence criterion for absolute voltage tolerance.
iabstol	1.0e-12 A	Convergence criterion for absolute current tolerance.
chargeabstol	1.0e-21 Coul	Charge absolute tolerance convergence criterion.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Temperature parameters

temp	27 C	Temperature.
tnom	27 C	Temperature measurement of the default component parameter.
tempeffects	all	Temperature effect selector. If tempeffect = vt, only thermal voltage varies with temperature; if tempeffect = tc, parameters that start with tc are active and thermal voltage is dependent on temperature; and if tempeffect = all, all built-in temperature models are enabled. Possible values are vt, tc and all.

#### Output parameters

save	selected	Defines node voltage waveforms to be saved. selected ... voltage waveforms for nodes saved by the user are created, if no node voltage is saved, then allpub is used. allpub ... all node voltages, voltage source currents, inductor currents, and iprobe currents are saved. lvlpub ... all node voltages, voltage source currents, inductor currents, and iprobe currents down nestlvl levels are saved. none ... no node voltage is saved except one randomly selected node voltage. nooutput ... disables any waveform writing. all ... same as allpub, additionally saves internal nodes of active devices. lvl ... same as lvlpub, additionally saves internal nodes of active devices. Possible values are all, lvl, allpub, lvlpub, selected, none and nooutput.
------	----------	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>saveselectedtoallpub</code>	<code>default</code>	Controls when <code>save=selected</code> is changed to <code>save=allpub</code> . <code>default ... save=selected</code> will automatically switch to <code>save=allpub</code> if no node voltages or currents through voltage sources/ inductors are saved. <code>nooutput ... save=selected</code> will automatically switch to <code>save=allpub</code> if no waveforms saved for any output, including all current probes, device operating point parameters etc.. Possible values are <code>default</code> and <code>nooutput</code> .
<code>nestlvl</code>	<code>infinity</code>	Defines hierarchical level when <code>save=lvl</code> or <code>lvlpub</code> is used. <code>nestlvl=1</code> refers to top level, <code>nestlvl=2</code> includes top level and one level below, ....
<code>currents</code>	<code>selected</code>	Defines element, device, and inline subckt current waveforms to be saved. <code>selected ...</code> waveforms for element, device, and inline subckt currents saved by the user are created. <code>all ...</code> save all element, device and inline subckt currents on all hierarchy levels. <code>nonlinear ...</code> save currents for all nonlinear elements on all hierarchy levels. <code>none ...</code> no currents are saved, but individually saved currents are not disabled. Possible values are <code>all</code> , <code>nonlinear</code> , <code>selected</code> and <code>none</code> .
<code>subcktprobelvl</code>	<code>0</code>	Defines level up to which subckt terminal current waveforms are automatically saved. <code>0 ...</code> no subckt currents are saved. <code>1 ...</code> current through top level subckt instance terminals are saved. <code>2 ...</code> current through terminals of subckt instances in the top two levels of hierarchy are saved. ....
<code>pwr</code>	<code>none</code>	Defines element, subckt, and total design power waveforms (extension <code>:pwr</code> , unit W) to be saved. <code>all ...</code> all element, subckt, and total power. <code>subckt ...</code> all subckt and total power. <code>devices ...</code> all element and total power. <code>total ...</code> total power. <code>none ...</code> no power. Possible values are <code>all</code> , <code>subckts</code> , <code>devices</code> , <code>total</code> and <code>none</code> .

## Spectre Circuit Simulator Reference

### Analysis Statements

---

subcktiprobes	yes	Defines whether subckt terminal currents saved are calculated by summing up branch currents, or by inserting iprobe elements. all ... uses iprobes for both, individual subckt port current save and subcktprobevl statements (Spectre default). save ... only inserts iprobes for individual subckt port current save statements but not for subcktprobevl statements (APS default). no ... doesn't insert any iprobe. yes ... maintains backward compatibility by using all for Spectre, SpectreX and save for APS. auto ... smart iprobe insertion, subckt ports with large numbers of branches use iprobe elements, subckt ports with small number of branches use branch current sum (SpectreX enforced behavior, use -preset_override to honor other settings but be aware of potential performance degradation). Possible values are no, save, all, yes and auto.
useprobes	no	Defines whether element current saved is taken from element or from inserted iprobe element. yes ... inserts iprobes for individual element current save, and currents statement. no ... use element current, no iprobes inserted. Possible values are no and yes.
useterms	default	Defines whether device/subckt terminal index (M1:1) or name (M1:d) is saved, when using currents or subcktprobevl options. Doesn't apply to individual save statements, i.e. save M1:d. default ... save device terminal currents with terminal name and subckt terminal currents with terminal index. index ... save both with terminal index. name ... save both with terminal name. Possible values are default, name and index.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

usenodes	name	Defines how identify last part of hier name with digital in save statement, as device/subckt terminal index first or as terminal/node name first. i.e. For save mn.4 name ... identify 4 as node/terminal name first, if no node/terminal named 4, use it as index. index ... Identify 4 as terminal index first, if no fourth terminals, identify 4 as name.. Possible values are name and index.
savefilter	none	Option defines whether global or wildcard save statements create waveforms for all nodes (none), or exclude nodes being only RC connected (rc). Option only impacts waveform creation but not reduction or node preservation inside Spectre. Possible values are none, rc and colon.
saveports	no	If set to yes, subckt ports will be saved when nodes inside subckts are saved with wildcards, i.e. "save * subckt=abc". Possible values are no and yes.
save_time_window	0	Specifies the time window for all saved waveforms. Multiple time windows are supported: save_time_window=[1u 2u 5u 6u]. If a waveform is saved with an individual time_window argument, then the save_time_window option is ignored for this signal.
wfdebug	none	Flag to store the measurement signal in a raw file. Possible values are none, measure and assert.
mcwf	yes	Flag to generate waveform in Monte Carlo simulation. Possible values are no and yes.
saveahdlvars	selected	AHDL variables to output. Possible values are all, selected and allwithnodes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

spfpubenode	net_only	When net_only, it is the only net name in SPF file that is used to match the wild card in .probe/.ic statements. The hierarchical structure of the net name is restored like pre-layout netlist. When all, the net name in SPF file is stored as pre-layout node name, and the nodes in SPF file are preserved like the current behavior. The wild card in .probe/.ic will match all the nodes and net names. Possible values are net_only and all.
dspf_nodename	original	When original, The current behavior, .connect is created for nodes cut by DSPF translator. When net_only, it is the only net name in DSPF file that is used to match the wild card in .probe/.ic statements. The hierarchical structure of the net name is restored like pre-layout netlist. When all, the net name in DSPF file is stored as pre-layout node name, and the nodes in DSPF file are preserved like the current behavior. The wild card in .probe/.ic will match all the nodes and net names. Possible values are original, net_only and all.
redundant_currents	no	Defines whether one or both currents through two terminal devices are saved when 'currents=all' or 'save *:currents' are used. The option doesn't apply to individually saved terminal currents like 'save R0:1 R0:2'. no...only one terminal current is saved, yes...both terminal currents are saved. Possible values are no and yes.
sine_sync_period	no	If set to yes, sine-type source proposes time points exactly at 0, 30, 60, 90, 120, ... degree.. Possible values are no and yes.
pwl_ignore_error_msg	no	If set to yes, will ignore pwl source error message if the time of two points are same.. Possible values are no and yes.
disable_check_nf	no	If set to yes, disable node folding checking.. Possible values are no and yes.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

disable_selfheating	no	If set to yes, disable self-heating flow in the bsimsoi, hisim_hv, bht, bht0, vbic, ucsd_hbt, hbt, bsimbulk, psp103, hisimsoi, r3, hvmos, bsimcmg, bsimimg, jfet100, utsoi2, asm_gan, mvsg_gan and lutsoi models.. Possible values are no and yes.
wb_need_propose_time	no	If set to yes, enable wave buffer propose time.. Possible values are no and yes.
strict_td	yes	strict delay or not.. Possible values are no and yes.
wb_reltol	0.001 no	signal compress reltol in wavebuffer..
wb_vabstol	1e-6 no	signal compress vabstol in wavebuffer..
wb_iabstol	1e-12 no	signal compress iabstol in wavebuffer..
b4_tunneling_current_max_exp	34 no	max exp value for b4 tunneling current..
vcr_need_reject_time	no	If set to yes, enable vcr reject time.. Possible values are no and yes.
skip_bjt_limit	no	If set to yes, skip bjt limit.. Possible values are no and yes.
clamp_bjt_cap	no	if set to yes, clamp bjt negative cap.. Possible values are no and yes.
skip_bsim4_limit	no	If set to yes, skip bsim4 limit.. Possible values are no and yes.
src_allbrkpts	no	If set to yes, set all user-defined points for a pwl-type source as breakpoints in transient analysis. When both the global option and the 'allbrkpts' option of individual sources are set, the latter take precedence.. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>ahdlomainerror</code>	<code>warning</code>	AHDL domain error handling selector. If <code>ahdlomainerror=error</code> , incorrect AHDL domain input is treated as an error; If <code>ahdlomainerror=warning</code> , incorrect AHDL domain input is treated as warning; If <code>ahdlomainerror=none</code> , AHDL domain error is ignored. Possible values are error, warning, none, erroriter and warniter.
<code>rawfmt</code>	<code>psfxl</code>	Output raw data file format. Possible values are nutbin, nutascii, psfbin, psfascii, psfbinf, psfxl, dsfmwo, sst2, fsdb, fsdb5, wdf, uwi, tr0ascii and vwdb.
<code>rawfile</code>	<code>%C:r.raw</code>	Output raw data file name.
<code>precision</code>	<code>%g</code>	Format specification of double for psfascii. Example: "%15.12g" outputs 12 decimal digits in mantissa. The default value "%g" prints 6 decimal digits.
<code>psfversion</code>		Version of psfxl format to use. Default is "1.1"..
<code>fsdbversion</code>		Version of fsdb format to use. Default is "6.3" Valid versions are 5.7, 5.8, 6.1, or 6.3.
<code>uwifmt</code>		User-defined output format. To specify multiple formats, use <code>:</code> as a delimiter. The option is valid only when waveform format is defined as uwi.
<code>uwilib</code>		Absolute path to the user-defined output format library. This option is used along with <code>uwifmt</code> . Use <code>:</code> as a delimiter to specify more than one library.
<code>exportfile</code>		oceanEval output file name.
<code>wfmaxsize</code>	<code>infinity</code>	Limits the maximum size of the output waveform. This option is valid when uwi format is set.
<code>disk_check_thres h</code>	<code>1E8</code>	The threshold of disk space check.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

disk_check_autor esume	no	Automatically resume the simulation once the available disk space is larger than the specified threshold. Possible values are no and yes.
disk_check_user_ quota_cmd		A command to query the disk quota for a user. While checking the disk space available for a run, Spectre considers both the available disk space and the user disk quota returned by the command specified by this option. This command result should be in YAML format containing keyword 'available: <available user disk quota>' to indicate available user disk quota. The supported disk storage unit are G, g, Gb and GB. If no specified disk storage unit is used, the default disk storage unit will be byte..
gennetlistdir	no	Output netlist directory in logfile. Possible values are no and yes.
procopt	no	Fix affinity for the simulation run. Possible values are no and yes.

#### Convergence parameters

homotopy	all	Method used when there is no convergence on initial attempt of DC analysis. You can specify methods and their orders by defining vector setting such as homotopy=[source ptran gmin]. Possible values are none, gmin, source, dptran, ptran and all.
newton	normal	Method used on DC analysis. You can specify methods newton=[none normal].. Possible values are none and normal.
noiseon_type	all	The noiseon_type defines the noise sources which are enabled by the noiseon/off_inst option.. Possible values are thermal, flicker, shot, ign and all.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

noiseoff_type	all	The noiseoff_type defines the noise sources which are disabled by the noiseon/off_inst option.. Possible values are thermal, flicker, shot, ign and all.
limit	dev	Limiting algorithms to aid DC convergence. Possible values are delta, log, dev, l2, newton and none.
device_limit	dev	Limiting method for nonlinear device to aid DC convergence. Possible values are dev, delta and dev_delta.
maxdeltav	0.3 V	Maximum change in voltage allowed per Newton iteration when limit=delta (default 0.3 V) or limit=l2 (default 10 V).
automaxdeltavscale	0.1	Scale for maxdeltav calculation when maxdeltav is set automatically, must be larger than 0 and less than 1.
maxvddd	3.0 V	Maximum value of Vdd for DC simulation.
maxvdd	0 V	MaxVdd set from user, used for some optimization check in APS.
maxphysicalv	infinity V	Maximum physical voltage allowed in solution.
maxphysicali	infinity A	Maximum physical current allowed in solution.
maxphysicalmmf	infinity A	Maximum physical magnetomotive force allowed in solution.
maxphysicalflux	infinity Wb	Maximum physical flux allowed in solution.
maxphysicalunitless	infinity	Maximum physical unitless signals allowed in solution.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

ahdl_device_limit	dev	Limiting method for AHDL device to aid DC convergence. Possible values are dev, delta and dev_delta.
ahdl_maxdeltav	0.3 V	Maximum change in voltage allowed for AHDL devices per Newton iteration when ahdl_device_limit=delta (default 0.3 V).
gmethod	dev	Stamp gdev, gnode, or both in the homotopy methods (other than dptran). See the detailed description below the parameter descriptions for more information. Possible values are dev, node and both.
dptran_gmethod	node	Stamp gdev, gnode, or both in the dptran (homotopy) methods. Possible values are dev, node and both.
gmin_start	1.0	Initial gmin in gmin-stepping (homotopy) methods.
gmin_converge	1.0e-13	Maximum DC gmin value allow DC to converge.
try_fast_op	yes	Speed up the DC solution. For hard-to-converge designs, this feature fails and other methods are applied. In corner cases, this feature may have negative effects. If the DC analysis is unusually slow, the memory usage of the processes keeps increasing, or if DC analysis gets stuck even before homotopy methods start, set this option to no. Possible values are no and yes.
icversion	1	Convert initial condition to initial guess, when .ic statements exist in the netlist and there are no other options to set IC or nodeset.
dcmaxiters	5	Maximum number of Newton iterations in DC simulation.
dcminiters	2	Minimum number of Newton iterations in DC simulation.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>stackfet_dcmaxiters</code>	100	Maximum number of Newton iterations in DC simulation for StackFET.
<code>dc_diagnose_time</code>	1800	Time interval for printing out diagnose information in DC simulation.
<code>dc_diagnose_elements</code>	10	Number of elements connected to non-converge nodes in printing diagnose information in DC simulation.
<code>ptranmaxsteps</code>	200	Maximum number of steps for ptran phase in DC simulation.
<code>dptranmaxsteps</code>	300	Maximum number of steps for dptran in DC simulation.
<code>ptranmaxsteps1</code>	500	Maximum number of steps for ptran phase in 1st-stage DC simulation.
<code>dptranmaxsteps1</code>	500	Maximum number of steps for dptran phase in 1st-stage DC simulation.
<code>dcoptmaxsteps</code>	300	Maximum number of steps for dcopt in DC simulation.
<code>dcoptautoptransteps</code>	100	Initial number of steps for ptran phase in DC simulation with <code>dcoptmaxsteps=auto</code> .
<code>dcoptautodptrans steps</code>	100	Initial number of steps for dptran phase in DC simulation with <code>dcoptmaxsteps=auto</code> .
<code>dcoptic</code>	<code>aps1</code>	Initial condition for dcopt in DC simulation. Possible values are <code>no</code> , <code>aps1</code> , <code>aps2</code> and <code>xps</code> .
<code>dcoptictype</code>	<code>nodeset</code>	Initial condition type for dcopt in DC simulation. Possible values are <code>nodeset</code> and <code>ic</code> .
<code>reusedc</code>	<code>no</code>	Reuse dcopt results from second analysis. Possible values are <code>no</code> , <code>dcopt</code> and <code>previc</code> .

## Spectre Circuit Simulator Reference

### Analysis Statements

---

icpriority	netlist	Set the ic priority order. If set to netlist, the order from lowest to highest is readNS, netlist NS, readIC, netlist IC. If set to file, the order from lowest to highest is netlist NS, readNS, netlist IC readIC. Possible values are file and netlist.
subcktoppoint	no	Generate operating point parameters for sub-circuit instances. Possible values are no and yes.
out_dev_stack		Dump expanded stacked device macro netlisting a separate file, Dump complete DC OP output in a separate file.
fastmcmethod		Enable sss standalone running.
detect_path_vpn	[...]	Define vpn nodes for detecting path.
detect_path_vgnd	[...]	Define vgnd nodes for detecting path.
detect_path_clock	[...]	Define clock for detecting path.
detect_path_config		Define extra options for detecting path in tcl file.
subcktcap	0	Select the sub-circuit capacitance output mode. If this is set to 0 then the sum of the four capacitances is 0, e.g. cgg + cgs + cgd + cgb = 0; If set to 1, the diagonal capacitance is equal to the sum of off-diagonal capacitances, e.g. cgg = cgs + cgd + cgb.
subcktopfreq	0.159155 Hz	This parameter is used to set the frequency used in AC analyses run as part of sub-circuit operating point parameter calculations. The default value is 1/(2*pi) Hz.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Multithreading parameters

<code>multithread</code>	<code>off</code>	Enable or disable multithreading capability. When multithreading is enabled but the number of threads (nThreads) is not specified, Spectre will automatically detect the number of processors and select the proper number of threads to use. (See important note about using multithreading in the detailed description below the parameter descriptions). Possible values are off and on.
<code>forcemt</code>	<code>no</code>	Force the multithread behaviour, no is to choose number of threads automatically according to circuit size, yes is to use the number of threads specified by user strictly, exit is when number of threads specified more than needed according to circuit size. Default is no.. Possible values are no, yes and exit.
<code>nthreads</code>	<code>1</code>	Specifies the number of threads for multithreading.

#### Component parameters

<code>rabsshort</code>	<code>0 Ω</code>	When this option is set, all fixed value resistors with absolute value of $R \leq \text{rabsshort}$ are shorted. Default value is 0 for Spectre, and 1m for APS. Rabsshort can additionally be applied to variable resistors using the option 'short_cut_var_elem=yes'.
<code>rabsclamp</code>	<code>0.0 Ω</code>	All fixed value and variable resistors with absolute value of $R \leq \text{rabsclamp}$ are clamped to 'rabsclamp'.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

rcut	infinity $\Omega$	The value of fixed value resistors with $R > \text{rcut}$ is limited to 'rcut' value. Default is infinity for Spectre and $\text{MAX}(1/\text{GMIN}, 1\text{e}12)$ for Spectre APS and SpectreX. For bsource resistor, default is $1\text{e}12$ for Spectre and $\text{MAX}(1/\text{GMIN}, 1\text{e}12)$ for Spectre APS and SpectreX. If GMIN is not specified, the value of $\text{MAX}(1/\text{GMIN}, 1\text{e}12)$ is $1\text{e}12$ . rcut can additionally be applied to variable resistors using the option 'short_cut_var_elem=yes'.
rcut_bsource	$1\text{e}12 \ \Omega$	When rcut_bsource=value is given, all instance bsource resistors with $R > \text{value}$ are clamped to value. It has high priority than rcut if both are set. Default is $1\text{e}12$ Ohm for bsource resistor.
rclamp_bsource	$1 \ \Omega$	When rclamp_bsource=value is given, all instance bsource resistors with $R < \text{value}$ are clamped to value. It has high priority than rclamp if both are set. Default is $0.001$ Ohm for bsource resistor.
rclamp_bsource_m factor	ignore	Defines whether the rclamp_bsource clamping value is scaled with the M-factor (scaled) or not (ignored). Possible values are scaled and ignore.
rthresh	$0.001 \ \Omega$	All instance resistors with $\text{abs}(R) < \text{rthresh}$ will use resistance form, unless their instance parameter or model parameter overwrites it.
cclamp	$0.0 \ \text{F}$	All capacitors including bsource capacitors with $C \leq \text{cclamp}$ are clamped to 'cclamp'.
ccut	– infinity $\text{F}$	Capacitors with $C \leq \text{ccut}$ are replaced by open. ccut can additionally be applied to variable capacitors using the option 'short_cut_var_elem=yes'.
cgnd	– infinity $\text{F}$	Coupling capacitors with $C \leq \text{cgnd}$ are split into 2 grounded capacitors with same C value.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

cmax	infinity F	All capacitors with $C \geq \text{cmax}$ are clamped to 'cmax'.
kcut	0.0	Fixed value mutual inductors with absolute value of $K \leq \text{kcut}$ are replaced by an open. kcut can additionally be applied to variable value mutual inductors using the option 'short_cut_var_elem=yes'. Default value is 0 in Spectre, and 1m in APS.
lshort	0 H	Fixed value inductors with $L \leq \text{lshort}$ are shorted. lshort can additionally be applied to variable value inductors using the option 'short_cut_var_elem=yes'.
dcut	[...]	Replaces all diode instances using the specified model definitions with an open. dcut=all cuts all diode elements.
qcut	[...]	Replaces all BJT instances using the specified model definitions with an open. qcut=all cuts all BJT elements.
vabsshort	0.0 V	Voltage sources with absolute DC voltage less than or equal to vabsshort are shorted. Spectre by default isn't shorting any voltage source, Spectre APS uses a default value of 1e-9 V.
scalem	1.0	Model scaling factor.
scale	1.0	Device instance scaling factor.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

scalefactor	1.0	ScaleFactor for Device Model Technology Scaling. The options parameter scalefactor enables device model providers to scale device technology independent of the design dimension scaling done by circuit designers. The resulting device instance scaling is defined by 'scale * scalefactor'. If the foundry uses a technology scale factor of 0.9 (scalefactor=0.9), and the circuit designer uses a design scale factor of 1e-6 (scale=1e-6), then the compounded scaling of the device instance dimension is 0.9e-6. Unlike options parameter scale, scalefactor cannot be used as a netlist parameter and cannot be altered or used in sweep statements.
ishrink	1.0	Ishrink factor.
compatible	spectre	Encourage device equations to be compatible with a foreign simulator. This option does not affect input syntax. Possible values are spectre, spice2, spice3, cdsspice, hspice, spiceplus, eldo, sspice, mica, tispice and pspice.
diode_jtun_current	hspice	the flag to enable or disable the jtun current backward to hspice. Possible values are spectre and hspice.
approx	no	Use approximate models. Difference between approximate and exact models is generally less public. Possible values are no and yes.
auto_minductor	no	Automatic insertion of missing mutual inductor coupling. For more information, see detailed description below the parameter descriptions. Possible values are no and yes.
kmax	no	Sets 1.0 as the maximum absolute value for coupling co-efficient of mutual inductors. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

`nport_bbspice_to`    `yes`  
`_linear`

When set to yes, if BBSpice fails or generates large fitting errors, automatically switch `nport interp` option from `bbspice` to `linear`. This auto-switching depends on analysis when BBSpice generates large fitting errors. If set to no, disable this feature. Possible values are no and yes.

`nport_default_in`    `auto_switch`  
`terp`                    `tch`

If '`nport_default_interp`' is set to '`auto_switch`', '`nport`' automatically chooses '`bbspice`' for analyses, such as '`pss`' and the '`tstab`' interval of '`hb`' and '`pss`', and chooses '`linear`' for analyses, such as '`ac`', '`dc`', '`tran`', and '`sp`'. For all `nport` elements in the netlist that do not have '`interp`' set, their '`interp`' option will be set to the value specified in the global option '`nport_default_interp`'. If an '`nport`' instance has the '`interp`' option explicitly specified, the instance '`nport`' option will take priority over the global option. Possible values are `spline`, `rational`, `linear`, `bbspice`, `mvf` and `auto_switch`.

`nport_default_passivity`    `disable`

Check and enforce passivity of S-parameter for all `nport` instances. Default is `disable`, which means this global option has no effect. If set to a value other than `disable`, all `nport` elements in the netlist without a value for passivity explicitly set, will have their passivity argument set to the same value as specified in this global option. If an `nport` instance already has the passivity option specified, the instance option will take priority if both are present. Possible values are `no`, `check`, `enforce`, `fit_weak_enforce`, `fit_enforce` and `disable`.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

nportbbsfittedfiledir		The directory to which the BBSpice fitted S-Parameter file will be put. If it is not specified, the file will be in directory named as 'BBSpiceOutput'. If a relative path is specified, the path is relative to the current working directory.
nportirreuse	yes	Reuse impulse responses data for all nport instances. If no, disable this feature. Possible values are no and yes.
nportwaitbbsv2	no	If set to yes, bbspice ver. 2 will wait if another process is running bbspice ver. 2 on the same s-parameter file. Only work with nportbbsversion=2. Possible values are no and yes.
nportirfiledir		The directory to which the nport impulse response file will be written. If it is not specified, the file will be written to /home/<username>/.cadence/mmsim/. If a relative path is specified, the path is relative to the current working directory.
nportcompress	yes	Nport compression improves the efficiency of S-parameter simulation of large nport files when a certain percentage of the ports is unused, i.e., open or short circuited. Nport compression does not impact simulation accuracy. This option turns off compression if set to no and attempts to force compression if set to yes. If left unspecified, compression is on if $N \geq 10$ and the ratio of used ports is less than or equal to 0.8.. Possible values are no and yes.
nportunusedportgmin	0.0	Default is 0, which leaves the port open-circuited. A small value loads open-circuited ports with a finite but large resistance. This introduces a small error in the response, but it induces losses which help obtain a passive response.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>nportunusedportr min</code>	<code>0.0</code>	Default is 0, which leaves the port short-circuited. A small value will insert a small resistance in place of short circuited ports. This introduces a small error in the response, but it induces losses which help obtain a passive response.
<code>nportcompressfil edir</code>		The directory where the compressed nport S-parameter file is written to. If unspecified, it is stored in outdir.
<code>nportbbsmaxwaitt ime</code>	<code>0.5</code>	The maximum waiting time of Spectre if another run locks BBSpice fitting on the same S-parameter file, in unit of hour. Once waiting time reaches the specified value, Spectre exits with error.
<code>hb_lowmem_defaul t_mode</code>	<code>0</code>	Default lowmem mode for harmonic balance and following small signal analysis. If '0' is set, the low memory mode is turned off in default. If '1' is set, the standard low memory mode is turned on in default. If 'lowmem' option is explicitly specified in HB or following small signal analysis, the analysis option will take priority over this global option.

#### Noise parameters

<code>noiseon_inst</code>	<code>[...]</code>	The list of instances to be considered as noisy throughout noise analysis, which include, noise, sp noise, pnoise and tran noise.
<code>noiseoff_inst</code>	<code>[...]</code>	The list of instances not to be considered as noisy throughout noise analyses, which include noise, sp noise, pnoise and tran noise.

#### Error-checking parameters

<code>topcheck</code>	<code>full</code>	Check circuit topology for errors. Possible values are no, min, full, fixall, errmin and errfull.
-----------------------	-------------------	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

iccheck	all	Check if nodes with initial conditions have capacitive path to ground or connected to ground by vsource. IC for such node is treated as nodeset. Possible values are no, vsource, cap and all.
iccap	0 F	iccap (capacitance) is added to each IC node to prevent discontinuity between OP point and time t=0 in transient analysis.
ignshorts	no	Silently ignore shorted components. Possible values are no and yes.
diagnose	no	Print additional information that might help diagnose accuracy and convergence problems. Possible values are no, yes and detailed.
diagnose_start	0 s	Start time to enable the detailed diagnosis mode.
diagnose_end	(s)	End time to enable the detailed diagnosis mode. Default is transient stop time.
threshold_start	1	Start Gdev to enable the DC detailed diagnosis mode.
threshold_end	0	End Gdev to enable the DC detailed diagnosis mode.
debugstepdrop	no	Generates warnings on dramatic step size drop and notices when step size is recovered. Possible values are no and yes.
stepdropsize	100	The threshold size of step drop to report a dramatic change warning.
checklimitfile		File to which assert violations are written public.
dochecklimit	yes	Check asserts in the netlist. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

sfx_assert	no	Check asserts in the netlist for FX mode. Possible values are no and yes.
asserts_silent_init	[...]	Array of asserts or assert groups which don't generate message during initializing.
checklimitdest	sqldb	Destination(s) where violations are written. Possible values are file, psf, sqldb and both.
checklimitdetails	no	Print detailed information of expression. Possible values are no and yes.
checklimit_full_duration	no	If set to yes, assert violation includes the duration for which the violation takes place, and uses interpolation when the value exceeds the max/min bound value. Possible values are no and yes.
mod_assert_expand_sub	no	If expanding a mod name fails in an assert statement, try to expand the name as a subckt name. Possible values are no and yes.
probe_compatible	spectre	Flag to enable or disable optimization for probe wildcard statement. Possible values are hspice and spectre.
rampsource	yes	When performing dc hysteresis sweep, rampsource should be set to no. Possible values are no and yes.
devcheck_stat	none	Enable or disable output device-checking statistics. If set to none, no statistics related to asserts will be written to either screen or logfile. If set to sub, it reports the number of asserts per subckt per assert, which are enabled. If set to inst, it reports full instance name of an assert, which is enabled. It is not recommended when there are many asserts. Possible values are none, inst and sub.
opptcheck	yes	Check operating point parameters against soft limits. Possible values are no and yes.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Resistance parameters

<code>gmin</code>	<code>1e-12 S</code>	<code>gmin</code> (conductance) is added to each nonlinear branch of the device to prevent simulation non-convergence. Large <code>gmin</code> impacts accuracy of current probe, small <code>gmin</code> may cause circuit convergence issue. For circuit that is sensitive to leakage current, it is recommended to set <code>gmin</code> to a small value or zero.
<code>gmindc</code>	<code>1e-12 S</code>	Minimum conductance across each non-linear device in DC analysis. If <code>gmindc</code> is not specified, the value of <code>gmindc</code> will be equal to <code>gmin</code> .
<code>gminfloat</code>	<code>1e-12 S</code>	Minimum conductance added between the floating node and the ground node. If <code>gminfloat</code> is not specified, then the <code>gmin</code> value is used.
<code>gminfloatdc</code>	<code>1e-12 S</code>	Minimum conductance added between the DC floating node and the ground node in DC analysis. If <code>gminfloatdc</code> is not specified, then its value will be equal to <code>gminfloat</code> .
<code>gminfloathb</code>	<code>1e-9 S</code>	Extra conductance-like effect added between the floating node and the ground node in HB analysis, affecting only DC frequency.
<code>dcopttol</code>	<code>1e-8 S</code>	Tolerance used for <code>dcopt</code> DC analysis.
<code>fix_zero_diagonal</code> <code>1</code>	<code>no</code>	Fix matrix zero diagonal issue in simulation. Possible values are <code>no</code> and <code>yes</code> .
<code>gshunt</code>	<code>0.0 S</code>	Minimum conductance added for all nodes in tran analysis.
<code>gshuntdc</code>	<code>0.0 S</code>	Minimum conductance added for all nodes in dc analysis.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

gmindcmax	0 S	The maximum value that gmindc can be increased to during the automatically increasing gmindc method for convergence. The default value is gmindc.
dcdampsol	no	Damp DC solution during Newton iteration to assist convergence.. Possible values are no and yes.
dcfloatnodecheck	no	Relax DC floating node residue check in DC simulation to assist convergence.. Possible values are no and yes.
dcuseprevic	no	DC use previous analysis OP point as nodeset for the current DC analysis. Possible values are no and yes.
dcoptfilter	yes	Filter first-stage DC solution in DCOPT process.. Possible values are no and yes.
dcoptfiltervtol	0 V	The voltage tolerance for first-stage DC solution filtering in DCOPT process.
dc_auto_rforce	no	Automatically adjust the rforce value in DC simulation to get accurate IC setting.. Possible values are no and yes.
ignore_blowup	no	Ignore signal blowup in simulation.. Possible values are no and yes.
dc_ic_abstol	1.0e-6	IC absolute toleration for DC IC accuracy check..
dc_ic_reltol	0.001	IC relative toleration for DC IC accuracy check..
dccheck	no	DC solution physical check. If set to yes, DC will be rerun with limit=delta if the solution obtained by the first run is not physical. If set to detail, the solution differences between the two runs will be printed out in the log file.. Possible values are no, yes and detail.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

dc_check_multiple_solutions	no	Check for DC multiple solutions.. Possible values are no and yes.
dc_solution_vabs_tol	1.0e-3	DC solution voltage absolute toleration for multiple solution check..
dc_solution_vrel_tol	0.1	DC solution voltage relative toleration for multiple solution check..
dc_solution_iabs_tol	1.0e-6	DC solution current absolute toleration for multiple solution check..
dc_solution_irel_tol	0.1	DC solution current relative toleration for multiple solution check..
gmin_check	max_v_only	Specifies that effect of gmin should be reported if significant. Possible values are no, max_v_only, max_only and all.
fix_singular_matrix	no	Fix matrix singular issue by inserting gmin. Possible values are no, yes, print and both.
rforce	1 $\Omega$	Resistance used when forcing nodesets and node-based initial conditions.
va_gclamp	1 no	When va_gclamp=value is given, va g with g<value are clamped to value. Default is 0.0..
ahdl_exp_limit	1 no	When set to a positive value, ahdl exp() function will use it to limit exp() function arguments.

#### Quantity parameters

value	V	Default value quantity.
flow	I	Default flow quantity.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

quantities	no	Print quantities. If quantities=min, the simulator prints out all defined quantities; if quantities=full, the simulator also prints a list of nodes and their quantities. Possible values are no, min and full.
------------	----	---

#### Annotation parameters

audit	detailed	Print time required by various parts of the simulator. Possible values are no, brief, detailed and full.
inventory	detailed	Print summary of components used. Possible values are no, brief, detailed and full.
narrate	yes	Narrate the simulation. Possible values are no, yes and compact.
debug	no	Give debugging messages. Possible values are no and yes.
info	yes	Give informational messages. Possible values are no and yes.
note	yes	Give notice messages. Possible values are no and yes.
maxnotes	5	Maximum number of times a notice is issued per analysis. Note that this option has no effect on notices issued as part of parsing the netlist. Use the -maxnotes command-line option to control the number of all notices issued.
stver_note	no	Give -stver suggestion warning for invalid parameter. Possible values are no and yes.
warn	yes	Display warning messages. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

maxwarns	5	Maximum number of times a warning message is issued per analysis. Note that this option has no effect on warnings issued as part of parsing the netlist. Use the -maxwarns command-line option to control the number of all warnings issued.
maxwarnstologfile	5	Maximum number of times a warning message is printed to the log file per analysis. Note that this option has no effect on warnings printed as part of parsing the netlist. Use the -maxwarnstolog command-line option to control the number of all warnings printed to the log file.
maxnotestologfile	5	Maximum number of times a notice message is printed to the log file per analysis. Note that this option has no effect on notices printed as part of parsing the netlist. Use the -maxnotestolog command-line option to control the number of all notices printed to the log file.
error	yes	Generate error messages. Possible values are no and yes.
digits	5	Number of digits used when printing numbers.
measdgt	0	Number of decimal digits (in floating point numbers) in measurement output in mt0 format.
notation	eng	The notation to be used to display real numbers to the screen. Possible values are eng, sci and float.
cols	80	Width of screen in characters.
colslog	80	Width of log-file in characters.
title		Circuit title.
simstat	basic	Print simulation phase statistics report. Possible values are basic and detailed.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Matrix parameters

pivotdc	no	Use numeric pivoting on every iteration of DC analysis. Possible values are no and yes.
dc_pivot_check	no	During DC analysis, the numeric pivoting is only performed when bad pivot is detected. Possible values are no and yes.
tran_pivot_check	no	During Tran analysis, perform numerical pivoting when bad pivot is detected. Possible values are no and yes.
tran_pivot_thres hold	0.00001	When tran_pivot_check=yes, the relative pivot threshold below which the matrix will be reordered..
checklimit_skip_ subs	[...]	Array of subcircuit masters or subcircuit master patterns to be skipped in device checking. Patterns can have any wildcard symbols.
checklimit_skip_ file		File which contains the subcircuit masters or subcircuit master patterns to be skipped in device checking. Patterns can have any wildcard symbols.
checklimit_skip_ insts	[...]	Array of instances to be skipped in device checking. Instances can have any wildcard symbols.
checklimit_skip_ insts_file		File which contains the instances to be skipped in device checking. Instances can have any wildcard symbols.
checklimit_skip_ ie_connect_devic es	0	The devices connected to AMS IE elements (hierarchy level must be lesser than the value specified for this option) will be skipped in device checking.
pivrel	0.001	Relative pivot threshold.
pivabs	0	Absolute pivot threshold.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>complexpivrel</code>	<code>1e-5</code>	Relative pivot threshold for small signal analyses.
<code>preorder</code>	<code>partial</code>	Try this option when simulation runs out of memory or if the simulation is unreasonably slow for the size of your design. It controls the amount of matrix pre-ordering that is done and may lead to much fewer matrix fill-ins in some cases. Known cases include designs with very large number of small resistors and large number of behavioral instances containing voltage based equations. Possible values are <code>partial</code> , <code>full</code> and <code>limited</code> .
<code>limit_diag_pivot</code>	<code>yes</code>	If set to <code>yes</code> , there is a limit on the number of matrix fill-ins when selecting a pivot from a diagonal. For backward compatibility, set this option to <code>no</code> . Possible values are <code>no</code> and <code>yes</code> .
<code>rebuild_matrix</code>	<code>no</code>	If set to <code>yes</code> , rebuild circuit matrix at the beginning of <code>ac</code> , <code>dc</code> , <code>dcmatch</code> , <code>montecarlo</code> , <code>pz</code> , <code>stb</code> , <code>sweep</code> , <code>tdr</code> , and <code>tran</code> analyses. This is to ensure consistent matrix ordering at the beginning of the analyses for consistent results. Notice that rebuild circuit matrix can result in performance overhead. Possible values are <code>no</code> and <code>yes</code> .

#### Fault analysis parameters

<code>faultmerge</code>	<code>yes</code>	If <code>no</code> , generate a separate fault for each instance that has an optimized parallel connectivity and iterated instances that have ' <code>&lt; number [ : number ] &gt;</code> ' in their name. If <code>yes</code> , reduce the number of defects by taking into account merged terminals for iterated and parallel devices. This parameter is applicable in the <code>+aps</code> and <code>+preset</code> flows only. Possible values are <code>no</code> and <code>yes</code> .
<code>faultcustomprese rve</code>	<code>[...]</code>	Preserve all the specified masters for custom fault generation and simulation.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>faultmacro</code>	<code>no</code>	If yes, generates the requested faults only at the boundary of the subcircuits specified by the <code>mosmacro</code> option and skip all instances below those subcircuit levels. If no, generate the faults at all levels in the specified subcircuits. Possible values are no and yes.
-------------------------	-----------------	---

#### Sensitivity parameters

<code>sensfile</code>		Output sensitivity data file name.
<code>sensformat</code>	<code>tabular</code>	Format of sensitivity data. Possible values are tabular and list.
<code>senstype</code>	<code>partial</code>	Type of sensitivity being calculated. Possible values are partial and normalized.
<code>sensfileonly</code>	<code>no</code>	Enable or disable raw output of sensitivity. results. Possible values are no and yes.
<code>sensbinparam</code>	<code>no</code>	Sensitivity for binning models. Possible values are no, uncorrelated and correlated.
<code>paramrangefile</code>		Parameter range file.

#### Performance parameters

<code>minr</code>	<code>0.0</code>	All parasitic resistors inside devices less than global <code>minr</code> will be removed. The order of checking devices is the follows: 1. Check if resistors are smaller than local <code>minr</code> . If yes, check if it is a MOSFET or BJT. If it is a MOSFET, drop the resistor, if it is BJT, clamp to the <code>minr</code> value, and give a warning message for both cases. 2. Check global <code>minr</code> , All Parasitic resistors less than global <code>minr</code> are removed and a warning message is issued. 3. If the resistor is not removed and is smaller than 0.001, issue a warning.
-------------------	------------------	--



## Spectre Circuit Simulator Reference

### Analysis Statements

---

short_bjtr	0.0	All parasitic resistors inside bipolar devices less than global short_bjtr will be removed. This option is applied to all bipolar junction transistors.
cscalefactor	1e-3	Scaling factor for very large capacitor.
cthresh	1e38	Large capacitors can cause transient convergence problems, especially as the time-step shrinks. This is similar to the issues that small resistors cause. There is an alternative exact representation of a capacitor, which does not suffer from this problem, but can introduce an equation to be solved. If a capacitance is greater than cthresh, then the alternative formulation is used. Accuracy is not impacted by the change.
highvoltage	no	Enable optimized Spectre settings for high voltage designs including voltage, and current binning, excluding VerilogA and dangling nodes from convergence checks, and optimized large capacitance handling. Possible values are no and yes.
vrefgnd		Specify reference ground.
macro_mode		Specify macro_mode configuration.
verilogalang	relax	AHDL Verilog-A language syntax check mode selector. If verilogalang=strict, show archaic syntax input as an error during Verilog-A parsing; If verilogalang=relax, show archaic syntax input as a warning during Verilog-A parsing. Possible values are strict and relax.
minstepversion	1	Minstep algorithm flow control. If minstepversion=0, desperate big jump is taken when minstep is reached. If minstepversion=1, a forward/backward search algorithm is taken to find a converged solution at small step size. By default, minstepversion=1.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

max_minstep_nonc onv	200	The maximum number of convergence failures allowed below minstep within 5% of stop time.
max_approach_min step	300	The maximum number of times allowed to approach minstep within 5% of stop time.
max_consecutive_ minstep	1000	The maximum number of consecutive steps allowed to be less than 10*minstep.
discontinuity	no	Detect device discontinuity and enable robust transient simulation. Possible values are no and robust.
autostop	no	If yes, tran analysis is terminated when all event-type measurement expressions have been evaluated. Event-type expressions use thresholding, event, or delay type functions. If the value is spice, autostop is consistent with spice simulator. Possible values are no, yes and spice.
large_newton_rej ect	1	Reject Newton iteration when large residue or delta solution occurs during transient analysis.

### Model parameters

soft_bin	singlemo dels	If set to singlemodels, it is used only on non-binned models. Possible values are singlemodels, no and allmodels.
xpssdc_disable_o utput	1	When set to 1, disable waveform output for xpssdc child process.
filter_wildcard_ colon_ic	0	When set to 1, ignore colon node in ic wildcard expansion.
filter_wildcard_ terminal_ic	0	When set to 1, ignore terminal in ic wildcard expansion.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

postlayoutpreset	...	Enables local net based postlayout optimization. Common application is to disable postlayout optimization for differential signals in comparators and other sensitive circuits. Example: opt1 options postlayoutpreset=off postlnets=[I1.in1 I1.in2]. Wildcard is supported for the net name (I1.*). Default takes the value from the +postlayoutpreset command line option.. Possible values are off, cx, ax, mx, lx and vx.
rcr_net_vpn		The virtual power node names, globally.
rcr_net_vpn_groups		The virtual power node groups, globally.
protect_sub		The subckt names specified in the vector and its descendant subckts will be preserved.
protect_inst		The inst names specified in the vector and its descendant insts will be preserved.
rcr_net_node_auto_res_bump		The auto res_bump value specified on the value.
rcr_net_node_fmax		The fmax specified will be applied to the RCR engine of the RC network which contains the node.
rcr_net_preserve_pn_ports	0	When set to 1, preserves ports of parasitic networks that are connected to a DC vsource.
rcr_net_preserve_ground_reg_ports	0	When set to 1, preserves ports of parasitic networks that are connected to a ground regular type.
rcr_net_preserve_vpn_ports	0	When set to 1, preserves ports of parasitic networks that are connected to a VPN source.
rcr_net_preserve_active_word_line_ports	0	When set to 1, preserves ports of parasitic networks that are connected to an active word-line node.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>rcr_net_preserve _idle_word_line_ ports</code>	0	When set to 1, preserves ports of parasitic networks that are connected to an idle word-line node.
<code>rcr_net_preserve _active_bit_line_ ports</code>	0	When set to 1, preserves ports of parasitic networks that are connected to an active bit-line node.
<code>rcr_net_preserve _idle_bit_line_p orts</code>	0	When set to 1, preserves ports of parasitic networks that are connected to an idle bit-line node.
<code>rcr_net_preserve _active_state_po rts</code>	0	When set to 1, preserves ports of parasitic networks that are connected to an active-state node.
<code>rcr_net_preserve _idle_state_port s</code>	0	When set to 1, preserves ports of parasitic networks that are connected to an idle-state node.
<code>rcr_net_preserve _san_to_dout_por ts</code>	0	When set to 1, preserves ports of parasitic networks that are connected to a san-to-dout node.
<code>rcr_net_preserve _din_to_bitline_ ports</code>	0	When set to 1, preserves ports of parasitic networks that are connected to a din-to-bitline node.
<code>rcr_net_preserve _san_ports</code>	0	When set to 1, preserves ports of parasitic networks that are connected to a san node.
<code>rcr_net_preserve _active_sa_to_gn d_ports</code>	0	When set to 1, preserves ports of parasitic networks that are connected to active nodes from sa to gnd.
<code>rcr_net_preserve _idle_sa_to_gnd_ ports</code>	0	When set to 1, preserves ports of parasitic networks that are connected to idle nodes from sa to gnd.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>rcr_net_preserve</code>	0	When set to 1, preserves ports of parasitic networks that are connected to active nodes from clk to sa-to-gnd controlling gate.
<code>_active_clk_to_sgc_ports</code>		
<code>rcr_net_preserve</code>	0	When set to 1, preserves ports of parasitic networks that are connected to idle nodes from clk to sa-to-gnd controlling gate.
<code>_idle_clk_to_sgc_ports</code>		
<code>rcr_net_preserve</code>	0	When set to 1, preserves ports of parasitic networks that are connected to active heavy load nodes.
<code>_active_heavy_load_ports</code>		
<code>rcr_net_preserve</code>	0	When set to 1, preserves ports of parasitic networks that are connected to idle heavy load nodes.
<code>_idle_heavy_load_ports</code>		
<code>rcr_net_preserve</code>	0	When set to 1, preserves ports of parasitic networks that are connected to a source and drain device terminals, when set to 2, preserves drain/source on idle reg network, preserves on all ports for other no type network, when set to 3, preserves drain/source ports, when set to 4, do not preserve port for idle REG network, preserve all ports or drain/source port on other no type networks..
<code>_drain_source_signals</code>		
<code>rcr_net_preserve</code>	0	When set to 1, preserves drain/source on idle reg network..
<code>_idle_reg_drain_source</code>		
<code>rcr_net_preserve</code>	0	When set to 1, preserves drain/source on active reg network..
<code>_active_reg_drain_source</code>		
<code>rcr_net_merge_pn</code>	0	When set to 1, allows merging of those ports of power or vpn parasitic networks, that are connected to idle state nodes.
<code>_idle_state_ports</code>		
<code>rcr_net_active_word_line_only</code>	0	When set to 1, short all resistors of the parasitic networks that are connected to an active word-line.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>rcr_net_idle_word_line_only</code>	0	When set to 1, short all resistors of the parasitic networks that are connected to an idle word-line.
<code>rcr_net_active_bit_line_only</code>	0	When set to 1, short all resistors of the parasitic networks that are connected to an active bit-line.
<code>rcr_net_idle_bit_line_only</code>	0	When set to 1, short all resistors of the parasitic networks that are connected to an idle bit-line.
<code>rcr_net_active_state_only</code>	0	When set to 1, short all resistors of the parasitic networks that are connected to an active state.
<code>rcr_net_idle_reg_only</code>	0	When set to 1, short all resistors of the parasitic networks that are connected to an idle reg signal.
<code>rcr_net_idle_state_only</code>	0	When set to 1, short all resistors of the parasitic networks that are connected to an idle state.
<code>rcr_net_state_idle_vpn_only</code>	0	When set to 1, short all resistors of Idle VPN parasitic networks that are connected to state parasitic networks.
<code>rcr_net_floating_1_only</code>	0	When set to 1, short all resistors of FL1 parasitic networks that are connected to idle state parasitic networks.
<code>rcr_net_floating_2_only</code>	0	When set to 1, short all resistors of FL2 parasitic networks that are connected to idle state parasitic networks.
<code>rcr_net_floating_3_only</code>	0	When set to 1, short all resistors of FL3 parasitic networks that are connected to idle state parasitic networks.
<code>rcr_net_ground_only</code>	0	When set to 1, short all resistors of ground parasitic networks.
<code>rcr_net_ground_reg_only</code>	0	When set to 1, short all resistors of ground reg parasitic networks.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>rcr_net_active_sa_to_gnd_only</code>	0	When set to 1, short all resistors of the parasitic networks that are connected to active nodes from sa to gnd path.
<code>rcr_net_idle_sa_to_gnd_only</code>	0	When set to 1, short all resistors of the parasitic networks that are connected to idle nodes from sa to gnd path.
<code>rcr_net_dump_protect</code>	0	this parameter is used to dump all preserved nodes in RCR flow.
<code>rcr_net_option_file</code>		User specified RCR option for each specific node.
<code>rcr_fmax</code>	1	this parameter is used to set the max frequency used by rc reduction.
<code>rcr_rshort</code>	1.00E-03	this parameter is used to set the minimum value of resistor that should be cut.
<code>rcr_cgnd</code>	1.00E-18	this parameter is used to set the minimum value of capacitance that should be decoupled to ground.
<code>rcr_net_cgnd</code>	1.00E-18	this parameter is used by the rc net flow to set the minimum value of capacitance that should be decoupled to ground.
<code>bsource_rmin_val_mfactor</code>	ignore	Defines whether the bsource resistor min_val is scaled with the M-factor (scaled) or not (ignored). Possible values are scaled and ignore.
<code>mosvar_noscale</code>	0	Enable(0)/Disable(1) scale support for mosvar model.
<code>tmisweep</code>	0	Enable(1)/Disable(0) printing out of tmideg files when etmiPrint=1..
<code>dcmatchscaling</code>	0	Backward compatibility controller for dcmatch scaling behavior.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

dcmatch_fitting	0	Dcmatch fitting (sigmaOut) for customer.
negative_cap_opt	no	Specifies the compensation level for negative capacitances. The degree of compensation is yes > es > hs > vs > ds > hfs > dcs > no. Default value is 'vs' for the mx, lx, and vx preset options of SpectreX and 'no' for other Spectre modes.. Possible values are no, yes, es, hs, vs, ds, hfs and dcs.
negative_cap_opt _devices	[...]	Specifies the models that should be compensated. For example, <option_name>=[bsim3v3] enables compensation for bsim3v3 model. <option_name>=[bsim3v3 bsim4] enables compensation for both bsim3v3 and bsim4 models..
etmiprintfd	0	Output the TMI result file. By default it is closed.
dump_global_opti ons	0	Internal option for RelXpert to dump all the global options . By default it is closed.
tmiaging_remove_ slash	0	Internal option to remove the slash character before dot in instance name. By default it is closed.
output_compact_m apping	0	Internal option for RelXpert to output compact mapping file. By default it is closed.
tmipath		Location of TMI shared object libraries to be used by tmiBsim4 models.
tmiflag	0	Activate TMI flow. By default TMI flow is not activated.
tmimtsflag	0	Activate TMI MTS flow. By default TMI MTS flow is not activated.
macinitforminiso lver	0	Add to judge if plnit is called from Minisolver.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

omipath		Location of OMI shared object libraries to be used by omi models.
omiflag	0	Activate OMI flow. By default OMI flow is not activated.
omiage	0	Activate OMI Aging model flow. By default, it is not activated.
omisave	1	Enable or disable aging information to be saved.
omiinput		Input file name, including the full path to read back TMI information for aging model.
omisort		Sort aging information.
tmiage	0	Activate TMI Aging model flow. By default, it is not activated.
tmiinput		input file name including full path to read back TMI information for aging model.
tmisave	1	Enable or disable aging information to be saved.
etmiusrinput		This option can be set in model files or netlists.
tmisort		Sort aging information.
print_including	no	Enable Spectre to print including information in include statement. Possible values are no and yes.
annotateonalter	yes	Annotate on all alter statements. Possible values are no and yes.
reelaborateonalter	force	Whether re-elaborate for every alter statement. Possible values are lazy and force.
scale_redefined	ignore	Disallow redefining of the option scale in netlist. Possible values are error, ignore and warning.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>printstep</code>	<code>no</code>	Enable Spectre to print results by equal step defined in <code>.tran</code> statement. Possible values are <code>no</code> and <code>yes</code> .
<code>inlinesubcktcurrent</code>	<code>subckt</code>	Save inline subcircuit terminal current as inline device current or subcircuit current. By default, inline subckt current is saved. Possible values are <code>device</code> and <code>subckt</code> .
<code>nonconv_topnum</code>	<code>10</code>	Top number of non-convergence nodes to be printed public.
<code>dotprobeformat</code>	<code>flat</code>	Print <code>.probe</code> signal with original name or hierarchical name. Possible values are <code>flat</code> and <code>hier</code> .
<code>hierprobe</code>	<code>hier</code>	Controls the hierarchical delimiter used in a hierarchical name. When <code>flat</code> , the delimiter used is the value of the <code>hier_delimiter</code> option. When <code>hier</code> , the delimiter used is <code>.</code> or <code>dot</code> . Possible values are <code>flat</code> and <code>hier</code> .
<code>dcmmod</code>	<code>0</code>	DCMismatch analysis version selector. If set to 1 or 3, uses Bsim short-channel mismatch equation for BSIM3 and BSIM4 devices; if set to 2 or 3, provides compatibility with Monte Carlo analysis.
<code>vthmod</code>	<code>std</code>	Vth output selector. 'std' outputs model equation Vth; 'vthcc' outputs constant current Vth and may impact simulation performance; 'vthcc_ext' outputs constant current Vth on external nodes. Possible values are <code>std</code> , <code>vthcc</code> and <code>vthcc_ext</code> .
<code>noisemethod</code>	<code>noign</code>	option to control induced gate noise. <code>noign</code> : no induced gate noise. <code>subckt</code> : compatible with verilogA. Possible values are <code>noign</code> and <code>subckt</code> .
<code>vdsatmod</code>	<code>std</code>	Vdsat output selector. Possible values are <code>std</code> , <code>gds</code> and <code>slow_gds</code> .
<code>vdsat_c</code>	<code>0.08</code>	Coefficient for Gds based Vdsat.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>vdsat_vdl</code>	<code>0.3 V</code>	Coefficient for Gds based Vdsat.
<code>vdsat_vadj</code>	<code>0.0 V</code>	Coefficient for Gds based Vdsat.
<code>ivthn</code>	<code>0.0 A</code>	NMOS Vth current parameter.
<code>ivthp</code>	<code>0.0 A</code>	PMOS Vth current parameter.
<code>ivthw</code>	<code>0.0 m</code>	Width offset for constant current Vth.
<code>ivthl</code>	<code>0.0 m</code>	Length offset for constant current Vth.
<code>ivth_vdsmin</code>	<code>0.05 V</code>	Minimum Vds in constant current Vth calculation public.
<code>gform_vcr</code>	<code>0</code>	Use Gform for VCR device. Specify 1 for yes and 0 for no. Default is no.
<code>bin_relref</code>	<code>no</code>	When a global relref is used, signals share a common reference value for convergence checks. If there is a large voltage in the circuit this can artificially relax convergence checks for other signals. This option creates voltage bins so that large voltages do not impact nodes with small voltage swings. Possible values are no and yes.
<code>optimize_bsource</code>	<code>no</code>	Enable the optimization in bsource evaluation. Default value is res in ++aps and no in others.. Possible values are no, yes, res, cap, nores, nocap and rescap.
<code>optimize_diode</code>	<code>no</code>	Enable the optimization in diode evaluation. Possible values are no and yes.
<code>opt_diode_enhance</code>	<code>no</code>	Enable the optimization in diode evaluation in spectre x. Possible values are no and yes.
<code>optimize_bsimcmg</code>	<code>no</code>	Enable the optimization in bsimcmg evaluation. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>enable_device_bbox</code>	<code>no</code>	Enable writing bbox to param map. Possible values are no and yes.
<code>memory_estimate</code>	<code>no</code>	Enable calculate not optimized CMI memory. Possible values are no and yes.
<code>mdlthresholds</code>	<code>exact</code>	When set to exact, certain functions in MDL, for example, <code>cross()</code> , control the transient time-step to place a time-point at a threshold crossing. This can improve accuracy. However for applications such as cell characterization an interpolated value can give the required accuracy with better performance.. Possible values are exact and interpolated.
<code>dio_allow_scaling</code>	<code>no</code>	Use this flag to enable SCALE parameter for diode mode. Possible values are no and yes.

#### Stitching parameters

<code>spf</code>		This option specifies the DSPF file to be stitched and its stitching scope. The syntax is <code>spf="scope filename"</code> . The scope can be a subcircuit or an instance. Use "spectre -h stitch" for more information.
<code>dpf</code>		This option specifies DPF file to be stitched and its stitching scope. The syntax is <code>dpf="scope filename"</code> . The scope can be a subcircuit or an instance. Use "spectre -h stitch" for more information.
<code>spef</code>		This option specifies the SPEF file to be stitched and its stitching scope. The syntax is <code>spef="scope filename"</code> . The scope can be a subcircuit or an instance. Use "spectre -h stitch" for more information.
<code>spfscale</code>	<code>1.0</code>	This option specifies the scaling factor of all the elements in the parasitic files (DSPF/SPEF/DPF). Use "spectre -h stitch" for more details.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>spfscaler</code>	1	This option specifies the scale factor for parasitic resistors.
<code>spfscalec</code>	1	This option specifies the scale factor for parasitic capacitors.
<code>spfinstancesection</code>	1	This option controls the backannotation of device parameters in the instance section of the DSPF file. If <code>spfinstancesection</code> is turned off, the instance section is ignored (that is, the device parameters are not changed during stitching). The default is on. Use "spectre -h stitch" for more information.
<code>spfxtorprefix</code>		This option specifies the prefix in the device or net names in the DSPF/SPEF/DPF file. The syntax is <code>spfxtorprefix="substring [replace_substring]"</code> . Use "spectre -h stitch" for more information.
<code>speftriplet</code>	2	This option specifies the value to be used for stitching in the SPEF file. It is effective only when the values in the SPEF file are represented by triplets (for example, 0.325:0.41:0.495).. Possible values are 1, 2 and 3.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

`spfbusdelim`

This option maps the bus delimiter between schematic netlist and parasitic file (i.e. DSPF, SPEF, or DPF). The option defines the bus delimiter in the schematic netlist, and optionally the bus delimiter in the parasitic file. By default the bus delimiter of the parasitic file is taken from the parasitic file header (that is, `*IBUSBIT []`, `*IBUS_BIT []`, or `*IBUS_DELIMITER []`). If the bus delimiter is not defined in the parasitic file header, you need specify it using the `spfbusdelim` option in schematic netlist. For example, the option `spfbusdelim="<>"` maps `A<1>` in the schematic netlist to `A_1` in the DSPF file, if the bus delimiter header in the DSPF file is `"_"`. The option `spfbusdelim="@ []"` maps `A@1` in the schematic netlist to `A[1]` in the DSPF file (the bus delimiter in DSPF header will be ignored). See `"spectre -h stitch"` for more information.

`spfinstarraydeli  
m`

This option specifies the supplemental bus delimiter, if more than one bus delimiter is used in pre-layout netlist (usually for instance array indexing). It is similar to the `spfbusdelim` option. Use `"spectre -h stitch"` for more information.

`spfpstparam`      `replace`

This option specifies the instance parameters sourced for stitched instances. When set to `replace`, only the post-layout parameters are used and the pre-layout parameters are discarded. When set to `append`, both pre-layout and post-layout parameters are merged for the stitched instances where the post-layout parameters have higher priority. Possible values are `append` and `replace`.

`spfswapterm`

This option specifies the swappable terminals of a subcircuit macro-model. The syntax is `spfswapterm="terminal1 terminal2 subcktname"`. Use `"spectre -h stitch"` for more information.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

spfaliasterm		<p>At times, the terminal names of devices in DSPF/ SPEF/DPF files are different from those in the simulation model library. This happens in technology nodes that use subcircuits to model devices. The syntax is <code>spfaliasterm=&lt;modellsubckt&gt;</code> <code>&lt;prelayout_term0&gt;=&lt;spf_alias1&gt;</code> <code>&lt;prelayout_term2&gt;=&lt;spf_alias2&gt; ...</code> <code>&lt;prelayout_termN&gt;=&lt;spf_aliasN&gt;</code>". Multiple statements are supported. Use "spectre -h stitch" for more information.</p>
spfpwrnet		<p>This option specifies power net during stitching..</p>
spfcnet		<p>This option specifies the net that has its total capacitance stitched. All other parasitic components such as parasitic resistors that are associated with this net are ignored. The complete hierarchical names are required and multiple statements are supported. Wildcards are supported. Use "spectre -h stitch" for more information.</p>
spfrcnet		<p>This option specifies the name of the net to be stitched with parasitic resistors and capacitors. The other nets are stitched with lumped total capacitances. Multiple statements are supported. Wildcards are supported and you can specify as many nets as needed. Complete hierarchical net names are required. Use "spectre -h stitch" for more information.</p>
spfnetcmin	0	<p>This option allows you to select the net for stitching by the value of its total node capacitance. If a net's total node capacitance exceeds spfnetcmin, all parasitics associated with the net are stitched correctly, otherwise, only the total capacitance is added to the net node. Use "spectre -h stitch" for more information.</p>

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>spfskipnetfile</code>		This option allows you to specify the nets to be skipped as a list in a text file. The syntax is <code>spfskipnetfile="filename"</code> . Only one file can be specified. The format in the file is one line per net. Use "spectre -h stitch" for more information.
<code>spfmglimit</code>		This option specifies the maximum number of messages to be printed in the <code>spfrpt</code> file. The syntax is <code>spfmglimit="number STITCH-ID_1 STITCH-ID_2"</code> . Use "spectre -h stitch" for more details.
<code>spfskipnet</code>		This option specifies the net to be skipped for stitching, that is, all parasitic components of the net are not stitched. Wildcards are supported. You can specify multiple <code>spfskipnet</code> statements. Use "spectre -h stitch" for more information.
<code>spfcnetfile</code>		This option has the same functionality as <code>spfcnet</code> , except that it accepts a text file in which all the C-only stitched nets are listed. Only one file can be specified. The syntax is <code>spfcnetfile="filename"</code> . The format of the file requires you to specify one line per net. Use "spectre -h stitch" for more information.
<code>spfrcnetfile</code>		This option has the same functionality as <code>spfrcnet</code> , except that it accepts a text file in which all the RC stitched nets are specified. Only one file can be specified. The syntax is <code>spfrcnetfile="filename"</code> . The format of the file requires you to specify one line per net. Use "spectre -h stitch" for more details.
<code>spfshortpin</code>	<code>yes</code>	short pin nodes if set to yes or 1.. Possible values are no and yes.
<code>spf_probe_mode</code>	<code>1</code>	Option to control probe name for <code>spf</code> node. If 0, flattened node name in format <code>&lt;net_name&gt;#&lt;flatten_nodename&gt;</code> , if 1, hierarchical node name following the net. Possible values are 0 and 1.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

`spfskipinst` This option specifies the instance to be skipped for stitching, that is, the nets and instances of this instance are not stitched. Wildcards are supported. You can specify multiple `spfskipinst` statements. Use "spectre -h stitch" for more information.

`spfskipsubckt` This option specifies the subckt to be skipped for stitching, that is, the nets and instances in this subckt are not stitched. You can specify multiple `spfskipsubckt` statements. Use "spectre -h stitch" for more information.

#### Miscellaneous parameters

`ckptclock`                      0 s      Clock time checkpoint period. Default is 1800s for Spectre.

`param_topchange`            yes      If set to yes, Spectre will support parametric topology change caused by device internal node changes when the device, model, or netlist parameter value is altered.. Possible values are no and yes.

`redefinedparams`            error      Specify whether parameters can be redefined in the netlist. When set to warning or ignore, the simulator allows you to redefine parameters in the netlist. However, it honors only the last definition of the redefined parameter. Depending on the value set, the simulator displays warning messages for the redefined parameters or does not display any message. When set to error, the simulator does not allow you to redefine parameters in the netlist and displays an error message. Possible values are error, ignore, warning and warn.

`redefinableparam`  
s      If `redefinedParams` is enabled, parameters in this option can still be redefined..

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>duplicateports</code>	<code>error</code>	Specify whether duplicate ports are allowed in the definition of the subcircuit. When set to warning or ignore, the duplicate ports are shorted. Depending on the value that is set, the simulator displays warning messages for the duplicate ports or does not display any message. When set to error, the simulator does not allow duplicate ports in the definition of the subcircuit and displays an error message. Possible values are error, ignore and warning.
<code>duplicate_subckt</code>	<code>error</code>	Specify whether duplicate subcircuit definitions are allowed. When set to warning or ignore, the simulator allows duplicate subcircuit definitions. However, it honors only the last subcircuit definition. Depending on the value that is set, the simulator displays warning messages for the duplicate subcircuit definitions or does not display any message. When set to error, the simulator does not allow duplicate subcircuit definitions and displays an error message. Possible values are error, ignore and warning.
<code>table_param_rede</code> <code>fined_input</code>	<code>error</code>	Specify whether lines with total duplicate input columns are allowed in table file of <code>table_param</code> . When set to warning or ignore, the simulator allows lines with total duplicate input columns. However, it honors only the last line definition. Depending on the value that is set, the simulator displays warning messages for the lines with total duplicate input columns definitions or does not display any message. When set to error, the simulator does not allow lines with total duplicate input columns and displays an error message. Possible values are error, ignore and warning.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>duplicateinstance</code>	<code>error</code>	Specify whether duplicate instance definitions are allowed. When set to warning or ignore, the simulator allows duplicate instance definitions. However, it honors only the last instance definition. Depending on the value that is set, the simulator displays warning messages for the duplicate instance definitions or does not display any message. When set to error, the simulator does not allow duplicate instance definitions and displays an error message. Possible values are error, ignore and warning.
<code>duplicatemodel</code>	<code>error</code>	Specify whether duplicate Spectre model definitions are allowed. When set to warning or ignore, the simulator allows duplicate model definitions. However, it honors only the last model definition. Depending on the value that is set, the simulator displays warning messages for the duplicate model definitions or does not display any message. When set to error, the simulator does not allow duplicate model definitions and displays an error message. Possible values are error, ignore and warning.
<code>rmdbg_preserve</code>	<code>no</code>	When enabled, all instances of masters specified in option <code>preserve_master</code> will not be removed even if they are dangling.. Possible values are no and yes.
<code>warning_limit</code>	<code>5</code>	The maximum number of warning messages to be displayed according to specified in the immediately following <code>warning_id</code> .
<code>use_veriloga</code>	<code>0</code>	Determine whether to take Verilog-A as high priority. If set to '1', the Verilog-A model takes priority over the other subckt/models. If set to '0', the Verilog-A model does not take priority over the other subckt/models..
<code>warning_change_severity</code>	<code>warning</code>	Change the severity of warning messages specified in the immediately following <code>warning_id</code> . Possible values are error, warning and notice.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

warning_id	[...]	Vector of warning message identifiers, such as [SPECTRE-16462 SPECTRE-16684]. Used in conjunction with warning_limit or warning_change_severity.
preserve_master		Preserve all instances of the masters specified.
preserve_inst		Preserve the instances specified in the vector. Use 'preserve_inst=all' to preserve all instances.
preserve_subckt		Preserve all instances within the subcircuits specified.
preserve_assert	lazy	Specify whether to execute the preservation for voltage check for assert statements. If the value is lazy, only execute the preservation for current checking; otherwise it will execute complete preservation. Possible values are lazy and force.
soa_warn	yes	If soa_warn is no, SOA check will be disabled. Possible values are no and yes.
idovvdsclamp	1e-9	Option to tune the clamp value of IdovVds..
soa_dest	file	Destinations where SOA violations are written. If set to file, violations will be written in log. If set to sqldb, violations will be written in sqldb. Possible values are file and sqldb.
BHT_New_Eval	1	Option to choose which kind of evaluate function to use, 0 for old version with Vciei derivates and 1 for new one without Vciei derivates.
cmi_opts	0	The option for customer cmi.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>parasitics</code>	<code>0</code>	Set parasitics for RC reduction. When used as the scoped option, specify the "+mts" command line option. The option has higher priority than the value from the command line option, "+parasitics". When used as a global option, the value from the command line, "+parasitics", has a higher priority.
<code>portmismatch_severity</code>	<code>error</code>	If set ignore/warn/error out when ports number mismatch for subckt.. Possible values are error, ignore and warning.
<code>implicit_subckt_param</code>	<code>no</code>	Allows the implicit parameter to be declared for the spectre format. Possible values are no and yes.
<code>mdltrigtargmode</code>	<code>cross</code>	This option specifies how to evaluate the trig and targ of the output format. If mdltrigtargmode = deltax, we will use the deltax function to evaluate the trig and targ function. If mdltrigtargmode = cross, we will use the cross function to evaluate the trig and targ function. If mdltrigtargmode = details, we will output the trig and targ middle results. Possible values are deltax, cross and details.
<code>leaki_times</code>	<code>0</code>	Specifies input-steady state time points.
<code>macro_mos</code>	<code>[...]</code>	The list of subcircuit names, which is a macro model.
<code>skip_macro_mos</code>	<code>[...]</code>	The list of subcircuit names, which is skipped to be a macro model.
<code>hier_delimiter</code>	<code>.</code>	Used to set hierarchical delimiter. Length of hier_delimiter should not be longer than 1, except the leader escape character.
<code>missing_icnodeset</code>		Write the node which is not in the ic or nodeset to the file.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

hier_ambiguity	strict	When set to lower, the simulator will partially flatten the hier name from the back to check if an object can be found. Lower is not recommended. Possible values are strict and lower.
skip	none	Skip the simulating specified subckt instances. For now, cut is supported, which will leave the subckt ports disconnected. Possible values are none, cut and load.
podeautovdd	0	When set to 1, find the max vdd of each pode automatically.
hasmcmfactorcorrection	0	Specifies whether this subckt or subckt instance has mfactor correction for montecarlo.
mdlpsfoutput	no	mdl output psf file into raw file or not.. Possible values are no and yes.
exprProbeSeparated	yes	for Fx or XPS, the expression probe output into separated stream or not. Possible values are no and yes.
mc_scalarfile_flush_gradual	yes	Determines if scalarfile was printed incremental or not.. Possible values are no and yes.
mc_scalarfile_stat	default	Determines how scalarfile statistics data is printed.. Possible values are default and transposed.
mc_stat_list	default	Determines how many scalarfile statistics function is printed.. Possible values are default and all.
measureterm	pass	If set to error, error out if two measures refer to the same node. Possible values are pass and error.
devropt	none	If set to all, diode, or mosfet, connected resistors will be folded to related instances.. Possible values are none, all, diode and mosfet.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

preservenode	none	preservenode=save - preserve node during RC reduction if it is saved with individual save statement; make node port of rcnet, don't allow this node to be merged in port merging, no impact on node saved with wildcard. preservenode=noneexplicit - don't preserve saved nodes in RC reduction, just use waveform aliases (default). preservenode=all - doesn't do anything.. Possible values are none, all, explicit and save.
probedepthcount	hierarchy	Hierarchy count the instance level, delimiter counts the delimiter level. Possible values are hierarchy and delimiter.
subckt_switch	full	If set to simple, complex subckt will be mapped to simple subckt in mapsubckt. Possible values are full and simple.
duplicate_module	error	Specifies whether the duplicate module definitions are allowed. When set to warning or ignore, the simulator allows duplicate module definitions. However, it honors only the last module definition. Depending on the value that is set, the simulator decides whether to display warning messages for the duplicate module definitions or not. When set to error, the simulator does not allow the duplicate module definitions and displays an error message. Possible values are error, ignore and warning.
ms_vpn		The virtual power node names, globally.
ms_vpnv	0 V	The voltage value for the virtual power network sources in scope.
ms_vgnd		The virtual ground node names, globally.
ms_vgndv	0 V	The voltage value for the virtual power network grounds in scope.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

mtlinereuse	yes	This parameter defines whether to reuse RLGC data for all mtline instances. If no, this feature is disabled. Possible values are no and yes.
mtlinecachedir		The directory in which the mtline RLGC data file will be written. If an absolute path is not given, the file will be written to /home/<username>/.cadence/mmsim/.
mtlineemsolver	cpl	EM 2D solver for transmission line. Possible values are lmg and cpl.
disable_save_pre serve	no	Disables preserve by save statements. If no, does not disable the save preserve. If yes, disables the save preserve. Possible values are no and yes.
mdlanalysisnames	original	Flag to indicate whether or not to keep spectre analysis name in MDL flow. Possible values are rename and original.
check_fanout_fil ter_mode	0	Determine the fanout filtering mode: 0 is net based, 1 is node based. Default is 0..
check_format		Determine the format of the checker violation report. If set to sql, format is sqldb. If set to xml, format is xml and sqldb. If set to text, format is text and sqldb. If set to csv, format is csv and sqldb. If set to all, format is xml, text and sqldb.. Possible values are xml, text, csv, sql and all.
mismatchlevel	0	With default, mismatch parameter is defined based on parameter scope concept: mismatch parameter can only have one value in one scope. For example, if a mismatch parameter appears multiple times in a subckt of the netlist, that mismatch parameter has a unique value in that subckt for each Monte Carlo run. With mismatchlevel=1, the mismatch parameter is allowed to have a different value for each appearance in the same scope. Possible values are 0 and 1.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

local_meas	no	Specify whether local measurements definitions are allowed. When set to ignore, the simulator allows local measurements definitions. Depending on the value that is set, the simulator displays warning messages for the local measurements definitions or does not display any message. When set to warning, the simulator does not allow and ignores the local measurement definitions with a warning message. Possible values are no and yes.
uwiprecision	double	Specify the uwi raw file precision. Possible values are double and float.
ms_part_report	0	Depth of the ms partition report, default is 0, disabled..
tmi_fast_core_data	0	Speed up TMI SOA/Age performance by fast TMI coredata evaluation..
thermalmap	%C:r.raw	Output raw data file name.
ose_random_mod	0	Use OSE Random Method. Possible values are 0 and 1.
etmiIdsMod	0	Switch to select Ids or total DC drain current for TMI core data ids.
osc_output_port		Output raw data file name.
print_mode	probe	Only works for XPS-S mode. When value is probe, XPS-S will convert .print to .probe. Generate netlist.print file when value is print.. Possible values are probe and print.
passfail_reverse_logic	no	Pass fail bisection treats NaN as PASS, while not NaN value as FAIL. Possible values are no and yes.
ms_vpni_start	0.0	MS mode start time to have accurate currents. enable vpn current feedback after this time..

## Spectre Circuit Simulator Reference

### Analysis Statements

---

ms_vpni_stop	0.0	MS mode stop time to have accurate currents. disable vpn current feedback after this time..
diagnose_dump_to pnum	5	Top number to dump single device netlist.
ignorevaref	yes	Flag to remove Verilog-A signals from the computation of references for convergence tolerances, if it is enabled, simulation accuracy will be improved, but transient runtime may be increased. Possible values are no and yes.
ignoredgref	yes	Flag to exclude dangling nodes from quantity calculation. Possible values are no and yes.
minr_mfactor	scaled	whether minr will be scaled. Possible values are scaled and ignore.
print_section	no	If yes, output sections used in simulations. If no, do not output sections.. Possible values are no and yes.
rmdgccs	no	If yes, dangling controlled current source will be removed.. Possible values are no and yes.
sortinstance	no	Sort devices of the netlist. Possible values are no and yes.
checktoppinconn	no	Check if the top level subckts have dangling terminals. Possible values are no, warning and error.
hsfe	no	If yes, the same as +hsfe in command line.. Possible values are no, yes and always.
macrocommand	no	If yes, the same as +mac in command line.. Possible values are no, yes and csfe.
keeprc_conn_term cur	yes	If yes, it functions as the SFE_KEEPRC_CONN_TERMCUR environment variable is set. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

lkmatrix	yes	When enabled, converts inductors and mutual inductors to single instance of rclk_matrix. Possible values are no and yes.
skipinvalidalter	no	If yes, the alter statement with an invalid object is ignored.. Possible values are no and yes.
detect_path_acc	1	Select a speed method for detecting critical path, possible values are 1 and 2..
mtm_primitive	[...]	Specify the primitive names which can be worked with +mac.
inlinesubcktop	device	Specifies whether inline subcircuit operating point should be saved as inline device operating point or subcircuit operating point. Default is device.. Possible values are device and subckt.
bjt504ver	504.12	mextram504 avalanche noise version control..
check_ahdl_mode	1	This option specifies how to handle AHDL in dyn_dcpth, dyn_floatdcpth, dyn_float_tran_stat. When set to 1, all AHDL are considered conducting. When set to 2, only elastic branches are considered conducting. Possible values are 1 and 2.
spfr2iprobe	0	Value for resistors to be replaced by iprobe.
maxwildcardicreport	0	Limit the number of wildcard ic/nodeset report to log file.
bsimcmg_bug_fix	0	Control bsimcmg fix, 1 for only disable dits fix.
vlog_search_parameter	yes	Search parameter in parent for vlog case.. Possible values are no and yes.
expr_reltol	1e-12	Relative tolerance for relational operators used in expression within the netlist.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

hbtmodel	hbt	Specify to use ucsd_hbt or hbt. Possible values are hbt and ucsd_hbt.
variation_overwrite	no	support define variation with same name in a block and overwrite previous one.. Possible values are no and yes.
meastype	no	If set to yes, allow independent from/to for trig and targ. Possible values are no and yes.
probe_depth_zero	none	all : will ignore level=0 or effectively set level=INT_MAX; none : will remove the probe statement since level=0 cannot find any names (default).. Possible values are all and none.
sqlldb_violating_ratio	no	Print violating ratio value in sqlldb file. Possible values are no and yes.
dcmethod	auto	DC simulation gmethod and dptran_gmethod setting; auto: automatically switch gmethod to resolve convergence difficulties; both: set gmethod=both and dptran_gmethod=both; std: set gmethod=dev and dptran_gmethod=node. Possible values are auto, both and std.
dcpath_bulk_current	no	If set to yes, bulk current through MOSFET is traced, in dyn_dcpath (Transient based and Leakage based approaches) and dyn_floatdcpath (Transient based and Leakage based approaches) checks. If set to no, bulk current is not traced. This option is supported only in Spectre, Spectre APS and Spectre X. Possible values are no and yes.
dc_reverse_swp	no	Reverse the sweep order in DC sweep.. Possible values are no and yes.
lprobe_vlth	1.0	for logic probe ,it specifies the voltage threshold for the logic 0 (zero) state..

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>lprobe_vhth</code>	<code>1.0</code>	for logic probe ,it specifies the voltage threshold for the logic 1 (one) state..
<code>pode_inst_names</code>	<code>[...]</code>	Specify the names that a PODE device can contain.
<code>pode_preserve_inst_names</code>	<code>[...]</code>	Specify the names of the PODEs to be preserved.
<code>pode_model_names</code>	<code>[...]</code>	Specify the names that a PODE model can contain.
<code>simkit_gmin_scaled</code>	<code>no</code>	If yes, gmin is scaled by mult in simkit models. Possible values are no and yes.
<code>expr_abstol</code>	<code>0.0</code>	Absolute tolerance for relational operators used in expression within the netlist.
<code>sci_lib</code>		The shared library containing a C model using the SpectreFX C-model interface will be loaded.
<code>rel_sfe_opt</code>	<code>no</code>	Apply pre-sim optimization for reliability analysis. Possible values are no and yes.
<code>outputmatchexcludeway</code>	<code>otheritem</code>	Assume 2 wildcard probe with except, assume net1 match wildcard2 and also match except for wildcard2, otheritem means go on trying the other wildcard .probe. break means not match other wildcard .probe. Possible values are otheritem and break.
<code>pdkmacrosubckt</code>	<code>no</code>	If yes, spectre will detect instance whose next upcoming instance's name is current instance name + "_macro" and master name of such instance should be in the list provided by TSMC or user defined subckt with tmimacro=yes. Then, for any current measurement towards such instance, spectre will map the next upcoming instance's current to current instance. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

tmimacro	no	If user defined subckt with tmimacro=yes, then we consider such subckt as TSMC provided macro subckt that used in option pdkmacrosubckt. Possible values are no and yes.
enable_dc_sweep_output_info	no	Enable op point info output for every dc sweep points. Possible values are no and yes.
do_const_check	yes	Enable all constant parameter check in pre-simulation stage. Possible values are no and yes.
dcic	yes	Enable ic set in dc analysis. Possible values are no and yes.
mpbusdelimiter	none	mapping bus delimiter if set. Possible values are none, angle and square.
local_tnom_primitive	0	By default tnom is only applied locally to elements having a model definition inside the local scope, but not to primitive elements. The option local_tnom_primitive=yes will enable tnom to also be applied to all primitive instances inside the local scope..
dc_fix_zero_diagonal	yes	Fix matrix zero diagonal issue in DC simulation. Possible values are no and yes.
ac_fix_zero_diagonal	no	Fix matrix zero diagonal issue in AC simulation. Possible values are no and yes.
bht_max_limit_temperature	600.0	Set the maximum limit for the iteration of the temperature value for BHT model..
allowmz	yes	Specify whether the data of bit vsource can have 'm' or 'z'. When it is set to 'no', simulator will error out if data contain 'm' or 'z'.. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

rangecheckseverity	error	The severity given when Model or Inst parameter override the range. error, msgerror for the basic Model/Inst parameter is out of range, and the simulation to stop; warning, msgwarning for the basic Model/Inst parameter is out of range, and continue the simulation.. Possible values are error and warning.
negative_resistor_size_clamp	yes	If yes, clamp negative leff/weff to 0. Possible values are no and yes.
separatealiasmodel		Specify primitive name to request simulator disable some optimizations to CMI instance so that sfeGetOriginalModelID(cmiInst) can get alias model name. It only used by model developer in special situation..
dspf_subckt_without_instance_call	ignore	A flag that decides whether error out, or give warning, or ignore when a subckt from DSPF is not instantiated.. Possible values are ignore, warning and error.
cabscut	- infinity F	Capacitors with $C \leq \text{abs}(\text{ccut})$ are replaced by open. cabscut can additionally be applied to variable capacitors using the option 'short_cut_var_elem=yes'.
rmdg_veriloga	no	If yes, dangling verilog-A instances will be removed.. Possible values are no and normal.
createbinsofmodelgroup	active	only create model data for activated binning model, create data for all binning cards if set to all.. Possible values are active and all.
dspf_use_hier_delimiter	dspf	A flag that decides whether to use netlist hier delimiter to replace DIVIDER of DSPF.. Possible values are schematic and dspf.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>override_spice_modelgroup</code>	<code>no</code>	Control Spectre's behavior when handling model bin definition whose model group's name has been specified before, if no, the new model bins. Possible values are no and yes.
<code>mdlpostfindsignalname</code>	<code>strict</code>	flag to decide to find a signal by name or store all signal names into a container then use the container to find name. Possible values are strict and ambiguous.
<code>mc_correlate_truncmode</code>	<code>correlatefirst</code>	Determines whether to calculate truncation firstly or to calculate correlation firstly for Monte Carlo statistical variables.. Possible values are truncatefirst and correlatefirst.

Important considerations for `currents` and `useprobes` options

Adding probes to circuits that are sensitive to numerical noise might affect the solution. In such cases, accurate solution may be obtained by reducing the value of `reltol`.

The following devices always use probes to save currents (even with `useprobes=no`): port, delay, switch, hbt, transformer, core, winding, fourier, d2a, a2d, a2ao, and a2ai.

`senstype` parameter

When `senstype` is set to `partial`, the sensitivity being calculated is the partial derivative of a differentiable output variable  $F$  with respect to a design parameter  $p$ :

$$D(F \text{ w.r.t. } p) = \frac{dF}{dp}$$

This definition is not scale free. When `senstype` is set to `normalized`, the sensitivity being calculated is the normalized sensitivity

$$S(F \text{ w.r.t. } p) = \frac{d \ln F}{d \ln p} = \frac{p}{F} \frac{dF}{dp} = \frac{p}{F} D(F \text{ w.r.t. } p)$$



## Spectre Circuit Simulator Reference

### Analysis Statements

---

When either F or p take a zero value, the above normalized definition no longer provides a useful measure in this case, the following two semi-normalized sensitivities are used:

$$S(F \text{ w.r.t. } p) = \frac{dF}{d \ln p} = p \frac{dF}{dp} = p D(F \text{ w.r.t. } p) \quad \text{if } F \neq 0$$

And:

$$S(F \text{ w.r.t. } p) = \frac{d \ln F}{dp} = \frac{1}{F} \frac{dF}{dp} = \frac{1}{F} D(F \text{ w.r.t. } p) \quad \text{if } p \neq 0$$

When both F and p are zero, partial sensitivity is used.

topcheck parameter:

When topcheck=full, the topology check is performed and gmin is inserted between isolated nodes and ground. A heuristic topology check is also performed to find nodes that may be isolated due to the numerical nature of the circuit. For example, nodes isolated by reverse biased diodes in MOSFETS.

Use topcheck=fixall to attach gmin to all types of isolated nodes. Including the ones detected by the heuristic topology check.

When topcheck=min, the topology check is performed and gmin is inserted between isolated nodes and ground. A heuristic topology check is not performed.

When topcheck=no, the topology check is not performed.

topcheck=errmin (topcheck=errfull) is similar to topcheck=min (topcheck=full) but the simulation will stop if floating nodes are found.

Important considerations for using multithreading:

Multithreading is only available for devices evaluation for BSIM3v3 and BSIM4. Multithreading does not work with table model. If there is an instance of a primitive using table model, multithreading is not applied to all instances of that primitive.

Multithreading can be turned on/off using the command-line option, environment variable 'SPECTRE\_DEFAULTS' setting, or by using the 'multithread' parameter in the 'options' statement from the input file. The command-line option takes priority over the

## Spectre Circuit Simulator Reference

### Analysis Statements

---

'SPECTRE\_DEFAULTS' environment variable setting. In addition, the latter takes priority over the setting in the netlist options statement.

Using multithreading on circuits that are sensitive to numerical noise might affect the solution. The solution should still be within acceptable tolerance specified by the tolerance parameters in the Spectre input file. Because the order of evaluation of devices is different for each multithreading run of the same simulation, this could lead to different round off error in the computation. It is possible that the same result may not be reproducible when multithreading is used.

Multithreading works best when the following capabilities are not used: useprobes=yes, save-current/SOA/alarm for multithreaded devices.

When using device multithreading on hyperthreading enabled system:

Allows one threads for each physical processor.

Because the device evaluation is almost exclusively floating point computation and each physical processor still has one floating point unit, each can handle one device evaluation at a time. Allowing additional thread(s) for device evaluation on the same physical processor will not have any benefit.

On a multi-processor system with hyperthreading enabled, device multithreading would allow an extra thread for each additional physical processors. Multithreading performance not only depends on the simulator but also on how well the operating system manages multiple threads and multiple processes.

gmethod parameter

The parameter controls how conductance is stamped in the homotopy methods (other than dptran).

If gmethod=node the conductance is added from node to ground. In case of gmethod=dev the conductance is stamped in the devices.

dptran\_gmethod parameter

This parameter controls how conductance is stamped in the dptran (homotopy) method. See gmethod for more information on how this option affects circuits.

auto\_minductor parameter

When this parameter is set to yes, the simulator automatically calculates the missing second-order coupling by multiplying the two first-order coefficients. This calculation is only an estimation and may not be correct for many geometries.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

For example, consider two mutual inductors K1 and K2:

```
K1 mutual_inductor coupling=.65 ind1=L1 ind2=L2
```

```
K2 mutual_inductor coupling=.65 ind1=L2 ind2=L3
```

In this example, Spectre automatically inserts the coupling between L1 and L3, if missing, and the coupling co-efficient is  $0.65 \times 0.65 = 0.4225$ .

\* This option can be applied locally to a subckt by defining it inside a subckt definition or by using it with a subckt or inst scope:

```
subckt chip1 ( in out )
```

```
  scopedOptions options tnom=27 scale=0.1
```

```
  .....
```

```
ends chip1
```

```
subckt chip2 ( in out )
```

```
  scopedOptions options tnom=25 scale=0.2
```

```
  .....
```

```
ends chip2
```

```
o1 options scale=1.2 subckt=chip1
```

```
o2 options scale=1.5 subckt=chip2
```

```
o3 options scale=1.2 inst=pll
```

```
o4 options scale=1.5 inst=receiver
```

## Periodic AC Analysis (pac)

### Description

The periodic AC (PAC) analysis is used to compute transfer functions for circuits that exhibit frequency translation. Such circuits include mixers, switched-capacitor filters, samplers, phase-locked loops, and the like. PAC is a small-signal analysis similar to AC 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 transfer-functions that include frequency translation, which is not the case when linearizing about a DC operating point because linear time-invariant circuits do not exhibit frequency translation. In addition, the frequency of the sinusoidal stimulus is not constrained by the period of the large periodic solution.

Computing the small-signal response 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 part of the PSS analysis, the periodically time-varying representation of the circuit is computed and saved for later use. The second step is to apply the small stimulus to the periodically varying linear representation to compute the small signal response. This is done using the PAC analysis. A PAC analysis cannot be used alone, it must follow a PSS analysis. However, any number of periodic small-signal analyses such as PAC, PSP, PXF, and PNoise, can follow a PSS analysis.

Modulated small signal measurements are possible by using Analog Design Environment (ADE) environment. The `modulated` option for PAC and the other modulated parameters are set by the Analog Design Environment (ADE). PAC analyses with this option produces results that could have limited use outside such an environment. Direct Plot is configured to analyze these results and combine several wave forms to measure AM and PM response due to single sideband or modulated stimuli. For details, see the SpectreRF User Guide.

Unlike AC analysis, PAC analysis can print the time - domain simulation results by specifying the `outputperiod` parameter. In addition, unlike other analyses in Spectre, this analysis can only sweep frequency.

Unlike other analyses in Spectre, this analysis can only sweep frequency.

### Syntax

```
Name ... pac parameter=value ...
```

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### 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.
valuesfile		Name of the file containing the sweep values.
sweeptype	unspecified	Specifies if the sweep frequency range is the absolute frequency of input or if it is relative to the port harmonics. When the unspecified value is used, Spectre RF sweeps using relative when autonomous simulation is performed or when the analysis is PSP; for other cases Spectre RF sweeps the absolute value of the input. Possible values are absolute, relative and unspecified.
relharmnum	1	Harmonic to which relative frequency sweep should be referenced.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Sampled analysis parameters

<code>ptvtype</code>	<code>timeaveraged</code>	Specifies whether the PTV analysis will be traditional or sampled under certain conditions. Possible values are <code>timeaveraged</code> and <code>sampled</code> .
<code>sampleprobe</code>		The crossing event at this port triggers the sampled small signal computation.
<code>thresholdvalue</code>	<code>0</code>	Sampled measurement is done when the signal crosses this value.
<code>crossingdirection</code>	<code>all</code>	Specifies the transitions for which sampling must be done. Possible values are <code>all</code> , <code>rise</code> , <code>fall</code> and <code>ignore</code> .
<code>maxsamples</code>	<code>16</code>	Maximum number of sampled events to be processed during the sampled analysis.
<code>extrasampletimepoints</code>	<code>[...]</code>	Additional time points for sampled PTV analysis.

#### Output parameters

<code>sidebands</code>	<code>[...]</code>	Array of relevant sidebands for the analysis.
<code>maxsideband</code>	<code>7</code>	An alternative to the <code>sidebands</code> array specification, which automatically generates the array: <code>[-maxsideband ... 0 ... +maxsideband]</code> . For the shooting analysis, the default value is 7. For HB small signal analysis, the default value is the <code>harms/maxharms</code> setting in the HB large signal analysis. It is ignored in HB small signal when it is larger than the <code>harms/maxharms</code> value of large signal.
<code>freqaxis</code>		Specifies whether the results should be printed as per the input frequency, the output frequency, or the absolute value of the output frequency. Default is <code>absout</code> . Possible values are <code>absout</code> , <code>out</code> and <code>in</code> .

## Spectre Circuit Simulator Reference

### Analysis Statements

---

save		Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, none and nooutput.
nestlvl		Levels of subcircuits to output.
outputperiod	0.0 (no output)	Time-domain output period. The time-domain small-signal response is computed for the period specified, rounded to the nearest integer multiple of the <code>pss</code> period.

#### Convergence parameters

tolerance		Tolerance for linear solver. The default value is 1.0e-9 for shooting-based solver and 1.0e-4 for harmonic balance-based solver.
relativeTol		Relative tolerance for harmonic balance-based linear solver. Default value is 1.0e-2.
gear_order	2	Gear order used for small-signal integration.
solver	turbo	Solver type. Possible values are std, turbo, std_hh and turbo_hh.
oscsolver	turbo	Oscillator solver type. It is recommended that you use ira for huge circuits. Possible values are std, turbo, ira and direct.
resgmrescycle	short	Restarts GMRES cycle. Possible values are instant, short, long, recycleinstant, recycleshort, recyclelong and custom.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>hbprecond_solver</code>	<code>autoset</code>	Select a linear solver for the GMRES preconditioner. Default is <code>autoset</code> . With <code>autoset</code> , the simulator will automatically select the appropriate preconditioner. The preconditioner affects the rate of convergence of the linear matrix solver used in periodic small-signal analysis. When <code>autoset</code> is selected, the simulator may decide to switch to a different preconditioner after the analysis begins. When that happens, the simulator may issue a warning instructing you to choose a different preconditioner during subsequent runs. Although not required, choosing a different preconditioner according to the simulator's instructions may speed up subsequent analyses. Possible values are <code>basicsolver</code> , <code>blocksolver</code> , <code>autoset</code> , <code>blockdense</code> , <code>blocksolver2</code> and <code>directsolver</code> .
<code>lowmem</code>	<code>0</code>	Harmonic balance low memory mode; Possible values are 0, 1, or number ( $\geq 10$ ). The default value is '0', the low memory mode is turned off; if '1' is set, the standard low memory mode is turned on; If a number no less than 10 is set, Spectre interprets the value as the memory requested in GigaBytes..
<code>krylov_size</code>	<code>200</code>	This parameter is used to set maximum iteration count of the linear matrix solver used in periodic small-signal analysis. After reaching $1.25 \times \text{krylov\_size}$ iterations, the iteration is forced to terminate because of the poor rate of convergence. Increase the <code>krylov_size</code> if the simulation reports insufficient norm reduction errors in GMRES.
<code>osc_version</code>	<code>dts</code>	Specifies the method to use in small signal analysis for autonomous circuit. Possible values are <code>floquet</code> , <code>augmented</code> and <code>dts</code> .
<code>osc_accuracy</code>	<code>2</code>	Accuracy control in small signal analysis for autonomous circuit when <code>osc_version=dts</code> . The higher this value, the more iterations GMRES solver will take. Maximum effective value is 5.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>freqdivide</code>	<code>1</code>	Large signal frequency division. Used for oscillator circuit with divider in when <code>osc_version=dtb</code> for shooting engine.
-------------------------	----------------	---

#### Annotation parameters

<code>annotate</code>	<code>sweep</code>	Degree of annotation. Possible values are <code>no</code> , <code>title</code> , <code>sweep</code> , <code>status</code> , <code>steps</code> and <code>detailed_hb</code> .
-----------------------	--------------------	---

<code>title</code>		Analysis title.
--------------------	--	-----------------

#### Modulation conversion parameters

<code>modulated</code>	<code>no</code>	Compute transfer functions/conversion between modulated sources and outputs. Possible values are <code>single</code> , <code>first</code> , <code>second</code> and <code>no</code> .
------------------------	-----------------	---

<code>inmodharmnum</code>	<code>1</code>	Harmonic value for the input source modulation.
---------------------------	----------------	---

<code>outmodharmvec</code>	<code>[...]</code>	Harmonic list for the output modulations.
----------------------------	--------------------	---

<code>moduppersideband</code>	<code>1</code>	Index of the upper sideband included in the modulation of an output for PAC and HBAC, or an input for PXF.
-------------------------------	----------------	--

<code>modsource</code>		Refer the output noise to this component.
------------------------	--	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Rapid distortion analysis parameters

perturbation	linear	The type of PAC analysis. The default is linear for normal PAC analysis. im2ds stands for im2 distortion summary and ds stands for distortion summary. Triple beat analysis is not supported for shooting engine, PSS must be run before PAC with 'flexbalance=yes' or 'harmonicbalance=yes'. Value multiple_beat is deprecated and will be removed in the future: use triple_beat instead. Possible values are linear, ds, ip3, ip2, im2ds, multiple_beat and triple_beat.
flin_out	0 Hz	Frequency of linear output signal.
fim_out	0 Hz	Frequency of IM output signal.
out1	NULL	Output signal 1.
out2	NULL	Output signal 2.
contriblist	NULL	Array of device names for distortion summary. When contriblist=[""], distortion from each non-linear device is calculated.
maxharm_nonlin	4	Maximum harmonics of input signal frequency induced by non-linear effect.
rfmag	0	RF source magnitude.
rfdbm	0	RF source dBm.
rf1_src	NULL	Array of RF1 source names for IP3/IP2/IM2.
rf2_src	NULL	Array of RF2 source names for IP3/IP2/IM2.
rf_src	NULL	Array of RF source names for triple beat analysis, triple beat is not supported in shooting engine, PSS must be run before PAC with 'flexbalance=yes' or 'harmonicbalance=yes'.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>freqs</code>	NULL	Array of RF source frequencies for triple beat analysis, triple beat is not supported in shooting engine, PSS must be run before PAC with 'flexbalance=yes' or 'harmonicbalance=yes'.
<code>rfampls</code>	NULL	RF source amplitudes; the units are dBm for ports, Voltage for v-sources and Ampere for i-sources.

You can select the set of periodic small-signal output frequencies of interest by setting either the `maxsideband` or the `sidebands` parameters. For a given set of  $n$  integer numbers representing sidebands  $K_1, K_2, \dots, K_n$ , the output frequency at each sideband is computed as  $f(\text{out}) = f(\text{in}) + K_i * \text{fund}(\text{pss})$ , where  $f(\text{in})$  represent the (possibly swept) input frequency, and  $\text{fund}(\text{pss})$  represents the fundamental frequency used in the corresponding PSS analysis. Therefore, when analyzing a down-converting mixer, while sweeping the RF input frequency, the most relevant sideband for IF output is  $K_i = -1$ . When simulating an up-converting mixer, while sweeping IF input frequency, the most relevant sideband for RF output is  $K_i = 1$ . By setting the `maxsideband` value to  $K_{\text{max}}$ , all  $2 * K_{\text{max}} + 1$  sidebands from  $-K_{\text{max}}$  to  $+K_{\text{max}}$  are generated.

The number of requested sidebands does not change substantially the simulation time. However, the `maxacfreq` of the corresponding PSS analysis should be set to guarantee that  $| \max\{f(\text{out})\} |$  is less than `maxacfreq`; otherwise, the computed solution might be contaminated by aliasing effects. The PAC simulation is not executed for  $| f(\text{in}) |$  greater than `maxacfreq`. Diagnostic messages are printed for those extreme cases, indicating how `maxacfreq` should be set in the PSS analysis. In majority of the simulations, however, this is not an issue, because `maxacfreq` is never allowed to be smaller than 40x the PSS fundamental.

With PAC, the frequency of the stimulus and of the response are usually different (this is an important area in which PAC differs from AC). The `freqaxis` parameter is used to specify whether the results should be output versus the input frequency (`in`), the output frequency (`out`), or the absolute value of the output frequency (`absout`).

You can define sweep limits by specifying the end points or by providing the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. In addition, You can specify a step size parameter (`step`, `lin`, `log`, or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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 values that the sweep parameter should take by using the `values` parameter. If you specify 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.

## Periodic Noise Analysis (pnoise)

### Description

The Periodic Noise, or PNoise analysis is similar to the conventional noise analysis, except that it includes frequency conversion effects. Hence it is useful for predicting the noise behavior of mixers, switched-capacitor filters, and other periodically driven circuits. It is particularly useful for predicting the phase noise of autonomous circuits, such as oscillators.

PNoise analysis linearizes the circuit about the periodic operating point computed in the prerequisite PSS analysis. It is the periodically time-varying nature of the linearized circuit that accounts for the frequency conversion. In addition, the affect of a periodically time-varying bias point on the noise generated by the various components in the circuit is also included.

The time-average of the noise at the output of the circuit is computed in the form of 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 the input-referred noise or noise figure is desired, specify the input source by using the `iprobe` parameter. For input-referred noise, use `vsource` or `isource` as the input probe; for noise figure, use a `port` as the probe. Currently, only a `vsource`, an `isource`, or a `port` may be used as an input probe. If the input source is noisy, as is a `port`, the noise analysis computes 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 its `noisetemp` must be set to 16.85C (290K). 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, 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.

The reference sideband (`refsideband`) specifies which conversion gain is used when computing input-referred noise, noise factor, and noise figure. The reference sideband specifies the input frequency relative to the output frequency with:

$$lf(input) = lf(out) + refsideband * fund(pss)$$

Use `refsideband=0` when the input and output of the circuit are at the same frequency (such as with amplifiers and filters). When `refsideband` differs from 0, the single side-band noise figure is computed.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

The noise analysis always computes the total noise at the output, which includes contributions from the input source and the output load. The amount of the output noise that is attributable to each noise source in the circuit is also computed and output individually. If the input source is identified (using iprobe) and is a vsource or isource, the input-referred noise is computed, which includes the noise from the input source itself. Finally, if the input source is identified (using iprobe) and is noisy, as is the case with ports, the noise factor and noise figure are computed. Therefore if:

No = total output noise

Ns = noise at the output due to the input probe (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

NI = 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,

$$IRN = \sqrt{No^2/G^2}$$

$$F = (No^2 - NI^2)/Ns^2$$

$$NF = 10 \cdot \log_{10}(F)$$

$$Fdsb = (No^2 - NI^2)/(Ns^2 + Nsi^2)$$

$$NFdsb = 10 \cdot \log_{10}(Fdsb)$$

## Spectre Circuit Simulator Reference

### Analysis Statements

---

$$F_{\text{ieee}} = (N_o^2 - N^2 - N_{so}^2) / N_s^2$$

$$NF_{\text{ieee}} = 10 \cdot \log_{10}(F_{\text{ieee}}).$$

When the results are output,  $N_o$  is named `out`,  $IRN$  is named `in`,  $G$  is named `gain`,  $F$ ,  $NF$ ,  $F_{dsb}$ ,  $NF_{dsb}$ ,  $F_{\text{ieee}}$ , and  $NF_{\text{ieee}}$  are named `F`, `NF`, `Fdsb`, `NFdsb`, `Fieee`, and `NFieee`, respectively.

In a phase noise analysis for an oscillator, the line width, which is also known as the corner frequency, is defined as either the full width at half maximum (FWHM), or as twice the half power (-3dB) width (HW). In the absence of  $1/f$  noise and ignoring any noise floor, the phase noise spectrum satisfies the Lorentzian equation:

$$L(f) = (1/\pi) * [\pi * c * f_{osc}^2] / [(\pi * c * f_{osc}^2)^2 + f^2],$$

Where ' $c$ ' is a constant that defines the phase noise characteristics of the oscillator, ' $f_{osc}$ ' is the fundamental frequency of the oscillator, and ' $f$ ' is the offset frequency of the oscillator. Therefore:

$$\text{line width} := \text{FWHM} = 2 * \text{HW} = 2 * \pi * c * f_{osc}^2.$$

Unlike other analyses in Spectre, this analysis can only sweep frequency.

### Syntax

Name [p] [n] ... pnoise parameter=value ...

The optional terminals (p and n) specify the output of the circuit. If you do not specify the terminals, you must specify the output with a probe component.

### Parameters

Sweep interval parameters

<code>start</code>	0	Start sweep limit.
<code>stop</code>		Stop sweep limit.
<code>center</code>		Center of sweep.
<code>span</code>	0	Sweep limit span.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

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.
valuesfile		Name of the file containing the sweep values.
sweepstype	unspecified	Specifies if the sweep frequency range is the absolute frequency of input or if it is relative to the port harmonics. When the unspecified value is used, Spectre RF sweeps using relative when autonomous simulation is performed or when the analysis is PSP; for other cases Spectre RF sweeps the absolute value of the input. Possible values are absolute, relative and unspecified.
relharmnum	1	Harmonic to which relative frequency sweep should be referenced.
stophalfund	no	If yes, automatically choose stop frequency of pnoise/hbnoise to be half of the fundamental frequency obtained from pss/hb analysis. Only works for noisetype = sampled/pmjitter/timedomain. Possible values are no and yes.

#### Probe parameters

oprobe	Compute total noise at the output defined by this component.
iprobe	Refer the output noise to this component.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

refsideband		Conversion gain associated with this sideband; is used when computing input-referred noise or noise figure.
-------------	--	---

#### Sampled analysis parameters

thresholdvalue	0	Sampled measurement is done when the signal crosses this value.
crossingdirection	all	Specifies the transitions for which sampling must be done. Possible values are all, rise, fall and ignore.
maxsamples	16	Maximum number of sampled events to be processed during the sampled analysis.
sampleratio	1	The ratio between sampled frequency and fund frequency (sampled frequency/fund frequency)..
externalsourcedata		Name of PXF analysis that provides information to compute the contribution of external jitter sources.
measurement	NULL	Specifies the jitter event list that will be measured.

#### Output parameters

noisetype	timeaverage	Specifies computation of time-averaged or time-sampled noise information. Possible values are timeaverage, correlations, timedomain, pmjitter and sampled.
-----------	-------------	--



## Spectre Circuit Simulator Reference

### Analysis Statements

---

maxsideband	7	In shooting pnoise, the parameter determines the maximum sideband that is included when computing noise, that is either up-converted or down-converted to the output by the periodic drive signal. In HB pnoise, the parameter determines the size of the small signal system when HB pnoise is performed. This parameter is critical for the accuracy of HB pnoise analysis. Using a small maxsideband may cause accuracy loss. The default value for the shooting pnoise is 7. For the HB pnoise, the default is the <code>harms/maxharms</code> setting in the HB large signal analysis.
sidebands	[...]	Array of relevant sidebands for the analysis.
save		Signals to output. The option 'save' specifies the signals to be saved in the result. 'allpub' saves all signals at all levels of hierarchy in the schematic, including the internal signals of device models. 'all' works like 'allpub'. 'lvl' saves all signals through the level of hierarchy set in 'nestlvl' option. 'lvlpub' works like 'lvl'. 'selected' is not recommended to use here. Possible values are all, lvl, allpub, lvlpub and selected.
nestlvl		Levels of subcircuits to output.
maxcycles	0	Maximum cycle correlation frequency included when computing noise, that is either up-converted or down-converted to the output by the periodic drive signal.
noiseskipcount	-1	Calculate time-domain noise on only one of every noiseskipcount time points. When < 0, the parameter is ignored. When >=0, the simulator uses this parameter and ignores numberofpoints.
noisetimepoints	[...]	Additional time points for time-domain noise analysis.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

numberofpoints	5	Number of time points of interest in the period where time domain PSD is calculated. Simulator divides the period evenly into N segments (N=numberofpoints) and calculates time domain PSD on the starting time point of each segment. When < 0, the parameter is ignored.
saveallsidebands	no	Save noise contributors by sideband. Possible values are no and yes.
separatenoise	no	Separate noise into sources and transfer functions. Possible values are no and yes.
cyclo2txtfile	no	Output cyclo-stationary noise to text file as input source of next stage. Possible values are no and yes.
noiseout	usb	Specify noise output. You can set a vector like noiseout=[usb am pm]. And all are using single sideband(SSB) convention, half of the total power. Possible values are usb, lsb, am and pm.

#### Convergence parameters

tolerance		Tolerance for linear solver. The default value is 1.0e-9 for shooting-based solver and 1.0e-4 for harmonic balance-based solver.
relativeTol		Relative tolerance for harmonic balance-based linear solver. Default value is 1.0e-2.
gear_order	2	Gear order used for small-signal integration.
solver	turbo	Solver type. Possible values are std, turbo, std_hh and turbo_hh.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>oscsolver</code>	<code>turbo</code>	Oscillator solver type. It is recommended that you use <code>ira</code> for huge circuits. Possible values are <code>std</code> , <code>turbo</code> , <code>ira</code> and <code>direct</code> .
<code>resgmrescycle</code>	<code>short</code>	Restarts GMRES cycle. Possible values are <code>instant</code> , <code>short</code> , <code>long</code> , <code>recycleinstant</code> , <code>recycleshort</code> , <code>recyclelong</code> and <code>custom</code> .
<code>hbprecond_solver</code>	<code>autoset</code>	Select a linear solver for the GMRES preconditioner. Default is <code>autoset</code> . With <code>autoset</code> , the simulator will automatically select the appropriate preconditioner. The preconditioner affects the rate of convergence of the linear matrix solver used in periodic small-signal analysis. When <code>autoset</code> is selected, the simulator may decide to switch to a different preconditioner after the analysis begins. When that happens, the simulator may issue a warning instructing you to choose a different preconditioner during subsequent runs. Although not required, choosing a different preconditioner according to the simulator's instructions may speed up subsequent analyses. Possible values are <code>basicsolver</code> , <code>blocksolver</code> , <code>autoset</code> , <code>blockdense</code> , <code>blocksolver2</code> and <code>directsolver</code> .
<code>lowmem</code>	<code>0</code>	Harmonic balance low memory mode; Possible values are <code>0</code> , <code>1</code> , or number ( $\geq 10$ ). The default value is <code>'0'</code> , the low memory mode is turned off; if <code>'1'</code> is set, the standard low memory mode is turned on; If a number no less than 10 is set, Spectre interprets the value as the memory requested in GigaBytes..
<code>ppv</code>	<code>no</code>	If set to <code>yes</code> , save the oscillator PPV after performing noise analysis. Possible values are <code>no</code> and <code>yes</code> .
<code>ppvfile</code>		File to which the PPV of oscillator is written.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

augmented	yes	This parameter will be removed in future release soon. The parameter 'noiseout' is recommended. If set to <code>yes</code> , the frequency-aware PPV method is used to calculate the total noise of the oscillator; if set to <code>pmonly</code> , only the PM part of the oscillator noise is calculated; if set to <code>amonly</code> , only the AM part of the oscillator noise is calculated. The output of AM/PM noise is using double sideband convention. Possible values are <code>no</code> , <code>yes</code> , <code>pmonly</code> and <code>amonly</code> .
lorentzian	cornerfreqonly	This option determines if the Lorentzian plot is used in the oscillator noise analysis. 'lorentzian=yes' is only valid for 'noisetype=timeaverage'. Possible values are <code>no</code> , <code>cornerfreqonly</code> and <code>yes</code> .
pnoisemethod	default	This option selects the shooting pnoise method. Possible values are <code>default</code> and <code>fullspectrum</code> .
krylov_size	200	This parameter is used to set maximum iteration count of the linear matrix solver used in periodic small-signal analysis. After reaching <code>1.25*krylov_size</code> iterations, the iteration is forced to terminate because of the poor rate of convergence. Increase the <code>krylov_size</code> if the simulation reports insufficient norm reduction errors in GMRES.
osc_version	dts	Specifies the method to use in small signal analysis for autonomous circuit. Possible values are <code>floquet</code> , <code>augmented</code> and <code>dts</code> .
osc_accuracy	2	Accuracy control in small signal analysis for autonomous circuit when <code>osc_version=dts</code> . The higher this value, the more iterations GMRES solver will take. Maximum effective value is 5.
freqdivide	1	Large signal frequency division. Used for oscillator circuit with divider in when <code>osc_version=dts</code> for shooting engine.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Annotation parameters

<code>annotate</code>	<code>sweep</code>	Degree of annotation. Possible values are no, title, sweep, status, steps and detailed_hb.
<code>title</code>		Analysis title.
<code>oscmacrogene</code>	<code>no</code>	If set to yes, harmonic balance steady-state and phase noise data are saved. Applies only if there is pm noise out. Possible values are no and yes.
<code>oscmacrosave</code>		File to which harmonic balance steady-state solution and phase noise data are to be written.

In practice, noise can mix with each of the harmonics of the periodic drive signal applied in the PSS analysis and end up at the output frequency. However, the PNoise analysis includes only the noise that mixes with a finite set of harmonics that are typically specified using the `maxsideband` parameter; however, in special circumstances, the harmonics may be specified with the `sidebands` parameter. If  $K_i$  represents sideband  $i$ , then

$$f(\text{noise\_source}) = f(\text{out}) + K_i * \text{fund}(\text{pss})$$

The `maxsideband` parameter specifies the maximum  $|K_i|$  included in the PNoise calculation. Therefore, noise at frequencies less than  $f(\text{out}) - \text{maxsideband} * \text{fund}(\text{pss})$  and greater than  $f(\text{out}) + \text{maxsideband} * \text{fund}(\text{pss})$  are ignored. If selected sidebands are specified using the `sidebands` parameter, then only those specified are included in the calculation. When specifying the sidebands ensure that you include a sideband that contributes significant noise to the output; otherwise, the results will be erroneous.

The number of requested sidebands does not change the simulation time substantially. However, the `maxacfreq` of the corresponding PSS analysis should be set to guarantee that  $|\text{Imax}\{f(\text{noise\_source})\}|$  is less than `maxacfreq`; otherwise, the computed solution might be contaminated by aliasing effects. The PNoise simulation is not executed for  $|f(\text{out})|$  greater than `maxacfreq`. Diagnostic messages are printed for those extreme cases, indicating which `maxacfreq` should be set in the PSS analysis. In majority of simulations, however, this is not an issue, because `maxacfreq` is never allowed to be smaller than 40x the PSS fundamental.

Phase Noise measurements are possible by using the Analog Design Environment (ADE). Two pnoise analyses are pre-configured for this simulation and most of the analysis named `mod2` has limited use outside of the Analog Design Environment (ADE) environment. Direct

## Spectre Circuit Simulator Reference

### Analysis Statements

---

Plot is configured to analyze these results and combine several wave forms to measure AM and PM components of the output noise. For details, see the Spectre RF User Guide.

You can define sweep limits by specifying the end points or by providing the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. In addition, You can specify a step size parameter (`step`, `lin`, `log`, or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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 values that the sweep parameter should take by using the `values` parameter. If you specify 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.

## Periodic S-Parameter Analysis (psp)

### Description

The periodic SP (PSP) analysis is used to compute scattering and noise parameters for n-port circuits that exhibit frequency translation, such as mixers. It is a small-signal analysis like SP analysis, except, as in PAC and PXF, the circuit is first linearized about a quasi-periodically 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 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. Similar to PNoise, but unlike SP, the noise features of the PSP 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 part of the PSS analysis, the periodically time-varying representation of the circuit is computed and saved for later use. The second step is to apply small-signal excitations to compute the n-port S-parameters and noise parameters. This is done using PSP analysis. PSP analysis cannot be used independently, it must follow a PSS analysis. However, any number of periodic small-signal analyses such as PAC, PSP, PXF, PNoise, can follow a PSS analysis.

Unlike other analyses in Spectre, this analysis can only sweep frequency.

### Syntax

```
Name psp parameter=value ...
```

### Parameters

Sweep interval parameters

start	0	Start sweep limit.
stop		Stop sweep limit.
center		Center of sweep.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

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.
valuesfile		Name of the file containing the sweep values.
sweepstype	unspecified	Specifies if the sweep frequency range is the absolute frequency of input or if it is relative to the port harmonics. When the unspecified value is used, Spectre RF sweeps using relative when autonomous simulation is performed or when the analysis is PSP; for other cases Spectre RF sweeps the absolute value of the input. Possible values are absolute, relative and unspecified.

#### Port parameters

ports	[...]	List of active ports. Ports are numbered in the specified order. For noise figure computation, the input is considered as port 1 and the output as port 2.
portharmsvec	[...]	List of harmonics that are active on specified list of ports. Must have a one-to-one correspondence with the 'ports' vector.
harmsvec	[...]	List of harmonics, in addition to the ones associated with specific ports by portharmsvec, that are active.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Output parameters

freqaxis		Specifies whether the results should be printed as per the input frequency, the output frequency, or the absolute value of the input frequency. Default is in. Possible values are absin, in and out.
file		Output file name. It just supports S-parameters in PSP, HBSP, and QPSP analysis.
datafmt	spectre	Data format of the S-parameter output file. Possible values are spectre, touchstone and touchstone2.
datatype	realimag	Data type of the S-parameter output file. Possible values are realimag, magphase and dbphase.
noisedata	no	Should noise data be saved to the S-parameter output file; if yes, in what format. Possible values are no and twoport.

#### Noise parameters

donoise	yes	Perform noise analysis. If oprobe is specified as a valid port, this parameter is set to yes, and a detailed noise output is generated. Possible values are no and yes.
---------	-----	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Probe parameters

maxsideband	7	In shooting pnoise, the parameter determines the maximum sideband that is included when computing noise, that is either up-converted or down-converted to the output by the periodic drive signal. In HB pnoise, the parameter determines the size of the small signal system when HB pnoise is performed. This parameter is critical for the accuracy of HB pnoise analysis. Using a small maxsideband may cause accuracy loss. The default value for the shooting pnoise is 7. For the HB pnoise, the default is the <code>harms/maxharms</code> setting in the HB large signal analysis.
-------------	---	---

#### Convergence parameters

tolerance		Tolerance for linear solver. The default value is 1.0e-9 for shooting-based solver and 1.0e-4 for harmonic balance-based solver.
relativeTol		Relative tolerance for harmonic balance-based linear solver. Default value is 1.0e-2.
gear_order	2	Gear order used for small-signal integration.
solver	turbo	Solver type. Possible values are std, turbo, std_hh and turbo_hh.
oscsolver	turbo	Oscillator solver type. It is recommended that you use ira for huge circuits. Possible values are std, turbo, ira and direct.
resgmrescycle	short	Restarts GMRES cycle. Possible values are instant, short, long, recycleinstant, recycleshort, recyclelong and custom.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>osc_version</code>	<code>mts</code>	Specifies the method to use in small signal analysis for autonomous circuit. Possible values are floquet, augmented and mts.
<code>osc_accuracy</code>	<code>2</code>	Accuracy control in small signal analysis for autonomous circuit when <code>osc_version=mts</code> . The higher this value, the more iterations GMRES solver will take. Maximum effective value is 5.
<code>hbprecond_solver</code>	<code>autoset</code>	Select a linear solver for the GMRES preconditioner. Default is <code>autoset</code> . With <code>autoset</code> , the simulator will automatically select the appropriate preconditioner. The preconditioner affects the rate of convergence of the linear matrix solver used in periodic small-signal analysis. When <code>autoset</code> is selected, the simulator may decide to switch to a different preconditioner after the analysis begins. When that happens, the simulator may issue a warning instructing you to choose a different preconditioner during subsequent runs. Although not required, choosing a different preconditioner according to the simulator's instructions may speed up subsequent analyses. Possible values are <code>basicsolver</code> , <code>blocksolver</code> , <code>autoset</code> , <code>blockdense</code> , <code>blocksolver2</code> and <code>directsolver</code> .
<code>lowmem</code>	<code>0</code>	Harmonic balance low memory mode; Possible values are 0, 1, or number ( $\geq 10$ ). The default value is '0', the low memory mode is turned off; if '1' is set, the standard low memory mode is turned on; If a number no less than 10 is set, Spectre interprets the value as the memory requested in GigaBytes..
<code>krylov_size</code>	<code>200</code>	This parameter is used to set maximum iteration count of the linear matrix solver used in periodic small-signal analysis. After reaching $1.25 \times \text{krylov\_size}$ iterations, the iteration is forced to terminate because of the poor rate of convergence. Increase the <code>krylov_size</code> if the simulation reports insufficient norm reduction errors in GMRES.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Annotation parameters

<code>annotate</code>	<code>sweep</code>	Degree of annotation. Possible values are no, title, sweep, status, steps and detailed_hb.
<code>title</code>		Analysis title.
<code>sprobes</code>	<code>[...]</code>	List of s-probes.
<code>sprobeharmsvec</code>	<code>[...]</code>	List of probe harmonics. Set 2 harmonics for each probe..

To specify the PSP analysis, the port and port harmonic relations must be specified. You can select the ports of interest by setting the `port` parameter, and the set of periodic small-signal output frequencies of interest by setting `portharmsvec` or the `harmsvec` parameters. For a given set of  $n$  integer numbers representing the harmonics  $K_1, K_2, \dots, K_n$ , the scattering parameters at each port are computed at the frequencies  $f(\text{scattered}) = f(\text{rel}) + K_i * \text{fund}(\text{pss})$ , where  $f(\text{rel})$  represents the relative frequency of a signal incident on a port,  $f(\text{scattered})$  represents the frequency to which the relevant scattering parameter represents the conversion, and  $\text{fund}(\text{pss})$  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 is  $K_i = 0$ . Hence, we can associate  $K_2 = 1$  with the IF port and  $K_1 = 0$  with the RF port.  $S_{21}$  represents the transmission of signal from the RF to IF and  $S_{11}$  the reflection of signal back to the RF port. If the signal was in the lower sideband, a choice of  $K_1 = -1$  would be more appropriate.

Either `portharmsvec` or `harmsvec` can be used to specify the harmonics of interest. If `portharmsvec` is given, the harmonics must be in one-to-one correspondence with the ports, with each harmonic associated with a single port. If harmonics are specified in the optional `harmsvec` parameter, all possible frequency-translating scattering parameters associated with the specified harmonics are computed.

With PSP, the frequency of the input and of the response are usually different (this is an important area 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 `sweep_type` parameters behave differently in PSP than in PAC and PXF.

The `sweep_type` parameter controls the way the frequencies in the PSP analysis are swept. A `relative sweep` is a sweep relative to the analysis harmonics (not the PSS fundamental),

## Spectre Circuit Simulator Reference

### Analysis Statements

---

and an `absolute` sweep is a sweep of the absolute input source frequency. For example, with a PSS fundamental of 100MHz, `portharmsvec` set to [9 1] to examine a down-converting mixer, `sweep_type=relative`, and a sweep range of `f(rel)=0->50MHz`, 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 `sweep_type=absolute` and sweeping the frequency from 900->950MHz would calculate the same quantities, because `f(abs)=900->950MHz`, and `f(rel) = f(abs) - K1 * fund(pss) = 0->50MHz`, where, `K1=9` and `fund(pss) = 100MHz`.

Usually, it is not necessary to sweep frequency in PSP over more than one fundamental PSS period.

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`). If `freqaxis` is `absin`, the S-parameters at negative frequencies are taken conjugate and output at corresponding positive frequencies.

Unlike in PAC/PXF/PNoise, increasing the number of requested ports and harmonics increases the simulation time substantially.

To ensure accurate results in PSP, the `maxacfreq` of the corresponding PSS analysis should be set to guarantee that  $|\max\{f(\text{scattered})\}|$  is less than `maxacfreq`; otherwise, the computed solution might be contaminated by aliasing effects.

PSP analysis also computes noise figures, equivalent noise sources, and noise parameters. The noise computation, which is skipped only when `donoise=no`, requires additional simulation time. If:

No = total output noise at frequency `f`

Ns = noise at the output due to the input probe (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

NI = noise at the output due to the output probe (the load)

IRN = input referred noise

G = gain of the circuit

F = noise factor (single side band)

## Spectre Circuit Simulator Reference

### Analysis Statements

---

NF = noise figure (single side band)

Fdsb = double sideband noise factor

NFdsb = double sideband noise figure

Fieee = IEEE single sideband noise factor

NFieee = IEEE single sideband noise figure

Then:

$$\text{IRN} = \sqrt{\text{No}^2/\text{G}^2}$$
$$\text{F} = (\text{No}^2 - \text{NI}^2)/\text{Ns}^2$$
$$\text{NF} = 10 \cdot \log_{10}(\text{F})$$
$$\text{Fdsb} = (\text{No}^2 - \text{NI}^2)/(\text{Ns}^2 + \text{Nsi}^2)$$
$$\text{NFdsb} = 10 \cdot \log_{10}(\text{Fdsb})$$
$$\text{Fieee} = (\text{No}^2 - \text{NI}^2 - \text{Nso}^2)/\text{Ns}^2$$
$$\text{NFieee} = 10 \cdot \log_{10}(\text{Fieee}).$$

When the results are output, IRN is named `in`, G is named `gain`, F, NF, Fdsb, NFdsb, Fieee, and NFieee are named `F`, `NF`, `Fdsb`, `NFdsb`, `Fieee`, and `NFieee`, respectively. Note that the gain computed by PSP is the voltage gain from the actual circuit input to the circuit output, and not the gain from the internal port voltage source to the output.

To ensure accurate noise calculations, the `maxsideband` or `sidebands` parameters must be set to include the relevant noise folding effects. `maxsideband` is only relevant to the noise computation features of PSP.

You can define sweep limits by specifying the end points or by providing the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. In addition, You can specify a step size parameter (`step`, `lin`, `log`, or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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 values that the sweep parameter should take by using the `values` parameter. If you specify 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.

## Periodic Steady-State Analysis (pss)

### Description

This analysis computes the periodic steady-state (PSS) response of a circuit by using harmonic balance (in the frequency domain) or shooting (in the time domain). The simulation time of PSS analysis is independent of the time-constants of the circuit. In addition, PSS analysis determines the periodic operating point for the circuit. The periodic operating point can then be used during a periodic time-varying small-signal analysis, such as PAC, PXF, PNOISE, PSP, or PSTB.

Generally, harmonic balance (HB) is very efficient in simulating weak non-linear circuits while shooting is more suitable for highly non-linear circuits with sharply rising and falling signals. HB is also advantageous over shooting in handling frequency dependent components, such as delay, transmission line, and S-parameter data.

PSS analysis can handle both autonomous (non-driven) and driven (non-autonomous) circuits. Autonomous circuits, even though they are not driven by a time-varying stimulus, generate non-constant waveforms. Driven circuits require some time-varying stimulus to generate a time-varying response. The most common example of an autonomous circuit is an oscillator. Common driven circuits include amplifiers, filters, and mixers. When PSS is applied to autonomous circuits, it requires you to specify a pair of nodes, `p` and `n`. This is how PSS analysis determines whether it is being applied to an autonomous or a driven circuit. If the pair of nodes is supplied, PSS assumes the circuit is autonomous; if not, the circuit is assumed to be driven.

With driven circuits, specify the analysis period, or its corresponding fundamental frequency `fund`. The `period` must be an integer multiple of the period of the drive signal or signals. Autonomous circuits have no drive signal, and the actual period of oscillation is not known precisely in advance. Instead, you specify an estimate of the oscillation period and PSS analysis computes the precise period along with the periodic solution waveforms.

PSS analysis consists of two phases, an initial transient phase, which initializes the circuit, and the shooting or harmonic balance phase, which computes the periodic steady-state solution. The transient phase consists of three intervals. The first interval starts at `tstart`, which is normally 0, and continues through the onset of periodicity `tonset` for the independent sources. The onset of periodicity, which is automatically generated, is the minimum time for which all sources are periodic. The second interval is an optional user-specified stabilization interval whose length is `tstab`. The final interval length is `period` for driven circuits, or four times `period` for autonomous circuits. This interval has a special use for the autonomous PSS analysis, that is, the PSS analysis monitors the waveforms in the

## Spectre Circuit Simulator Reference

### Analysis Statements

---

circuit and develops a better estimate of the oscillation period. After the initial transient phase is complete, the shooting or HB phase begins. In this phase, the circuit is iteratively solved using Newton method to find the periodic steady-state solution (and the period when applied to autonomous circuits).

### Syntax

```
Name [p] [n] pss parameter=value ...
```

### Parameters

#### Simulation interval parameters

period	(s)	Steady state analysis period (or its estimate for autonomous circuits).
fund	(Hz)	Alternative to period specification. Steady state analysis fundamental frequency (or its estimate for autonomous circuits).
autofund	no	If the value is yes, the program ignores period/fund value and calculates the fundamental frequency automatically from source information. Possible values are no and yes.
harms	9 for shooting, 10 for HB	For shooting, it is the number of solution harmonics to output when outputtype=freq or all; for HB, it directly determines the solution dimension to be solved and impacts the accuracy and convergence of the simulation.
autoharms	no	Activates automatic harmonic number calculation in harmonic balance. Applies only if tstab>0 or if autotstab=yes. If a steady-state is reached, Spectre does a spectrum analysis to calculate the optimal number of harmonics for HB. The minimum number of harmonics is specified by maxharms. If steady-state is not reached to sufficient tolerance, autoharms may be disabled. Possible values are no and yes.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

tstab	0.0 s	Extra stabilization time after the onset of periodicity for independent sources.
autotstab	no	Activates the automatic initial transient (tstab) in harmonic balance and PSS shooting. If set to yes, the simulator decides whether to run tstab and for how long. Typically, the initial length of tstab is 50 periods; however, it could be longer depending on the type of circuit and its behavior. If steady-state is reached (or nearly reached), tstab terminates early. Possible values are no and yes.
envlp_autotstab	no	Activates the automatic initial envlp (tstabenvlp) in PSS shooting. If set to yes, the simulator decides how long to run tstabenvlp. Typically, the initial length of tstabenvlp is 50 periods; however, it could be longer depending on the type of circuit and its behavior. If steady-state is reached (or nearly reached), tstabenvlp terminates early. Possible values are no and yes.
autosteady	no	Activates the automatic steady state detection during initial transient (tstab) in harmonic balance and PSS shooting. When steady state is reached (or nearly reached), tstab terminates early. This parameter applies only when tstab>0 or when autotstab=yes. autosteady=no 0 turns this feature off; autosteady=yes 1 activates this feature; autosteady=2 runs autosteady with lower steady state tolerance than autosteady=1. autosteady=2 may help pss convergence but with higher tstab costs. Possible values are 0, 1, 2, no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>tstabenvlp_autos</code> <code>no</code> <code>steady</code>		<p>Activates the automatic steady state detection during <code>tstabenvlp</code> in PSS shooting. When steady state is reached, <code>tstabenvlp</code> terminates early. This parameter applies only when <code>tstabenvlpstop&gt;0</code> or when <code>envlp_autotstab=yes</code>.</p> <p><code>tstabenvlp_autosteady=no 0</code> turns this feature off; <code>tstabenvlp_autosteady=yes 1</code> activates this feature; <code>tstabenvlp_autosteady=2</code> runs autosteady with lower steady state tolerance than <code>tstabenvlp_autosteady=1</code>. <code>tstabenvlp_autosteady=2</code> may help pss convergence but with higher <code>tstabenvlp</code> costs. Possible values are 0, 1, 2, no and yes.</p>
--	--	---

<code>tstart</code>	<code>0.0 s</code>	Initial transient analysis start time.
<code>tstabenvlpstop</code>	<code>0.0 s</code>	Determines the stop time of envelope <code>tstab</code> .
<code>tstabenvlpstep</code>	<code>0.0 s</code>	Determines the step size of envelope <code>tstab</code> .

#### Time-step parameters

<code>transres</code>	<code>1e-</code> <code>9*stop s</code>	<p>Transition resolution. The transient analysis attempts to stop at corners of input waveforms (for example, corners of rising/falling edge of a pulse). If such events occur within a time less than <code>transres</code>, the analysis combines the events into one and forces only one time point. The rest of the steps are determined by error control. This may lead to loss of detail.</p>
<code>maxstep</code>	<code>(s)</code>	Maximum time step. The default is derived from <code>errpreset</code> .
<code>maxacfreq</code>		Maximum frequency requested in a subsequent periodic small-signal analysis. The default is derived from <code>errpreset</code> and <code>harms</code> . This parameter is valid only for shooting.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>step</code>	<code>0.001*period</code>	Minimum time step that would be used solely to maintain the aesthetics of the results. This parameter is valid only for shooting.
-------------------	---------------------------	---

#### Initial-condition parameters

<code>ic</code>	<code>all</code>	The value to be used to set the initial condition. Possible values are <code>dc</code> , <code>dev</code> and <code>all</code> .
-----------------	------------------	--

<code>skipdc</code>	<code>no</code>	If set to yes, there is no DC analysis for initial transient. Possible values are <code>no</code> , <code>yes</code> and <code>sigrampup</code> .
---------------------	-----------------	---

<code>readic</code>		File that contains initial condition.
---------------------	--	---------------------------------------

<code>oscic</code>	<code>default</code>	Oscillator IC method. It determines how the starting values for the oscillator are calculated. <code>oscic=lin</code> provides you an accurate initial value, but it takes time; <code>oscic=lin_ic</code> is not recommended, which is the older version of <code>oscic=lin</code> for shooting analysis for backward compatibility; <code>oscic=fastic</code> is fast, but it is less accurate. ' <code>oscic=skip</code> ' directly uses the frequency provided by you as the initial guess frequency. It is only for the two-tier method. Possible values are <code>default</code> , <code>lin</code> , <code>lin_ic</code> , <code>fastic</code> and <code>skip</code> .
--------------------	----------------------	---

<code>useprevic</code>	<code>no</code>	If set to yes or ns, use the converged initial condition from previous analysis as <code>ic</code> or <code>ns</code> . Possible values are <code>no</code> , <code>yes</code> and <code>ns</code> .
------------------------	-----------------	--

#### Convergence parameters

<code>readns</code>		File that contains an estimate of the initial transient solution.
---------------------	--	---

<code>cmin</code>	<code>0 F</code>	Minimum capacitance from each node to ground.
-------------------	------------------	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Output parameters

harmsvec	[...]	Array of desired harmonics. An alternative form of <code>harms</code> that allows selection of specific harmonics. This parameter is valid only for shooting.
outputtype	all''	Output type. Possible values are all, time and freq.
save		Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, none and nooutput.
nestlvl		Levels of subcircuits to output.
oppoint	no	Should operating point information be computed for initial timestep; if yes, where should it be printed (screen or file). Possible values are no, screen, logfile and rawfile.
skipstart	0 s	The time to start skipping output data.
skipstop	stop s	The time to stop skipping output data.
skipcount	1	Save only one of every skipcount points.
strobeperiod	0 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.
saveinit	no	If set to yes, the waveforms for the initial transient before steady state are saved. Possible values are no and yes.
harmoutputlist	[...]	Array of harmonics indices for hb output.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### State-file parameters

<code>write</code>		File to which initial transient solution (before steady-state) is written.
<code>writefinal</code>		File to which final transient solution in steady-state is written. This parameter is now valid only for shooting.
<code>swapfile</code>		Temporary file to hold steady-state information. It tells Spectre to use a regular file, rather than virtual memory to hold the periodic operating point. Use this option if Spectre complains about not having enough memory to complete the analysis. This parameter is now valid only for shooting.
<code>writepss</code>		File to which the converged steady-state solution is written. The file of shooting and HB cannot be mutually reused.
<code>readpss</code>		File from which a previously converged steady-state solution is read. For shooting method, PSS loads the solution and checks the residue of the circuit equations only. The solution is re-used if the residue is satisfactory. Otherwise, the solution is re-converged using the finite difference method. The results from shooting QPSS cannot be used in HB QPSS analysis and vice-versa.
<code>checkpss</code>	<code>yes</code>	If set to yes, the previous PSS results (from <code>readpss</code> file) are checked and PSS+MIC is rerun if any condition has changed. If set to no, the simulator assumes that nothing has changed and uses the solution from the file without checking and running PSS+MIC again. This parameter is now valid only for shooting. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Integration method parameters

<code>method</code>		Integration method. The default is derived from <code>errpreset</code> . This parameter is valid only for shooting. Possible values are <code>euler</code> , <code>trap</code> , <code>traponly</code> , <code>gear2</code> and <code>gear2only</code> .
<code>tstabmethod</code>		Integration method used in stabilization time. The default is <code>traponly</code> for autonomous circuits, or is derived from <code>errpreset</code> for driven circuits. Possible values are <code>euler</code> , <code>trap</code> , <code>traponly</code> , <code>gear2</code> and <code>gear2only</code> .

#### Accuracy parameters

<code>errpreset</code>		Selects a reasonable collection of parameter settings. Possible values are <code>liberal</code> , <code>moderate</code> and <code>conservative</code> .
<code>relref</code>		Reference used for the relative convergence criteria. The default is derived from <code>errpreset</code> . Possible values are <code>pointlocal</code> , <code>alllocal</code> , <code>sigglobal</code> and <code>allglobal</code> .
<code>lteratio</code>		Ratio used to compute LTE tolerances from Newton tolerance. The default is derived from <code>errpreset</code> .
<code>lteminstep</code>	<code>0.0 s</code>	Local truncation error is ignored if the step size is less than <code>lteminstep</code> .
<code>steadyratio</code>		Ratio used to compute steady-state tolerances from LTE tolerance. The default is derived from <code>errpreset</code> .
<code>maxperiods</code>		Maximum number of iterations allowed before convergence is reached in shooting or harmonic balance Newton iteration. For PSS and QPSS, the default is 20 for driven circuits, and 50 for oscillators; For HB, the default is 100.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

max_innerkrylov_size	20	The maximum iteration number allowed in inner gmres solver in two-level gmres linear solver. The default value is 12 for large signal and 20 for small signal analysis.
max_outerkrylov_size	4	The maximum iteration number allowed in inner gmres solver in two-level gmres linear solver. The default value is 4.
itres	1e-4 for shooting, 0.9 for HB	Controls the residual for iterative solution of linearized matrix equation at each Newton iteration. Tightening the parameter can help with the Newton convergence, but does not affect the result accuracy. The value should be between [0, 1]. Default value for shooting APS flow is 1e-3.
inexactNewton	no	Inexact Newton method. Possible values are no and yes.
finitediff		Enable finite difference method refinement for driven circuits after shooting method. Possible values are no, yes and refine.
highorder		Perform a high-order refinement after low-order convergence. The Multi-Interval Chebyshev polynomial spectral algorithm is used. This parameter is only valid for shooting. Possible values are no and yes.
psaratio	1	Ratio used to compute high-order polynomial spectral accuracy from Newton tolerance. This parameter is only valid for shooting.
maxorder		The maximum order of the Chebyshev polynomials used in waveform approximation. Possible values are from 2 to 16. Default value is 16 for driven circuits and 12 for autonomous circuits. This parameter is valid only for shooting.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

fullpssvec		Use the full vector containing solutions at all PSS time steps in the linear solver. The default is derived from the size of the equation and the property of the PSS time steps. This parameter is only valid for shooting. Possible values are no and yes.
------------	--	--

fdharms	10	Number of harmonics considered for distributed (frequency-domain) components, such as nport, delay, mtline, and delayed controlled sources. This parameter is valid only for shooting and for those components for which 'Fmax' parameter of neither model nor instance is set.
---------	----	---

#### Harmonic Balance parameters

harmonicbalance	no	Use Harmonic Balance engine instead of time-domain shooting. Possible values are no and yes.
-----------------	----	--

flexbalance	no	Same parameter as <code>harmonicbalance</code> . Possible values are no and yes.
-------------	----	--

pinnode		Node to pin during autonomous HB simulation.
---------	--	--

pinnodeminus		Second node to pin during autonomous HB simulation. Needed only when differential nodes exist in oscillator.
--------------	--	--

pinnode rank		Harmonic rank to pin during autonomous HB simulation.
--------------	--	---

pinnodemag		This parameter gives an estimate of the magnitude of the pin node voltage. Default value is 0.01.
------------	--	---

oversamplefactor	1	Oversample device evaluations.
------------------	---	--------------------------------

oversample	[...]	Array of oversample factors for each tone. This parameter overrides <code>oversamplefactor</code> .
------------	-------	---



## Spectre Circuit Simulator Reference

### Analysis Statements

---

oscmetho		Osc Newton method for autonomous HB. Possible values are onetier and twotier.
hbmhomotopy	tone	Name of Harmonic Balance homotopy selection methods. Possible values are tstab, source, gsweep, tone, inctone, aggregation and steptone.
sweepic	none	IC extrapolation method in sweep HB analysis. Possible values are none, linear and log.
gstart	1.e-7	Start conductance for hbmhomotopy of gsweep.
gstop	1.e-12	Stop conductance for hbmhomotopy of gsweep.
glog	5	Number of steps, log sweep for hbmhomotopy of gsweep.
backtracking	yes	This parameter is used to activate the backtracking utility of Newton's method. The default is yes. Possible values are no, yes and forced.
excludeconv with BK	yes	Possible values are no and yes.
krylov_size	10	The minimum iteration count of the linear matrix solver used in HB large-signal analysis. After reaching krylov_size iterations, the iteration is forced to terminate because of the poor rate of convergence. Increase krylov_size if the simulation reports insufficient norm reduction errors in GMRES.
hbmprcond_solver	basicsolver	Choose a linear solver for the GMRES preconditioner. Possible values are basicsolver, blocksolver, autoset, blockdense, blocksolver2 and directsolver.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

lowmem	0	Harmonic balance low memory mode; Possible values are 0, 1, or number ( $\geq 10$ ). The default value is '0', the low memory mode is turned off; if '1' is set, the standard low memory mode is turned on; If a number no less than 10 is set, Spectre interprets the value as the memory requested in GigaBytes.
--------	---	--

#### Annotation parameters

annotate	sweep	Degree of annotation. Possible values are no, title, sweep, status, estimated, steps, iters, detailed, rejects, alliters, detailed_hb and internal_hb.
annotateic	no	Degree of annotation for initial condition. Possible values are no, title, sweep, status, estimated, steps, iters, detailed, rejects and alliters.
title		Analysis title.

#### Newton parameters

maxiters	5	Maximum number of iterations per time step.
restart	no	Restart the DC/PSS solution if set to 'yes'; if set to 'no', reuse the previous solution as an initial guess; if set to 'firstonly', restart if it is the first point of sweep (supported only in HB). The default value is 'no' for HB and 'yes' for shooting. Possible values are no, yes and firstonly.

#### Circuit age

circuitage	(Years)	Stress time. Age of the circuit used to simulate hot-electron degradation of MOSFET and BSIM circuits.
------------	---------	--

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Tstab save/restart parameters

saveperiod		Save the tran analysis periodically on the simulation time.
saveperiodhistory	no	Maintains the history of saved files. If yes, maintains all the saved files. Possible values are no and yes.
saveclock	(s)	Save the tran analysis periodically on the wall clock time. The default is 1800s for Spectre. This parameter is disabled in the APS mode by default.
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.

#### Oscillator PPV parameters

ppv	no	If set to yes, save the oscillators' perturbation projection vector (PPV), representing the oscillators' phase sensitivity to perturbations in the voltage or current at the nodes of the oscillator. Possible values are no and yes.
-----	----	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Compression parameters

xdbccompression	no	Sets the automatic gain compression analysis. In automatic gain compression analysis, Spectre automatically sweeps the input excitation until the gain, as defined by the analysis parameter xdbgain, compresses by the amount specified by the analysis parameter xdblevel. In gain compression analysis, Spectre outputs the hb solution at the calculated compression point only. Dependent analyses, such as hbnoise and hbac, are supported and calculated about the calculated compression level. Auxiliary output includes the gain and voltage/power compression curves. These outputs are available for analysis and post-processing in ADE. The possible values are yes and no. Default is no.
xdblevel	[...]	Sets the gain compression level for compression analysis. The reference point for gain compression is the small-signal gain of the circuits, or as specified by the analysis parameter xdbref. Default is 1.
xdbgain	power	Chooses between the voltage gain and transducer power gain as the target for compression point calculation. When xdbgain=power, the gain is defined as $G \text{ (dB)} = P_{\text{load}} \text{ (dBm)} - P_{\text{available}} \text{ (dBm)}$ . When xdbgain=voltage, the gain is defined as $G \text{ (dB)} = \text{dB20}( V_{\text{load}} / V_{\text{source}} )$ . In both cases, Spectre sweeps the excitation source until $\text{xdbref} - G = \text{xdblevel}$ , where the analysis parameter xdbref defines the reference level for compression calculation. Possible values are power and voltage. Default is power.
xdbref	linear	Sets the reference point for gain compression calculations. When xdbref=linear, spectre uses the small-signal gain as the reference. When xdbref=max, spectre uses the maximum observed gain as the reference. Possible values are linear and max. Default is linear.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

xdbsource		The instance name of the excitation source, which is swept automatically to reach the compression level. When xdbgain=power, the excitation source must be a port instance. When xdbgain=voltage, the excitation source must be a vsource instance.
xdbload		The instance name of the load termination. When xdbgain is power, xdbload can be a port, a resistor, or a current probe.
xdbnodep		The output terminals for voltage gain calculation when xdbgain=voltage. If either is left unspecified, the terminal is assumed to be the global ground.
xdbnoden		The output terminals for voltage gain calculation when xdbgain=voltage. If either is left unspecified, the terminal is assumed to be the global ground.
xdbrefnode		The reference node when xdbload is a current probe. The default is the ground node.
xdbharm	[ . . . ]	The Integer array which specifies the harmonic indexes of the output voltage or power component.
xdbsteps	100	The maximum number of steps for the compression point search. The simulator terminates if xdbsteps exceeds before the compression point is found. The default is 100.
xdbmax		The maximum input power (or voltage) for the compression point search. Default is 30 dBm when xdbgain=power, and 2.0 V when xdbgain=voltage.
xdbstart		The starting input power (or voltage) for the compression point search. Default is (xdbmax-70) dBm when xdbgain=power, and xdbmax/20000 when xdbgain=voltage.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>xdbtol</code>	<code>0.01</code>	Sets the tolerance for compression analysis. This tolerance is used in compression curve fitting and calculating the compression point.
<code>xdbrapid</code>	<code>no</code>	Sets the automatic gain compression analysis in rapid mode. In this mode, Spectre does not trace the compression curve and calculates only the compression point.
<code>xdbcpi</code>		Sets the estimated input-referred compression point for rapid compression analysis.
<code>backoff</code>	<code>0.0</code>	The backoff point. If defined, an additional harmonic balance analysis will be performed after the compression analysis is done. Default is 0 dBm when <code>xdbgain=power</code> , and 0 V when <code>xdbgain=voltage</code> .

#### Memory estimation parameters

<code>memoryestimate</code>	<code>no</code>	Sets the memory usage estimate for Harmonic Balance. If yes, a memory estimate is printed in the log file. You can use this memory estimate to plan the computing resources before submitting harmonic balance runs. In memory estimate mode, a short simulation is performed first, and the engine exits after printing the estimate in the log file without saving any results. If no, the simulation continues after the memory estimate is printed. Memory estimation is not recommended for simulations that require less than 500MB approximately. For PSS analysis, memory estimate mode does not apply unless <code>flexbalance=yes</code> . <code>memoryestimate=1</code> estimates the memory usage for large-signal analysis and <code>memoryestimate=2</code> estimates both large-signal analysis and small-signal simulations. Possible values are no and yes.
-----------------------------	-----------------	--

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Oscillator tuning mode parameters

tuneparam		When set, tuneparam enables the tuning mode oscillator analysis. In the tuning mode analysis, a circuit parameter is automatically varied to reach the oscillation frequency specified by the fundfreqs parameter. The tuning parameter can be a device instance parameter (as determined by the parameters tunedev) or a netlist parameter. This mode applies only to autonomous circuits (oscillators).
tunedev		Sets the instance name of a device whose parameter (identified by tuneparam) will be varied such that the circuit oscillates at the specified frequency. Applies only in tuning mode autonomous analysis. Tunedev must be used with tuneparam.
tunerange	[...]	The tuning range of the parameter identified by tuneparam. Although tunerange is not required, it can aid in convergence if set. Tunerange also can be used with 'tunestep' or 'tunelin' to decide the acceptable discrete parameter set.
tunestep		Specify the step size between two adjacent discrete tuning points. Must be used with 'tunerange'.
tunelin		Specify the numbers of discrete tuning points. Must be used with 'tunerange'.
tunevalues	[...]	Specify the values of discrete tuning points.

#### LSSP parameters

lsspports	[...]	Specifies the list of ports on which the large-signal 2-port S-parameters are calculated.
-----------	-------	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

lsspharms	[...]	Specifies the output harmonic for large-signal S-parameter calculations. The input harmonic is defined by the frequency parameters on the input port instance.
lsspfile		Identifies the file name for large-signal S-parameter output.
lsspdatafmt	touchstone	Sets the file format of the large-signal S-parameter output. Possible values are spectre and touchstone. Default is touchstone.
lsspdatype	msgphase	Sets the data format of the large-signal S-parameter output. Possible values are realimag, magphase and phase. Default is magphase.

#### Dynamic parameters

param		Name of the parameter to be updated during pss/hb analysis (tstab stage). When param=paramName and param_vec=[t1 value1 t2 value2 ... valuen tn], paramName is set to value1 at t1, value2 at t2, and so on. Pss/hb analysis uses the last value in the param_vec(valuen) to find the steady state.
paramset		Name of the dynamic parameter set.
param_vec	[...] s	The time_value points to param=name.
param_file		The file that contains the time_value points to param=name.
sub		Name of the subcircuit instance parameter specified in param=name.
mod		Name of the device model parameter specified in param=name.



**Spectre Circuit Simulator Reference**  
Analysis Statements

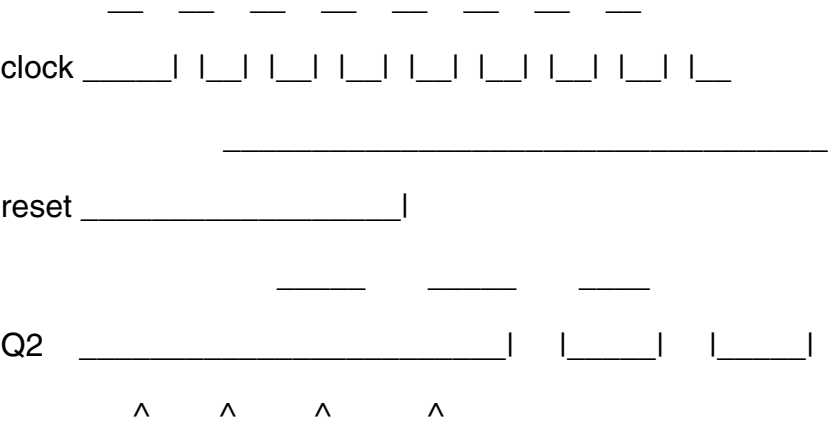
---

dev	Name of the device instance parameter specified in param=name.	
param_step	Defines the frequency of updating the dynamic parameter values. If param_step=0, it updates the parameter value at a given time point.	

Osc macro source parameters

oscmacrogene	no	If set to yes, harmonic balance steady-state solution is saved. Possible values are no and yes.
oscmacroprobe		Specify probe of which the current data is to be saved.
oscmacroout	[...]	Specify nodes of which the voltage data is to be saved.
oscmacrosave		File to which harmonic balance steady-state solution is to be written.

The initial transient analysis provides a flexible mechanism to direct the circuit to a particular steady-state solution of interest, and to avoid undesired solutions. Another use of the initial transient simulation is to help in convergence by eliminating large but fast decaying modes that are present in many circuits. For example, in case of driven circuits, consider the reset signal in the figure below.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

`tstartl tonsetl tinitl tstopl`  
`|<-tstab->|<- period->|`

In the above figure, the initial transient analysis runs from `tstart` to `tstop`. If initial transient results are relevant, you can output them by setting `saveinit` to `yes`. The steady-state results are always computed for the specified `period`, from `tinit` to `tstop`. By default, `tstart` and `tstab` are set to zero, while `tinit`, `tonset` and `tstop` are always automatically generated.

It happens in some circuits that the linearity of the relationship between the initial and final state depends on when the shooting or HB begins. Conceptually, when shooting or HB begins should not matter, as long as it is after the time when the stimuli have become periodic, because the periodic response repeats endlessly. However, in practice, starting at a good point can improve the convergence, and starting at a bad point can degrade the convergence and slow down the analysis. In general, it is best to try to avoid starting the shooting interval at a point where the circuit is undergoing strong nonlinear behavior. For example, when shooting is used to simulate switch-capacitor filters, it is best if `tinit` falls at the beginning of a clock transition, preferably a transition that follows a relatively long period of settling. If instead `tinit` occurred during a clock transition or soon after one, it is likely that the opamps will undergo slew-rate limiting at the start of the shooting interval, which will slow convergence. Switching mixers follow similar rules.

When PSS analysis simulates oscillators, either transient or linear initialization is performed to obtain an initial guess of the steady-state solution and the oscillating frequency. Two initialization methods are implemented based on transient and linear analysis, respectively. When `oscic=default` is specified, transients initialization is used and the length of the transient is specified by `tstab`. It is necessary to start the oscillator by using initial conditions, or by using a brief impulsive stimulus, just as you would if you were simulating the turn-on transient of the oscillator using transient analysis. Initial conditions would be provided for the components of the oscillators' resonator. If an impulsive stimulus is used, it should be applied so as to couple strongly into the oscillatory mode of the circuit, and poorly into any other long-lasting modes, such as those associated with bias circuitry. The Designer's Guide to Spice and Spectre [K. S. Kundert, Kluwer Academic Publishers, 1995] describes techniques for starting oscillators in some depth. When `oscic=default` is specified, `oscic=lin`, linear initialization is used. In this method, both oscillation frequency and amplitude are estimated based on linear analysis at DC solution. No impulsive stimulus or initial conditions are needed. Linear initialization is suitable for linear type of oscillators, such as LC and crystal oscillators. Note that `tstab` transient is still performed after linear initialization though it can be significantly shortened (or skipped in HB). Either way, specifying a non-zero `tstab` parameter can improve convergence.

By default, only the time-domain results are computed in shooting. If you specify either `harms` or `harmsvec`, or set `outputtype` to `freq` or `all`, the frequency-domain results will also be

## Spectre Circuit Simulator Reference

### Analysis Statements

---

computed. If frequency-domain results are requested, but the desired harmonics are not specified, its default value is 9. The time-domain output waveform generation can be inhibited by setting `outputtype` to `freq`.

The accuracy of the results does not depend on the number of harmonics that are requested, but only on the accuracy parameters, which are set in the same fashion as in the transient analysis. Besides a few new parameters, like `steadyratio` and `maxacfreq`, all the others parameters work in PSS analysis in exactly the same manner as they work on transient analysis. For HB, besides `reltol`, `abstol`, `steadyratio` and `lteratio`, the number of harmonics has the most impact on the accuracy of simulation results. When too few harmonics are used, an error occurs due to the aliasing effect. To obtain accurate results, `harms` should be big enough to cover the signal bandwidth.

Several parameters determine the accuracy of the PSS analysis. `reltol` and `abstol` control the accuracy of the discretized equation solution. These parameters determine how well charge is conserved and how accurately steady-state or equilibrium points are computed. You can set the integration errors in the computation of the circuit dynamics (such as time constants), relative to `reltol` and `abstol`, by setting the `lteratio` parameter.

For shooting, the `steadyratio` parameter adjusts the maximum allowed mismatch in node voltages or current branches from the beginning to the end of the steady-state period. For HB, the `steadyratio` parameter adjusts the maximum allowed error in the node voltages or in the current branches of the steady-state. This value is multiplied by `lteratio` and `reltol` to determine the convergence criterion. The relative convergence norm is printed along with the actual mismatch value at the end of each iteration, indicating the progress of the steady-state iteration.

For shooting, the parameter `maxperiods` controls the maximum number of shooting iterations for PSS analysis. Its default value is set to 20 for driven PSS and 50 for autonomous PSS. For HB, the parameter 'maxperiods' controls the maximum number of HB iterations for both driven and autonomous HB analysis. Its default value is set to 100.

The `finitediff` parameter allows the use of finite difference (FD) after shooting. Usually this eliminates the above mismatch in node voltages or current branches. It can also refine the grid of time steps. In some cases, numerical error of the linear solver still introduces a mismatch. You can set `steadyratio` to a smaller value to activate a tighter tolerance for the iterative linear solver. If `finitediff` is set to `no`, FD method is turned off. If it is set to `yes`, PSS applies FD method and tries to improve the beginning small time steps, if necessary. If it is set to `refine`, PSS applies FD method and tries to refine the time steps. When the simulation uses second-order method, uniform second-order gear is used. `finitediff` is automatically changed from `no` to `yes` when `readpss` and `writepss` are specified to re-use PSS results.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

The `maxacfreq` parameter is used to automatically adjust the `maxstep` and reduce errors due to aliasing in frequency-domain results. By default, the `maxacfreq` is set to four times the frequency of the largest requested harmonic, but is never set to less than forty times the fundamental.

The parameter `relref` determines how the relative error is treated. The `relref` values are as follows:

`relref=pointlocal`: Compares the relative errors in quantities at each node to that node alone.

`relref=alllocal`: Compares the relative errors at each node to the largest values found for that node alone for all past time.

`relref=sigglobal`: Compares relative errors in each circuit signal to the maximum for all signals at any previous point in time.

`relref=allglobal`: Same as `relref=sigglobal`, except that it also compares the residues (KCL error) for each node to the maximum of each node's past history.

The `errpreset` parameter lets you adjust the simulator parameters to fit your needs quickly. In most cases, it should also be the only parameter you need to adjust.

Guidelines for using `errpreset` in driven circuits in shooting are as follows:

If the circuit contains only one periodic tone and you are only interested in obtaining the periodic operating point, set `errpreset` to `liberal`. This setting provides reasonably accurate result and the fastest simulation speed.

If the circuit contains more than one periodic tone and you are interested in intermodulation results, set `errpreset` to `moderate`. This setting provides accurate results.

If you want a very low noise floor in your simulation result and accuracy is your main interest, set `errpreset` to `conservative`.

The effect of `errpreset` on other parameters for driven circuits is shown in the following table.

---

Parameter defaults and estimated numerical noise floor in simulation  
result as a function of `errpreset`

---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

`errpreset`   `reltol`   `relref`   `method`   `Iteratio`   `steadyratio`   `maxstep`

-----

<code>liberal</code>	<code>1e-3</code>	<code>sigglobal</code>	<code>traponly</code>	<code>3.5</code>	<code>0.001</code>	<code>period/50</code>
----------------------	-------------------	------------------------	-----------------------	------------------	--------------------	------------------------

<code>moderate</code>	<code>1e-3</code>	<code>alllocal</code>	<code>gear2only</code>	<code>3.5</code>	<code>0.001</code>	<code>period/200</code>
-----------------------	-------------------	-----------------------	------------------------	------------------	--------------------	-------------------------

<code>conservative</code>	<code>1e-4</code>	<code>alllocal</code>	<code>gear2only</code>	<code>*</code>	<code>0.01</code>	<code>period/200</code>
---------------------------	-------------------	-----------------------	------------------------	----------------	-------------------	-------------------------

\* : `Iteratio`=10.0 for conservative `errpreset`. Only if user-specified `reltol` <= `1e-4` \* `10.0/3.5`, `Iteratio` is set to 3.5.

The new `errpreset` settings include a new default `reltol`, which is an enforced upper limit for appropriate setting. An increase of `reltol` above the default value is ignored by the simulator. You can decrease this value in the options statement. The only way to increase `reltol` is to relax `errpreset`.

Estimated numerical noise floor for a weak non-linear circuit is -70dB for liberal, -90dB for moderate, and -120dB for conservative settings. For a linear circuit, the noise floor is even lower. Multi-interval Chebyshev (MIC) is activated when you explicitly set `highorder=yes`, which drops numerical noise floor by at least 30dB. MIC falls back to the original method if it encounters difficulty converging. You can tighten `psratio` to further drop numerical noise floor.

Spectre sets the value of `maxstep` so that it cannot be larger than the value given in the table. Except for `reltol` and `maxstep`, `errpreset` does not change the value of parameters that you explicitly set. The actual values used for the PSS analysis are given in the log file. If `errpreset` is not specified in the netlist, `liberal` settings is used. For HB, only `reltol` is affected by `errpreset` and the effect is the same as that in shooting. However, `Iteratio` remains 3.5 and `steadyratio` remains 1 with all values of `errpreset`.

Guidelines for using `errpreset` in autonomous circuits are as follows:

If you want a fast simulation with reasonable accuracy, you can set `errpreset` to `liberal`.

If you have some concern for accuracy, you can set `errpreset` to `moderate`.

If accuracy is your main interest, you can set `errpreset` to `conservative`.

The effect of `errpreset` on other parameters for autonomous circuits is shown in the following table.

-----

Parameter defaults as a function of `errpreset`

## Spectre Circuit Simulator Reference

### Analysis Statements

---

```
-----
errpreset  reltol relref  method  lteratio steadyratio maxstep
-----
```

```
liberal    1e-3  sigglobal traonly  3.5    0.001    period/50
```

```
moderate   1e-4  alllocal  gear2only 3.5    0.01     period/200
```

```
conservative 1e-5  alllocal  gear2only *    0.1     period/400
```

\* : lteratio=10.0 for conservative `errpreset` by default. Only if user-specified `reltol`  $\leq 1e-5 \times 10.0 / 3.5$ , lteratio is set to 3.5.

The value of `reltol` can be decreased from default in the options statement. The only way to increase `reltol` is to relax `errpreset`. Spectre sets the value of `maxstep` so that it cannot be larger than the value given in the table. Except for `reltol` and `maxstep`, `errpreset` does not change the value of any parameters you have explicitly set. The actual values used for the PSS analysis are given in the log file. If `errpreset` is not specified in the netlist, `liberal` settings will be used. Multi-interval Chebyshev (MIC) is activated when you explicitly set `highorder=yes`, which drops numerical noise floor by at least 30dB. MIC falls back to the original method if it encounters difficulty in converging. You can tighten `psratio` to further drop numerical noise floor.

A long stabilization (by specifying a large `tstab`) can help with PSS convergence. However, it can slow down simulation. By default, in the stabilization stage, the following settings are used: `reltol=1e-3`, `maxstep=period/25`, `relref=sigglobal`, and `method=traonly`. These settings are overwritten when `maxstep`, `relref`, or `tstabmethod` are specified explicitly in `pss` statement, or `reltol` is specified explicitly in options statement.

If the circuit you are simulating has infinitely fast transitions (for example, a circuit that contains nodes with no capacitance), Spectre might have convergence problems. To avoid this, you must prevent the circuit from responding instantaneously. You can accomplish this by setting `cmin`, the minimum capacitance to ground at each node, to a physically reasonable nonzero value. This often significantly improves Spectre convergence.

You can specify the initial condition for the transient analysis by using the `ic` statement or the `ic` parameter on the capacitors and inductors. If you do not specify the initial condition, the DC solution is used as the initial condition. The `ic` parameter on the transient analysis controls the interaction of various methods of setting the initial conditions. The effects of individual settings are as follows:

`ic=dc`: All initial conditions are ignored, and the DC solution is used.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

`ic=node`: The `ic` statements are used, and the `ic` parameter on the capacitors and inductors is ignored.

`ic=dev`: The `ic` parameters on the capacitors and inductors are used, and the `ic` statements are ignored.

`ic=all`: Both `ic` statements and `ic` parameters are used, and the `ic` parameters override the `ic` statements.

If you specify an initial condition file with the `readic` parameter, initial conditions from the file are used, and any `ic` statements are ignored.

After you specify the initial conditions, Spectre computes the actual initial state of the circuit by performing a DC analysis. During this analysis, Spectre forces the initial conditions on nodes by using a voltage source in series with a resistor whose resistance is `rforce` (see `options`).

With the `ic` statement, it is possible to specify an inconsistent initial condition (one that cannot be sustained by the reactive elements). Examples of inconsistent initial conditions include setting the voltage on a node with no path of capacitors to ground, or setting the current through a branch that is not an inductor. If you initialize Spectre inconsistently, its solution jumps, that is, it changes instantly at the beginning of the simulation interval. You should avoid such changes because Spectre can have convergence problems while trying to make the jump.

You can skip DC analysis entirely by using the parameter `skipdc`. If DC analysis is skipped, the initial solution is trivial or is given in the file that you specified by using the `readic` parameter, or if the `readic` parameter is not specified, by the values specified on the `ic` statements. Device-based initial conditions are not used for `skipdc`. Nodes that you do not specify with the `ic` file or `ic` statements start at zero. You should not use this parameter unless you are generating a nodeset file for circuits that have trouble in the DC solution; it usually takes longer to follow the initial transient spikes that occur when the DC analysis is skipped than it takes to find the real DC solution. The `skipdc` parameter might also cause convergence problems in the transient analysis.

The possible settings of parameter `skipdc` and their descriptions are as follows:

`skipdc=no`: Initial solution is calculated using normal DC analysis (default).

`skipdc=yes`: Initial solution is given in the file specified by the `readic` parameter or the values specified on the `ic` statements.

`skipdc=sigrampup`: Independent source values start at 0 and ramp up to their initial values in the first phase of the simulation. The waveform production in the time-varying independent source is enabled after the rampup phase. The rampup simulation is from

## Spectre Circuit Simulator Reference

### Analysis Statements

---

`tstart` to `time=0 s`, and the main simulation is from `time=0 s` to `tstab`. If the `tstart` parameter is not specified, the default `tstart` time is set to  $-0.1 \cdot tstab$ .

Nodesets help find the DC or the initial transient solution. You can specify nodesets in the circuit description file with `nodeset` statements or in a separate file by using the `readns` parameter. When nodesets are specified, Spectre computes an initial guess of the solution by performing DC analysis, while forcing the specified values on to nodes by using a voltage source in series with a resistor whose resistance is `rforce`. Spectre then removes these voltage sources and resistors and computes the required solution from this initial guess.

Nodesets have two important uses. First, if a circuit has two or more solutions, nodesets can bias the simulator towards computing the required solution. Second, this is a convergence aid. By estimating the solution of the largest possible number of nodes, you might be able to eliminate a convergence problem or significantly speed up convergence.

When you simulate the same circuit multiple times, it is recommended that you use both `write` and `readns` parameters and assign the same file name to both parameters. DC analysis then converges quickly even if the circuit has changed since the last simulation, and the nodeset file is automatically updated.

Nodesets and initial conditions have similar implementation, but produce different effects. Initial conditions define the solution, whereas nodesets only influence it. When you simulate a circuit with a transient analysis, Spectre forms and solves a set of differential equations. Because differential equations have an infinite number of solutions, a complete set of initial conditions must be specified to identify the required solution. Any initial conditions that you do not specify are computed by the simulator to be consistent. The transient waveforms then start from initial conditions. Nodesets are usually used as a convergence aid and do not affect the final results. However, in a circuit with more than one solution, such as a latch, nodesets bias the simulator towards finding the solution closest to the nodeset values.

The `method` parameter specifies the integration method. The possible settings and their meanings are as follows:

`method=euler:` Backward-Euler is used exclusively.

`method=traponly:` Trapezoidal rule is used almost exclusively.

`method=trap:` Backward-Euler and the trapezoidal rule are used.

`method=gear2only:` Gear's second-order backward-difference method is used almost exclusively.

`method=gear2:` Backward-Euler and second-order Gear are used.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

The trapezoidal rule is usually the most efficient when you want high accuracy. This method can exhibit point-to-point ringing, but you can control this by tightening the error tolerances. For this reason, though, if you choose very loose tolerances to get a quick answer, the backward-Euler or second-order Gear will probably give better results than the trapezoidal rule. Second-order Gear and backward-Euler can make systems appear more stable than they really are. This effect is less pronounced with second-order Gear or when you request high accuracy.

Spectre provides two methods for reducing the number of output data points saved: `strobing`, based on the simulation time, and `skipping` time points, which saves only every N'th point.

The parameters `strobeperiod` and `strobedelay` control the strobing method. `strobeperiod` sets the interval between the points that you want to save, and `strobedelay` sets the offset within the period relative to `skipstart`. The simulator forces a time step on each point to be saved, so that the data is computed, and not interpolated.

The skipping method is controlled by `skipcount`. If this is set to N, only every N'th point is saved.

The parameters `skipstart` and `skipstop` apply to both data reduction methods. Before `skipstart` and after `skipstop`, Spectre saves all computed data.

With parameter 'hbmhomotopy', you can specify harmonic balance homotopy selection methods. The possible values of parameter 'hbmhomotopy' and their descriptions are as follows:

'hbmhomotopy=tstab': Simulator runs a transient analysis and generates an initial guess for harmonic balance analysis; it is recommended for nonlinear circuits or circuits with frequency dividers.

'hbmhomotopy=source': For driven circuit, the simulator ignores `tstab` and accordingly increases the source power level; for oscillators, the simulator accordingly adjusts the probe magnitude until the probe has no effect on the oscillators. It is recommended for strongly nonlinear or high Q circuits.

'hbmhomotopy=tone': This method is valid only for multi-tone circuit. The simulator first solves a single-tone circuit by turning off all the tones, except the first one, and then solves the multi-tone circuit by restoring all the tones and using the single-tone solution as its initial guess. It is recommended for multi-tone simulation with a strong first tone.

'hbmhomotopy=inctone': Simulator first solves a single tone, then turns on moderate tones incrementally till all tones are enabled. It is recommended for circuits with one strong large tone.

## **Spectre Circuit Simulator Reference**

### **Analysis Statements**

---

'hbhomotopy=gsweep': A resistor, whose conductance is  $g$ , is connected with each node, and the sweep of  $g$  is controlled by  $gstart$ ,  $gstop$ , and  $glog$ . It is recommended for circuits containing high-impedance or quasi-floating nodes.

## Periodic STB Analysis (pstb)

### Description

The periodic STB (PSTB) analysis is used to evaluate the local stability of a periodically 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.

Evaluating the stability of a periodically varying circuit is a two-step process. In the first step, the small stimulus is ignored and PSS analysis is used to compute the periodic steady-state response of the circuit to a possibly large periodic stimulus. As part of the PSS analysis, the periodically time-varying representation of the circuit is computed and saved for later use. In the second step, a probe is used to compute the loop gain of the zero sideband. The local stability can be evaluated using gain margin, phase margin, or a Nyquist plot of the loop gain. To perform PSTB analysis, a probe instance must be specified as `probe` parameter.

The loop-based algorithm requires that a `probe` be placed 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. For the time-varying property of the circuit, the loop gain at different places can be different, but all values can be used to evaluate the stability. The loop-based algorithm provides stability information for single-loop circuits and for multi-loop circuits in which a `probe` component can be placed on a critical wire to break all loops. For a typical multi-loop circuit, such a critical wire may not be available. The loop-based algorithm can be used only on individual feedback loops to ensure that they are stable.

The device based algorithm requires the `probe` be a gain instant, such as a bjt transistor or a mos transistor. The device-based algorithm evaluates the loop gain around the `probe`, which can be involved in multiloops.

Unlike other analyses in Spectre, this analysis can only sweep frequency.

### Syntax

```
Name pstb parameter=value ...
```

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Parameters

##### Sweep interval parameters

<code>start</code>	<code>0</code>	Start sweep limit.
<code>stop</code>		Stop sweep limit.
<code>center</code>		Center of sweep.
<code>span</code>	<code>0</code>	Sweep limit span.
<code>step</code>		Step size, linear sweep.
<code>lin</code>	<code>50</code>	Number of steps, linear sweep.
<code>dec</code>		Points per decade.
<code>log</code>	<code>50</code>	Number of steps, log sweep.
<code>values</code>	<code>[...]</code>	Array of sweep values.
<code>valuesfile</code>		Name of the file containing the sweep values.

##### Probe parameters

<code>probe</code>	Probe instance around which the loop gain is calculated.
<code>localgnd</code>	Node name of local ground. If not specified, the probe is referenced to global ground.

##### Output parameters

<code>save</code>	Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, none and nooutput.
-------------------	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>nestlvl</code>		Levels of subcircuits to output.
----------------------	--	----------------------------------

#### Convergence parameters

<code>tolerance</code>		Tolerance for linear solver. The default value is 1.0e-9 for shooting-based solver and 1.0e-4 for harmonic balance-based solver.
------------------------	--	--

<code>relativeTol</code>		Relative tolerance for harmonic balance-based linear solver. Default value is 1.0e-2.
--------------------------	--	---

<code>gear_order</code>	2	Gear order used for small-signal integration.
-------------------------	---	---

<code>solver</code>	turbo	Solver type. Possible values are std, turbo, std_hh and turbo_hh.
---------------------	-------	---

<code>oscsolver</code>	turbo	Oscillator solver type. It is recommended that you use ira for huge circuits. Possible values are std, turbo, ira and direct.
------------------------	-------	---

<code>resgmrescycle</code>	short	Restarts GMRES cycle. Possible values are instant, short, long, recycleinstant, recycleshort, recyclelong and custom.
----------------------------	-------	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

hbprecond_solver	autoset	Select a linear solver for the GMRES preconditioner. Default is autoset. With autoset, the simulator will automatically select the appropriate preconditioner. The preconditioner affects the rate of convergence of the linear matrix solver used in periodic small-signal analysis. When autoset is selected, the simulator may decide to switch to a different preconditioner after the analysis begins. When that happens, the simulator may issue a warning instructing you to choose a different preconditioner during subsequent runs. Although not required, choosing a different preconditioner according to the simulator's instructions may speed up subsequent analyses. Possible values are basicsolver, blocksolver, autoset, blockdense, blocksolver2 and directsolver.
lowmem	0	Harmonic balance low memory mode; Possible values are 0, 1, or number ( $\geq 10$ ). The default value is '0', the low memory mode is turned off; if '1' is set, the standard low memory mode is turned on; If a number no less than 10 is set, Spectre interprets the value as the memory requested in GigaBytes..
krylov_size	200	This parameter is used to set maximum iteration count of the linear matrix solver used in periodic small-signal analysis. After reaching $1.25 \times \text{krylov\_size}$ iterations, the iteration is forced to terminate because of the poor rate of convergence. Increase the krylov_size if the simulation reports insufficient norm reduction errors in GMRES.
osc_version	dts	Specifies the method to use in small signal analysis for autonomous circuit. Possible values are floquet, augmented and dts.
osc_accuracy	2	Accuracy control in small signal analysis for autonomous circuit when osc_version=dts. The higher this value, the more iterations GMRES solver will take. Maximum effective value is 5.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Annotation parameters

<code>annotate</code>	<code>sweep</code>	Degree of annotation. Possible values are no, title, sweep, status, steps and detailed_hb.
<code>title</code>		Analysis title.

You can define sweep limits by specifying the end points or by providing the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. In addition, You can specify a step size parameter (`step`, `lin`, `log`, or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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 values that the sweep parameter should take by using the `values` parameter. If you specify 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.

## Periodic Transfer Function Analysis (pxf)

### Description

A conventional transfer function analysis computes the transfer function from every source in the circuit to a single output. Unlike a conventional AC analysis that computes the response from a single stimulus to every node in the circuit, the Periodic Transfer Function or PXF analysis computes the transfer functions from any source at any frequency to a single output at a single frequency. Thus, like PAC analysis, PXF analysis includes frequency conversion effects.

The PXF analysis directly computes such useful quantities as conversion efficiency (transfer function from input to output at required frequency), image and sideband rejection (input to output at undesired frequency), and LO feed-through and power supply rejection (undesired input to output at all frequencies).

As with a PAC, PSP, and PNoise analyses, a PXF analysis must follow a PSS analysis.

Unlike other analyses in Spectre, this analysis can only sweep frequency.

### Syntax

```
Name [p] [n] ... pxf parameter=value ...
```

The optional terminals (p and n) specify the output of the circuit. If you do not specify the terminals, you must specify the output with a probe component.

### 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.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

lin	50	Number of steps, linear sweep.
dec		Points per decade.
log	50	Number of steps, log sweep.
values	[...]	Array of sweep values.
valuesfile		Name of the file containing the sweep values.
sweepstype	unspecified	Specifies if the sweep frequency range is the absolute frequency of input or if it is relative to the port harmonics. When the unspecified value is used, Spectre RF sweeps using relative when autonomous simulation is performed or when the analysis is PSP; for other cases Spectre RF sweeps the absolute value of the input. Possible values are absolute, relative and unspecified.
relharmnum	1	Harmonic to which relative frequency sweep should be referenced.

#### Probe parameters

probe	Compute every transfer function to this probe component.
-------	--

#### Sampled analysis parameters

ptvtype	timeaveraged	Specifies whether the PTV analysis will be traditional or sampled under certain conditions. Possible values are timeaveraged and sampled.
sampleprobe		The crossing event at this port triggers the sampled small signal computation.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

thresholdvalue	0	Sampled measurement is done when the signal crosses this value.
crossingdirection	all	Specifies the transitions for which sampling must be done. Possible values are all, rise, fall and ignore.
maxsamples	16	Maximum number of sampled events to be processed during the sampled analysis.
extrasampletimepoints	[...]	Additional time points for sampled PTV analysis.
sampleratio	1	The ratio between sampled frequency and fund frequency (sampled frequency/fund frequency)..

#### Jitter parameters

externalsources		Pairs of terminals or nodes corresponding to external jitter sources.
extcorrsources1		Pairs of terminals and nodes for the first group of correlated external jitter sources.
extcorrsources2		Pairs of terminals and nodes for the second group of correlated external jitter sources.
deterministicsources		Pairs of terminals or nodes corresponding to deterministic jitter sources.
determinsourcesfreqs		Frequency list corresponding to the external deterministic jitter sources.

#### Output parameters

stimuli	sources	Stimuli used for pxf analysis. Possible values are sources and nodes_and_terminals.
---------	---------	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

sidebands	[...]	Array of relevant sidebands for the analysis.
maxsideband	7	An alternative to the <code>sidebands</code> array specification, which automatically generates the array: [ -maxsideband ... 0 ... +maxsideband ]. For the shooting analysis, the default value is 7. For HB small signal analysis, the default value is the <code>harms/maxharms</code> setting in the HB large signal analysis. It is ignored in HB small signal when it is larger than the <code>harms/maxharms</code> value of large signal.
freqaxis		Specifies whether the results should be printed as per the input frequency, the output frequency, or the absolute value of the input frequency. The default is <code>absin</code> . Possible values are <code>absin</code> , <code>in</code> and <code>out</code> .
save		Signals to output. Possible values are <code>all</code> , <code>lvl</code> , <code>allpub</code> , <code>lvlpub</code> , <code>selected</code> , <code>none</code> and <code>nooutput</code> .
nestlvl		Levels of subcircuits to output.
Convergence parameters		
tolerance		Tolerance for linear solver. The default value is 1.0e-9 for shooting-based solver and 1.0e-4 for harmonic balance-based solver.
relativeTol		Relative tolerance for harmonic balance-based linear solver. Default value is 1.0e-2.
gear_order	2	Gear order used for small-signal integration.
solver	turbo	Solver type. Possible values are <code>std</code> , <code>turbo</code> , <code>std_hh</code> and <code>turbo_hh</code> .

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>oscsolver</code>	<code>turbo</code>	Oscillator solver type. It is recommended that you use <code>ira</code> for huge circuits. Possible values are <code>std</code> , <code>turbo</code> , <code>ira</code> and <code>direct</code> .
<code>resgmrescycle</code>	<code>short</code>	Restarts GMRES cycle. Possible values are <code>instant</code> , <code>short</code> , <code>long</code> , <code>recycleinstant</code> , <code>recycleshort</code> , <code>recyclelong</code> and <code>custom</code> .
<code>hbprecond_solver</code>	<code>autoset</code>	Select a linear solver for the GMRES preconditioner. Default is <code>autoset</code> . With <code>autoset</code> , the simulator will automatically select the appropriate preconditioner. The preconditioner affects the rate of convergence of the linear matrix solver used in periodic small-signal analysis. When <code>autoset</code> is selected, the simulator may decide to switch to a different preconditioner after the analysis begins. When that happens, the simulator may issue a warning instructing you to choose a different preconditioner during subsequent runs. Although not required, choosing a different preconditioner according to the simulator's instructions may speed up subsequent analyses. Possible values are <code>basicsolver</code> , <code>blocksolver</code> , <code>autoset</code> , <code>blockdense</code> , <code>blocksolver2</code> and <code>directsolver</code> .
<code>lowmem</code>	<code>0</code>	Harmonic balance low memory mode; Possible values are <code>0</code> , <code>1</code> , or number ( $\geq 10$ ). The default value is <code>'0'</code> , the low memory mode is turned off; if <code>'1'</code> is set, the standard low memory mode is turned on; If a number no less than 10 is set, Spectre interprets the value as the memory requested in GigaBytes..
<code>krylov_size</code>	<code>200</code>	This parameter is used to set maximum iteration count of the linear matrix solver used in periodic small-signal analysis. After reaching $1.25 \times \text{krylov\_size}$ iterations, the iteration is forced to terminate because of the poor rate of convergence. Increase the <code>krylov_size</code> if the simulation reports insufficient norm reduction errors in GMRES.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>osc_version</code>	<code>dts</code>	Specifies the method to use in small signal analysis for autonomous circuit. Possible values are floquet, augmented and dts.
<code>osc_accuracy</code>	<code>2</code>	Accuracy control in small signal analysis for autonomous circuit when <code>osc_version=dts</code> . The higher this value, the more iterations GMRES solver will take. Maximum effective value is 5.
<code>freqdivide</code>	<code>1</code>	Large signal frequency division. Used for oscillator circuit with divider in when <code>osc_version=dts</code> for shooting engine.

#### Annotation parameters

<code>annotate</code>	<code>sweep</code>	Degree of annotation. Possible values are no, title, sweep, status, steps and detailed_hb.
<code>title</code>		Analysis title.

#### Modulation conversion parameters

<code>modulated</code>	<code>no</code>	Compute transfer functions/conversion between modulated sources and outputs. Possible values are single, first, second and no.
<code>outmodharmnum</code>	<code>1</code>	Harmonic for the PXF output modulation.
<code>inmodharmvec</code>	<code>[...]</code>	Harmonic list for the PXF modulated sources.
<code>moduppersideband</code>	<code>1</code>	Index of the upper sideband included in the modulation of an output for PAC and HBAC, or an input for PXF.

The variable of interest at the output can be voltage or current, and its frequency is not constrained by the period of the large periodic solution. While sweeping the selected output frequency, you can select the periodic small-signal input frequencies of interest by setting

## Spectre Circuit Simulator Reference

### Analysis Statements

---

either the `maxsideband` or the `sidebands` parameter. For a given set of  $n$  integer numbers representing the sidebands  $K_1, K_2, \dots, K_n$ , the input signal frequency at each sideband is computed as  $f(\text{in}) = f(\text{out}) + K_i * \text{fund}(\text{pss})$ , where,  $f(\text{out})$  represents the (possibly swept) output signal frequency and  $\text{fund}(\text{pss})$  represents the fundamental frequency used in the corresponding PSS analysis. Thus, when analyzing a down-converting mixer and sweeping the IF output frequency,  $K_i = +1$  for the RF input represents the first upper-sideband, while  $K_i = -1$  for the RF input represents the first lower-sideband. By setting the `maxsideband` value to  $K_{\text{max}}$ , all  $2 * K_{\text{max}} + 1$  sidebands from  $-K_{\text{max}}$  to  $+K_{\text{max}}$  are selected.

The number of requested sidebands does not change substantially the simulation time. However, the `maxacfreq` of the corresponding PSS analysis should be set to guarantee that  $|\max\{f(\text{in})\}|$  is less than `maxacfreq`; otherwise, the computed solution might be contaminated by aliasing effects. The PXF simulation is not executed for  $|f(\text{out})|$  greater than `maxacfreq`. Diagnostic messages are printed for those extreme cases, indicating how `maxacfreq` should be set in the PSS analysis. In majority of simulations, however, this is not an issue, because `maxacfreq` is never allowed to be smaller than 40x the PSS fundamental.

With PXF, the frequency of the stimulus and of the response are usually different (this is an important area in which PXF differs from XF). The `freqaxis` parameter is used 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`).

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, and you 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 can be a current probe. If the probe component computes more than one current, you use the `porti` parameter to select the appropriate current. It is an error to specify both `portv` and `porti`. If neither is specified, the probe component provides a reasonable default.

The `stimuli` parameter specifies the inputs for the transfer functions. There are two choices. `stimuli=sources` indicates that the sources present in the circuit should be used. The `xfmag` parameters provided by the sources may be used to adjust the computed gain to compensate for gains or losses in a test fixture. One can limit the number of sources in hierarchical netlists by using the `save` and `nestlvl` parameters. `stimuli=nodes_and_terminals` indicates that all possible transfer functions should be computed.

This is useful when it is not known in advance which transfer functions are interesting. 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

## Spectre Circuit Simulator Reference

### Analysis Statements

---

assuming that a unit magnitude value (voltage) source is connected in series with the terminal. By default, the transfer functions from a small set of terminals are computed. If transfer functions from specific terminals are required, specify the terminals in the save statement. You must use the `:probe` modifier (for example, `Rout:1:probe`) or specify `useprobes=yes` on the options statement. If transfer functions from all terminals are required, specify `currents=all` and `useprobes=yes` on the options statement.

Modulated small signal measurements are possible by using the Analog Design Environment ( ADE ) environment. The `modulated` option for PXF and other modulated parameters are set by the Analog Design Environment (ADE). PXF analyses with this option produce results that could have limited use outside such an environment. Direct Plot is configured to analyze these results and combine several wave forms to measure AM and PM transfer function from single sideband or modulated stimuli to the specified output. For details, see the Spectre RF User Guide.

You can define sweep limits by specifying the end points or by providing the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. In addition, You can specify a step size parameter (`step`, `lin`, `log`, or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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 values that the sweep parameter should take by using the `values` parameter. If you specify 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.

## PZ Analysis (pz)

### Description

The PZ analysis linearizes the circuit about the DC operating point and computes the poles and zeros of the linearized network. To compute zeros, you need to specify input sources and output voltages or currents. If no input or output is given, only poles are computed. If there are frequency-dependent components, poles and zeros are computed by approximating those components as equivalent conductances and capacitances evaluated at 1Hz. The PZ analysis uses default direct solver (method=qz) for better accuracy. Performance is better in small-to-medium sized circuits. For large circuits, a Krylov subspace iterative solver (method=arnoldi) can be used for better performance, but, with lesser accuracy.

**Note:** Note: A frequency-dependent component means that the capacitance or conductance-equivalent representation of the component varies with the frequency. Examples are transmission lines or bjts with excess phases. A linear capacitor is not a frequency-dependent component.

Spectre can perform the analysis while sweeping a parameter. The parameter can be temperature, component instance parameter, component model parameter, or netlist parameter. If changing a parameter affects the DC operating point, the operating point is recomputed at each step. You can sweep the parameter `temp` or a netlist parameter by specifying the parameter name without a `dev` or `mod` parameter. After the analysis is complete, the modified parameter returns to its original value.

Pole-zero cancellation is performed when a neighboring pole-zero pair is located within `absdiff` distance. The distance is also determined relatively as `reldiff` times the magnitude of the pole or zero. Spectre uses the larger value of the two distances for cancellation. By default, a lower bound of resistance is enforced. You may remove this limitation by defining the resistor parameter `rac`. However, this may affect pz results.

### Syntax

```
Name ... pz parameter=value ...
```

### Parameters

#### Probe parameters

<code>iprobe</code>	Input probe for zeros of the transfer function.
---------------------	---



## Spectre Circuit Simulator Reference

### Analysis Statements

---

`oprobe` Output probe for zeros of the transfer function.

#### Port parameters

`portv` Voltage across the specified `oprobe` port is the output of the analysis.

`porti` Current through the specified `oprobe` port is the output of the analysis. Should be used when `oprobe` is a voltage source or a current probe.

#### 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.

`valuesfile` Name of the file containing the sweep values.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Sweep variable parameters

dev		Device instance whose parameter value is to be swept.
mod		Model whose parameter value is to be swept.
param		Name of parameter to sweep.
freq	(Hz)	Frequency at which components will be evaluated in setting up the linearized network.

#### State-file parameters

readns		File that contains estimate of DC solution (nodeset).
useprevic	no	If set to yes or ns, use the converged initial condition from previous analysis as ic or ns. Possible values are no, yes and ns.

#### Output parameters

oppoint	no	Should operating point information be computed, and if so, where should it be sent. Possible values are no, screen, logfile and rawfile.
zeroonly	no	If set, only zeros are requested. Possible values are no and yes.

#### Filtering parameters

fmax	(Hz)	Maximum pole and zero frequency value to filter out spurious poles and zeros. This parameter is passed to psf outputs for plotting filtering.
------	------	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

docancel	yes	If set, pole-zero cancellation is requested. Possible values are no and yes.
absdiff	1e-6 Hz	Pole-Zero cancel absolute distance in Hz.
reldiff	1e-4	Pole-Zero cancel relative distance.

#### Convergence parameters

prevoppoint	no	Use the operating point computed on the previous analysis. Possible values are no and yes.
restart	yes	Restart the DC solution from scratch if any condition has changed. If not, use the previous solution as initial guess. Possible values are no and yes.

#### Annotation parameters

annotate	sweep	Degree of annotation. Possible values are no, title, sweep, status and steps.
title		Analysis title.

#### Miscellaneous parameters

method	qz	Method to perform pz analysis. Possible values are qz and arnoldi.
pole_emphasis	lowfreq	If set to lowfreq, calculate low-frequency poles more accurately. If set to highfreq, calculate high-frequency poles more accurately. Possible values are lowfreq and highfreq.
numpoles		Maximum number of poles requested, only for arnoldi method.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

numzeros		Maximum number of zeros requested, only for arnoldi method.
sigmar	0.1	root finding control parameter, only for arnoldi method.
sigmai	0.0	root finding control parameter, only for arnoldi method.

### Examples

```
mypz pz
Pole analysis is performed.
mypz2 (n1 n2) pz iprobe=VIN
Input is VIN and output is voltage difference between nodes n1 and n2. Both pole
and zero analyses are performed.
mypz3 (n1 n2) pz iprobe=I1
Input is I1, output is voltage difference between n1 and n2. Both pole and zero
analyses are performed.
mypz4 pz iprobe=VIN oprobe=IP1 porti=1
Input is VIN, output is current through IP1, where IP1 is an iprobe. Both pole and
zero analyses are performed.
mypz5 pz iprobe=VIN oprobe=V3 porti=1
Input is VIN, output is current through voltage source V3. Both pole and zero
analyses will be performed.
mypz6 pz iprobe=VIN oprobe=R3 portv=1
Input is VIN, output is the voltage across the resistor R3. Both pole and zero
analyses will be performed.
mypz7 (n1 n2) pz iprobe=I1 param=temp start=25 stop=100 step=25
Sweep temperature from 25 C to 100 C with increment of 25 C.
parameters rval=2.0
R2 3 4 resistor r=rval
...
sweep1 sweep param=rval start=1 stop=10 step=1 {
    mypz8 (n1 n2) iprobe=VIN
}
External sweep parameter rval from 1 to 10 with increment of 1.
mypz9 (n1 n2) pz iprobe=VIN docancel=no
Do not perform pole-zero cancellation.
```

### Note:

## Spectre Circuit Simulator Reference

### Analysis Statements

---

**Note:** Note: `portI` allows you to select a current associated with a specific device given in `oprobe` as an output. This device, however, has to have its terminal currents as network variables. Thus, to avoid confusion, `portI` should be used exclusively with voltage sources and current probes and with other components that have voltage-defined branches.

When PZ analysis finishes, a table is printed by default. The table includes values of poles/zeros and a brief notification on the "right-hand side poles", if there are any. This information can also be viewed graphically. In addition to the direct results, "DC gain" is also listed at the end. It calculates the gain of transfer function  $H(s)$ , given  $s=0$ , including the contribution of the excluded poles/zeros (blocked by user-specified `fmax`). "Constant factor" calculates the ratio of the coefficients of the leading terms in the numerator and denominator of  $H(s)$ .

## Quasi-Periodic AC Analysis (qpac)

### Description

The quasi periodic AC (QPAC) analysis is used to compute transfer functions for circuits that exhibit multitone frequency translation. Such circuits include mixers, switched-capacitor filters, samplers, phase-locked loops, and the like. It is a small-signal analysis like AC analysis, except that the circuit is first linearized about a quasi-periodically varying operating point, as opposed to a simple DC operating point. Linearizing about a quasi-periodically time-varying operating point allows transfer-functions that include frequency translation, whereas simply linearizing about a DC operating point could not because linear time-invariant circuits do not exhibit frequency translation. In addition, the frequency of the sinusoidal stimulus is not constrained by the period of the large periodic solution.

Computing the small-signal response of a quasi-periodically varying circuit is a two-step process. First, the small stimulus is ignored and the quasi-periodic steady-state response of the circuit to possibly large periodic stimuli is computed using QPSS analysis. As part of the QPSS analysis, the quasi-periodically time-varying representation of the circuit is computed and saved for later use. The second step is to apply the small stimulus to the periodically varying linear representation to compute the small signal response. This is done using the QPAC analysis.

A QPAC analysis cannot be used independently; it must follow a QPSS analysis. However, any number of quasi-periodic small-signal analyses, such as QPAC, QPSP, QPXF, QPNOISE, can follow a QPSS analysis.

Unlike other analyses in Spectre, this analysis can only sweep frequency.

### Syntax

```
Name qpac parameter=value ...
```

### Parameters

Sweep interval parameters

start	0	Start sweep limit.
stop		Stop sweep limit.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

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.
valuesfile		Name of the file containing the sweep values.
sweepstype		Specifies if the sweep frequency range is an absolute frequency, that is, actual frequency, or if it is relative to the "relharmvec" sideband frequency. Possible values are absolute and relative.
relharmvec	[...]	Sideband - vector of QPSS harmonics to which relative frequency sweep should be referenced.

#### Output parameters

sidevec	[...]	Array of relevant sidebands for the analysis.
clockmaxharm	7	An alternative to the <code>sidevec</code> array specification, which automatically generates the array: [ -clockmaxharm ... 0 ... +clockmaxharm ][-maxharm(QPSS)[2]...0...maxharm(QPSS)[2] ][...].
freqaxis		Specifies whether the results should be printed as per the input frequency, the output frequency, or the absolute value of the output frequency. The default is <code>absout</code> . Possible values are <code>absout</code> , <code>out</code> and <code>in</code> .

## Spectre Circuit Simulator Reference

### Analysis Statements

---

save		Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, none and nooutput.
------	--	---

nestlvl		Levels of subcircuits to output.
---------	--	----------------------------------

#### Convergence parameters

tolerance		Tolerance for linear solver; the default value is 1.0e-9 for shooting-based solver and 1.0e-4 for harmonicbalance-based solver.
-----------	--	---

relativeTol		Relative tolerance for harmonicbalance-based linear solver; the default value is 1.0e-2.
-------------	--	--

gear_order	2	Gear order used for small-signal integration, 1 or 2.
------------	---	---

solver	turbo	Solver type. Possible values are std and turbo.
--------	-------	---

resgmrescycle	short	Restarts GMRES cycle. Possible values are instant, short, long, recycleinstant, recycleshort, recyclelong and custom.
---------------	-------	---

hbprecond_solver	autoset	Select a linear solver for the GMRES preconditioner. Default is autoset. With autoset, the simulator will automatically select the appropriate preconditioner. The preconditioner affects the rate of convergence of the linear matrix solver used in periodic small-signal analysis. When autoset is selected, the simulator may decide to switch to a different preconditioner after the analysis begins. When that happens, the simulator may issue a warning instructing you to choose a different preconditioner during subsequent runs. Although not required, choosing a different preconditioner according to the simulator's instructions may speed up subsequent analyses. Possible values are basicsolver, blocksolver, autoset, blockdense, blocksolver2 and directsolver.
------------------	---------	--



## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>lowmem</code>	0	Harmonic balance low memory mode; Possible values are 0, 1, or number ( $\geq 10$ ). The default value is '0', the low memory mode is turned off; if '1' is set, the standard low memory mode is turned on; If a number no less than 10 is set, Spectre interprets the value as the memory requested in GigaBytes..
<code>krylov_size</code>	200	This parameter is used to set maximum iteration count of the linear matrix solver used in periodic small-signal analysis. After reaching $1.25 \times \text{krylov\_size}$ iterations, the iteration is forced to terminate because of the poor rate of convergence. Increase the <code>krylov_size</code> if the simulation reports insufficient norm reduction errors in GMRES.

#### Annotation parameters

<code>annotate</code>	<code>sweep</code>	Degree of annotation. Possible values are no, title, sweep, status, steps and detailed_hb.
<code>title</code>		Analysis title.

You can select the set of periodic small-signal output frequencies of interest by setting either the `clockmaxharm` or the `sidevec` parameter. Sidebands are vectors in QPAC. Assuming that there is one large tone and one moderate tone in QPSS, a sideband K1 is represented as [K1\_1 K1\_2]. Corresponding frequency is as follows:

$$K1\_1 * \text{fund}(\text{large tone of QPSS}) + K1\_2 * \text{fund}(\text{moderate tone of QPSS})$$

If there are L large and moderate tones in QPSS analysis and a given set of n integer vectors representing the sidebands

$K1 = \{ K1\_1, \dots, K1\_j, \dots, K1\_L \}$ ,  $K2, \dots, Kn$ . the output frequency at each sideband is computed as follows:

$$f(\text{out}) = f(\text{in}) + \text{SUM}_{j=1\_to\_L} \{ Ki\_j * \text{fund}_j(\text{qpss}) \},$$

Where,  $f(\text{in})$  represents the (possibly swept) input frequency, and  $\text{fund}_j(\text{qpss})$  represents the fundamental frequency used in the corresponding QPSS analysis. Thus, when analyzing a down-converting mixer while sweeping the RF input frequency, the most relevant sideband

## Spectre Circuit Simulator Reference

### Analysis Statements

---

for IF output is  $\{-1, 0\}$ . When simulating an up-converting mixer while sweeping IF input frequency, the most relevant sideband for RF output is  $\{1, 0\}$ . You would enter `sidevec` as a sequence of integer numbers, separated by spaces. The set of vectors  $\{1\ 1\ 0\}\{1\ -1\ 0\}\{1\ 1\ 1\}$  becomes `sidevec=[ 1 1 0 1 -1 0 1 1 1]`. For `clockmaxharm`, only the large tone - the first fundamental is affected by this entry; the rest moderate tones are limited by `maxharms`, specified for a QPSS analysis. Given `maxharms=[k1max k2max ... knmax]` in QPSS and `clockmaxharm=Kmax` all  $(2 \cdot K_{\text{max}} + 1) \cdot (2 \cdot k_{2\text{max}} + 1) \cdot (2 \cdot k_{3\text{max}} + 1) \cdot \dots \cdot (2 \cdot k_{n\text{max}} + 1)$  sidebands are generated.

The number of requested sidebands changes substantially the simulation time.

With QPAC, the frequency of the stimulus and of the response are usually different (this is an important area in which QPAC differs from AC). The `freqaxis` parameter is used to specify whether the results should be output versus the input frequency (`in`), the output frequency (`out`), or the absolute value of the output frequency (`absout`).

You can specify sweep limits by specifying the end points or the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. You can specify a step size parameter (`step`, `lin`, `log`, or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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 values that the sweep parameter should take by using the `values` parameter. If you specify 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.

## Quasi-Periodic Noise Analysis (qpnoise)

### Description

The Quasi-Periodic Noise, or QPNOISE analysis, is similar to the conventional noise analysis, except that it includes frequency conversion and intermodulation effects. Hence, it is useful for predicting the noise behavior of mixers, switched-capacitor filters, and other periodically or quasi-periodically driven circuits.

QPNOISE analysis linearizes the circuit about the quasi-periodic operating point computed in the prerequisite QPSS analysis. It is the quasi-periodically time-varying nature of the linearized circuit that accounts for the frequency conversion and intermodulation. The effect of a quasi-periodically time-varying bias point on the noise generated by the various components in the circuit is also included.

The time-average of the noise at the output of the circuit is computed in the form of spectral density versus frequency. The output of the circuit is specified with 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 the input-referred noise is required, specify the input source by using the `iprobe` parameter. Currently, only a `vsource`, an `isource`, or a `port` may be used as an input probe. If the input source is noisy, as is a `port`, the noise analysis computes 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 its `noisetemp` must be set to 16.85C (290K). 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, 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.

The reference sideband (`refsideband`) specifies which conversion gain is used when computing input-referred noise, noise factor, and noise figure. The reference sideband satisfies:

$$lf(input) = lf(out) + refsideband \text{ frequency shift}.$$

The reference sideband option ('`refsidebandoption`') specifies whether to consider the input at the frequency or at the individual quasi-periodic sideband that is specified. Note that Different sidebands can lead to the same frequency.

Sidebands are vectors in QPNOISE. Assuming one large tone and one moderate tone in QPSS, a sideband `Ki` is a vector [`Ki_1` `Ki_2`]. It gives the frequency at:

## Spectre Circuit Simulator Reference

### Analysis Statements

---

$Ki\_1 * fund$  (large tone of QPSS) +  $Ki\_2 * fund$  (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.

The noise analysis always computes the total noise at the output, which includes contributions from the input source and the output load. The amount of the output noise that is attributable to each noise source in the circuit is also computed and output individually. If the input source is identified (using `iprobe`) and is a `vsource` or `isource`, the input-referred noise is computed, which includes the noise from the input source itself. Finally, if the input source is identified (using `iprobe`) and is noisy, as is the case with ports, the noise factor and noise figure are computed. Thus, if:

$No$  = total output noise

$Ns$  = noise at the output due to the input probe (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

$NI$  = 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:

$$IRN = \sqrt{No^2 / G^2}$$

$$F = (No^2 - NI^2) / Ns^2$$

$$NF = 10 * \log_{10}(F)$$

## Spectre Circuit Simulator Reference

### Analysis Statements

---

$$Fdsb = (No^2 - NI^2)/(Ns^2 + Nsi^2)$$

$$NFdsb = 10 \cdot \log_{10}(Fdsb)$$

$$Fieee = (No^2 - NI^2 - Nso^2)/Ns^2$$

$$NFieee = 10 \cdot \log_{10}(Fieee).$$

When the results are output, No is named `out`, IRN is named `in`, G is named `gain`, F, NF, Fdsb, NFdsb, Fieee, and NFieee are named `F`, `NF`, `Fdsb`, `NFdsb`, `Fieee`, and `NFieee`, respectively.

The computation of gain and IRN in QPNOISE assumes that the circuit under test is impedance-matched to the input source. This can introduce inaccuracy into the gain and IRN computation.

Unlike other analyses in Spectre, this analysis can only sweep frequency.

### Syntax

```
Name [p] [n] qpnoise parameter=value ...
```

The optional terminals (p and n) specify the output of the circuit. If you do not specify the terminals, you must specify the output with a probe component.

### Parameters

Sweep interval parameters

<code>start</code>	0	Start sweep limit.
<code>stop</code>		Stop sweep limit.
<code>center</code>		Center of sweep.
<code>span</code>	0	Sweep limit span.
<code>step</code>		Step size, linear sweep.
<code>lin</code>	50	Number of steps, linear sweep.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

dec		Points per decade.
log	50	Number of steps, log sweep.
values	[...]	Array of sweep values.
valuesfile		Name of the file containing the sweep values.
sweepstype		Specifies if the sweep frequency range is an absolute frequency, that is, actual frequency, or if it is relative to the "relharmvec" sideband frequency. Possible values are absolute and relative.
relharmvec	[...]	Sideband - vector of QPSS harmonics to which relative frequency sweep should be referenced.

#### Probe parameters

oprobe		Compute total noise at the output defined by this component.
iprobe		Refer the output noise to this component.
refsideband	[...]	Conversion gain associated with this sideband is used when computing input-referred noise or noise figure.
refsidebandoption	individual	Whether to view the sideband as a specification of a frequency or a specification of an individual sideband. Possible values are freq and individual.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Output parameters

clockmaxharm	7	In shooting pnoise, the parameter determines the maximum sideband that is included when computing noise, that is either up-converted or down-converted to the output by the periodic drive signal. 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 a small maxsideband can cause accuracy loss. The default value for the shooting pnoise is 7. For the HB pnoise, the default is the harms/maxharms setting in the HB large signal analysis..
sidevec	[...]	Array of relevant sidebands for the analysis.
save		Signals to output. The option 'save' specifies the signals to be saved in the result. 'allpub' saves all signals at all levels of hierarchy in the schematic, including the internal signals of device models. 'all' works like 'allpub'. 'lvl' saves all signals through the level of hierarchy set in 'nestlvl' option. 'lvlpub' works like 'lvl'. 'selected' is not recommended to use here. Possible values are all, lvl, allpub, lvlpub and selected.
nestlvl		Levels of subcircuits to output.
saveallsidebands	no	Save noise contributors by sideband. Possible values are no and yes.
noiseout	usb	Specify noise output. You can set a vector like noiseout=[usb am pm]. And all are using single sideband(SSB) convention, half of the total power. Possible values are usb, lsb, am and pm.
separatenoise	no	Separate Noise into sources and transfer functions. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Convergence parameters

tolerance		Tolerance for linear solver; the default value is 1.0e-9 for shooting-based solver and 1.0e-4 for harmonicbalance-based solver.
relativeTol		Relative tolerance for harmonicbalance-based linear solver; the default value is 1.0e-2.
gear_order	2	Gear order used for small-signal integration, 1 or 2.
solver	turbo	Solver type. Possible values are std and turbo.
resgmrescycle	short	Restarts GMRES cycle. Possible values are instant, short, long, recycleinstant, recycleshort, recyclelong and custom.
hbprecond_solver	autoset	Select a linear solver for the GMRES preconditioner. Default is autoset. With autoset, the simulator will automatically select the appropriate preconditioner. The preconditioner affects the rate of convergence of the linear matrix solver used in periodic small-signal analysis. When autoset is selected, the simulator may decide to switch to a different preconditioner after the analysis begins. When that happens, the simulator may issue a warning instructing you to choose a different preconditioner during subsequent runs. Although not required, choosing a different preconditioner according to the simulator's instructions may speed up subsequent analyses. Possible values are basicsolver, blocksolver, autoset, blockdense, blocksolver2 and directsolver.
lowmem	0	Harmonic balance low memory mode; Possible values are 0, 1, or number ( $\geq 10$ ). The default value is '0', the low memory mode is turned off; if '1' is set, the standard low memory mode is turned on; If a number no less than 10 is set, Spectre interprets the value as the memory requested in GigaBytes..



## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>krylov_size</code>	<code>200</code>	This parameter is used to set maximum iteration count of the linear matrix solver used in periodic small-signal analysis. After reaching $1.25 * \text{krylov\_size}$ iterations, the iteration is forced to terminate because of the poor rate of convergence. Increase the <code>krylov_size</code> if the simulation reports insufficient norm reduction errors in GMRES.
--------------------------	------------------	--

#### Annotation parameters

<code>annotate</code>	<code>sweep</code>	Degree of annotation. Possible values are no, title, sweep, status, steps and detailed_hb.
<code>title</code>		Analysis title.

In practice, noise can mix with each of the harmonics of the quasi-periodic drive signal applied in the QPSS analysis and end up at the output frequency. The QPNOISE analysis includes only the noise that mixes with a finite set of harmonics that are specified using the `clockmaxharm` and `sidevec` parameters. Sidebands are vectors in quasi-periodic analyses. For one large tone and one moderate tone in QPSS, a sideband K1 is represented as [K1\_1 K1\_2]. Corresponding frequency shift is as follows:

$$K1\_1 * \text{fund (large tone of QPSS)} + K1\_2 * \text{fund (moderate tone of QPSS)}$$

Assuming that there are L large and moderate tones in QPSS analysis and a given set of n integer vectors representing the sidebands:

$$K1 = \{ K1\_1, \dots, K1\_j, \dots, K1\_L \},$$

K2, ... , Kn.

If Ki represents sideband i, then:

$$f(\text{noise\_source}) = f(\text{out}) + \text{SUM}_{j=1\_to\_L} \{ Ki\_j * \text{fund}_j(\text{qpss}) \},$$

The `clockmaxharm` parameter affects only clock frequency. It can be less or more than `maxharms[1]` in QPSS. Moderate tones are limited by `maxharms` specified in QPSS. Only the selected sidebands specified using the `sidevec` parameter are included in the calculation. Care should be taken when specifying the `sidevec` or `clockmaxharm` in QPNOISE and `maxharms` in QPSS. Noise results are erroneous if you do not include the sidebands that contribute significant noise to the output.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

The number of requested sidebands changes substantially the simulation time.

You can specify sweep limits by specifying the end points or the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. You can specify a step size parameter (`step`, `lin`, `log`, or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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 values that the sweep parameter should take by using the `values` parameter. If you specify 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.

## Quasi-Periodic S-Parameter Analysis (qpsp)

### Description

The quasi-periodic SP (QPSP) 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, phase-locked loops, and the like. It is a small-signal analysis like SP analysis, except, as done in QPAC and QPXF, the circuit is first linearized about a quasi-periodically varying operating point as opposed to a simple 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. QPSP can also calculate noise parameters in frequency-converting circuits. QPSP computes noise figure (both single-sideband and double-sideband), input referred noise, equivalent noise parameters, and noise correlation matrices. As in QPNOISE, but unlike SP, the noise features of the QPSP 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 quasi-periodically varying circuit is a two-step process. First, the small stimulus is ignored and the quasi-periodic steady-state response of the circuit to possibly large periodic stimulus is computed using QPSS analysis. As part of the QPSS analysis, the quasi-periodically time-varying representation of the circuit is computed and saved for later use. The second step is to apply small-signal excitations to compute the n-port S-parameters and noise parameters. This is done using the QPSP analysis. A QPSP analysis cannot be used independently, it must follow a QPSS analysis. However, any number of periodic small-signal analyses, such as QPAC, QPSP, QPXF, QPNOISE, can follow a single QPSS analysis.

Unlike other analyses in Spectre, this analysis can only sweep frequency.

### Syntax

```
Name qpsp parameter=value ...
```

### Parameters

Sweep interval parameters

<code>start</code>	<code>0</code>	Start sweep limit.
<code>stop</code>		Stop sweep limit.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

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.
valuesfile		Name of the file containing the sweep values.
sweepstype		Specifies if the sweep frequency range is an absolute frequency, that is, 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 and 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 port 2.
sprobes	[...]	List of s-probes.
sprobeharmsvect	[...]	List of sprobe harmonics. Set 2 * fundsize harmonics for each sprobe..
portharmsvect	[...]	List of the reference sidebands for the specified list of ports. Must have a one-to-one correspondence with the 'ports' vector.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

harmsvec	[...]	List of sidebands, in addition to ones associated with specific ports by portharmsvec, that are active. Call them secondary.
----------	-------	--

#### Output parameters

freqaxis		Specifies whether the results should be printed as per the input port frequency, the output port frequency, or the absolute value of the input frequency. The default is <code>in</code> . Possible values are <code>absin</code> , <code>in</code> and <code>out</code> .
----------	--	--

file		Output file name. It just supports S-parameters in PSP, HBSP, and QPSP analysis.
------	--	--

datafmt	spectre	Data format of the S-parameter output file. Possible values are <code>spectre</code> , <code>touchstone</code> and <code>touchstone2</code> .
---------	---------	---

datatype	realimag	Data type of the S-parameter output file. Possible values are <code>realimag</code> , <code>magphase</code> and <code>dbphase</code> .
----------	----------	--

noisedata	no	Should noise data be saved to the S-parameter output file; if yes, in what format. If 'twoport' is selected, the first value in the 'ports' vector parameter will be the input and the second one will be the output. Possible values are <code>no</code> and <code>twoport</code> .
-----------	----	--

#### Noise parameters

donoise	yes	Perform noise analysis. If <code>oprobe</code> is specified as a valid port, this is set to <code>yes</code> , and a detailed noise output is generated. Possible values are <code>no</code> and <code>yes</code> .
---------	-----	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Probe parameters

clockmaxharm	7	In shooting pnoise, the parameter determines the maximum sideband that is included when computing noise, that is either up-converted or down-converted to the output by the periodic drive signal. 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 a small maxsideband can cause accuracy loss. The default value for the shooting pnoise is 7. For the HB pnoise, the default is the harms/maxharms setting in the HB large signal analysis..
--------------	---	--

#### Convergence parameters

tolerance		Tolerance for linear solver; the default value is 1.0e-9 for shooting-based solver and 1.0e-4 for harmonicbalance-based solver.
relativeTol		Relative tolerance for harmonicbalance-based linear solver; the default value is 1.0e-2.
gear_order	2	Gear order used for small-signal integration, 1 or 2.
solver	turbo	Solver type. Possible values are std and turbo.
resgmrescycle	short	Restarts GMRES cycle. Possible values are instant, short, long, recycleinstant, recycleshort, recyclelong and custom.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>hbprecond_solver</code>	<code>autoset</code>	Select a linear solver for the GMRES preconditioner. Default is <code>autoset</code> . With <code>autoset</code> , the simulator will automatically select the appropriate preconditioner. The preconditioner affects the rate of convergence of the linear matrix solver used in periodic small-signal analysis. When <code>autoset</code> is selected, the simulator may decide to switch to a different preconditioner after the analysis begins. When that happens, the simulator may issue a warning instructing you to choose a different preconditioner during subsequent runs. Although not required, choosing a different preconditioner according to the simulator's instructions may speed up subsequent analyses. Possible values are <code>basicsolver</code> , <code>blocksolver</code> , <code>autoset</code> , <code>blockdense</code> , <code>blocksolver2</code> and <code>directsolver</code> .
<code>lowmem</code>	<code>0</code>	Harmonic balance low memory mode; Possible values are 0, 1, or number ( $\geq 10$ ). The default value is '0', the low memory mode is turned off; if '1' is set, the standard low memory mode is turned on; If a number no less than 10 is set, Spectre interprets the value as the memory requested in GigaBytes..
<code>krylov_size</code>	<code>200</code>	This parameter is used to set maximum iteration count of the linear matrix solver used in periodic small-signal analysis. After reaching $1.25 \times \text{krylov\_size}$ iterations, the iteration is forced to terminate because of the poor rate of convergence. Increase the <code>krylov_size</code> if the simulation reports insufficient norm reduction errors in GMRES.

#### Annotation parameters

<code>annotate</code>	<code>sweep</code>	Degree of annotation. Possible values are <code>no</code> , <code>title</code> , <code>sweep</code> , <code>status</code> , <code>steps</code> and <code>detailed_hb</code> .
<code>title</code>		Analysis title.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

To specify the QPSP analysis, the port and sideband combinations must be specified. You can select the ports of interest by setting the `port` parameter and the set of periodic small-signal output frequencies of interest by setting `port harmsvec` or `harmsvec` parameter. Sidebands are vectors in QPSP. Assuming that there are one large tone and one moderate tone in QPSS, a sideband K1 is represented as [K1\_1 K1\_2]. The corresponding frequency is as follows:

$$K1\_1 * fund \text{ (large tone of QPSS)} + K1\_2 * fund \text{ (moderate tone of QPSS)} = \text{SUM}_{j=1\_to\_L} \{Ki\_j * fund\_j(qpss)\}$$

It is also assumed that there are L (1 large plus L-1 moderate) tones in QPSS analysis and a given set of n integer vectors representing the sidebands:

$$K1 = \{ K1\_1, \dots, K1\_j, \dots, K1\_L \}, K2, \dots, Kn.$$

If we specify the relative frequency, the scattering parameters at each port are computed at the following frequencies:

$$f(\text{scattered}) = f(\text{rel}) + \text{SUM}_{j=1\_to\_L} \{Ki\_j * fund\_j(qpss)\},$$

Where,  $f(\text{rel})$  represents the relative frequency of a signal incident on a port,  $f(\text{scattered})$  represents the frequency to which the relevant scattering parameter represents the conversion, and  $fund\_j(qpss)$  represents the fundamental frequency used in the corresponding QPSS analysis.

In the analysis of a down-converting mixer with a blocker and of the signal in the upper sideband, sweep the input frequency of the signal coming into RF port. The most relevant sideband for this input is  $Ki = \{1, 0\}$ , and for IF output, it is  $Ki = \{0, 0\}$ . Hence, you can associate  $K1 = \{1, 0\}$  with the RF port and  $K2 = \{0, 0\}$  with the IF port. S21 represents the transmission of a signal from RF to IF, and S11 represents the reflection of the signal back to the RF port. If the signal is in the lower sideband, a choice of  $K1 = \{-1, 0\}$  would be more appropriate.

Either `port harmsvec` or `harmsvec` can be used to specify the sidebands of interest. If `port harmsvec` is given, the sidebands must be in one-to-one correspondence with the ports, with each sideband associated with a single port. If sidebands are specified in the optional `harmsvec` parameter, all possible frequency-translating scattering parameters associated with the specified sidebands on each port are computed.

With QPSP, the frequency of the input and of the response are usually different (this is an important area in which QPSP differs from SP). Because the QPSP computation involves inputs and outputs at frequencies that are relative to multiple sidebands, the `freqaxis` and `sweep type` parameters behave somewhat differently in QPSP than in QPAC and QPXF.

The `sweep type` parameter controls the way the frequencies in the QPSP analysis are swept. A `relative sweep` is a sweep relative to the port sideband (not the QPSS fundamental),



## Spectre Circuit Simulator Reference

### Analysis Statements

---

and an `absolute` sweep is a sweep of the absolute input source frequency. For example, with a QPSS fundamentals of 1000MHz (LO) and 966MHz (blocker in RF channel), `port harmsvec` could be set to [0 1 -1 1] to examine a downconverting mixer. If `sweep type` is set to `relative` with a sweep range of  $f(\text{rel}) = -10\text{MHz} \leftrightarrow 10\text{MHz}$ . S21 would represent the strength of the signal transmitted from the input port in the range 956->976MHz to the output port to the frequencies 24->44MHz. Using `sweep type=absolute` and sweeping the frequency from 966->976MHz would calculate the same quantities, because  $f(\text{abs}) = 956 \leftrightarrow 976\text{MHz}$ , and  $f(\text{rel}) = f(\text{abs}) - (K1\_1 * \text{fund\_1}(\text{qpss}) + K1\_2 * \text{fund\_2}(\text{qpss})) = -10\text{MHz} \leftrightarrow 10\text{MHz}$ ; where,  $K1\_1=0$ ,  $K1\_2=1$ ,  $\text{fund\_1}(\text{qpss}) = 1000\text{MHz}$ , and  $\text{fund\_2}(\text{qpss}) = 966\text{MHz}$ .

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`).

An increase in the number of requested ports increases the simulation time substantially. The same happens if you increase the number of sidebands to be included in the noise computations.

QPSP analysis also computes noise figures, equivalent noise sources, and noise parameters. The noise computation, which is skipped only when `donoise=no`, requires additional simulation time.

If:

No = total output noise at frequency  $f$

Ns = noise at the output due to the input probe (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

NI = noise at the output due to the output probe (the load)

IRN = input referred noise

G = gain of the circuit

F = noise factor (single side band)

NF = noise figure (single side band)

Fdsb = double sideband noise factor

NFdsb = double sideband noise figure

## Spectre Circuit Simulator Reference

### Analysis Statements

---

$F_{ieee}$  = IEEE single sideband noise factor

$NF_{ieee}$  = IEEE single sideband noise figure

Then:

$IRN = \sqrt{N_o^2 / G^2}$

$F = (N_o^2 - N_i^2) / N_s^2$

$NF = 10 \cdot \log_{10}(F)$

$F_{dsb} = (N_o^2 - N_i^2) / (N_s^2 + N_{si}^2)$

$NF_{dsb} = 10 \cdot \log_{10}(F_{dsb})$

$F_{ieee} = (N_o^2 - N_i^2 - N_{so}^2) / N_s^2$

$NF_{ieee} = 10 \cdot \log_{10}(F_{ieee})$ .

When the results are output,  $IRN$  is named `in`,  $G$  is named `gain`,  $F$ ,  $NF$ ,  $F_{dsb}$ ,  $NF_{dsb}$ ,  $F_{ieee}$ , and  $NF_{ieee}$  are named `F`, `NF`, `Fdsb`, `NFdsb`, `Fieee`, and `NFieee`, respectively. Note that the gain computed by QPSP is the voltage gain from the actual circuit input to the circuit output, and not the gain from the internal port voltage source to the output.

To ensure accurate noise calculations, the `clockmaxharm` parameters must be set to include the relevant noise folding effects. `clockmaxharm` is relevant only to the noise computation features of QPSP.

You can specify sweep limits by specifying the end points or the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. You can specify a step size parameter (`step`, `lin`, `log`, or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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 values that the sweep parameter should take by using the `values` parameter. If you specify 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.

## Quasi-Periodic Steady State Analysis (qpss)

### Description

Quasi-periodic steady-state (QPSS) analysis computes circuit response with multiple fundamental frequencies using harmonic balance (in frequency domain) or shooting. QPSS can compute circuit responses with closely spaced or incommensurate fundamentals, which cannot be resolved by PSS efficiently. The simulation time of QPSS analysis is independent of the time-constants of the circuit. In addition, QPSS analysis sets the circuit quasi-periodic operating point, which can then be used during a quasi-periodic time-varying small-signal analysis, such as QPAC, QPXF, QPSP, and QPNOISE.

Generally, harmonic balance(HB) is very efficient in simulating weak nonlinear circuits while shooting is more suitable for computing a circuit response to several moderate input signals, in addition to a large signal. The large signal, which represents a LO or clock signal, is usually the one that causes the most nonlinearity or the largest response. A typical example is the intermodulation distortion measurements of a mixer with two closely spaced moderate input signals. HB is more efficient than shooting in handling frequency-dependent components, such as delay, transmission line, and S-parameter data.

QPSS consists of three phases. First, an initial transient analysis with all moderate input signals suppressed is carried out. Second, a number of (at least 2) stabilizing iterations are run with all signals activated. Finally, the Newton method is followed.

When the shooting method is used, QPSS employs the Mixed Frequency Time (MFT) algorithm extended to multiple fundamental frequencies. For details of 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.

Similar to shooting in PSS, shooting in QPSS uses Newton method as its backbone. However, instead of doing a single transient integration, each Newton iteration does a number of transient integrations of one large signal period. Each of the integrations differs by a phase-shift in each moderate input signal. The number of integrations is determined by the numbers of harmonics of moderate fundamentals specified by `maxharms`. Given `maxharms=[k1 k2 ... kn]`, QPSS always treats `k1` as the maximum harmonic of the large signal, and the total number of integrations is  $(2*k2+1)*(2*k3+1)*...*(2*kn+1)$ . 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

## Spectre Circuit Simulator Reference

### Analysis Statements

---

integrations are consistent with a frequency domain transformation with a shift of one large signal period.

QPSS inherits most of the PSS parameters and adds a few new ones. The most important ones are `funds` and `maxharms`. They replace the PSS parameter, `fund` (or `period`) and `harms`, respectively. The `funds` parameter accepts a list of names of fundamentals that are present in the sources. These names are specified in the sources by the `fundname` parameter. In both shooting and HB QPSS analysis, the first fundamental is considered as the large signal. A few heuristics can be used for picking the large fundamental.

- (1) Pick the fundamental that is not a sinusoidal.
- (2) Pick the fundamental that causes the most nonlinearity.
- (3) Pick the fundamental that causes the largest response.

The `maxharms` parameter accepts a list of numbers of harmonics that are required to sufficiently model responses due to different fundamentals.

The semi-autonomous simulation is a special QPSS analysis combining the autonomous simulation and the QPSS. To perform the semi-autonomous simulation, you need to specify an initial frequency guess for the oscillator inside the circuit, and two oscillator terminals, similar to the autonomous simulation in PSS. For example:

```
myqpss (op on) qpss funds=[1.1GHz frf] maxharms=[5 5] tstab=1u flexbalance=yes
```

The semi-autonomous simulation is only available in the frequency domain.

### Syntax

```
Name ... qpss parameter=value ...
```

### Parameters

QPSS fundamental parameters

<code>funds</code>	<code>[...]</code>	Array of fundamental frequency names for fundamentals to use in analysis.
--------------------	--------------------	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

maxharms	[...]	Array of number of harmonics of each fundamental to consider for each fundamental.
autoharms	no	Activates automatic harmonic number calculation in harmonic balance. Applies only if tstab>0 or if autotstab=yes. If a steady-state is reached, Spectre does a spectrum analysis to calculate the optimal number of harmonics for HB. The minimum number of harmonics is specified by maxharms. If steady-state is not reached to sufficient tolerance, autoharms may be disabled. Possible values are no and yes.
selectharm		Name of harmonics selection methods. Default is diamond when maximorder is set; otherwise, default is box. Possible values are box, diamond, funnel, axis, widefunnel, crossbox, crossbox_hier and arbitrary.
evenodd	[...]	Array of even, odd, or all strings for moderate tones to select harmonics.
maximorder		Maximum intermodulation order of harmonics in diamond, funnel, widefunnel, crossbox and crossbox_hier cuts. For example, given a two-tone case, a harmonic, [hx hy], is in the diamond, funnel or widefunnel harmonic set when  hx + hy  <= maximorder. And it is in the crossbox or crossbox_hier set when  hx  <= maximorder and  hy  <= maximorder.
mappingfft	no	This parameter is valid only for funnel harmonic cut now. When it is set to yes, it is possible to save more memory but maybe at accuracy expense. Possible values are no and yes.
axisbw		The width of the band part of wide funnel harmonic cut along axis.
minialiasorder		The order of mini aliasing harmonic order.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>selharmvec</code>	<code>[...]</code>	Array of harmonics indices.
<code>harmlist</code>	<code>[...]</code>	Array of harmonics indices for time domain qpss.
<code>freqdivide</code>		Large signal frequency division.
<code>freqdividevector</code>	<code>[...]</code>	Array of frequency division factors for each tone. This parameter overrides <code>freqdivide</code> .

#### Simulation interval parameters

<code>tstab</code>	<code>0.0 s</code>	Extra stabilization time after the onset of periodicity for independent sources.
<code>autotstab</code>	<code>no</code>	Activates the automatic initial transient ( <code>tstab</code> ) in harmonic balance and PSS shooting. If set to yes, the simulator decides whether to run <code>tstab</code> and for how long. Typically, the initial length of <code>tstab</code> is 50 periods; however, it could be longer depending on the type of circuit and its behavior. If steady-state is reached (or nearly reached), <code>tstab</code> terminates early. Possible values are no and yes.
<code>envlp_autotstab</code>	<code>no</code>	Activates the automatic initial envlp ( <code>tstabenvlp</code> ) in PSS shooting. If set to yes, the simulator decides how long to run <code>tstabenvlp</code> . Typically, the initial length of <code>tstabenvlp</code> is 50 periods; however, it could be longer depending on the type of circuit and its behavior. If steady-state is reached (or nearly reached), <code>tstabenvlp</code> terminates early. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

autosteady	no	Activates the automatic steady state detection during initial transient (tstab) in harmonic balance and PSS shooting. When steady state is reached (or nearly reached), tstab terminates early. This parameter applies only when tstab>0 or when autotstab=yes. autosteady=no 0 turns this feature off; autosteady=yes 1 activates this feature; autosteady=2 runs autosteady with lower steady state tolerance than autosteady=1. autosteady=2 may help pss convergence but with higher tstab costs. Possible values are 0, 1, 2, no and yes.
tstabenvlp_autosteady	no	Activates the automatic steady state detection during tstabenvlp in PSS shooting. When steady state is reached, tstabenvlp terminates early. This parameter applies only when tstabenvlpstop>0 or when envlp_autotstab=yes. tstabenvlp_autosteady=no 0 turns this feature off; tstabenvlp_autosteady=yes 1 activates this feature; tstabenvlp_autosteady=2 runs autosteady with lower steady state tolerance than tstabenvlp_autosteady=1. tstabenvlp_autosteady=2 may help pss convergence but with higher tstabenvlp costs. Possible values are 0, 1, 2, no and yes.
stabcycles	2	Stabilization cycles with both large and moderate sources enabled.
tstart	0.0 s	Initial transient analysis start time.
tstabenvlpstop	0.0 s	Determines the stop time of envelope tstab.
tstabenvlpstep	0.0 s	Determines the step size of envelope tstab.

#### Time-step parameters

maxstep	(s)	Maximum time step. The default is derived from errpreset.
---------	-----	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

transres	1e-9*stop s	Transition resolution. The transient analysis attempts to stop at corners of input waveforms (for example, corners of rising/falling edge of a pulse). If such events occur within a time less than transres, the analysis combines the events into one and forces only one time point. The rest of the steps are determined by error control. This may lead to loss of detail.
maxacfreq		Maximum frequency requested in a subsequent periodic small-signal analysis. The default is derived from <code>errpreset</code> and <code>harms</code> . This parameter is valid only for shooting.
step	0.001*period s	Minimum time step that would be used solely to maintain the aesthetics of the results. This parameter is valid only for shooting.

#### Initial-condition parameters

ic	all	The value to be used to set the initial condition. Possible values are dc, node, dev and all.
skipdc	no	If set to yes, there is no DC analysis for initial transient. Possible values are no, yes and sigrampup.
readic		File that contains initial condition.
useprevic	no	If set to yes or ns, use the converged initial condition from previous analysis as ic or ns. Possible values are no, yes and ns.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>oscic</code>	<code>default</code>	Oscillator IC method. It determines how the starting values for the oscillator are calculated. <code>oscic=lin</code> provides you an accurate initial value, but it takes time; <code>oscic=lin_ic</code> is not recommended, which is the older version of <code>oscic=lin</code> for shooting analysis for backward compatibility; <code>oscic=fastic</code> is fast, but it is less accurate. <code>'oscic=skip'</code> directly uses the frequency provided by you as the initial guess frequency. It is only for the two-tier method. Possible values are <code>default</code> , <code>lin</code> , <code>lin_ic</code> , <code>fastic</code> and <code>skip</code> .
<code>tone_homotopy</code>	<code>[...]</code>	Array of homotopy options for convergence assistance with multi-divider cases, when <code>hbhomotopy=aggregation</code> . The number of entries should be equal to that of tones. The possible values for each entry are 0, 1 and 2. '0' means no convergence assistance is applied for that tone. '1' means only transient stabilization ( <code>tstab</code> ) is run for that tone and '2' means both <code>tstab</code> and a single-tone HB simulation should be run to obtain the initial guess for multi-tone HB simulation.

#### Convergence parameters

<code>readns</code>		File that contains an estimate of the initial transient solution.
<code>cmin</code>	<code>0 F</code>	Minimum capacitance from each node to ground.

#### Output parameters

<code>save</code>		Signals to output. Possible values are <code>all</code> , <code>lvl</code> , <code>allpub</code> , <code>lvlpub</code> , <code>selected</code> , <code>none</code> and <code>nooutput</code> .
<code>nestlvl</code>		Levels of subcircuits to output.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

oppoint	no	Should operating point information be computed for initial timestep; if yes, where should it be printed (screen or file). Possible values are no, screen, logfile and rawfile.
skipstart	0 s	The time to start skipping output data.
skipstop	stop s	The time to stop skipping output data.
skipcount	1	Save only one of every skipcount points.
strobeperiod	0 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.
saveinit	no	If set to yes, the waveforms for the initial transient before steady state are saved. Possible values are no and yes.

#### State-file parameters

write	File to which initial transient solution (before steady-state) is written.
writefinal	File to which final transient solution in steady-state is written. This parameter is now valid only for shooting.
swapfile	Temporary file to hold steady-state information. It tells Spectre to use a regular file, rather than virtual memory to hold the periodic operating point. Use this option if Spectre complains about not having enough memory to complete the analysis. This parameter is now valid only for shooting.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

`writexpss`

File to which final quasi-periodic steady-state solution is 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 results from shooting QPSS cannot be used in HB QPSS analysis and vice-versa.

`readqpss`

File from which final quasi-periodic steady-state solution is 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 results from shooting QPSS cannot be used in HB QPSS analysis and vice-versa.

`selharmread`

File from which arbitrary harmonic data needs to be read.

`selharmwrite`

File to which harmonic data needs to be written.

#### Integration method parameters

`method`

Integration method. The default is derived from `errpreset`. This parameter is valid only for shooting. Possible values are `euler`, `trap`, `traponly`, `gear2` and `gear2only`.

#### Accuracy parameters

`errpreset`

Selects a reasonable collection of parameter settings. Possible values are `liberal`, `moderate` and `conservative`.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>relref</code>		Reference used for the relative convergence criteria. The default is derived from <code>errpreset</code> . Possible values are <code>pointlocal</code> , <code>alllocal</code> , <code>sigglobal</code> and <code>allglobal</code> .
<code>lteratio</code>		Ratio used to compute LTE tolerances from Newton tolerance. The default is derived from <code>errpreset</code> .
<code>lteminstep</code>	<code>0.0 s</code>	Local truncation error is ignored if the step size is less than <code>lteminstep</code> .
<code>steadyratio</code>		Ratio used to compute steady-state tolerances from LTE tolerance. The default is derived from <code>errpreset</code> .
<code>maxperiods</code>		Maximum number of iterations allowed before convergence is reached in shooting or harmonic balance Newton iteration. For PSS and QPSS, the default is 20 for driven circuits, and 50 for oscillators; For HB, the default is 100.
<code>max_innerkrylov_size</code>	<code>20</code>	The maximum iteration number allowed in inner gmres solver in two-level gmres linear solver. The default value is 12 for large signal and 20 for small signal analysis.
<code>max_outerkrylov_size</code>	<code>4</code>	The maximum iteration number allowed in inner gmres solver in two-level gmres linear solver. The default value is 4.
<code>itres</code>	<code>1e-4 for shooting, 0.9 for HB</code>	Controls the residual for iterative solution of linearized matrix equation at each Newton iteration. Tightening the parameter can help with the Newton convergence, but does not affect the result accuracy. The value should be between [0, 1]. Default value for shooting APS flow is 1e-3.
<code>inexactNewton</code>	<code>no</code>	Inexact Newton method. Possible values are <code>no</code> and <code>yes</code> .

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>finitediff</code>		Options for finite difference method refinement after quasi-periodic shooting method. <code>finitediff</code> is changed from no to same grid automatically when <code>readqpss</code> and <code>writeqpss</code> are used to reuse QPSS results. Possible values are no, yes and refine.
-------------------------	--	---

#### Harmonic Balance parameters

<code>harmonicbalance</code>	<code>no</code>	Use Harmonic Balance engine instead of time-domain shooting. Possible values are no and yes.
<code>flexbalance</code>	<code>no</code>	Same parameter as <code>harmonicbalance</code> . Possible values are no and yes.
<code>hbpartition_defs</code>	<code>[...]</code>	Define HB partitions.
<code>hbpartition_fundratios</code>	<code>[...]</code>	Specify HB partition fundamental frequency ratios.
<code>hbpartition_harms</code>	<code>[...]</code>	Specify HB partition harmonics.
<code>oversamplefactor</code>	<code>1</code>	Oversample device evaluations.
<code>oversample</code>	<code>[...]</code>	Array of oversample factors for each tone. This parameter overrides <code>oversamplefactor</code> .
<code>sweepic</code>	<code>none</code>	IC extrapolation method in sweep HB analysis. Possible values are none, linear and log.
<code>hbhomotopy</code>	<code>tone</code>	Name of Harmonic Balance homotopy selection methods. Possible values are <code>tstab</code> , <code>source</code> , <code>gsweep</code> , <code>tone</code> , <code>inctone</code> , <code>aggregation</code> and <code>steptone</code> .
<code>gstart</code>	<code>1.e-7</code>	Start conductance for <code>hbhomotopy</code> of <code>gsweep</code> .
<code>gstop</code>	<code>1.e-12</code>	Stop conductance for <code>hbhomotopy</code> of <code>gsweep</code> .

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>glog</code>	5	Number of steps, log sweep for hbhomotopy of gsweep.
<code>backtracking</code>	yes	This parameter is used to activate the backtracking utility of Newton's method. The default is yes. Possible values are no, yes and forced.
<code>excludeconvgwith BK</code>	yes	Possible values are no and yes.
<code>krylov_size</code>	10	The minimum iteration count of the linear matrix solver used in HB large-signal analysis. After reaching <code>krylov_size</code> iterations, the iteration is forced to terminate because of the poor rate of convergence. Increase <code>krylov_size</code> if the simulation reports insufficient norm reduction errors in GMRES.
<code>hbprecond_solver</code>	<code>basicsolver</code>	Choose a linear solver for the GMRES preconditioner. Possible values are <code>basicsolver</code> , <code>blocksolver</code> , <code>autoset</code> , <code>blockdense</code> , <code>blocksolver2</code> and <code>directsolver</code> .
<code>lowmem</code>	0	Harmonic balance low memory mode; Possible values are 0, 1, or number ( $\geq 10$ ). The default value is '0', the low memory mode is turned off; if '1' is set, the standard low memory mode is turned on; If a number no less than 10 is set, Spectre interprets the value as the memory requested in GigaBytes.

#### Annotation parameters

<code>annotate</code>	sweep	Degree of annotation. Possible values are no, title, sweep, status, estimated, steps, iters, detailed, rejects, alliters, <code>detailed_hb</code> and <code>internal_hb</code> .
<code>annotateic</code>	no	Degree of annotation for initial condition. Possible values are no, title, sweep, status, estimated, steps, iters, detailed, rejects and alliters.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

title		Analysis title.
-------	--	-----------------

#### Newton parameters

maxiters	5	Maximum number of iterations per time step.
----------	---	---

restart	no	Restart the DC/PSS/QPSS solution if set to 'yes'; if set to 'no', reuse the previous solution as an initial guess; if set to 'firstonly', restart if it is the first point of sweep (supported only in HB). The default value is 'no' for HB and 'yes' for shooting. Possible values are no, yes and firstonly.
---------	----	---

#### Circuit age

circuitage	(Years)	Stress time. Age of the circuit used to simulate hot-electron degradation of MOSFET and BSIM circuits.
------------	---------	--

#### Tstab save/restart parameters

ckptperiod		Checkpoint the analysis periodically by using the specified period.
------------	--	---

saveperiod		Save the tran analysis periodically on the simulation time.
------------	--	---

saveperiodhistory	no	Maintains the history of saved files. If yes, maintains all the saved files. Possible values are no and yes.
-------------------	----	--

saveclock	(s)	Save the tran analysis periodically on the wall clock time. The default is 1800s for Spectre. This parameter is disabled in the APS mode by default.
-----------	-----	--

savetime	[...]	Save the analysis states into files on the specified time points.
----------	-------	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

`savefile` Save the analysis states into the specified file.

`recover` Specify the file to be restored.

#### Compression parameters

`xdbcompression`      `no` Sets the automatic gain compression analysis. In automatic gain compression analysis, Spectre automatically sweeps the input excitation until the gain, as defined by the analysis parameter `xdbgain`, compresses by the amount specified by the analysis parameter `xdblevel`. In gain compression analysis, Spectre outputs the hb solution at the calculated compression point only. Dependent analyses, such as `hnoise` and `hbac`, are supported and calculated about the calculated compression level. Auxiliary output includes the gain and voltage/power compression curves. These outputs are available for analysis and post-processing in ADE. The possible values are yes and no. Default is no.

`xdblevel`              `[...]` Sets the gain compression level for compression analysis. The reference point for gain compression is the small-signal gain of the circuits, or as specified by the analysis parameter `xdbref`. Default is 1.

`xdbgain`                `power` Chooses between the voltage gain and transducer power gain as the target for compression point calculation. When `xdbgain=power`, the gain is defined as  $G \text{ (dB)} = P_{\text{load}} \text{ (dBm)} - P_{\text{available}} \text{ (dBm)}$ . When `xdbgain=voltage`, the gain is defined as  $G \text{ (dB)} = 20 \log(|V_{\text{load}}|/|V_{\text{source}}|)$ . In both cases, Spectre sweeps the excitation source until `xdbref` -  $G = \text{xdblevel}$ , where the analysis parameter `xdbref` defines the reference level for compression calculation. Possible values are power and voltage. Default is power.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

xdbref	linear	Sets the reference point for gain compression calculations. When xdbref=linear, spectre uses the small-signal gain as the reference. When xdbref=max, spectre uses the maximum observed gain as the reference. Possible values are linear and max. Default is linear.
xdbsource		The instance name of the excitation source, which is swept automatically to reach the compression level. When xdbgain=power, the excitation source must be a port instance. When xdbgain=voltage, the excitation source must be a vsource instance.
xdbload		The instance name of the load termination. When xdbgain is power, xdbload can be a port, a resistor, or a current probe.
xdbnodep		The output terminals for voltage gain calculation when xdbgain=voltage. If either is left unspecified, the terminal is assumed to be the global ground.
xdbnoden		The output terminals for voltage gain calculation when xdbgain=voltage. If either is left unspecified, the terminal is assumed to be the global ground.
xdbrefnode		The reference node when xdbload is a current probe. The default is the ground node.
xdbharm	[...]	The Integer array which specifies the harmonic indexes of the output voltage or power component.
xdbsteps	100	The maximum number of steps for the compression point search. The simulator terminates if xdbsteps exceeds before the compression point is found. The default is 100.
xdbmax		The maximum input power (or voltage) for the compression point search. Default is 30 dBm when xdbgain=power, and 2.0 V when xdbgain=voltage.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>xdbstart</code>		The starting input power (or voltage) for the compression point search. Default is (xdbmax-70) dBm when xdbgain=power, and xdbmax/20000 when xdbgain=voltage.
<code>xdbtol</code>	<code>0.01</code>	Sets the tolerance for compression analysis. This tolerance is used in compression curve fitting and calculating the compression point.
<code>xdbrapid</code>	<code>no</code>	Sets the automatic gain compression analysis in rapid mode. In this mode, Spectre does not trace the compression curve and calculates only the compression point.
<code>xdbcpi</code>		Sets the estimated input-referred compression point for rapid compression analysis.
<code>backoff</code>	<code>0.0</code>	The backoff point. If defined, an additional harmonic balance analysis will be performed after the compression analysis is done. Default is 0 dBm when xdbgain=power, and 0 V when xdbgain=voltage.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Memory estimation parameters

<code>memoryestimate</code>	<code>no</code>	Sets the memory usage estimate for Harmonic Balance. If yes, a memory estimate is printed in the log file. You can use this memory estimate to plan the computing resources before submitting harmonic balance runs. In memory estimate mode, a short simulation is performed first, and the engine exits after printing the estimate in the log file without saving any results. If no, the simulation continues after the memory estimate is printed. Memory estimation is not recommended for simulations that require less than 500MB approximately. For PSS analysis, memory estimate mode does not apply unless <code>flexbalance=yes</code> . <code>memoryestimate=1</code> estimates the memory usage for large-signal analysis and <code>memoryestimate=2</code> estimates both large-signal analysis and small-signal simulations. Possible values are no and yes.
-----------------------------	-----------------	--

Most of QPSS analysis parameters are inherited from PSS analysis and their meanings remain essentially unchanged. Two new important parameters are `funds` and `maxharms`. They replace and extend the role of `fund` and `harms` parameters of PSS analysis. One important difference is that `funds` accepts a list of fundamental names, instead of actual frequencies. The frequencies associated with fundamentals are figured out automatically by the simulator. An important feature is that each input signal can be a composition of more than one source. However, these sources must have the same fundamental name. For each fundamental name, its fundamental frequency is the greatest common factor of all frequencies associated with the name. Omitting fundamental name in the `funds` parameter is an error that stops the simulation. If `maxharms` is not given, a warning message is issued, and the number of harmonics defaults to 1 for each fundamental.

For QPSS analyses, the role of some PSS parameters is extended compared to their role in PSS analysis. In QPSS, the parameter `maxperiods` that controls the maximum number of shooting iterations for PSS analysis also controls the number of the maximum number of shooting iterations for QPSS analysis. Its default value is set to 50.

The `tstab` parameter controls both the length of the initial transient integration with only the clock tone activated and the number of stable iterations with moderate tones activated. The stable iterations are run before shooting or HB Newton iterations.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

The `errpreset` parameter lets you adjust several simulator parameters to fit your needs. In most cases, `errpreset` should be the only parameter you need to adjust. If you want a fast simulation with reasonable accuracy, set `errpreset` to `liberal`. If you have some concern for accuracy, set `errpreset` to `moderate`. If accuracy is your main interest, set `errpreset` to `conservative`.

If you do not specify `steadyratio`, it is always 1.0, and it is not affected by `errpreset`. The following table shows the effect of `errpreset` on other parameters in shooting.

-----  
Parameter defaults as a function of `errpreset`  
-----

<code>errpreset</code>	<code>reltol</code>	<code>relref</code>	<code>method</code>	<code>lteratio</code>	<code>maxstep</code>
-----					
<code>liberal</code>	1e-3	<code>sigglobal</code>	<code>gear2only</code>	3.5	clock period/80
<code>moderate</code>	1e-4	<code>sigglobal</code>	<code>gear2only</code>	3.5	clock period/100
<code>conservative</code>	1e-5	<code>sigglobal</code>	<code>gear2only</code>	*	clock period/200

\* : `lteratio`=10.0 for conservative `errpreset` by default. However, when the specified `reltol`  $\leq 1e-5 \times 10.0 / 3.5$ , `lteratio` is set to 3.5.

The new `errpreset` settings include a new default `reltol` that is actually an enforced upper limit for appropriate setting. An increase of `reltol` above the default is ignored by the simulator. You can decrease this value in the options statement. The only way to increase `reltol` is to relax `errpreset`. Spectre sets the value of `maxstep` so that it is no larger than the value given in the table. Except for `reltol` and `maxstep`, `errpreset` does not change the value of any parameters 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. For HB, only `reltol` is affected by `errpreset`, and the effect is the same as that in shooting. However, `lteratio` remains 3.5 and `steadyratio` remains 1 with all values of `errpreset`.

With parameter 'hbmhomotopy', you can specify harmonic balance homotopy selection methods. The possible values of parameter 'hbmhomotopy' and their descriptions are as follows:

## Spectre Circuit Simulator Reference

### Analysis Statements

---

'hbhomotopy=tstab': Simulator runs a transient analysis and generates an initial guess for harmonic balance analysis; it is recommended for nonlinear circuits or circuits with frequency dividers.

'hbhomotopy=source': For driven circuit, the simulator ignores tstab and accordingly increases the source power level; for oscillators, the simulator accordingly adjusts the probe magnitude until the probe has no effect on the oscillators. It is recommended for strongly nonlinear or high Q circuits.

'hbhomotopy=tone': This method is valid only for multi-tone circuit. The simulator first solves a single-tone circuit by turning off all the tones, except the first one, and then solves the multi-tone circuit by restoring all the tones and using the single-tone solution as its initial guess. It is recommended for multi-tone simulation with a strong first tone.

'hbhomotopy=inctone': Simulator first solves a single tone, then turns on moderate tones incrementally till all tones are enabled. It is recommended for circuits with one strong large tone.

'hbhomotopy=gsweep': A resistor, whose conductance is  $g$ , is connected with each node, and the sweep of  $g$  is controlled by  $gstart$ ,  $gstop$ , and  $glog$ . It is recommended for circuits containing high-impedance or quasi-floating nodes.

## Quasi-Periodic Transfer Function Analysis (qpxf)

### Description

A conventional transfer function analysis computes the transfer function from every source in the circuit to a single output. Unlike a conventional AC analysis that computes the response from a single stimulus to every node in the circuit, the Quasi Periodic Transfer Function or QPXF analysis computes the transfer functions from any source 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 such useful quantities as conversion efficiency (transfer function from input to output at required frequency), image and sideband rejection (input to output at undesired frequency), and LO feed-through and power supply rejection (undesired input to output at all frequencies).

As with a QPAC, QPSP, and QPNOISE analyses, a QPXF analysis must follow a QPSS analysis.

Unlike other analyses in Spectre, this analysis can only sweep frequency.

### Syntax

```
Name [p] [n] qpxf parameter=value ...
```

The optional terminals (p and n) specify the output of the circuit. If you do not specify the terminals, you must specify the output with a probe component.

### Parameters

Sweep interval parameters

start	0	Start sweep limit.
stop		Stop sweep limit.
center		Center of sweep.
span	0	Sweep limit span.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>step</code>		Step size, linear sweep.
<code>lin</code>	50	Number of steps, linear sweep.
<code>dec</code>		Points per decade.
<code>log</code>	50	Number of steps, log sweep.
<code>values</code>	[...]	Array of sweep values.
<code>valuesfile</code>		Name of the file containing the sweep values.
<code>sweepstype</code>		Specifies if the sweep frequency range is an absolute frequency, that is, actual frequency, or if it is relative to the "relharmvec" sideband frequency. Possible values are absolute and relative.
<code>relharmvec</code>	[...]	Sideband - vector of QPSS harmonics to which relative frequency sweep should be referenced.

#### Probe parameters

<code>probe</code>		Compute every transfer function to this probe component.
--------------------	--	--

#### Output parameters

<code>stimuli</code>	<code>sources</code>	Stimuli used for xf analysis. Possible values are sources and nodes_and_terminals.
<code>sidevec</code>	[...]	Array of relevant sidebands for the analysis.
<code>clockmaxharm</code>	7	An alternative to the <code>sidevec</code> array specification, which automatically generates the array: [ -clockmaxharm ... 0 ... +clockmaxharm ] [-maxharm(QPSS)[2]...0...maxharm(QPSS)[2] ][...].

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>freqaxis</code>		Specifies whether the results should be printed as per the input frequency, the output frequency, or the absolute value of the input frequency. The default is <code>absin</code> . Possible values are <code>absin</code> , <code>in</code> and <code>out</code> .
<code>save</code>		Signals to output. Possible values are <code>all</code> , <code>lvl</code> , <code>allpub</code> , <code>lvlpub</code> , <code>selected</code> , <code>none</code> and <code>nooutput</code> .
<code>nestlvl</code>		Levels of subcircuits to output.

#### Convergence parameters

<code>tolerance</code>		Tolerance for linear solver; the default value is 1.0e-9 for shooting-based solver and 1.0e-4 for harmonicbalance-based solver.
<code>relativeTol</code>		Relative tolerance for harmonicbalance-based linear solver; the default value is 1.0e-2.
<code>gear_order</code>	<code>2</code>	Gear order used for small-signal integration, 1 or 2.
<code>solver</code>	<code>turbo</code>	Solver type. Possible values are <code>std</code> and <code>turbo</code> .
<code>resgmrescycle</code>	<code>short</code>	Restarts GMRES cycle. Possible values are <code>instant</code> , <code>short</code> , <code>long</code> , <code>recycleinstant</code> , <code>recycleshort</code> , <code>recyclelong</code> and <code>custom</code> .



## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>hbprecond_solver</code>	<code>autoset</code>	Select a linear solver for the GMRES preconditioner. Default is <code>autoset</code> . With <code>autoset</code> , the simulator will automatically select the appropriate preconditioner. The preconditioner affects the rate of convergence of the linear matrix solver used in periodic small-signal analysis. When <code>autoset</code> is selected, the simulator may decide to switch to a different preconditioner after the analysis begins. When that happens, the simulator may issue a warning instructing you to choose a different preconditioner during subsequent runs. Although not required, choosing a different preconditioner according to the simulator's instructions may speed up subsequent analyses. Possible values are <code>basicsolver</code> , <code>blocksolver</code> , <code>autoset</code> , <code>blockdense</code> , <code>blocksolver2</code> and <code>directsolver</code> .
<code>lowmem</code>	<code>0</code>	Harmonic balance low memory mode; Possible values are 0, 1, or number ( $\geq 10$ ). The default value is '0', the low memory mode is turned off; if '1' is set, the standard low memory mode is turned on; If a number no less than 10 is set, Spectre interprets the value as the memory requested in GigaBytes..
<code>krylov_size</code>	<code>200</code>	This parameter is used to set maximum iteration count of the linear matrix solver used in periodic small-signal analysis. After reaching $1.25 \times \text{krylov\_size}$ iterations, the iteration is forced to terminate because of the poor rate of convergence. Increase the <code>krylov_size</code> if the simulation reports insufficient norm reduction errors in GMRES.

#### Annotation parameters

<code>annotate</code>	<code>sweep</code>	Degree of annotation. Possible values are <code>no</code> , <code>title</code> , <code>sweep</code> , <code>status</code> , <code>steps</code> and <code>detailed_hb</code> .
<code>title</code>		Analysis title.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

The variable of interest at the output can be voltage or current, and its frequency is not constrained by the period of the large periodic solution. While sweeping the selected output frequency, select the periodic small-signal input frequencies of interest by setting either the `clockmaxharm` or `sidevec` parameter. Sidebands are vectors in QPXF. Assuming that there is one large tone and one moderate tone in QPSS, a sideband K1 is represented as [K1\_1 K1\_2]. The corresponding frequency is as follows:

$$K1\_1 * fund \text{ (large tone of QPSS)} + K1\_2 * fund \text{ (moderate tone of QPSS)}$$

In addition, assume that there are L (1 large plus L-1 moderate) tones in QPSS analysis and a given set of n integer vectors representing the sidebands:

$$K1 = \{ K1\_1, \dots, K1\_j, \dots, K1\_L \}, K2, \dots, Kn.$$

The input signal frequency at each sideband is computed as follows:

$$f(in) = f(out) + \text{SUM}_{j=1\_to\_L} \{ K1\_j * fund\_j(qpss) \},$$

Where,  $f(out)$  represents the (possibly swept) output signal frequency, and  $fund\_j(pss)$  represents the fundamental frequency used in the corresponding QPSS analysis. Thus, when analyzing a down-converting mixer and sweeping the IF output frequency,  $Ki = \{1, 0\}$  for the RF input represents the first upper-sideband, while  $Ki = \{-1, 0\}$  for the RF input represents the first lower-sideband.

Enter `sidevec` as a sequence of integer numbers, separated by spaces. The set of vectors  $\{1 \ 1 \ 0\} \{1 \ -1 \ 0\} \{1 \ 1 \ 1\}$  becomes `sidevec=[ 1 1 0 1 -1 0 1 1 1]`. For `clockmaxharm`, only the large tone- the first fundamental is affected; the rest moderate tones are limited by `maxharms`, specified for a QPSS analysis. Given `maxharms=[k1max k2max ... knmax]` in QPSS and `clockmaxharm=Kmax`, all  $(2*Kmax + 1)*(2*k2max+1)*(2*k3max+1)*\dots*(2*knmax+1)$  sidebands are generated.

The number of requested sidebands changes substantially the simulation time.

With QPXF, the frequency of the stimulus and of the response are usually different (this is an important area in which QPXF differs from XF). The `freqaxis` parameter is used 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`).

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, and you use the `portv` parameter to select the appropriate pair of terminals. Alternatively, you can specify a voltage to be the output by giving a pair of nodes on the QPXF analysis statement.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

Any component that naturally computes current as an internal variable can be a current probe. If the probe component computes more than one current, you use the `porti` parameter to select the appropriate current. It is an error to specify both `portv` and `porti`. If neither is specified, the probe component provides a reasonable default.

The `stimuli` parameter specifies the inputs for the transfer functions. There are two choices. `stimuli=sources` indicates that the sources present in the circuit should be used. The `xfmag` parameters provided by the sources may be used to adjust the computed gain to compensate for gains or losses in a test fixture. You can limit the number of sources in hierarchical netlists by using the `save` and `nestlvl` parameters. `stimuli=nodes_and_terminals` indicates that all possible transfer functions should be computed.

This is useful when it is not known in advance which transfer functions are interesting. 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 from a small set of terminals are computed. If transfer functions from specific terminals are required, specify the terminals in the `save` statement. You must use the `:probe` modifier (for example, `Rout:1:probe`) or specify `useprobes=yes` on the options statement. If transfer functions from all terminals are required, specify `currents=all` and `useprobes=yes` on the options statement.

You can specify sweep limits by specifying the end points or the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. You can specify a step size parameter (`step`, `lin`, `log`, or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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 values that the sweep parameter should take by using the `values` parameter. If you specify 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.

## Reliability Analysis (reliability)

### Description

This analysis computes the reliability for MOSFETs and the circuit. The three reliability modules are:

1. MOSFET Hot-Carrier Injection(HCI) module:

Predicts transistor and circuit performance degradation due to HCI effects.

2. PMOSFET Negative Bias Temperature Instability(NBTI) module:

Predicts PMOSFET and circuit performance degradation due to NBTI and NBTI recovery effects.

3. NMOSFET Positive Bias Temperature Instability(PBTI) module:

Predicts NMOSFET and circuit performance degradation due to PBTI effects.

Synopsis:

```
name reliability <global options> {  
    <reliability control statements> ...  
    <stress simulation statements> ...  
    <aging testbench statements> ...  
    <aging/post-stress simulation statements> ...  
}
```

**Note:** Note: Starting from MMSIM10.1.0 release version, the <global options> are ignored.

The other four sections should be listed in the specified order.

### Syntax

```
Name reliability parameter=value ...
```

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Parameters

Analysis parameters		Analysis title.
age time	[...]	The time in future when the transistor degradation and degraded SPICE model parameters are to be calculated.
deltad value	[...]	Specifies the degradation value. The degradation value can be transconductance, linear or saturation drain current degradation, threshold voltage shift, or any other degradation monitor, depending on the definitions of the lifetime parameters H and m..
vec_deltad item	[...]	Specifies the degradation value in vector.
deltad mod	[...]	Specifies the name of a model whose lifetime is calculated. The model name must be the same as specified in the .model card.
maskdev type	include	Includes or excludes the specified devices or the devices that belong to the listed subcircuit or model during reliability analysis. Possible values are include and exclude.
deg_ratio type	include	Includes or excludes the specified devices for built-in agemos model. Possible values are include and exclude.
deg_ratio dev	[...]	Specifies the instances to be included or excluded for degradation ratio of built-in agemos model.
maskdev mod	[...]	Specifies the models for which the related devices should be included or excluded while performing reliability analysis.
maskdev dev	[...]	Specifies the instances to be included or excluded during reliability analysis.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

maskdev sub	[...]	Specifies the subcircuit(s) for which the related devices should be included or excluded while performing reliability analysis.
maskdev tmioff	all	Specifies the analysis which needs to be turned off for masked devices in tmi. Possible values are none, aging, rtshe and all.
report_model_param value	no	Determines whether to print the stress and aged parameters in the.bm# file. Possible values are no and yes.
load_external_cmi value	no	Load the external user-defined CMI flag. Possible values are no and yes.
load_external_cmi debug	no	The flag of outputting the debug information for loading external customerdefined CMI shared library.
accuracy level	1	Specifies methods used in the reliability simulation when performing integration and substrate current calculation. Possible values are 1 and 2.
minage value	0.0	Specifies the smallest age value for which degraded SPICE model parameters are calculated.
igatemethod type	calc	Specifies the method used for obtaining the gate currents of MOSFETs. Possible values are calc and spice.
idmethod type	ids	Specifies how the simulator obtains the drain current (Id) of MOSFETs to perform reliability calculations. Possible values are ids, idrain and idstatic.
dumpagemodel file		Output file name of dump age model.
output_aging_model_info file		Output file of aging model info.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>dumpagemodel dev</code>	<code>[...]</code>	Specifies the instances whose aged modelcard will be output.
<code>urilib file</code>		Specifies URI library file name.
<code>urilib files</code>	<code>[...]</code>	Specifies URI libraries file name for multiple uri libraries.
<code>urilib uri_mode</code>	<code>scaleparam</code>	Specifies which method should be used to perform aging simulation.. Possible values are <code>agemos</code> , <code>scaleparam</code> , <code>mixed_mode</code> , <code>appendage</code> , <code>new_appendage</code> , <code>appendage2</code> and <code>appendage1</code> .
<code>urilib debug</code>	<code>0</code>	Specifies the debug mode for URI library. The value can be 0 or positive. When specified, a flag is added to the URI library indicating whether the debug information should be printed. <code>debugMode</code> is not 0 prints debug messages. Default value is 0.
<code>bias_mode_urilib</code> <code>bjt</code>	<code>-1</code>	Specifies the bjt bias mode for pnp type..
<code>relxtran start</code>	<code>0.0</code>	Specifies the start time of reliability analysis during transient simulation.
<code>relxtran stop</code>	<code>0.0</code>	Specifies the stop time of reliability analysis during transient simulation. If <code>stop_time</code> is not specified, the software stops in <code>.tran</code> statement.
<code>isubmethod type</code>	<code>calc</code>	Specifies the method used for obtaining the substrate currents of MOSFET. Possible values are <code>calc</code> and <code>spice</code> .
<code>opmethod type</code>	<code>calc</code>	Specifies whether the <code>Igate</code> or <code>Isub</code> value should be obtained from the SPICE models (such as BSIM3 or BSIM4) or whether the internal <code>Igate</code> or <code>Isub</code> equation should be used. Possible values are <code>calc</code> and <code>spice</code> .

## Spectre Circuit Simulator Reference

### Analysis Statements

---

degsort threshold	0.0	Prints MOS transistors based on the threshold and number settings. The results are sorted in the descending order of degradation.
degsort number	0	Prints only the specified number of transistors having the highest degradations. For example, if number=100, the software will print the first 100 transistors with highest degradations.
agelevel_only value	[...]	Sets the level for reliability analysis, which is essentially the age level number of the reliability analysis to be performed.
gradual_aging_ag estep type	lin	Sets the type of agestep, linear or logarithm, the default type is linear. Possible values are lin and log.
gradual_aging_ag estep start	0	Specifies the start time of agestep in gradual aging flow, the default value is 0.
gradual_aging_ag estep stop	0	Specifies the stop time of agestep in gradual aging flow.
gradual_aging_ag estep total_step	1	Specifies the total step numbers of agestep in gradual aging flow.
gradual_aging_ag estep profile	no	Select this option to run profile gradual aging. In profile gradual aging, results of next age step doesn't depend on the results of the previous age step. Possible values are no and yes.
gradual_aging_ag epoint points	[...]	Specifies points of agepoints.
gradual_aging_ag epoint profile	no	Select this option to run profile gradual aging. In profile gradual aging, results of next age point doesn't depend on the results of the previous age point. Possible values are no and yes.
gradual_aging_sa ve steps	[...]	Save the specified step of gradual aging result files.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

gradual_aging_save savedatainseparatedir		Dump the waveform files in a separate directory for gradual aging result files. Possible values are no and yes.
rel_savedatainseparatedir value	no	Dump the waveform files in a separate directory. Possible values are no and yes.
gradual_aging_alter time	[...]	Set time for gradualAging alter.
gradual_aging_alter param	[...]	Set params for gradualAging alter.
gradual_aging_alter value	[...]	Set values for gradualAging alter.
simmode type		Mode of reliability analysis. Possible values are stress, aging and all.
simmode file		File of simMode.
simmode mapping		Device Name Mapping File of simMode.
simmode tmifile		Input file for TMI Aging flow.
macrodevice sub	[...]	Specifies the instances for macrodevice.
combinedeg value	no	Determines whether to combine the external URI results with internal URI results in the bo0 file. Possible values are no and yes.
keep_aged_data value	yes	Determines whether to keep the value of the aged model parameters after reliability analysis. Possible values are no and yes.
check_neg_aging type	error	Specifies the message type to check the negative value in bt0. Default is error. Possible values are error, warn and ignore.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

check_neg_aging clamp	no	Clamps the negative age value in the bt0 file. Possible values are no and yes.
enable_negative_ age value	no	Enable negative age value. Possible values are no and yes.
output_binary_fi le value	no	Specifies the format of reliability analysis output file. Possible values are no and yes.
deg_ratio hci	1.0	The degradation ratio for HCI effect.
deg_ratio nbti	1.0	The degradation ratio for NBTI effect.
deg_ratio pbti	1.0	The degradation ratio for PBTI effect.
deg_ratio bti	1.0	The degradation ratio for NBTI/PBTI effect.
type_urilib append	inline	Compatibility with RelXpert. Possible values are inline, sub and dev.
enable_ade_proce ss value	no	Enable reliability in ADE. Possible values are no and yes.
macrodevice type	appendpa rams	Specifies the instances for macrodevice. Possible values are appendparams and agemos.
tmi_aging_mode type		Mode of tmi aging flow. Possible values are aging, she and all.
omi_aging_mode type		Mode of omi aging flow. Possible values are aging, she and all.
simmode omifile		Input file for OMI Aging flow.
output_region_pa ram value	no	Prints the region method of the parameter. Possible values are no and yes.
tmi_she_mindtemp value	0.0	The minimum delta temperature of self-heating.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>enable_tmi_uri</code>	<code>no</code>	Enable the TMI and URI flow. Possible values are no and yes.
<code>fast_aging_mode</code>		Mode of XPS native reliability. Possible values are agemos and ratio.
<code>aging_analysis_name</code>	<code>value</code>	Aging analysis name. Possible values are tran, dc and ac.
<code>deg_ratio_hcin</code>	<code>1.0</code>	The degradation ratio for NMOS HCI effect, it is for TMI aging flow only.
<code>deg_ratio_hcip</code>	<code>1.0</code>	The degradation ratio for PMOS HCI effect, it is for TMI aging flow only.
<code>deg_ratio_btin</code>	<code>1.0</code>	The degradation ratio for PBTI effect, it is for TMI aging flow only.
<code>deg_ratio_btip</code>	<code>1.0</code>	The degradation ratio for NBTI effect, it is for TMI aging flow only.
<code>output_device_degrad</code>	<code>[...]</code>	Sets the bias voltage for the device degradation.
<code>output_device_degrad</code>	<code>tmi_lib_inc</code>	Sets the TMI library include/section for gm/gds degradation.
<code>deltad</code>	<code>item</code>	Specifies the electrical parameter for lifetime estimation. It is only available for TMI aging flow. Possible values are didsat, didlin, dvtlin, dvtsat and lifetime.
<code>degsort</code>	<code>item</code>	Specifies which item must be used for sorting. It is only available for TMI aging flow. Possible values are didsat, didlin, dvtlin, dvtsat and lifetime.
<code>output_device_degrad</code>	<code>[...]</code>	Sets the bias voltage of vdlin for the device degradation.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>output_device_degrad vgsat</code>	<code>[...]</code>	Sets the bias voltage of vgsat for the device degradation.
<code>output_device_degrad vglin</code>	<code>[...]</code>	Sets the bias voltage of vglin for the device degradation.
<code>degsort phys</code>		Specify which reliability effect used for sorting. It is only available for TMI aging flow. Possible values are hci_bti, hci and bti.
<code>output_inst_parameter list</code>	<code>[...]</code>	Specify the parameter names to be output into bo0 or psf file.
<code>degradation_check_exception file</code>		Set degradation check exception list.
<code>degradation_check type</code>		Specify the message type(warning or error) for degradation_check. Possible values are warn and error.
<code>degradation_check parameter</code>		Specify the degradation_check parameter's name. Possible values are deltad, dvth, didlin, didsat, did, dgm and dgds.
<code>degradation_check value</code>	<code>DBL_MAX</code>	Specify the degradation_check value.
<code>degradation_check agelevel</code>	<code>-1</code>	Specify the degradation_check agelevel.
<code>degradation_check error</code>		Set degradation check error message output file.
<code>degradation_check_output file</code>		Set degradation check result output file.
<code>rel_mode type</code>		Specifies the analysis type used for obtaining the device reliability values. Possible values are aging_she, all, aging_she and aging_thermal.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>rel_mode tmode</code>		Specifies the temperature mode. Possible values are static and dynamic.
<code>gradual_aging_wi th_deg value</code>	<code>no</code>	Call <code>calcDegradation</code> before <code>calcAge</code> in gradual aging flow. Possible values are no and yes.
<code>degradation_chec k sub</code>	<code>[...]</code>	Set subckt for degradation check.
<code>degradation_chec k dev</code>	<code>[...]</code>	Set device for degradation check.
<code>degradation_chec k mod</code>	<code>[...]</code>	Set model for degradation check.
<code>output_she_power value</code>	<code>no</code>	Set output device power in SHE flow. Possible values are no and yes.
<code>enable_tmishe_ur i value</code>	<code>no</code>	Enable TMI and URI flow and pass <code>dtemp</code> of TMI into URI. Possible values are no and yes.
<code>output_format type</code>		Specifies reliability output format: SQL, DPL or BIN.. Possible values are SQL, DPL and BIN.
<code>skip_tmi_aging_s he_tran value</code>		The flag whether to run transient of aging+SHE. Possible values are no and yes.
<code>output_device_de grad file</code>		Specifies file included <code>output_device_degrad</code> options.
<code>vdsmethod type</code>	<code>external</code>	Specifies to get external or internal voltage of Vds. Possible values are external and internal.
<code>output_device_de grad skipaging</code>	<code>no</code>	Output <code>bt0</code> file and skip aging analysis for saving time. Possible values are no and yes.
<code>output_device_de grad vthsat</code>	<code>no</code>	Output <code>vthsat</code> for the device degradation in <code>bt0</code> file. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

output_device_degrad op	[...]	Specifies op parameters for device degradation in bt0 file..
output_subckt_degrad op	[...]	Specifies op parameters for subckt degradation in bt0 file..
output_device_degrad vthmod		Specifies method how to calculate vth. Possible values are cmi, max_gm and const_current.
output_device_degrad vgs		Sets the bias voltage of vgs for the dc sweep.
deg_num_output_device_degrad top	100	Output top_deg_num list for the device degradation in bo0 file.
deg_num_output_subckt_degrad top	100	Output top_deg_num list for the subckt degradation in bo0 file.
deg_table_line_break value	yes	Specifies whether to break line when device name is too long in degradation table. Possible values are no and yes.
param_output_inst_param sort		Specify the parameter whose value is to be used to sort output order.
params_output_inst_param sum	[...]	Specify the parameters the sum of whose values must be printed in the output file.
output_subckt_degrad vdd	[...]	Sets the bias voltage for the subckt degradation.
output_subckt_degrad vmlin	[...]	Sets the bias voltage of vmlin for the subckt degradation.
output_subckt_degrad vgsat	[...]	Sets the bias voltage of vgsat for the subckt degradation.
output_subckt_degrad vglin	[...]	Sets the bias voltage of vglin for the subckt degradation.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

output_subckt_degrad sub	[...]	Sets the subckt name to do subckt degradation.
output_subckt_degrad node	[...]	Sets subckt terminal nodes mapping to drain, gate, source and bulk for subckt degradation.
enable_tmi_uri tmishe	no	Enable TMI and URI flow and pass dtemp of TMI into URI. Possible values are no and yes.
output_subckt_degrad file		Specifies file included output_subckt_degrad options.
output_subckt_degrad ivthp	0.0	Sets PMOS vth current, it has higher priority than ivth defined in global option.
output_subckt_degrad ivthn	0.0	Sets NMOS vth current, it has higher priority than ivth defined in global option.
output_subckt_degrad ivth	[...]	Sets constant vth current for specified model.
degsort value	no	Specifies to enable to sort degradation data in decreasing order. Possible values are no and yes.
agelevelonly type	include	Includes or excludes the specified aging levels. Possible values are include and exclude.
preset age	0.0	Specifies age value for preset.
preset agelevel		Specifies agelevel for preset.
preset mod	[...]	Specifies model list for preset.
preset dev	[...]	Specifies device list for preset.
preset deg	0.0	Specifies degradation value for preset.
preset lifetime	0.0 s	Specifies lifetime value (unit is second) for preset.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

presct sub	[...]	Specifies subckt list for presct.
urilib scale_mode	original	Specifies parameter scale mode for URI library. The value can be original or effective. If value is original, send original parameter value to URI for param scale, else send effective value. Possible values are original and effective.
gradual_aging_pa ss_param value	no	Whether to pass URI instance state to next gradual aging step. Possible values are no and yes.
gradual_aging_ea rly_exit value	no	Whether to enable early exit in gradual aging flow. Possible values are no and yes.
win_relxtran mult	[...]	Specifies multiple time windows for reliability analysis.
gradual_aging_ea rly_exit time		gradual aging time from which early exit is triggered.
output_subckt_de grad correct_bias_volt age	yes	Specifies whether to correct the sign of voltage bias defined in output_subckt_degrad for reliability analysis. Possible values are no and yes.
output_subckt_de grad output_bias_volt age	no	Specifies whether to output the subckt names and their bias conditions defined in output_subckt_degrad to bt0 and bt0.dpl files. Possible values are no and yes.
output_device_de grad correct_bias_volt age	yes	Specifies whether to correct the sign of voltage bias defined in output_device_degrad for reliability analysis. Possible values are no and yes.
output_device_de grad output_bias_volt age	no	Specifies whether to output the model names and their bias conditions defined in output_device_degrad to bt0 and bt0.dpl files. Possible values are no and yes.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

output_device_degrad keep_zero_result		Output all the devices included devices of zero degradation in bt0 file. Possible values are no and yes.
enable_bias_runaway value	no	Enable bias runaway detecting flow. Possible values are no and yes.
scaleparam_urilib disable	no	Specifies whether need to scale params, the option only works in stress only mode. Possible values are no and yes.
depend_type_urilib bias	all	Specifies the bias dependent type for URI model.. Possible values are all, voltage and none.
depend_params_urilib bias	[...]	Specifies the bias dependent params for URI model..
summary_config_output_device_degrad degradation		Specifies file included degradation_summary_config options.
urilib appendage_check	yes	Specifies the flag that indicates whether a validity check should be run for the given appendage model.. Possible values are no and yes.

#### Detailed Description and Examples:

age time=[...]

The duration in the future at which the transistor degradation and degraded SPICE model parameters are to be calculated. The degraded SPICE model parameters are used in aged circuit simulation. The calculated transistor degradation can be transconductance, linear or saturation drain current, degradation, threshold voltage shift, and or any other degradation monitor. While specifying the value, attach the suffix y (year), h (hour) or m (minute). There should be no space between the number and suffix. For example, 10m, 1e-5sec. Note: Currently, specifying multiple time values is not supported.

deltad value=<deltad\_value>

Requests the calculation of lifetime for each transistor under the circuit operating conditions using the specified degradation value. You can use multiple deltad statements for different

## Spectre Circuit Simulator Reference

### Analysis Statements

---

types of transistors. The degradation value can be transconductance, linear or saturation drain current degradation, threshold voltage shift, or any other degradation monitor, depending on the definitions of the lifetime parameters H and m. `deltad_value` can be in decimal notation (xx.xx) or in engineering notation (x.xxe+xx).

`maskdev type={include | exclude} {sub=<subckt_list> | mod=<model_list> | dev=<device_list>}`

Includes or excludes the specified devices or the devices that belong to the listed subcircuit or model during reliability analysis. Note: Starting from the MMSIM10.1 version, the include and exclude values are mutually exclusive.

`report_model_param value={yes | no}(defaults to no)`

Determines whether to print the stress and aged parameters in the .bm# file. Possible values are yes and no. When set to yes, the stress and aged parameters are printed to the .bm# file.

`accuracy level={1 | 2}(defaults to 1)`

Specifies methods used in the reliability simulation when performing integration and substrate current calculation. In other words, specifies trapezoidal integration when performing integrations and calculates  $I_{sub}$  for  $V_{gs} < V_{th}$ . When set to 1, the software uses backward Euler integration and sets  $I_{sub}=0$  when  $V_{gs} < V_{th}$ . When set to 2, the software uses trapezoidal integration and calculates  $I_{sub}$  when  $V_{gs} < V_{th}$ . Setting accuracy to 2 is more accurate, but increases simulation time when compared to when accuracy is set to 1.

`minage value=<minage_value>`

Specifies the smallest Age value for which degraded SPICE model parameters are calculated. This statement speeds up aging calculation by using stress SPICE model parameters if the transistor Age value is smaller than the specified `minage_value`. `minage_value` can be in decimal notation (xx.xx) or in engineering notation (x.xxe+xx).

`igatemethod type={calc | spice}(defaults to calc)`

Specifies the method used for obtaining the gate terminal current of a MOSFET. During MOSFET HCI simulation, the gate terminal current is required for calculating the degradation value. The simulator can either calculate this value or obtain it from the SPICE output rawfile if the SPICE simulator in use provides such an option. If this command is not used, the simulator calculates the gate terminal current by using its own model parameters. Possible values are calc (default) and spice. When calc is specified, the gate terminal current is calculated using the model parameters and when spice is specified, the gate terminal current value is obtained from the SPICE output rawfile.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

`idmethod type={ids | idrain}(defaults to ids)`

Specifies how the simulator obtains the drain current (Id) to perform reliability calculations. The following types of drain currents, which are available from SPICE, are supported by reliability analysis:

- \* Dynamic drain current (also called AC drain current) - this is the current that flows in to the drain node.

- \* Static drain current (also called channel drain current, DC drain current, or Ids) Possible values are ids (Ids static current ) or idrain (dynamic drain current). Default: ids.

`uri_lib file={"filename"} uri_mode={agemos | appendage} debug={0 | 1}`

Loads the Unified Reliability interface (URI) shared library and specifies which method (uri\_mode) should be used to perform aging simulation. Note: appendage mode is not supported in the MMSIM10.1 release.

`relx_tran start=<start_time> stop=<stop_time>`

Specifies the start and stop time for reliability analysis during transient analysis. If stop\_time is not specified, the software stops in .tran statement.

`isubmethod type={calc | spice}(defaults to calc)`

Specifies the method used for obtaining substrate terminal current of a MOSFET. During MOSFET HCI simulation, the substrate terminal current is required for calculating the degradation value. The Spectre RelXpert simulator can either calculate this value or obtain it from the SPICE output rawfile if the SPICE simulator in use provides such an option. If this command is not used, the simulator calculates the substrate terminal current by using its own model parameters. Possible values are calc (default) and spice. When calc is specified, the substrate terminal current is calculated using the model parameters and when spice is specified, the substrate terminal current value is obtained from the SPICE output rawfile.n

`opmethod type={calc | spice}(defaults to calc)`

Specifies whether the Igate or Isub value should be obtained from the SPICE models (for example, BSIM3 or BSIM4) or the internal Igate or Isub equation should be used. Possible values are calc or spice. calc calculates the gate and substrate terminal current using the Cadence Igate and Isub model equations (Default). spice obtains the gate and substrate terminal current value from the SPICE model.

`degsort {threshold=<threshold_value> | number=<number_value>}`

## Spectre Circuit Simulator Reference

### Analysis Statements

---

Prints MOS transistors based on the threshold value or number settings. The results are sorted in the descending order of degradation. Note: The threshold and number arguments are mutually exclusive. Therefore, only one of them can be specified with degsort to print the sorted device degradation results. When threshold\_value is specified, the transistors having degradation values greater than threshold value are printed. The threshold value can be in decimal notation (xx.xx) or in engineering notation (x.xxe+xx). When number\_value is specified, only the first <number\_value> transistors having the highest degradations are printed. For example, if number=100, the software will print the first 100 transistors with highest degradations.

agelevel\_only value=[<level\_value> <model\_list>, <level\_value> <model\_list>, ...]

Specifies the age level for performing reliability analysis on the specified model(s). You can specify different age levels for different set of models. Note: This option also supports the URI defined agelevel statement. If model names are not specified, the simulation is performed on all of the devices at the specified age level. The following levels can be used to specify Cadence internal ageMOS models:

- \* 1: Specifies HCI reliability analysis.
- \* 2: Specifies NBTI reliability analysis.
- \* 3: Specifies PBTI reliability analysis.

### Example

```
rel reliability {  
    // reliability control statements  
    age time = [10h 20y 30y]  
    deltad value = 0.1  
    accuracy level = 2  
    agelevel_only value=[0 nch, 1 pch]  
    relx_tran start=1us stop=10us  
    idmethod type=idrain  
    maskdev type=include mod=[pch] dev=[mdut6] sub=[inv]  
    minage value = 0.00001  
    report_model_param value=yes  
    uri_lib file="./libURI.so" uri_mode=agemos debug=1  
    degsort number=100  
    igatemethod type=spice  
    isubmethod type=spice  
    opmethod type=spice  
    // stress/stress statements.
```

## Spectre Circuit Simulator Reference Analysis Statements

---

```
tran_stress tran start=0 step=1us stop=10us
// aging testbench statements.
change1 alter param=rel_temp value=125
// aging simulation statements.
tran_aged tran start = 0 step = 1us stop = 10us
}
```

## Deferred Set Options (set)

### Description

The deferred set options statement sets or changes various program control options. You can set the options in any order and, once set, the options retain their value until reset. The set statement is queued with all analyses and is executed sequentially (The changes made to these options are deferred until the statement setting them is encountered). To set `temp`, `tnom`, `scalem`, or `scale`, use the `alter` statement. For further options, see individual analyses.

### Syntax

```
Name set parameter=value ...
```

### Parameters

#### Tolerance parameters

<code>reltol</code>	<code>0.001</code>	Relative convergence criterion.
<code>residualtol</code>	<code>1.0</code>	Tolerance ratio for residual (multiplies <code>reltol</code> ).
<code>vabstol</code>	<code>1.0e-6 V</code>	Voltage absolute tolerance convergence criterion.
<code>iabstol</code>	<code>1.0e-12 A</code>	Current absolute tolerance convergence criterion.

#### Temperature parameters

<code>tempeffects</code>	<code>all</code>	Temperature effect selector. If <code>tempeffect = vt</code> , only thermal voltage varies with temperature; if <code>tempeffect = tc</code> , parameters that start with <code>tc</code> are active and thermal voltage is dependent on temperature; and if <code>tempeffect = all</code> , all built-in temperature models are enabled. Possible values are <code>vt</code> , <code>tc</code> and <code>all</code> .
--------------------------	------------------	--

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Convergence parameters

homotopy	all	Method used when no convergence on initial attempt of DC analysis. Users can specify methods and their orders by giving vector setting such as homotopy=[source ptran gmin]. Possible values are none, gmin, source, dptran, ptran, all, arclength and tranrampup.
newton	normal	Method used on DC analysis. Users can specify methods newton=[none   normal], and the default value is normal.. Possible values are none and normal.
limit	dev	Limiting algorithms to aid DC convergence. Possible values are delta, log, dev and l2.
gmethod	dev	Stamp gdev, gnode or both in the homotopy methods (other than dptran). See below for more information. Possible values are dev, node and both.
dptran_gmethod	node	Stamp gdev, gnode or both in the dptran (homotopy) methods. Possible values are dev, node and both.
try_fast_op	yes	This feature often speeds up the DC solution. For hard to converge designs, this feature fails and other methods are applied. In corner cases, this feature may have negative effects. If the DC analysis is unusually slow or if the memory usage of various processes keeps growing or if DC gets stuck even before homotopy methods start, try setting this option to no.. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Component parameters

<code>compatible</code>	<code>spectre</code>	Encourage device equations to be compatible with a foreign simulator. This option does not affect input syntax. Possible values are <code>spectre</code> , <code>spice2</code> , <code>spice3</code> , <code>cdsspace</code> , <code>hspice</code> , <code>spiceplus</code> , <code>eldo</code> , <code>sspace</code> , <code>mica</code> , <code>tispace</code> and <code>pspace</code> .
<code>approx</code>	<code>no</code>	Use approximate models. Difference between approximate and exact models is generally very small. Possible values are <code>no</code> and <code>yes</code> .

#### Error-checking parameters

<code>diagnose</code>	<code>no</code>	Print additional information that might help diagnose accuracy and convergence problems. Possible values are <code>no</code> and <code>yes</code> .
<code>opptcheck</code>	<code>yes</code>	Check operating point parameters against soft limits. Possible values are <code>no</code> and <code>yes</code> .

#### Resistance parameters

<code>gmin</code>	<code>1e-12 S</code>	Minimum conductance across each nonlinear device.
<code>gmin_check</code>	<code>max_v_only</code>	Specifies that effect of <code>gmin</code> should be reported if significant. Possible values are <code>no</code> , <code>max_v_only</code> , <code>max_only</code> and <code>all</code> .
<code>gmindc</code>	<code>1e-12 S</code>	Minimum conductance across each nonlinear device in DC analysis. If <code>gmindc</code> is not given explicitly, the value of <code>gmindc</code> will be equal to <code>gmin</code> . Default value is <code>1.0e-12</code> .
<code>rforce</code>	<code>1 Ω</code>	Resistance used when forcing nodesets and node-based initial conditions.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Quantity parameters

<code>quantities</code>	<code>no</code>	Print quantities. If <code>quantities=min</code> , the simulator will print out all defined quantities; if <code>quantities=full</code> , the simulator will also print a list of nodes and their quantities. Possible values are <code>no</code> , <code>min</code> and <code>full</code> .
-------------------------	-----------------	--

#### Annotation parameters

<code>narrate</code>	<code>yes</code>	Narrate the simulation. Possible values are <code>no</code> and <code>yes</code> .
<code>annotate</code>	<code>no</code>	Degree of annotation. Possible values are <code>no</code> and <code>title</code> .
<code>debug</code>	<code>no</code>	Give debugging messages. Possible values are <code>no</code> and <code>yes</code> .
<code>info</code>	<code>yes</code>	Give informational messages. Possible values are <code>no</code> and <code>yes</code> .
<code>note</code>	<code>yes</code>	Give notice messages. Possible values are <code>no</code> and <code>yes</code> .
<code>maxnotes</code>	<code>5</code>	Maximum number of times a notice is issued per analysis. Note that this option has no effect on notices issued as part of parsing the netlist. Use the <code>-maxnotes</code> command-line option to control the number of all notices issued.
<code>warn</code>	<code>yes</code>	Give warning messages. Possible values are <code>no</code> and <code>yes</code> .
<code>maxwarns</code>	<code>5</code>	Maximum number of times a warning message is issued per analysis. Note that this option has no effect on warnings issued as part of parsing the netlist. Use the <code>-maxwarns</code> command-line option to control the number of all warnings issued.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

maxwarnstologfile	5	Maximum number of times a warning message is printed to the log file per analysis. Note that this option has no effect on warnings printed as part of parsing the netlist. Use the -maxwarnstolog command-line option to control the number of all warnings printed to the log file.
maxnotestologfile	5	Maximum number of times a notice message is printed to the log file per analysis. Note that this option has no effect on notices printed as part of parsing the netlist. Use the -maxnotestolog command-line option to control the number of all notices printed to the log file.
error	yes	Generates error messages. Possible values are no and yes.
digits	5	Number of digits used when printing numbers.
measdgt	0	Number of decimal digits in floating point numbers in measurement output in mt0 format.
notation	eng	The notation to be used for displaying real numbers to the screen. Possible values are eng, sci and float.

#### Matrix parameters

pivotdc	no	Use numeric pivoting on every iteration of DC analysis. Possible values are no and yes.
pivrel	0.001	Relative pivot threshold.
pivabs	0	Absolute pivot threshold.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>preorder</code>	<code>partial</code>	Try this option when simulation runs out of memory or if the simulation is unreasonably slow for the size of your design. It controls the amount of matrix reordering that is done and may lead to much fewer matrix fill-ins in some cases. Known cases are: designs with very large number of small resistors or large number of behavioral instances containing voltage based equations.. Possible values are partial, full and limited.
<code>limit_diag_pivot</code>	<code>yes</code>	If set to yes, there is a limit on the number of matrix fill-ins when selecting a pivot from a diagonal. For backward compatibility set this to no.. Possible values are no and yes.
<code>rebuild_matrix</code>	<code>no</code>	If yes, rebuild circuit matrix at the beginning of ac, acmatch, dc, dcmatch, montecarlo, pz, stb, sweep, tdr, and tran analyses. This is to ensure consistent matrix ordering at the beginning of the analyses for consistent results. Notice that rebuild circuit matrix can incur performance overhead. Possible values are no and yes.
<code>icversion</code>	<code>1</code>	Convert initial condition to initial guess, when .ic statements exist in netlist and there are no other options to set IC or nodeset.
<code>minstepversion</code>	<code>1</code>	Minstep algorithm flow control. If minstepversion=0, desperate big jump is taken when minstep is reached. If minstepversion=1, a forward/backward search algorithm is taken to find a converged solution at small step size. By default, minstepversion=1.
<code>try_hard_in_disaster</code>	<code>yes</code>	When minstepversion=1, this option means whether simulator should try hard to search for a converged solution in disaster stage. Possible values are no and yes.

**Shell Command (shell)**

**Description**

The shell analysis passes a command to the operating system command interpreter given in the SHELL environment variable. The command behaves as if it were typed into the command interpreter, except that any %X codes in the command are expanded first.

The default action of the shell analysis is to terminate the simulation.

**Syntax**

Name shell parameter=value ...

**Parameters**

cmd	kill %P	Shell command.
iferror	quit	Action to be taken if the command returns nonzero error status. Possible values are continue and quit.
annotate		Degree of annotation. Possible values are no, title and yes.

## S-Parameter Analysis (sp)

### Description

The S-parameter analysis linearizes the circuit about the DC operating point and computes S-parameters of the circuit taken as an N-port. The port statements define the ports of the circuit. Each active port is turned on sequentially, and a linear small-signal analysis is performed. Spectre converts the response of the circuit at each active port into S-parameters and outputs these parameters. There must be at least one active port statement in the circuit.

Sp analysis can output an ASCII model file that can later be read-in by a `nport` component. The file name and the model type can be specified by `file` and `paramtype` parameters, respectively. Currently, only `paramtype=s` and `paramtype=y` are supported, for the other possible values of `paramtype`, the file still generates s-parameters. The text format can be either Spectre native or Touchstone.

Spectre supports probes in sp analysis. Sprobe is a special testbench to allow in-situ probing of bi-directional impedances. If the 'sprobes' parameter is given, the parameters S1, S2, Z1, Z2, StabIndex of specified probe will be computed and outputted to rawfile. Here S1 is the scattering parameter looking left from the probe; S2 is the scattering parameter looking right from the probe; Z1 is the impedance looking left from the probe; Z2 is the impedance looking right from the probe; StabIndex is the stability Index. If sprobes and ports are given, 'ports' are used to compute normal S-param, while 'sprobes' are used to compute probe params.

If 'sprobeports' set, a different probe mode is enabled. It can calculate the complete set of network parameters for the circuit looking at the left of probe and the circuit looking at the right of the probe. In this mode, probe is open.

Spectre can perform AC analysis while sweeping a parameter. The parameter can be frequency, temperature, component instance parameter, component model parameter, or netlist parameter. If changing a parameter affects the DC operating point, the operating point is recomputed at each step. You can sweep the circuit temperature by giving the parameter name as `temp`, without a `dev` or `mod` parameter. In addition, you can sweep a netlist parameter by giving the parameter name without a `dev` or `mod` parameter. After the analysis is complete, the modified parameter returns to its original value.

### Syntax

```
Name sp parameter=value ...
```

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Parameters

<code>prevoppoint</code>	<code>no</code>	Use the operating point computed in the previous analysis. Possible values are no and yes.
--------------------------	-----------------	--

#### Sweep interval parameters

<code>start</code>	<code>0</code>	Start sweep limit.
<code>stop</code>		Stop sweep limit.
<code>center</code>		Center of sweep.
<code>span</code>	<code>0</code>	Sweep limit span.
<code>step</code>		Step size, linear sweep.
<code>lin</code>	<code>50</code>	Number of steps, linear sweep.
<code>dec</code>		Points per decade.
<code>log</code>	<code>50</code>	Number of steps, log sweep.
<code>values</code>	<code>[...]</code>	Array of sweep values.
<code>valuesfile</code>		Name of the file containing the sweep values.

#### Sweep variable parameters

<code>dev</code>	Device instance whose parameter value is to be swept.
<code>mod</code>	Model whose parameter value is to be swept.
<code>param</code>	Name of parameter to sweep.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>freq</code>	(Hz)	Frequency when a parameter other than frequency is being swept.
-------------------	------	---

#### Port parameters

<code>ports</code>	[...]	List of active ports. Ports are numbered in the order given.
--------------------	-------	--

#### State-file parameters

<code>readns</code>		File that contains an estimate of the DC solution (nodeset).
---------------------	--	--

<code>write</code>		DC operating point output file at the first step of the sweep.
--------------------	--	--

<code>writefinal</code>		DC operating point output file at the last step of the sweep.
-------------------------	--	---

#### Initial condition parameters

<code>force</code>	<code>none</code>	The set of initial conditions to use. Possible values are none, node, dev and all.
--------------------	-------------------	--

<code>readforce</code>		File that contains initial conditions.
------------------------	--	--

<code>skipdc</code>	<code>no</code>	Skip DC analysis. Possible values are no and yes.
---------------------	-----------------	---

<code>useprevic</code>	<code>no</code>	If set to yes or ns, use the converged initial condition from previous analysis as ic or ns. Possible values are no, yes and ns.
------------------------	-----------------	--

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Output parameters

<code>file</code>		Output file name.
<code>mode</code>	<code>ss</code>	S-parameters mode selector. Can be <code>mm</code> for mixed-mode.
<code>datafmt</code>	<code>spectre</code>	Data format of the S-parameter output file. Possible values are <code>spectre</code> , <code>touchstone</code> and <code>touchstone2</code> .
<code>paramtype</code>	<code>s</code>	Output parameter type. If set to <code>s</code> , the S-parameters will be output to raw data. If <code>file</code> is specified, the S-parameters will be output to the file. If set to <code>y</code> , both S-parameters and Y-parameters will be output to raw data. If <code>file</code> is specified, the normalized Y-parameters will be output to the file. If set to <code>z</code> , the behavior is similar to <code>y</code> . If set to <code>yz</code> , then S-parameters, Y-parameters, and Z-parameters will be output to raw data. However, if <code>file</code> is specified, only the S-parameters will be output to the file. Possible values are <code>s</code> , <code>y</code> , <code>z</code> and <code>yz</code> .
<code>datatype</code>	<code>realimag</code>	Data type of the S-parameter output file. Possible values are <code>realimag</code> , <code>magphase</code> and <code>dbphase</code> .
<code>noisedata</code>	<code>no</code>	Should noise data be saved to the S-parameter output file; if yes, in what format. Possible values are <code>no</code> , <code>twoport</code> and <code>cy</code> .
<code>oppoint</code>	<code>no</code>	Determines whether operating point information should be computed:if yes, where should it be printed (screen or file). Operating point information is not printed if the operating point computed in the previous analysis remains unchanged. Possible values are <code>no</code> , <code>screen</code> , <code>logfile</code> and <code>rawfile</code> .



## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Noise parameters

<code>donoise</code>	<code>no</code>	Perform noise analysis. If <code>oprobe</code> is specified as a valid port, this is set to yes, and a detailed noise output is generated. Possible values are no and yes.
<code>oprobe</code>		Compute total noise at the output defined by this component.
<code>iprobe</code>		Input probe. Refer the output noise to this component.

#### Convergence parameters

<code>restart</code>	<code>yes</code>	Restart the DC solution from scratch if any condition has changed. If not, use the previous solution as an initial guess. Possible values are no and yes.
----------------------	------------------	---

#### Annotation parameters

<code>annotate</code>	<code>sweep</code>	Degree of annotation. Possible values are no, title, sweep, status and steps.
<code>title</code>		Analysis title.

#### Sprobe parameters

<code>sprobes</code>	<code>[...]</code>	List of s-probes.
<code>sprobeports</code>	<code>[...]</code>	List of sprobeport.

If the list of active ports is specified with the `ports` parameter, the ports are numbered sequentially from one in the order given. Otherwise, all ports present in the circuit are active, and the port numbers used are those that were assigned on the port statements. If `donoise=yes` is specified, the noise correlation matrix is computed. If in addition, the output is specified using `oprobe`, the amount that each noise source contributes to the output is

## Spectre Circuit Simulator Reference

### Analysis Statements

---

computed. Finally, if an input is also specified (using `iprobe`), the two-port noise parameters are computed (`F`, `Fmin`, `NF`, `NFmin`, `Gopt`, `Bopt`, and `Rn`).

Spectre will output file in touchstone Version 2.0 format when `datafmt=touchstone2`. Network data for Y- and Z-parameters in touchstone Version 2.0 files is not normalized, but in spectre and touchstone version 1.0 file, it is normalized.

If the `mode` parameter is set to `mm`, differential and common-mode S-parameters (denoted as mixed-mode S-parameters) are calculated. When `mode=mm`, there must be  $2N$ , with  $N > 1$ , active port statements in the circuit. The mixed-mode S-parameters are calculated referring to the pairing of the ports, with the port numbers ordered in pair as (1,2) (3,4), and so on in the ports list. With `mm`, Spectre calculates differential-to-differential, differential-to-common, common-to-differential, and common-to-common S-parameters. A combination of mixed-mode and standard S-parameters is calculated if the 'mode' parameter is set to, say, `m12m34s5`. Then, additional differential-to-standard, common-to-standard, standard-to-differential, and standard-to-common S-parameters are calculated. In the example of `mode=m12m34s5`, the standard single-end port is port number 5, the two mixed-mode port pairs are (1,2) and (3,4); with Spectre placing restriction of the number on active ports to 5 given in the port list. [Mixed-Mode Order] keyword is used in touchstone Version 2.0 files to describe mixed-mode network parameters order. If `mode=m12m34s5`, [Mixed-Mode Order] is set to `D1,2 D3,4 C1,2 C3,4 S5`.

`sprobeports` is used to specify the `sprobeport` instances corresponding to `sprobes`. `sprobeport` is used to set ports in the circuit looking to the left of `sprobe` with `portL` and set ports in the circuit looking to the right of `sprobe` with `portR`. To ensure that `sprobeports` setting works well even when some `sprobe` has no `sprobeport` set, set `sprobeport` for this `sprobe` to a blank string. Example:

```
sp sp sprobes=[spb1 spb2 spb3] sprobeports=["" spb_port2 ""]
```

```
spb_port2 sprobeport portL=[...] portR=[...]
```

`spb_port2` is for `sprobe spb2`. `Sprobe spb1` and `spb3` have no ports setting.

When `portL` is set for a `sprobeport`, S-Parameters for ports in `portL` and the left port of the corresponding `sprobe` will be calculated. And when `portR` is set for a `sprobeport`, S-Parameters for the right port of the corresponding `sprobe` and ports in `portR` will be calculated.

When `donoise=yes`, two-port noise parameters will be calculated. For circuit looking to the left of `sprobe`, the 1st port in `portL` is used as in-port, the left port of `sprobe` is used as out-port. For circuit looking to the right of `sprobe`, the 1st port in `portR` is used as out-port, the right port of `sprobe` is used as in-port.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

You can define sweep limits by specifying the end points or the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. You can specify a step size parameter (`step`, `lin`, `log` or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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. All frequencies are in Hertz.

The small-signal analysis begins by linearizing the circuit about an operating point. By default, this analysis computes the operating point, if it is not known, or recomputes it if any significant component or circuit parameter has changed. However, if an operating point was computed during a previous analysis, you can set `prevoppoint=yes` to avoid recomputing it. For example, if `prevoppoint=yes` and the previous analysis was a transient analysis, the operating point is the state of the circuit at the final time point.

## Stability Analysis (stb)

### Description

The STB analysis linearizes the circuit about the DC operating point and computes the loop gain and gain and phase margins (if the sweep variable is frequency) for a feedback loop or a gain device.

Spectre can perform the analysis while sweeping a parameter. The parameter can be frequency, temperature, component instance parameter, component model parameter, or netlist parameter. If changing a parameter affects the DC operating point, the operating point is recomputed at each step. You can sweep the circuit temperature by giving the parameter name as `temp`, without a `dev` or `mod` parameter. You can sweep a netlist parameter by giving the parameter name without a `dev` or `mod` parameter. After the analysis is complete, the modified parameter returns to its original value.

### Syntax

```
Name stb parameter=value ...
```

### Parameters

<code>prevoppoint</code>	<code>no</code>	Use the operating point computed on the previous analysis. Possible values are <code>no</code> and <code>yes</code> .
--------------------------	-----------------	---

#### Sweep interval parameters

<code>start</code>	<code>0</code>	Start sweep limit.
<code>stop</code>		Stop sweep limit.
<code>center</code>		Center of sweep.
<code>span</code>	<code>0</code>	Sweep limit span.
<code>step</code>		Step size, linear sweep.
<code>lin</code>	<code>50</code>	Number of steps, linear sweep.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

dec		Points per decade.
log	50	Number of steps, log sweep.
values	[...]	Array of sweep values.
valuesfile		Name of the file containing the sweep values.

#### Sweep variable parameters

dev		Device instance whose parameter value is to be swept.
mod		Model whose parameter value is to be swept.
param		Name of parameter to sweep.
freq	(Hz)	Frequency when parameter other than frequency is being swept.

#### Probe parameters

probe		Probe instance around which the loop gain is calculated.
localgnd		Node name of local ground. If not specified, the probe is referenced to global ground.

#### State-file parameters

readns		File that contains estimate of DC solution (nodeset).
write		DC operating point output file at the first step of the sweep.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>writefinal</code>		DC operating point output file at the last step of the sweep.
-------------------------	--	---

#### Initial condition parameters

<code>force</code>	<code>none</code>	Which set of initial conditions to use. Possible values are none, node, dev and all.
--------------------	-------------------	--

<code>readforce</code>		File that contains initial conditions.
------------------------	--	--

<code>skipdc</code>	<code>no</code>	Skip the DC analysis. Possible values are no and yes.
---------------------	-----------------	---

<code>useprevic</code>	<code>no</code>	If set to yes or ns, use the converged initial condition from previous analysis as ic or ns. Possible values are no, yes and ns.
------------------------	-----------------	--

#### Output parameters

<code>save</code>		Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, none and nooutput.
-------------------	--	---

<code>nestlvl</code>		Levels of subcircuits to output.
----------------------	--	----------------------------------

<code>oppoint</code>	<code>no</code>	Should operating point information be computed; if yes, where should it be sent. Operating point information would not be output if the operating point computed in the previous analysis remains unchanged. Possible values are no, screen, logfile and rawfile.
----------------------	-----------------	---

#### Convergence parameters

<code>restart</code>	<code>yes</code>	Restart the DC solution from scratch if any condition has changed. If not, use the previous solution as initial guess. Possible values are no and yes.
----------------------	------------------	--

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Annotation parameters

<code>annotate</code>	<code>sweep</code>	Degree of annotation. Possible values are no, title, sweep, status and steps.
<code>title</code>		Analysis title.
<code>margininfo_unstable</code>	<code>no</code>	Print GainMargin and PhaseMargin info when the circuit is actually not stable. Possible values are no and yes.
<code>diffprobe</code>		DiffProbe instance around which the loop gain is calculated.
<code>swap</code>		Swap decides the connection of diffprobes to the circuit.
<code>mode</code>		Mode decides the calculation mode for diffprobes, e.g. CM for common-mode or DM for differential mode.

You can define sweep limits by specifying the end points or the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. You can specify a step size parameter (`step`, `lin`, `log` or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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. All frequencies are in Hertz.

The small-signal analysis begins by linearizing the circuit about an operating-point. By default this analysis computes the operating point, if it is not known, or recomputes it if any significant component or circuit parameter has changed. However, if a previous analysis computed an operating point, you can set `prevoppoint=yes` to avoid recomputing it. For example, if you use this option when the previous analysis was a transient analysis, the operating point is the state of the circuit on the final time point.

Nodesets help find the DC or the initial transient solution. You can specify nodesets in the circuit description file with `nodeset` statements or in a separate file by using the `readns` parameter. When nodesets are specified, Spectre computes an initial guess of the solution by performing DC analysis, while forcing the specified values on to nodes by using a voltage source in series with a resistor whose resistance is `rforce`. Spectre then removes these voltage sources and resistors and computes the required solution from this initial guess.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

Nodesets have two important uses. First, if a circuit has two or more solutions, nodesets can bias the simulator towards computing the required solution. Second, this is a convergence aid. By estimating the solution of the largest possible number of nodes, you might be able to eliminate a convergence problem or significantly speed up convergence.

When you simulate the same circuit multiple times, it is recommended that you use both `write` and `readns` parameters and assign the same file name to both parameters. DC analysis then converges quickly even if the circuit has changed since the last simulation, and the nodeset file is automatically updated.

During the initial operating point DC analysis, you may force some of the circuit variables to the values given in the `ic` file, `ic` statements, or `ic` parameter on the capacitors and inductors. The `ic` parameter controls the interaction of the various methods for setting the force values. The effects of individual settings are as follows:

`force=none`: Any initial condition specifiers are ignored.

`force=node`: The `ic` statements are used, and the `ic` parameter on the capacitors and inductors are ignored.

`force=dev`: The `ic` parameters on the capacitors and inductors are used, and the `ic` statements are ignored.

`force=all`: Both the `ic` statements and the `ic` parameters are used, with the `ic` parameters overriding the `ic` statements.

If you specify an `ic` file with the `readforce` parameter, force values from the file are used, and any `ic` statements are ignored.

After you specify the initial conditions, Spectre computes the DC operating point with the specified nodes forced to the given value by using a voltage source in series with a resistor whose resistance is `rforce` (see `options`).

### Understanding Loop-Based and Device-Based Algorithms

Two algorithms-loop-based and device based are available for small-signal stability analysis. Both algorithms are based on the calculation of Bode's return ratio. Loop gain waveform, gain margin, and phase margin are the analysis output.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

The `probe` parameter must be specified to perform stability analysis. When it points to a current probe or voltage source instance, the loop-based algorithm is run; when it points to a supported active device instance, the device-based algorithm is run.

#### Loop-Based Algorithm

The loop-based algorithm calculates the true loop gain, which consists of normal loop gain and reverse loop gain. The loop-based algorithm requires the `probe` being placed 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.

The loop-based algorithm provides accurate stability information for single loop circuits and also for multi-loop circuits in which a `probe` component can be placed on a critical wire to break all loops. For a general multi-loop circuit, such a critical wire may not be available. The loop-based algorithm can only be performed on individual feedback loops to ensure that they are stable. Although the stability of all feedback loops is only 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.

For the loop based algorithm, a probe needs to be placed on the feedback loop. The common way is to insert a current probe or voltage source instance in the netlist, and the `probe` parameter points to this instance. The second way to insert the probe is by specifying the terminal of some instance.

The syntax format :

`probe = X:n`

`x` can be an instance or a subcircuit instance and `n` is the terminal index. The second way has a limitation: We can not specify a branch whose current consists of more than one instance's terminal current. A current probe or voltage source instance needs to be inserted manually.

This can be shown below:

```

      _____
      |----|___|-----|
      |-----||-----|
      |           |<--- probe
      |  ___   |___|---||---|
      |---|___|--||___   |

```

## Spectre Circuit Simulator Reference

### Analysis Statements

---

```

===      | |      |
|-----|-----|

```

No instance can specify the probe's location.

#### Device-Based Algorithm

The device-based algorithm calculates the loop gain around a particular active device. This algorithm is often applied to assess the stability of circuit design in which local feedback loops cannot be neglected; the loop-based algorithm cannot be performed for these applications because the local feedback loops are inside the devices and cannot be accessed from the schematic or netlist level to insert the `probe` component.

With the `probe` parameter pointing to a particular active device, the dominant controlled source in the device is nulled during the analysis. The dominant controlled source is defined as by nulling this source renders the active device to be passive. The device-based algorithm produces accurate stability information for a circuit in which a critical active device can be identified, so that nulling the dominant gain source of this device renders the whole network passive.

#### Stability Analysis of Differential Feedback Circuits

A balanced fully differential feedback circuit is illustrated below:

```

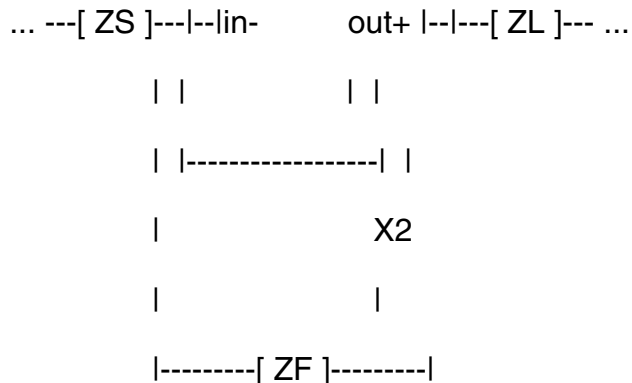
|-----[ ZF ]-----|
|                      |
|                      X1
| |-----| |
| |                      | |
... ---[ ZS ]---|--lin+      out- |--[ ZL ]--- ...
|      OPAMP      |

```

## Spectre Circuit Simulator Reference

### Analysis Statements

---



The feedback loops are broken at X1 and X2, with x1in and x2in being the input side nodes and x1out and x2out being the output side nodes. The following subcircuit connects these four nodes together:

```
subckt diffprobe x1in x2in x1out x2out
    ibranch inout x1out iprobe
    vinj inout x1in iprobe
    evinj x2in x2out x1in x1out vcvs gain=0
    fiinj 0 x2out pcccs probes=[ibranch vinj] coeffs=[0 1 1] gain=0
ends diffprobe
```

If the 'localgnd' parameter is specified, the above subcircuit should be modified as follows:

```
subckt diffprobe x1in x2in x1out x2out localgnd
    ibranch inout x1out iprobe
    vinj inout x1in iprobe
    evinj x2in x2out x1in x1out vcvs gain=0
    fiinj localgnd x2out pcccs probes=[ibranch vinj] coeffs=[0 1 1] gain=0
ends diffprobe
```

## Spectre Circuit Simulator Reference

### Analysis Statements

---

Let `diffprobe_inst` be the instance of subcircuit `diffprobe`, the following analysis measures the differential-mode loop gain:

```
DAlterv alter dev=diffprobe_inst.evinj param=gain value=-1
```

```
DAlteri alter dev=diffprobe_inst.fiinj param=gain value=-1
```

```
DMloopgain stb probe=diffprobe_inst.vinj
```

and, the following analysis measures the common-mode loop gain:

```
CAlterv alter dev=diffprobe_inst.evinj param=gain value=1
```

```
CAlteri alter dev=diffprobe_inst.fiinj param=gain value=1
```

```
CMloopgain stb probe=diffprobe_inst.vinj
```

## Spectre Circuit Simulator Reference

### Analysis Statements

---

## Reliability Stress Analysis (stress)

### Syntax

Name stress parameter=value ...

### Parameters

age_time	[...]	The duration in the future at which the transistor degradation and degraded SPICE model parameters are to be calculated.
deltad	[...]	Specifies the degradation value.
deltad_mod	[...]	Specifies the name of a model whose lifetime is calculated. The model name must be the same as specified in the .model card.
deg_ratio_type	include	Includes or excludes the specified devices or the device during TMI Aging Simulation flow. Possible values are include and exclude.
deg_ratio_dev	[...]	Specifies the instances to be included or excluded for degradation ratio for TMI Aging flow.
report_model_param	no	Determines whether to print the stress and aged parameters in the .bm# file. Possible values are no and yes.
accuracy	1	Specifies methods used in the reliability simulation when performing integration and substrate current calculation. Possible values are 1 and 2.
minage	0.0	Specifies the smallest Age value for which degraded SPICE model parameters are calculated.
igatemethod	calc	Specifies the method used for obtaining the gate currents of MOSFETs. Possible values are calc and spice.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>idmethod</code>	<code>ids</code>	Specifies how the simulator obtains the drain current (Id) of MOSFETs to perform reliability calculations. Possible values are <code>ids</code> , <code>idrain</code> and <code>idstatic</code> .
<code>dumpagemodel</code>		Output file name of dump age model.
<code>dumpagemodel_dev</code>	<code>[...]</code>	Specifies the instances whose aged modelcard will be output.
<code>urilib_file</code>		Specifies URI library file name.
<code>urilib_mode</code>	<code>agemos</code>	Specifies which method should be used to perform aging simulation. Note: appendage mode is not supported in the MMSIM10.1.0 release. Possible values are <code>agemos</code> , <code>scaleparam</code> , <code>mixed_mode</code> , <code>appendage</code> , <code>new_appendage</code> , <code>appendage2</code> and <code>appendage1</code> .
<code>urilib_debug</code>	<code>0</code>	Specifies the debug mode for URI library. The value can be 0 or 1. When specified, a flag is added to the URI library indicating whether the debug information should be printed. <code>debugMode=1</code> prints debug messages. Default value is 0.
<code>relxtran_start</code>	<code>0.0</code>	Specifies the start time of reliability analysis during transient simulation.
<code>relxtran_stop</code>	<code>0.0</code>	Specifies the stop time of reliability analysis during transient simulation. If <code>stop_time</code> is not specified, the software stops in <code>.tran</code> statement.
<code>isubmethod</code>	<code>calc</code>	Specifies the method used for obtaining the substrate currents of MOSFET. Possible values are <code>calc</code> and <code>spice</code> .
<code>opmethod</code>	<code>calc</code>	Specifies whether the <code>Igate</code> or <code>Isub</code> value should be obtained from the SPICE models (for example, BSIM3 or BSIM4) or whether the internal <code>Igate</code> or <code>Isub</code> equation should be used. Possible values are <code>calc</code> and <code>spice</code> .

## Spectre Circuit Simulator Reference

### Analysis Statements

---

degsort_threshold	0.0	Prints MOS transistors based on the threshold and number settings. The results are sorted in the descending order of degradation.
degsort_num	0	Prints only the first <number> transistors having the highest degradations. For example, if number=100, the software will print the first 100 transistors with highest degradations.
combine_deg	no	Determines whether to combine the external URI results with internal URI results in bo0 file. Possible values are no and yes.
keep_aged_data	yes	Determines whether to keep the value of the aged model parameters after reliability analysis. Possible values are no and yes.
check_neg_aging_type	error	Specifies the message type to check the negative value in bt0. Default is error. Possible values are error, warn and ignore.
check_neg_aging_clamp	no	Clamps the negative age value in the bt0 file. Possible values are no and yes.
tranflag	0	Specify the reliability transient analysis flag..
stress_tran_name	[...]	Specifies stress transient analysis name.
aging_tran_name	[...]	Specifies aging transient analysis name.
enablenegativeage	no	Enable the negative age value. Possible values are no and yes.
output_binary_file	no	Specifies reliability analysis output file format. Possible values are no and yes.
deg_ratio_hci	1.0	The degradation ratio for HCI effect.
deg_ratio_nbti	1.0	The degradation ratio for NBTI effect.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

deg_ratio_pbti	1.0	The degradation ratio for PBTI effect.
deg_ratio_bti	1.0	The degradation ratio for NBTI/PBTI effect.
resetanalysisparam	tran	Reset the analysis parameters in the reliability block. Possible values are tran, dc and ac.
urilib_type	inline	Compatible with RelXpert. Possible values are inline, sub and dev.
enable_ade_process	no	running reliability in ADE. Possible values are no and yes.
output_region_param	no	Prints the region method of the parameter. Possible values are no and yes.
deg_ratio_hcin	1.0	The degradation ratio for NMOS HCI effect.
deg_ratio_hcip	1.0	The degradation ratio for PMOS HCI effect.
deg_ratio_btin	1.0	The degradation ratio for PBTI effect.
deg_ratio_btip	1.0	The degradation ratio for NBTI effect.
deltad_item		Specify the electrical parameter for lifetime estimation. It is only available for TMI aging flow. Possible values are didsat, didlin, dvtlin, dvtsat and lifetime.
degsort_item	0	Specify which item is used for sorting. It is only available for TMI aging flow. Possible values are didsat, didlin, dvtlin, dvtsat and lifetime.
simmode		Mode of reliability analysis. Possible values are stress, aging and all.
simmode_file		File of simMode.
simmode_tmifile		Input file for TMI Aging flow.



**Spectre Circuit Simulator Reference**  
Analysis Statements

---

tmi_aging_mode		Mode of tmi aging flow. Possible values are aging, she and all.
degsort	no	Specifies to enable to sort degradation data in decreasing order. Possible values are no and yes.

## Sweep Analysis (sweep)

### Description

The `sweep` analysis sweeps a parameter, running the list of analyses (or multiple analyses) for each value of the parameter. The swept parameter can be circuit temperature, a device instance parameter, a device model parameter, a netlist parameter, or a subcircuit parameter for a particular subcircuit instance.

A set of parameters can be swept simultaneously, using the `paramset` parameter. The other sweep interval or variable parameters cannot be specified with the `paramset` parameter. Do `spectre -h paramset` for information on defining a `paramset`.

Within a sweep statement, you can specify analyses statements. These statements should be bound within braces. The opening brace is required at the end of the line defining the sweep. Sweep statements can be nested.

You can sweep the circuit temperature by giving the parameter name as `param=temp`, without a `dev`, `mod`, or `sub` parameter. You can sweep a top-level netlist parameter by giving the parameter name without a `dev`, `mod`, or `sub` parameter. You can sweep a subcircuit parameter for a particular subcircuit instance by specifying the subcircuit instance name with the `sub` parameter and the subcircuit parameter name with the `param` parameter. The same can be done using `dev` for the device instance name or `mod` for the device model name.

After the analysis is complete, the modified parameter returns to its original value.

### Syntax

```
Name sweep parameter=value ...
```

### Parameters

Sweep interval parameters

<code>start</code>	<code>0</code>	Start sweep limit.
<code>stop</code>		Stop sweep limit.
<code>center</code>		Center of sweep.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>span</code>	<code>0</code>	Sweep limit span.
<code>step</code>		Step size, linear sweep.
<code>lin</code>	<code>50</code>	Number of steps, linear sweep.
<code>dec</code>		Points per decade.
<code>log</code>	<code>50</code>	Number of steps, log sweep.
<code>values</code>	<code>[...]</code>	Array of sweep values.
<code>valuesfile</code>		Name of the file containing the sweep values.

#### Sweep variable parameters

<code>dev</code>		Device instance whose parameter value is to be swept.
<code>sub</code>		Subcircuit instance whose parameter value is to be swept.
<code>mod</code>		Model whose parameter value is to be swept.
<code>param</code>		Name of parameter to sweep.
<code>paramset</code>		Name of parameter set to sweep.
<code>paramtype</code>	<code>normal</code>	Define the type of the sweep variable only for loadpull analysis. Possible values are normal, pwr and freq.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Fault analysis parameters

<code>faults</code>	<code>[...]</code>	Names of the fault blocks from the list to perform direct fault analysis. This is required parameter for DFA, 'faults=[*]' includes all faults.
<code>nominal</code>	<code>yes</code>	Nominal (fault free) simulation will be performed during fault analysis. Possible values are no and yes.
<code>faultsid</code>	<code>[...]</code>	Indexes of faults from the list to perform fault analysis. If specified, simulation performed for requested subset of fault list.
<code>faultsname</code>	<code>[...]</code>	Names of faults from the list to perform fault analysis. If specified, simulation performed for requested subset of fault list.
<code>faultsinst</code>	<code>[...]</code>	List of instances to perform fault analysis. If specified, simulation performed only for faults within given instances.
<code>faultsamplenum</code>		Number of samples in random sampling of fault list during simulation.
<code>faultsamplratio</code>	<code>[R1 R2]</code>	A set of two values R1 and R2 (in percentage) to simulate the samples between the given range. 'faultsamplratio=R' is equivalent to 'faultsamplratio=[0 R]'.
<code>faultseed</code>	<code>1</code>	Optional starting seed for random number generator during fault sampling.
<code>faultsamplmethod</code>	<code>random</code>	Method to be used for fault sampling. Possible values are random, randomweighted, randomuniform and weightsorted.
<code>faultsamplereplacement</code>	<code>yes</code>	Perform fault sampling with sample replacement during fault analysis. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

faultdetratio		Auto-stop fault simulation when the specified detection ratio achieved.
faultconfidlevel		Auto-stop fault simulation when the specified detection ratio achieved with given confidence level.
faultcollapse	no	Perform fault collapsing in the list before fault simulation. Possible values are no and yes.
faultduplicate	yes	Perform fault simulation for all faults from the list. If 'no', identical and equivalent faults will be excluded from simulation. Possible values are no and yes.
faultsort	yes	Perform fault simulation according to the order in the fault list. If 'no', random order applied. Possible values are no and yes.

#### Annotation parameters

annotate	sweep	Degree of annotation. Possible values are no, title and sweep.
title		Analysis title.
distribute		Distribute a sweep analysis to reduce simulation time by using more computer cores across multiple computers. Possible values are no, fork, rsh, ssh, lsf, sge, nc and auto.
numprocesses		Specifies the number of jobs in distributed mode.
savedatainseparatedir	no	Whether to save data for each plot in a separate directory. savedatainseparatedir=name is only used for direct fault analysis (DFA). Possible values are no, yes and name.
component		component instance.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

You can specify sweep limits by specifying the end points or the center value and the span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. You can specify a step size parameter (`step`, `lin`, `log`, or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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.

Example:

```
swp sweep param=temp values=[-50 0 50 100 125] {  
    oppoint dc oppoint=logfile  
}
```

## **Time-Domain Reflectometer Analysis (tdr)**

### **Description**

The time-domain reflectometer analysis linearizes the circuit about the DC operating point and computes the reflection coefficients versus time, looking from the active ports into the circuit.

### **Syntax**

Name `tdr parameter=value ...`

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Parameters

stop		Stop time.
settling	stop	Time required for circuit to settle.
start	- 0.1*stop	Time output waveforms begin.
smoothing	2	Window smoothing parameter (useful range is 0 to 15).
vel	1	Propagation velocity of medium normalized to c.
points	64	Number of time points.
ports	[...]	List of active ports. If not given, all ports are used.
readns		File that contains estimate of DC solution (nodeset).
useprevic	no	If set to yes or ns, use the converged initial condition from previous analysis as ic or ns. Possible values are no, yes and ns.
restart	yes	Restart the DC solution from scratch if any condition has changed. If not, use the previous solution as initial guess. Possible values are no and yes.
annotate	sweep	Degree of annotation. Possible values are no, title, sweep, status and steps.
title		Analysis title.
oppoint	no	Should operating point information be computed; if yes, where should it be sent. Operating point information would not be output if the operating point computed in the previous analysis remains unchanged. Possible values are no, screen, logfile and rawfile.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>prevoppoint</code>	<code>yes</code>	Use the operating point computed on the previous analysis. Possible values are no and yes.
--------------------------	------------------	--

Such a small-signal analysis begins by linearizing the circuit about an operating point. By default, this analysis computes the operating point, if it is not yet known, or recomputes it, if any significant component or circuit parameter has changed. However, if a previous analysis computed an operating point, you can set `prevoppoint=yes` to avoid recomputing it. For example, if you use this command when the previous analysis was a transient analysis, the operating point is the state of the circuit on the final time point.

## **THERMAL Analysis (thermal)**

### **Description**

### **Syntax**

Name thermal parameter=*value* ...

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Parameters

title		Analysis title.
annotate	sweep	Specifies the degree of annotation. Possible values are no, title, sweep and status.
config		Specifies the name of configured file of electro-thermal analysis.
sort	trise	Specifies the sorting criteria for the thermal analysis output file. Possible values are trise and pwr.
maxinst	all	Specifies the number of sorted device.
trise_lmt	300 C	Specifies the maximum Trise.
thermal_step_ratio	1	If ratio is X, it means that do thermal solver once every X tran steps.
method	steadystate	Activate either steady state (steadystate) or dynamic (dynamic) thermal analysis. Possible values are steadystate and dynamic.
numiter	2	Number of electro-thermal iterations to be performed.
res_trise_rpt	no	Update resistor temperature. Valid values are no, yes, and laststep. Only instance temperatures are updated when this option is set to no. Possible values are no and yes.
thermal_step	0.0	Thermal time step (seconds) for dynamic thermal solver. Set this parameter to zero to get identical electrical and thermal time steps.
save_inst	[...]	List of device names for which dynamic temperature and power will be reported.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

probe_output_for mat	binary	Specify the format of the output by the probe. Valid values are binary, txt, and both. Possible values are binary, txt, ascii and both.
chip_bbox	[...]	Delimit the boudry of the chip by specifying the value of coordinate. The order of the input value of is left bottom and then top right.
thermal_trise_step	0.0	Trise threshold for using thermal solver.
saveonlylaststep data	no	Dump the waveform files in a separate directory for steady-state. Possible values are no and yes.
thermal_windows	[...]	Boundary time to execute the thermal analysis. Array should have an even number of values. Note that only one time window is supported now.
auto_adaptive_mesh	[...]	Select the hottest region on channel layer and rerun with adaptive mesh automatically .
thermal_db_type	binary	Select thermal data output format. Possible values are binary and txt.
thermal_data_for mat	binary	Select trise data output format. Possible values are none, binary and txt.
frame_freq	3	Specify the time step interval for thermal trise data output.
thermal_ignore_parasitic_rc	no	Select whether ignore parasitic RC for thermal analysis. Possible values are no, yes and pwrOnly.
save_channelonly	no	Select the output layer for thermal trise data output. Possible values are no and yes.
check_inst_overlap	no	Check overlapped instances. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

keep_ori_cap	no	Keep original cap value from tdb file or not. Possible values are no and yes.
keep_pkg_grid	no	Keep package grid from tdb file or not. Possible values are no and yes.
save_layer	[...]	List of layer names for which trise will be reported in thermal temp file.
trise_lowerlmt	0.0 C	Specifies the minimum Trise.
thermal_flushpoints	-1	Specify the flush points number for thermal data output.
thermal_flush_time	-1	Specify the flush interval for thermal data output.
save_timepoints	[...]	Save temperature file at specific time points..
save_timewindows	[...]	Save temperature file during specific time intervals..

## Spectre Circuit Simulator Reference

### Analysis Statements

---

## Transient Analysis (tran)

### Description

This analysis computes the transient response of a circuit over the interval from `start` to `stop`. The initial condition is taken to be the DC steady-state solution, if not otherwise given.

### Syntax

```
Name tran parameter=value ...
```

### Parameters

Simulation interval parameters

<code>stop</code>	<code>(s)</code>	Stop time.
<code>tpoints</code>	<code>[...] s</code>	Multiple of pairs<pstep, stop>.
<code>start</code>	<code>0 s</code>	Start time.
<code>pstep</code>	<code>(s)</code>	print step.
<code>outputstart</code>	<code>start s</code>	Output is saved only after this time is reached.
<code>autostop</code>	<code>no</code>	If yes, the analysis is terminated when all event-type measurement expressions have been evaluated. Event-type expressions use thresholding, event, or delay type functions. If the value is <code>spice</code> , <code>autostop</code> is consistent with <code>spice</code> simulator. Possible values are <code>no</code> , <code>yes</code> and <code>spice</code> .

Time-step parameters

<code>maxstep</code>	<code>(s)</code>	Maximum time step. The default is derived from <code>errpreset</code> .
----------------------	------------------	---

## Spectre Circuit Simulator Reference

### Analysis Statements

---

step	0.001* (s top- start) s	Minimum time step used by the simulator solely to maintain the aesthetics of the computed waveforms.
minstep	(s)	Minimum time step. If specified, the error tolerance requirements may be ignored when step size is less than minstep.
istep	0.001* (s top- start) s	When step size is greater than istep, the local truncation error checking is enabled for algebraic nodes.

#### Initial-condition parameters

ic	all	The value to be used to set the initial condition. Possible values are dc, node, dev and all.
skipdc	no	If set to yes, there is no DC analysis for transient. Possible values are no, yes, useprevic, waveless, rampup, autodc and sigrampup.
rampupratio	0.1	The rampup ratio for skipdc=rampup and sigrampup.
rampuptime	(s)	The rampup time for skipdc=rampup. The default value is set to rampupratio*stop.
readic		File that contains initial condition.
useprevic	no	If set to yes or ns, use the converged initial condition from previous analysis as ic or ns. Possible values are no, yes and ns.
linearic	no	Enable linear IC method to calculate initial conditions automatically from a type of stability analysis in the range $[0.5 \cdot \text{oscfreq}, 1.5 \cdot \text{oscfreq}]$ . Overrides user-defined initial conditions if instability is detected. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>oscfreq</code>	<code>0.0</code>	Estimation of the oscillation frequency when linear IC method is enabled.
----------------------	------------------	---

#### Convergence parameters

<code>readns</code>		File that contains an estimate of the initial transient solution.
---------------------	--	---

<code>cmin</code>	<code>0 F</code>	Minimum capacitance from each node to ground.
-------------------	------------------	---

#### State-file parameters

<code>write</code>		File to which initial transient solution is to be written.
--------------------	--	--

<code>writefinal</code>		File to which final transient solution is to be written.
-------------------------	--	--

<code>ckptperiod</code>		Checkpoint the analysis periodically by using the specified period.
-------------------------	--	---

<code>saveperiod</code>		Save the tran analysis periodically on the simulation time.
-------------------------	--	---

<code>saveperiodhistory</code>	<code>no</code>	Maintains the history of saved files. If yes, maintains all the saved files. Possible values are no and yes.
--------------------------------	-----------------	--

<code>saveclock</code>	<code>(s)</code>	Save the tran analysis periodically on the wall clock time. The default is 1800s for Spectre. This parameter is disabled in the APS mode by default.
------------------------	------------------	--

<code>savetime</code>	<code>[...]</code>	Save the analysis states into files on the specified time points.
-----------------------	--------------------	---

<code>savefile</code>		Save the analysis states into the specified file.
-----------------------	--	---

<code>recover</code>		Specify the file to be restored.
----------------------	--	----------------------------------



## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Integration method parameters

<code>method</code>	Integration method. The default is derived from <code>errpreset</code> . Possible values are <code>euler</code> , <code>trap</code> , <code>traponly</code> , <code>gear2</code> , <code>gear2only</code> , <code>trapgear2</code> and <code>trapeuler</code> .
---------------------	---

#### Emir output parameters

<code>emirformat</code>	<code>none</code>	Format of the EM/IR database file. Possible values are <code>none</code> and <code>vavo</code> .
<code>emirstart</code>	<code>(s)</code>	EM/IR start time.
<code>emirstop</code>	<code>(s)</code>	EM/IR stop time.
<code>emirfile</code>		Name of the EM/IR database file. The default is <code>'%A_emir_vavo.db'</code> . The file is output to raw directory.
<code>emirtimewindows</code>	<code>[...]</code>	Time check windows of emir. Array should have an even number of values <code>[b_begin1 b_end1 b_begin2 b_end2 ...]</code> , the priority is higher than <code>'emirstart'</code> and <code>'emirstop'</code> .

#### Accuracy parameters

<code>errpreset</code>	Selects a reasonable collection of parameter settings. Possible values are <code>liberal</code> , <code>moderate</code> and <code>conservative</code> .
<code>relref</code>	Reference used for the relative convergence criteria. The default is derived from <code>errpreset</code> . Possible values are <code>pointlocal</code> , <code>alllocal</code> , <code>sigglobal</code> and <code>allglobal</code> .
<code>lteratio</code>	Ratio used to compute LTE tolerances from Newton tolerance. The default is derived from <code>errpreset</code> .

## Spectre Circuit Simulator Reference

### Analysis Statements

---

fastbreak	no	If yes, VHDLAMS Break statement is handled using faster Verilog method. Possible values are no and yes.
d2aminstep	0	Minimum stepsize that can be taken when there is a D2A event. If this is zero, the simulators min step size is chosen.
fastcross	cm	Specifies how VerilogA @cross, @above functions and AMSDesigner connect modules impact the time step control. no ... enforces time step at exact crossing point, discrete ... optimized time step control with best accuracy/performance trade-off, cm ... same as discrete for Spectre, but further performance optimization of connect modules in AMSD, yes ... better performance with cross function tolerances relaxed.. Possible values are no, yes, discrete and cm.
transres	1e-9*stop s	Transition resolution. The transient analysis attempts to stop at corners of input waveforms (for example, corners of rising/falling edge of a pulse). If such events occur within a time less than transres, the analysis combines the events into one and forces only one time point. The rest of the steps are determined by error control. This may lead to loss of detail.
lteminstep	0.0 s	Local truncation error is ignored if the step size is less than lteminstep.
ltethstep	1e-12 s	LTE tolerance can be relaxed for signal with discontinuity when step size is less than ltethstep.
d2atimetol	0	Time tolerance for D2A events. The smaller the value, the closer the time when the analog solver actually executes the d2a event may be to the time when the event gets reported.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

vrefmax		Upper limit of the voltage reference for relative error calculation.
irefmax		Upper limit of the current reference for relative error calculation.
vrefbins	[1 3 5 10]	Specifies the voltage bins when bin_relref=yes is used. Default setting is vrefbins=[1 3 5 10].
irefbins	[1e-6 1e-5 1e-4 1e-3 1e-2 1e-1]	Specifies the current bins when bin_irelref=yes is used. Default setting is irefbins=[1e-6 1e-5 1e-4 1e-3 1e-2 1e-1].
resetref	no	If yes, reset reference voltages/currents. Possible values are no and yes.

#### Annotation parameters

annotate	sweep	Degree of annotation. Possible values are no, title, sweep, status, estimated, steps, iters, detailed, rejects, alliters, detailed_hb and internal_hb.
annotateic	no	Degree of annotation for initial condition. Possible values are no, title, sweep, status, estimated, steps, iters, detailed, rejects and alliters.
annotatedigits	4	number of significant digits for annotate tran time. Possible values are 0, 1, 2 ...16. If set to 0, the number of significant digits is dynamic according to current time and step. If set to n (n = 1, 2 ...16), the number is equal to n..
title		Analysis title.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Output parameters

save		Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, none and nooutput.
nestlvl		Levels of subcircuits to output.
oppoint	no	Should operating point information be computed for initial timestep; if yes, where should it be printed (screen or file). Possible values are no, screen, logfile and rawfile.
skipstart	0 s	The time to start skipping output data.
skipstop	stop s	The time to stop skipping output data.
skipcount	1	Save only one of every skipcount points.
strobeperiod	0 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.
strobeoutput	strobeonly	Specifies which time points to output during strobe. Possible values are strobeonly, all and none.
strobestep	0 s	Equivalent to strobeperiod.
strobefreq		The reciprocal of strobeperiod (strobestep).
strobestart	0 s	Equivalent to skipstart.
strobestop	stop s	Equivalent to skipstop.
strobetimes	[...] s	Times in ascending order when strobe output performed.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

progress_t		Print out the annotate message every interval specified by progress_t in terms of minutes. Note that this degrades performance.
progress_p		Print out the annotate message every progress_p percent of transient time. Note that this degrades performance.
compression	no	Perform global waveform compression. Possible values are no, all, wildcardonly and yes.
comppreset	moderate	Defines waveform compression accuracy. Possible values are moderate, conservative and liberal.
compref	pointlocal	Reference used for relative error criteria in waveform compression. Possible values are alllocal, pointlocal, sigglobal and abstol.
compvabstol	1.0e-3 V	Absolute voltage tolerance for waveform compression.
compiabstol	1.0e-12 A	Absolute current tolerance for waveform compression.
comppwrabstol	1.0e-15 W	Absolute power tolerance for waveform compression.
compreltol	0.001	Relative tolerance for waveform compression.
complvl		Enables waveform compression for specified hierarchy level and below (top level=1). All levels above specified level are not compressed. Complvl has higher priority than global compression statement.
flushpoints		Periodically flush all unwritten data from the buffer to the outputs after calculating the specified number of points.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>flushtime</code>	(s)	Periodically flush unwritten data from the buffer to the outputs after the specified time has elapsed.
<code>flushofftime</code>	(s)	Real time to stop flushing outputs.
<code>flushpercent</code>		Flush unwritten data in the buffer to outputs for every X% of the transient progress.
<code>infonames</code>	[...]	Names of info analyses to be performed at the time point in the <code>infotimes</code> array.
<code>infotimes</code>	[...] s	Times when the analysis specified by <code>infonames</code> is performed.
<code>infotime_pair</code>	yes	If set to yes, there is a 1:1 correspondence between the values in <code>infonames</code> and in <code>infotimes</code> . For example, <code>infonames[0]</code> will be run at <code>infotimes[0]</code> , <code>infonames[1]</code> will be run at <code>infotimes[1]</code> , etc. If set to no, all analysis in <code>infonames</code> will be run at each <code>infotimes</code> time. Possible values are no and yes.
<code>acnames</code>	[...]	Names of ac, noise, sp, stb, xf, or lf analyses to be performed at each time point in the <code>actimes</code> array. The named small-signal analyses are not run separately, but only as part of the transient analysis.
<code>actimes</code>	[...] s	Times when analyses specified in <code>acname</code> array are performed.
<code>actime_pair</code>	yes	If set to yes, there is a 1:1 correspondence between the values in <code>acnames</code> and in <code>actimes</code> . For example, <code>acnames[0]</code> will be run at <code>actimes[0]</code> , <code>acnames[1]</code> will be run at <code>actimes[1]</code> , etc. If set to no, all analysis in <code>acnames</code> will be run at each <code>actimes</code> time. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Newton parameters

maxiters	5	Maximum number of iterations per time step.
dcmaxiters	150	Maximum number of dc iterations in tranFindInitialState.
restart	yes	Restart the DC solution from scratch if any condition has changed. If not, use the previous solution as initial guess. Possible values are no and yes.

#### Circuit age

circuitage	(Years)	Stress time. Age of the circuit used to simulate hot-electron degradation of MOSFET and BSIM circuits.
------------	---------	--

#### Transient noise parameters

trannoisemethod	default	use this option to enable the adaptive noise step control. Possible values are default and adaptive.
noiseemax	0 Hz	The bandwidth of pseudorandom noise sources. A valid (nonzero) noiseemax turns on the noise sources during transient analysis. The maximum time step of the transient analysis is limited to 0.5/noiseemax.
noisescale	1	Noise scale factor applied to all generated noise. Can be used to artificially inflate the small noise to make it visible above transient analysis numerical noise floor, but it should be small enough to maintain the nonlinear operation of the circuit .
noiseseed		Seed for the random number generator. Should be positive integer. Specifying the same seed allows you to reproduce a previous experiment.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>mc_auto_noiseseed</code>	<code>no</code>	Regenerate the seed at every monte carlo iteration using same mechanism. Possible values are yes, no. Default is no.. Possible values are no and yes.
<code>noisefmin</code>	<code>1 Hz</code>	If specified, the power spectral density of the noise sources depends on the frequency in the interval from noisefmin to noisefmax. Below noisefmin, the noise power density is constant. If noisefmin=noisefmax, then only white noise would be included, and noise sources are evaluated only at noisefmax for all models. 1/noisefmin should be smaller than the requested time duration of transient analysis.
<code>noiseon</code>	<code>[...]</code>	The list of instances to be considered as noisy during transient noise analysis.
<code>noiseoff</code>	<code>[...]</code>	The list of instances to be considered as not noisy during transient noise analysis.

#### Dynamic parameters

<code>param</code>		Name of the parameter to be updated to a different value with time during tran. You can use param=isnoisy with param_vec=[...] to turn On or Off the transient noise in time windows. For example, param=isnoisy param_vec=[0ns 0 100ns 1 500ns 0] . The transient noise is OFF (param value is 0) from time 0 to 100ns and the noise is ON (param value is 1) from 100ns to 500ns and OFF from 500ns to stop time.
<code>paramset</code>		Name of the dynamic parameter set.
<code>param_vec</code>	<code>[...] s</code>	The time_value points to param=name.
<code>param_file</code>		The file that contains the time_value points to param=name.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

sub	Name of the subcircuit instance parameter specified in param=name.
mod	Name of the device model parameter specified in param=name.
dev	Name of the device instance parameter specified in param=name.
param_step	Defines the frequency of updating the dynamic parameter values. If param_step=0, it updates the parameter value at a given time point.
paramsetfile	The file that contains the paramset data used for dynamic parameter change using event-triggered analysis.
steppreset	Used to dynamically change time-step control related settings in Spectre X. Example: tr1 tran stop=10u param=steppreset param_vec=[0 vx 3u mx]. Possible values are cx, ax, mx, lx and vx.
speed	Used to dynamically change time-step control related settings in Spectre FX. Example: tr1 tran stop=10u param=speed param_vec=[0 vx 3u mx]. Possible values are cx, ax, mx, lx, vx and sx.

#### Fault analysis parameters

faulttimes	[...] s	Fault times in ascending time order where transient fault analysis is performed. 'faulttimes' is required parameter for TFA, and optional for DFA. 'faultstart/faultstep/faultstop', 'faultevent', or 'faultfile' can be used as an alternative.
faultstep	(s)	The time interval between the fault times.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>faultstart</code>	<code>0 s</code>	The start time, or the first fault time, of transient fault analysis.
<code>faultstop</code>	<code>stop s</code>	The stop time, or the last fault time, of transient fault analysis.
<code>faultfile</code>		Name of the file that contains the fault times in ascending time order for transient fault analysis.
<code>faultevent</code>	<code>[...] s</code>	Events to perform transient fault analysis.
<code>faultautostop</code>	<code>no</code>	Auto-stop fault analysis when the fault is detected based on assert violation. Possible values are no, func, check, both and all.
<code>faultdetratio</code>		Auto-stop fault simulation when the specified detection ratio achieved.
<code>faultconfidlevel</code>		Auto-stop fault simulation when the specified detection ratio achieved with given confidence level.
<code>faultskipdc</code>	<code>yes</code>	Enforce skipdc=yes during fault analysis when no solution found during initial DC analysis. Possible values are no and yes.
<code>faultstrobe</code>	<code>no</code>	Time point strobe will be enabled during fault analysis. Possible values are no and yes.
<code>faultmethod</code>	<code>linear</code>	Simulation method used for transient fault analysis. Possible values are linear, maxiters, onestep, leadtime, testpoint and timezero.
<code>faulttablefmt</code>	<code>txt</code>	Format of fault table file. Possible values are txt and bin.
<code>faults</code>	<code>[...]</code>	Names of the fault blocks from the fault list to perform transient fault analysis. Default value is faults=[*].

## Spectre Circuit Simulator Reference

### Analysis Statements

---

faultsid	[...]	Indexes of faults from the list to perform fault analysis. If specified, simulation performed for requested subset of fault list.
faultsname	[...]	Names of faults from the list to perform fault analysis. If specified, simulation performed for requested subset of fault list.
faultsinst	[...]	List of instances to perform fault analysis. If specified, simulation performed only for faults within given instances.
faultsamplenum		Number of samples in random sampling of fault list during simulation.
faultsamplratio	[R1 R2]	A set of two values R1 and R2 (in percentage) to simulate the samples between the given range. 'faultsamplratio=R' is equivalent to 'faultsamplratio=[0 R]'. 'faultsamplratio=R' is equivalent to 'faultsamplratio=[0 R]'.
faultseed	1	Optional starting seed for random number generator during fault sampling.
faultsamplmethod	random	Method to be used for fault sampling. Possible values are random, randomweighted, randomuniform and weightsorted.
faultsamplreplace	yes	Perform fault sampling with sample replacement during fault analysis. Possible values are no and yes.
faultsort	yes	Perform fault simulation according to the order in the fault list. If 'no', random order applied. Possible values are no and yes.
faultcollapse	no	Perform fault collapsing in the list before fault simulation. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>faultduplicate</code>	<code>yes</code>	Perform fault simulation for all faults from the list. If 'no', identical and equivalent faults will be excluded from simulation. Possible values are no and yes.
<code>faultsave</code>	<code>testpoints</code>	Set of data to be saved during transient fault analysis. Possible values are testpoints, all and none.
<code>faultddm</code>		File containing the command line options to run 'spectre_ddmrpt' when fault simulation finished.
<code>faultdb</code>		File name to save fault simulation data in the format specified by 'faultfmt'.
<code>faultfmt</code>		File format to save fault simulation data. Possible values are csv, xl and sql.
<code>faultleadtime</code>	<code>[...] s</code>	The lead time intervals where the fault will be injected before the fault time when <code>faultmethod=leadtime</code> .
<code>faultmaxiters</code>		Maximum number of iterations per time step for transient fault analysis. Default is 50 for fault method 'maxiters', 10 for 'onestep', and 5 for other methods.
<code>faultlimiting</code>	<code>log</code>	Limiting algorithm to aid Newton convergence during transient fault simulation. Possible values are delta, log, dev and none.
<code>faulttrampsteps</code>	<code>0</code>	Maximum number of steps for fault conductance stepping to improve convergence after fault injection with method 'maxiters'. Recommended value <code>faulttrampsteps=10</code> .
<code>faultlimsteps</code>	<code>yes</code>	Auto stop slow simulation for the faults taking above average number of time steps. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

faultmaxsteps		Maximum steps allowed to perform one fault simulation. Simulation auto-stops if not finished earlier than specified step count.
faultlimtime		Maximum elapsed time allowed to perform one fault simulation (in hours). Simulation auto-stops if not finished earlier than specified time.
faultsafecheck	no	Enable auto-stop during the nominal simulation when the assert with 'safecheck' specified has been violated. Possible values are no and yes.
faulttrampinterval	0	The coefficient (between 0 and 1) to define the size of time interval to ramp fault conductance after fault injection when 'faultmethod=leadtime' specified.
faultic	no	Perform dc analysis after fault injection as initial conditions for transient fault analysis. Possible values are no, bridge, open, custom, param, all and yes.
faultactivestart	0	The time to begin circuit activity tracing for fault list generation.
faultactivevabstol	0.1	The minimum voltage value to consider an element as active for fault list generation.
faultactiveiabstol	1e-6	The minimum current value to consider an element as active for fault list generation.

#### Custom errpreset parameters

maxstepratio	Maximum time step ratio for custom 'errpreset'.
reltolratio	Reltol ratio for custom 'errpreset'.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>thermal_node_win</code>	<code>[...]</code>	Time windows of thermal node analysis. Array should have an even number of values [ <code>b_begin1 b_end1 b_begin2 b_end2 ...</code> ], currently it only supports one time window.
<code>msgSuppress</code>	<code>[...]</code>	Used to change the message severity threshold after the given time point, messages are suppressed if the run time is after the given time point and its severity is below the threshold. Arguments should have pairs of time and message severity, supported message severity are 'none', 'notice', 'warning', 'error'. 'none' means suppress nothing. Example: <code>tr1 tran stop 100u msgSuppress=[10u notice 20u warning 30u error 60u none ...]</code> The message severity threshold becomes invalid after tran analysis is finished..

You can specify the initial condition for the transient analysis by using the `ic` statement or the `ic` parameter on the capacitors and inductors. If you do not specify the initial condition, the DC solution is used as the initial condition. The `ic` parameter on the transient analysis controls the interaction of various methods of setting the initial conditions. The effects of individual settings are as follows:

`ic=dc`: All initial conditions are ignored, and the DC solution is used.

`ic=node`: The `ic` statements are used, and the `ic` parameter on the capacitors and inductors is ignored.

`ic=dev`: The `ic` parameters on the capacitors and inductors are used, and the `ic` statements are ignored.

`ic=all`: Both `ic` statements and `ic` parameters are used, and the `ic` parameters override the `ic` statements.

If you specify an initial condition file with the `readic` parameter, initial conditions from the file are used, and any `ic` statements are ignored.

After you specify the initial conditions, Spectre computes the actual initial state of the circuit by performing a DC analysis. During this analysis, Spectre forces the initial conditions on nodes by using a voltage source in series with a resistor whose resistance is `rforce` (see `options`).

## Spectre Circuit Simulator Reference

### Analysis Statements

---

With the `ic` statement, it is possible to specify an inconsistent initial condition (one that cannot be sustained by the reactive elements). Examples of inconsistent initial conditions include setting the voltage on a node with no path of capacitors to ground, or setting the current through a branch that is not an inductor. If you initialize Spectre inconsistently, its solution jumps, that is, it changes instantly at the beginning of the simulation interval. You should avoid such changes because Spectre can have convergence problems while trying to make the jump.

You can skip DC analysis entirely by using the parameter `skipdc`. If DC analysis is skipped, the initial solution is trivial or is given in the file that you specified by using the `readic` parameter, or if the `readic` parameter is not specified, by the values specified on the `ic` statements. Device-based initial conditions are not used for `skipdc`. Nodes that you do not specify with the `ic` file or `ic` statements start at zero. You should not use this parameter unless you are generating a nodeset file for circuits that have trouble in the DC solution; it usually takes longer to follow the initial transient spikes that occur when the DC analysis is skipped than it takes to find the real DC solution. The `skipdc` parameter might also cause convergence problems in the transient analysis.

The possible settings of parameter `skipdc` and their descriptions are as follows:

`skipdc=no`: Initial solution is calculated using normal DC analysis (default).

`skipdc=yes`: Initial solution is given in the file specified by the `readic` parameter or the values specified on the `ic` statements.

`skipdc=useprevic`: Initial solution obtained from the previous analysis is used.

`skipdc=waveless`: Same initial solution as `skipdc=yes`, but the waveform production in the time-varying independent sources is disabled during the transient analysis. Independent source values are fixed to their initial values (not their DC values).

`skipdc=rampup`: Independent source values start at 0 and ramp up to their initial values from `start` to `rampuptime`. If `rampuptime` is not given, `rampuptime` will be set to `rampupratio*stop`. After that their values remain constant. Zero initial solution is used.

`skipdc=autodc`: Same as `skipdc=waveless` if a nonzero initial condition is specified. Otherwise, same as `skipdc=rampup`.

`skipdc=sigrampup`: Independent source values start at 0 and ramp up to their initial values in the first phase of the simulation. Unlike `skipdc=rampup`, the waveform production in the time-varying independent source is enabled after the rampup phase. The rampup simulation is from the 'start' parameter. If the `start` parameter is not specified, the default `start` time is set to `-rampupratio*stop`.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

Nodesets help find the DC or the initial transient solution. You can specify nodesets in the circuit description file with nodeset statements or in a separate file by using the `readns` parameter. When nodesets are specified, Spectre computes an initial guess of the solution by performing DC analysis, while forcing the specified values on to nodes by using a voltage source in series with a resistor whose resistance is `rforce`. Spectre then removes these voltage sources and resistors and computes the required solution from this initial guess.

Nodesets have two important uses. First, if a circuit has two or more solutions, nodesets can bias the simulator towards computing the required solution. Second, this is a convergence aid. By estimating the solution of the largest possible number of nodes, you might be able to eliminate a convergence problem or significantly speed up convergence.

When you simulate the same circuit multiple times, it is recommended that you use both `write` and `readns` parameters and assign the same file name to both parameters. DC analysis then converges quickly even if the circuit has changed since the last simulation, and the nodeset file is automatically updated.

Nodesets and initial conditions have similar implementation, but produce different effects. Initial conditions define the solution, whereas nodesets only influence it. When you simulate a circuit with a transient analysis, Spectre forms and solves a set of differential equations. Because differential equations have an infinite number of solutions, a complete set of initial conditions must be specified to identify the required solution. Any initial conditions that you do not specify are computed by the simulator to be consistent. The transient waveforms then start from initial conditions. Nodesets are usually used as a convergence aid and do not affect the final results. However, in a circuit with more than one solution, such as a latch, nodesets bias the simulator towards finding the solution closest to the nodeset values.

The `method` parameter specifies the integration method. The possible settings and their meanings are as follows:

`method=euler`: Backward-Euler is used exclusively.

`method=traponly`: Trapezoidal rule is used almost exclusively.

`method=trapeuler`: Backward-Euler and the trapezoidal rule are used.

`method=gear2only`: Gear's second-order backward-difference method is used almost exclusively.

`method=gear2`: Backward-Euler and second-order Gear are used.

`method=trapgear2`: Allows all three integration methods to be used.

`method=trap`: An advanced version of trap that uses all three integration methods.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

The trapezoidal rule is usually the most efficient when you want high accuracy. This method can exhibit point-to-point ringing, but you can control this by tightening the error tolerances. For this reason, though, if you choose very loose tolerances to get a quick answer, the backward-Euler or second-order Gear will probably give better results than the trapezoidal rule. Second-order Gear and backward-Euler can make systems appear more stable than they really are. This effect is less pronounced with second-order Gear or when you request high accuracy.

Several parameters determine the accuracy of the transient analysis. `reltol` and `abstol` control the accuracy of the discretized equation solution. These parameters determine how well charge is conserved and how accurately steady-state or equilibrium points are computed. You can set the integration errors in the computation of the circuit dynamics (such as time constants), relative to `reltol` and `abstol` by setting the `lteratio` parameter.

The parameter `relref` determines how the relative error is treated. The `relref` options are as follows:

`relref=pointlocal`: Compares the relative errors in quantities at each node to that node alone.

`relref=alllocal`: Compares the relative errors at each node to the largest values found for that node alone for all past time.

`relref=sigglobal`: Compares relative errors in each circuit signal to the maximum for all signals at any previous point in time.

`relref=allglobal`: Same as `relref=sigglobal`, except that it also compares the residues (KCL error) for each node to the maximum of each node's past history.

The `errpreset` parameter lets you adjust the simulator parameters to fit your needs quickly. You can set `errpreset` to `conservative` if the circuit is very sensitive, or you can set it to `liberal` for a fast, but possibly inaccurate, simulation. The setting `errpreset=moderate` suits most needs.

The effect of `errpreset` on other parameters is shown in the following table. In this table,  $T = \text{stop} - \text{start}$ .

<code>errpreset</code>	<code>reltol</code>	<code>relref</code>	<code>method</code>	<code>maxstep</code>	<code>lteratio</code>
-----					
<code>liberal</code>	<code>* 10</code>	<code>sigglobal</code>	<code>trapgear2</code>	<code>Interval/50</code>	<code>3.5</code>
<code>moderate</code>		<code>sigglobal</code>	<code>traponly</code>	<code>Interval/50</code>	<code>3.5</code>

## Spectre Circuit Simulator Reference

### Analysis Statements

---

conservative \* 0.1 alllocal gear2only Interval/100 10.0

The default value for `errpreset` is moderate.

The value of `reltol` is increased or decreased from its value in the options statement, but it is not allowed to be larger than 0.01. Spectre sets the value of `maxstep` so that it is no larger than the value given in the table. Except for `reltol` and `maxstep`, `errpreset` does not change the value of any parameters you have explicitly set. The actual values used for the transient analysis are given in the log file.

'errpreset' also controls the LTE Check:

Liberal	Moderate	Conservative
-----	-----	-----

LTE Check   Caps/Inds   Loose nodes   strict nodes

It controls how the simulator follows signals other than capacitor voltages and inductor currents. When `errpreset=liberal`, the timestep is not controlled to follow these signals. When `errpreset=moderate`, the timestep is reduced to follow large changes in these signals. When `errpreset=conservative`, the timestep is reduced to follow small changes in these signals.

If the circuit you are simulating has infinitely fast transitions (for example, a circuit that contains nodes with no capacitance), Spectre might have convergence problems. To avoid this, you must prevent the circuit from responding instantaneously. You can accomplish this by setting `cmin`, the minimum capacitance to ground at each node, to a physically reasonable nonzero value. This often significantly improves Spectre convergence.

Spectre provides two methods for reducing the number of output data points saved: `strobing`, based on the simulation time, and `skipping` time points, which saves only every N'th point.

The parameters `strobeperiod` and `strobedelay` control the strobing method. `strobeperiod` sets the interval between the points that you want to save, and `strobedelay` sets the offset within the period relative to `skipstart`. The simulator forces a time step on each point to be saved, so that the data is computed, and not interpolated.

The skipping method is controlled by `skipcount`. If this is set to N, only every N'th point is saved.

The parameters `skipstart` and `skipstop` apply to both data reduction methods. Before `skipstart` and after `skipstop`, Spectre saves all computed data.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

If you do not want any data saved before a given time, use `outputstart`. If you do not want any data saved after a given time, change the `stop` time.

#### Dynamic Parameters during Transient Analysis

The parameters defined in the `Dynamic parameters` section allows you to change temperature, design parameters or some option parameters (`reltol`, `residualtol`, `vabstol`, `iabstol`, and `isnoisy`) during transient simulation.

Example1: change temperature during tran with `param_step=0`(default).

```
tran1 tran stop=0.5u param=temp param_vec=[0ns 20 50ns 25]
```

In this tran run, the temperature is 20C from 0ns-50ns, then it changes to 25C at 50ns. After tran is done, the temperature is reset back to its default value.

You can also define time value pairs in a file and give the file name through parameter `param_file`.

The format of the file is defined as follows:

; comments

tscale tscale\_value

time value

20 50.0

30 60.0

where, the comment line starts with a semicolon (;), `tscale` is a keyword, and `tscale_value` is a value, such as 1.0e-6 or 1.0e-9 that is applied to each time point under the time column. `time` and `value` are two key words that identify the time and value columns. The values under the time column define the time points and each time point is scaled by `tscale_value`. The values under the value column define the values for the dynamic parameter.

Note that no unit is supported in the file format.

Example2: change temperature during tran with `param_step=10ns`

```
tran1 tran stop=0.5u param=temp param_vec=[0ns 20 50ns 25] param_step=10ns
```

In this tran run, the temperature is interpolated with slope  $(25-20)/(50\text{ns}-0\text{ns})$  and updated every `param_step` (10ns).

## Spectre Circuit Simulator Reference

### Analysis Statements

---

Example3: change design parameter.

```
tran1 tran stop=0.5u param=gain sub=x1 param_vec=[0 5 1u 20]
```

Example4: turn On and Off transient noise in time windows.

```
tran1 tran stop=0.5u noisefmax=10G noiseseed=1
```

```
param=isnoisy param_vec=[0ns 0 100ns 1 500ns 0 ]
```

The transient noise is OFF from time 0 to 100ns. Noise is ON from 100ns to 500ns and noise is OFF from 500ns to stop time.

The default value for `compression` is `no`. The output file stores data for every signal at every time point for which Spectre calculates a solution. Spectre saves the X-axis data only once, because every signal has the same x value. If `compression=all`, Spectre writes data to the output file only if the signal value changes by at least two times the convergence criteria. 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 the simulation time, setting `compression=all` results in a smaller output data file. If the signals in your circuit move around a lot, setting `compression=all` results in a larger output data file. The `compvabstol`, `compiabstol` and `comppwrabstol` can be used to control output compression abstol for voltage, current and power respectively. The possible value (`alllocal`, `pointlocal`, `sigglobal` and `abstol`) of compression in transient statement will be deprecated in the next release.

## Special Current Saving Options (uti)

### Description

This command is used to report the dynamic current of all devices connected to the specified voltage source during dynamic simulation.

### Syntax

```
Name uti parameter=value ...
```

### Parameters

signal		specify the name of signal for which voltage drop must be calculated.
start		specify the start name of the measure.
clockcycle	(s)	specify the length of the clock cycle.
intervals		specify the number of measurement intervals within a clock cycle.
cycles		specify the number of clock cycles for which this measure will be calculated.
filename		specify the root name of the files containing the peak, average and RMS tap currents for this measure.
termflag		specify the terminals which will be output.
method		specify the method used in post-processing clock analysis data.
namemangling		specify namemangling.
utigzip	no	Zip uti output files. Possible values are no and yes.

**Spectre Circuit Simulator Reference**  
Analysis Statements

---

compression	no	If yes, enable UTI output compression. Default value is no. Possible values are no and yes.
-------------	----	---

## Transfer Function Analysis (xf)

### Description

The transfer function analysis linearizes the circuit about the DC operating point and performs a small-signal analysis that calculates the transfer function from every independent source in the circuit to a designated output. The variable of interest at the output can be voltage or current.

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, and you use the `portv` parameter to select the appropriate pair of terminals. Alternatively, you can specify a voltage to be the output by giving a pair of nodes on the `xf` analysis statement.

Any component that naturally computes current as an internal variable can be a current probe. If the probe component computes more than one current (as transmission lines, microstrip lines, and N-ports do), you use the `porti` parameter to select the appropriate current. It is an error to specify both `portv` and `porti`. If neither is specified, the probe component provides a reasonable default.

The `stimuli` parameter specifies the inputs for the transfer functions. There are two choices. `stimuli=sources` indicates that the sources present in the circuit should be used. The `xfmag` parameters provided by the sources may be used to adjust the computed gain to compensate for gains or losses in a test fixture. You can limit the number of sources in hierarchical netlists by using the `save` and `nestlvl` parameters.

The transfer functions computed versus output and source types are as follows:

Source	Output Type		Source
Type	voltage	current	Amplitude
-----+-----+-----+-----			
<code>vsource</code>	$V(out)/V(src)$	$I(out)/V(src)$	$V(src)=xfmag$
<code>isource</code>	$V(out)/I(src)$	$I(out)/I(src)$	$I(src)=xfmag$
<code>port</code>	$2 \cdot V(out)/V(src)$	$2 \cdot I(out)/V(src)$	$V(src)=2 \cdot xfmag$

where, `xfmag` defaults to 1 for each source type. For the `port`, `V(src)` is the internal source voltage.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

Specifying `stimuli=nodes_and_terminals` indicates that all possible transfer functions should be computed. This is useful when it is not known in advance which transfer functions are interesting. 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 potential (voltage) source is connected in series with the terminal. By default, the transfer functions from a small set of terminals are computed. If transfer functions from specific terminals are required, specify the terminals in the save statement. You must use the `:probe` modifier (for example, `Rout:1:probe`) or specify `useprobes=yes` on the options statement. If transfer functions from all terminals are required, specify `currents=all` and `useprobes=yes` on the options statement.

Spectre can perform AC analysis while sweeping a parameter. The parameter can be frequency, temperature, component instance parameter, component model parameter, or netlist parameter. If changing a parameter affects the DC operating point, the operating point is recomputed at each step. You can sweep the circuit temperature by giving the parameter name as `temp`, without a `dev` or `mod` parameter. In addition, you can sweep a netlist parameter by giving the parameter name without a `dev` or `mod` parameter. After the analysis is complete, the modified parameter returns to its original value.

### Syntax

```
Name [p] [n] xf parameter=value ...
```

The optional terminals (p and n) specify the output of the circuit. If you do not specify the terminals, you must specify the output with a probe component.

### Parameters

<code>prevoppoint</code>	<code>no</code>	Use the operating point computed in the previous analysis. Possible values are no and yes.
--------------------------	-----------------	--

#### Sweep interval parameters

<code>start</code>	<code>0</code>	Start sweep limit.
<code>stop</code>		Stop sweep limit.
<code>center</code>		Center of sweep.



## Spectre Circuit Simulator Reference

### Analysis Statements

---

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.
valuesfile		Name of the file containing the sweep values.

#### Sweep variable parameters

dev		Device instance whose parameter value is to be swept.
mod		Model whose parameter value is to be swept.
param		Name of parameter to sweep.
freq	(Hz)	Frequency when a parameter other than frequency is being swept.

#### Probe parameters

probe		Compute every transfer function to this probe component.
-------	--	--

#### State-file parameters

readns		File that contains an estimate of the DC solution (nodeset).
--------	--	--

## Spectre Circuit Simulator Reference

### Analysis Statements

---

<code>write</code>		DC operating point output file at the first step of the sweep.
<code>writefinal</code>		DC operating point output file at the last step of the sweep.

#### Initial condition parameters

<code>force</code>	<code>none</code>	The set of initial conditions to use. Possible values are none, node, dev and all.
<code>readforce</code>		File that contains initial conditions.
<code>skipdc</code>	<code>no</code>	Skip DC analysis. Possible values are no and yes.
<code>useprevic</code>	<code>no</code>	If set to yes or ns, use the converged initial condition from previous analysis as ic or ns. Possible values are no, yes and ns.

#### Output parameters

<code>stimuli</code>	<code>sources</code>	Stimuli used for xf analysis. Possible values are sources and nodes_and_terminals.
<code>save</code>		Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, none and nooutput.
<code>nestlvl</code>		Levels of subcircuits to output.
<code>oppoint</code>	<code>no</code>	Determines whether operating point information should be computed:if yes, where should it be printed (screen or file). Operating point information is not printed if the operating point computed in the previous analysis remains unchanged. Possible values are no, screen, logfile and rawfile.

## Spectre Circuit Simulator Reference

### Analysis Statements

---

#### Convergence parameters

<code>restart</code>	<code>yes</code>	Restart the DC solution from scratch if any condition has changed. If not, use the previous solution as an initial guess. Possible values are no and yes.
----------------------	------------------	---

#### Annotation parameters

<code>annotate</code>	<code>sweep</code>	Degree of annotation. Possible values are no, title, sweep, status and steps.
<code>title</code>		Analysis title.

You can define sweep limits by specifying the end points or the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. You can specify a step size parameter (`step`, `lin`, `log` or `dec`) to determine whether the sweep is linear or logarithmic. If you do not specify 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. All frequencies are in Hertz.

The small-signal analysis begins by linearizing the circuit about an operating point. By default, this analysis computes the operating point, if it is not known, or recomputes it if any significant component or circuit parameter has changed. However, if an operating point was computed during a previous analysis, you can set `prevoppoint=yes` to avoid recomputing it. For example, if `prevoppoint=yes` and the previous analysis was a transient analysis, the operating point is the state of the circuit at the final time point.

Nodesets help find the DC or the initial transient solution. You can specify nodesets in the circuit description file with `nodeset` statements or in a separate file by using the `readns` parameter. When nodesets are specified, Spectre computes an initial guess of the solution by performing DC analysis, while forcing the specified values on to nodes by using a voltage source in series with a resistor whose resistance is `rforce`. Spectre then removes these voltage sources and resistors and computes the required solution from this initial guess.

Nodesets have two important uses. First, if a circuit has two or more solutions, nodesets can bias the simulator towards computing the required solution. Second, this is a convergence aid. By estimating the solution of the largest possible number of nodes, you might be able to eliminate a convergence problem or significantly speed up convergence.

When you simulate the same circuit multiple times, it is recommended that you use both `write` and `readns` parameters and assign the same file name to both parameters. DC

## Spectre Circuit Simulator Reference

### Analysis Statements

---

analysis then converges quickly even if the circuit has changed since the last simulation, and the nodeset file is automatically updated.

During the initial operating point DC analysis, you may force certain circuit variables to use the values given in the `ic` file, `ic` statements, or `ic` parameter on the capacitors and inductors. The `ic` parameter controls the interaction of various methods of setting the force values. The effects of individual settings are as follows:

`force=none`: All initial conditions are ignored.

`force=node`: The `ic` statements are used, and the `ic` parameters on the capacitors and inductors are ignored.

`force=dev`: The `ic` parameters on the capacitors and inductors are used, and the `ic` statements are ignored.

`force=all`: Both `ic` statements and `ic` parameters are used, with the `ic` parameters overriding the `ic` statements.

If you specify an `ic` file with the `readforce` parameter, force values from the file are used and any `ic` statements are ignored.

After you specify the initial conditions, Spectre computes the DC operating point with the specified nodes forced to the given value by using a voltage source in series with a resistor whose resistance is `rforce` (see `options`).

---

## Other Simulation Topics

---

This chapter discusses the following topics:

- [AHDL Linter Usage \(ahdllint\)](#) on page 479
- [Using analogmodel for Model Passing \(analogmodel\)](#) on page 483
- [Behavioral Source Use Model \(bsource\)](#) on page 485
- [Checkpoint - Restart \(checkpoint\)](#) on page 494
- [Configuring CMI Shared Objects \(cmiconfig\)](#) on page 496
- [Built-in Mathematical and Physical Constants \(constants\)](#) on page 498
- [Convergence Difficulties \(convergence\)](#) on page 500
- [encryption \(encryption\)](#) on page 503
- [Expressions \(expressions\)](#) on page 506
- [The fastdc command line option \(fastdc\)](#) on page 511
- [Fault List for Transient Fault Analysis \(faults\)](#) on page 512
- [Dynamic Force Voltage Node \(force\\_voltage\)](#) on page 514
- [User Defined Functions \(functions\)](#) on page 516
- [Global Nodes \(global\)](#) on page 517
- [IBIS Component Use Model \(ibis\)](#) on page 518
- [Initial Conditions \(ic\)](#) on page 525
- [The Structural if-statement \(if\)](#) on page 526
- [Include File \(include\)](#) on page 528
- [Spectre Netlist Keywords \(keywords\)](#) on page 530
- [Library - Sectional Include \(library\)](#) on page 533

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

- [Tips for Reducing Memory Usage \(memory\)](#) on page 538
- [Node Sets \(nodeset\)](#) on page 540
- [Parameter Soft Limits \(param\\_limits\)](#) on page 541
- [Netlist Parameters \(parameters\)](#) on page 544
- [Parameter Set - Block of Data \(paramset\)](#) on page 547
- [The postlayout command line option \(postlayout\)](#) on page 548
- [Pspice include File \(pspice\\_include\)](#) on page 549
- [Tips for Reducing Memory Usage with SpectreRF \(rfmemory\)](#) on page 552
- [Output Selections \(save\)](#) on page 557
- [Savestate - Recover \(savestate\)](#) on page 560
- [Sensitivity Analyses \(sens\)](#) on page 564
- [SpectreRF Summary \(spectrerf\)](#) on page 567
- [Stitch Flow Use Model \(stitch\)](#) on page 568
- [Subcircuit Definitions \(subckt\)](#) on page 574
- [Vec/Vcd/Evcd Digital Stimulus \(vector\)](#) on page 579
- [Verilog-A Usage and Language Summary \(veriloga\)](#) on page 582

## AHDL Linter Usage (ahdllint)

### Description

The AHDL Linter technology offers a powerful set of capabilities for upfront identification of issues that worry designers even well beyond the model creation stage. The AHDL Linter feature helps identify complex design modeling issues and can be used on block simulation or SOC verifications performed just before tape out. Early warnings enable designers to avoid expensive design iterations, and meeting quality and time-to-market goals.

You can use the AHDL Linter feature to analyze Spectre APS, and Spectre XPS test cases containing Verilog-A models.

AHDL Linter checks each Verilog-A behavioral model in the design and suggests which line should be improved to:

- Avoid potential convergence or performance problems
- Improve model accuracy, reusability, and portability

AHDL linter consists of static and dynamic lint checks. Static lint checks are performed at compilation stage. Dynamic lint checks are performed during analysis for dynamic modeling issues that may cause convergence or performance problems during simulation.

### *Using the AHDL Linter Feature*

The AHDL Linter feature is activated in Spectre APS, and Spectre XPS using the following command-line options:

```
% spectre +aps -ahdllint netlist.scs
% spectre +xps +cktpreset=sram -ahdllint netlist.scs
```

where `netlist.scs` is written using Spectre, SPICE, or eSpice syntax and includes one or more Verilog-A models.

You can perform static linter checks on a netlist file that includes one or more Verilog-A models, as follows:

```
spectre +aps -ahdllint=static test.scs // test.scs includes the Verilog-A files
```

Spectre, in this case, performs only linter checks on all Verilog-A files and does not run the simulation.

## Spectre Circuit Simulator Reference

### Other Simulation Topics

You can also perform static lint checks on a Verilog-A file directly without specifying the top-level netlist file (.scs), as follows:

```
spectre +aps -ahdllint=static test.va //perform static lint check on test.va
```

The `-ahdllint` option accepts the following arguments:

no	Disables lint checks. There is no change in the existing compilation or simulation error messages.
warn	(default) Turns on the static lint and dynamic lint checks. Except the models with attribute ( <code>* ahdllint = "no" *</code> ), the static lint checks all models, continues the simulation, and then performs dynamic lint checks.
error	Turns on the static lint check. Dynamic lint checks are performed only when static lint issues are not detected. Except the models with attribute ( <code>* ahdllint = "no" *</code> ), the static lint checks all models. The simulator generates an error and exits if there is any static lint warning reported after compiling all the models of the circuit. If there are no static lint warnings, the simulator continues the simulation and performs dynamic lint checks. However, in the case of dynamic lint issues, the simulator does not error out.
force	Similar to <code>warn</code> , but this option overrides the model attribute ( <code>* ahdllint = "no" *</code> ), and forces to check all models, continue the simulation, and perform dynamic lint checks.
static	Performs static lint checks on a Verilog-A file directly.

**Note:** The `-ahdllint` option can be used without any argument, which is equivalent to specifying `-ahdllint=warn`. You can also set the `-ahdllint` option as default by setting the `SPECTRE_DEFAULTS` environment variable, as follows:

```
setenv SPECTRE_DEFAULTS "-ahdllint=warn"
```

You can use the `-ahdllint_maxwarn =n` command-line option to control the maximum number of static and dynamic warnings. The default value is 5.

You can also control the maximum number of warnings by using the environment variable `SPECTRE_DEFAULTS` or `ULSTRASIM_DEFAULTS`, as show below.

```
setenv SPECTRE_DEFAULTS "-ahdllint -ahllint_maxwarn=10"
setenv ULSTRASIM_DEFAULTS "-ahdllint -ahllint_maxwarn=10"
```

You can use the `-ahdllint_warn_id=<ID>` option to control the warning messages for a specific message ID. `ID` should be a positive integer value between 5001 and 6000 or 8001



and 9999. In addition, the `-ahdllint_warn_id` option should always be used with the `-ahdllint_maxwarn` option and should be directly specified after it in the command line, otherwise, the option will be ignored. For example:

```
spectre +aps -ahdllint -ahdllint_maxwarn=2 -ahdllint_warn_id=5001 input.scs
```

In the above example, Spectre will display only two warning messages related to the message ID 5001.

You can specify the `-ahdllint_maxwarn` and `-ahdllint_warn_id` options multiple times at the command line. For example:

```
spectre +aps -ahdllint -ahdllint_maxwarn=1 -ahdllint_warn_id=5001 -ahdllint_maxwarn=3 -ahdllint_warn_id=8001 input.scs
```

In the above example, only one warning message related to the message ID 5001 will be displayed. In addition, three warning messages related to the message ID 8001 will be displayed.

You can also specify multiple IDs with the `-ahdllint_warn_id` option. However, the IDs must be specified using double quotes with a space between them. For example:

```
spectre +aps -ahdllint -ahdllint_maxwarn=2 -ahdllint_warn_id="5001 5003 5005" input.scs
```

Use the `-ahdllint_minstep=value` command-line option to set the minimal time step size boundary that will trigger an AHDL Linter warning, when the step size imposed by a Verilog-AMS filter or function, such as `transition`, `cross`, `above`, `$bound_step`, and `timer` is smaller than `value`. The default value is `1e-12`.

You can use the `-ahdllint_summary_maxentries=<value>` command-line option to control the maximum number of messages to be displayed in dynamic lint summary. The default value is 25. For example:

```
spectre -ahdllint_summary_maxentries=20 input.scs
```

By default, the static and dynamic lint warnings and the dynamic lint summary output are sent to the simulation log file. You can use the `-ahdllint_log=file` command-line option to output all the information to a specified file instead of the output log file.

### **Filtering AHDL Messages**

You can filter the AHDL Linter messages using the `options` control statement, as follows:

```
Name options warning_limit=num warning_id=id
```

where:

`warning_limit` specifies the number of messages to be printed.

`warning_id` specifies the message id that needs to be printed.

#### Example1

```
myoptions options warning_limit=0 warning_id=[AHDLLINT-8004 AHDLLINT-8005]
```

In the above example, Spectre APS/XPS will not print any message with the id AHDLLINT-8004 and AHDLLINT-8005.

#### Example2

```
myoptions2 options warning_limit=1 warning_id=[AHDLLINT-8005]
```

In the above example, Spectre APS/XPS will print only one message with the id AHDLLINT-8005.

For more information about the `options` control statement, see `spectre -h options`.

### ***Using the ahdhelp Utility***

You can use the `ahdhelp` utility to view the extended help on various messages reported by AHD Linter. To view the extended help, you need to provide the AHD Linter id number as an argument, as shown below.

```
% ahdhelp 8005
```

The following is an example of the `ahdhelp` utility:

```
% ahdhelp 5012
```

```
AHDLLINT-5012: Detected $abstime in an equality expression inside a conditional.
```

```
Analog simulations select timepoints using a variable timestep size algorithm based on accuracy considerations, so the value of "time" only changes between discrete real values. In order to perform an event at a particular time, the @(timer) construct must be used to tell the simulator to step to a particular timepoint and execute the desired code when that event actually occurs.
```

```
example:
```

```
    if ($abstime==50n) xval=2;
```

```
solution:
```

```
    @(timer(50n)) xval=2;
```

## Using analogmodel for Model Passing (analogmodel)

### Description

`analogmodel` is a reserved word in Spectre that allows you to bind an instance to different masters based on the value of a special instance parameter called `modelname`. An instance of `analogmodel` must have a parameter named `modelname`, whose string value represents the name of the master this instance will be bound to. The value of `modelname` can be passed into subcircuits.

The `analogmodel` keyword is used by the Cadence Analog Design Environment to enable model name passing through the schematic hierarchy.

### Sample Instance Statement

```
name [(]node1 ... nodeN[)] analogmodel modelname=mastername [[param1=value1]
...[paramN=valueN]]
```

---

name	Name of the statement or instance label.
[(]node1...nodeN[)]	Names of the nodes that connect to the component.
analogmodel	Special device name to indicate that this instance will have its master name specified by the value of the <code>modelname</code> parameter on the instance.
modelname	Parameter to specify the master of this instance indicated by <code>mastername</code> . The <code>mastername</code> must either be a valid string identifier or a netlist parameter that must resolve to a valid master name, a primitive, a model, a subckt, or an AHDL module.
param1 param2...	Parameter values for the component. Depending on the master type, these can either be device parameters or netlist parameters. It is optional to specify these parameter values.

---

### Example

```
//example spectre netlist to illustrate modelname parameter
simulator lang=spectre
parameters b="bottom"
include "VerilogAStuff.va"
topInst1 (out in) top
topInst2 (out in) analogmodel modelname="VAMaster" //VAMaster is defined in
                                                    "VerilogAStuff.va"
topInst3 (out in) analogmodel modelname="resistor" //topInst3 binds to a primitive
topInst4 (out 0) analogmodel modelname="myOwnRes" //topInst4 binds to modelcard
```

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

```

                                                                    "myOwnRes" defined below
v1 in 0 vsource dc=1
    model myOwnRes resistor r=100
    subckt top out in
        parameters a="mid"
        x1 (out in) analogmodel modelname=a //topInst1.x1 binds to "mid"
    ends top
subckt mid out in
    parameters c="low"
    x1 (out in) analogmodel modelname=b //topInst1.x1.x1 binds to "bottom"
    x2 (out in) analogmodel modelname=c //topInst1.x1.x1.x2 binds to "low"
ends mid
subckt low out in
    x1 (out in) analogmodel modelname="bottom" //topInst1.x1.x1.x2.x1 binds to
                                                "bottom"
ends low
subckt bottom out in
    x1 (out in) analogmodel modelname="resistor" //x1 binds to primitive "resistor"
ends bottom
dcl dc
// "VerilogAStuff.va"
`include "constants.h"
`include "discipline.h"
module VAMaster(n1, n2);
inout n1, n2;
electrical n1, n2;
parameter r=1k;
analog begin
I(n1, n2) <+ V(n1, n2)/r;
end
endmodule
```

## Behavioral Source Use Model (bsource)

### Description

Behavioral source enables you to model a resistor, inductor, capacitor, voltage or current source as a behavioral component. Using `bsource`, you can express the value of a resistance, capacitance, voltage or current as a combination of node voltages, branch currents, time expression, and built-in Spectre expressions.

The syntax for `bsource` is as follows:

```
name (node1 node2) bsource behav_param param_list
```

where `behav_param` can be:

---

<code>c=simple_expr,</code>	capacitance between the nodes
<code>g=simple_expr,</code>	conductance between the nodes
<code>i=generic_expr,</code>	current through <code>bsource</code>
<code>l=simple_expr,</code>	inductance between the nodes
<code>phi=simple_expr,</code>	flux in the <code>bsource</code> device
<code>q=simple_expr,</code>	charge in <code>bsource</code> device
<code>r=simple_expr,</code>	resistance between the nodes
<code>v=generic_expr,</code>	voltage across the nodes

---

`simple_expr` is defined as an Spectre expression, which contains:

- Netlist parameters.
- Current simulation time, `$time`.
- Current frequency, `$freq`.
- Node voltage, `v(a,b)`, where `a` and `b` are nodes in the Spectre netlist, or node voltage `v(a)`, which is the voltage between node `a` and ground.
- Branch currents, `i("inst_id:index")`, where `inst_id` is an instance name given in the netlist and `index` is the port index that starts from 1. The default value of `index` is 1.

**Note:** If the value of the port index is set to 0, `simple_expr` treats it as the default value 1.

`generic_expr` is defined as a `simple_expr` or `ddt()` or `idt()` of `simple_expr`.

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

`param_list` is `param_name=value`.

`param_name` can have the multiplicity factor `m`. The value of `m` defaults to 1.

### Temperature Parameters

---

<code>tc1</code>	Linear temperature co-efficient. Valid for all behavioral elements. Default value is 0 1/C.
<code>tc2</code>	Quadratic temperature co-efficient. Valid for all behavioral elements. Default value is 0 C <sup>2</sup> .
<code>temp</code>	Parameter for ambient temperature. Valid for all behavioral elements. Default value is \$temperature - 273.15.
<code>tnom</code>	Parameters measurement temperature. Valid for all behavioral elements. Default value is 27.0.
<code>trise</code>	Temperature rise for ambient. Valid for all behavioral elements. Default value is 0.0.
<code>T</code>	Effective value of temperature. Valid for all behavioral elements. Default value is <code>temp+trise-tnom</code> .
<code>tc1c</code>	Linear temperature coefficient of capacitor. Valid for resistor type behavioral element. Default value is 0 1/C.
<code>tc2c</code>	Quadratic temperature coefficient of capacitor. Valid for resistor type behavioral element. Default value is 0 C <sup>2</sup> .

---

### Clipping Parameters

---

<code>max_val</code>	Maximum value of <code>bsource</code> expression. Valid for all behavioral elements, but used with <code>i</code> and <code>v</code> elements to clip the current or voltage between the specified values.
<code>min_val</code>	Minimum value of <code>bsource</code> expression. Valid for all behavioral elements, but used with <code>i</code> and <code>v</code> elements to clip the current or voltage between the specified values.
<code>bv_max</code>	Generate a warning when <code>bsource</code> voltage of two terminals exceeds the value specified by <code>bv_max</code> . Valid for resistor and capacitor.

---

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

#### Noise Parameters

---

<code>af</code>	Flicker noise exponent. Valid for r and g elements. Default value is 2.
<code>fexp</code>	Flicker noise frequency exponent. Valid for r, g, v, and i elements. Default value is 1.
<code>ef</code>	The alias parameter of <code>fexp</code> .
<code>ldexp</code>	Flicker (1/f) noise L exponent. Valid for r and g elements. Default value is 1.0.
<code>lf</code>	The alias parameter of <code>ldexp</code> .
<code>wdexp</code>	Flicker (1/f) noise W exponent. Valid for r and g elements. Default value is 1.0.
<code>wf</code>	The alias parameter of <code>wdexp</code> .
<code>isnoisy</code>	Specifies whether to generate noise. Valid for r, g, i, and v elements. Valid values are <code>yes</code> and <code>no</code> . Default value is <code>yes</code> .
<code>kf</code>	Flicker noise co-efficient. Valid for r and g elements. Default value is 0.
<code>white_noise</code>	White noise expression. Valid for v and i elements.
<code>flicker_noise</code>	Flicker noise expression. Valid for v and i elements.

---

#### DC Mismatch Parameters

---

<code>mr</code>	DC-Mismatch parameter. Valid only for r.
-----------------	--

---

#### Other Parameters

---

<code>scale</code> , <code>scaleb</code> , <code>scaler</code> , <code>scalei</code> , <code>scalec</code>	Bsource scaling factor. Default value is 1.0.
--	---

---

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

c <sub>type</sub>	Different implementations of a capacitor. When the value is 1, bsource current is $\text{ddt}(\text{cap} * V(\text{node1}, \text{node2}))$ , where <i>cap</i> is the bsource capacitor value with temp effect, mfactor effect, scale effect and so on. $V(\text{node1}, \text{node2})$ is the voltage between the bsource terminals. When the value is 2, the current is $\text{ddt}(\text{cap})$ . When the value is 0 or any other value, the current value is $\text{cap} * \text{ddt}(V(\text{node1}, \text{node2}))$ . Default value is 0.
-------------------	---

---

For the detailed algorithm, refer to "*Affirma Spectre DC Device Matching Analysis Tutorial*."

All the parameters in the `param_name` table are instance parameters. `white_noise` and `flicker_noise` may be assigned behavioral expressions; the other parameters must be assigned constant or parametric expressions.

### Supported Instance Parameters

`bsource` supports the following instance parameters for Spectre primitives:

---

Resistor	<code>isnoisy, m, r, tc1, tc2, trise, kf, af, fexp, ldexp, wdexp, l, w, mr</code>
Capacitor	<code>c, m, tc1, tc2, trise, ic, c<sub>type</sub></code>
Inductor	<code>l, m, tc1, tc2, trise</code>

---

### Mathematical Definitions

- $i = \text{ddt}(q) = \text{ddt}(\text{simple\_expr})$
- $v = \text{ddt}(\phi) = \text{ddt}(\text{simple\_expr})$
- $v = i * r = \text{simple\_expr} * v$
- $i = g * v = \text{simple\_expr} * v$
- $i = c * \text{ddt}(v) = \text{simple\_expr} * \text{ddt}(v)$
- $v = l * \text{ddt}(i) = \text{simple\_expr} * \text{ddt}(i)$



## Spectre Circuit Simulator Reference

### Other Simulation Topics

#### Operating Point Parameters

##### **Capacitor**

cap (F)	Capacitance at operating point
---------	--------------------------------

##### **Conductor**

g (S)	Conductance at operating point
v (V)	Voltage at operating point
i (A)	Current through the conductor
pwr (W)	Power dissipation.

##### **Current Source**

v (V)	Voltage across the source
i (A)	Current through the source
pwr (W)	Power dissipation

##### **Inductor**

ind (H)	Inductance at operating point
i (A)	Current at operating point

##### **Charge**

cap (F)	Capacitance at operating point
ddt_v (V/s)	Voltage gradient at operating point

##### **Resistor**

v (V)	Voltage at operating point
i (A)	Current through the resistor

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

<code>res</code> (Ohm)	Resistance at operating point
<code>pwr</code> (W)	Power dissipation

---

### ***Voltage Source***

---

<code>v</code> (V)	Voltage across the source
<code>i</code> (A)	Current through the source
<code>pwr</code> (W)	Power dissipation

---

### **Temperature Effects on bsource**

The equation for computing temperature factor is as follows:

```
tempFactor = [1 + tc1*(temp+trise-tnom)+tc2*(temp+trise-tnom)^2]
```

### **Frequency Effects on bsource**

The support for frequency-dependent bsource is available. You can use a frequency-dependent resistor, capacitor, inductor, or bsource component in ac/noise/xf/sp/hb/hbac/hbnoise analyses.

The syntax is as follows:

```
name ( node1 node2 ) model_name behav_param=freq_expression
```

where:

- `name` is the name of the instance
- `model_name` can be bsource, resistor, capacitor, and inductor
- `behav_param` is the r, c or l parameter.
- `freq_expression` can be specified by arbitrary expressions using the `$freq` keyword.

Following are some examples using `$freq`:

```
res (n1 n2) bsource r= 100 + sqrt($freq)/1e6
res (n1 n2) resistor r= 100 + sqrt($freq)/1e6
cap (n1 n2) bsource c=1e-12/(1+($freq)/1e9)
cap (n1 n2) capacitor c=1e-12/(1+($freq)/1e9)
ind (n1 n2) bsource l=1e-9+1e-9/(1+($freq)/1e9)
ind (n1 n2) inductor l=1e-9+1e-9/(1+($freq)/1e9)
```

## Examples of bsource Usage

### Non-linear resistor/capacitor/inductor modeling

#### Frequency-dependent resistor component

```
res1 (n1 n2) bsource r=100*(1+(1/2)*v(n1,n2))
res2 (n1 n2) resistor r=100*(1+(1/2)*v(n1,n2))
```

In this example, the resistance of `res1` is frequency-dependent. Therefore, if you run `hb` and `hbnoise` analyses for different frequencies, the resistance varies accordingly.

#### Frequency-dependent capacitor component

```
C1 (1 0) capacitor c=1e-12/(1+($freq)/1e9)
V1 (1 0) vsource type=sine freq=1G fundname="freq"
ac ac start=1 stop=100M
hb hb tstab=0n funds=["freq"] maxharms=[5] errpreset=conservative
hbac hbac start=1 stop=1G dec=3
```

In this example, the capacitance of `C1` is frequency-dependent. Therefore, if you run `hb` and `hbnoise` analyses for different frequencies, the capacitance varies accordingly.

#### Frequency-dependent inductor component

```
L1 (1 0) inductor l=1e-9+1e-9/(1+($freq)/1e9)
V1 (1 0) vsource type=sine freq=1G fundname="freq"
ac ac start=1 stop=100M
hb hb tstab=0n funds=["freq"] maxharms=[5] errpreset=conservative
```

In this example, the inductance of `L1` is frequency-dependent. Therefore, if you run `hb` analysis for different frequencies, the inductance varies accordingly.

### Charge model for capacitor

```
cap (n1 n2) bsource q=1.0e-6*v(n1,n2)
```

### Voltage and current (Sinewave) source

```
vsrc (n1 n2) bsource v=10.0*sin(2*pi*freq*$time)
isrc (n1 n2) bsource i=1.0e-3*sin(2*pi*freq*$time)
```

### Current-controlled current source

```
vsrc (n1 n2) vsource v=10
cccs1 (n3 n4) bsource i=gain*i("vsrc:1")
```

### Current-controlled voltage source

```
vsrc (n1 n2) vsource v=10
```

```
ccvs1 (n3 n4) bsource v=100*i("vsrc:1")
```

### **Voltage-controlled voltage source**

```
vsrc (n1 n2) resistor r=100k  
vcvs1 (n3 n4) bsource v=gain*v(n1,n2)
```

### **Voltage-controlled current source**

```
vsrc (n1 n2) resistor r=100k  
vccs1 (n3 n4) bsource i=v(n1,n2)/2000.0
```

### **Restricting voltage limit**

```
res (n1 n2) bsource r=100*(1+(1/2)*v(n1,n2)) max_val=105 min_val=95
```

### **Using temperature coefficient for resistor**

```
res (n1 n2) bsource r=100 tc1=0.01 tc2=0.003 trise=10 tnom=30
```

### **Using DC mismatch parameter for resistor**

```
res (n1 n2) bsource r=100 mr=0.3
```

### **Using model card**

```
model model_card_res resistor tc1=0.1 tc2=0.1  
res (n1 n2) model_card_res r=100*(1+(1/2)*v(n1,n2))
```

### **Sample netlist with bsource**

```
vsrc1 (n1 n2) bsource v=10*sin(2*pi*freq1*$time)  
vsrc2 (n3 n4) bsource v=10*cos(2*pi*freq2*$time)  
cccs1 (n5 n6) bsource i=gain*i("vsrc1:1")  
res (n5 n6) bsource r=100*(1+(1/2)*v(n5,n6))  
  
tran1 tran stop=1u  
  
altAnal altergroup {  
    cccs1 (n5 n6) bsource i=gain*i("vsrc2:1")  
    res (n5 n6) bsource r=100*(1+(1/3)*pow(v(n5,n6),2))  
}  
tran2 tran stop=1u
```

**Note:** With standard (simple) syntax for resistor/capacitor/inductor, the `bsource` keyword is not required in the statement. However, if the keyword is specified, Spectre treats the resistor/capacitor/inductor as a behavioral source.

#### **bsource Compilation**

Spectre automatically translates bsource components into verilogA models for simulation. As such, they are compiled prior to simulation just like any other verilogA model.

## Checkpoint - Restart (checkpoint)

### Description

Spectre has the ability to save the checkpoint files generated during analyses, and to restart an analysis from its checkpoint file. Checkpoint files can be generated in the following ways:

- Periodically, based on real time (wall clock time).
- Asynchronous UNIX signals.
- By other methods unique to the analyses.

To generate checkpoint files periodically based on real time, set the Spectre option `ckptclock` to the time interval, in seconds. This option is turned on by default with a value of 1800 seconds (30 minutes). Spectre deletes the checkpoint file if the simulation completes normally. If the simulation terminates abnormally, the checkpoint file is not deleted.

If Spectre receives the UNIX signal `USR2`, Spectre immediately writes a checkpoint file. If Spectre receives interrupt signals, such as `QUIT`, `TERM`, `INT`, or `HUP`, Spectre attempts to write a checkpoint file and then exits. For other fatal signals, Spectre may not write a checkpoint file.

The name of the checkpoint file is a combination of the circuit name and the analysis name with the extension `.ckpt`. For example, if the circuit is named `test1` and the transient analysis is named `timeSweep`, the checkpoint file is named `test1.timeSweep.tran.ckpt`.

Spectre keeps only the latest checkpoint file. It creates a new checkpoint file with a temporary name. After the file is successfully written, Spectre deletes the previous checkpoint file created earlier and renames the new file.

Currently, only transient analysis supports checkpoint files and restart.

### Checkpoint

Transient analysis can generate checkpoint files periodically based on the transient simulation time. This is done by using a transient analysis parameter named `ckptperiod`, which is turned off by default. To enable the checkpoint feature, the argument `+checkpoint` must be added to the `spectre` command.

#### Restart

To restart an analysis from a checkpoint file, use the `+recover` option with the `spectre` command. Spectre searches the analyses log for the checkpoint file. If the checkpoint file for the analysis exists, Spectre skips over any previous analyses, and restarts the analysis by using the information from the file.

For more information, see *Recovering From Transient Analysis Terminations* section in the *Spectre Classic Simulator, Spectre APS, Spectre X, and Spectre XPS User Guide*.

## Configuring CMI Shared Objects (cmiconfig)

### Description

Spectre supports the ability to install devices dynamically from shared objects at run time. CMI Configuration files are used to determine and locate the set of shared objects to be installed as follows:

1. Spectre first reads the default CMI configuration file that specifies the default shared objects provided by Cadence.
2. The configuration file specified by the value of the `CMI_CONFIG` environment variable is then read.
3. The third configuration file that Spectre reads is `~/cmiconfig`.
4. Finally, the configuration file specified in the `-cmiconfig` command-line argument is read.

Each CMI configuration file modifies the existing configuration established by the configuration files read before.

The following commands can be used in a CMI configuration file.

**setpath:** Specifies and resets the search path

```
setpath <path> or setpath ( <path1> <path2> ... <pathN> )
```

**prepend:** Adds a path before the current search path

```
prepend <path> or prepend ( <path1> <path2> ... <pathN> )
```

**append:** Adds a path after the current search path

```
append <path> or append ( <path1> <path2> ... <pathN> )
```

**load:** Adds a shared object to the list of shared objects to load

```
loads [path/]<shared_object_name>
```

**unload:** Removes a shared object from the list of shared objects to load

```
unload <shared_object_name>
```

For example, given the following CMI configuration file:

```
append /hm/MMSIM_INSTALL/tools.lnx86/cmi/lib/cmi/5.0
load libbjtx+tfet.so
load libmosx.so
```



## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

The shared objects `libbjtx+tfet.so` and `libmosx.so` are loaded from `/hm/MMSIM_INSTALL/tools.lnx86/cmi/lib/cmi/5.0`, in addition to the default shared objects provided by Cadence.

For more information, refer to the *Using Compiled-Model Interface* chapter in the *Spectre Classic Simulator*, *Spectre APS*, *Spectre X*, and *Spectre XPS User Guide*.

## Built-in Mathematical and Physical Constants (constants)

### Description

Spectre supports the following list of built-in mathematical and physical constants:

**Note:** M\_ is a mathematical constant

M_E	2.7182818284590452354	$\exp(1) = e$
M_LOG2E	1.4426950408889634074	$\log_2(e)$
M_LOG10E	0.43429448190325182765	$\log_{10}(e)$
M_LN2	0.69314718055994530942	$\ln(2)$
M_LN10	2.30258509299404568402	$\ln(10)$
M_PI	3.14159265358979323846	$\pi$
M_TWO_PI	6.28318530717958647652	$2 * \pi$
M_PI_2	1.57079632679489661923	$\pi/2$
M_PI_4	0.78539816339744830962	$\pi/4$
M_1_PI	0.31830988618379067154	$1/\pi$
M_2_PI	0.63661977236758134308	$2/\pi$
M_2_SQRTPI	1.12837916709551257390	$2/\sqrt{\pi}$
M_SQRT2	1.41421356237309504880	$\sqrt{2}$
M_SQRT1_2	0.70710678118654752440	$\sqrt{1/2}$
M_DEGPERRAD	57.2957795130823208772	number of degrees per radian

**Note:** P\_ is a physical constant

P_Q	1.6021918e-19	charge of electron in coulombs
P_C	2.997924562e8	speed of light in vacuum in meters/sec
P_K	1.3806226e-23	Boltzmann's constant in joules/Kelvin

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

P_H	6.6260755e-34	Planck's constant in joules*sec
P_EPS0	8.85418792394420013968e-12	permittivity of vacuum in farads/ meter
P_U0	(4.0e-7 * M_PI)	permeability of vacuum in henrys/ meter
P_CELSIUS0	273.15	zero Celsius in Kelvin

---

These constants can be used in expressions, or anywhere where a numeric value of the expression is expected.

## Convergence Difficulties (convergence)

### Description

If you are having convergence difficulties, try the following suggestions:

1. Evaluate and resolve any notice, warning, or error messages.
2. Ensure that the topology checker is being used (set `topcheck=full` on options statement) and heed any warnings it generates.
3. Perform sanity check on the parameter values by using the parameter range checker (use `+param param-limits-file` as a command line argument) and heed any warnings. Print the minimum and maximum parameter value by using `info` analysis. Ensure that the bounds given for instance, model, output, temperature-dependent, and operating-point (if possible) parameters are reasonable.
4. Small floating resistors connected to high impedance nodes can cause convergence difficulties. Avoid small floating resistors, particularly small parasitic resistors in semiconductors. Instead, use voltage sources or iprobes to measure current.
5. Use realistic device models. Check all component parameters, particularly nonlinear device model parameters, to ensure that they are reasonable.
6. Increase the value of `gmin` (on options statement).
7. Loosen tolerances, particularly absolute tolerances like `iabstol` (on options statement). If tolerances are set too tight, they might preclude convergence.
8. Try to simplify the nonlinear component models to avoid regions that might contribute to convergence problems in the model.

### DC Convergence Suggestions

After you have a solution, write it to a nodeset file by using the `write` parameter, and read it back in on subsequent simulations by using the `readns` parameter.

1. If you have an estimate of what the solution should be, use `nodeset` statements or a nodeset file, and set as many nodes as possible.
2. If convergence difficulties occur when using nodesets or initial conditions, try increasing `rforce` (on options statement).
3. If this is not the first analysis and the solution from the previous analysis is far from the solution for this analysis, set `restart=yes`.

4. If simulating a bipolar analog circuit, ensure that the region parameter on all transistors and diodes is set correctly.
5. If the analysis fails at an extreme temperature, but succeeds at room temperature, try adding a DC analysis that sweeps temperature. Start at room temperature, sweep to the extreme temperature, and write the final solution to a `nodeset` file.
6. Use numeric pivoting in the sparse matrix factorization by setting `pivotdc=yes` (on options statement). Sometimes, it is also necessary to increase the pivot threshold to a value in the range of 0.1 to 0.5 by using `pivrel` (on options statement).
7. Divide the circuit into smaller pieces and simulate them individually. However, ensure that the results are close to what they would be if you had simulated the whole circuit. Use the results to generate nodesets for the whole circuit.
8. Check the connections to ground. Convergence problems might result if there are no connections to ground.
9. If all else fails, replace the DC analysis with a transient analysis and modify all the independent sources to start at zero and ramp to their DC values. Run transient analysis well beyond the time when all the sources have reached their final value (remember that transient analysis is cheap when none of the signals in the circuit are changing) and write the final point to a `nodeset` file. To make transient analysis more efficient, set the integration method to backward Euler (`method=euler`) and loosen the local truncation error criteria by increasing `lteratio`, say to 50. Occasionally, this approach fails, or is slow because the circuit contains an oscillator. Often, for finding the dc solution, the oscillation can be eliminated by setting the minimum capacitance from each node to ground (`cmin`) to a large value.

### Transient Convergence Suggestions

1. Ensure that a complete set of parasitic capacitors is used on nonlinear devices to avoid jumps in the solution waveforms. On MOS models, specify nonzero source and drain areas.
2. Use the `cmin` parameter to install a small capacitor from every node in the circuit to ground. This usually eliminates any jump in the solution.

## The `dcopt` command line option (`dcopt`)

### Description

The `dcopt` option speeds up the DC simulation in Spectre and APS for post-layout circuits that consist of large number of parasitic resistors and capacitors, which require significant time in DC simulation. It can also be used to solve the non-convergence issues in DC simulations of any other circuit.

**Note:** In some cases, the DC solution obtained using the `dcopt` command-line option may not be as accurate as the true DC solution.

### Definition

`+dcopt` in the command line

## encryption (encryption)

### Description

Encryption enables you to protect your proprietary parameters, subcircuits, models, netlists, and release your libraries to your customers without revealing sensitive information.

#### 1. Define Encryption blocks in the netlist

Keywords (`protect`, `unprotect`) are used for defining an encryption block. The dot keywords are used in the context of the spice mode: `.protect`, `.unprotect`. `protect`, `unprotect` can be abbreviated to `prot`, `unprot` (or, `.prot`, `.unprot` in spice).

If the whole file needs to be encrypted, one method is to put `protect` at the beginning of the file and `unprotect` at the end of the file. Alternatively, you can use the `-all` option of the `spectre encrypt` command.

Examples of netlist with protected blocks:

#### Example: Protection of subckts

```
protect
subckt sub1 ( ... )
....
ends sub1
subckt sub2 ( ... )
.....
ends sub2
unprotect
```

#### 2. Encrypt the netlist

`spectre_encrypt` is a standalone encryptor, and is used as follows:

```
spectre_encrypt [-i input_file] [-o output_file] [-all]
```

where

`[-i Input_file]`: Netlist to be encrypted

`[-o output _file]`: Output file of the encrypted netlist

`[-all]`: The whole file will be encrypted in the input netlist

**Note:** You can separately encrypt the include files or the library files in the netlist. Spectre encryptor does not encrypt the included files automatically.

### 3. Simulate the encrypted netlist

There is no difference in how you run Spectre on an encrypted or unencrypted netlist. Spectre automatically decrypts an encrypted netlist.

For encrypted netlists, Spectre turns on the protection for devices, models, signals, and parameters in the encrypted blocks. Any error or warning messages and the outputs from the protected information is suppressed or filtered out.

The following list describes how protection is implemented:

- ❑ Circuit inventory does not include encrypted parts.
- ❑ The `info` command suppresses all information on encrypted parts.
- ❑ Errors on encrypted parts are reported as:  
`Error has occurred within the encrypted block (no details are given).`
- ❑ If a portion of the model is encrypted, the entire model is encrypted.
- ❑ Any command that references the protected elements results in an error message reporting that those elements do not exist.
- ❑ Protected device and model parameters cannot be altered directly through `alter`. However, if they depend on other alterable parameters, protected parameters are recalculated. Protected devices and models can be replaced in the `altergroup`.
- ❑ Protected nodes are output in an encrypted format when the `ic` file is requested (similarly, for nodes in checking point and restart).
- ❑ The encryption feature is not available in models using CMI 2.0.

### 4. Output operating points on protected devices

By default, all information about the protected devices is suppressed and is not visible. However, IP providers have the control to expose the operating points of the protected devices to the end users for back annotation.

Keywords (`visible`, `invisible`) or (`.visible`, `.invisible`) in the spice netlist content are defined to expose the operating points of encrypted devices.

The operating point of the protected devices between visible and invisible is not suppressed by adding `what=oppoints` to the `visible` statement.

For example:

```
prot
x1 n1 n2 n3 n4 nmos
visible what=oppoints
x2 n5 n6 n7 n8 nmos
```



## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

```
x3  n9 n10 n10 n8 pmos
invisible
X4  n11 n12 n13 n13 pmos
unprot
```

For more information, refer to the [Encryption](#) chapter in *Spectre Classic Simulator*, *Spectre APS*, *Spectre X*, and *Spectre XPS User Guide*.

## Expressions (expressions)

### Description

An expression is a construct that combines operands with operators to produce a result that is a function of the values of the operands and the semantic meaning of the operators. Any legal operand is also an expression in itself. Legal operands include numeric constants and references to the top-level netlist parameters or subcircuit parameters. Calls to algebraic and trigonometric functions are also supported. The supported operators, algebraic, and trigonometric functions are listed after the examples.

### Examples:

```
simulator lang=spectre
parameters p1=1 p2=2          // declare some top-level parameters
r1 (1 0) resistor r=p1        // the simplest type of expression
r2 (1 0) resistor r=p1+p2     // a binary (+) expression
r3 (1 0) resistor r=5+6/2     // expression of constants, = 8
x1 s1 p4=8                    // instantiate a subcircuit, defined in the following lines
subckt s1
parameters p1=4 p3=5 p4=6     // subcircuit parameters
r1 (1 0) resistor r=p1        // another simple expression
r2 (1 0) resistor r=p2*p2     // a binary multiply expression
r3 (1 0) resistor r=(p1+p2)/p3 // a more complex expression
r4 (1 0) resistor r=sqrt(p1+p2) // an algebraic function call
r5 (1 0) resistor r=3+atan(p1/p2) // a trigonometric function call
r6 (1 0) RESMOD r=(p1 ? p4+1 : p3) // the ternary operator
ends
// a model card, containing expressions
model RESMOD resistor tc1=p1+p2 tc2=sqrt(p1*p2)
// some expressions used with analysis parameters
time_sweep tran start=0 stop=(p1+p2)*50e-6 // use 5*50e-6 = 150 us
// a vector of expressions (see notes on vectors below)
dc_sweep dc param=p1 values=[0.5 1 +p2 (sqrt(p2*p2)) ] // sweep p1
```

### Using Expressions:

The Spectre native netlist language allows expressions to be used where numeric values are expected on the right-hand side of an "=" sign, or within a vector, where the vector itself is on the right-hand side of an "=" sign. Expressions can be used when specifying device or analysis instance parameter values (for example, specifying the resistance of a resistor or the stop time of a transient analysis, as outlined in the preceding example), when specifying

## Spectre Circuit Simulator Reference

### Other Simulation Topics

model parameter values in model cards (for example, specifying `bf=p1*0.8` for a bipolar model parameter, `bf`), or when specifying initial conditions and nodesets for individual circuit nodes.

### Operators

The following operators are supported, listed in the order of decreasing precedence. Parentheses can be used to change the order of evaluation. For a binary expression, such as `a+b`, `a` is the first operand and `b` is the second operand. All operators are left associative, with the exceptions of the "to the power of" operator (`**`) and the ternary operator (`? :`), which are right associative. For logical operands, any nonzero value is considered true. The relational and equality operators return a value of 1 to indicate true, or 0 to indicate false. There is no short circuiting of logical expressions involving `&&` and `||`.

Operator	Symbol(s)	Value
Unary +, Unary -	<code>+</code> , <code>-</code>	Value of operand, negative of operand.
To the power of	<code>**</code>	First operand raised to the power of second operand
Multiply, Divide	<code>*</code> , <code>/</code>	Sum, Difference of operands
Binary Plus/Minus	<code>+</code> , <code>-</code>	Sum, Difference of operands
Shift	<code>&lt;&lt;</code> , <code>&gt;&gt;</code>	First operand shifted left or right by the number of bits specified by the second operand
Relational	<code>&lt;</code> , <code>&lt;=</code> , <code>&gt;</code> , <code>&gt;=</code>	Less than, less than or equal, greater than, greater than or equal
Equality	<code>==</code> , <code>!=</code>	True if operands are equal, not equal
Bitwise AND	<code>&amp;</code>	Bitwise AND (of integer operands)
Bitwise Exclusive NOR	<code>~^</code> (or <code>^~</code> )	Bitwise Exclusive NOR (of integer operands)
Bitwise OR	<code> </code>	Bitwise OR (of integer operands)
Logical AND	<code>&amp;&amp;</code>	True only if both operands are true.
Logical OR	<code>  </code>	True if either operand is true
Ternary Operator	<code>(cond) ? x : y</code>	Returns <code>x</code> if <code>cond</code> is true, and <code>y</code> if <code>cond</code> is false, where <code>x</code> and <code>y</code> are expressions

## Spectre Circuit Simulator Reference

### Other Simulation Topics

#### Algebraic and Trigonometric Functions

The trigonometric and hyperbolic functions expect their operands to be specified in radians. The atan2() and hypot() functions are useful for converting from Cartesian to polar form.

Function	Description	Domain
log(x)	Natural logarithm	$x > 0$
ln(x)	Natural logarithm	$x > 0$
log10(x)	Decimal logarithm	$x > 0$
exp(x)	Exponential	$x < 80$
sqrt(x)	Square Root	$x > 0$
min(x,y)	Minimum value	All x, all y
max(x,y)	Maximum value	All x, all y
abs(x)	Absolute value	All x
pow(x,y)	x to the power of y	All x, all y
int(x)	integer value of x	All x
floor(x)	largest integer $\leq x$	All x
ceil(x)	smallest integer $\geq x$	All x
fmod(x,y)	floating point modulus	All x, all y, except y=0
sgn(x)	The sign of x	All x
sign(x,y)	sgn(y)*fabs(x)	All x, all y
sin(x)	Sine	All x
cos(x)	Cosine	All x
tan(x)	Tangent	All x, except $x = n \cdot (\pi/2)$ , where n odd
asin(x)	Arc-sine	$-1 \leq x \leq 1$
acos(x)	Arc-cosine	$-1 \leq x \leq 1$
atan(x)	Arc-tangent	All x

## Spectre Circuit Simulator Reference

### Other Simulation Topics

<code>atan2(x,y)</code>	Arc-tangent of x/y	All x, all y
<code>hypot(x,y)</code>	$\sqrt{x^2 + y^2}$	All x, all y
<code>sinh(x)</code>	Hyperbolic sine	All x
<code>cosh(x)</code>	Hyperbolic cosine	All x
<code>tanh(x)</code>	Hyperbolic tangent	All x
<code>asinh(x)</code>	Arc-hyperbolic sine	All x
<code>acosh(x)</code>	Arc-hyperbolic cosine	$x \geq 1$
<code>atanh(x)</code>	Arc-hyperbolic tangent	$-1 \leq x \leq 1$

User-defined functions are also supported. See `spectre -h functions` for a description of user-defined functions.

A large number of built-in mathematical and physical constants are available for use in expressions. See `spectre -h constants` for a list of these constants.

The `table_param()` function is also supported. This function can be used to return the target column value that matches the integer inputs and the interpolation of the float inputs.

Usage:

`table_param( str, n, int1 ..., intn, m, float1 ..., floatm, offsetOut)`

Parameter	Description
<code>str</code>	Name of the table file.
<code>n</code>	Number of integer input values where $0 \leq n \leq \text{INT\_MAX}$
<code>int1...intn</code>	Integer input value.
<code>m</code>	Number of float input values where $0 \leq m \leq \text{INT\_MAX}$
<code>float1...floatm</code>	Float input value.

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

---

<code>offsetOut</code>	Column offset of the output value starting with 1.
------------------------	--

---

Example:

```
simulator lang=spice
```

```
.param p1 = 'table_param( str("example.table"), 3, 4, 5, 6, 3, 0.01, 0.02, 0.03, 1 )'
```

example.table:

```
#int1 int2 int3 float1 float2 float3 out1 out2 out3 out4
```

```
4 5 6 0.01 0.02 0.03 7 8 9 10
```

In this `table_param` function, the value of `n` and `m` is 3, which means there are three integer columns, three float columns, and four output columns. In addition, `offsetOut = 1` indicates that we are referring to the `out1` column

### Using Expressions in Vectors

Expressions can be used as vector elements, as in the following example:

```
dc_sweep dc param=p1 values=[0.5 1 +p2 (sqrt(p2*p2)) ] // sweep p1
```

When expressions are used within vectors, anything other than constants, parameters, or unary expressions (unary `+`, unary `-`) must be surrounded by parentheses. Vector elements should be space separated for clarity, though this is not mandatory. The preceding "dc\_sweep" example shows a vector of four elements, namely 0.5, 1, `+p2`, and `sqrt(p2*p2)`.

**Note:** The square root expression is surrounded by parentheses.

## The fastdc command line option (fastdc)

### Description

The `fastdc` option speeds up the DC simulation in Spectre and APS for large-scale circuits and for cases where DC convergence is slow or there is difficulty in DC convergence.

The `fastdc` option provides a way to quickly obtain the approximate DC solution to start transient analysis.

**Note:** The DC solution obtained using the `fastdc` command-line option may not be as accurate as the true DC solution. The solution accuracy depends on the value specified for `fastdc`. The available values are 0, 1, 2, 3, and 4 (default).

0 - The most accurate DC result and closest to true DC solutions with the slowest performance. It should be used for the circuits that require accurate DC solution, such as analog circuits.

1 - The conservative DC result with slow performance. It should be used for the circuits that are sensitive to DC solution, such as analog dominated circuits.

2 - The reasonable DC result with moderate performance. It should be used for the circuits that are lightly sensitive to DC solution, such as mixed-signal circuits.

3 - The liberate DC results with fast performance. It should be used for the circuits that are less sensitive to DC solution, such as digital dominated circuits.

4 - The least accurate DC result with the fastest performance. It should be used for the circuits that are not sensitive to DC solution, such as digital and memory circuits.

### Definition

**Note:** `+fastdc` (equal to `+fastdc=4`) or `+fastdc=x` (0, 1, 2, 3, 4) in the command line

## Fault List for Transient Fault Analysis (faults)

### Description

The faults block is used to specify the faults to be simulated during Transient Fault Analysis. It can contain the following types of faults: bridges (or equivalently shorts), opens, custom, and parametric.

The bridge (short) block specifies the resistance value between any two nodes of the circuit.

The node names are specified hierarchically. Local node names can be used when the `sub` or `dev` parameter is specified in faults block.

Optional `dev` parameter is used to specify device instance associated with the bridge.

The open block specifies the nodes that will be split into two.

Parameter `r` specifies resistance between the new fault node and the original node.

Optional parameter `c` specifies capacitance between the new fault node and the original node.

The last block describes topology change: how the instances connected to the original node will be connected to the new fault node. All terminals of the instances specified in the open statement will be connected to the new node. All terminals that are not specified will remain connected to the original node.

If no instance list is specified, a warning is issued and the fault is ignored. The terminals are specified by name or numerically starting from 1. If the specified terminal number 1 is the first terminal of the instance. If the specified terminal number of an instance in the instance list is not connected to the original node, an error is generated. If no terminal list is specified for an instance, all the terminals connected to the original node will remain connected to the original node. The node and instance names can be specified hierarchically.

The parametric block specifies alter statements for a single parameter. Device, subcircuit, model, or global parameters can be used in alter statement.

Within the faults block, the bridge, short, open, parametric, and custom blocks are optional. If none are specified, a warning is issued and the faults block is ignored. If more than one bridge, short, open, parametric, or custom block is specified within a faults block, they are concatenated and treated as one block. Every fault name in a faults block must be unique. Duplicate names result in an error.



More than one faults block may be specified in the netlist and each faults block must have a unique name and a set of parameters associated with it.

Fault grading can be performed by specifying `weight_factor` for each fault block, and `weight_expr` to apply weighting function for each fault in the block.

Optional weight parameter can be used to assign weight value for each fault.

### Definition

### Example

```
parameters resBridge=10 resOpen=1e9
faultlist1 faults {
    bridge {
        bridge_1 (g 0) r=resBridge
        bridge_2 (d s) r=10
    }
    open {
        open_1 (a) r=resOpen { M2:d M1:g }
    }
}
```

## Dynamic Force Voltage Node (force\_voltage)

### Description

The command is used to force the node voltage.

### Definition

Name `force_voltage` <parameter=value> ...

### Parameters

#### *Design parameters*

---

1	<code>node=[...]</code>	Nodes to which the check is applied.
2	<code>voltage</code>	The voltage value for the nodes to be forced.
3	<code>time=0 sec</code>	The start time of force.
4	<code>slope=0.1e-9 sec/v</code>	The voltage change rate.
5	<code>rforce=1 Ohm</code>	The resistance used to force voltage.

---

#### *Wildcard scoping*

---

6	<code>xnode=" [...]"</code>	Nodes to be excluded from the check. Default is none.
7	<code>inst=[...]</code>	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
8	<code>xinst=[...]</code>	Subcircuit instances to be excluded from the check. Default is none.
9	<code>subckt=[...]</code>	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits ( <code>subckt=*</code> ).
10	<code>xsubckt=[...]</code>	The instances of the specified subcircuits that are excluded from the check. Default is none.

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

---

11	<code>depth=8</code>	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.
----	----------------------	--

---

## User Defined Functions (functions)

### Description

Spectre's user-defined function capability allows you to build upon the provided set of built-in mathematical and trigonometric functions. You can write your own functions, and call these functions from within any expression. The syntax for calling a user-defined function is the same as the syntax for calling a built-in algebraic or trigonometric function. The user-defined functions must be defined before they are referenced (called). Arguments to user-defined functions are taken as real values, and the functions return real values. A user-defined function may contain only a single statement in braces, and this statement must return an expression (which is typically an expression involving the function arguments). The return expression may reference the built-in parameters `temp` and `tnom`. User-defined functions must be declared only at the top level, and must not be declared within subcircuits. User-defined functions may be called from anywhere an expression can be currently used in Spectre. User-defined functions may call other functions (both user-defined and built-in), however, any user-defined function needs to be declared before it can be called. User-defined functions can override built-in mathematical and trigonometric functions.

**Note:** Only real values for arguments and return values are supported in this release.

See `spectre -h expressions` for a list of built-in algebraic and trigonometric functions.

### Definition

```
real myfunc( [real arg1, ...real argn] ) {  
  
}
```

### Examples

```
real myfunc( real a, real b ) {  
    return a+b*2+sqrt(a*sin(b));  
}
```

An example of a function calling a previously defined function is as follows:

```
real yourfunc( real a, real b ) {  
    return a+b*myfunc(a,b);    // call "myfunc"  
}
```

The final example shows how a user-defined function may be called from an expression in the Spectre netlist:

```
r1 (1 0) resistor r=myfunc(2.0, 4.5)
```

## Global Nodes (global)

### Description

The global statement allows a set of nodes to be designated as common to the main circuit and all subcircuits. Thus, components inside subcircuits can be attached to global nodes, even though the subcircuit terminals are not attached to these nodes.

Any number of global nodes may be specified using the global statement. To do this, follow the keyword global with a list of the node names that you wish to declare as global. The first node name that appears in this list is taken to be the name of the ground node. Ground is also known as the datum or reference node. If a global statement is not used, 0 is taken to be the name of the ground node.

Ground is always treated as global even if a global statement is not used. Multiple global statements are supported.

### Definition

```
global <ground> <node> ...
```

## IBIS Component Use Model (ibis)

### Description

IBIS (I/O Buffer Information Specification) is a standard for electronic behavioral specification of integrated circuit (IC) input/output analog characteristics. It allows you to define a model for the IC component package, and a buffer model for each pin. IBIS standard also allows you to describe a board-level component containing several components on a common substrate or printed circuit board (PCB). For example, a `SIMM` module is a board-level component that is used to attach several DRAM components on the PCB to a motherboard through edge connector pins. Board pins, components on the board, and connections between them are defined in an Electrical Board Description file with extension `.ebd`. Component pins, buffers, and package descriptions are in a separate IBIS file with extension `.ibs`. An Additional Package file with extension `.pkg` can be used to describe advanced package models.

Spectre supports IBIS 6.0 and previous versions. Some keywords are not supported.

IBIS files can be referenced in Spectre netlist by using the `ibis_include` statement:

```
ibis_include "DRAM.ibs" [options]
```

IBIS file "DRAM.ibs" is translated into Spectre netlist format by using `ibis2subckt` utility. The output file, "DRAM.scs", containing subcircuit definitions for each IBIS component, board, and package found in the input IBIS file, is included in the netlist. The list of options may consist of: `corner={typ|min|max|slow|fast}`, `swsel=<int>` and `mdsel=<int>`. These options are transferred to `ibis2subckt`. They are used to select IBIS buffer model corner, change position of series-switch models, and choose the required models from model selector list.

IBIS component and board subcircuits can be instantiated in a Spectre netlist along with regular Spectre primitives. Subcircuit name is the same as the component name, but is appended with the suffix `_ibis`. Subcircuit terminals are component pins, arranged in the order they are listed in the IBIS file. Each pin terminal is followed by a number of signal terminals, depending on the type of the pin buffer model. For example, if `m1` is defined in IBIS file as an input buffer model, and `m2` - as I/O type, the following IBIS component:

```
[Component] IC
[Pin]  signal_name  model_name  R_pin    L_pin    C_pin
p1      s1          GND
p3      s3          m1
p4      s4          NC
p5      s5          m2
p2      s2          POWER
```

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

can be instantiated in the netlist as:

```
x_ic ( p1 p3 s3_in p4 p5 s5_in s5_out s5_en p2 ) IC_ibis
```

`ibis2subckt` can also be used as a stand-alone utility with the following command-line arguments:

```
ibis2subckt -in <IBIS files> -out <subckt file> -corner {typ|min|max|slow|fast} -  
swsel <int> -mdsel <int>
```

Default values are:

```
corner typ  
mdsel -1  
swsel -1
```

### SPICE NETLIST SUPPORT

For the IBIS component, Spectre also supports the `SPICE .IBIS` and `.EBD` statements.

The supported parameters of `.IBIS` syntax are `file`, `component`, `mod_sel`, `package`, and `typ`.

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

#### .IBIS Parameters

---

<code>file = &lt;string&gt;</code>	Specifies the IBIS file name with suffix <code>.ibs</code> .
<code>component = &lt;string&gt;</code>	Specifies the used component name in the <code>.ibs</code> file.
<code>mod_sel = &lt;string1=string2&gt;</code>	Maps the model selector name ( <code>string1</code> ) to the actual model name ( <code>string2</code> ), given as [Model selector] in the <code>.ibis</code> file. Multiple selectors are supported.
<code>package = &lt;3 0 1 2&gt;</code>	Specifies the type of package. Default value is 3 (use the best available package model). <code>package=0</code> means no package is used. <code>package=1</code> means that an RLC package is used with the same values for all pins provided in the [Package] section. <code>package=2</code> means that an RLC package is used with individual RLC values for each pin provided in the [Pin] section. <code>package=3</code> means that an advanced package model is used in the [Package Model] section.
<code>typ = &lt;typ min max fast slow&gt;</code>	Specifies the corner of the IBIS buffer. Default value is <code>typ</code> (typical).

---

The supported parameters of the `.EBD` syntax are `file`, `model`, and `component`.

#### .EBD Parameters

---

<code>file = &lt;string&gt;</code>	Specifies the EBD file name with the suffix <code>.ebd</code>
<code>model = &lt;string&gt;</code>	Specifies name of the board-level model provided in <code>.ebd</code> file
<code>component = &lt;string&gt;</code>	Specifies the component name of the ibis buffer. Multiple components are supported.

---

#### Examples

```
.ibis I1
+ file = file.ibs
+ component = Component
+ mod_sel = DQ=DQ1,CQ=CQ1
+ package=0
```



## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

```
+ typ=slow

.ebd pkg
+ file = file.ebd
+ model = XXXX
+ component = Component1
+ component = Component2
```

### Supported IBIS Keywords in Spectre

**Note:** The supported keywords list below is based on IBIS version 6.0.

The following shows the sections and keywords in the IBIS file that are supported in Spectre:

```
.ibis FILE
    FILE Header Section
        [IBIS Ver]
        [Comment Char]
        [File Name]
        [File Rev]
        [Date]
        [Source]
        [Notes]
        [Disclaimer]
        [Copyright]
    [Component]                Si Location, Timing_location
        [Manufacturer]
        [Package]              R_pkg, L_pkg, C_pkg
        [Pin]                  signal_name, model_name, R_pin, L_pin, C_pin
        [Package Model]
        [Pin Mapping]          pulldown_ref, pullup_ref, gnd_clamp_ref,
                                power_clamp_ref, ext_ref
        [Diff Pin]             inv_pin, vdiff, tdelay_typ, tdelay_min, tdelay_max
        [Series Pin Mapping]    pin_2, model_name, function_table_group
        [Series Switch Groups]  On (m), Off (m)
    [Model Selector]
        [Model]                Model_type, Polarity, Enable, Vinl, Vinh, C_comp,
                                C_comp_pullup, C_comp_pulldown,
                                C_comp_power_clamp, C_comp_gnd_clamp, Vmeas,
                                Cref, Rref, Vref
        [Model Spec]           Vinh, Vinl
        [Add Submodel]
        [Driver Schedule]
        [Temperature Range]
```

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

[Voltage Range]	
[Pullup Reference]	
[Pulldown Reference]	
[Power Clamp Reference]	
[GND Clamp Reference]	
[C Comp Corner]	C_comp, C_comp_pullup, C_comp_pulldown, C_comp_power_clamp, C_comp_gnd_clamp
[Pulldown]	
[Pullup]	
[GND Clamp]	
[POWER Clamp]	
[ISSO PU]	
[ISSO PD]	
[Rgnd]	
[Rpower]	
[Rac]	
[Cac]	
[On]	
[Off]	
[R Series]	
[L Series]	
[Rl Series]	
[C Series]	
[LC Series]	
[RC Series]	
[Series Current]	
[Series MOSFET]	Vds
[Ramp]	dV/dt_r, dV/dt_f, R_load
[Rising Waveform]	R_fixture, V_fixture, V_fixture_min, V_fixture_max, C_fixture, L_fixture,
[Composite Current]	
[Falling Waveform]	_fixture, V_fixture, V_fixture_min, V_fixture_max, C_fixture, L_fixture,
[Composite Current]	
[External Model]	Language, Corner, Parameters, Ports, D_to_A, A_to_D
[End External Model]	
[Sub Model]	Sub model type
[Sub Model Spec]	V_trigger_r, V_trigger_f
[POWER Pulse Table]	
[GND Pulse Table]	
[Pulldown]	

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

```
[Pullup]
[GND Clamp]
[POWER Clamp]
[Ramp]                                dV/dt_r, dV/dt_f, R_load
[Rising Waveform]                    R_fixture, V_fixture, V_fixture_min,
                                     V_fixture_max, C_fixture, L_fixture,
[Falling Waveform]                  R_fixture, V_fixture, V_fixture_min,
                                     V_fixture_max, C_fixture, L_fixture,
[Define Package Model]
[Manufacturer]
[OEM]
[Description]
[Number of Sections]
[Number of Pins]
[Pin Numbers]                        Len, L, R, C, Fork, Endfork
[Model Data]
    [Resistance Matrix]              Banded_matrix, Sparse_matrix, Full_matrix
    [Bandwidth]
    [Row]
    [Inductance Matrix]              Banded_matrix, Sparse_matrix, Full_matrix
    [Bandwidth]
    [Row]
    [Capacitance Matrix]             Banded_matrix, Sparse_matrix, Full_matrix
    [Bandwidth]
    [Row]
    [End Model Data]
[End Package Model']
[END]
```

The following shows the sections and keywords in the package file that are supported in Spectre:

.pkg FILE

```
FILE Header Section
[IBIS Ver]
[Comment Char]
[File Name]
[File Rev]
[Date]
[Source]
[Notes]
[Disclaimer]
[Copyright]
```

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

```
[Define Package Model]
  [Manufacturer]
  [OEM]
  [Description]
  [Number of Sections]
    Number of Pins]
  [Pin Numbers]          Len, L, R, C, Fork, Endfork
  [Model Data]
    [Resistance Matrix]  Banded_matrix, Sparse_matrix, Full_matrix
    [Bandwidth]
    [Row]
    [Inductance Matrix]  Banded_matrix, Sparse_matrix, Full_matrix
    [Bandwidth]
    [Row]
    [Capacitance Matrix] Banded_matrix, Sparse_matrix, Full_matrix
    [Bandwidth]
    [Row]
    [End Model Data]
  [End Package Model']
[END]
```

The following shows the sections and keywords in the EBD file that are supported in Spectre:

.pkg FILE

```
FILE Header Section
  [IBIS Ver]
  [Comment Char]
  [File Name]
  [File Rev]
  [Date]
  [Source]
  [Notes]
  [Disclaimer]
  [Copyright]
[Begin Board Description]
  [Manufacturer]
  [Number of Pins]
  [Pin List]          signal_name
  [Path Description]  Len, L, R, C, Fork, Endfork, Pin, Node
  [Reference Designator Map]
  [End Board Description]
[END]
```

## Initial Conditions (ic)

### Description

The `ic` statement is used to provide initial conditions for nodes in transient analysis. It can occur multiple times in the input, and the information provided in all the occurrences is collected. Initial conditions are accepted only for inductor currents and node voltages where the nodes have a path of capacitors to ground. For more information, read the description of transient analysis.

**Note:** Specifying `cmin` for a transient analysis, does not satisfy the condition that a node has a capacitive path to ground.

### Definition

```
ic <X[:param]=value> ...
```

This statement takes a list of signals with the state information to the DC and transient analyses. `X` can be a node, a component, or a subcircuit and `param` can either be a component output parameter or a terminal index. To specify a class of signals, use the pattern matching characters `*` for any string and `?` for any character.

The concept of nodes for the statement has been generalized to signals where a signal is a value associated with a topological node of the circuit or some other unknown that is solved by the simulator, such as the current through an inductor or the voltage of the internal node in a diode. Topological nodes can either be at the top-level or in a subcircuit.

For example:

```
ic 7=0 out=1 OpAmp1.comp=5 L1:1=1.0u
```

where, `7=0` implies that node 7 should start at 0V, node `out` should start at 1V, node `comp` in subcircuit `OpAmp1` should start at 5V, and the current through the first terminal of `L1` should start at 1uA.

For more information, refer to *The `ic` and `nodeset` Statements* in the *Spectre Classic Simulator, Spectre APS, Spectre X, and Spectre XPS User Guide*.

## The Structural if-statement (if)

### Description

The structural if-statement can be used to conditionally instantiate other instance statements.

### Definition

```
if <condition> <statement1> [ else <statement2> ]
```

The condition is a Boolean expression based on the comparisons of various arithmetic expressions that are evaluated during circuit hierarchy flattening. The statement1 and statement2 fields can be ordinary instance statements, if-statements, or a list of these within braces ({}). The ordinary instance statements need a newline to terminate them. The `else` part is optional. When if-statements are nested without braces, an `else` matches the closest previous unmatched `if` at the same level.

It is possible to have duplicate instance names within the if statement under strict topological conditions. These conditions are as follows:

- References to an instance with duplicate names is possible only within a structural if statement that has both an "if" part and an "else" part.
- Both the "if" part and the "else" part must be a simple one-statement block, or another structural if statement to which these same rules apply.
- The duplicate instances must have the same number of terminals and be bound to the same list of nodes.
- The duplicate instances must refer to the same primitive or model.
- Where duplicate instances refer to a model, the underlying primitive must be the same.

This feature allows automatic model selection based on any netlist or subcircuit parameter. As an example, consider using Spectre's inline subcircuits and structural if statement to implement automatic model selection based on bipolar device area. Here, the duplicate instances are the inline components.

```
model npn_default bjt is=3.2e-16 va=59.8
model npn10x10 bjt is=3.5e-16 va=61.5
model npn20x20 bjt is=3.77e-16 va=60.5
// npn_mod chooses scaled models binned on area!
// if ( area < 100e-12 ) use model npn10x10
// else if ( area < 400e-12 ) use model npn20x20
// else use model npn_default
```

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

```
inline subckt npn_mod (c b e s)
  parameters area=5e-12
  if ( area < 100e-12 ) {
    npn_mod (c b e s) npn10x10 // 10u * 10u, inline device
  } else if ( area < 400e-12 ) {
    npn_mod (c b e s) npn20x20 // 20u * 20u, inline device
  } else {
    npn_mod (c b e s) npn_default // 5u * 5u, inline device
  }
ends npn_mod
q1 (1 2 0 0) npn_mod area=350e-12 // gets 20x20 model
q2 (1 3 0 0) npn_mod area=25e-12 // gets 10x10 model
q3 (1 3 0 0) npn_mod area=1000e-12 // gets default model
```

For additional information, refer to the *Conditional Instances* section in the *Spectre Classic Simulator, Spectre APS, Spectre X, and Spectre XPS User Guide*.

## Include File (include)

### Description

File inclusion allows the circuit description to be spread over several files. The `include` statement itself is replaced by the contents of the file named. An included file may also contain include statements. If the name given is not an absolute path specification, the path is taken relative to the directory of the file currently being read.

To read existing SPICE library and model files, Spectre automatically switches to SPICE input mode when it opens an include file. Thus, all files that use the Spectre native language must begin with a `simulator lang=spectre` statement. The one exception is files that end with a `".scs"` file extension, which are treated specially and are read in Spectre input mode. This language mode treatment applies to files included by both the Spectre `include` statement, and the CPP `#include` statement.

After reading the include file, Spectre restores the language processing mode to what it was before the file was included, and continues reading the original file starting at the line after the include statement. Lines cannot be continued across file boundaries.

The CPP `#include` statement differs from Spectre `include` statement in that the CPP macro processing is not performed on files included by Spectre, but is performed on files included by CPP. If your netlist contains a `#include` statement, you must run CPP to perform this inclusion; otherwise, an error occurs.

If the file to be included cannot be found in the same directory as the including file, both the Spectre `include` and CPP `#include` search for the file to be included along the search path specified by the `-I` command-line arguments.

The Spectre `include` statement allows you to include a library section. The following is the syntax for specifying a library reference:

```
include "file" section=sectionName
```

where, `file` is the name of the library file to be included, and `sectionName` matches the name of the section defined in the library. The library reference statement looks like an `include` statement, except for the specification of the library section. When the file is being inserted, only the named section is actually included.

The Spectre `include` statement allows you to embed special characters in the name of the file to be included. The Spectre `include` statement automatically expands the `~` character to the user's home directory, will expand the environment variables and `%` codes, such as

```
include "~/models/${SIMULATOR}_pd/npn.scs"
```



## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

which looks in the directory given by the environment variable `SIMULATOR`, followed by `_pd`, which is under the `models` directory in the user's home directory.

**Note:** These special character features are not available with the CPP `#include` statement.

For additional information, see *Input Data from Multiple Files* in the *Spectre Classic Simulator*, *Spectre APS*, *Spectre X*, and *Spectre XPS User Guide*.

### Definition

```
include "filename"
```

## Spectre Netlist Keywords (keywords)

### Description

The following are Spectre keywords, including netlist keywords and built-in mathematical and physical constants. Spectre has special use for these keywords, and they are reserved in certain contexts. You should follow the rules given below when using them to prevent errors.

- Netlists keywords cannot be used as instance names, subckt names, model names, or function names.
- Built-in mathematical and physical constants cannot be used as node names, instance names, subckt names, model names, function names, or parameter names.

---

Keyword	Keyword Type
M_1_PI	Mathematical Constant
M_2_PI	Mathematical Constant
M_2_SQRTPI	Mathematical Constant
M_DEGPERRAD	Mathematical Constant
M_E	Mathematical Constant
M_LN10	Mathematical Constant
M_LN2	Mathematical Constant
M_LOG10E	Mathematical Constant
M_LOG2E	Mathematical Constant
M_PI	Mathematical Constant
M_PI_2	Mathematical Constant
M_PI_4	Mathematical Constant
M_SQRT1_2	Mathematical Constant
M_SQRT2	Mathematical Constant
M_TWO_PI	Mathematical Constant
P_C	Mathematical Constant
P_CELSIUS0	Mathematical Constant
P_EPS0	Mathematical Constant

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

---

P_H	Mathematical Constant
P_K	Mathematical Constant
P_Q	Mathematical Constant
P_U0	Mathematical Constant
altergroup	Netlist Keyword
correlate	Netlist Keyword
else	Netlist Keyword
end	Netlist Keyword
ends	Netlist Keyword
export	Netlist Keyword
for	Netlist Keyword
freq	Netlist Reserved Word
function	Netlist Keyword
global	Netlist Keyword
ic	Netlist Keyword
if	Netlist Keyword
in	Netlist Keyword
inline	Netlist Keyword
input	Netlist Keyword
invisible	Netlist Keyword
library	Netlist Keyword
local	Netlist Keyword
march	Netlist Keyword
model	Netlist Keyword
nodeset	Netlist Keyword
out	Netlist Keyword
output	Netlist Keyword
parameters	Netlist Keyword
paramset	Netlist Keyword

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

---

plot	Netlist Keyword
print	Netlist Keyword
protect	Netlist Keyword (The first four letters of the keyword (in lowercase or uppercase) can be used as an abbreviation. For example, prot and PROT are valid keywords).
pwr	Netlist Keyword
real	Netlist Keyword
return	Netlist Keyword
save	Netlist Keyword
scalem	Netlist Reserved Word
sens	Netlist Keyword
statistics	Netlist Keyword
subckt	Netlist Keyword
temp	Netlist Reserved Word
temper	Netlist Reserved word
time	Netlist Reserved Word
tnom	Netlist Reserved Word
to	Netlist Keyword
truncate	Netlist Keyword
unprotect	Netlist Keyword. (The first six letters of the keyword (in lowercase or uppercase) can be used as an abbreviation. For example, unprot and UNPROT are valid keywords).
vary	Netlist Keyword
visible	Netlist Keyword

---

## Library - Sectional Include (library)

### Description

Library inclusion allows the circuit description to be spread over several files. The library statement itself is replaced by the contents of the specified section of the library file. A library section may also contain library reference statements. If the file name given is not an absolute path specification, the path is taken relative to the directory of the file currently being read.

There are two types of library statements. One that references a library section, and another that defines a library section. The definition of a library section is prohibited in the netlist.

In order to read existing SPICE library and model files, Spectre automatically switches to SPICE input mode when it opens a library file. Thus, all files that use the Spectre native language must contain a `simulator lang=spectre` statement within each section of the library or the file can have a `.scs` filename extension. After reading the library section, Spectre restores the language processing mode and continues reading the original file starting at the line after the library statement. Lines cannot be continued across file boundaries.

Spectre allows only one library per file, but a library may contain multiple sections (typically, one section per process corner).

### Definition

Inside netlist (reference library section)

### Sample Library

```
library corner_lib
section tt
    model nch bsim3v3 type=n mobmod=1 capmod=2 version=3.1
    + xj=1.7e-7 vsat=7.99e4 at=3.6e4 a0=0.799 ags=0.4
    + a1=0 a2=1 keta=-0.05 nch=2.8e17 ngate=1.31e20 k1=0.74
    model pch bsim3v3 type=p mobmod=1 capmod=2 version=3.1
    + xj=1.7e-7 vsat=1.38e5 at=1e5 a0=1.3 ags=0.3
    + a1=1.1e-4 a2=1 keta=0 nch=4.1e17 ngate=7.6e19 k1=0.88
    model knpn bjt is=10e-13 bf=170 va=58.7 ik=5.63e-3 rb=rbn rbm=86
    + re=3.2 cje=0.25e-12 pe=0.76 me=0.34 tf=249e-12 cjc=0.34e-12 pc=0.55
    + mc=0.35 ccs=2.4e-12 ms=0.35 ps=0.53 rc=169
    model kpnp bjt type=pnp is=10e-13 bf=60 va=43.1 ik=0.206e-3 rb=rbp rbm=64.3
    + re=33.8 cje=0.16e-12 pe=0.5 me=0.26 tf=36e-9 cjc=0.72e-12 pc=0.58
    + mc=0.34 ccs=2.5e-12 ps=0.53 ms=0.35 rc=276
```

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

```
endsection
section ss
    model nch bsim3v3 type=n mobmod=1 capmod=2 version=3.1
    + xj=1.7e-7 vsat=7.99e4 at=3.6e4 a0=0.799 ags=0.4
    + a1=0 a2=1 keta=-0.05 nch=2.8e17 ngate=1.31e20 k1=0.74
    model pch bsim3v3 type=p mobmod=1 capmod=2 version=3.1
    + xj=1.7e-7 vsat=1.38e5 at=1e5 a0=1.3 ags=0.3
    + a1=1.1e-4 a2=1 keta=0 nch=4.1e17 ngate=7.6e19 k1=0.88
    model knpn bjt is=10e-13 bf=70 va=58.7 ik=5.63e-3 rb=rbn rbm=86
    + re=3.2 cje=0.25e-12 pe=0.76 me=0.34 tf=249e-12 cjc=0.34e-12 pc=0.55
    + mc=0.35 ccs=2.4e-12 ms=0.35 ps=0.53 rc=169
    model kpnp bjt type=PNP is=10e-13 bf=30 va=43.1 ik=0.206e-3 rb=rbp rbm=64.3
    + re=33.8 cje=0.16e-12 pe=0.5 me=0.26 tf=36e-9 cjc=0.72e-12 pc=0.58
    + mc=0.34 ccs=2.5e-12 ps=0.53 ms=0.35 rc=276
endsection
section ff
    model nch bsim3v3 type=n mobmod=1 capmod=2 version=3.1
    + xj=1.7e-7 vsat=7.99e4 at=3.6e4 a0=0.799 ags=0.4
    + a1=0 a2=1 keta=-0.05 nch=2.8e17 ngate=1.31e20 k1=0.74
    model pch bsim3v3 type=p mobmod=1 capmod=2 version=3.1
    + xj=1.7e-7 vsat=1.38e5 at=1e5 a0=1.3 ags=0.3
    + a1=1.1e-4 a2=1 keta=0 nch=4.1e17 ngate=7.6e19 k1=0.88
    model knpn bjt is=10e-13 bf=220 va=58.7 ik=5.63e-3 rb=rbn rbm=86
    + re=3.2 cje=0.25e-12 pe=0.76 me=0.34 tf=249e-12 cjc=0.34e-12 pc=0.55
    + mc=0.35 ccs=2.4e-12 ms=0.35 ps=0.53 rc=169
    model kpnp bjt type=PNP is=10e-13 bf=90 va=43.1 ik=0.206e-3 rb=rbp rbm=64.3
    + re=33.8 cje=0.16e-12 pe=0.5 me=0.26 tf=36e-9 cjc=0.72e-12 pc=0.58
    + mc=0.34 ccs=2.5e-12 ps=0.53 ms=0.35 rc=276
endsection
endlibrary
```

## Library Multi-Technology Simulation Mode (Imts)

### Description

Activate TMI Library Multi-Technology Simulation mode. Under this mode, multiple TMI shared libraries are allowed to be simulated in the same netlist. Please make sure that all TMI shared libraries are included in their own sub-circuit in TMI MTS simulation.

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

Here is an example of a system consisting of blocks designed with different TMI shared libraries:

```
subckt subckt1 ( in out )
include "TMI_PDK_1.scs" section=sec1
.....
ends subckt1
subckt subckt2 ( in out )
include "TMI_PDK_2.scs" section=sec2
.....
ends subckt2
```

In the inventory of log file, you will see the following primitives:

Circuit inventory:

```
subckt1_tmibsim4      10      subckt=subckt1
subckt2_tmibsimcmg    10      subckt=subckt2
```

### Features Supported in Spectre

The TMI MTS flow has been constantly enhanced and qualified for different Spectre features and analyses. Refer to the following table to learn about the features supported in this flow.

This table will be updated every time a new feature is qualified in Spectre.

---

<b>Spectre version</b>	<b>Feature supported in the TMI MTS flow</b>
21.1ISR12	XF analysis Stb analysis
21.1ISR8	Severity switch for TMI MTS feature check
21.1ISR7	Checklimit analysis

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---



---

#### **Spectre version    Feature supported in the TMI MTS flow**

---

21.1ISR6	<p>Unsupported feature check to block unsupported features</p> <p>RF analysis (SP, PSS, Pnoise, HB, HBnoise, HBAC, HBSP)</p> <p>Command-line options:</p> <ul style="list-style-type: none"> <li>■ -help , -help name</li> <li>■ +log &lt;file&gt; , =log &lt;file&gt;</li> <li>■ -raw &lt;raw&gt;</li> <li>■ -format &lt;fmt&gt;</li> <li>■ -outdir &lt;path&gt;</li> <li>■ -V, -W</li> <li>■ -ldir</li> <li>■ maxwarns &lt;number&gt; -maxnotes &lt;number&gt;</li> <li>■ maxwarnstolog &lt;number&gt;</li> <li>■ maxnotestolog &lt;number&gt;</li> <li>■ lqtimeout</li> <li>■ +lmode &lt;value&gt;</li> <li>■ +lorder &lt;value&gt;</li> <li>■ -ahdllibdir path</li> <li>■ +postlayout[=hpa/upa/legacy/legacy_rf]</li> <li>■ +errpreset=liberal/moderate/conservative</li> <li>■ +liclog</li> <li>■ +config &lt;file&gt;</li> <li>■ +multithread</li> </ul>
----------	--



## Spectre Circuit Simulator Reference

### Other Simulation Topics

---



---

<b>Spectre version</b>	<b>Feature supported in the TMI MTS flow</b>
------------------------	--

---

	<ul style="list-style-type: none"> <li>■ +postpreset=value</li> <li>■ +preset=value</li> <li>■ -preset_override</li> <li>■ -env ade</li> <li>■ +logstatus</li> </ul>
20.1ISR14	ETMIS ETMIF
20.1ISR13	Noise analysis Assert/SOA APS (mode moderate/liberal/conservative) ++APS (mode moderate/liberal/conservative) Spectre X (mode AX, CX, MX)
20.1ISR7	alter/altergroup (only Spectre baseline)  Sweep (only Spectre baseline)  Monte Carlo (only Spectre baseline)
20.1ISR6	DC (only Spectre baseline)  AC (only Spectre baseline)  Transient (only Spectre baseline)

---

## Tips for Reducing Memory Usage (memory)

### Description

If you are facing an insufficient memory problem, try the following suggestions:

1. Try using a 64-bit executable if you are using the 32-bit one.
2. Try `ulimit/unlimit` command to adjust memory limitations.
3. Try another machine that has more memory, if hardware limit is the cause
4. Refer to `spectre -h rfmemory`, if you faced the problem during RF analyses.

## Multi-Technology Simulation Mode (mts)

### Description

Multi-Technology Simulation (MTS) mode enables the simulation of a system consisting of blocks designed with different processes. Under this mode, models and modelgroups referenced using `include` or `ahdl_include` statements in a subcircuit are locally scoped to that subcircuit only. In addition, selected process options parameters (`temp`, `tnom`, `scale`, `scalem`, `reltol`, `cmin`, and `postlpreset`), when specified in a subcircuit, are locally scoped to that subcircuit only.

MTS mode is enabled by default across all Spectre simulators. To turn the MTS mode off, use the `-mts` command-line option.

Here is an example of a system consisting of blocks designed with different processes:

```
subckt chip1 ( in out )
    scopedOptions options tnom=27 scale=0.1
    .....
ends chip1
subckt chip2 ( in out )
    scopedOptions options tnom=25 scale=0.2
    .....
ends chip2
```

In the log file, you will see the following user options:

```
Scoped user options:
    tnom = 27      subckt=chip1
    scale = 0.1    subckt=chip1
    tnom = 25      subckt=chip2
    scale = 0.2    subckt=chip2m
```

## Node Sets (nodeset)

### Description

The `nodeset` statement is used to provide an initial guess for nodes in DC analysis or to provide the initial condition calculation for transient analysis. The `nodeset` statement can occur multiple times in the input and the information provided in all the occurrences is collected. For more information, read the description of DC analysis.

### Definition

```
nodeset <X>[:param]=value
```

This statement takes a list of signals with the state information to the DC and transient analyses. `X` can be a node, a component, or a subcircuit, and `param` can be either a component output parameter or a terminal index. To specify a class of signals, use the pattern matching character `*` for any string and `?` for any character.

The concept of nodes for the statement has been generalized to signals, where a signal is a value associated with a topological node of the circuit or some other unknown that is solved by the simulator, such as the current through an inductor or the voltage of the internal node in a diode. Topological nodes can be either at the top-level or in a subcircuit.

For example:

```
nodeset 7=0 out=1 OpAmp1.comp=5 L1:1=1.0u
```

where, `7=0` implies that node 7 should be about 0V, node `out` should be about 1V, node `comp` in subcircuit `OpAmp1` should be about 5V, and the current through the first terminal of `L1` should be about 1uA.

For more information, refer to *[The ic and nodeset Statements](#)* in the *Spectre Classic Simulator*, *Spectre APS*, *Spectre X*, and *Spectre XPS User Guide*.

## Parameter Soft Limits (param\_limits)

### Description

The parameter values passed to Spectre components and analysis are subject to both hard and soft limits. If you set a parameter to a value that violates a hard limit, such as giving  $z0=0$  to a transmission line, Spectre issues an error message and quits. If the given parameter value violates a soft limit, a warning is issued, but Spectre uses the value of the component as given. Hard limits are used to prevent you from using values that would cause Spectre to fail or put a model in an invalid region. Soft limits are used to call attention to unusual parameter values that might have been given mistakenly. If a parameter value violates a soft limit, a message similar to one of the following sample messages is printed:

```
Parameter rb has the unusually small value of 1uOhms.
```

or

```
Parameter rb has the unusually large value of 1MOhms.
```

Spectre has built-in soft limits on a few parameter values. However, it is possible for you to override these limits or to provide limits on parameters that do not have built-in limits. To do so, create a parameter range limits file and run Spectre by providing the name of the file after the `+param` command-line option. For example:

```
spectre +param limits-file input-file
```

Limits are specified using the following syntax:

```
[PrimitiveName] [model] [LowerLimit <[=]] [[]Param[]] [<[=] UpperLimit]
```

The limits can be given as strict (using `<=`) or nonstrict (using `<`). If the limits are strict, there can be no space between `<` and `=`. The limits for one parameter are given on one line. There is no way of continuing the specification of the limits for a parameter over more than one line. If a parameter is specified more than once, the limits specified at last override the earlier limits. The primitive name must be a Spectre primitive name, and not a name used for SPICE compatibility. For example, `mos3` must be used instead of `mos`. Parameter limits can be written using Spectre native mode metric scale factors. Therefore, a limit of `f <= 1.0e6` can also be written as `f <= 1M`.

### Examples

```
mos3          0.5u <= 1 <= 100u
              0.5u <= w
              0 < as <= 1e-8
              0 < ad <= 1e-8
model |vto| <= 3
```

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

It is not necessary to specify the primitive name each time. If the primitive name is not specified, it is assumed to be the same as the previous parameter. Upper and lower limits may be specified, but if these are not specified, there is no limit on the parameter value. Therefore, in the example, if  $w$  is less than  $0.5\mu\text{m}$ , a warning is issued, but there is no limit on how large  $w$  can be. If a parameter is mentioned, but no limits are given, all limits are disabled for that parameter. Limits are placed on model parameters by giving the model keyword. If the model keyword is not given, the limits are applied to instance parameters. Notice that you can also place upper or lower limits on the absolute value of a parameter. For example:

```
resistor 0.1 < |r| < 1M
```

indicates that the absolute value of  $r$  should be greater than  $0.1\text{ Ohm}$  and less than  $1\text{ MOhm}$ . There can be no spaces between the absolute value symbols and the parameter name.

### Examples

```
1 <= x < 0.5
```

```
1 <= y <= 1
```

```
1 < z < 1
```

In the first case, the lower bound is larger than the upper bound, which indicates that the range of  $x$  is all real numbers, except those from  $0.5$  to  $1$ , including  $0.5$ . The limits are applied separately, therefore,  $x$  must be both greater than or equal to  $1$  ( $1 \leq x$ ) and less than  $0.5$  ( $x < 0.5$ ). The second case specifies that  $y$  should be  $1$ , and the third case specifies that  $z$  should not be  $1$ .

It is possible to specify limits for any scalar parameter that takes a real number, an integer, or an enumeration. To specify the limits of a parameter that takes enumerations, use the indexes associated with the enumerations. For example, consider the region parameter of the `bjt`. There are four possible regions: `off`, `fwd`, `rev`, and `sat` (see `spectre -help bjt`). Each enumeration is assigned a number starting at  $0$  and counting up. Therefore, `off`= $0$ , `fwd`= $1$ , `rev`= $2$ , and `sat`= $3$ . The specification `bjt 3 <= region <= 1` indicates that a warning should be printed if `region`=`rev` because the conditions ( $3 \leq \text{region}$ ) and ( $\text{region} \leq 1$ ) exclude only (`region`= $2$ ) and region  $2$  is `rev`.

It is possible to read a parameter limits file from within another file. To do so, use an `include` statement. For example,

```
include "filename"
```

temporarily suspends the reading of the current file until the contents of `filename` have been read. Include statements may be nested arbitrarily deep with the condition that the operating system may limit the number of files that Spectre may have open at once. Paths in file names are taken to be relative to the directory that contains the current file, and not from the directory from which Spectre was run.

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

Spectre can be instructed to always read a parameter limits file by using the SPECTRE\_DEFAULTS environment variable. For example, if you put the following in your shell initialization file (.profile for sh, .cshrc for csh)

```
setenv SPECTRE_DEFAULTS "+param /cds/etc/spectre/param.lmts"
```

Spectre always reads the specified limits file.

For more information, see *Selecting Limits for Parameter Value Warning Messages* in the *Spectre Classic Simulator, Spectre APS, Spectre X, and Spectre XPS User Guide*.

## Netlist Parameters (parameters)

### Description

The Spectre native netlist language allows parameters to be specified and referenced in the netlist, both at the top-level scope and within subcircuit declarations (run `spectre -h subckt` for more details on parameters within subcircuits).

### Definition

```
parameters <param=value> [param=value]...
```

### Examples:

```
simulator lang=spectre
parameters p1=1 p2=2           // declare some parameters
r1 (1 0) resistor r=p1         // use a parameter, value=1
r2 (1 0) resistor r=p1+p2      // use parameters in an expression, value=3
x1 s1 p4=8                     // "s1" is defined below, pass in value 8 for "p4"
subckt s1
parameters p1=4 p3=5 p4=6      // note: no "p2" here, p1 "redefined"
r1 (1 0) resistor r=p1         // local definition used: value=4
r2 (1 0) resistor r=p2         // inherit from parent(top-level) value=2
r3 (1 0) resistor r=p3         // use local definition, value=5
r4 (1 0) resistor r=p4         // use passed-in value, value=8
r5 (1 0) resistor r=p1+p2/p3   // use local+inherited/local = (4+2/5) = 4.4
ends
time_sweep tran start=0 stop=(p1+p2)*50e-6 // use 5*50e-6 = 150 us
dc_sweep dc param=p1 values=[0.5 1 +p2 (sqrt(p2*p2)) ] // sweep p1
```

### Parameter Declaration

Parameters can be declared anywhere in the top-level circuit description or on the first line of a subcircuit definition. Multiple parameters can be declared on a single line. When parameters are declared in the top-level, their values must be specified. When parameters are declared within subcircuits, their default values are optionally specified.

### Parameter Inheritance

Subcircuit definitions inherit parameters from their parent (enclosing subcircuit definition or top-level definition). This inheritance continues across all levels of nesting of subcircuit



definitions, that is, if a subcircuit `s1` is defined, which itself contains a nested subcircuit definition `s2`, then any parameters accessible within the scope of `s1` are also accessible from within `s2`. In addition, any parameters declared within the top-level circuit description are also accessible within both `s1` and `s2`. However, any subcircuit definition can redefine a parameter that it has inherited. In this case, if no value is specified for the redefined parameter when the subcircuit is instantiated, the redefined parameter uses the locally defined default value, rather than inheriting the actual parameter value from the parent.

### Parameter Referencing

Spectre netlist parameters can be referenced anywhere. A numeric value is normally specified on the right-hand side of an "=" sign or within a vector, where the vector itself is on the right-hand side of an "=" sign. This includes referencing of parameters in expressions (run `spectre -h expressions` for more details on netlist expression handling), as indicated in the preceding examples. You can use expressions containing parameter references when specifying device or analysis instance parameter values (for example, specifying the resistance of a resistor or the stop time of a transient analysis, as outlined in the preceding example), when specifying model parameter values in model cards (for example, specifying `bf=p1*0.8` for a bipolar model parameter `bf`), or when specifying initial conditions and nodesets for individual circuit nodes.

### Altering/Sweeping Parameters

Just as certain Spectre analyses (for example, `sweep`, `alter`, `ac`, `dc`, `noise`, `sp`, `xf`) can sweep device instance or model parameters, they can also sweep netlist parameters. Run `spectre -h <analysis>` to view the details for any of these analyses, where `<analysis>` is the analysis of interest.

### Temperature as a Parameter

You can use the reserved parameters `temp` and `tnom` anywhere an expression can be used, including within expressions and user-defined functions. The `temp` parameter always represents the simulator (circuit) temperature, and `tnom` always represents the measurement temperature. All expressions involving `temp` or `tnom` are re-evaluated everytime the circuit temperature or measurement temperature changes.

You can also alter or sweep the `temp` and `tnom` parameters by using any of the techniques available for altering or sweeping the netlist or subcircuit parameters (with the exception of `s`).

This capability allows you to write temperature dependent models, for example, by using `temp` in an equation for a model or an instance parameter. For example:

```
r1 1 0 res r=(temp-tnom)*15+10k // temp is temperature
```

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

```
o1 options temp=55          // causes a change in above resistor r1
```

#### Reserved Parameters

The following parameters are reserved and must not be declared as either top-level parameters or subcircuit parameters: temp, tnom, scale, scalem, freq, time.

For additional information, refer to the *Parameters Statement* section in the *Spectre Classic Simulator*, *Spectre APS*, *Spectre X*, and *Spectre XPS User Guide*.

## Parameter Set - Block of Data (paramset)

### Description

A parameter set is a block of data, which can be referenced by a sweep analysis. Within a paramset, the first row contains an array of top-level netlist parameters. All other rows contain numbers that are used to alter the value of the parameters during the sweep. Each row represents an iteration of the sweep. This data should be bound within braces. The opening brace is required at the end of the line defining the paramset. The paramset cannot be defined within subcircuits or cannot be nested.

### Definition

```
<Name> paramset {  
}
```

### Example:

```
data paramset {  
    p1  p2  p3  
    1.1 2.2 3.3  
    4.4 5.5 6.6  
}
```

## The postlayout command line option (postlayout)

### Description

This option enables various technologies to speed up post-layout simulation including RC reduction, efficient coupling capacitor handling, advanced DC algorithms optimized for large post-layout circuits, along with other advanced numerical techniques.

The `+postlayout` option delivers best simulation performance, with acceptable accuracy, for custom digital designs, memory designs, and analog/mixed-signal designs where high simulation precision is not required.

`+postlayout=hpa` provides higher simulation precision for analog /mixed-signal designs.

`+postlayout=upa` delivers the ultimate post-layout simulation accuracy, similar to `+postlayout=hpa`, but with RC reduction completely disabled.

Even though not recommended, `+rcnet_fmax=N` command-line option can be used to adjust the simulation accuracy associated with `+postlayout`. The unit for `N` is in GHz and the default value is 125. `N=25` is considered aggressive, whereas `N=250` is considered conservative. The `+rcnet_fmax` option not only controls RC reduction accuracy with any local RC net, but also defines the accuracy/performance trade-off of other non-reduction based techniques. `rcnet_fmax` is available as a Spectre command-line argument or option.

`+postlayout=legacy/legacy_rf` options provide backward compatibility with the `+parasitics/+parasitics=rf` options respectively. Accuracy of the `+postlayout=legacy` option can also be adjusted using the `rcr_fmax` option.

### Definition

`+postlayout` in the command line

## Pspice\_include File (pspice\_include)

### Description

Spectre supports PSPICE netlist format targeting to include PCB components that are modeled in PSPICE format. This solution does not support PSPICE designs. A top-level netlist and control statement needs to be defined in Spectre or SPICE format. The recommended approach is to define a subckt in PSPICE netlist format and to instantiate the subckt in a Spectre netlist.

A PSPICE netlist can be included in Spectre by using the following include statements.

```
pspice_include <file> (Spectre format)
.pspice_include <file> (SPICE format)
```

PSPICE file inclusion allows the circuit description to be spread over several files. The include statement itself is replaced by the contents of the file named. If the name given is not an absolute path specification, it is taken relative to the directory of the file currently being read.

The PSPICE file defined in the pspice\_include statement is required to contain only PSPICE netlist format. A PSPICE file may also contain include statements such as .inc or .lib to include other PSPICE files. The pspice\_include statement cannot be used in a PSPICE netlist.

When Spectre-simulator opens a pspice\_include file then it automatically switches to PSPICE mode. When in PSPICE mode, all elements and device models used in the PSPICE netlist are simulated using PSPICE default values and equations. Switching from PSPICE mode to Spectre mode inside a PSPICE file is not supported. After reading the include file, Spectre restores the language processing mode to what it was before the file was included, and continues reading the original file starting at the line after the include statement. Lines cannot be continued across file boundaries.

### Definition

```
Pspice_include "filename"
```

## Dynamic Release Voltage Node (release\_voltage)

### Description

The command is used to release the node voltage.

### Definition

Name `release_voltage` <parameter=value> ...

### Parameters

#### *Design parameters*

1	<code>node=[...]</code>	Nodes to which the check is applied.
2	<code>time=0 sec</code>	The start time of release.
3	<code>rforce=1 Ohm</code>	The resistance used to force voltage.
4	<code>t2z=0 sec</code>	The time from force to release voltage.

#### *Wildcard scoping*

5	<code>xnode=" [...]"</code>	Nodes to be excluded from the check. Default is none.
6	<code>inst=[...]</code>	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
7	<code>xinst=[...]</code>	Subcircuit instances to be excluded from the check. Default is none.
8	<code>subckt=[...]</code>	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits ( <code>subckt=*</code> ).
9	<code>xsubckt=[...]</code>	The instances of the specified subcircuits that are excluded from the check. Default is none.

**Spectre Circuit Simulator Reference**  
Other Simulation Topics

---

---

10	<code>depth=8</code>	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.
----	----------------------	--

---

## Tips for Reducing Memory Usage with SpectreRF (rfmemory)

### Description

**Problem:** How can you reduce memory usage when running SpectreRF simulations? What are the things that you need to be aware of?

**Solution:** The amount of swap/memory that Spectre/SpectreRF requires depends on the following:

- The analysis type you are running (PSS, QPSS, transient, dc, and so on). PSS and QPSS can take up a lot of memory.
- PSS and QPSS shooting can take up a lot of memory. If a reasonably small number of harmonics is needed for your circuit, then harmonic balance will require less memory than the shooting method, especially for systems with multiple input frequencies. More details are provided below.
- The amount of memory required is roughly proportional to the number of nodes in the circuit. Remember that most device models have multiple nodes inside the model.
- The number of nodes/nets you save has only a small effect on memory use and runtime. The number of nodes/nets you save does change the amount of disk space required to save the simulation result.

This help is broken into four sections; Shooting, Harmonic Balance, Small-Signal Analyses, and Envelope.

### ***PSS analysis***

The amount of memory required depends on the number of time points in the PSS solution. The number of time points in the solution depends on several factors.

- If you increase the number of harmonics above 10, the minimum number of time points is  $20 \times \text{number of harmonics}$ . Closely spaced input frequencies require a large number of harmonics to be calculated, and thus require large amount of memory.

For example, consider a system with 2.4GHz and 2.4GHz minus 1MHz, or 2.399GHz. To get to the second harmonic of 2.4 GHz requires 4,800 harmonics, which in turn requires 96,000 timepoints. This large number of timepoints would require a large amount of memory. For such systems, use harmonic balance (hb) instead.



- Increasing the accuracy by selecting conservative or by reducing the value of reltol and/or vabstol has the effect of producing more timepoints. A rough estimate is that for every power of 10 smaller you set reltol, five times as many timepoints will be produced.
- Setting a small maxstep compared to the PSS period or setting maxacfreq above  $40 \times \text{PSS beat frequency}$  will produce more timepoints. Unless you see warning messages in the small-signal analyses that instruct you to set maxacfreq, it is better to leave maxstep and maxacfreq unspecified and set conservative and/or a smaller values for reltol and vabstol. This will have the effect of taking more timesteps only in areas of transition, and longer timesteps where the waveform is static.
- The waveforms produced by the circuit can also affect the number of timepoints. If there are fast transitions in your circuit, the timestep will be reduced in the areas of the transitions to preserve high accuracy of the simulation.

### ***QPSS analysis***

- QPSS Analysis should be used only for nonlinear circuits like sampling circuits or switched-capacitor circuits. For other circuits, use hb instead.
- QPSS analysis runs a series of PSS analyses that depend on the number of moderate tones and the number of harmonics set in each moderate tone. Each PSS analysis is one cycle of the signal designated as Large in the QPSS Choosing Analyses Form with all of the input signals applied. The number of harmonics for the large tone and the accuracy settings mentioned above set the number of timepoints for each PSS analysis. The number of PSS analyses (called transient integrations in the Spectre output window) is  $(2 \times \text{number of harmonics on moderate tone 1}) + 1 \times (2 \times \text{number of harmonics on moderate tone 2}) + 1$  and continuing for more input tones. Assuming three harmonics on two moderate tones the number of PSS analyses is  $(2 \times 3) + 1 \times (2 \times 3) + 1$ , or 49 total PSS analyses.

### ***Harmonic Balance analysis***

- The amount of memory required depends primarily on the number of harmonics in the solution. Setting accuracy options will have only a small effect on the memory size.
- The number of harmonics in the solution depends on the number of input frequencies, the number of harmonics to be calculated for each frequency, and whether frequency cuts are used. The default number of harmonics is calculated as follows:  $(2 \times \text{number of harmonics on tone 1}) + 1 \times (2 \times \text{number of harmonics on tone 2}) + 1 \times (2 \times \text{number of harmonics on tone 3}) + 1$  and increasing if more inputs are present.
- For a single input and 7 harmonics,  $(2 \times 7) + 1$  or 15 harmonics are needed. If there are two inputs with 7 and 3 harmonics set, then  $15 \times 7$  or 105 harmonics are needed by default.

- Frequency cuts can reduce the number of harmonics that needs to be calculated. For more information on frequency cuts, see the SpectreRF User Guide, Chapter 3. If you want to use frequency cuts, the safest general recommendation is to use the funnel cut with maximum order set between 5 and 9. Use the smallest value of maximum order that produces accurate results.

### ***Small-Signal analysis***

Once the PSS, QPSS or HB analysis runs, additional memory is required to run the small-signal analyses. The amount of memory is difficult to predict because different analyses require different amounts of memory. As a rough estimate, the amount of memory might double or triple shortly after the small-signal analysis starts. For shooting method, setting maximum sidebands above about 50 will require more memory. For harmonic balance, the memory depends on the number of harmonics present in the harmonic balance large-signal analysis.

### ***Envelope method***

- Envelope method has two engines; shooting and harmonic balance. Harmonic balance is recommended because it always requires less memory and time to run, and it has a fast envelope level 1 analysis available.
- Transistor-level and Level 1 harmonic balance envelope is harmonic balance with a transient engine. The memory required is only slightly more than a single-tone hb analysis with the same number of harmonics that are used in envelope.
- For transistor-level envelope, a harmonic balance simulation is done at each time in the envelope analysis. For fast envelope, a series of harmonic balance analyses are run at the start to characterize the circuit, and then a behavioral simulation runs after that.

You can consider using the following strategies:

1. Check to see how much swap and memory you have on your computer. Greater than 2GB RAM and 5GB swap is recommended.

A 64-bit Spectre executable is available with MMSIM releases. This means that there is no 4GB process size limit, which was the case with the 32-bit Spectre executable. This allows much larger circuits to be simulated. If the memory required for simulation is larger than the amount of RAM installed on the machine, the machine will start swapping which dramatically slows the simulation and may cause the simulation to almost completely stop because the disk is so active. The best solution is more RAM on the system. A partial solution for shooting only is to use the swap file option described below. At least one large machine should be available for post-parasitic simulations to complete successfully without swapping.

2. If you are using shooting and your machine starts to swap, consider setting the swapfile option. This option writes much of the data from pss and qpss shooting large-signal analyses to disk files on your system. Large files will be produced. If the disk is local to the machine that is running the simulation, this will speed things up slightly compared to swapping in the operating system because Spectre manages the disk operations to put the data in concentric cylinders on the machine, but it is still fundamentally much slower than running on a machine that has more RAM. If the disk is a networked drive, the network latency may actually slow the simulation down. Only setting the option will allow runtime comparisons. Harmonic balance does not have this option.
3. For post-layout simulations, consider using RC reduction to reduce the number of nodes in the circuit.
4. Choose the largest possible `PSSfund` frequency. A higher fundamental frequency (`PSSfund`) yields a shorter simulation time.
5. If you have two or more closely spaced frequencies, use harmonic balance instead of PSS.
6. When shooting is used, try to use 10 harmonics or less and don't set the `macacfreq` or `maxstep` options unless you get errors from the small-signal analyses. Don't over-tighten the accuracy options.
7. In harmonic balance, use the smallest number of harmonics you can that produce accurate results, and consider using frequency cuts for circuits with multiple input frequencies.
8. If you are using names in harmonic balance or if you are using qpss, make sure that all the sources at the same frequency have the same Frequency Name 1 property.

**Note:** This is NOT the Instance Name property which all must be unique.

Setting the Frequency Name 1 property to the same name signifies to Spectre to treat those sources as a single frequency. For example, imagine you have differential RF inputs at 2.4GHz and I and Q differential LOs at 2.41GHz. There are two RF sources and four LO sources. Both RF sources have the same frequency and all four LO sources have the same frequency. As an example, you might set the RF sources to have the frequency name 1 property "RF" and all the LO sources "LO." In this case, Spectre will solve for two different frequencies as it should. If you named the sources RF1, RF2, LO1, LO2, LO3, and LO4, Spectre will consider these as all separate input frequencies. This becomes a six-tone simulation instead of a two-tone simulation and will take much longer to run and require hugely more memory.

9. In shooting small-signal analyses try to keep the number of sidebands to about 50 or less. More than 50 sidebands will require more memory. Full spectrum pnoise requires less memory than traditional pnoise. In hb small-signal analyses, leave the number of

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

sidebands field unspecified and use the smallest number of harmonics in the harmonic balance analysis that you can that also produce accurate results.

- 10.** In envelope analysis, use the smallest number of harmonics you can to get an accurate result.
- 11.** Run Spectre when no other users are using the same machine so all the RAM is available for your use.
- 12.** In harmonic balance, run the memory estimator on the circuit to have an understanding of the rough memory consumption.

## Output Selections (save)

### Description

The `save` statement indicates that the values of specific nodes or signals must be saved in the output file. It works in conjunction with the `save` parameter for most analyses. The output file is written in Cadence Parameter Storage Format (PSF), or in Nutmeg/SPICE3 format, which is controlled by a command-line argument or a global option (see the options statement). See options help for the complete list of supported formats. An appropriate postprocessor should be used to view the output, generate plots, or do any further processing.

### Definition

```
save X[:param] ... [depth=num] [sigtype=node|dev|subckt|all]
[devtype=component_type] [subckt=subckt_master]
[exclude=[wildcard_patterns_list]] [compression=no|yes] [ports=yes|no]
[filter=none|rc] [probelvl=num] time_window=[t1 t2 t3 t4 ...]
```

The `save` statement takes a list of signals followed by some optional parameters as an argument. `X` can be a node, a component, or a subcircuit and `param` can be a component output parameter or a terminal index. To specify a class of signals, use the pattern matching character `*` for any string and `?` for any character. The parameters control pattern matching. `depth` controls the depth of pattern matching and, by default, matches signals at all hierarchical levels. `sigtype` with the default value `node` defines the type of `X`. If `sigtype=all`, `X` can be a node, a component, or a subcircuit. `devtype` defines the component type and has no default value. `subckt` is used to save signals that are contained only in instances of a given subcircuit master. The signals matching a pattern from the list specified with `exclude` are not saved. When `subckt` is given, the wildcard patterns in the `save` statement and the depth of pattern matching must be relative to the subcircuit master.

The concept of nodes for the `save` statement is generalized to signals where a signal is a value associated with a topological node of the circuit or some other unknown that is solved by the simulator, such as the current through an inductor or the voltage of the internal node in a diode. Topological nodes can be at the top level or in the subcircuit.

By default, the waveforms defined in the `save` statement are not compressed. Compression can be enabled globally for selected hierarchy levels in the transient statement (`compression, complvl`). Compression can also be enabled or disabled individually for each `save` statement. The compression settings in the `save` statement overwrite the compression settings in the transient statement.

`time_window` is used to specify the output time interval for transient analysis. Signals in `[t1 t2]` and `[t3 t4] ...` are outputted.

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

The `ports` parameter saves the subckt ports names as well if `yes` is specified. The default value is `no`. The `filter` parameter works with wildcarding to filter out the nodes that connect only to parasitics, if `rc` is specified. The default value is `none`.

To probe terminal currents hierarchically at a specified depth, specify the depth using the `probelvl` option to obtain the terminal current for all terminals connected within that depth.

For example:

```
save 7 out OpAmp1.comp M1:currents D3:oppoint L1:1 R4:pwr
```

specifies that node 7, node `out`, node `comp` in subcircuit `OpAmp1`, the currents through the terminals of `M1`, the `oppoint` information for diode `D3`, the current through the first terminal of `L1`, and the instantaneous power dissipated by `R4` should be saved. These outputs are saved in addition to any outputs specified by the `save` parameter for the analysis.

To specify a component terminal current, specify the name of the component and the name or the index of the terminal separated by a colon. If `currents` is specified after the component and the colon, all the terminal currents for the component are saved unless the component has only two terminals, in which case only the current through the first terminal is saved. Current is positive if it enters the terminal flowing into the component.

If a component name is followed by a colon and `oppoint`, then the operating point information associated with the component is computed and saved. If the colon is followed by an operating point parameter name (see each component for list of operating point parameters), then the value of that parameter is output.

If only a component name is given, all available information about the component, including the terminal currents and the operating point parameter values, is saved.

### Examples of pattern matching

■ `save x*.*1 depth=3`

Saves the voltages of all nodes from level 2 to level 3 whose name starts with `x` and ends in 1. For example, `x1.n1`, `x1.x2.x3` but not `x1.x2.x3.x4`.

■ `save x*.*1 sigtype=subckt`

Saves all terminal currents of subcircuits from level 2 and above whose name starts with `x` and ends in 1. For example, `x1.x21:2`, `x1.x2.x31:3`.

■ `save *:c devtype=bjt`

Saves all collector currents

■ `save * subckt=inv`

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

Saves the voltages of all nodes in the instances of the subcircuit `inv`. For example, `X1.n1` for an instance `X1` of `inv` but not `net091` at the top-level

- `save * exclude=[X1* X2*]`

Saves the voltages of all nodes excluding the ones whose names start with `x1` or `x2`. For example, `net091`, `X0.res3.n2` but not `X21.res3.n2`.

- `save X1:A probelvl=3`

Saves all terminal currents that are connected to terminal `A` of subcircuit `X1`, till the third-level depth.

- `save nA:currents probelvl=2`

Saves the currents of all terminals that are connected to node `nA`, till the second-level depth.

- `save N1 time_window=[1m 2m 3m 4m]`

Save node `N1` from 1ms to 2ms and 3ms to 4ms.

- `save N1 time_window=[1m 2m]`

`save N1 time_window=[0.5m 2.5m 3m 4m]`

Save node `N1` from 0.5ms to 2.5ms and 3ms to 4ms

- `save N1 time_window = [ 1m 2m]`

Saves all waveforms starting with `N` for entire simulation time, except `N1` which is only saved from 1ms to 2ms.

For more information, see *[The Save Statement](#)* in the *Spectre Classic Simulator*, *Spectre APS*, *Spectre X*, and *Spectre XPS User Guide*.

## Savestate - Recover (savestate)

### Description

Savestate-Recover is a transient analysis feature. It is a replacement for Checkpoint-Restart capability.

The Checkpoint-Restart capability saves only the circuit solution for the timepoint at which the simulation is interrupted. Because there is no history information saved for the circuit, glitches, convergence issues, and inaccuracies can result when the simulation is resumed. The Savestate-Recover feature saves the complete state of the circuit, avoiding these issues.

Savestate-Recover provides the following functions:

- You have the option to save circuit information at set intervals or at multiple points during transient analysis. If the simulation halts unexpectedly, you can restart transient analysis from any saved timepoint.
- You can experiment with different accuracy settings over different transient time periods to obtain the optimal speed/accuracy trade-off.

### Requirements for Savestate-Recover

When the simulation is restarted:

- Netlist topology must not be changed. Topology changes or the removal/addition of nodes in the restore file causes a fatal error.
- You may edit any netlist parameter as long as the circuit topology remains the same.
- The stop time of Transient analysis must be larger than the timepoint corresponding to the savestate file.
- Saved state file is binary and platform dependent.
- Savestate-Recover works only for transient analysis and the transient analysis must not be within a Sweep or Monte Carlo analysis.

### Use Model

- Savestate

Savestate is enabled/disabled by using the `spectre` command, as follows.

```
spectre [+savestate ] [-savestate ] ...
```



## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

where:

- ❑ `+savestate` - Enables Savestate. You may use `+ss` as an abbreviation for `+savestate`.
- ❑ `-savestate` - Disables Savestate. You may use `-ss` as an abbreviation for `-savestate`.

By default, `savestate` is on, and `checkpoint` is off.

- Define saved time points and state file in the `tran` statement. For example:

```
DoTran tran stop=stoptime [ [saveperiod=time] | [saveclock=clock_time] |  
[savetime=[time1 time2...]] [savefile=file.srf]
```

where:

- ❑ `saveperiod`, `saveclock`, and `savetime` define the time points to save the states.
  - If `saveperiod` is given, Spectre generates a saved state file periodically based on the transient simulation time. Only the last saved state file is kept.
  - If `saveclock` is given, Spectre generates a saved state file periodically based on real time (wall clock time). Its default value is 1800 seconds (30 minutes).
  - If `savetime` is given, Spectre generates a saved state file on each specified time point.
- ❑ `savefile` defines where the saved states are written.
  - If `savefile` is not defined, the default file name is `%C.%A.srf`.

Where `%C` is the input circuit file name, and `%A` is the analysis name.

If multiple save time points are given, That is,

```
analysisName tran stop=stoptime savetime=[time1 time2 ...] savefile=filename
```

the saved state file is `filename_at_time1`, `filename_at_time2`, and so on.

Besides saving the state based on `saveperiod` or `saveclock` or `savetime`, if `savestate` is enabled, Spectre automatically saves the states to a file when an interrupt signal like `QUIT`, `TERM`, `INT`, or `HUP` is received for the first time. If interrupt signals are received more than once, Spectre quits immediately.

The `saveclock`, `saveperiod`, and `savetime` parameters should not be specified at the same time.

If more than one parameter is specified, Spectre reads them in the following order:

```
saveperiod
```

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

```
savetime  
saveclock
```

#### ■ Recover

There are two ways to recover the simulation from the saved state file. The first is to define the recover file using the `spectre` command. The second is to define the recover file in a `tran` statement.

Defining the recover file in the `tran` statement is strongly recommended, especially if there are multiple analyses statements in the netlist.

#### ■ Recover from command line

```
spectre [+recover[=filename]] [-recover] ...
```

where:

- ❑ `+recover` - Enables recover. You may use `+rec` as an abbreviation for `+recover`.
- ❑ `-recover` - Disables recover. You may use `-rec` as an abbreviation for `-recover`.

#### ■ Recover from the `tran` statement

```
analysisName tran recover=filename ...
```

By default, recover is disabled.

#### ■ Output Directory on Recovering

When recovering from a saved state in a Spectre run by using `+recover=state_file` in a command-line option, a new raw directory is created to avoid overwriting the previous simulation results. However, if `recover=state_file` is given in a `tran` statement, the default raw directory is used.

When defining `recover=state_file` in a `tran` statement, use a different tran name to avoid previous simulation results to be overwritten by the recovered results.

When a new raw directory is created, the raw directory name is the same as the default raw directory, except that an index (starting from 0) is suffixed to raw, such as `*.raw#`, where # is 0, 1, 2, and so on.

For example, in the first run, enter:

```
spectre input.scs
```

Spectre saves the simulation state in a file on a time point. By default, `input.raw` directory is created.

When Spectre runs in recover mode:

```
spectre +recover=saved_state_file input.scs
```

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

a new raw directory named `input.raw0` is created. The index of the raw directory is increased by one at each successive recover run. Another use model is that you define multiple transient-analysis runs in a netlist. In the first transient-analysis run, Spectre saves the simulation state on a time point.

In the next transient analysis run, simulation is continued from the saved time point. For example:

```
tran1 tran step=1ps stop=200ns savetime=[50ns] savefile=tran1_save
tran2 tran step=1p stop=400ns recover=tran1_save_at_50.00ns
```

In this case, the default raw directory is used.

For more information, see [Recovering From Transient Analysis Terminations](#) section in the *Spectre Classic Simulator, Spectre APS, Spectre X, and Spectre XPS User Guide*.

## Sensitivity Analyses (sens)

### Description

Use the `sens` control statement to find partial or normalized sensitivities of the output variables with respect to component and instance parameters for the list of the analyses performed. Currently, DC and AC sensitivity analyses are supported. The results of the sensitivity analyses are stored in the output files written in Cadence Parameter Storage Format (PSF). The global option parameter `sensstype` (see the options statement) is used to control the type of sensitivity being calculated. In addition, you can use `+sensdata filename` command-line argument or a global option (see the options statement) to direct sensitivity analyses results into a specified ASCII file.

### Definition

```
sens (output_variables_list) to (design_parameters_list) for (analyses_list)
```

where:

- `output_variables_list` = `ovar1 ovar2, and so on.`
- `design_parameters_list` = `dpar1, dpar2, and so on.`
- `analyses_list` = `anal1 anal2 , and so on.`

The list of design parameters may include valid instance and model parameters. You can also specify device instances or device models without a modifier. In this case, Spectre attempts to compute sensitivities with respect to all corresponding instance or model parameters. Caution should be exercised in using this option as warnings or errors may be generated if many instance and model parameters cannot be modified. If no design parameters are specified, then all the instance and model parameters are added. The list of the output variables for both AC and DC analyses may include node voltages and branch currents. For DC analyses, it may also include device instance operating point parameters.

### Examples

- `sens (q1:betadc 2 Out) to (vcc:dc nbjt1:rb) for (analDC)`

For this statement, DC sensitivities of `betadc` operating point parameter of transistor `q1` and of nodes `2` and `Out` are computed with respect to `dc` voltage level of voltage source `vcc` and model parameter `rb` for the DC analysis `analDC`. The results are stored in the raw directory with the name in the format `analysisname.sens.analysisstype`, that is, `dc.sens.analDC`.

- `sens (1 n2 7) to (q1:area nbjt1:rb) for (analAC)`

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

For this statement, AC sensitivities of nodes 1, n2, and 7 are computed with respect to the area parameter of transistor q1 and the model parameter `rb` for each frequency of the AC analysis `analAC`. The results are stored in the output file `analAC.sens.ac`.

- `sens (1 n2 7) for (analAC)`

For this statement, AC sensitivities of nodes 1, n2, and 7 are computed with respect to all instance and model parameters of all devices in the design for each frequency of the AC analysis `analAC`. The results are stored in the file in the format `ac.sens.analAC`.

- `sens (vbb:p q1:int_c q1:gm 7) to (q1:area nbjt1:rb) for (analDC1)`

For this statement, DC sensitivities of branch current `vbb:p`, the operating point parameter `gm` of the transistor `q1`, the internal collector voltage `q1:int_c` and the node 7 voltage are computed with respect to instance parameter `area` for instance `q1` and model parameter `rb` for model `nbt1`.

For additional information, see [\*Sensitivity Analysis\*](#) in the *Spectre Classic Simulator*, *Spectre APS*, *Spectre X*, and *Spectre XPS User Guide*.

## Options for Sparam Standalone Checking and Fitting Tool (sparam)

### Description

- `+checking` - Checking quality of original S-parameter data including:
  - ❑ DC point: Checking if DC points are provided, and also DC imaginary is zero.
  - ❑ Fmax: Reporting maximum freq in the data.
  - ❑ Fmin: Reporting minimum freq in the data after DC point.
  - ❑ Reciprocity: Reporting if S-parameter matrix is symmetric.
  - ❑ Discontinuity in Real/Imaginary: Reporting S-parameter elements with discontinuity in either real or imaginary part.
  - ❑ Discontinuity in Phase: Reporting S-parameter elements with discontinuity in phase.
  - ❑ Data rotation in Polar coordinate system: Reporting S-parameter elements with counter-clockwise rotations and possible causality issue.
  - ❑ Passivity: Checking S-parameter data passivity. Absolute tolerance of passivity criteria is 1e-6.
- `+fitting [=rfm]`
  - ❑ `'rfm'`: only generates rational fitted model.
- `+multithread=<N>` - Enables the multithreading capability, where, N is the number of threads specified. A maximum of 64 threads are allowed.
  - ❑ `'+mt'` can be used as an abbreviation of `'+multithread'`

## SpectreRF Summary (spectrerf)

### Description

SpectreRF is an optional collection of analyses that are useful for circuits that are driven with a large periodic signal. Examples include mixers, oscillators, switched-capacitor filters, sample-and-holds, chopper stabilized amplifiers, frequency multipliers, frequency dividers, and samplers. They efficiently and directly compute the periodic and quasiperiodic steady-state solution of such circuits and are capable of computing large-and-small-signal behavior, including noise behavior. Therefore, SpectreRF is capable of computing the noise figure or intermodulation distortion of a mixer, the phase noise and harmonic distortion of an oscillator, and the frequency-response and noise behavior of a switched-capacitor filter. For more information about the SpectreRF analyses, run `spectre -help analysisName` where `analysisName` is `pss`, `pac`, `pxf`, `pnoise`, `psp`, `pstb`, `qpss`, `qpac`, `qpxf`, `qpnoise`, `qpssp`, `envlp`, `hb`, `hbac`, `hbsp`, or `hbnoise`.

## Stitch Flow Use Model (stitch)

### Description

Stitching enables Spectre APS to plug in the parasitic elements while simulation is running. Compared to the flat RC netlist and the hierarchical RC netlist approaches, the Stitching flow has the following advantages:

- Reuse of the pre-layout simulation test-bench. There is no need to change the probe and the measure statements.
- What-if analysis powered by selective stitching.

Spectre APS stitching is enabled by options.

### Parasitic File Loading Parameters

- `spf`

This option specifies the to-be-stitched DSPF file and its stitching scope. The syntax is `spf="scope filename"`. The scope can be a subcircuit or an instance. When a subcircuit is specified as the scope, the DSPF file is stitched to all the instances of that subcircuit. When an instance is specified as the scope, the DSPF file is stitched to that instance only. Multiple DSPF files can be specified for stitching by using the option multiple times.

Example:

```
spf="mem mem.dspf"
```

- `dpf`

This option specifies the to-be-stitched DPF file and its stitching scope. The syntax is `dpf="scope filename"`. The scope can be a subcircuit or an instance. When a subcircuit is specified as the scope, the DPF file is stitched to all the instances of that subcircuit. When an instance is specified as the scope, the DPF file is stitched to that instance only. Multiple DPF files can be specified for stitching by using the option multiple times.

Example:

```
dpf="X1.XPLL PLL.dpf" dpf="X1.XMEM mem.dpf"
```

This means that the `PLL.dpf` file needs to be stitched to the `X1.XPLL` instance and the `mem.dpf` file needs to be stitched to the `X1.XMEM` instance.

- `spef`



This option specifies the to-be-stitched SPEF file and its stitching scope. The syntax is `spef="scope filename"`. The scope can be a subcircuit or an instance. When a subcircuit is specified as the scope, the SPEF file is stitched to all the instances of that subcircuit. When an instance is specified as the scope, the SPEF file is stitched to that instance only. Multiple SPEF files can be specified for stitching by using the option multiple times.

Example:

```
spef="adc a.spef"
```

### Stitching Parsing Options

#### ■ `spfscale`

This option specifies the scaling factor of all the elements in the parasitic files (DSPF/SPEF/DPF). For example, consider that the width of a device in the DSPF file is  $w=2$ . If `spfscale=1.0e-6` is used, the actual width will be  $w=2.0e-6$ .

#### ■ `spfscalec`

This option specifies the scaling factor of all the capacitors in the parasitic files (DSPF/SPEF/DPF). Example, suppose a capacitor's value in the DSPF file is 2. If `spfscalec=1.0e-15` is used, the actual capacitor value will be  $2.0e-15$ .

#### ■ `spfscler`

This option specifies the scaling factor of all the resistors in the parasitic files (DSPF/SPEF/DPF). Example, suppose a resistor's value in the DSPF file is 2. If `spfscler=1.0e-3` is used, the actual capacitor value will be  $w=2.0e-3$ .

#### ■ `spfswapterm`

This option specifies the swappable terminals of a subcircuit macro-model. The syntax is `spfswapterm="terminal1 terminal2 subcktname"`.

Example:

```
spfswapterm="n1 n2 nch_mac"
```

This indicates that terminals `n1` and `n2` of subckt `nch_mac` are swappable. In general, this is applicable to devices that are modeled by subcircuits. Multiple `spfswapterm` statements are supported.

#### ■ `spfxtorprefix`

This option specifies the prefix in the names (devices and nets) in the DSPF/SPEF/DPF file. The device names in the prelayout netlist and the DSPF/SPEF file often do not

match. The `spfxtorprefix` option can be used to help match the device names. The syntax is `spfxtorprefix="<substring> [<replace_substring>]"`.

Example:

```
spfxtorprefix="XM X"
```

`XX1/XM1` exists in the prelayout netlist but the corresponding device name in the DSPF file is `XM1/XM1`. This option will change `XM` to `X`.

#### ■ `spfaliasterm`

Sometimes the terminal names of devices in DSPF/SPEF/DPF files are different from those in the simulation model library. This happens often in the technology nodes that uses subcircuits to model devices. The syntax is `spfaliasterm="<model|subckt> <prelayout_term1>=<spf_alias1> <prelayout_term2>=<spf_alias2>... <prelayout_termN>=<spf_aliasN>"`. Multiple statements are supported.

Example:

```
spfaliasterm="nfet_mac n1=D n2=G n3=S n4=B"
```

This means that in subckt `nfet_mac`, terminal `n1` corresponds to terminal `D` in the DSPF file, `n1` corresponds to terminal `G`, `n3` corresponds to terminal `S` and `n4` corresponds to terminal `B`.

#### ■ `speftriplet`

This option specifies the value that should be used for stitching in the SPEF file. This is effective only when the values in the SPEF file are represented by triplets (for instance, `0.325:0.41:0.495`). Default value is 2. Possible values are 1, 2 and 3.

#### ■ `spfinstancesection`

This option controls the backannotation of device parameters in the instance section of the DSPF file. If `spfinstancesection` is turned off, the instance section is ignored (that is, the device parameters are not changed during stitching). Default is on.

#### ■ `spfbusdelim = busdelim_schematic [busdelim_parasitic]`

This option maps the bus delimiter between schematic netlist and parasitic file (i.e. DSPF, SPEF, or DPF). The option defines the bus delimiter in the schematic netlist, and optionally the bus delimiter in the parasitic file. By default, the bus delimiter of the parasitic file is taken from the parasitic file header (i.e. `*|BUSBIT []`, `*|BUS_BIT []`, or `*|BUS_DELIMITER []`). If the bus delimiter is not defined in the parasitic file header, you need to specify it by using the `spfbusdelim` option in schematic netlist.

Example:

- ❑ `spfbusdelim=<>` - A<1> in the schematic netlist is mapped to A\_1 in the DSPF file, if the bus delimiter header in the DSPF file is "\_".
- ❑ `spfbusdelim=@ []` - A@1 in the schematic netlist is mapped to A[1] in the DSPF file (the bus delimiter in DSPF header will be ignored).

■ **spfinstarraydelim**

This option specifies the supplemental bus delimiter, if more than one bus delimiter is used in the pre-layout netlist (usually for instance array indexing). This option is similar to `spfbusdelim`.

## Selective Stitching Options

■ **spfcnet**

This option specifies the net that has its total capacitance stitched. All other parasitic components, say parasitic resistors, associated with this net are ignored. The full hierarchical names are required. Multiple statements are supported. Wildcards are supported.

Example:

```
spfcnet=X1.netA
```

■ **spfcnetfile**

This option has the same functionality as `spfcnet`. However, it accepts a text file in which all the C-only stitched nets are listed. Only one file can be specified. The syntax is `spfcnetfile="filename"`. The format in the file is one line per net.

Example:

```
spfcnetfile="nets.tex"
```

The format in the `nets.tex` is:

```
netA
netB
netC
```

■ **spfrcnet**

This option specifies the name of the net to be stitched with parasitic resistors and capacitors. The other nets are stitched with lumped total capacitances. Multiple statements are supported. Wildcards are supported and you can specify multiple nets. Full hierarchical names are required.

Example:

```
spfrcnet=netA
```

#### ■ `spfrcnetfile`

This option has the same functionality as `spfrcnet`. However, it accepts a text file in which all the RC stitched nets are specified. Only one file can be specified. The syntax is `spfrcnetfile="filename"`. The format in the file is one line per net.

Example:

```
spfrcnetfile="nets.tex"
```

The format in the `nets.tex` is:

```
netA
netB
netC
```

#### ■ `spfnetcmin`

This option allows you to select the net for stitching by the value of its total node capacitance. If a net's total node capacitance exceeds `spfnetcmin`, all parasitics associated with the net are stitched correctly; otherwise, only the total capacitance is added to the net node.

Example:

```
spfnetcmin=1.0e-6
```

#### ■ `spfskipnet`

This option specifies the net to be skipped for stitching, that is, all parasitic components of the net are not stitched. Wildcards and multiple statements are supported.

Example:

```
spfskipnet=X1.nodeA
```

#### ■ `spfskipnetfile`

This option allows you to specify the nets to be skipped as a list in the text file called `file_name`. The syntax is `spfskipnetfile="filename"`. Only one file can be specified. The format in the file is one line per net.

Example:

```
spfskipnetfile="nets.tex"
```

`nets.tex` file format is:

```
netA
netB
netC
```

## Stitching Message Control Option

### ■ `spfmsglimit`

This option specifies the maximum number of messages to be printed in the `spfrpt` file. The messages in the `spfrpt` file are categorized by their ID number (STITCH-ID). This option specifies the maximum number of messages for a particular type of messages by using their STITCH-ID. The syntax is `spfmsglimit="number STITCH-ID_1 STITCH-ID_2"`. When STITCH-ID is not specified, the tool assigns the maximum message number limit to all messages categories (STITCH-IDs).

Example:

```
spfmsglimit="10 STITCH-0010"
```

This tells the tool to print not more than 10 messages for the STITCH-0010 message category, in the meantime, for the other message categories, the default maximum limit of 50 messages will apply.

For additional information, see *[Parasitic Backannotation of DPSF/SPEF/DPF Files](#)* in the *Spectre Classic Simulator, Spectre APS, Spectre X, and Spectre XPS User Guide*.

## Subcircuit Definitions (subckt)

### Description

#### *Hierarchical Circuit*

The `subckt` statement is used to define a subcircuit. Subcircuit definitions are circuit macros that can be expanded anywhere in the circuit any number of times. When an instance in your input file refers to a subcircuit definition, the instances specified within the subcircuit are inserted into the circuit. Subcircuits may be nested. Therefore, a subcircuit definition may contain instances of other subcircuits. Subcircuits may also contain component, analysis, or model statements. Subcircuit definitions can also be nested, in which case the innermost subcircuit definition can only be referenced from within the subcircuit in which it is defined, and cannot be referenced from elsewhere.

Instances that instantiate a subcircuit definition are referred to as subcircuit calls. The node names (or numbers) specified in the subcircuit call are substituted, in order, for the node names given in the subcircuit definition. All instances that refer to a subcircuit definition must have the same number of nodes as are specified in the subcircuit definition and in the same order. Node names inside the subcircuit definition are strictly local unless declared otherwise in the input file with a global statement.

#### *Subcircuit Parameters*

Parameter specification in subcircuit definitions is optional. In the case of nested subcircuit definitions, parameters that have been declared for the outer subcircuit definition are also available within the inner subcircuit definition. Parameters that are specified are referred to by name, optionally followed by an = sign and a default value. If, when making a subcircuit call, you do not specify a particular parameter, this default value is used in the macro expansion. Subcircuit parameters can be used in expressions within the subcircuit consisting of subcircuit parameters, constants, and various mathematical operators. Run `spectre -h expressions` for details about Spectre expression handling capability. Run `spectre -h parameters` for details about how Spectre handles netlist parameters, including subcircuit parameters, and how they inherit within nested subcircuit definitions.

Subcircuits always have an implicitly defined parameter `m`. This parameter is passed to all components in the subcircuit, and each component is expected to multiply it by its own multiplicity factor. In this way, it is possible to efficiently model several copies of the subcircuit in parallel. Any attempt to explicitly define `m` on a `parameters` line results in an error. In addition, because `m` is only implicitly defined, it is not available for use in expressions in the subcircuit.

#### ***Inline Subcircuits***

An inline subckt is a special case of a subckt where one of the devices or models instantiated within this subckt does not get its full hierarchical name. It instead inherits the subckt call name. An inline subckt is syntactically denoted by the presence of the keyword `inline` before the `subckt`. It is called in the same manner as a regular subcircuit. The body of the inline subcircuit can typically contain one of the following, based on different use models:

- Multiple device instances, one of which is the `inline` component
- Multiple device instances, one of which is `inline` and one or more parameterized models
- A single `inline` device instance and a parameterized model to which the device instance refers

The `inline` component is denoted by giving it the same name as the inline subcircuit. When the subcircuit is flattened, the `inline` component does not take on a hierarchical name such as `X1.M1`, it instead takes on the name of the subckt call, such as `X1`. Any non-inline components in the subckt take on the regular hierarchical name, as if the subcircuit were a regular subckt (that is, not an `inline` subckt).

#### ***Probing the inline device***

Spectre allows the following list of items to be saved or probed for primitive devices. These would also apply to devices modeled as the inline components of inline subcircuits:

- All terminal currents. For example, save `q1:currents`
- Specific (index) terminal current. For example, save `q1:1` (`#1=collector`)
- Specific (named) terminal current. For example, save `q1:b` (`"b"=base`)
- Save all operating point info. For example, save `q1:oppoint`
- Save specific operating point info. For example, save `q1:vbe`
- Save all currents and oppoint info. For example, save `q1`

#### ***Parameterized Models and Inline Subckts***

Inline subckts can be used in the same way as regular subcircuits to implement parameterized models, however, inline subckts provide some powerful new options. When an inline subcircuit contains both a parameterized model and an inline device that references that model, you can create instances of the device, and each device automatically gets an appropriately scaled model assigned to it. For example, the instance parameters to an inline

subckt could represent something like emitter width and length of a BJT device, and within the subckt, a model card that is parameterized for emitter width and length and scales can be created accordingly. When you instantiate the macro, supply the values for the emitter width and length, and a device is instantiated with an appropriate geometrically scaled model. Again, the inline device does not get a hierarchical name and can be probed in the same manner as the inline device in the previous section on modeling parasitics, that is, the inline device can be probed just as if it were a simple device, and not actually embedded in a subckt

### **Automatic Model Selection using Inline Subckts**

See `spectre -h if` for a description on how to combine Spectres `structural if` statement with inline subckts to perform automatic model selection based on any netlist/subckt parameter.

For additional information on subcircuits, see *Subcircuits* in the *Spectre Classic Simulator, Spectre APS, Spectre X, and Spectre XPS User Guide*.

### **Definition**

```
[inline] subckt <Name> (<node1> ... <nodeN>)
```

### **Example 1: subckt**

```
subckt coax (i1 o1 i2 o2)
    parameters zin=50 zout=50 vin=1 vout=1 len=0
    inner i1 o1 i2 o2  tline z0=zin vel=vin len=len
    outer o1 0  o2 0  tline z0=zout vel=vout len=len
ends coax
```

Defines a parameterized coaxial transmission line macro from two ideal transmission lines. To instantiate this subcircuit, one could use an instance statement such as:

```
Coax1 pin nin out gnd coax zin=75 zout=150 len=35m
```

### **Example 2: inline subckt - parasitics**

Consider the following example of an inline subcircuit, which contains a mosfet instance, and two parasitic capacitances:

```
inline subckt s1 (a b)                // "s1" is name of subckt
    parameters p1=1u p2=2u
    s1 (a b 0 0) mos_mod l=p1 w=p2    // "s1" is "inline" component
    cap1 (a 0) capacitor c=1n
    cap2 (b 0) capacitor c=1n
ends s1
```



The following circuit creates a simple mos device instance M1, and calls the inline subcircuit s1 twice (M2 and M3)

```
M1 (2 1 0 0) mos_mod
M2 (5 6) s1 p1=6u p2=7u
M3 (6 7) s1
```

This expands/flattens to:

```
M1 (2 1 0 0) mos_mod
M2 (5 6 0 0) mos_mod l=6u w=7u // the "inline" component, inherits call name
M2.cap1 (5 0) capacitor c=1n // a regular hierarchical name
M2.cap2 (6 0) capacitor c=1n
M3 (6 7 0 0) mos_mod l=1u w=2u // the "inline" component, inherits call name
M3.cap1 (6 0) capacitor c=1n
M3.cap2 (7 0) capacitor c=1n
```

Here, the final flattened names of the three mosfets (one for each instance) are M1, M2, and M3, rather than M1, M2.s1, and M3.s1 as they would be if s1 was a regular subcircuit. However, the parasitic capacitors (which you may not be interested in, or perhaps, even aware of, if the inline subckt definition was written by a different modeling engineer) have full hierarchical names.

### Example 3: inline subckt - scaled models

Consider the following example, in which a parameterized model is declared within an inline subcircuit for a bipolar transistor. The model parameters are emitter width, length, and area, and also the temperature delta (trise) of the device above nominal. Ninety-nine instances of a 4\*4 transistor are then placed and one instance of a transistor with area=50 is placed. Each transistor gets an appropriately scaled model.

Declare a subckt, which instantiates a transistor with a parameterized model. The parameters are emitter width and length.

```
inline subckt bjtmod (c b e s)
parameters le=1u we=2u area=le*we trise=0
model mod1 bjt type=npn bf=100+(le+we)/2*(area/le-12)
+      is=1e-12*(le/we)*(area/le-12)
bjtmod (c b e s) mod1 trise=trise // "inline" component
ends bjtmod
```

some instances of this subckt

```
q1 (2 3 1 0) bjtmod le=4u we=4u // trise defaults to zero
q2 (2 3 2 0) bjtmod le=4u we=4u trise=2
q3 (2 3 3 0) bjtmod le=4u we=4u
.....
```

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

.....

q99 (2 3 99 0) bjtmod le=4u we=4u

q100 (2 3 100 0) bjtmod le=1u area=50e-12

## Vec/Vcd/Evcd Digital Stimulus (vector)

### Description

Spectre supports Digital Vector (VEC), Verilog-Value Change Dump (VCD), and Extended Verilog-Value Change Dump (EVCD).

### VEC

To process digital vector files, the following command card needs to be specified in the netlist.

In Spice netlist:

```
.vec "vector_filename" [HLCheck = 0|1]
```

or

```
.vec vector_filename [HLCheck = 0|1]
```

Quotation marks can be double or single in Spice Netlist.

In Spectre netlist,

```
vec_include "vector_filename" [HLCheck = 0|1]
```

where,

`vector_filename` is the filename of the digital vector file.

`HLCheck = 0 | 1` is a special flag (default = off) to create the vector output check for the H and L states of input signals. Bidirectional and output signals always check H and L states and are unaffected by the HLCheck flag.

Normally, you do not need to use the HLCheck flag.

Each command card specifies only one VEC file. If a netlist needs to include multiple VEC files, multiple `.vec/vec_include` cards must be used. For example, if a netlist contains three VEC files, it needs three `.vec` cards, as given below:

```
.vec "file1.vec"  
.vec "file2.vec"  
.vec "file3.vec"
```

## VCD/EVCD

VCD and EVCD formats are widely used in digital circuit design and contain different kinds of information for transistor-level simulation. You need to provide signal information, such as timing characteristics, voltage threshold, and driving ability of input signals, for each VCD or EVCD file.

Because VCD and EVCD formats are compatible, the same signal information file can be shared between them.

The VCD file (ASCII format) contains information about value changes for selected variables in the circuit design. Spectre simulator supports two types of VCD files:

`Four states` - represents variable changes in 0, 1, x (unknown or not needed), and z (tri-state) without providing strength information and port direction.

`Extended` - represents variable changes in all states and provides strength information and port direction.

To process the VCD/EVCD file in Spectre, the following command card needs to be specified in the netlist.

In Spice netlist:

```
.vcd "vcd_filename" "signal_info_filename"
.evcd "evcd_filename" "signal_info_filename"
```

In Spectre netlist:

```
vcd_include "vcd_filename" "signal_info_filename"
evcd_include "evcd_filename" "signal_info_filename"
```

Each command card specifies only one VCD file. If a netlist needs to include multiple VCD files, multiple `.vcd/vcd_include` cards must be used. For example, if a netlist contains three VCD files, it needs three `.vcd` cards, as given below:

```
.vcd "file1.vcd" "file1.signal"
.vcd "file2.vcd" "file2.signal"
.vcd "file3.vcd" "file3.signal"
```

## Output Check

For VEC, VCD, and EVCD output check, the results are written in two files under the raw directory, one a check error report file `%A.vecerr` and the other a check summary report file `%A.veclog` where, `%A` is the analysis name.

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

To find VEC VCD and EVCD file, `signal_info` file format description and examples, refer to the *Digital Vector File Format* chapter in the *Spectre Classic Simulator*, *Spectre APS*, *Spectre X*, and *Spectre XPS User Guide* for details.

## Verilog-A Usage and Language Summary (veriloga)

### Description

Verilog-A is an analog hardware description language standard from Open Verilog International. It enables analog circuit behavior to be described at a high level of abstraction. Behavioral descriptions of modules and components may be instantiated in a Spectre netlist along with regular Spectre primitives.

Verilog-A descriptions are written in files different from the Spectre netlist file. These descriptions are written in modules (see the module `alpha` below). To include a module in the Spectre netlist, first add the line `ahdl_include "VerilogAfile.va` to the Spectre netlist file (where, `VerilogAfile.va` is the name of the file in which the required module is defined). The module is instantiated in the Spectre netlist in the same manner as Spectre primitives, for example:

```
name (node1 node2) alpha arg1=4.0 arg2=2
```

This instantiates an element `alpha`, which has two nodes and two parameters.

AHDL Linter can be used to improve Verilog-A model quality. It can help to avoid potential convergence or performance problems, and to improve model accuracy, reusability and portability. Refer to the *Verilog-A Language Reference manual* for more information.

Verilog-A simulation performance has been improved by compiling the Verilog-A modules. This is explained in more detail in the Verilog-A compilation section below.

### Module Template

The following is a Verilog-A module template

```
include "discipline.h"
include "constants.h"
module alpha( n1, n2 );
electrical n1, n2;
parameter real arg1 = 2.0;
parameter integer arg2 = 0;
    real local;
    // this is a comment
    analog begin
        @ ( initial_step ) begin
            // performed at the first timestep of an analysis
        end
    end
```

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

```
// module behavioral description
V(n1, n2) <+ I(n1, n2) * arg1;
@ ( final_step ) begin
// performed at the last time step of an analysis
end
end
endmodule
```

### Verilog-A Compilation

The first time a verilogA file is used in simulation, Spectre performs a one-time compilation step. Following the initial compilation, recompilation is performed only if the Verilog-A source is changed.

The compiled C code flow stores the compiled shared objects in a database on the disk for the simulation to use. The shared objects are stored in a directory named `ahdlSimDB`. By default, this database is created in the current working directory and given a name created by appending `.ahdlSimDB` to the circuit name. You can specify an alternative location for `ahdlSimDB` by setting the `CDS_AHDLDMI_SIMDB_DIR` environment variable to the path of a directory, as follows:

```
setenv CDS_AHDLDMI_SIMDB_DIR /projects/ahdlcmiSimDirs
```

If the path is writable, `ahdlSimDB` is created there. If the path is not writable or does not exist, an error is reported.

To store compiled objects, use a second type of database, named `ahdlShipDBs`. To create such databases, set the `CDS_AHDLDMI_SHIPDB_COPY` to YES, as follows:

```
setenv CDS_AHDLDMI_SHIPDB_COPY YES
```

In this case, an `ahdlShipDB` is created for each Verilog-A file in the directory that contains the Verilog-A files, if the directory is writable. If the directory is not writable, no `ahdlShipDBs` are created for the modules in the Verilog-A file that is being processed.

If the `CDS_AHDLDMI_SHIPDB_DIR` environment variable (or the equivalent, but obsolete, `CDS_AHDLDMI_DIR` variable) is also set to a writable path, the `ahdlShipDB` database is created there and shared by all the Verilog-A files used for simulations that are run while this environmental variable is set. If the `CDS_AHDLDMI_SHIPDB_DIR` is not set to a writable path or the path does not exist, a warning is reported and `ahdlShipDBs` are not created.

The reuse of Verilog-A compiled library can also be achieved by command-line options, refer to the usage of the Spectre command-line options: `-ahdlshipdbdir` and `-ahdlshipdbmode`.

## **Language Summary**

The following provides a summary of the Verilog-A analog hardware description language. For more information refer to *Verilog-A Reference Manual*.



## ***Analog Operators/Waveform Filters***

---

`ddt(x <, abstol> )`      Differentiate x with respect to time.

`idt(x, ic <, assert <, abstol> > )`  
Integrate x with respect to time. Output = ic during dc analysis and when assert is 1.

`idtmod(x <, <ic <, modulus <, offset <, abstol> > > > )`  
Circular Integration of x with respect to time. Output = ic during DC analysis. Integration is performed with given offset and modulus, if specified.

`transition(x <, delay <, trise <, tfall <, timetol> > > > )`  
Specify details of signal transitions. For efficient simulation, it is recommended that x not be a continuous signal, that is, a function of a probe. See the Verilog-A manual for further explanation of this issue.

`slew(x <, SRpos <, SRneg>> )`  
Model slew rate behavior.

`delay(x, time_delay, max_delay)`  
Response(t) = x(t - time\_delay).

`zi_nd(x, numer, denom, period, < ttransition <, sample offset time > )`  
z-domain filter function, numerator-denominator form.

`zi_zd(x, zeros, denom, period, < ttransition <, sample offset time > )`  
z-domain filter function, zero-denominator form.

`zi_np(x, numer, poles, period, < ttransition <, sample offset time > )`  
z-domain filter function, numerator-pole form.

`zi_zp(x, zeros, poles, period, < ttransition <, sample offset time > )`  
z-domain filter function, zero-pole form.

`laplace_nd(x, numer, denom, <, abstol > )`  
s-domain filter function, numerator-denominator form.

`laplace_zd(x, zeros, denom, <, abstol > )`  
s-domain filter function, zero-denominator form.

`laplace_np(x, numer, poles, <, abstol > )`

## **Spectre Circuit Simulator Reference**

### **Other Simulation Topics**

---

---

s-domain filter function, numerator-pole form.

`laplace_zp(x, zeros, poles, <, abstol > )`

s-domain filter function, zero-pole form.

---

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

#### ***Built-In Mathematical Functions***

---

<code>abs(x)</code>	Absolute value
<code>exp(x)</code>	Exponential if $x < 80$
<code>ln(x)</code>	Natural logarithm
<code>log(x)</code>	Log base 10
<code>sqrt(x)</code>	Square root
<code>min(x,y)</code>	Minimum
<code>max(x,y)</code>	Maximum
<code>pow(x,y)</code>	x to the power of y

---

#### ***Noise Functions***

---

`white_noise( power <, tag > )`

Generates white noise with given power. Noise contributions with the same tag are combined for a module.

`flicker_noise( power, exp <, tag > )`

Generates pink noise with given power at 1 Hz that varies in proportion to  $1/f^{\text{exp}}$ . Noise contributions with the same tag are combined for a module.

`noise_table( vector <, tag > )`

Generates noise where power is determined by linear interpolation from the given vector of frequency-power pairs. Noise contributions with the same tag are combined for a module.

---

#### ***AC Analysis Stimuli***

---

`ac_stim( <analysis_name <, mag <, phase > > > )`

Small signal source of specified magnitude and phase in radians, active for given analysis.

---

## **Analog Events**

Analog events must be contained in an analog event detection statement; @(analog\_event) statement.

---

```
cross(x, direction <, timetol <, abstol <, enable >>>)
```

Generates an event when x crosses zero.

```
above(x <, timetol <, abstol <, enable >>>)
```

Generates an event when x becomes greater than or equal to zero. An above event can be generated and detected during initialization. By contrast, a cross event can be generated and detected only after at least one transient time step is complete.

```
timer(start_time <, period <, timetol <, enable >>> )
```

Set (optionally periodic) breakpoint event at time = start\_time.

```
initial_step< ( arg1 <, arg2 <, etc... > > )
```

Generate an event at the initial step of an analysis. arg1, arg2, and so on. Examples of analyses strings are "dc", "tran", "ac", "pss", "noise", "pdisto", "qpss", "pac", "pnoise", "pxf", "sp", "tdr", "xf", "envlp", "psp", "qpssp", "qpac", "qpnoise", "qpxf", "static", "ic", and so on.

```
final_step< ( arg1 <, arg2 <, etc... > > )
```

Generate an event at the final step of an analysis. arg1, arg2, and so on. Examples of analyses strings are "dc", "tran", "ac", "pss", "noise", "pdisto", "qpss", "pac", "pnoise", "pxf", "sp", "tdr", "xf", "envlp", "psp", "qpssp", "qpac", "qpnoise", "qpxf", "static", "ic", and so on.

---

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

#### ***Timestep Control***

---

<code>bound_step(max_step)</code>	Limit timestep, (timestep <= max_step).
<code>last_crossing(x, direction)</code>	Return time when expression last crossed zero in a given direction.
<code>discontinuity(n)</code>	Hint to simulator that discontinuity is present in nth derivative.

---

#### ***Simulator IO Functions***

---

<code>\$display(argument_list)</code>	Print data to stdout. Formatting strings may be interspersed between arguments/data.
<code>\$fdisplay(fp_ptr, argument_list)</code>	Print data to a file. Formatting strings may be interspersed between arguments/data.
<code>\$strobe(argument_list)</code>	Print data to stdout. Formatting strings may be interspersed between arguments/data.
<code>\$fstrobe(fp_ptr, argument_list)</code>	Print data to a file. Formatting strings may be interspersed between arguments/data.
<code>\$fscanf(fp_ptr, "format string" &lt;, arguments&gt;)</code>	Read data from a file
<code>\$fopen("filename", mode)</code>	Open a file for reading/writing
<code>\$fclose(fp_ptr)</code>	Close a file
<code>\$finish&lt;(n)&gt;</code>	Finish the simulation
<code>\$stop&lt;(n)&gt;</code>	Stop the simulation

---

#### ***Simulator Environment Functions***

---

<code>\$realtime</code>	Returns current simulation time
<code>\$temperature</code>	Returns ambient simulation temperature (K)
<code>\$vt</code>	Returns thermal voltage
<code>\$vt(temp)</code>	Returns thermal voltage at given temp
<code>\$analysis(analysis_string1&lt;, analysis_string2 &lt;, ...&gt;&gt;)</code>	

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

Returns true(1) if the current analysis phase matches one of the given analyses strings. The following are the examples of analyses strings: "dc", "tran", "ac", "pss", "noise", "pdisto", "qpss", "pac", "pnoise", "pxf", "sp", "tdr", "xf", "envlp", "psp", "qpssp", "qpac", "qpnoise", "qpxf", "static", "ic", and so on.

---

### ***Parameter Functions***

---

<code>\$pwr( x )</code>	Assignment of model power consumption. Adds the expression x to the pwr parameter of a module.
-------------------------	--

---

### ***Data Types***

---

<code>integer</code>	Discrete numerical type.
<code>real</code>	Continuous numerical type.

---

### ***Data Qualifiers***

---

<code>parameter</code>	Indicates that a variable is a parameter and so may be given a different value when the module is instantiated, and that it may not be assigned a different value inside the module.
------------------------	--

---

### ***Structural Statements***

Structural statements are used inside the module block but outside the analog block.

---

```
module_or_primitive #(<.param1(expr1)<,...>>) inst_name (<node1 <,  
...>> );
```

Creates a new instance of module\_or\_primitive named inst\_name.

---

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

#### **Environment Variables**

---

CDS_AHDL_CMI_SHIPDB_COPY	Set this environment variable to <code>YES</code> to use AHDL ship databases (ahdlShipDBs). When the environment variable is to <code>YES</code> , the software creates an ahdlShipDB for each Verilog-A file in the directory that contains the Verilog-A file, if the directory is writable. If the directory is not writable, the software does not create any ahdlShipDBs for the modules in the Verilog-A file (see CDS_AHDL_CMI_SHIPDB_DIR).
CDS_AHDL_CMI_SHIPDB_DIR	Specifies the path to a directory for ahdlShipDB. All Verilog-A files share this path. If the path specified by this variable is not writable, or the path does not exist, the software does not create any ahdlShipDB and generates a warning. For example, <code>setenv CDS_AHDL_CMI_SHIPDB_DIR /export/shared/objects</code> .
CDS_AHDL_COMPILE_C_MAX_LOAD	Set this environment variable to change the <code>max_load</code> value for parallel compilation of AHDL generated C code. When the system load average is above <code>max_load</code> , parallel compilation is turned off. For example, <code>setenv CDS_AHDL_COMPILE_C_MAX_LOAD 4.0</code> .
CDS_AHDL_DDT_SCALE	Set this environment variable to scale the value of <code>ddt</code> to the specified range for better convergence. The default value is <code>1e-17</code> .
CDS_AHDL_FINISH_MODE	Set this environment variable to <code>1</code> to cause the <code>\$finish</code> function to stop the current analysis instead of stopping the whole simulation. The default value is <code>0</code> .
CDS_AHDL_IDT_SCALE	Set this environment variable to scale the value of <code>idt</code> to the specified range for better convergence. The default value is <code>1</code> .
CDS_AHDL_IGNORE_HIDDEN_STATE	Set this environment variable to <code>YES</code> to process all hidden state variables as non-hidden state variables for all Verilog-A modules.

## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

CDS_AHDL_IGNORE_OPPOINT	Set this environment variable to YES to ignore all operating point calculations. Set the environment variable to NO to disable optimization. The default value is YES for compact models and NO for normal models.
CDS_AHDL_REUSE_LIB	In +aps, ++aps, +xps and +ms modes, the software automatically searches the pre-run shared library and reuses it. Set this environment variable to NO to disable the automatic search and reuse of pre-run shared library.
CDS_CMI_COMPLEVEL	Set this environment variable to control GCC optimization level when compiling AHDL-generated C code flow. The environment value is from 0 to 3, which implies that GCC will use level O0 ~ O3 to compile AHDL C code. O3 is the default compilation level and gives better optimization.
CDS_VLOGA_INCLUDE	Set this environment variable to specify the directory that contains the file with the "include" directive. You can specify more than one directory by using comma(,), colon(:), or semicolon(;) as delimiters to separate the directories. For example, <code>setenv CDS_VLOGA_INCLUDE "dir1;dir2"</code> or <code>setenv CDS_VLOGA_INCLUDE dir1:dir2</code> .
CDS_AHDL_LRM_COMPATIBILITY	Set this environment variable to specify the version of LRM. The default value is 2.3. For example, <code>setenv CDS_AHDL_LRM_COMPATIBILITY 2.4</code> .
CDS_AHDL_AUTOGDEV_SUPPORT	Set this environment variable to NO to prevent a Verilog-A model from automatically adding the simulation parameter <code>gdev</code> at a non-linear branch when the kernel uses the <code>gdev</code> method. The default value is YES, which means that the Verilog-A model adds the simulation parameter <code>gdev</code> by default.



## Spectre Circuit Simulator Reference

### Other Simulation Topics

---

---

CDS_AHDL_AUTOGMIN_INSERT	Set this environment variable to YES to enable a Verilog-A model to add the simulation parameter gmin at a non-linear branch. The default value is NO.
--------------------------	--

---

## **Spectre Circuit Simulator Reference**

### Other Simulation Topics

---

---

## Circuit Checks

---

This chapter discusses the following topics:

- [Dynamic Subckt Instance Activity Check \(dyn\\_activity\)](#) on page 597
- [Dynamic Active Node Check \(dyn\\_actnode\)](#) on page 599
- [Dynamic Capacitor Voltage Check \(dyn\\_capv\)](#) on page 601
- [Dynamic DC Leakage Path Check \(dyn\\_dcpv\)](#) on page 603
- [Dynamic Delay Check \(dyn\\_delay\)](#) on page 606
- [Dynamic Diode Voltage Check \(dyn\\_diodev\)](#) on page 609
- [Dynamic Excessive Element Current Check \(dyn\\_exi\)](#) on page 611
- [Dynamic Excessive Rise, Fall, Undefined State Time Check \(dyn\\_exrf\)](#) on page 613
- [Dynamic Floating Node Induced DC Leakage Path Check \(dyn\\_floatdcpv\)](#) on page 622
- [Dynamic Glitch Check \(dyn\\_glitch\)](#) on page 635
- [Dynamic HighZ Node Check \(dyn\\_highz\)](#) on page 638
- [Dynamic MOSFET Voltage Check \(dyn\\_mosv\)](#) on page 644
- [Dynamic Node Capacitance Check \(dyn\\_nodecap\)](#) on page 646
- [Dynamic Noisy Node Check \(dyn\\_noisynode\)](#) on page 648
- [Dynamic Pulse Width Check \(dyn\\_pulsewidth\)](#) on page 654
- [Dynamic Resistor Voltage Check \(dyn\\_resv\)](#) on page 656
- [Dynamic Rise Fall Edge Check \(dyn\\_risefall\)](#) on page 658
- [Dynamic Subckt Port Voltage/Current Check \(dyn\\_subcktpv\)](#) on page 664
- [Dynamic Subckt Port Power Check \(dyn\\_subcktpwr\)](#) on page 666

## Spectre Circuit Simulator Reference

### Circuit Checks

---

- [Static Capacitor Check \(static\\_capacitor\)](#) on page 671
- [Static Capacitor Voltage Check \(static\\_capv\)](#) on page 673
- [Static DC Leakage Path Check \(static\\_dcpv\)](#) on page 677
- [Static Diode Voltage Check \(static\\_diodev\)](#) on page 679
- [Static ERC Check \(static\\_erc\)](#) on page 681
- [Static HighZ Node Check \(static\\_highz\)](#) on page 686
- [Static MOSFET Voltage Check \(static\\_mosv\)](#) on page 688
- [Static NMOS to vdd count \(static\\_nmos2vdd\)](#) on page 690
- [Static NMOS Forward Bias Bulk Check \(static\\_nmosb\)](#) on page 692
- [Static Always Conducting NMOSFET Check \(static\\_nmosvgs\)](#) on page 695
- [Static PMOS Forward Bias Bulk Check \(static\\_pmosb\)](#) on page 699
- [Static Always Conducting PMOSFET Check \(static\\_pmosvgs\)](#) on page 702
- [Static Resistor Check \(static\\_resistor\)](#) on page 707
- [Static Resistor Voltage Check \(static\\_resv\)](#) on page 709
- [Static Subckt Port Voltage Check \(static\\_subcktport\)](#) on page 713
- [Static Transmission Gate Check \(static\\_tgate\)](#) on page 715
- [Static Voltage Domain Device Check \(static\\_voltdomain\)](#) on page 721

## Dynamic Subckt Instance Activity Check (dyn\_activity)

### Description

This check reports activity percentage of an instance relative to a circuit. The activity percentage is the ratio of the events in the instance versus the events in the entire circuit.

The results are written to the dynamic.xml file, which can be viewed in a Web browser. This check is supported only by XPS.

### Syntax

Name dyn\_activity parameter=value ...

### Parameters

#### Filtering parameters

time_window	[tstart, tstop] sec	Time window during which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend.
error_limit	10000	Maximum number of errors reported. Default is 10000.
min_activity	0	Any block activity below the minimum activity percentage will not be reported. The value can be between 0 and 1. Default is 0.

#### Wildcard scoping

inst	[...]	Subcircuit instances to which the check is applied. Default includes all instances (inst=*).
xinst	[...]	Subcircuit instances to be excluded from the check. Default is none.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

subckt	[ . . . ]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[ . . . ]	The instances of the specified subcircuits that are excluded from the check. Default is none.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

#### ***Related Topics***

[Dynamic Subckt Instance Activity Check](#)

## Dynamic Active Node Check (dyn\_actnode)

### Description

The active node checking analysis detects nodes with voltage changes that exceed the user-defined threshold,  $d_v$ . The voltage change is defined as the peak-to-peak voltage ( $V_{pp}$ ) within a time window.

Protected nodes are ignored from active and inactive detection.

Protectives nodes are not considered in the denominator of calculating percentage of active nodes.

The results are written to the dynamic.xml file, which can be viewed in a Web browser.

### Syntax

Name `dyn_actnode parameter=value ...`

### Parameters

Design check parameters

node	[...]	Nodes to which the check is applied. Default is none.
dv	0.1 v	Voltage change threshold for the active nodes. Default is 0.1 volt.
type	act	Report only inactive or active nodes or both. Possible values are act, inact and both.

Filtering parameters

time_window	[tstart, tstop] sec	Time window during which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend.
-------------	------------------------	---

## Spectre Circuit Simulator Reference

### Circuit Checks

---

error_limit	10000	Maximum number of errors reported. Default is 10000.
fanout	all	Fanout setting to filter node with specified connection. Possible values are all, gate and bulk.
filter	optimized	Filter setting to report all nodes or less nodes of interest. This parameter is supported only with Spectre FX. The default value is optimized. Possible values are rc, optimized and none.

#### Wildcard scoping

inst	[...]	Subcircuit instances to which the check is applied. Default includes all instances (inst=*).
xinst	[...]	Subcircuit instances to be excluded from the check. Default is none.
subckt	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[...]	The instances of the specified subcircuits that are excluded from the check. Default is none.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

#### ***Related Topics***

#### **Dynamic Active Node Check**



## Dynamic Capacitor Voltage Check (dyn\_capv)

### Description

Reports capacitor elements fulfilling the conditional expression on element voltages for a duration longer than the user-specified threshold duration.

Supported capacitor variables are: v(1,2), v(1), and v(2)

Supported operators are: +, -, \*, /, ==, !=, <, <=, >, >=, ||, &&, and !

The results are written to the dynamic.xml file, which can be viewed in a Web browser. This check is supported only by XPS. For Spectre and Spectre APS use assert statement.

### Syntax

Name dyn\_capv parameter=value ...

### Parameters

Design check parameters

cond		The conditional expression to be fulfilled. Default is none.
duration	5.00E-09 sec	Duration threshold. Default is 5.00E-09 sec.

Filtering parameters

time_window	[tstart, tstop] sec	The time window during which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend.
error_limit	10000	Maximum number of errors reported. Default is 10000.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### Wildcard scoping

inst	[ . . . ]	Subcircuit instances to which the check is applied. Default includes all instances (inst=*).
xinst	[ . . . ]	Subcircuit instances to be excluded from the check. Default is none.
subckt	[ . . . ]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[ . . . ]	The instances of the specified subcircuits that are excluded from the check. Default is none.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

#### ***Related Topics***

[Dynamic Capacitor Voltage Check](#)

## Dynamic DC Leakage Path Check (dyn\_dcpath)

### Description

Reports conductance paths between user-specified nets. This check can be used in two ways. One is based on transient-analysis and other is based on leakage-analysis. Both approaches cannot be combined in one check statement. If you specify both approaches in one check statement, then leakage analysis gets higher priority than transient analysis. Both approaches are explained below:

**\* Transient based approach:**

The check based on transient-analysis reports a qualifying path carrying an absolute current higher than the current threshold (`ith`) and for a time longer than the user-specified duration (`duration`), within the user-specified time window (`time_window`).

**\* Leakage analysis based approach:**

The check based on leakage analysis is evoked when parameter `leaki_times` is specified. The `leaki_times` should be carefully selected in standby or power-down mode (see Leakage Current Simulation in user-manual). The check is performed at specified times (`leaki_times`) of a transient simulation. The check based on leakage-analysis reports qualifying paths carrying an absolute current higher than the current threshold (`ith`) at user-specified time points (`leaki_times`).

If more than 2 nets are specified, Spectre checks the conducting path between each pair of nodes. For example, if `net=[vdc1 vdc2 0]` is specified, then the conducting path between `vdc1` and `vdc2`, `vdc1` and `0`, and `vdc2` and `0` is checked.

Higher accuracy settings are recommended when using the XPS solver:  
`+cktpreset=sram_pwr`.

The results are written to the `dynamic.xml` file, which can be viewed in a Web browser.

### Syntax

Name `dyn_dcpath` parameter=value ...

## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### Parameters

##### Design check parameters

net	[...]	The leakage path between the voltage source nodes is checked. <code>net</code> defines a set of nodes and not a pair. For example, <code>node = [vdd vdd1 0]</code> checks the leakage path between vdd and vdd1, vdd and 0, and vdd1 and 0.
ith	5.00E-05 A	The leakage path with the absolute current higher than the threshold value is reported. Default is 5.00E-05 A.
duration	5.00E-09 sec	Duration threshold. Default is 5.00E-09 sec.
sort	no	If set to no, sorting is not performed. If set to current, <code>dyn_dcpath</code> will sort violations by max current. Default is no. Possible values are no and current.

##### Filtering parameters

filter	no	If set to no, filtering is not performed. If set to yes, filtering is performed, <code>dyn_dcpath</code> checker will report only one path if the elements in the path are parallel and identical. Default is no. This option is only supported in SpectreFx. Possible values are no and yes.
time_window	[tstart, tstop] sec	Time window during which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend.
leaki_times	[...] sec	time point(s) at which dynamic <code>dcpath</code> check is performed. This parameter is not supported in MS.
error_limit	10000	Maximum number of errors reported. Default is 10000.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

xinst	[...]	Subcircuit instances to be excluded from the check. This parameter is only supported in XPS. Default is none.
-------	-------	---

#### ***Related Topics***

*Dynamic DC Leakage Current Path Check*

## Dynamic Delay Check (dyn\_delay)

### Description

Checks timing delays between two signals and reports nodes with edge delay errors.

The delay is measured between user-specified nodes and a reference node. A timing delay error occurs when the transition time of a signal falls outside the range of  $\text{refTime} + \text{min\_time}$  and  $\text{refTime} + \text{max\_time}$ , where  $\text{refTime}$  is the transition time of the reference signal. In other words, a timing delay error occurs when the delay between the signal and the reference signal is outside the range of  $\text{min\_time}$  and  $\text{max\_time}$ .

$\text{ref\_vhth}$  and  $\text{vhth}$  parameters are used for triggering rising edge measurements, while  $\text{ref\_vlth}$  and  $\text{vlth}$  parameters are used for triggering falling edge measurements. For example, a delay measurement from a rising reference signal to a falling signal includes measuring the delay from the time the reference signal crosses  $\text{ref\_vhth}$  to the time the signal crosses  $\text{vlth}$ .

If the  $\text{subckt}$  parameter is specified then  $\text{node}$  and  $\text{ref\_node}$  are considered as local nodes to the specified subcircuit. In other words,  $\text{node}$  and  $\text{ref\_node}$  will belong to the instances of the specified subcircuit. Only one  $\text{subckt}$  value can be specified per check, with no wildcard.

If the  $\text{subckt}$  parameter is not specified,  $\text{node}$  and  $\text{ref\_node}$  are considered as global nodes with hierarchical names starting from the top level.

The results are written to the `dynamic.xml` file, which can be viewed in a Web browser.

### Syntax

```
Name dyn_delay parameter=value ...
```

### Parameters

Design check parameters

<code>node</code>	<code>[...]</code>	Nodes to which the check is applied. Default is none.
<code>ref_node</code>		Name of the referenced node.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

<code>min_time</code>	<code>(sec)</code>	Minimum delay between the signal and the reference signal transition.
<code>max_time</code>	<code>(sec)</code>	Maximum delay between the signal and the reference signal transition.
<code>edge</code>	<code>rise</code>	Edge type of the signal net. Default is rise. Possible values are rise, fall and both.
<code>ref_edge</code>	<code>rise</code>	Edge type of the reference signal. Default is rise. Possible values are rise, fall and both.

#### Digitize parameters

<code>delay_chk_time</code>	<code>0.0 sec</code>	Delay time checking window.
<code>vlth</code>	<code>0.2 V</code>	Low voltage threshold for the signal net.
<code>ref_vlth</code>	<code>0.2 V</code>	Low voltage threshold for the referenced net.
<code>vhth</code>	<code>0.8 V</code>	High voltage threshold for the signal net.
<code>ref_vhth</code>	<code>0.8 V</code>	High voltage threshold for the referenced net.

#### Filtering parameters

<code>time_window</code>	<code>[tstart, tstop]</code> <code>sec</code>	Time window to which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend.
<code>error_limit</code>	<code>10000</code>	Maximum number of errors reported. Default is 10000.

**Spectre Circuit Simulator Reference**  
Circuit Checks

---

Wildcard scoping

subckt	none	Instances of the specified subcircuit to which the check is applied. Default is none.
--------	------	---

***Related Topics***

*Dynamic Delay Check*



## Dynamic Diode Voltage Check (dyn\_diodev)

### Description

Reports diode elements fulfilling the conditional expression on element voltages for a time longer than a user-specified duration threshold.

Supported diode variables:  $v(a,c)$ ,  $v(a)$ ,  $v(c)$

Supported operators: +, -, \*, /, ==, !=, <, <=, >, >=, ||, &&, and !

The results are written to the dynamic.xml file, which can be read with a web browser. This check is supported only by XPS. For Spectre and Spectre APS use assert statement.

### Syntax

Name dyn\_diodev parameter=value ...

### Parameters

Design check parameters

cond		The conditional expression to be fulfilled. Default is none.
duration	5.00E-09 sec	Duration threshold. Default is 5.00E-09 sec.
model	[...]	Diode device model names to be checked.

Filtering parameters

time_window	[tstart, tstop] sec	Time window to which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend.
error_limit	10000	Maximum number of errors reported. Default is 10000.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### Wildcard scoping parameters

inst	[...]	Subcircuit instances to which the check is applied. Default includes all instances (inst=*).
xinst	[...]	Subcircuit instances to be excluded from the check. Default is none.
subckt	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[...]	The instances of the specified subcircuits that are excluded from the check. Default is none.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

#### ***Related Topics***

[Dynamic Diode Voltage Check](#)

## Dynamic Excessive Element Current Check (dyn\_exi)

### Description

Reports elements and devices carrying currents (absolute value) higher than the current threshold (ith) for a time longer than the duration threshold (duration).

The results are written to the dynamic.xml file, which can be viewed in a Web browser.

### Syntax

Name dyn\_exi parameter=value ...

### Parameters

Design check parameters

dev	[...]	The instance names of device or elements to be checked.
ith	(ampere)	Current threshold. Default is none.
duration	5.00E-09 sec	Duration threshold. Default is 5.00E-09 sec.

Filtering parameters

time_window	[tstart, tstop] sec	The time window during which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend.
error_limit	10000	Maximum number of errors reported. Default is 10000.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### Wildcard scoping

inst	[...]	Subcircuit instances to which the check is applied. Default includes all instances (inst=*).
xinst	[...]	Subcircuit instances to be excluded from the check. Default is none.
subckt	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[...]	The instances of the specified subcircuits that are excluded from the check. Default is none.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

#### ***Related Topics***

*Dynamic Excessive Element Current Check*

## Dynamic Excessive Rise, Fall, Undefined State Time Check (dyn\_exrf)

### Description

Reports the nodes with excessive rise times, fall times, or with an undefined state. The rise and fall times are measured between low (vlth) and high (vhth) voltage thresholds. The duration longer or shorter than the user-specified rise and fall time threshold are reported. Undefined states are defined by node voltages between low and high voltage threshold and are reported when their duration is longer than the undefined time threshold (utime).

The results are written to the dynamic.xml file, which can be viewed in a Web browser.

### Syntax

Name dyn\_exrf parameter=value ...

### Parameters

Design check parameters

node	[...]	Nodes to which the check is applied. Default is none.
rise	(sec)	Risetimes longer than the specified value are reported. Default is none.
fall	(sec)	Falltimes longer than the specified value are reported. Default is none.
utime	(sec)	The undefined time threshold. An undefined state is defined by node voltages between the low and high voltage thresholds.
dv	0 V	Specifies the delta voltage tolerance for a monotonic edge check. If the delta voltage for an edge is higher than the delta voltage tolerance, it is reported as a violation. This parameter is supported only in APS mode. Default is 0.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

fanout	all	Fanout setting to filter node with specified connection. When fanout=gate_has_driver_no_moscap, floating nodes only connecting to MOSCAPs will not be checked. When fanout=gate_no_moscap, gates only connecting to MOSCAPs will not be checked. This option is supported only in XPS. Possible values are all, gate, bulk, gate_has_driver_no_moscap and gate_no_moscap.
filter	no	If set to no, filtering is not performed. If set to rc, dyn_exrf checker will report only one node in same parasitic sub. This option is supported only in XPS. Possible values are no and rc.
inverse	no	If set to no, reports risetimes and falltimes longer than the specified value. If set to yes, reports risetimes and falltimes shorter than the specified value. This option is supported only in XPS. Default is no. Possible values are no and yes.
sort	no	If set to no, sorting is not performed. If set to duration, dyn_exrf will sort violations by duration. Default is no. Possible values are no and duration.

#### Digitize parameters

vlth	(V)	Low voltage threshold for rise, fall, and utime measurements (default is none).
vhth	(V)	High voltage threshold for rise, fall, and utime measurements (default is none).

## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### Filtering parameters

<code>time_window</code>	<code>[tstart, tstop]</code> sec	Time window to which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend.
<code>error_limit</code>	10000	Maximum number of errors reported. Default is 10000.

#### Wildcard scoping

<code>inst</code>	<code>[...]</code>	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
<code>xinst</code>	<code>[...]</code>	Subcircuit instances to be excluded from the check. Default is none.
<code>subckt</code>	<code>[...]</code>	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits ( <code>subckt=*</code> ).
<code>xsubckt</code>	<code>[...]</code>	The instances of the specified subcircuits that are excluded from the check. Default is none.
<code>depth</code>	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

#### ***Related Topics***

#### ***Dynamic Capacitor Voltage Check***

## Dynamic Floating Node Statistical Check (dyn\_float\_tran\_stat)

### Description

The dyn\_float\_tran\_stat check reports leaky MOSFETs/BJTs caused by floating nodes. It also detects floating nodes caused by other floating nodes.

The check deploys a statistical method of forcing (pinging) different voltages on the gate of MOSFET and the base of BJT. Ping refers to randomly forcing a node to 0 or VDD, or ramping from 0 to VDD. Only MOSFET gates or BJT base nodes are pinged. Ping is realized by applying a voltage source in series with a large resistor (ping\_resistor) on each node. The strength of a ping is weaker (because of large ping\_resistor) than other drivers on each node.

#### dyn\_float\_tran\_stat Algorithm

- Transient simulation is run till ping\_start.
- Qualified nodes are identified to exercise pings according to domains
- For each qualified node, domain\_voltage is evaluated
- Pings are created randomly using seed value. One third of the qualified nodes are selected for HIGH. One third of qualified nodes are selected for LOW. One third of qualified nodes are selected for RAMP.
- ping\_duration and ping\_resistor are evaluated
- Pings are applied sequentially at ping\_start.
- For each ping:
  - All MOSFET drain currents and BJT collector currents are stored as reference current (Iref) right before applying the ping
  - For the first half of ping\_duration:
    - HIGH/LOW to each node is forced through a resistor with ping\_resistor value.
    - Device current (Iping) is checked at 0.95 of 0.5\*ping\_duration. A fail is detected if:  
$$|I_{ref} - I_{ping}| > \max(ping_{ibsthresh}, ping_{irelthresh} * \min(|I_{ref}|, |I_{ping}|))$$
  - For the second half of ping\_duration:



## Spectre Circuit Simulator Reference

### Circuit Checks

---

- The RAMP is forced to each node through a resistor with the ping\_resistor value and the maximum (Iping,max) and minimum (Iping,min) device current is checked during the RAMP. A fail is detected in any of the following cases:

$$|I_{ref}-I_{ping,max}| > \max(\text{pingiabsthresh\_ramp}, \text{pingirelthresh}*\min(|I_{ref}|,|I_{ping,max}|))$$

$$|I_{ref}-I_{ping,min}| > \max(\text{pingiabsthresh\_ramp}, \text{pingirelthresh}*\min(|I_{ref}|,|I_{ping,min}|))$$

The results are written to the dynamic.xml file, which can be viewed in a Web browser. This design check is supported only in Spectre, Spectre APS modes. It is not supported in Spectre XPS FastSPICE , Spectre XPS SPICE, and Spectre XPS MS modes.

### Syntax

Name dyn\_float\_tran\_stat parameter=value ...

### Parameters

Design check parameters

domains	[...]	Pairs of power supply domains in which floating nodes are checked. Example: domains=[VDD VSS VPP VSS]. Domain also defines domain voltage (domain_voltage) value for forcing node voltages. Default is none. Wildcarding is not allowed.
ping_start	[...]	Time at which the check is initiated.
ping_count	300	Number of statistical periods being used. Default is 300.
ping_duration	0	Duration of a ping. A ping consists of two phases: high/low forcing phase and ramp up forcing phase. Length of each phase is half of ping_duration. Default is none..

## Spectre Circuit Simulator Reference

### Circuit Checks

---

ping_iabsthresh	1	Current threshold for device violation during high/low forcing (first phase). Specify a value and a list of instance name and value pairs in the format [value0 [inst1 value1 inst2 value2 ... ]]. If inst value pairs are not specified, value0 is used for MOSFETs and BJTs in all domains. If an inst value pair is specified, the given value is used for all MOSFETs and BJTs of the given inst. For all other MOSFETs and BJTs, value0 is used. Each value must be a number. It cannot be a netlist parameter or a mathematical expression. If value0 is not specified, the default value 1 indicates that the default value is $\text{domain\_voltage}/\text{ping\_resistance}$ .
ping_iabsthresh_ramp	1	Current threshold for device violation during ramp forcing (second phase). Specify a value and a list of instance name and value pairs in the format [value0 [inst1 value1 inst2 value2 ... ]]. If inst value pairs are not specified, value0 is used for MOSFETs and BJTs in all domains. If an inst value pair is specified, the given value is used for all MOSFETs and BJTs of the given inst. For all other MOSFETs and BJTs, value0 is used. Each value must be a number. It cannot be a netlist parameter or a mathematical expression. If value0 is not specified, the default value 1 indicates that the default value is $100 * \text{domain\_voltage}/\text{ping\_resistance}$ .
ping_irelthresh	1000	Relative threshold for floating node detection for both high/low and ramp forcing. Default is 1000.
seed		Seed for random number generation. Default is none.
domain_voltage_thresh	0.0	Lowest threshold for domain voltage. If a node has voltage less than <code>domain_voltage_thresh</code> , it will be excluded from ping. Default is 0.0.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

save	no	To probe gate voltage of MOSFETs, base voltage of BJTs, drain current of MOSFETs and collector current of BJTs. When value is set to ping, netlist or all, a new waveform file will be written in raw directory. The format of this new waveform file will be <CheckName>_<PingStart>.tran.tran, where <CheckName> is the name of check statement and <PingStart> is the time specified in ping_start parameter. The waveform will start from PingStart until ping simulation is finished. If save=ping, only save ping voltages and mos/bjt currents, if save=netlist, only save the ones saved in the netlist, if save=all, both of signals in the save=ping and the ones saved in the netlist are saved, Default is no. Possible values are ping, no, all and netlist.
distribute	no	Distribute method. Possible values are no, fork and lsf.
numprocesses	0	Number of processes for distributed analysis.
report_top_current	0	number of top n devices with the largest leakage current to be reported. When set to 0, report will contain all devices.
ping_duration_max	0	maximum value of ping_duration applied. Default is none.
ping_resistor_max	10e+9	Maximum value of ping_resistor applied. Specify a value and a list of instance name and value pairs in the format [value0 [inst1 value1 inst2 value2 ... ]]. If inst value pairs are not specified, value0 is used in all domains. If an inst value pair is specified, the given value is used at all ports and nodes of the given inst. For all other nodes, value0 is used. Each value must be a number. It cannot be a netlist parameter or a mathematical expression.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

ping_resistor_min	100e+6	Minimum value of ping_resistor applied. Specify a value and a list of instance name and value pairs in the format [value0 [inst1 value1 inst2 value2 ... ]]. If inst value pairs are not specified, value0 is used in all domains. If an inst value pair is specified, the given value is used at all ports and nodes of the given inst. For all other nodes, value0 is used. Each value must be a number. It cannot be a netlist parameter or a mathematical expression.
time_constant	6	time constant. Default is 6.0.
domain_detection_method	dev	Domain detection, find the ping nodes through device or node. Possible values are dev and node.
stop_nodes	[...]	stop nodes during domain tracking. Default is none.
inst	[...]	Subcircuit instances to which the check is applied. Default includes all instances (inst=*).
xinst	[...]	Subcircuit instances to be excluded from the check. Default is none.
subckt	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[...]	The instances of the specified subcircuits that are excluded from the check. Default is none.
node	[...]	Nodes to which the check is applied. Default is none.
xnode	[...]	Nodes to be excluded from the check. Default is none.
ping_voltage_threshold	0	Threshold for ping voltage.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

ping_index	[...]	Selectively re-ping a circuit with a list of pings.
domain_detection _voltage_atol	1e-20	Voltage tolerance for volatge-based domain detection.
ping_timeout	0	Interrupt any individual ping if it takes longer than ping_timeout. The specified time must be a CPU time. Units is in seconds. Default is none.
freeze	no	When set to yes, independent source is frozen during ping simulation. The value of independent source throughout a ping simulation is constant. The constant value is equal to the value of independent source at ping_start. Possible values are no and yes.

## Dynamic Floating Node Induced DC Leakage Path Check (dyn\_floatdcpath)

### Description

Reports DC leakage paths caused by floating gate.

This check can be used in two ways. One is based on transient analysis and other is based on leakage analysis. Both approaches cannot be combined in one check statement. If you specify both approaches in one check statement, then leakage analysis gets higher priority than transient analysis. Both approaches are explained below:

#### \* Transient based approach:

The check based on transient analysis is evoked when parameters `time_window` or/and `duration` are specified. It reports the qualifying path carrying an absolute current higher than the parameter `ith` for a duration longer than the specified duration (`duration`), within the specified time window (`time_window`). This qualifying path must also have a MOSFET with gate floating for duration longer than the specified duration (`duration`), within the specified time window (`time_window`). The violations will be the union of `dyn_highz` and `dyn_dcpath`, if same setting are used. This methodology does not give the worst case leakage path. If settings of `dyn_dcpath` and `dyn_floatdcpath` are same then the violations of `dyn_floatdcpath` will be subset of `dyn_dcpath`. The following parameters are irrelevant: `Vmin`, `vmax`, `points`, `rforce`, `leaki_times`, `sweep`, `floatgate`, `detailed_report`, `debug_net`, `distribute`, `numprocesses`. The floating node detection is similar to `dyn_highz`. The path detection is similar to `dyn_dcpath`.

#### \* Leakage analysis based approach:

The check based on leakage analysis is evoked when parameter `leaki_times` is specified. The `leaki_times` should be carefully selected in standby or power-down mode (see Leakage Current Simulation in user-manual). The check is performed at specified times (`leaki_times`) of a transient simulation. Floating nodes and MOSFET devices with floating gates are detected. Potential leakage paths (`sweep=no`) or worst leakage paths (`sweep=all,single,all_once`) caused by floating gate MOSFET devices are reported. The leakage paths are checked between user specified power supply nodes (`net`). The floating node detection and path detection, in Leakage analysis based approach, are explained below:

#### \*\* Floating node detection (`leaki_times`):

A node is considered floating (or in high impedance state) if it has no conducting path to a power supply or ground. A report of all MOSFETs with floating gates can be printed by using

## Spectre Circuit Simulator Reference

### Circuit Checks

---

the parameter `floatgate`. The following device conducting rules are used for the floating node detection:

- MOSFET is conducting if mos region is triode OR saturation
- Resistors, controlled resistors, `phy_res`, relay and inductors are conducting if  $R \leq \text{res\_th}$
- BJT is conducting if  $V_{be} > \text{bjt\_vbe}$  OR  $I_c > \text{bjt\_ith}$
- Diode is conducting if  $V > \text{diode\_vth}$
- Vsources and iprobes are considered conducting
- Isources, VCCS, and CCCS are conducting if  $i > \text{isource\_ith}$
- JFET is considered conducting
- Mutual inductors and controlled capacitors are not conducting
- VerilogA: conducting path depends on module details

\*\* Path detection (`leaki_times`):

Once the floating nodes are identified, the leakage paths caused by the floating nodes need to be detected. You can use the current-based method or conducting-rule-based method to detect the leakage path. These two methods are explained below:

\*\*\* When `sweep` parameter is set to `single`, `all`, or `all_once` then the leakage path detection is current-based. In the current-based path detection method, floating node voltage is swept, and the current in the path is measured. After measurement, current-based method reports qualifying paths carrying an absolute current higher than the parameter `ith`. Either all floating nodes are swept at once (`sweep=all`), or each floating node is swept individually (`sweep=single`). The following parameters are relevant to current-based path detection method: `vmin`, `vmax`, `points`, `rforce`, and `ith`.

\*\*\* When `sweep` parameter is set to `no`, then the leakage path detection is based on following conducting rules:

- MOSFET with floating gate is always considered conducting.
- Resistors, controlled resistors, and inductors are conducting if  $R \leq \text{res\_th}$
- BJT is conducting if  $V_{be} > \text{bjt\_vbe}$  OR  $I_c > \text{bjt\_ith}$
- Diode is conducting if  $V > \text{diode\_vth}$

## Spectre Circuit Simulator Reference

### Circuit Checks

---

- Vsources is conducting if V=0 otherwise it is not conducting
- iprobes are considered conducting
- Isources, VCCS, and CCCS are conducting if i>isource\_ith
- VerilogA: conducting path depends on module details

The following parameters are irrelevant with sweep=no: Vmin, vmax, points, and rforce.

If more than 2 nets are specified, Spectre checks the leakage path between each pair of nets. For example, if net=[vdc1 vdc2 0] is specified, then the conducting path between vdc1 and vdc2, vdc1 and 0, and vdc2 and 0 is checked.

Higher accuracy settings are recommended when using the XPS solver:  
+cktpreset=sram\_pwr.

The results are reported into a file with the extension dynamic.xml, which can be read with a web browser.

### Syntax

Name dyn\_floatdcpath parameter=value ...

### Parameters

Design check parameters

leaki_times	[...] sec	The time point(s) at which the dynamic floatdcpath check is performed.
duration	5.00E-09 sec	Duration threshold. Default is 5.00E-09 sec. This option is supported only in Spectre and APS.
time_window	[tstart, tstop] sec	Time window during which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend. This option is supported only in Spectre and APS.
error_limit	10000	Maximum number of errors reported. Default is 10000.



## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### Floating node detection parameters

mos_ith	100E-9	Mosfet ids conducting threshold for high impedance node detection. Default is 100 nA.
mos_gds	1.0E-5	Mosfet gds conducting threshold for floating node detection. Default is 1e-5.
bjt_vbe	0.4	BJT vbe conducting threshold for floating node detection. Default is 0.4 V.
bjt_ith	50E-9	BJT ic conducting threshold for floating node detection. Default is 50 nA.
res_th_va	10E6	VerilogA resistor conducting threshold for high impedance node detection. Only applicable on AMS IE element and on one or two port AHDL module. Default is 10MOhms.
res_th	1E12	Resistor conducting threshold for floating node detection. Default is 1 TOhm.
diode_vth	0.6	Diode voltage conducting threshold for high impedance node detection. Default is 0.6V.
isource_ith	1.0E-12	Current source conducting threshold for floating node detection. Default is 1 pA.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

mos_conduct_rule	tri_sat	The conducting rule to decide whether a MOSFET is conducting or not. Possible Values tri_sat: Specifies that a MOSFET is conducting when region is either triode or saturation. tri_sat_sub: Specifies that a MOSFET is conducting when region is either triode, saturation, or subthreshold. ids_gds: Specifies that a MOSFET is conducting when ids>mos_ith or gds>mos_gds, where mos_ith and mos_gds are check parameters. vgs: Specifies that a MOSFET is conducting when vgs>vth (Only supported in XPS Fastspice). The region, ids, gds and vgs are oppoint parameters of MOSFET. Default is tri_sat.Possible values are vgs, ids_gds, tri_sat and tri_sat_sub.
debug_net	[...]	Hierarchical net(s) to debug. Wildcard is not allowed. Default is none. The debug report will be generated at leaki_times.
nonconducting_subckt	[...]	All instances of specified subcircuit are considered non-conducting during the floating node detection. All devices inside the subcircuit are considered non-conducting during the floating node detection. Wildcard is supported. Default is none.
conducting_subckt	[...]	All instances of specified subcircuit are considered conducting during the floating node detection. All devices inside the subcircuit are considered conducting during the floating node detection. Wildcard is supported. Default is none.
nonconducting_primitive	[...]	All instances of specified primitive or VerilogA modules are considered non-conducting during the floating node detection. Wildcard is supported. Default is none.
nonfloat_vpn	[...]	All specified vpn(s) will be marked as non-floating nodes during floating node detection. Wildcard is not allowed. Default is none. This option is supported only in Spectre XPS FastSPICE mode.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

float_vpn	[...]	All channel connections of specified vpn(s) will be marked as floating nodes during floating node detection. Wildcard is not allowed. Default is none. This option is supported only in Spectre XPS FastSPICE mode.
-----------	-------	---

#### Floating node detection filtering parameters

skip_domains	[...]	Pairs of power supply domains in which floating nodes are skipped. Default is none. Wildcarding is not allowed.
floatgate	no	Whether to report all MOSFETs with floating gate at leaki_times. When floatgate=gate_has_driver_no_moscap, floating MOSCAPs will not be reported. Possible values are no, yes and gate_has_driver_no_moscap. Default is no. Possible values are no, yes and gate_has_driver_no_moscap.
filter	no	If set to no, filtering is not performed. If set to rc, dyn_floatdcpath will check only one node in same parasitic sub. This option is supported only in XPS. Possible values are no and rc.
filter_diocon_no des	0	Filter out highz nodes connected to MOSDIODE. When set to 1, highz node that is connected to PMOSDIODE is filtered out given that its source is connected to VDD. Moreover, a highz node tat is connected to NMOSDIODE is filtered out given that its source is connected to 0. When set to 2, highz node that is connected to PMOSDIODE is filtered out given that its source voltage is greater than drain voltage. Moreover, a highz node that is connected to NMOSDIODE is filter out given that its drain voltage is greater than source voltage. This option is supported only in Spectre and APS. Possible values are 0, 1 and 2.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

node	[ . . . ]	Nodes to which the floating node detection is applied. Default is all( node=* ).
xnode	[ . . . ]	Nodes to be excluded from the floating node detection. Default is none.
inst	[ . . . ]	Subcircuit instances to which the floating node detection is applied. Default includes all instances (inst=*).
xinst	[ . . . ]	Subcircuit instances to be excluded from the floating node detection. Default is none.
subckt	[ . . . ]	The instances of the specified subcircuit to which the floating node detection is applied. Default includes all subcircuits (subckt=*).
xsubckt	[ . . . ]	The instances of the specified subcircuits that are excluded from the floating node detection. Default is none.

#### Path detection parameters

net	[ . . . ]	Hierarchical node names between which the leakage path is checked. Wildcarding is not allowed.
-----	-----------	--

## Spectre Circuit Simulator Reference

### Circuit Checks

---

sweep	all	If set to no, then no DC sweep is performed and leakage path is based on conducting rules. If set to single, then each floating node is swept one at a time and leakage paths more than <code>ith</code> are reported. If set to all, then all floating nodes will be swept together and leakage paths more than <code>ith</code> are reported. If set to <code>all_once</code> , voltage of gates connected to nmos will be VDD, voltage of gates connected to pmos will be 0, voltage of gates connected to both nmos and pmos will be VDD/2, and leakage paths more than <code>ith</code> are reported. The default is all in Spectre and APS, and the default is no in XPS. The option <code>all_once</code> is supported only in XPS. The option <code>single</code> is not supported in XPS..Possible values are no, all, single and <code>all_once</code> .
ith	50E-6	Leakage path more than <code>ith</code> will be reported. Applicable only with <code>sweep=single</code> , <code>sweep=all</code> and <code>sweep=all_once</code> . The default value is 50uA.
points	3	Defines number of sweep points, applicable on both <code>sweep=single</code> and <code>sweep=all</code> .
vmin	0	Defines a minimum voltage for <code>sweep=single</code> and <code>single=all</code> . If specified, the voltage sweep will start from specified <code>vmin</code> voltage. If not specified, minimum voltage in a design will be used. Default is none.
vmax	0	Defines a maximum voltage for <code>sweep=single</code> and <code>single=all</code> . If specified, the voltage sweep will stop at specified <code>vmax</code> voltage. If not specified, maximum voltage in a design will be used. Default is none.
rforce	500E06	Change the <code>rforce</code> value only for <code>sweep=single</code> and <code>sweep=all</code> . The default value is 500 Mohm.
numprocesses	0	Number of processes for distributed analysis.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

distribute	no	Distribute method, only support sweep=single.Possible values are no, fork, lsf and sge.
------------	----	---

#### Path detection filtering parameters

detailed_path	per_fm	If set to yes, prints the detailed path. Default is per_fm, print one path per floating mosfet. If set to per_fn, print one path per floating node. The per_fn value is not supported in Spectre XPS.Possible values are per_fm, yes and per_fn.
xinst_path	[...]	Subcircuit instances to be excluded from the path detection. Default is none.
sort	no	If set to no, sorting is not performed. If set to current, dyn_floatdcpath will sort violations by max current. Default is no.Possible values are no and current.

#### ***Related Topics***

*Dynamic Floating Gate Induced Leakage Check*

## Dynamic Float Crosstalk Check (dyn\_floatxtalk)

### Description

Reports the glitches of high impedance nodes caused by crosstalk of parasitic capacitors in DSPF file. A high impedance state occurs when there is no conducting path from the node to any power supply or ground.

The following device conducting rules are used for the floating node detection:

- MOSFET is conducting if mos region is triode or saturation
- Resistors, controlled resistors, phy\_res, relay and inductors are conducting if  $R \leq \text{res\_th}$
- BJT is conducting if  $V_{be} > \text{bjt\_vbe}$  OR  $I_c > \text{bjt\_ith}$
- Diode is conducting if  $V > \text{diode\_vth}$
- Vsources and iprobes are considered conducting
- Isources, VCCS, and CCCS are conducting if  $i > \text{isource\_ith}$
- JFET is considered conducting
- Mutual inductors and controlled capacitors are not conducting
- VerilogA: conducting path depends on module details

The results are written to the dynamic.xml file, which can be viewed in a Web browser. This check is supported only with Spectre FX.

### Syntax

Name `dyn_floatxtalk` parameter=value ...

### Parameters

Design check parameters

<code>leaki_times</code>	<code>[...]</code> sec	The time point(s) at which the dynamic crosstalk check is performed.
--------------------------	---------------------------	--

## Spectre Circuit Simulator Reference

### Circuit Checks

---

rise_vth	0.2	The threshold of voltage change caused by pulling up of an aggressor. Default is 0.2 V.
fall_vth	0.2	The threshold of voltage change caused by pulling down of an aggressor. Default is 0.2 V.
cmin	1.0e-16	If the lumped capacitance between highZ NET and its aggressor is less than cmin, Caggr<cmin, such aggressor is ignored. Default is 1.0e-16 F.
capr	0.2	If the ratio of between the lumped capacitance and the total capacitor of HighZ NET is less than capr, Caggr/Ctotal<capr, such aggressor is ignored. Default is 0.2.
error_limit	10000	Maximum number of errors reported. Default is 10000.

#### Floating node detection parameters

mos_ith	100E-9	Mosfet ids conducting threshold for high impedance node detection. Default is 100 nA.
mos_gds	1.0E-5	Mosfet gds conducting threshold for floating node detection. Default is 1e-5.
bjt_vbe	0.4	BJT vbe conducting threshold for floating node detection. Default is 0.4 V.
bjt_ith	50E-9	BJT ic conducting threshold for floating node detection. Default is 50 nA.
res_th_va	10E6	VerilogA resistor conducting threshold for high impedance node detection. Only applicable on AMS IE element and on one or two port AHDL module. Default is 10MOhms.



## Spectre Circuit Simulator Reference

### Circuit Checks

---

res_th	1E12	Resistor conducting threshold for floating node detection. Default is 1 TOhm.
diode_vth	0.6	Diode voltage conducting threshold for high impedance node detection. Default is 0.6V.
isource_ith	1.0E-12	Current source conducting threshold for floating node detection. Default is 1 pA.
mos_conduct_rule	tri_sat	The conducting rule to decide if a mosfet is on. Use tri_sat for triode or saturation region based rule. Use tri_sat_sub for triode or saturation or subthreshold region based rule. Use ids_gds for ids>mos_ith or gds>mos_gds rule. Use vgs for vgs>vth rule (Only supported in XPS Fastspice). Possible values are tri_sat, tri_sat_sub, ids_gds, and vgs. Default is tri_sat..Possible values are vgs, ids_gds, tri_sat and tri_sat_sub.

#### Floating node detection filtering parameters

node	[...]	Nodes to which the floating node detection is applied. Default is all( node=* ).
xnode	[...]	Nodes to be excluded from the floating node detection. Default is none.
inst	[...]	Subcircuit instances to which the floating node detection is applied. Default includes all instances (inst=*).
xinst	[...]	Subcircuit instances to be excluded from the floating node detection. Default is none.
subckt	[...]	The instances of the specified subcircuit to which the floating node detection is applied. Default includes all subcircuits (subckt=*).

## Spectre Circuit Simulator Reference

### Circuit Checks

---

xsubckt	[ . . . ]	The instances of the specified subcircuits that are excluded from the floating node detection. Default is none.
---------	-----------	---

## Dynamic Glitch Check (dyn\_glitch)

### Description

A 'Glitch' occurs when:

- A low signal goes above the mid-level, and crosses the mid-level again within the user-defined duration.
- A high signal goes below the mid-level, and crosses the mid-level again within the user-defined duration.

This check applies only to blocks with single and constant power supply.

The results are written to the dynamic.xml file, which can be viewed in a Web browser.

### Syntax

Name `dyn_glitch` parameter=value ...

### Parameters

Design check parameters

node	[...]	Nodes to which the check is applied. Default is none.
duration	5.00E-09 sec	Duration threshold. Default is 5.00E-09 sec.
min_duration	0 sec	Minimum glitch duration threshold. Default is 0 sec.
max_duration	5.00E-09 sec	Maximum glitch duration threshold. Default is 5.00E-09 sec.
low	0 volt	Low-level voltage. Default is 0 V.
high	(volt)	High-level voltage. Default is none.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

mid	$0.5 * (\text{high} + \text{low})$ volt	Mid-level voltage for glitch detection. Default is $0.5 * (\text{high} + \text{low})$ .
-----	--	---

#### Filtering parameters

time_window	[tstart, tstop] sec	The time window during which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend.
error_limit	10000	Maximum number of errors reported. Default is 10000.
fanout	all	Fanout setting to filter node with specified connection. This option is supported only in XPS. Possible values are all, gate and bulk.

#### Wildcard scoping

inst	[...]	Subcircuit instances to which the check is applied. Default includes all instances (inst=*).
xinst	[...]	Subcircuit instances to be excluded from the check. Default is none.
subckt	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[...]	The instances of the specified subcircuits that are excluded from the check. Default is none.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

#### **Related Topics**

## **Spectre Circuit Simulator Reference**

### **Circuit Checks**

---

#### **Dynamic Glitch Check**

## Dynamic HighZ Node Check (dyn\_highz)

### Description

Reports the nodes that are in high impedance state (also known as floating) for a duration longer than the user-defined threshold. A high impedance state occurs when there is no conducting path from the node to any power supply or ground.

The following device conducting rules are used for the floating node detection:

- MOSFET is conducting if mos region is triode or saturation
- Resistors, controlled resistors, phy\_res, relay and inductors are conducting if  $R \leq \text{res\_th}$
- BJT is conducting if  $V_{be} > \text{bjt\_vbe}$  OR  $I_c > \text{bjt\_ith}$
- Diode is conducting if  $V > \text{diode\_vth}$
- Vsources and iprobes are considered conducting
- Isources, VCCS, and CCCS are conducting if  $i > \text{isource\_ith}$
- JFET is considered conducting
- Mutual inductors and controlled capacitors are not conducting
- VerilogA: conducting path depends on module details

The results are written to the dynamic.xml file, which can be viewed in a Web browser.

### Syntax

```
Name dyn_highz parameter=value ...
```

### Parameters

Design check parameters

node	[...]	Nodes to which the check is applied. Default is none.
------	-------	---

## Spectre Circuit Simulator Reference

### Circuit Checks

---

xnode	[...]	Nodes to be excluded from the check. Default is none.
xnode_file	[...]	Nodes to be excluded from the check, defined in file, wildcard is not supported. Default is none.
duration	5.00E-09 sec	Duration threshold. Default is 5.00E-09 sec.
mos_ith	100E-9	Mosfet ids conducting threshold for high impedance node detection. Default is 100nA.
mos_gds	1.0E-5	Mosfet gds conducting threshold for high impedance node detection. Default is 1e-5.
bjt_vbe	0.4	BJT vbe conducting threshold for high impedance node detection. Default is 0.4V.
bjt_ith	50E-9	BJT ic conducting threshold for high impedance node detection. Default is 50nA.
res_th_va	10E6	VerilogA resistor conducting threshold for high impedance node detection. Only applicable on AMS IE element and on one or two port AHDL module. Default is 10MOhms.
res_th	1E12	Resistor conducting threshold for high impedance node detection. Default is 1 TOhms.
diode_vth	0.6	Diode voltage conducting threshold for high impedance node detection. Default is 0.6V.
isource_ith	1.0E-12	Current source conducting threshold for high impedance node detection. Default is 1pA.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

mos_conduct_rule	tri_sat	The conducting rule to decide if a mosfet is on. Use tri_sat for triode or saturation region based rule. Use tri_sat_sub for triode or saturation or subthreshold region based rule. Use ids_gds for ids>mos_ith or gds>mos_gds rule. Use vgs for vgs>vth rule (Only supported in XPS Fastspice). Possible values are tri_sat, tri_sat_sub, ids_gds, and vgs. Default is tri_sat..Possible values are vgs, ids_gds, tri_sat and tri_sat_sub.
inverse	no	If set to no, reports all nodes that are in highz state. If set to yes, reports all nodes not being in highz state. Default is no.Possible values are no and yes.
sort	no	If set to no, sorting is not performed. If set to duration, dyn_highz will sort violations by duration. Default is no.Possible values are no and duration.
debug_net	[...]	Hierarchical net(s) to debug. Wildcard is not allowed. Default is none.
debug_time	[...]	Time point(s) at which debug report will be generated. Default is none.
debug_format	txt	When debug_format=txt,report will be writtern into a file <netlistName>_<checkName>_<debug_time>.report.log in txt format. When debug_format=sql,report will be writtern in xml/sql format. When debug_format = both, report will be writtern in all formats.Possible values are sql, txt and both.



## Spectre Circuit Simulator Reference

### Circuit Checks

---

debug_report	both	When debug_report = lowz, only lowz nodes are reported and the report will contain all conducting paths from a debug_net to ground causing debug_net to be lowz node. When debug_report = highz, only highz nodes are reported and report will contain all devices that are off causing a debug_net to be highz. When debug_report = both, both lowz and highz node are reported. Possible values are lowz, highz and both.
--------------	------	---

#### Filtering parameters

time_window	[tstart, tstop] sec	Time window to which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend.
error_limit	10000	Maximum number of errors reported. Default is 10000.
fanout	all	Fanout setting to filter node with specified connection. When fanout=gate_has_driver_no_moscap, floating nodes only connecting to MOSCAPs will not be checked. When fanout=gate_no_moscap, gates only connecting to MOSCAPs will not be checked. This option is supported only in XPS. Possible values are all, gate, bulk, gate_has_driver_no_moscap and gate_no_moscap.
filter	no	If set to no, filtering is not performed. If set to rc, dyn_highz checker will report only one node in same parasitic sub. This option is supported only in XPS. Possible values are no and rc.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

<code>filter_diocon_no</code>	<code>0</code>	Filter out highz nodes connected to MOSDIODE. When set to 1, highz node that is connected to PMOSDIODE is filtered out given that its source is connected to VDD. Moreover, a highz node that is connected to NMOSDIODE is filtered out given that its source is connected to 0. When set to 2, highz node that is connected to PMOSDIODE is filtered out given that its source voltage is greater than drain voltage. Moreover, a highz node that is connected to NMOSDIODE is filter out given that its drain voltage is greater than source voltage. This option is supported only in Spectre and APS. Possible values are 0, 1 and 2.
-------------------------------	----------------	---

#### Wildcard scoping

<code>inst</code>	<code>[...]</code>	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
<code>xinst</code>	<code>[...]</code>	Subcircuit instances to be excluded from the check. Default is none.
<code>subckt</code>	<code>[...]</code>	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits ( <code>subckt=*</code> ).
<code>xsubckt</code>	<code>[...]</code>	The instances of the specified subcircuits that are excluded from the check. Default is none.
<code>depth</code>	<code>8</code>	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.
<code>nonconducting_subckt</code>	<code>[...]</code>	All instances of specified subcircuit are considered non-conducting during the floating node detection. All devices inside the subcircuit are considered non-conducting during the floating node detection. Wildcard is supported. Default is none.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

<code>conducting_subckt</code> [...]	All instances of specified subcircuit are considered conducting during the floating node detection. All devices inside the subcircuit are considered conducting during the floating node detection. Wildcard is supported. Default is none.
<code>nonconducting_primitive</code> [...]	All instances of specified primitive or VerilogA modules are considered non-conducting during the floating node detection. Wildcard is supported. Default is none.

### ***Related Topics***

[Dynamic High Impedance Node Check](#)

## Dynamic MOSFET Voltage Check (dyn\_mosv)

### Description

Reports MOSFET devices fulfilling the conditional expression on device voltages and device size (w, l) for a time longer than the user-specified threshold duration.

Supported MOSFET variables are: v(g,s), v(g,d), v(g,b), v(d,s), v(d,b), v(s,b), v(g), v(d), v(s), v(b), l, and w

Supported operators are: +, -, \*, /, ==, !=, <, <=, >, >=, ||, &&, and !

The results are written to the dynamic.xml file, which can be viewed in a Web browser.

### Syntax

Name dyn\_mosv parameter=value ...

### Parameters

Design check parameters

model	[...]	MOSFET device model names to be checked.
cond		The conditional expression to be fulfilled. Default is none.
duration	5.00E-09 sec	Duration threshold. Default is 5.00E-09 sec.
sample	start	Report extreme value or the start point. Possible values are start and extreme.
sort	no	If set to no, sorting is not performed. If set to duration, dyn_mosv will sort violations by duration. Default is no. Possible values are no and duration.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### Filtering parameters

<code>time_window</code>	<code>[tstart, tstop]</code> sec	The time window during which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend.
<code>error_limit</code>	10000	Maximum number of errors reported. Default is 10000.

#### Wildcard scoping

<code>inst</code>	<code>[...]</code>	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
<code>xinst</code>	<code>[...]</code>	Subcircuit instances to be excluded from the check. Default is none.
<code>subckt</code>	<code>[...]</code>	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits ( <code>subckt=*</code> ).
<code>xsubckt</code>	<code>[...]</code>	The instances of the specified subcircuits that are excluded from the check. Default is none.
<code>depth</code>	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

#### ***Related Topics***

#### ***Dynamic MOSFET Voltage Check***

## Dynamic Node Capacitance Check (dyn\_nodecap)

### Description

Reports the node capacitance at specified times (`time`) of a transient simulation. Device capacitances, grounded capacitances, and coupling capacitances are combined into one value.

The results are written to the `dynamic.xml` file, which can be viewed in a Web browser.

### Syntax

Name `dyn_nodecap` parameter=value ...

### Parameters

Design check parameters

<code>node</code>	<code>[...]</code>	Nodes for which the capacitance needs to be checked, vsrc nodes are skipped.
-------------------	--------------------	--

Filtering parameters

<code>time</code>	<code>[...]</code> <code>sec</code>	time point(s) at which the dynamic nodecap check is performed.
<code>error_limit</code>	<code>10000</code>	Maximum number of errors reported. Default is 10000.
<code>fanout</code>	<code>all</code>	Fanout setting to filter node with specified connection. This option is supported only in XPS. Possible values are <code>all</code> , <code>gate</code> and <code>bulk</code> .

Wildcard scoping

<code>inst</code>	<code>[...]</code>	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
-------------------	--------------------	---

## Spectre Circuit Simulator Reference

### Circuit Checks

---

xinst	[...]	Subcircuit instances to be excluded from the check. Default is none.
subckt	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[...]	The instances of the specified subcircuits that are excluded from the check. Default is none.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.
intrinsic_cap_merge	no	Merge the internal captab node to external node. Default is no. This option is supported only in Spectre and APS. Possible values are no and yes.

### ***Related Topics***

#### ***Dynamic Node Capacitance Check***

## Dynamic Noisy Node Check (dyn\_noisynode)

### Description

Identifies the nodes with unstable or noisy node conditions. A node is considered unstable or noisy if its voltage fulfills the condition  $\text{abs}(dV/dt) > e1$  and  $\text{abs}(d(dV/dt)dt) > e2$  for a time longer than duration. Stable periods shorter than skip are ignored.

The results are written to the dynamic.xml file, which can be viewed in a Web browser.

### Syntax

Name dyn\_noisynode parameter=value ...

### Parameters

Design check parameters

node	[...]	Nodes to which the check is applied. Default is none.
duration	5.00E-07 sec	Duration threshold. Default is 5.00E-07 sec.
e1	5.00E04 volt/sec	The first derivative threshold. Default is 5.00E04.
e2	2.00E16 volt/ (sec <sup>2</sup> )	The second derivative threshold. Default is 2.00E16.
skip	50.0E-09 sec	The stable period less than skip is ignored. Default is 50.0E-09.



## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### Filtering parameters

time_window	[tstart, tstop] sec	The time window during which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend.
error_limit	10000	Maximum number of errors reported. Default is 10000.
fanout	all	Fanout setting to filter node with specified connection. This option is supported only in XPS. Possible values are all, gate and bulk.

#### Wildcard scoping

inst	[...]	Subcircuit instances to which the check is applied. Default includes all instances (inst=*).
xinst	[...]	Subcircuit instances to be excluded from the check. Default is none.
subckt	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[...]	The instances of the specified subcircuits that are excluded from the check. Default is none.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

#### ***Related Topics***

#### ***Dynamic Noisy Node Check***

## Dynamic Oscillation Check (dyn\_osc)

### Description

The dynamic oscillation check detects the unexpected circuit oscillations in transient analysis. The results are written to the dynamic.xml file, which can be viewed in a Web browser.

### Syntax

Name dyn\_osc parameter=value ...

### Parameters

Design check parameters

reltol	0.2	Relative tolerance of the check. Default is 0.2.
dvmin	1e-5 V	Minimum difference between peak and valley voltages. Differences less than dvmin are not checked. Default is 10*abstol.
dvmax	0.5 V	Maximum difference between peak and valley voltages. Differences larger than dvmax are not checked. Default is 0.5.
nth	500	The threshold of oscillation count. Default is 500.
method	0	Select the check method, 1 is to check non-monotonic oscillations. Default is 0.

Filtering parameters

time_window	[tstart, tstop] sec	Time intervals to which the check is applied. Default is the whole transient time.
-------------	---------------------------	--

## Spectre Circuit Simulator Reference

### Circuit Checks

---

<code>error_limit</code>	<code>10000</code>	Maximum number of errors reported. Default is 10000.
<code>fanout</code>	<code>all</code>	Fanout setting to filter node with specified connection. Possible values are all, gate and bulk.

#### Wildcard scoping

<code>node</code>	<code>[...]</code>	Nodes to which the check is applied. Default is none.
<code>inst</code>	<code>[...]</code>	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
<code>xinst</code>	<code>[...]</code>	Subcircuit instances to be excluded from the check. Default is none.
<code>subckt</code>	<code>[...]</code>	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits ( <code>subckt=*</code> ).
<code>xsubckt</code>	<code>[...]</code>	The instances of the specified subcircuits that are excluded from the check. Default is none.
<code>depth</code>	<code>8</code>	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

## Dynamic Power Density Check (dyn\_powerdensity)

### Description

Max-AVG power density is reliability analysis on circuits, it will extract max-pwr using sliding window.

The results are written to the dynamic.xml file, which can be viewed in a Web browser.

### Syntax

Name dyn\_powerdensity parameter=value ...

### Parameters

Design check parameters

Filtering parameters

time_window	[tstart, tstop] sec	The time window during which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend.
window	10e-9 sec	Width of a measuring window for average power calculation. These measure-windows will be inside the time_window. If not specified, there will be only one measure-window and it will be equal to time_window.
shift	5.00E-09 sec	Time shift between two measure-windows. Default is 5.00E-09 sec.
error_limit	10000	Maximum number of errors reported. Default is 10000.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### Wildcard scoping

<code>inst</code>	[...]	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
<code>xinst</code>	[...]	Subcircuit instances to be excluded from the check. Default is none.
<code>subckt</code>	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits ( <code>subckt=*</code> ).
<code>xsubckt</code>	[...]	The instances of the specified subcircuits that are excluded from the check. Default is none.
<code>depth</code>	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

## Dynamic Pulse Width Check (dyn\_pulsewidth)

### Description

Reports nodes with pulse width error.

The pulse widths that fall outside of `pwmin_low` and `pwmax_low`, and `pwmin_high` and `pwmax_high` are reported.

The results are written to the `dynamic.xml` file, which can be viewed in a Web browser.

### Syntax

```
Name dyn_pulsewidth parameter=value ...
```

### Parameters

Design check parameters

<code>node</code>	<code>[...]</code>	Nodes to which the check is applied. Default is none.
<code>pwmin_low</code>	<code>0.0 sec</code>	The minimum value of the pulse width in logic 0 state.
<code>pwmax_low</code>	<code>infinity sec</code>	The maximum value of the pulse width in logic 0 state.
<code>pwmin_high</code>	<code>0.0 sec</code>	The minimum value of the pulse width in logic 1 state.
<code>pwmax_high</code>	<code>infinity sec</code>	The maximum value of the pulse width in logic 1 state.
<code>fanout</code>	<code>all</code>	Fanout setting to filter node with specified connection. This option is supported only in XPS. Possible values are all, gate and bulk.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### Digitize parameters

vlth	0.2 V	Low voltage threshold for signal net.
vhth	0.8 V	High voltage threshold for signal net.

#### Filtering parameters

time_window	[tstart, tstop] sec	Time window during which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend.
error_limit	10000	Maximum number of errors reported. Default is 10000.

#### Wildcard scoping

inst	[...]	Subcircuit instances to which the check is applied. Default includes all instances (inst=*).
xinst	[...]	Subcircuit instances to be excluded from the check. Default is none.
subckt	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[...]	The instances of the specified subcircuits that are excluded from the check. Default is none.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

#### ***Related Topics***

#### ***Dynamic Pulse Width Check***

## Dynamic Resistor Voltage Check (dyn\_resv)

### Description

Reports the resistor elements fulfilling the conditional expression on element voltages for a duration longer than the user-specified duration threshold.

Supported resistor variables are: v(1,2), v(1), and v(2)

Supported operators are: +, -, \*, /, ==, !=, <, <=, >, >=, ||, &&, and !

The results are written to the dynamic.xml file, which can be viewed in a Web browser. This check is supported only by XPS. For Spectre and Spectre APS use assert statement.

### Syntax

```
Name dyn_resv parameter=value ...
```

### Parameters

Design check parameters

cond		The conditional expression to be fulfilled. Default is none.
duration	5.00E-09 sec	Duration threshold. Default is 5.00E-09 sec.

Filtering parameters

time_window	[tstart, tstop] sec	The time window during which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend.
error_limit	10000	Maximum number of errors reported. Default is 10000.



## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### Wildcard scoping

inst	[ . . . ]	Subcircuit instances to which the check is applied. Default includes all instances (inst=*).
xinst	[ . . . ]	Subcircuit instances to be excluded from the check. Default is none.
subckt	[ . . . ]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[ . . . ]	The instances of the specified subcircuits that are excluded from the check. Default is none.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

#### ***Related Topics***

[Dynamic Resistor Voltage Check](#)

## Dynamic Rise Fall Edge Check (dyn\_risefall)

### Description

The dynamic edge check detects the nodes whose voltages do not cross the user-defined thresholds VH and VL.

The results are written to the dynamic.xml file, which can be viewed in a Web browser.

### Syntax

```
Name dyn_risefall parameter=value ...
```

### Parameters

Design check parameters

node	[...]	Nodes to which the check is applied. Default is none.
vrefp	[...]	Power reference node to support multiple power domain.
vrefn	[...]	Ground reference node to support multiple power domain.
vh	0 V	Relative high voltage ratio for rise/fall edge measurements.
vl	0 V	Relative low voltage ratio for rise/fall edge measurements.
type		Check rise or fall edge. Possible values are rise and fall.
vh	0 V	High voltage threshold for rise/fall edge measurements.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

vl	0 V	Low voltage threshold for rise/fall edge measurements.
----	-----	--

#### Filtering parameters

time_window	[tstart, tstop] sec	Time window to which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend.
error_limit	10000	Maximum number of errors reported. Default is 10000.
fanout	all	Fanout setting to filter node with specified connection..Possible values are all, gate and bulk.
filter	optimize d	Filter setting to report all nodes or fewer nodes of interest.Possible values are rc, optimized and none.

#### Wildcard scoping

inst	[...]	Subcircuit instances to which the check is applied. Default includes all instances (inst=*).
xinst	[...]	Subcircuit instances to be excluded from the check. Default is none.
subckt	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[...]	The instances of the specified subcircuits that are excluded from the check. Default is none.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

monotonic	1	Flag to indicate that monotonic transitions are expected. Non-monotonic edge is reported violation. Set 0 to skip non-monotonic violations..
dvmin	0.01 V	Voltage resolution to control the sensitivity of non-monotonic checks..
duration	0 Sec	Delta voltage duration of non-monotonic checks..

## Dynamic Setup and Hold Check (dyn\_setuphold)

### Description

Reports nodes with setup or hold timing errors with reference to a clock signal. The transition time of the clock signal is specified using refTime.

For setup timing, transitions that happen between refTime + delay - setupTime and refTime + delay are reported.

For hold timing, transitions that happen between refTime + delay and refTime + delay + holdTime are reported.

ref\_vhth and vhth parameters are used for triggering rising edge measurements, while ref\_vlth and vlth are used for triggering falling edge measurements.

If the subckt parameter is specified, node and ref\_node are considered as local nodes to the specified subcircuit. In other words, node and ref\_node belong to instances of the specified subcircuit. Only one subckt value can be specified per check, with no wildcard.

If the subckt parameter is not specified, node and ref\_node are considered as global nodes with the hierarchical names starting from the top level.

The parameters inst, xinst, xsubckt and depth are not supported.

The results are written to the dynamic.xml file, which can be viewed in a Web browser.

### Syntax

```
Name dyn_setuphold parameter=value ...
```

### Parameters

Design check parameters

node	[...]	Nodes to which the check is applied. Default is none.
ref_node		Name of the referenced clock. Wildcards are not supported.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

setup_time	0.0 sec	Setup time violation window. If specified, the setup check is enabled. Default is 0.0 sec.
hold_time	0.0 sec	Hold time violation window. If specified, the hold check is enabled.
delay	0.0 sec	Delay of the reference signal.
edge	rise	Edge type of the signal net. Default is rise. Possible values are rise, fall and both.
ref_edge	rise	Edge type of the reference signal. Default is rise. Possible values are rise, fall and both.
fanout	all	Fanout setting to filter node with specified connection. This option is supported only in XPS. Possible values are all, gate and bulk.

#### Digitize parameters

setup_chk_time	0.0 sec	Setup time checking window.
hold_chk_time	0.0 sec	Hold time checking window.
vlth	0.2 V	Low voltage threshold for the signal net.
ref_vlth	0.2 V	Low voltage threshold for the referenced net.
vhth	0.8 V	High voltage threshold for the signal net.
ref_vhth	0.8 V	High voltage threshold for the referenced net.

#### Filtering parameters

time_window	[tstart, tstop] sec	Time window to which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend.
-------------	---------------------------	---

## Spectre Circuit Simulator Reference

### Circuit Checks

---

report	violation	Report all checks or violation only. Default is violation. Possible values are all and violation.
margin_stats	no	Whether to report margin stats. If margin_stats=no, do not report margin stats. If margin_stats=violation, only collect the margin stats of violation=yes. If margin_stats=all, collect all margin stats, including violation=yes and violation=no. Default is no. Possible values are no, violation and all.
error_limit	10000	Maximum number of errors reported. Default is 10000.

#### Wildcard scoping

subckt	none	Instances of the specified subcircuit to which the check is applied. Default is none.
--------	------	---

#### ***Related Topics***

[Dynamic Setup and Hold Check](#)

## Dynamic Subckt Port Voltage/Current Check (dyn\_subcktport)

### Description

Reports instances, of the user-specified `subckt`, fulfilling the conditional expression on port-voltages and port-currents for a time longer than the user-specified `duration`. Only one `subckt` is allowed per statement.

The voltages and currents of a port can be reference by `subckt`'s port-name. For example, consider a `subckt` definition `".subckt INV port_A port_B"`. Here, supported `subckt` port-name are: `v(port_A)`, `v(port_B)`, `v(port_A,port_B)`, `i(port_A)`, `i(port_B)`. The port-current is positive when current is going into a `subckt`.

Supported operators are: `+`, `-`, `*`, `/`, `==`, `!=`, `<`, `<=`, `>`, `>=`, `||`, `&&`, and `!`

All design instances of subcircuit specified in `subckt` will be checked.

The results are written to the `dynamic.xml` file, which can be viewed in a Web browser.

### Syntax

```
Name dyn_subcktport parameter=value ...
```

### Parameters

Design check parameters

<code>subckt</code>	Instances of the specified subcircuit to which the check is applied. Wildcard is not supported. Only one subcircuit is allowed per statement. Default is none.
<code>cond</code>	The conditional expression to be fulfilled. Default is none.



**Spectre Circuit Simulator Reference**  
Circuit Checks

---

Filtering parameters

time_window	[tstart, tstop] sec	Time window to which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend.
duration	5.00E-09 sec	Duration threshold. Default is 5.00E-09 sec.
error_limit	10000	Maximum number of errors reported. Default is 10000.

***Related Topics***

*Dynamic Subcircuit Port/Voltage/Current Check*

## Dynamic Subckt Port Power Check (dyn\_subcktpwr)

### Description

Reports port currents, port powers, and subcircuit powers.

The port current is positive when current is going into a subcircuit. This check will report average, RMS, and the maximum values of the current entering a port.

The power analysis can be done by using the parameter `power`. When parameter `power` is set to on, then two additional sections are generated. The first section reports the average, RMS and maximum of power entering at all ports, which is given in the parameter `port`. The second section reports the average, RMS and maximum of power consumed by each instance of a subcircuit that is defined by the `inst` parameter. Note that for the second section it is mandatory to have the parameter `port` set to `[*]`.

Note that Max is defined as the maximum of the absolute of a waveform within a `time_window`. For example, consider a current entering a port having a peak value of 1mA and another peak value of -2mA below 0. Therefore, the check will report "Max (A)" as -2mA and the time point will be reported in "Max Time(s)". Therefore, this check will use an absolute function to find the maximum current/power but it will report that finding with a regular current/power. Unlike Max, the average and RMS are defined as the average or RMS value of a regular waveform within a `time_window`, respectively.

The filter parameter (`filter`) can be used to filter out ports that are connected only to the gate of MOSFET.

The results are written to the `dynamic.xml` file, which can be viewed in a Web browser.

### Syntax

```
Name dyn_subcktpwr parameter=value ...
```

### Parameters

Design check parameters

<code>port</code>	<code>[...]</code>	Ports to be checked. Default is none.
<code>net</code>		Net. Default is none.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

power	off	report power or not. Possible values are off and on.
-------	-----	--

#### Filtering parameters

filter	none	check all ports or only those not connecting to gates. Possible values are none and gates.
--------	------	--

abs_avg	0	Calculate normal average value (0) or absolute average value (1). Default is 0. Possible values are 0 and 1.
---------	---	--

top	0	Report total current of top-level voltage source (1) or not (0). This parameter is supported only with Spectre FX. The default value is 0. Possible values are 0 and 1.
-----	---	---

ith	0.0	Port currents with either abs(AVG), RMS or abs(MAX) value larger than the specified <code>ith</code> are reported. In other words, if all abs(AVG), RMS and abs(MAX) values are below <code>ith</code> then it will be filtered out. Default is 0.0.
-----	-----	--

time_window	[tstart, tstop] sec	The time window during which the circuit check is applied. Multiple non-overlapping time windows are supported. Default time window is 0 to tend.
-------------	------------------------	---

error_limit	10000	Maximum number of errors reported. Default is 10000.
-------------	-------	--

#### Wildcard scoping

inst	[...]	Subcircuit instances to which the check is applied. Default is none.
------	-------	--

depth		Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is none.
-------	--	---

## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### ***Related Topics***

*Dynamic Subcircuit Port Power Check*

## Dynamic Uninitialized latch (dyn\_uninilatch)

### Description

This checks detect the uninitialized latch nodes, which may have more than one correct dc solution.

All nodes in a potential loop will be reported.

The results are reported into a file with the extension dynamic.xml, which can be read with a web browser.

This check is supported only by XPS.

### Syntax

Name `dyn_uninilatch parameter=value ...`

### Parameters

Design check parameters

<code>node</code>	<code>[...]</code>	Nodes to which the check is applied. Default is none.
-------------------	--------------------	---

Filtering parameters

<code>error_limit</code>	<code>10000</code>	Maximum number of errors reported. Default is 10000.
--------------------------	--------------------	--

Wildcard scoping

<code>inst</code>	<code>[...]</code>	Subcircuit instances to which the check is applied. Default includes all instances (inst=*).
<code>xinst</code>	<code>[...]</code>	Subcircuit instances to be excluded from the check. Default is none.

**Spectre Circuit Simulator Reference**  
Circuit Checks

---

subckt	[ . . . ]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[ . . . ]	The instances of the specified subcircuits that are excluded from the check. Default is none.

## Static Capacitor Check (static\_capacitor)

### Description

Reports all capacitors within or outside the range of `cmin` and `cmax`. It can also generate a distribution list of all the capacitors in a circuit.

If the type parameter is set to `range`, all capacitors outside the range of `cmin` and `cmax` will be reported. If the type parameter is set to `print`, all capacitors between `cmin` and `cmax` will be reported. In the report, the capacitor names can be sorted by clicking on the Device name header in a Web browser.

If the type parameter is set to `distr`, a distribution list will be generated for all capacitors. There are a maximum of 9 bins:

-Inf - 0, 0 - 10a, 10a - 100a, 100a - 1f, 1f - 10f, 10f - 100f, 100f - 1p, 1p - 10p, and 10p - Inf

However, if two or more consecutive bins are empty, they will merge into one bin, reducing the number of bins. If type is set to `distr`, the parameters `cmin`, `cmax`, and `error_limit` are ignored.

The results are written to the `static.xml` file, which can be viewed in a Web browser. This check is supported only by XPS and APS.

### Syntax

```
Name static_capacitor parameter=value ...
```

### Parameters

Design check parameters

<code>type</code>	<code>print</code>	Checking types. Possible values are <code>range</code> , <code>distr</code> and <code>print</code> .
<code>cmin</code>	<code>-1 F</code>	Minimum capacitor value.
<code>cmax</code>	<code>1 F</code>	Maximum capacitor value.

**Spectre Circuit Simulator Reference**  
Circuit Checks

---

Filtering parameters

error_limit	10000	Maximum number of errors reported. Default is 10000.
-------------	-------	--

***Related Topics***

*Static Capacitor Check*



## Static Capacitor Voltage Check (static\_capv)

### Description

Reports capacitor devices fulfilling the conditional expression on device voltages.

Supported capacitor variables are: v(1,2), v(1), and v(2)

Supported operators are: +, -, \*, /, ==, !=, <, <=, >, >=, ||, &&, and !

The results are written to the static.xml file, which can be viewed in a Web browser. This check is supported only by XPS and APS.

### Syntax

Name static\_capv parameter=value ...

### Parameters

Design check parameters

cond		The conditional expression to be fulfilled. Default is none.
rpt_node	no	report node voltages. Possible values are no, all, top and selected.

Digitize parameters

vpth	-0.4 V	PMOS threshold voltage. This value is used to calculate the voltage drop across a PMOS channel during voltage propagation. Default is -0.4 V.
vnth	0.5 V	NMOS threshold voltage. This value is used to calculate the voltage drop across a NMOS channel during voltage propagation. Default is 0.5 V.
pwl_time	infinity	Time for pwl src to be considered as constant vsrc.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

<code>vsrncodefile</code>	<code>[...]</code>	Defines file from which node voltages are read for performing the check.
---------------------------	--------------------	--

#### Filtering parameters

<code>error_limit</code>	<code>10000</code>	Maximum number of errors reported. Default is 10000.
--------------------------	--------------------	--

#### Wildcard scoping

<code>inst</code>	<code>[...]</code>	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
-------------------	--------------------	---

<code>xinst</code>	<code>[...]</code>	Subcircuit instances to be excluded from the check. Default is none.
--------------------	--------------------	--

<code>subckt</code>	<code>[...]</code>	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits ( <code>subckt=*</code> ).
---------------------	--------------------	--

<code>xsubckt</code>	<code>[...]</code>	The instances of the specified subcircuits that are excluded from the check. Default is none.
----------------------	--------------------	---

<code>depth</code>	<code>8</code>	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.
--------------------	----------------	--

#### ***Related Topics***

***Static Capacitor Voltage Check***

## Static Coupling Impact Check (static\_coupling)

### Description

Evaluate the possible coupling effects in the circuit.

For each victim node, find out its voltage level impact from aggressors by divide total aggressor's charge with its total capacitance (including cc+gc+device cap). Each aggressor's charge is calculated by multiply coupling aggressor's voltage level (using Vmax for worst case analysis) by coupling capacitance. In this check, only capacitors written in netlist will be counted.

The results are reported into a file with the extension static.xml, which can be read with a web browser.

### Syntax

```
Name static_coupling parameter=value ...
```

### Parameters

Design check parameters

node	[...]	Nodes to which the check is applied. Default is none.
------	-------	---

Filtering parameters

error_limit	100	Maximum number of errors reported. Default is 100.
-------------	-----	--

Digitize parameters

vpth	-0.4 V	PMOS threshold voltage. This value is used to calculate the voltage drop across a PMOS channel during voltage propagation. Default is -0.4 V.
------	--------	---

## Spectre Circuit Simulator Reference

### Circuit Checks

---

<code>vnth</code>	<code>0.5 V</code>	NMOS threshold voltage. This value is used to calculate the voltage drop across a NMOS channel during voltage propagation. Default is 0.5 V.
<code>pwl_time</code>	<code>infinity</code>	Time for pwl src to be considered as constant vsrc.
<code>vsrctnodefile</code>	<code>[...]</code>	Defines file from which node voltages are read for performing the check.

#### Wildcard scoping

<code>inst</code>	<code>[...]</code>	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
<code>xinst</code>	<code>[...]</code>	Subcircuit instances to be excluded from the check. Default is none.
<code>subckt</code>	<code>[...]</code>	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits ( <code>subckt=*</code> ).
<code>xsubckt</code>	<code>[...]</code>	The instances of the specified subcircuits that are excluded from the check. Default is none.
<code>depth</code>	<code>8</code>	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

## Static DC Leakage Path Check (static\_dcpath)

### Description

Reports the always conducting paths between the power supply nodes.

The results are written to the static.xml file, which can be viewed in a Web browser. This check is supported only by XPS and APS.

### Syntax

Name `static_dcpath` parameter=value ...

### Parameters

Design check parameters

<code>net</code>	[...]	The leakage path between the voltage source nodes is checked. <code>net</code> defines a set of nodes and not a pair. For example, <code>node = [vdd vdd1 0]</code> checks the leakage path between <code>vdd</code> and <code>vdd1</code> , <code>vdd</code> and <code>0</code> , and <code>vdd1</code> and <code>0</code> .
<code>rpt_node</code>	<code>no</code>	report node voltages. Possible values are <code>no</code> , <code>all</code> , <code>top</code> and <code>selected</code> .

Digitize parameters

<code>vpth</code>	<code>-0.4 V</code>	PMOS threshold voltage. This value is used to calculate the voltage drop across a PMOS channel during voltage propagation. Default is <code>-0.4 V</code> .
<code>vnth</code>	<code>0.5 V</code>	NMOS threshold voltage. This value is used to calculate the voltage drop across a NMOS channel during voltage propagation. Default is <code>0.5 V</code> .
<code>vsrncodefile</code>	[...]	Defines file from which node voltages are read for performing the check.

**Spectre Circuit Simulator Reference**  
Circuit Checks

---

pwl_time	infinity	Time for pwl src to be considered as constant vsrc.
----------	----------	---

Filtering parameters

error_limit	10000	Maximum number of errors reported. Default is 10000.
-------------	-------	--

***Related Topics***

*Static DC Leakage Path Check*

## Static Diode Voltage Check (static\_diodev)

### Description

Reports the diode devices fulfilling the conditional expression on device voltages.

Supported diode variables are: v(a,c), v(a), and v(c)

Supported operators are: +, -, \*, /, ==, !=, <, <=, >, >=, ||, &&, and !

The results are written to the static.xml file, which can be viewed in a Web browser. This check is supported only by XPS and APS.

### Syntax

Name static\_diodev parameter=value ...

### Parameters

Design check parameters

model	[...]	MOSFET device model names to be checked.
cond		The conditional expression to be fulfilled. Default is none.
rpt_node	no	report node voltages. Possible values are no, all, top and selected.

Digitize parameters

vpth	-0.4 V	PMOS threshold voltage. This value is used to calculate the voltage drop across a PMOS channel during voltage propagation. Default is -0.4 V.
vnth	0.5 V	NMOS threshold voltage. This value is used to calculate the voltage drop across a NMOS channel during voltage propagation. Default is 0.5 V.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

pwl_time	infinity	Time for pwl src to be considered as constant vsrc.
vsrcnodefile	[...]	Defines file from which node voltages are read for performing the check.

#### Filtering parameters

error_limit	10000	Maximum number of errors reported. Default is 10000.
-------------	-------	--

#### Wildcard scoping

inst	[...]	Subcircuit instances to which the check is applied. Default includes all instances (inst=*).
xinst	[...]	Subcircuit instances to be excluded from the check. Default is none.
subckt	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[...]	The instances of the specified subcircuits that are excluded from the check. Default is none.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

#### ***Related Topics***

#### ***Static Diode Voltage Check***



## Static ERC Check (static\_erc)

### Description

Performs various electrical rule checks and reports devices with violations.

The results are reported into a file with the extension static.xml, which can be read with a web browser.

This check is only supported by XPS and APS.

### Syntax

Name static\_erc parameter=value ...

### Parameters

Design check parameters

hotwell	off	Reports MOSFET with bulk not connected to VDD or GND.Possible values are off and on.
floatbulk	off	Reports MOSFET with floating bulk.Possible values are off, all and no_top.
floatgate	off	Reports MOSFET with floating gate.Possible values are off, all, no_top, no_moscap, no_top_moscap and poded.
dangle	off	Reports dangling node.Possible values are off, all and no_top.
gate2power	off	Reports PMOS with gate connected to ground and NMOS with gate connected to VDD.Possible values are off and on.
gate2ground	off	Reports NMOS with gate connected to ground and PMOS with gate connected to VDD.Possible values are off and on.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

<code>rpt_node</code>	<code>no</code>	report node voltages. Possible values are no, all, top and selected.
-----------------------	-----------------	--

#### Digitize parameters

<code>vlth</code>	<code>0.2 V</code>	Voltage below <code>vlth</code> is considered GND. Default is 0.2V. Applicable only with <code>gate2power</code> , <code>gate2ground</code> and <code>hotwell</code> .
-------------------	--------------------	--

<code>vhth</code>	<code>0.8 V</code>	Voltage above <code>vhth</code> is considered VDD. Default is 0.8V. Applicable only with <code>gate2power</code> , <code>gate2ground</code> and <code>hotwell</code> .
-------------------	--------------------	--

<code>vpth</code>	<code>-0.4 V</code>	PMOS threshold voltage. This value is used to calculate the voltage drop across a PMOS channel during voltage propagation. Default is -0.4 V.
-------------------	---------------------	---

<code>vnth</code>	<code>0.5 V</code>	NMOS threshold voltage. This value is used to calculate the voltage drop across a NMOS channel during voltage propagation. Default is 0.5 V.
-------------------	--------------------	--

<code>pwl_time</code>	<code>infinity</code>	Time for <code>pwl src</code> to be considered as constant <code>vsrc</code> .
-----------------------	-----------------------	--

<code>vsrccodefile</code>	<code>[...]</code>	Defines file from which node voltages are read for performing the check.
---------------------------	--------------------	--

#### Filtering parameters

<code>error_limit</code>	<code>10000</code>	Maximum number of errors reported. Default is 10000.
--------------------------	--------------------	--

#### Wildcard scoping

<code>inst</code>	<code>[...]</code>	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
-------------------	--------------------	---

## Spectre Circuit Simulator Reference

### Circuit Checks

---

xinst	[...]	Subcircuit instances to be excluded from the check. Default is none.
subckt	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[...]	The instances of the specified subcircuits that are excluded from the check. Default is none.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

#### ***Related Topics***

#### ***Static ERC Check***

## Static Highfanout Check (static\_highfanout)

### Description

This checks detect the MOSFETs having gate connected to a highfanout node, and report nodes having count larger than specified count.

The results are reported into a file with the extension static.xml, which can be read with a web browser.

### Syntax

Name static\_highfanout parameter=value ...

### Parameters

Design check parameters

node	[...]	Nodes to which the check is applied. Default is none.
upth	10	upth is to define the ratio of the width of PMOS (pull-up) and the loading of MOSFETs (with gate connection). The load of transmission gate are also considered. Default is 10.
downth	20	downth is to define the ratio of the width of NMOS (pull-down) and the loading of MOSFETs. Default is 20.
rcut	1e6	The resistor bigger than its value is cut. The default value is 1e6.

Filtering parameters

error_limit	10000	Maximum number of errors reported. Default is 10000.
-------------	-------	--

## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### Wildcard scoping

<code>inst</code>	[...]	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
<code>xinst</code>	[...]	Subcircuit instances to be excluded from the check. Default is none.
<code>subckt</code>	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits ( <code>subckt=*</code> ).
<code>xsubckt</code>	[...]	The instances of the specified subcircuits that are excluded from the check. Default is none.
<code>depth</code>	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

## Static HighZ Node Check (static\_highz)

### Description

Reports the nodes that do not have any possible conducting path to a DC power supply or ground.

The results are written to the static.xml file, which can be viewed in a Web browser. This check is supported only by XPS and APS.

### Syntax

```
Name static_highz parameter=value ...
```

### Parameters

Design check parameters

node	[...]	Nodes to which the check is applied. Default is none.
fanout	all	Fanout setting to filter node with specified connection. When fanout=gate_has_driver_no_moscap, floating nodes only connecting to MOSCAPs will not be checked. When fanout=gate_no_moscap, gates only connecting to MOSCAPs will not be checked. Possible values are all, gate, bulk, gate_has_driver_no_moscap and gate_no_moscap.
rpt_node	no	report node voltages. Possible values are no, all, top and selected.

Digitize parameters

vpth	-0.4 V	PMOS threshold voltage. This value is used to calculate the voltage drop across a PMOS channel during voltage propagation. Default is -0.4 V.
------	--------	---

## Spectre Circuit Simulator Reference

### Circuit Checks

---

<code>vnth</code>	<code>0.5 V</code>	NMOS threshold voltage. This value is used to calculate the voltage drop across a NMOS channel during voltage propagation. Default is 0.5 V.
<code>vsrncodefile</code>	<code>[...]</code>	Defines file from which node voltages are read for performing the check.
<code>pwl_time</code>	<code>infinity</code>	Time for pwl src to be considered as constant vsrc.

#### Filtering parameters

<code>error_limit</code>	<code>10000</code>	Maximum number of errors reported. Default is 10000.
--------------------------	--------------------	--

#### Wildcard scoping

<code>inst</code>	<code>[...]</code>	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
<code>xinst</code>	<code>[...]</code>	Subcircuit instances to be excluded from the check. Default is none.
<code>subckt</code>	<code>[...]</code>	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits ( <code>subckt=*</code> ).
<code>xsubckt</code>	<code>[...]</code>	The instances of the specified subcircuits that are excluded from the check. Default is none.
<code>depth</code>	<code>8</code>	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

#### ***Related Topics***

##### ***Static High Impedance Node Check***

## Static MOSFET Voltage Check (static\_mosv)

### Description

Reports MOSFET devices fulfilling the conditional expression on device voltages and device size (w, l).

Supported MOSFET variables: v(g,s), v(g,d), v(g,b), v(d,s), v(d,b), v(s,b), v(g), v(d), v(s), v(b), l, w Supported operators: +, -, \*, /, ==, !=, <, <=, >, >=, ||, &&, and !

The results are written to the static.xml file, which can be viewed in a Web browser. This check is supported only by XPS and APS.

### Syntax

```
Name static_mosv parameter=value ...
```

### Parameters

#### Design check parameters

model	[...]	MOSFET device model names to be checked.
cond		The conditional expression to be fulfilled. Default is none.
rpt_node	no	report node voltages. Possible values are no, all, top and selected.

#### Digitize parameters

vpth	-0.4 V	PMOS threshold voltage. This value is used to calculate the voltage drop across a PMOS channel during voltage propagation. Default is -0.4 V.
vnth	0.5 V	NMOS threshold voltage. This value is used to calculate the voltage drop across a NMOS channel during voltage propagation. Default is 0.5 V.



## Spectre Circuit Simulator Reference

### Circuit Checks

---

pwl_time	infinity	Time for pwl src to be considered as constant vsrc.
vsrccnodefile	[...]	Defines file from which node voltages are read for performing the check.

#### Filtering parameters

error_limit	10000	Maximum number of errors reported. Default is 10000.
-------------	-------	--

#### Wildcard scoping

inst	[...]	Subcircuit instances to which the check is applied. Default includes all instances (inst=*).
xinst	[...]	Subcircuit instances to be excluded from the check. Default is none.
subckt	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[...]	The instances of the specified subcircuits that are excluded from the check. Default is none.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

#### ***Related Topics***

#### ***Static MOSFET Voltage Check***

## Static NMOS to vdd count (static\_nmos2vdd)

### Description

This checks counts unpaired NMOS in a charging path from a fanout node to VDD and report paths having NMOS count larger than specified count.

Only fanout nodes are analyzed. Nodes only connecting to MOSDIODEs or MOSCAPs will not be checked.

The results are reported into a file with the extension static.xml, which can be read with a web browser.

### Syntax

```
Name static_nmos2vdd parameter=value ...
```

### Parameters

Design check parameters

node	[...]	Nodes to which the check is applied. Default is none.
------	-------	---

Filtering parameters

count	3	Maximum number of permitted NMOS. Default is 3.
error_limit	10000	Maximum number of errors reported. Default is 10000.

Wildcard scoping

inst	[...]	Subcircuit instances to which the check is applied. Default includes all instances (inst=*).
------	-------	--

## Spectre Circuit Simulator Reference

### Circuit Checks

---

xinst	[...]	Subcircuit instances to be excluded from the check. Default is none.
subckt	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[...]	The instances of the specified subcircuits that are excluded from the check. Default is none.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.
vpth	-0.4 V	PMOS threshold voltage. This value is used to calculate the voltage drop across a PMOS channel during voltage propagation. Default is -0.4 V.
vnth	0.5 V	NMOS threshold voltage. This value is used to calculate the voltage drop across a NMOS channel during voltage propagation. Default is 0.5 V.
vsrccnodefile	[...]	Defines file from which node voltages are read for performing the check.
pwl_time	infinity	Time for pwl src to be considered as constant vsrc.

## Static NMOS Forward Bias Bulk Check (static\_nmosb)

### Description

Reports the NMOS devices with a forward-biased bulk condition. A violation is generated when the bulk bias voltage meets the following conditions:

- when `mode = definite`:  

$$\min(V_b) \geq \min(V_d, V_s) + \text{abs}(v_t)$$
- when `mode = possible`:  

$$\max(V_b) \geq \min(V_d, V_s) + \text{abs}(v_t)$$

The results are written to the `static.xml` file, which can be viewed in a Web browser. This check is supported only by XPS and APS.

### Syntax

Name `static_nmosb` parameter=value ...

### Parameters

Design check parameters

<code>model</code>	[...]	MOSFET device model names to be checked.
<code>rpt_node</code>	<code>no</code>	report node voltages. Possible values are <code>no</code> , <code>all</code> , <code>top</code> and <code>selected</code> .
<code>mode</code>	<code>definite</code>	Mode <code>possible</code> will report all possible violations. Mode <code>definite</code> will report definite violations. Default is <code>definite</code> . Possible values are <code>definite</code> and <code>possible</code> .
<code>vt</code>	<code>0.3 V</code>	Threshold for p-n junction being checked. Default is <code>0.3V</code> .

## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### Digitize parameters

vpth	-0.4 V	PMOS threshold voltage. This value is used to calculate the voltage drop across a PMOS channel during voltage propagation. Default is -0.4 V.
vnth	0.5 V	NMOS threshold voltage. This value is used to calculate the voltage drop across a NMOS channel during voltage propagation. Default is 0.5 V.
pwl_time	infinity	Time for pwl src to be considered as constant vsrc.
vsrccodefile	[...]	Defines file from which node voltages are read for performing the check.

#### Filtering parameters

error_limit	10000	Maximum number of errors reported. Default is 10000.
-------------	-------	--

#### Wildcard scoping

inst	[...]	Subcircuit instances to which the check is applied. Default includes all instances (inst=*).
xinst	[...]	Subcircuit instances to be excluded from the check. Default is none.
subckt	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[...]	The instances of the specified subcircuits that are excluded from the check. Default is none.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### ***Related Topics***

*Static NMOS/PMOS Forward Bias Bulk Check*

## Static Always Conducting NMOSFET Check (static\_nmosvgs)

### Description

Reports NMOS devices potentially always conducting due to connectivity problems.

The following conditions are checked, and an error is reported if they are fulfilled: NMOS:  
 $\min(V_g) > \min(V_s/V_d) + \text{abs}(v_t)$

The results are written to the static.xml file, which can be viewed in a Web browser. This check is supported only by XPS and APS.

### Syntax

Name static\_nmosvgs parameter=value ...

### Parameters

Design check parameters

model	[...]	MOSFET device model names to be checked.
vt	0.0 V	MOSFET voltage threshold. Default is 0.0 V.
rpt_node	no	report node voltages. Possible values are no, all, top and selected.

Digitize parameters

vpth	-0.4 V	PMOS threshold voltage. This value is used to calculate the voltage drop across a PMOS channel during voltage propagation. Default is -0.4 V.
vnth	0.5 V	NMOS threshold voltage. This value is used to calculate the voltage drop across a NMOS channel during voltage propagation. Default is 0.5 V.
pwl_time	infinity	Time for pwl src to be considered as constant vsrc.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

<code>vsrncodefile</code>	<code>[...]</code>	Defines file from which node voltages are read for performing the check.
---------------------------	--------------------	--

#### Filtering parameters

<code>error_limit</code>	<code>10000</code>	Maximum number of errors reported. Default is 10000.
--------------------------	--------------------	--

#### Wildcard scoping

<code>inst</code>	<code>[...]</code>	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
-------------------	--------------------	---

<code>xinst</code>	<code>[...]</code>	Subcircuit instances to be excluded from the check. Default is none.
--------------------	--------------------	--

<code>subckt</code>	<code>[...]</code>	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits ( <code>subckt=*</code> ).
---------------------	--------------------	--

<code>xsubckt</code>	<code>[...]</code>	The instances of the specified subcircuits that are excluded from the check. Default is none.
----------------------	--------------------	---

<code>depth</code>	<code>8</code>	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.
--------------------	----------------	--

#### ***Related Topics***

***Static Always Conducting NMOS/PMOS Check***



## Static PMOS to gnd count (static\_pmos2gnd)

### Description

This checks counts unpaired PMOS in a discharging path from a fanout node to GND and report paths having PMOS count larger than specified count.

Only fanout nodes are analyzed. Nodes only connecting to MOSDIODEs or MOSCAPs will not be checked.

The results are reported into a file with the extension static.xml, which can be read with a web browser.

### Syntax

```
Name static_pmos2gnd parameter=value ...
```

### Parameters

Design check parameters

node	[...]	Nodes to which the check is applied. Default is none.
------	-------	---

Filtering parameters

count	3	Maximum number of permitted PMOS. Default is 3.
error_limit	10000	Maximum number of errors reported. Default is 10000.

Wildcard scoping

inst	[...]	Subcircuit instances to which the check is applied. Default includes all instances (inst=*).
------	-------	--

## Spectre Circuit Simulator Reference

### Circuit Checks

---

xinst	[...]	Subcircuit instances to be excluded from the check. Default is none.
subckt	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[...]	The instances of the specified subcircuits that are excluded from the check. Default is none.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.
vpth	-0.4 V	PMOS threshold voltage. This value is used to calculate the voltage drop across a PMOS channel during voltage propagation. Default is -0.4 V.
vnth	0.5 V	NMOS threshold voltage. This value is used to calculate the voltage drop across a NMOS channel during voltage propagation. Default is 0.5 V.
vsrccnodefile	[...]	Defines file from which node voltages are read for performing the check.
pwl_time	infinity	Time for pwl src to be considered as constant vsrc.

## Static PMOS Forward Bias Bulk Check (static\_pmosb)

### Description

Reports the PMOS devices with a forward-biased bulk condition.

A violation is generated when the bulk bias voltage meets the following conditions:

- when `mode = definite`:  
 $\max(V_b) \leq \max(V_d, V_s) - \text{abs}(v_t)$
- when `mode = possible`:  
 $\min(V_b) \leq \max(V_d, V_s) - \text{abs}(v_t)$

The results are written to the `static.xml` file, which can be viewed in a Web browser. This check is supported only by XPS and APS.

### Syntax

```
Name static_pmosb parameter=value ...
```

### Parameters

Design check parameters

<code>model</code>	<code>[...]</code>	MOSFET device model names to be checked.
<code>rpt_node</code>	<code>no</code>	report node voltages. Possible values are <code>no</code> , <code>all</code> , <code>top</code> and <code>selected</code> .
<code>mode</code>	<code>definite</code>	Mode <code>possible</code> will report all possible violations. Mode <code>definite</code> will report definite violations. Default is <code>definite</code> . Possible values are <code>definite</code> and <code>possible</code> .
<code>vt</code>	<code>0.3 V</code>	Threshold for p-n junction being checked. Default is 0.3V.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### Digitize parameters

<code>vpth</code>	<code>-0.4 V</code>	PMOS threshold voltage. This value is used to calculate the voltage drop across a PMOS channel during voltage propagation. Default is -0.4 V.
<code>vnth</code>	<code>0.5 V</code>	NMOS threshold voltage. This value is used to calculate the voltage drop across a NMOS channel during voltage propagation. Default is 0.5 V.
<code>pwl_time</code>	<code>infinity</code>	Time for pwl src to be considered as constant vsrc.
<code>vsrccodefile</code>	<code>[...]</code>	Defines file from which node voltages are read for performing the check.

#### Filtering parameters

<code>error_limit</code>	<code>10000</code>	Maximum number of errors reported. Default is 10000.
--------------------------	--------------------	--

#### Wildcard scoping

<code>inst</code>	<code>[...]</code>	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
<code>xinst</code>	<code>[...]</code>	Subcircuit instances to be excluded from the check. Default is none.
<code>subckt</code>	<code>[...]</code>	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits ( <code>subckt=*</code> ).
<code>xsubckt</code>	<code>[...]</code>	The instances of the specified subcircuits that are excluded from the check. Default is none.
<code>depth</code>	<code>8</code>	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### ***Related Topics***

*Static NMOS/PMOS Forward Bias Bulk Check*

## Static Always Conducting PMOSFET Check (static\_pmosvgs)

### Description

Reports PMOS devices potentially always conducting due to connectivity problems.

The following conditions are checked, and an error is reported if they are fulfilled: PMOS:  
 $\max(V_g) < \max(V_s/V_d) - \text{abs}(v_t)$

The results are written to the static.xml file, which can be viewed in a Web browser. This check is supported only by XPS and APS.

### Syntax

Name static\_pmosvgs parameter=value ...

### Parameters

Design check parameters

model	[...]	MOSFET device model names to be checked.
vt	0.0 V	MOSFET voltage threshold. Default is 0.0 V.
rpt_node	no	report node voltages. Possible values are no, all, top and selected.

Digitize parameters

vpth	-0.4 V	PMOS threshold voltage. This value is used to calculate the voltage drop across a PMOS channel during voltage propagation. Default is -0.4 V.
vnth	0.5 V	NMOS threshold voltage. This value is used to calculate the voltage drop across a NMOS channel during voltage propagation. Default is 0.5 V.
pwl_time	infinity	Time for pwl src to be considered as constant vsrc.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

<code>vsrncodefile</code>	<code>[...]</code>	Defines file from which node voltages are read for performing the check.
---------------------------	--------------------	--

#### Filtering parameters

<code>error_limit</code>	10000	Maximum number of errors reported. Default is 10000.
--------------------------	-------	--

#### Wildcard scoping

<code>inst</code>	<code>[...]</code>	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
-------------------	--------------------	---

<code>xinst</code>	<code>[...]</code>	Subcircuit instances to be excluded from the check. Default is none.
--------------------	--------------------	--

<code>subckt</code>	<code>[...]</code>	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits ( <code>subckt=*</code> ).
---------------------	--------------------	--

<code>xsubckt</code>	<code>[...]</code>	The instances of the specified subcircuits that are excluded from the check. Default is none.
----------------------	--------------------	---

<code>depth</code>	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.
--------------------	---	--

#### ***Related Topics***

***Static Always Conducting NMOS/PMOS Check***

## Static RCDelay Check (static\_rcdelay)

### Description

Reports nodes with excessive rise, or fall times. Rise and fall times are estimated.

Only fanout nodes are analyzed. Nodes connecting to MOSDIODEs or MOSCAPs will not be checked.

Either the top worst case rise/fall times (maxnrise, maxnfall), or nodes with rise/fall times above user defined thresholds (maxtrise, maxtfall) can be reported.

For each node the driving MOSFETs and the receiving MOSFET (gate connected to the analyzed node) will be reported.

The results are reported into a file with the extension static.xml, which can be read with a web browser.

### Syntax

Name static\_rcdelay parameter=value ...

### Parameters

Design check parameters

node	[...]	Nodes to which the check is applied. Default is none.
detailed_path	no	Report all possible (yes) or worst case (no) rise/fall time per node. Possible values are yes and no.
fanoutmargin	[0.1 0.9]	Specifies the relative fanout level (in ratio of VDD voltage) for which rise and fall time is measured. the range of values is [0.01 0.99]. Low margin should less than high margin.



## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### Filtering parameters

<code>cmin</code>	<code>1e-14</code>	Specifies the node capacitance threshold. Only nodes with total capacitance higher than the specified value will be reported. Default is 1e-14.
<code>maxnrise</code>	<code>0</code>	Report the top number nodes with highest rise-time. Default is none.
<code>minnrise</code>	<code>0</code>	Report the bottom number nodes with lowest rise-time. Default is none.
<code>maxnfall</code>	<code>0</code>	Report the top number nodes with highest fall-time. Default is none.
<code>minnfall</code>	<code>0</code>	Report the bottom number nodes with lowest fall-time. Default is none.
<code>maxtrise</code>	<code>infinity</code>	Report only those node names with rise time higher than the specified value. Default is infinity.
<code>mintrise</code>	<code>0</code>	Report only those node names with rise time lower than the specified value. Default is none.
<code>maxtfall</code>	<code>infinity</code>	Report only those node names with fall time higher than the specified value. Default is infinity.
<code>mintfall</code>	<code>0</code>	Report only those node names with fall time lower than the specified value. Default is none.
<code>maxvio_pernode</code>	<code>0</code>	Maximum number of violations per reported node.
<code>inverse</code>	<code>no</code>	If set to no, reports the nodes with rise/fall time higher than the specified maxtrise/maxtfall, and the nodes with rise/fall time lower than the specified mintrise/mintfall. If set to yes, reports the nodes with rise/fall time within [mintrise,maxtrise]/[mintfall,maxtfall]. Default is no. Possible values are no and yes.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### Wildcard scoping

<code>inst</code>	[ ... ]	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
<code>xinst</code>	[ ... ]	Subcircuit instances to be excluded from the check. Default is none.
<code>subckt</code>	[ ... ]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits ( <code>subckt=*</code> ).
<code>xsubckt</code>	[ ... ]	The instances of the specified subcircuits that are excluded from the check. Default is none.
<code>depth</code>	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

#### ***Related Topics***

[Static RC Delay Check](#)

## Static Resistor Check (static\_resistor)

### Description

Reports all resistors within or outside the range of `rmin` and `rmax`. It can also generate a distribution list of all the resistors in a circuit.

If the type parameter is set to `range`, all the resistors outside the range of `rmin` and `rmax` will be reported. If the type parameter is set to `print`, all the resistors between `rmin` and `rmax` will be reported. In the report, the resistor names can be sorted by clicking on the Device name header in the Web browser.

If the type parameter is set to `distr`, a distribution list will be generated for all resistors. There are a maximum of 12 bins:

-Inf - 0, 0 - 1m, 1m - 10m, 10m - 0.1, 0.1 - 1, 1 - 10, 10 - 100, 100 - 1k, 1k - 10k, 10k - 100k, 100k - 1Meg, and 1Meg - Inf

However, if two or more consecutive bins are empty, they will merge into one bin, reducing the number of bins. If type is set to `distr`, the parameters `rmin`, `rmax` and `error_limit` are ignored.

The results are reported into the `static.xml` file, which can be viewed in a Web browser. This check is supported only by XPS and APS.

### Syntax

```
Name static_resistor parameter=value ...
```

### Parameters

Design check parameters

<code>type</code>	<code>print</code>	Checking types. Possible values are <code>range</code> , <code>distr</code> and <code>print</code> .
<code>rmin</code>	<code>-1000E9</code> $\Omega$	Minimum resistor value.
<code>rmax</code>	<code>1000E9</code> $\Omega$	Maximum resistor value.

**Spectre Circuit Simulator Reference**  
Circuit Checks

---

Filtering parameters

<code>error_limit</code>	10000	Maximum number of errors reported. Default is 10000.
--------------------------	-------	--

***Related Topics***

*Static Resistor Check*

## Static Resistor Voltage Check (static\_resv)

### Description

Reports the resistor devices fulfilling the conditional expression on device voltages.

Supported resistor variables are: v(1,2), v(1), and v(2)

Supported operators are: +, -, \*, /, ==, !=, <, <=, >, >=, ||, &&, and !

The results are written to the static.xml file, which can be viewed in a Web browser. This check is supported only by XPS and APS.

### Syntax

Name static\_resv parameter=value ...

### Parameters

Design check parameters

cond		The conditional expression to be fulfilled. Default is none.
rpt_node	no	report node voltages. Possible values are no, all, top and selected.

Digitize parameters

vpth	-0.4 V	PMOS threshold voltage. This value is used to calculate the voltage drop across a PMOS channel during voltage propagation. Default is -0.4 V.
vnth	0.5 V	NMOS threshold voltage. This value is used to calculate the voltage drop across a NMOS channel during voltage propagation. Default is 0.5 V.
pwl_time	infinity	Time for pwl src to be considered as constant vsrc.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

<code>vsrncodefile</code>	<code>[...]</code>	Defines file from which node voltages are read for performing the check.
---------------------------	--------------------	--

#### Filtering parameters

<code>error_limit</code>	10000	Maximum number of errors reported. Default is 10000.
--------------------------	-------	--

#### Wildcard scoping

<code>inst</code>	<code>[...]</code>	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
-------------------	--------------------	---

<code>xinst</code>	<code>[...]</code>	Subcircuit instances to be excluded from the check. Default is none.
--------------------	--------------------	--

<code>subckt</code>	<code>[...]</code>	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits ( <code>subckt=*</code> ).
---------------------	--------------------	--

<code>xsubckt</code>	<code>[...]</code>	The instances of the specified subcircuits that are excluded from the check. Default is none.
----------------------	--------------------	---

<code>depth</code>	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.
--------------------	---	--

#### ***Related Topics***

#### ***Static Resistor Voltage Check***

## Static Stack Check (static\_stack)

### Description

This checks count PMOS and NMOS transistor stacks. Then reports stacks larger than specified count.

The results are written to the static.xml file, which can be viewed in a Web browser.

### Syntax

```
Name static_stack parameter=value ...
```

### Parameters

Design check parameters

model	[...]	MOSFET device model names to be checked.
-------	-------	--

Filtering parameters

count	3	Maximum number of permitted count. Default is 3.
-------	---	--

error_limit	10000	Maximum number of errors reported. Default is 10000.
-------------	-------	--

Wildcard scoping

inst	[...]	Subcircuit instances to which the check is applied. Default includes all instances (inst=*).
------	-------	--

xinst	[...]	Subcircuit instances to be excluded from the check. Default is none.
-------	-------	--

subckt	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
--------	-------	---

**Spectre Circuit Simulator Reference**  
Circuit Checks

---

xsubckt	[ . . . ]	The instances of the specified subcircuits that are excluded from the check. Default is none.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.



## Static Subckt Port Voltage Check (static\_subcktport)

### Description

Reports instances of the user-specified `subckt` fulfilling the conditional expression on port voltages. Only one `subckt` is allowed per statement.

Voltages of a port can be referenced by the port name of a subcircuit. For example, consider the subcircuit definition "subckt INV port\_A port\_B". Here, supported port names are: `v(port_A)`, `v(port_B)` and `v(port_A,port_B)`.

Supported operators are: `+`, `-`, `*`, `/`, `==`, `!=`, `<`, `<=`, `>`, `>=`, `||`, `&&`, and `!`.

All design instances of subcircuit specified in `subckt` will be checked.

The results are written to the `dynamic.xml` file, which can be viewed in a Web browser.

### Syntax

```
Name static_subcktport parameter=value ...
```

### Parameters

Design check parameters

<code>subckt</code>		Instances of the specified subcircuit to which the check is applied. Wildcard is not supported. Only one <code>subckt</code> is allowed per statement. Default is none.
<code>cond</code>		The conditional expression to be fulfilled. Default is none.
<code>vlth</code>	0.2 V	DC sources with voltage lower than <code>vlth</code> carry static 0. Default is 0.2 V.
<code>vhth</code>	0.8 V	DC sources with voltage higher than <code>vhth</code> carry static 1. Default is 0.8 V.

## Spectre Circuit Simulator Reference

### Circuit Checks

---

vpth	-0.4 V	PMOS threshold voltage. This value is used to calculate the voltage drop across a PMOS channel during voltage propagation. Default is -0.4 V.
vnth	0.5 V	NMOS threshold voltage. This value is used to calculate the voltage drop across a NMOS channel during voltage propagation. Default is 0.5 V.
pwl_time	infinity	Time for pwl src to be considered as constant vsrc.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.
vsrccodefile	[...]	Defines file from which node voltages are read for performing the check.

#### Filtering parameters

error_limit	10000	Maximum number of errors reported. Default is 10000.
-------------	-------	--

#### ***Related Topics***

[Static Subcircuit Port Voltage Check](#)

## Static Transmission Gate Check (static\_tgate)

### Description

Reports the transmission gates which cause potential leakage currents between power supplies.

These gates can be characterized by their node connectivity, based on the following:

- nodes which connect to the gate and NMOS drain/source terminals, but not to the PMOS drain/source terminals
- nodes which connect to the gate and PMOS drain/source terminals, but not to the NMOS drain/source terminals

The results are written to the static.xml file, which can be viewed in a Web browser. This check is supported only by XPS and APS.

### Syntax

```
Name static_tgate parameter=value ...
```

### Parameters

Design check parameters

node	[...]	Nodes to which the check is applied. Default is none.
------	-------	---

Filtering parameters

error_limit	10000	Maximum number of errors reported. Default is 10000.
-------------	-------	--

## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### Wildcard scoping

<code>inst</code>	[ ... ]	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
<code>xinst</code>	[ ... ]	Subcircuit instances to be excluded from the check. Default is none.
<code>subckt</code>	[ ... ]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits ( <code>subckt=*</code> ).
<code>xsubckt</code>	[ ... ]	The instances of the specified subcircuits that are excluded from the check. Default is none.
<code>depth</code>	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

#### ***Related Topics***

#### ***Static Transmission Gate Check***

## Static Topology Check (static\_topology)

### Description

Performs various topology checks and reports violations.

The results are reported into a file with the extension static.xml, which can be read with a web browser.

### Syntax

Name static\_topology parameter=value ...

### Parameters

Design check parameters

node	[...]	Nodes to which the check is applied. Default is none.
node_file		node_file points to a file in which report levels and node names are specified for checking
pin2gnd	off	Do top level subckt instance pin path check to ground node.Possible values are off and on.
fanout	all	Fanout setting to filter node with specified connection..Possible values are all and toppin.

Filtering parameters

error_limit	10000	Maximum number of errors reported. Default is 10000.
-------------	-------	--

## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### Wildcard scoping

<code>inst</code>	[...]	Subcircuit instances to which the check is applied. Default includes all instances ( <code>inst=*</code> ).
<code>xinst</code>	[...]	Subcircuit instances to be excluded from the check. Default is none.
<code>subckt</code>	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits ( <code>subckt=*</code> ).
<code>xsubckt</code>	[...]	The instances of the specified subcircuits that are excluded from the check. Default is none.

## Static Voltage Domain Conflict Check (static\_vconflict)

### Description

Reports low voltage domain driving MOSFET devices from high voltage domain.

Fanout violations - Reports PMOS and depletion NMOS of higher voltage domain driven by MOSFET devices of low voltage domain;

MultiDC violations - Reports Multi-DC sources and their connecting path elements;

The results are written to the static.xml file, which can be viewed in a Web browser.

This check is supported only by XPS and APS.

### Syntax

```
Name static_vconflict parameter=value ...
```

### Parameters

Design check parameters

rpt_node	no	report node voltages. Possible values are no, all, top and selected.
----------	----	--

Digitize parameters

vpth	-0.4 V	PMOS threshold voltage. This value is used to calculate the voltage drop across a PMOS channel during voltage propagation. Default is -0.4 V.
------	--------	---

vnth	0.5 V	NMOS threshold voltage. This value is used to calculate the voltage drop across a NMOS channel during voltage propagation. Default is 0.5 V.
------	-------	--

pwl_time	infinity	time for pwl src to be considered as constant vsrc.
----------	----------	---

**Spectre Circuit Simulator Reference**  
Circuit Checks

---

<code>vsrcnodefile</code>	<code>[...]</code>	Defines file from which node voltages are read for performing the check.
---------------------------	--------------------	--

Filtering parameters

<code>error_limit</code>	<code>10000</code>	Maximum number of errors reported. Default is 10000.
--------------------------	--------------------	--



## Static Voltage Domain Device Check (static\_voltdomain)

### Description

Reports High Voltage driving Low Voltage MOSFET and Low Voltage driving High Voltage MOSFET.

The results are written to the static.xml file, which can be viewed in a Web browser. This check is supported only by XPS and APS.

### Syntax

Name static\_voltdomain parameter=value ...

### Parameters

#### Design check parameters

model	[...]	MOSFET device model names to be checked. By default all kinds of MOSFET's will be checked.
rpt_node	no	report node voltages. Possible values are no, all, top and selected.

#### Digitize parameters

pwl_time	infinity	Time for pwl src to be considered as constant vsrc.
vsrccnodefile	[...]	Defines file from which node voltages are read for performing the check.

#### Filtering parameters

error_limit	10000	Maximum number of errors reported. Default is 10000.
-------------	-------	--

## Spectre Circuit Simulator Reference

### Circuit Checks

---

#### Wildcard scoping

inst	[...]	Subcircuit instances to which the check is applied. Default includes all instances (inst=*).
xinst	[...]	Subcircuit instances to be excluded from the check. Default is none.
subckt	[...]	The instances of the specified subcircuit to which the check is applied. Default includes all subcircuits (subckt=*).
xsubckt	[...]	The instances of the specified subcircuits that are excluded from the check. Default is none.
depth	8	Hierarchy levels (starting from top, instance, or subcircuit scope) to be checked. Default is 8.

#### ***Related Topics***

[Static Voltage Domain Device Check](#)

---

## References

---

This section gives additional details about the source documents referred to in the text.

- [antognetti88] Paolo Antognetti, Giuseppe Massobrio. *Semiconductor Device Modeling with SPICE*. McGraw-Hill, New York, 1988.
- [gear71] C. William Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice-Hall, 1971.
- [hammerstad80] E. Hammerstad, O. Jensen. "Accurate models for microstrip computer-aided design." *IEEE MTT-S 1980 International Microwave Symposium Digest*, pages 407-409.
- [jansen83] Rolf H. Jansen, Martin Kirschning. "Arguments and an accurate model for the power-current formulation of microstrip characteristic impedance." *Arch. Elek. Übertragung (AEU)*, vol. 37, 1983, pages 108-112.
- [kirschning82] M. Kirschning, R. H. Jansen. "Accurate model for effective dielectric constant of microstrip with validity up to millimetre-wave frequencies." *Electronic Letters*, vol. 18, no. 6, 18 March 1982, pages 272-273.
- [kundert90] Kenneth S. Kundert, Jacob K. White, Alberto Sangiovanni-Vincentelli. *Steady-State Methods for Simulating Analog and Microwave Circuits*. Kluwer Academic Publishers, 1990.
- [nagel75] Laurence W. Nagel. *SPICE2: A Computer Program to Simulate Semiconductor Circuits*. Ph. D. dissertation, University of California at Berkeley, May 1975. Available through Electronics Research Laboratory Publications, U. C. B., 94720; Memorandum No. UCB/ERL M520.
- [quarles89] Thomas L. Quarles. *Analysis of Performance and Convergence Issues for Circuit Simulation*. Ph. D. dissertation, University of California at Berkeley, April 1989. Extensively documents the Spice3 program. Available through Electronics Research Laboratory Publications, U. C. B., 94720; Memorandum No. UCB/ERL M89/42.

## Spectre Circuit Simulator Reference

### References

---

- [statz87]Hermann Statz, Paul Newman, Irl W. Smith, Robert A. Pucel, Hermann A. Haus. "GaAs FET device and circuit simulation in SPICE." *IEEE Transactions on Electron Devices*, vol. ED-34, no. 2, pages 160-169, February 1987.
- [vladimirescu81]A. Vladimirescu, Kaihe Zhang, A. R. Newton, D. O. Pederson, A. Sangiovanni-Vincentelli. *SPICE Version 2G User's Guide*, August 1981. Available through Industrial Liaison Program Software Distribution office, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, 94720.
- [yang82]Ping Yang, Pallab K. Chatterjee. "SPICE modeling for small geometry MOSFET circuits." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. CAD-1, no. 4, pages 169-182, October 1982.